

Rexroth IndraMotion MLC 13VRS Robot Control V2

R911341035
Edition 03

Reference Book



Title	Rexroth IndraMotion MLC 13VRS Robot Control V2														
Type of Documentation	Reference Book														
Document Typecode	DOK-MLC***-ROCO****V13-RE03-EN-P														
Internal File Reference	RS-5caa0b6da67931800a6846a501563330-3-en-US-4														
Purpose of Documentation	This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This reference book also includes the description of the "ML_Robot" library. Robot control V2 is supported by a kinematics interface. The interface components are described in the "ML_KinTech" library.														
Record of Revision	<table><tr><th>Edition</th><th>Release Date</th><th>Notes</th></tr><tr><td>Edition 01</td><td>08.2013</td><td>First edition Robot Control V2 for IndraMotion MLC 13VRS</td></tr><tr><td>Edition 02</td><td>10.2013</td><td>Robot Control V2 with KinInterface</td></tr><tr><td>Edition 03</td><td>11.2013</td><td>MC_KIN_REF_EXT, Trafo 111</td></tr></table>			Edition	Release Date	Notes	Edition 01	08.2013	First edition Robot Control V2 for IndraMotion MLC 13VRS	Edition 02	10.2013	Robot Control V2 with KinInterface	Edition 03	11.2013	MC_KIN_REF_EXT, Trafo 111
Edition	Release Date	Notes													
Edition 01	08.2013	First edition Robot Control V2 for IndraMotion MLC 13VRS													
Edition 02	10.2013	Robot Control V2 with KinInterface													
Edition 03	11.2013	MC_KIN_REF_EXT, Trafo 111													
Copyright	© Bosch Rexroth AG 2013 This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.														
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.														
Published by	Bosch Rexroth AG Bgm.-Dr.-Nebel-Str. 2 ■ 97816 Lohr a. Main, Germany Phone +49 9352 18 0 ■ Fax +49 9352 18 8400 http://www.boschrexroth.com/ System IndraLogic & Basis Motion Logic GBU (vha/PiGe)														
Note	This document has been printed on chlorine-free bleached paper.														

Table of Contents

	Page
1 About this Documentation.....	11
1.1 Validity of the Documentation.....	11
1.2 Documentation Structure.....	11
1.3 Using the Safety Instructions.....	12
1.3.1 Safety Instructions – Structure.....	12
1.3.2 Explaining Signal Words and Safety Alert Symbol.....	12
1.3.3 Symbols Used.....	12
1.4 Required and Supplementing Documentations.....	13
1.5 Names and Abbreviations.....	18
 2 Functional Description.....	 19
2.1 System Overview.....	19
2.2 Coordinate Systems and Transformations.....	19
2.2.1 Coordinate Systems.....	19
Coordinate Systems, Overview.....	19
Types of Coordinate Systems.....	19
Axis Coordinate System (ACS).....	21
Base Coordinate System (BCS).....	21
Machine Coordinate System (MCS).....	22
Product Coordinate System (PCS).....	22
2.2.2 Transformations.....	23
Transformations, Overview.....	23
ACS/BCS Transformation.....	23
BCS/MCS Transformation.....	24
MCS/PCS Transformation.....	25
2.3 State Diagram of a Kinematics.....	26
2.4 Robot Control, Commands and Functions.....	27
2.4.1 Robot Control, Commands and Functions, Overview.....	27
2.4.2 Command State.....	28
2.4.3 List of all Commands and Functions.....	29
2.5 Motion Types.....	31
2.5.1 Motion Types, Overview.....	31
2.5.2 Linear Motion Type.....	32
2.5.3 Circular Motion Type.....	33
2.5.4 Point to Point Motion Type.....	34
2.5.5 Absolute Positioning.....	34
2.5.6 Relative Positioning.....	34
2.6 Manual Mode.....	34
2.6.1 Manual Mode, General Information.....	34
2.6.2 Single-step Operation.....	36
2.6.3 Continuous Operation.....	36
2.7 Teach-In.....	36
2.7.1 Teach-In, General Information.....	36
2.7.2 Structure of the Point File.....	37

Table of Contents

	Page
2.7.3 Save Point.....	37
2.7.4 Load Point.....	38
2.7.5 Load Next Point.....	38
2.7.6 Delete Points.....	38
2.7.7 Save Point File.....	38
2.7.8 Load Point File.....	39
2.7.9 Delete Point File.....	39
2.8 Blending.....	39
2.8.1 Blending, General Information.....	39
2.8.2 Blending Radius.....	40
2.8.3 Slope Types.....	40
2.8.4 Block_Slope.....	41
2.8.5 Cont_Slope.....	43
2.8.6 Blending Possibilities with Various Motion Types.....	46
2.9 Stopping, Interrupting, Continuing Motions.....	46
2.9.1 Stopping Motions.....	46
2.9.2 Interrupting Motions.....	48
2.9.3 Continuing Motions.....	48
2.9.4 Waiting.....	49
2.10 Belt-synchronous Motion.....	49
2.10.1 Belt-synchronous Motion, General Information.....	49
2.10.2 Belt Configuration.....	50
2.10.3 Belt Values and Points on the Belt.....	51
2.10.4 Belt Operation.....	52
2.10.5 Blending and Belt Synchronization.....	53
2.10.6 Summary.....	54
2.11 Switching on Path.....	54
2.12 Traversing Range Limits.....	57
2.13 Errors and Error Reactions.....	59
2.13.1 Errors and Error Reactions, General Information.....	59
2.13.2 Error Reaction of the Commanded Motion.....	60
2.13.3 Belt-specific Error Reaction.....	60
2.14 Cyclical Command Value Specification.....	61
2.14.1 Cyclical Command Value Specification, General Information.....	61
2.14.2 Start and Operation of the Cyclical Command Value Specification.....	61
2.14.3 Cyclical Command Value Specification.....	62
2.14.4 Termination of Cyclic Command Value Specification.....	62
2.15 Performance Measurement.....	63
3 Interface Functions.....	65
3.1 Menu Structure of the Kinematics.....	65
3.1.1 Menu Item: Robot.....	65
3.1.2 Adding a New Kinematics.....	65
3.1.3 Menu item: Kinematics.....	67
3.2 Kinematics - Context Menu.....	67
3.2.1 Kinematics - Context Menu and/or Kinematics Menu Item, General Information.....	67

Table of Contents

	Page
3.2.2 Setup.....	68
3.2.3 Diagnostics.....	68
3.3 Kinematics – Dialogs.....	69
3.3.1 Overview on dialogs.....	69
3.3.2 Scaling/Units.....	70
3.3.3 Axis Lengths and Coupling.....	70
3.3.4 Motion Limit Values.....	70
3.3.5 Coordinate Offset.....	71
4 Transformation Schemes.....	73
4.1 Transformation Schemes, Overview.....	73
4.2 Transformation Schemes.....	74
4.2.1 Cartesian.....	74
4.2.2 2-Axis Delta with 1 Optional Orientation Axis.....	75
4.2.3 3-Axis Scara with 1 Optional Orientation Axis.....	77
4.2.4 4-Axis Scara with 1 Optional Orientation Axis.....	79
4.2.5 2-Axis Parallelogram.....	81
4.2.6 Inverted 2-Axis Parallelogram.....	82
4.2.7 Belt Kinematics.....	83
4.2.8 2-Axis Parallelogram V1.....	84
4.2.9 3-Axis Delta with 1 Optional Orientation Axis.....	85
4.2.10 2-Axis Parallelogram V2.....	87
4.2.11 3-Axis Twisting Crane.....	88
4.2.12 2-Axis Flex Stamper.....	89
4.2.13 3-Axis Delta with Linear Axes and 1 Optional Orientation Axis.....	91
4.2.14 2-Axis Delta Slide.....	94
4.2.15 3-Axis Parallelogram V3.....	95
4.2.16 4-Axis Speedpicker.....	96
4.3 User-defined Transformation.....	98
4.4 Extended User-defined Transformation.....	100
4.5 Mechanical Information.....	103
5 ML_Robot.library.....	107
5.1 Overview.....	107
5.2 Global Variables.....	108
5.2.1 Overview.....	108
5.2.2 Global Constants.....	109
5.2.3 KinData[] - Kinematic Data in the Shared Memory	109
5.2.4 DV_Kin[] - Kinematic Data via Function Variable Access	110
5.3 Function Blocks.....	112
5.3.1 Function Blocks, Overview.....	112
5.3.2 Standard Interfaces at Function Blocks.....	115
Motivation.....	115
Function Block Types.....	115
Function Block Types and their Interfaces.....	116

Table of Contents

	Page
Inputs and Outputs of State-controlled Function Blocks, Terminating Processing.....	117
Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing.....	120
Inputs and Outputs of State-controlled Function Blocks, Permanent Processing.....	123
Inputs and Outputs of Edge-controlled Function Blocks, Permanent Processing.....	126
5.3.3 Config Function Blocks.....	128
MC_GroupReset.....	128
MC_SetCartesianTransform.....	129
MC_SetKinTransform.....	131
MC_SetKinTransformExt.....	132
MC_UngroupAllAxes.....	133
ML_AddMechanicData.....	135
ML_GroupAllAxes.....	135
ML_GroupSetSafeZone.....	137
ML_KinSetDistanceBlendOption	138
ML_SetBeltDesyncRelConfiguration.....	139
ML_SetBeltConfiguration.....	140
ML_SetBeltErrorReaction.....	142
ML_SetBeltParameter.....	143
ML_SetCylindricTransform.....	145
ML_SetRotaryTableConfiguration.....	147
5.3.4 Info Function Blocks and Functions.....	148
MC_GroupReadActualPosition.....	148
ML_KinGetDistanceBlendOption.....	149
ML_RobotGetBeltConfiguration.....	150
ML_RobotGetBeltDesyncRelConfiguration.....	151
ML_RobotGetBeltErrorReaction.....	152
ML_RobotGetMechanic.....	153
ML_RobotGetPerformance.....	155
ML_RobotGetTrafoValues.....	156
ML_RobotMovePoint.....	157
ML_RobotReadCyclicPos.....	158
ML_RobotSetPerformanceEnable.....	159
ML_RobotTransformPoint.....	160
ML_SwitchOnPath.....	162
5.3.5 Movement Function Blocks.....	163
MC_GroupContinue.....	163
MC_GroupInterrupt.....	164
MC_GroupStop.....	165
MC_MoveCircularAbsolute.....	166
MC_MoveCircularRelative.....	169
MC_MoveDirectAbsolute.....	171
MC_MoveDirectRelative.....	173
MC_MoveLinearAbsolute.....	175
MC_MoveLinearRelative.....	177
ML_BeltDeSynchronisationWithLimits.....	179
ML_BeltSynchronisationWithLimits.....	180

Table of Contents

	Page
ML_GroupJogCont.....	181
ML_GroupJogStep.....	183
ML_GroupWait.....	186
ML_RobotOpenCyclicPos.....	187
ML_RobotWriteCyclicPos.....	188
5.3.6 Parameter Function Blocks.....	189
Accessing Kinematic Data.....	189
MB_GroupReadListParameter.....	190
MB_GroupReadParameter.....	191
MB_GroupReadRealParameter.....	192
MB_GroupReadSercosAttribute.....	193
MB_GroupReadSercosDataStatus.....	194
MB_GroupReadStringParameter.....	195
MB_GroupWriteListParameter.....	196
MB_GroupWriteParameter.....	198
MB_GroupWriteRealParameter.....	199
MB_GroupWriteStringParameter.....	199
5.3.7 RCL Program Function Blocks.....	201
MB_IRDProgSelect.....	201
MB_IRDProgStart.....	202
MB_IRDProgStop.....	202
ML_RoCoDir.....	203
ML_RoCoGetPoint.....	206
ML_RoCoGetPointList.....	209
ML_RoCoSetPoint.....	211
5.3.8 RCL User Interface Function Blocks.....	214
RCL User Interface, Overview.....	214
ML_KinGetUserBit.....	215
ML_KinGetUserBit_DWORD.....	215
ML_KinGetUserDInt.....	216
ML_KinGetUserReal.....	217
ML_KinSetUserBit.....	218
ML_KinSetUserDInt.....	219
ML_KinSetUserPoint.....	220
ML_KinSetUserReal.....	220
5.3.9 TeachIn Function Blocks.....	221
ML_GroupDeletePointFile.....	221
ML_GroupDeletePoints.....	222
ML_GroupLoadPointFile.....	223
ML_GroupReadNextPoint.....	224
ML_GroupReadPoint.....	225
ML_GroupSavePointFile.....	226
ML_GroupWritePoint.....	228
5.4 Data Types.....	229
5.4.1 Data Types, Overview.....	229
5.4.2 Config Data Types.....	230

Table of Contents

	Page
ML_BELT_ERROR_REACTION_MODE.....	230
ML_BELT_KIND.....	230
ML_KIN_AXES_CONFIG_DATA.....	230
ML_KIN_AXIS_CONFIG.....	231
ML_KIN_SAFE_ZONE_TYPE.....	231
ML_KIN_USED_AXES.....	231
MC_KIN_REF.....	232
MC_KIN_REF, Overview.....	232
MC_KIN_REF, Initialization.....	232
MC_KIN_REF, Forward.....	233
MC_KIN_REF, Inverse.....	233
Error Handling with Interface Functions.....	233
MC_KIN_REF_EXT.....	234
MC_KIN_REF_EXT, Overview.....	234
MC_KIN_REF_EXT, GetUsedAxes.....	234
MC_KIN_REF_EXT, DetermineMemory.....	234
MC_KIN_REF_EXT, Initialization.....	235
MC_KIN_REF_EXT, Forward.....	235
MC_KIN_REF_EXT, Inverse.....	235
MC_KIN_REF_EXT, GetMechanic.....	236
Error Handling with Interface Functions.....	236
5.4.3 Info Data Types.....	237
E_KIN_MECH_CMD.....	237
ML_KINEMATICS_DATA_SM.....	237
ML_KIN_CS_TYPE.....	239
ML_KIN_MECHANIC_DATA.....	240
ML_KIN_PERF_DATA.....	240
ML_POINTKIND.....	240
5.4.4 Jog Data Types.....	241
ML_COORD_BITS.....	241
ML_KIN_E_COORD_SYS.....	241
ML_KIN_E_JOG_VEL.....	241
ML_KIN_JOG.....	242
5.4.5 TeachIn Data Types.....	242
E_KIN_TEACH_IN_POINT_FILE_OPTIONS.....	242
5.4.6 Movement Data Types.....	243
ML_ARR_COORDS.....	243
ML_ARR_PLACEMENT.....	244
ML_CIRCMODES.....	244
ML_COORDSYS.....	244
ML_CYCLIC_MODE.....	245
ML_SET_MODE.....	246
ML_SLOPE_TYPE.....	246
5.4.7 RCL Data Types.....	246
ML_POINT_TYPE.....	246
ML_ROCO_FILE_DIR.....	247

Table of Contents

	Page
ML_ROCO_NAME.....	247
ML_ROCO_POINT	248
ML_ROCO_POINT_NAME.....	248
6 Kinematics Interface.....	251
6.1 Kinematics Interface, Introduction.....	251
6.2 ML_KinTech.library.....	254
6.2.1 ML_KinTech.library, Introduction and Overview.....	254
6.2.2 Kinematics Interface Structures.....	255
Kinematics Interface Structures, Overview.....	255
MB_KINEMATICS_CONTROL_TYPE01.....	256
MB_KINEMATICS_CONTROL_TYPE01, Overview.....	256
MB_KINEMATICS_CONTROL_TYPE01.....	256
MB_KINEMATICS_ADMINISTRATION.....	257
MB_KINEMATICS_MODE_UNION.....	260
MB_KINEMATICS_CONVEYOR_BELT.....	261
MB_KINEMATICS_CONVEYOR_BELT_V1.....	263
MB_KINEMATICS_RCL_POINT.....	264
MB_KINEMATICS_RCL_PROG.....	266
MB_KINEMATICS_SETUP.....	268
MB_KINEMATICS_COORDINATED.....	270
MB_KINEMATICS_RCL.....	272
MB_KINEMATICS_STATUS_TYPE01.....	272
MB_KINEMATICS_STATUS_TYPE01, Overview.....	272
MB_KINEMATICS_STATUS_TYPE01.....	273
MB_KINEMATICS_ADMIN_STATUS.....	273
MB_KINEMATICS_RCL_STATUS.....	276
MB_KINEMATICS_DIAGNOSIS.....	278
MB_KINEMATICS_SETUP_STATUS.....	280
6.2.3 Kinematics Interface Function Blocks.....	280
ML_KinematicsInitType01.....	280
ML_KinematicsInterfaceType01.....	283
ML_KinJogV1.....	285
ML_KinJog.....	287
6.2.4 Operation Modes.....	290
Overview.....	290
Kinematics Ready, Drives Ready.....	291
Kinematics Ready, Drives Halt.....	291
Kinematics Homing.....	292
Absolute PTP Positioning.....	293
Relative PTP Positioning.....	293
Absolute Linear Positioning.....	294
Relative Linear Positioning.....	295
Absolute Circular Positioning.....	296
Relative Circular Positioning.....	297
Kinematics Stopping.....	298

Table of Contents

	Page
"Interrupted" Operation Mode.....	299
"Continue" Operation Mode.....	299
"External Function Block" Operation Mode.....	300
"RCL" Operation Mode.....	300
6.2.5 Global Variables.....	301
6.2.6 First Steps, Kinematics Interface.....	302
Overview.....	302
Program for the Kinematics Interface.....	302
Kinematics Interface Visualizations.....	302
Overview.....	302
System Overview Visualization.....	303
Kinematics Overview Visualization.....	305
Referencing Overview (Homing) Visualization.....	306
Positioning Overview (Position) Visualization.....	307
Robot Control Language (RCL) Visualization.....	309
Belt Configuration (ConveyorBelt) Visualization.....	309
Jog/Setup Mode Visualization.....	310
Robot Control Language Point (RCLPoint) Visualization.....	311
Robot Control Language Program (RCLProg) Visualization.....	312
6.2.7 Kinematics Interface User Extension.....	313
Overview.....	313
Extending the Kinematics Interface Structure.....	314
Extending the Function Block TE_KinematicsInterface.....	314
7 Application Examples.....	315
7.1 Application Examples, Overview.....	315
7.2 Application example 1: Using Coordinate Systems.....	315
7.3 Application example 2: Belt Synchronization.....	316
7.4 Application Example 3: User-Defined Transformation.....	317
7.5 Application Example 4: Extended User-Defined Transformation.....	321
7.6 Sequential Programming with the Kinematics Interface.....	326
7.7 Registering a User-defined Transformation.....	328
8 Parameter.....	329
8.1 Parameter, General Information.....	329
8.2 Axis-Relevant Parameters.....	329
8.2.1 Axis-Relevant Parameters, General Information.....	329
8.2.2 A-0-0028, Travel range limit switch.....	329
8.2.3 A-0-0030, Position limit value, positive.....	329
8.2.4 A-0-0031, Position limit value, negative.....	330
8.2.5 A-0-0032, Velocity limit value, positive.....	330
8.2.6 A-0-0033, Velocity limit value, negative.....	330
8.2.7 A-0-0034, Acceleration limit value, bipolar.....	330
8.2.8 A-0-0036, Jerk limit value, bipolar.....	331
8.2.9 A-0-0038, Torque/force limit value, bipolar.....	331

Table of Contents

	Page
8.2.10 A-0-0224, Rapid stop deceleration.....	331
8.2.11 A-0-0225, Rapid stop jerk.....	332
8.2.12 A-0-0240, Feedrate override.....	332
8.2.13 A-0-0241, Feedrate override control word.....	332
8.3 Control-Related Parameters.....	332
8.3.1 Control-Related Parameters, General Information.....	332
8.3.2 C-0-0123, Maximum number of kinematics.....	332
8.3.3 C-0-2480, Configured kinematic number.....	332
8.3.4 C-0-2482, List of non-linked axes.....	333
8.3.5 C-0-2483, List of configured kinematics.....	333
8.3.6 C-0-2484, Kinematic configuration list.....	333
8.4 Kinematic-Related Parameters.....	334
8.4.1 Kinematic-Related Parameters, General Information.....	334
8.4.2 K-0-0001, Logic kinematic number.....	334
8.4.3 K-0-0002, Kinematic name.....	334
8.4.4 K-0-0003, Maximum path velocity.....	334
8.4.5 K-0-0005, Software limit switch, positive, MCS.....	335
8.4.6 K-0-0006, Software limit switch, negative, MCS.....	335
8.4.7 K-0-0007, Axis number of kinematics.....	335
8.4.8 K-0-0008, Axis configuration list.....	335
8.4.9 K-0-0009, Osci: list of all K-parameters that can be oscilloscoped.....	337
8.4.10 K-0-0010, List of all K-parameters.....	337
8.4.11 K-0-0011, List of all invalid K-parameters.....	337
8.4.12 K-0-0012, Checksum of all K-parameters.....	338
8.4.13 K-0-0013, IDN list of checksum K-parameters.....	338
8.4.14 K-0-0016, Project identification number.....	338
8.4.15 K-0-0019, Kinematic type.....	338
8.4.16 K-0-0020, Kinematic diagnostic message.....	338
8.4.17 K-0-0021, Kinematic status.....	339
8.4.18 K-0-0022, Extended kinematic status.....	340
8.4.19 K-0-0023, Kinematic diagnostic number.....	340
8.4.20 K-0-0031, Transformation scheme.....	340
8.4.21 K-0-0034, Maximum path acceleration (bipolar).....	340
8.4.22 K-0-0035, Maximum path jerk (bipolar).....	341
8.4.23 K-0-0040, Transformation, lengths of the mechanics.....	341
8.4.24 K-0-0041, Transformation, factors.....	341
8.4.25 K-0-0042, Offset of MCS system.....	341
8.4.26 K-0-0050, Distance units MCS.....	342
8.4.27 K-0-0051, Velocity units MCS.....	342
8.4.28 K-0-0052, Acceleration units MCS.....	343
8.4.29 K-0-0053, Jerk units MCS.....	343
8.4.30 K-0-0054, Measuring unit for kinematic data.....	344
8.4.31 K-0-0070, Belt number of kinematics.....	344
8.4.32 K-0-0096, Peak value of command buffer size.....	345
8.4.33 K-0-0097, Maximum command buffer size.....	345
8.4.34 K-0-0098, Current command buffer size.....	345

Table of Contents

	Page
8.4.35 K-0-0099, Kinematic configuration.....	345
8.4.36 K-0-0100, Current position, Cartesian Machine Coordinate System.....	345
8.4.37 K-0-0102, Current path velocity.....	346
8.4.38 K-0-0103, Current path acceleration.....	346
8.4.39 K-0-0104, Current path jerk.....	346
8.4.40 K-0-0150, Interpolated command position.....	346
8.4.41 K-0-0152, Interpolated path velocity.....	346
8.4.42 K-0-0153, Interpolated path acceleration.....	346
8.4.43 K-0-0154, Interpolated path jerk.....	346
8.4.44 K-0-0223, Selection of error reaction.....	347
8.4.45 K-0-0224, Emergency halt deceleration.....	347
8.4.46 K-0-0225, Emergency halt jerk.....	347
8.4.47 K-0-0240, Feedrate override.....	347
8.4.48 K-0-0241, Feedrate override control word.....	347
8.4.49 K-0-0242, Acceleration override.....	348
8.4.50 K-0-0243, Deceleration override.....	348
8.4.51 K-0-0250, Continuous slope lead time.....	348
8.4.52 K-0-0815, Odometer.....	348
8.4.53 K-0-1000, Command: Load basic kinematic parameters.....	349
8.4.54 K-0-2100, List of all K-parameters to be backed up.....	349
9 Diagnostics.....	351
9.1 Diagnostics, General Information.....	351
9.2 MLC_TABLE, ERROR_TABLE 16#0030	351
9.3 SERCOS_TABLE, ERROR_TABLE 16#0010	360
10 Appendix.....	363
10.1 Transformations, Comparing Robot Control V1 to Robot Control V2.....	363
10.2 Overview: Kinematic parameters.....	363
10.3 Switching from Robot Control V1 to Robot Control V2.....	367
10.3.1 Converting RCL Commands.....	367
10.3.2 Converting RCL Programs.....	372
10.3.3 Converting IEC Commands.....	376
11 Service and Support.....	379
Index.....	381

1 About this Documentation

1.1 Validity of the Documentation

Target group This documentation provides an overview on Robot Control V2 of IndraMotion MLC and its kinematics interface.

Required qualification: Individual who is able to assess the tasks assigned and identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

This documentation supports the user during the following phases:

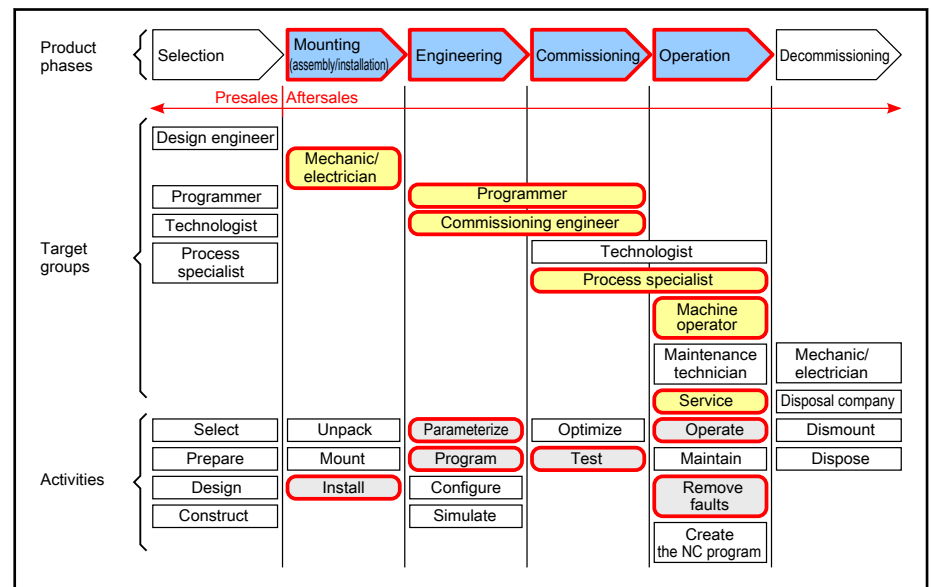


Fig. 1-1: Application phases

1.2 Documentation Structure

Robot Control V2 is based on the PLCopen / EN 61131-3 options.

Contained:

- [Functional Description, page 19](#), functional scope of Robot Control V2
- [Interface Functions, page 65](#), description of the IndraWorks interface of the "Robot" component
- [Transformation Schemes, page 73](#), description of the transformations
- [Application Examples on page 315](#)
- [Kinematics Interface on page 251](#) with library [ML_KinTech.library on page 254](#)
- Programming references for further information

For **both Robot Control types**, the description of the mutual library is contained in [ML_Robot.library, page 107](#).

About this Documentation

1.3 Using the Safety Instructions

1.3.1 Safety Instructions – Structure

The safety instructions are structured as follows:

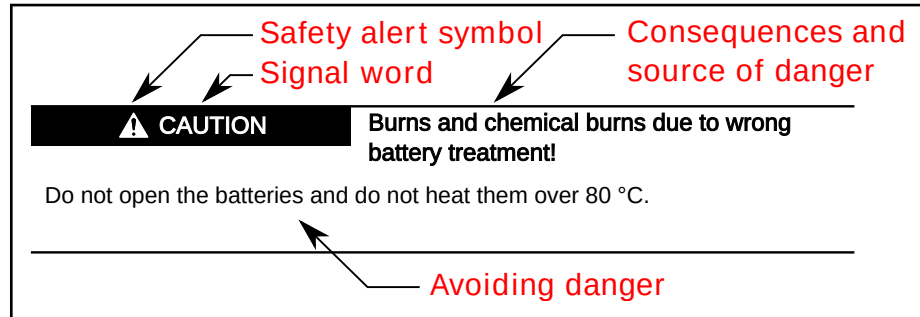


Fig. 1-2: Safety instructions – Structure

1.3.2 Explaining Signal Words and Safety Alert Symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and signifies the degree of danger.

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

DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury could occur.

NOTICE

In case of non-compliance with this safety instruction, property damage could occur.

1.3.3 Symbols Used

Notes are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.4 Required and Supplementing Documentations

Documentation titles with type codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software Installation DOK-IWORKS-SOFTINS*V13-CORS-EN-P, R911336880 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APRS-EN-P, R911336870 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Functional Description DOK-MLC***-FUNC****V13-APRS-EN-P, R911336295 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Functional Description DOK-XLC***-FUNC****V13-APRS-EN-P, R911336352 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APRS-EN-P, R911336356 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X

About this Documentation

IndraWorks		MLC	XLC
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Project Conversion DOK-XLCMLC-PROCONV*V13-APRS-EN-P, R911336366 This documentation describes the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks version 12 with IndraLogic 2G. Changes with regard to Motion and PLC are described in detail.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Commissioning DOK-MLC***-STARTUP*V13-CORS-EN-P, R911336308 This documentation describes the steps to commission and service the IndraMotion MLC system. It includes checklists for frequent tasks and a detailed description of the steps.	X	

Tab. 1-1: XCL/MLC documentation overview

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V13-LIRS-EN-P, R911336360 This documentation describes the function blocks, functions and data types of the RIL_Common-Types, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template DOK-XLCMLC-TF*GAT**V13-APRS-EN-P, R911336370 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.	X	X
/31/	Rexroth IndraMotion MLC 13VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V13-APRS-EN-P, R911336297 This documentation provides information on the RobotControl. The programming language RCL (RobotControl Language) is focused. The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/32/	Rexroth IndraMotion MLC 13VRS Robot Control V2 DOK-MLC***-ROCO****V13-RERS-EN-P, R911341035 The documentation provides information about the Robot Control V2. The focus of this documentation is on PLCopen programming. The program structure, variables, functions and motion commands as well as the required system parameters are described. Furthermore, the documentation contains the description of the ML_Robot library.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameters DOK-XLCMLC-PARAM***V13-RERS-EN-P, R911336364 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.	X	X

About this Documentation

Motion		MLC	XLC
/10/	Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKRS-EN-P, R911328670 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-07, MPB-07, MPD-07 and MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Functions DOK-INDRV*-MP*-17VRS**-APRS-EN-P, R911331236 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-17, MPM-17, MPC-17 and MPE-17.		

Tab. 1-2: XCL/MLC documentation overview

Field Buses		MLC	XLC
/4/	Rexroth IndraWorks 13VRS Field Buses DOK-IWORKS-FB*****V13-APRS-EN-P, R911336872 This documentation describes the supported field buses and their diagnostic function blocks.	X	X
/4a/	Rexroth IndraWorks 13VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V13-LIRS-EN-P, R911337857 This manual describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_Profine-tIO, RIL_ProfinetIODevice, RIL_EtherNet/IP Adapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.	X	X
/63/	SERCOS System Manual for I/O Devices DOK-CTRL-ILS3*****-APRS-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a SERCOS interface.	X	X

Tab. 1-3: XCL/MLC documentation overview

HMI		MLC	XLC
/8/	Rexroth IndraWorks 13VRS HMI DOK-IWORKS-HMI****V13-APRS-EN-P, R911336874 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 13VRS WinStudio DOK-IWORKS-WINSTUD*V13-APRS-EN-P, R911336882 This documentation describes the installation of the software, working with WinStudio and the creation and operation of applications.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI Connection DOK-XLCMLC-HMI*****V13-APRS-EN-P, R911336362 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.	X	X

Tab. 1-4: XCL/MLC documentation overview

About this Documentation

PLC		MLC	XLC
/3/	Rexroth IndraWorks 13VRS IndraLogic 2G Programming Instruction DOK-IWORKS-IL2GPRO*V13-APRS-EN-P, R911336876 This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.	X	X
/33/	Rexroth IndraWorks 13VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V13-LIRS-EN-P, R911336285 This documentation describes the system-comprehensive PLC libraries.	X	X

Tab. 1-5: XCL/MLC documentation overview

Technology		MLC	XLC
/30/	Rexroth IndraMotion MLC 13VRS Technology Libraries DOK-MLC***-TF*LIB**V13-LIRS-EN-P, R911336324 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.	X	
/60/	Rexroth IndraMotion MLC 13VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V13-LIRS-EN-P, R911336306 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.	X	
/62/	Rexroth IndraMotion MLC 13VRS RegisterControl (Application Manual) DOK-MLC***-REGI*CO*V13-APRS-EN-P, R911336304 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.	X	
/49/	Rexroth IndraMotion MLC 13VRS Winder Function Application DOK-MLC***-TF*WIND*V13-APRS-EN-P, R911336326 This application-related system documentation describes the application of the winder technology functions.	X	

Tab. 1-6: XCL/MLC documentation overview

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRRS-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRRS-EN-P, R911328474 This documentation describes the IndraControl L25 controls.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Function Modules DOK-CONTRL-FM*LXX**V13-APRS-EN-P, R911336283 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X

About this Documentation

Hardware		MLC	XLC
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PARS-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.		
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WARS-EN-P, R911297319 This documentation describes all diagnostics implemented in the following firmwares: - Drive controller firmwares from MPx-02 to MPx-08 and - firmwares of the supply devices of the type "HMV". It supports the operating crew as well as the programmer at troubleshooting.		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PRRS-EN-P, R911295012 This documentation is a project planning manual to select and use the control sections CSB01, CDB01 and CSH01 for drive controllers of the product families Rexroth IndraDrive M and Rexroth IndraDrive C.		

Tab. 1-7: XCL/MLC documentation overview

Diagnostics and service		MLC	XLC
/26/	Rexroth IndraWorks 13VRS IndraMotion Service Tool DOK-IWORKS-IMST****V13-RERS-EN-P, R911337707 This documentation describes the IndraMotion Service Tool (IMST). It is a web-based diagnostic tool to access the control systems IndraMotion MLC L25, L45 or 65 as well as the MLP via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. A PC with Internet Explorer version 6 or 7 is required.	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnostics DOK-XLCMLC-DIAG****V13-RERS-EN-P, R911336358 This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.	X	X

Tab. 1-8: XCL/MLC documentation overview

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 13VRS System Overview DOK-IM*PR*-SYSTEM**V13-PRRS-EN-P, R911336289 This documentation describes the product IndraMotion for Printing. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/48/	Rexroth IndraMotion for Packaging 13VRS System Overview DOK-IM*PA*-SYSTEM**V13-PRRS-EN-P, R911336287 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	

About this Documentation

System Overview		MLC	XLC
/9/	Rexroth IndraMotion MLC 13VRS System Overview DOK-MLC***-SYSTEM**V13-PRRS-EN-P, R911336310 This documentation provides an overview on the hardware/software components of the automation system IndraMotion MLC in the mentioned version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 13VRS System Overview DOK-XLC***-SYSTEM**V13-PRRS-EN-P, R911336354 This documentation provides an overview on hardware/software components of the automation system IndraLogic XLC in the respective version. It helps assembling a system.		X

Tab. 1-9: XCL/MLC documentation overview

First Steps		MLC	XLC
/25/	Rexroth IndraMotion MLC 13VRS, First Steps DOK-MLC***-F*STEP**V13-CORS-EN-P, R911336293 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.	X	
/25/	Rexroth IndraLogic XLC 13VRS, First Steps DOK-XLC***-F*STEP**V13-CORS-EN-P, R911336350 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.		X

Tab. 1-10: XCL/MLC documentation overview

1.5 Names and Abbreviations

Term	Explanation
ACS	Axes Coordinate System
BCS	Basic Coordinate System
GAT	Generic Application Template, application programs for Motion and Logic applications
MCS	Machine Coordinate System
PCS	Product Coordinate System
PTP	Point-To-Point
RCL	Programming language Robot Control Language (Robot Control V1)
TCP	Tool Center Point
MLPI	Motion Logic Programming Interface

Tab. 1-11: Names and abbreviations used

2 Functional Description

2.1 System Overview

In addition to single-axis motions, IndraMotion MLC also provides coordinated three-dimensional multi-axis motions.

Single-axis motions only enable coordinated motions of several axes by calculating the positions of the slave axes from the position of a master axis.

Multi-dimensional motions go beyond this. Here, axes are combined in a kinematics. No additional master axis is required. A path in space is defined for this kinematics.

Therefore, paths can be planned in a more flexible way than with simple master/slave relations. Furthermore, there may be problems with master/slave relations if an error occurs in one slave axis and the other axes do not detect it and keep moving. A kinematics knows which axes belong to the group and can therefore better react to errors.



The Robot Control functionality is only provided with IndraMotion MLC.

IndraMotion MLC can calculate several kinematics at the same time.

Number of kinematics, depending on the control type:

- IndraMotion MLC L25: 4 kinematics
- IndraMotion MLC L45: 16 kinematics
- IndraMotion MLC L65: 16 kinematics

One kinematics may comprise up to 16 axes.

2.2 Coordinate Systems and Transformations

2.2.1 Coordinate Systems

Coordinate Systems, Overview

The spatial position of a kinematics can be represented using several coordinate systems. For each position, the corresponding coordinate system has to be specified.

Coordinate system	Description
Axis Coordinate System (ACS) on page 21	Axis reference
Base Coordinate System (BCS) on page 21	Kinematics reference
Machine Coordinate System (MCS) on page 22	Reference to the machine system
Product Coordinate System (PCS) on page 22	Reference to the product or workpiece

Tab.2-1: Overview on coordinate systems

Types of Coordinate Systems

Coordinate systems can be subdivided into the following types:

Cartesian coordinate system

The Cartesian coordinate system has the following properties:

- **Cartesian:** The three coordinate axes are orthogonal (at a right angle) to each other

Functional Description

- **Three-dimensional:** There are three coordinate axes. A point in space is specified by three values (X, Y, Z)
- **Right-handed:** Use the right hand to form a three-dimensional, Cartesian coordinate system; the thumb is the X-axis, the index finger is the Y-axis and the middle finger is the Z-axis

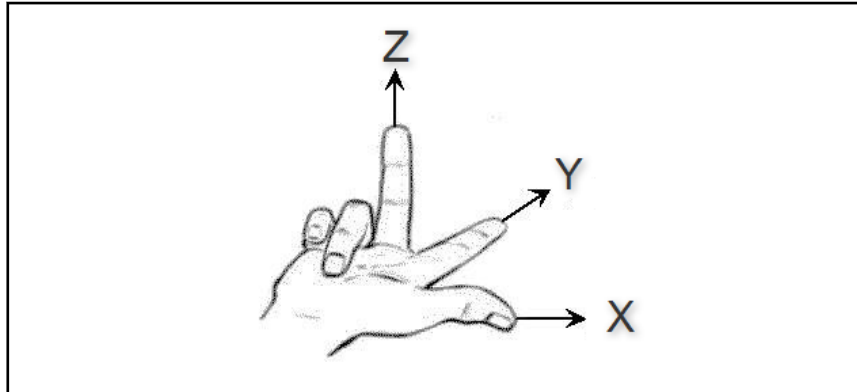


Fig.2-1: Right-handed coordinate system

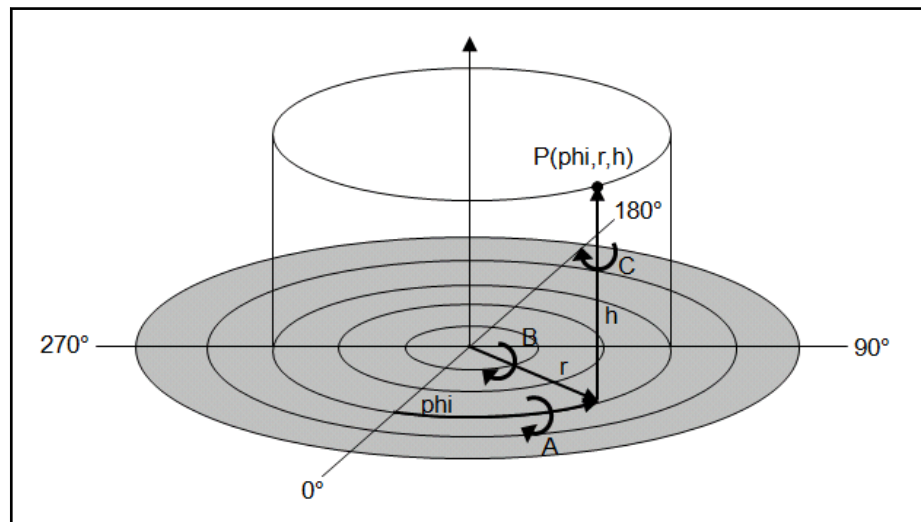
In addition to its spatial position, the orientation of an object is indicated by rotating it around the individual axes.

A is the rotation around the X-axis, B is the rotation around the Y-axis, and C is the rotation around the Z-axis.

Therefore, the position of an object is specified by the 6 tuple (X, Y, Z, A, B, C).

Cylindrical coordinate system

In a cylindrical coordinate system, the position of an object is specified by three coordinates. However, the coordinate axes are not orthogonal to each other as with a Cartesian coordinate system. The position of an object in a cylinder coordinate system is defined by (phi, r, h).



phi	Angle of the object in the mathematically positive direction of rotation (counterclockwise) from the 0° straight line (angle coordinate)
r	Distance of the object from the height axis (radial coordinate)
h	Object height from the reference plane, that is determined by the phi and r coordinate axes (vertical coordinate)
A	Rotation around the angle axis
B	Rotation around the radial axis
C	Rotation around the vertical axis

Fig.2-2: Cylindrical coordinate system

Functional Description

Therefore, the position of an object is specified by the 6 tuple (ϕ , r , h , A , B , C).

Axis-related coordinate system

In an axis-related coordinate system, the position and orientation of an object is determined by the axis positions.

A coordinate has to be specified for each axis of the kinematics.

Example:

Axis-related coordinate system of a 2-axis kinematics

The position of a 2-axis kinematics with rotary axes is specified using the angles α and β as coordinates (α , β).

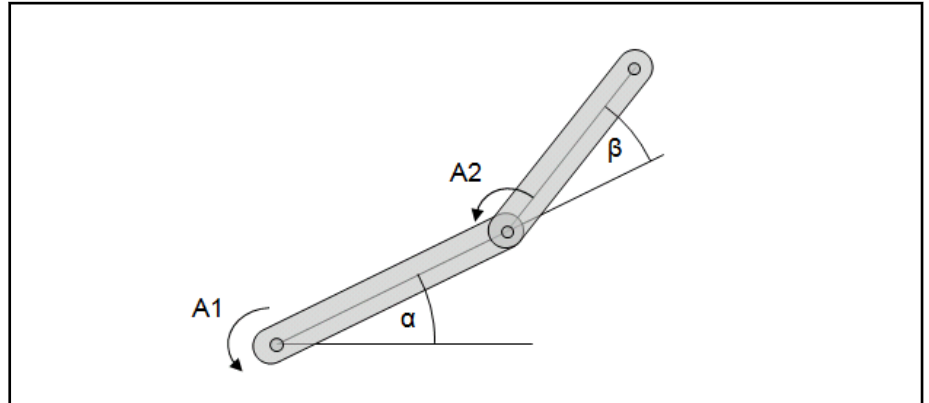


Fig.2-3: Axis-related coordinate system of a 2-axis kinematics

Axis Coordinate System (ACS)

The axis coordinate system (ACS = **A**xis **C**oordinate **S**ystem) is an axis-related coordinate system.

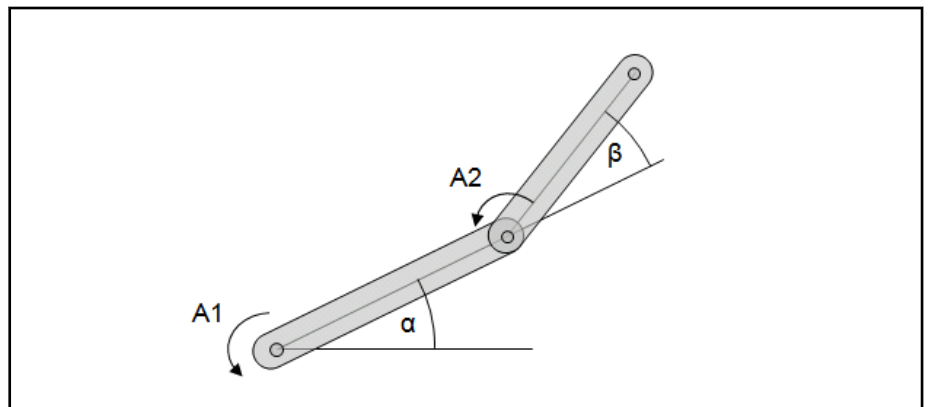


Fig.2-4: ACS

Base Coordinate System (BCS)

The base coordinate system (BCS = **B**ase **C**oordinate **S**ystem) is the coordinate system of the kinematics.

The BCS is a Cartesian coordinate system.

Functional Description

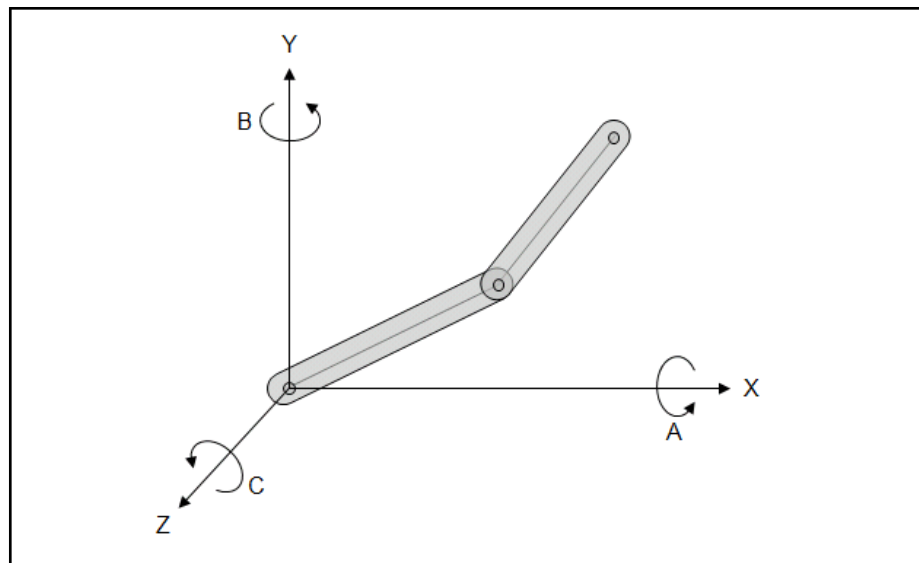


Fig.2-5: BCS

Machine Coordinate System (MCS)

The machine coordinate system (MCS = **M**achine **C**oordinate **S**ystem) is the coordinate system of the machine system.

The MCS is the joint coordinate system of all kinematics (BCS).

The MCS is a Cartesian coordinate system.

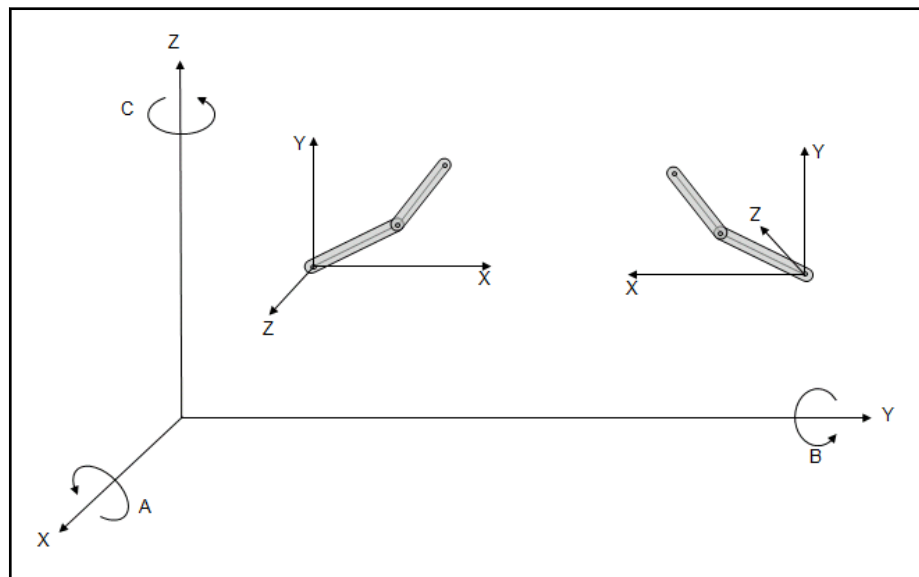


Fig.2-6: MCS

Product Coordinate System (PCS)

The product coordinate system (PCS = **P**roduct **C**oordinate **S**ystem) is the coordinate system of the product.

A belt-synchronous work space is defined by a PCS (see [Belt-synchronous Motion, General Information on page 49](#)).

16 PCS may be specified per kinematics.

The PCS is either a Cartesian or cylindrical coordinate system.

Functional Description

The use of product coordinate systems in connection with belt synchronization is described in [Application example 2: Belt Synchronization on page 316](#).

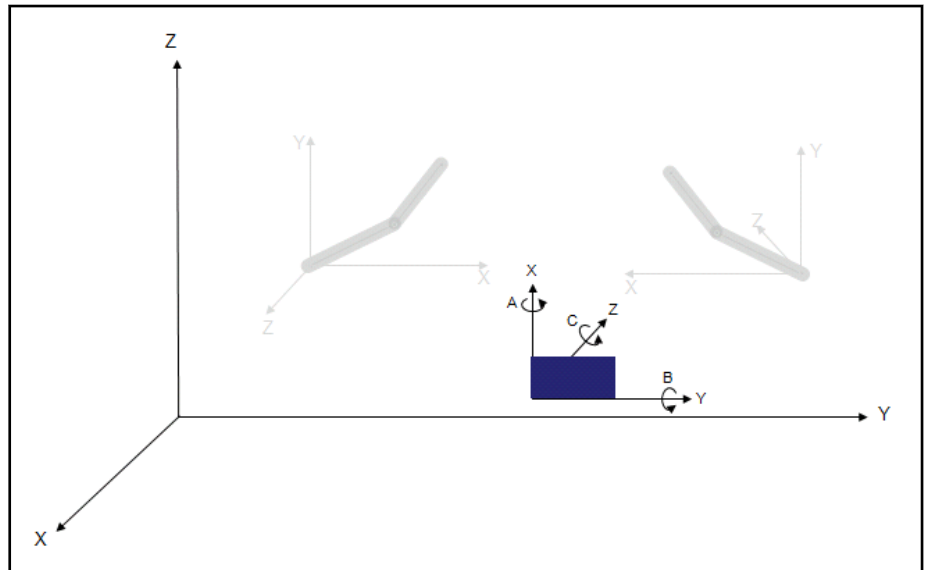


Fig.2-7: PCS (Cartesian)

2.2.2 Transformations

Transformations, Overview

Using transformations, positions may be converted from one coordinate system to another coordinate system.

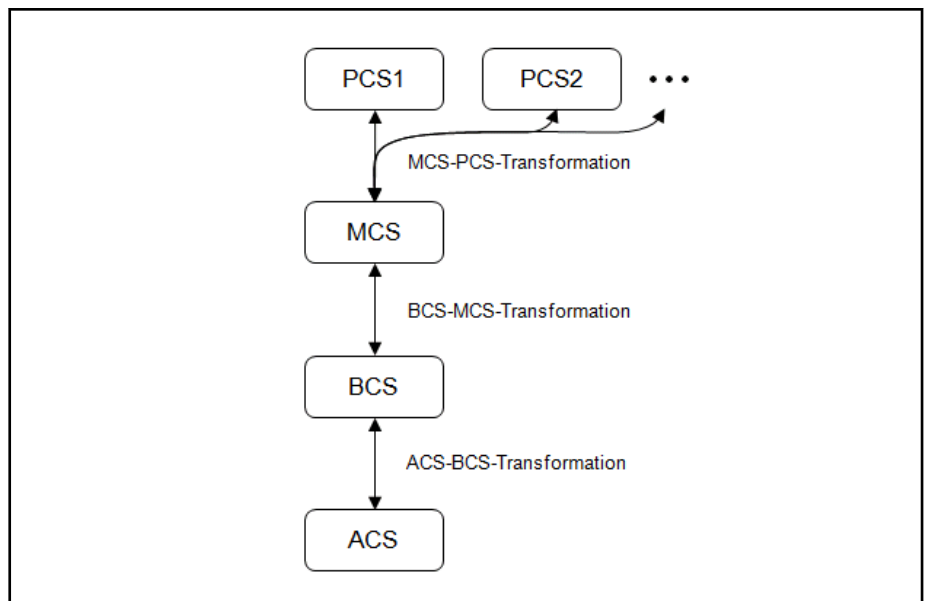


Fig.2-8: Overview of transformations

ACS/BCS Transformation

The ACS/BCS transformation converts ACS and BCS positions.

The transformation scheme of ACS/BCS transformation is configured in parameter K-0-0031 (Transformation scheme) (e.g. Cartesian, Scara, etc.) and configured in parameters K-0-0040 (Transformation, lengths of the mechanics) and K-0-0041 (Transformation, factors).

Functional Description

Refer to [Kinematic-Related Parameters](#), page 334

For a list of available transformation schemes, see: [Transformation Schemes](#), page 73.

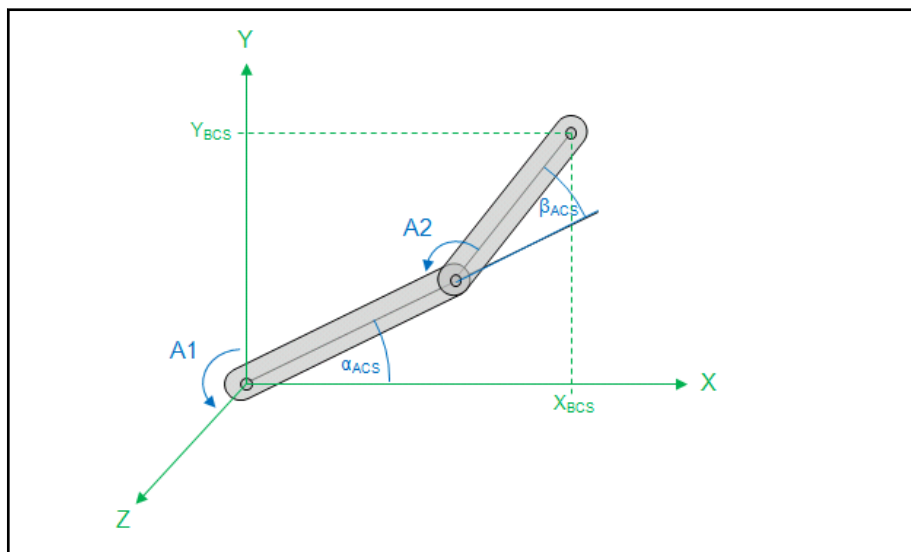


Fig.2-9: ACS/BCS

Example:

K-0-0031 = 9 (Scara)

K-0-0040 = 100, 200 (arm length L1, L2)

ACS: (α_{ACS} , β_{ACS}) = (90°, -90°)

BCS: (X_{BCS} , Y_{BCS} , Z_{BCS} , A_{BCS} , B_{BCS} , C_{BCS}) = (200, 100, 0, 0, 0, 0)

BCS/MCS Transformation

The BCS/MCS transformation converts BCS and MCS positions.

Rotation and displacement between BCS and MCS are configured in parameter K-0-0042 (Offset of MCS system).

Refer to [Kinematic-Related Parameters](#), page 334

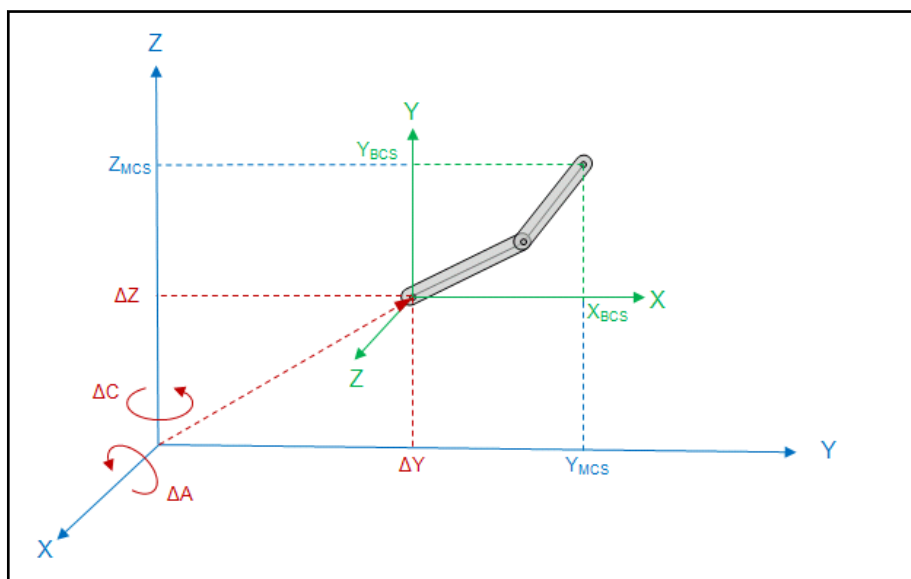


Fig.2-10: BCS/MCS

Functional Description

Example:

K-0-0042 = (ΔX , ΔY , ΔZ , ΔA , ΔB , ΔC) = (0, 111, 499, 0°, 0°, 90°)

BCS: (X_{BCS} , Y_{BCS} , Z_{BCS} , A_{BCS} , B_{BCS} , C_{BCS}) = (200, 100, 0, 0°, 0°, 0°)

MCS: (X_{MCS} , Y_{MCS} , Z_{MCS} , A_{MCS} , B_{MCS} , C_{MCS}) = (-100, 311, 499, 0°, 0°, 90°)



Rotation and displacement between BCS and MCS can be read out using the command [Get Trafo Values on page 30](#).

MCS/PCS Transformation

The MCS/PCS transformation converts MCS and PCS positions.

Rotation and displacement between MCS and PCS are configured using the following commands:

- [Set Cartesian Transform on page 30](#), with a Cartesian PCS
- [Set Cylindric Transform on page 30](#), with a cylindrical PCS

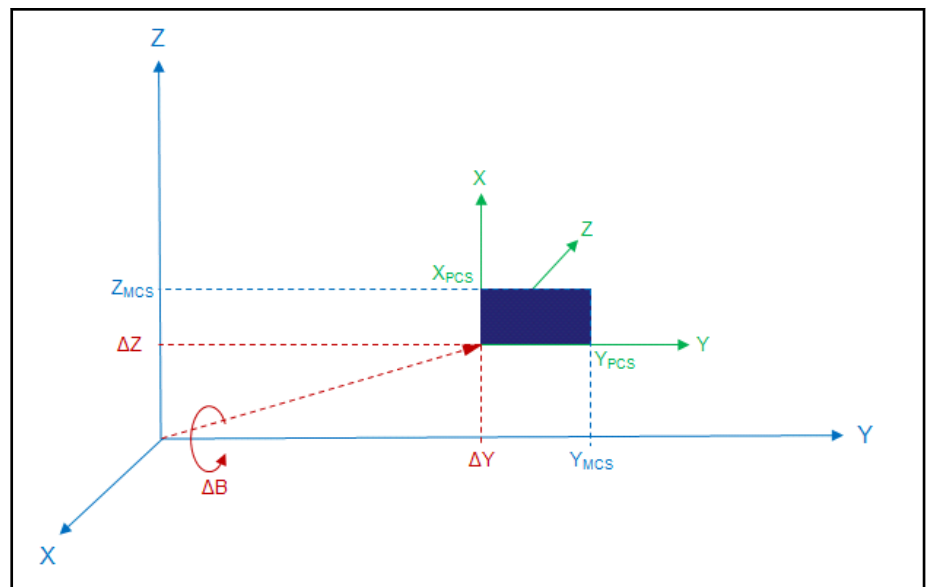


Fig.2-11: MCS/PCS

Example:

Set Cartesian Transform = (ΔX , ΔY , ΔZ , ΔA , ΔB , ΔC) = (0, 111, 499, 0°, -90°, 0°)

MCS: (X_{MCS} , Y_{MCS} , Z_{MCS} , A_{MCS} , B_{MCS} , C_{MCS}) = (0, 211, 699, 0°, 0°, 0°)

PCS: (X_{PCS} , Y_{PCS} , Z_{PCS} , A_{PCS} , B_{PCS} , C_{PCS}) = (200, 100, 0, 0°, 90°, 0°)



Rotation and displacement between MCS and PCS can be read out using the command [Get Trafo Values on page 30](#).

For spatial rotation of the product coordinate system only the degrees of freedom are supported that are also supported by the transformation, e.g. 3-axis delta transformation mechanically only supports rotation around the Z axis. The product coordinate system can therefore only be rotated around the Z axis.

Functional Description

2.3 State Diagram of a Kinematics

A kinematics has a state diagram, just as a single axis.

The state diagram describes the states and state transitions of a kinematics.

The active state of a kinematics is described by parameter K-0-0022, Extended kinematic status.

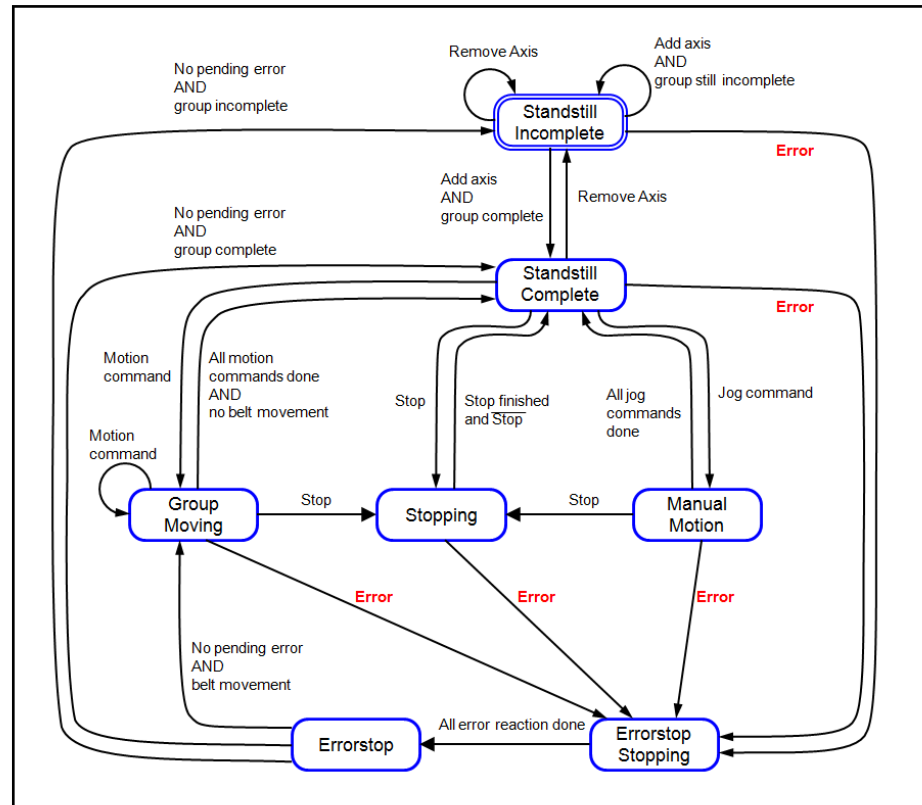


Fig.2-12: State diagram of a kinematics

States of a kinematics

- **Standstill incomplete**

"Standstill Incomplete" is the initial state.

In state "Standstill Incomplete" some of the axes configured in parameter K-0-0008 have not yet been added to the kinematics.

Use the commands [Group Axis on page 30](#) and [Group All Axes on page 30](#) to add axes to the kinematics. If all configured axes are added to the kinematics, the state switches to "Standstill Complete".

- **Standstill complete**

In state "Standstill Complete" all axes configured in parameter K-0-0008 are added to the kinematics.

Use the commands [Remove Axis on page 30](#) and [Remove All Axes on page 30](#) to remove axes from the kinematics. If an axis is removed from the kinematics, the state switches to "Standstill Incomplete".

- **Group Moving**

In state "Group Moving", at least one motion command (see [List of all motion commands and functions on page 29](#)) or belt-synchronous motion (see [Belt-synchronous Motion on page 49](#)) is active.

Functional Description

If all motion commands are done and no belt-synchronous motion is active, the state switches to "Standstill Complete".

- **Stopping**

In state "Stopping" a "Stop" command is active (see [Stop on page 29](#)).

Once the kinematics is in standstill and stopping is complete, the state switches to "Standstill Complete".

- **Manual Motion**

In state "Manual Motion" at least one manual mode command (jog command) ("[Jog Step on page 29](#)" or "[Jog Cont on page 29](#)") is active.

If all manual mode commands are done, the state switches to "Standstill Complete".

- **Errorstop Stopping**

In "Errorstop Stopping" state, the error reactions of the kinematics (see [Error Reaction of the Commanded Motion on page 60](#)) and the belts are executed (see [Belt-specific Error Reaction on page 60](#)).

The "Errorstop Stopping" state becomes active if a kinematics error occurs.

When all error reactions are completed, the state switches to "Error-stop".

- **Errorstop**

In "Errorstop" state, a kinematics error occurred and the error reactions are completed.

When the errors were acknowledged using "Clear error", the state switches to "Standstill Incomplete", "Standstill Complete" or "Group Moving", depending on the conditions met.

2.4 Robot Control, Commands and Functions

2.4.1 Robot Control, Commands and Functions, Overview

There are two types of actions regarding a kinematics:

- Commands
- Functions

Commands

The command is processed asynchronously. This means that issuing a command only gives the order to execute a certain action. The action itself is carried out at a later time. For this purpose, the command is put into the command buffer. This command buffer uses the FIFO principle (First In First Out). Commands that are issued first are processed prior to commands that are issued later.

Set the maximum number of commands in the command buffer using parameter K-0-0097, Maximum command buffer size.

Parameter K-0-0098, Current command buffer size contains the number of commands currently in the command buffer.

Parameter K-0-0096, Peak value of command buffer size contains the maximum number of commands that ever were in the command buffer.

In order to blend motions, the commands have to be buffered (see [Blending on page 39](#)).

The command state indicates the processing step of the command (see [Command State on page 28](#)).

Functional Description

Functions In contrast to commands, functions do not use the command buffer. Functions are processed directly when they are called (synchronous processing). This means that the result is available directly after processing the function.

Types of commands and functions

	Description
List of all motion commands and functions on page 29	Motion of a kinematics in space
List of all configuration commands on page 30	Configuration of a kinematics
List of all information functions on page 30	Information on a kinematics
List of all teach-in commands and functions on page 31	Save kinematics positions as points and reuse them

Tab.2-2: *Types of commands and functions*

Motion commands A motion command issued for a kinematics causes a spatial movement of the "Tool Center Point" (TCP).

The TCP is a specified point of the kinematics that usually refers to the kinematics top. The user sets the new TCP target position. The type of motion command determines the path used to reach the new TCP position.



The new target position can be set in any coordinate system. The coordinate system has no impact on the path used to reach the point.

Refer to [Coordinate Systems, Overview, page 19](#)

A motion command consists of the following:

- **End point and/or intermediate points:** Points on the motion path
- **Maximum values:** Maximum values for the velocity profile, such as maximum velocity and maximum acceleration
- **Motion type:** Type of path used to reach the intermediate points and the end point
Refer to [Motion Types, page 31](#)
- **Blending and/or buffer type:** Manner to start and/or blend the block
Refer to [Blending, page 39](#)

2.4.2 Command State

The command state indicates the processing step of the command in relation to the command buffer. A command can be in one of the following states:

Command state

State	Description
InTransmission	The command was issued and written to the command buffer
InBuffer	The command was prepared, it is in the command buffer and waits for activation
Active	Command is enabled
Done	The command was executed successfully
Aborted	The command was aborted
Error	A command error occurred

Tab.2-3: *Command state*

2.4.3 List of all Commands and Functions

The same function/command can be called via several interfaces. The following table lists the call names in the respective interface and the reference name of the functional description as well as the Robot Control version that supports it.

The following interfaces are used:

- **PLC:** PLC function block name
- **MLPI core:** Function name of the C/C++ Motion Logic Programming Interface
- **MLPI COM:** Function name of the COM Motion Logic Programming Interface
- **MLPI Java:** Function name of the Java Motion Logic Programming Interface

Keep in mind that MLPI is only available for Robot Control V2 as a rule.

List of all motion commands and functions

Reference description	PLC	MLPI core	MLPI COM MLPI Java	V1	V2
Move Linear Absolute	MC_MoveLinearAbsolute	mlpiRobotMoveLinearAbs	MoveLinearAbsolute	x	x
Move Linear Relative	MC_MoveLinearRelative	mlpiRobotMoveLinearRel	MoveLinearRelative	x	x
Move Circular Absolute	MC_MoveCircularAbsolute	mlpiRobotMoveCircularAbs	MoveCircularAbsolute	x	x
Move Circular Relative	MC_MoveCircularRelative	mlpiRobotMoveCircularRel	MoveCircularRelative		x
Move Direct Absolute	MC_MoveDirectAbsolute	mlpiRobotMoveDirectAbs	MoveDirectAbsolute	x	x
Move Direct Relative	MC_MoveDirectRelative	mlpiRobotMoveDirectRel	MoveDirectRelative		x
Synchronization with Limits	ML_BeltSynchronization-WithLimits	mlpiRobotSyncOnWithLimits	SyncOnWithLimits		x
Desynchronization with Limits	ML_BeltDeSynchronization-WithLimits	mlpiRobotSyncOffWithLimits	SyncOffWithLimits		x
Jog Cont	ML_GroupJogCont	mlpiRobotJogCont	JogCont		x
Jog Step	ML_GroupJogStep	mlpiRobotJogStep	JogStep		x
Open Cyclic Position	ML_RobotOpenCyclicPos	mlpiRobotOpenCyclicPos			x
Write Cyclic Position (function)	ML_RobotWriteCyclicPos	mlpiRobotWriteCyclicPos			x
Stop	MC_GroupStop	mlpiRobotStop	Stop		x
Interrupt	MC_GroupInterrupt	mlpiRobotInterrupt	Interrupt		x
Continue	MC_GroupContinue	mlpiRobotContinue	Continue		x
Wait	ML_GroupWait	mlpiGroupWait	Wait		x

V1

Robot Control V1

V2

Robot Control V2

Tab.2-4:

List of all motion commands and functions

Functional Description

List of all configuration commands

Reference description	PLC	MLPI core	MLPI COM MLPI Java	V1	V2
Group Reset	MC_GroupReset	mlpiRobotReset	Reset	x	x
Group Axis	MC_AddAxisToGroup	mlpiMotionAddAxisToGroup		x	x
Remove Axis	MC_RemoveAxis- FromGroup	mlpiMotionRemAxis- FromGroup		x	x
Group All Axes	ML_GroupAllAxes	mlpiRobotAddAllAxis- ToGroup	AddAllAxisToGroup	x	x
Ungroup All Axes	MC_UngroupAllAxes	mlpiRobotRemAllAxis- FromGroup	RemAllAxisFromGroup	x	x
Set Kinematik Trans- formation	MC_SetKinTransform			x	x
Set Kinematics Trans- form Extended	MC_SetKinTransformExt				x
Set Cartesian Trans- formation	MC_SetCartesianTransform	mlpiRobotSetCartesian- Transform	SetCartesianTransform		x
Set Cylindric Transfor- mation	ML_SetCylindricTransform	mlpiRobotSetCylindric- Transform	SetCylindricTransform		x
Set Belt Configuration	ML_SetBeltConfiguration	mlpiRobotSetBeltConfigura- tion	SetBeltConfiguration		x
Set Belt Desync Rela- tive Configuration	ML_SetBeltDesyncRelCon- figuration	mlpiRobotSetBeltDesyn- cRelConfiguration	SetBeltDesyncRelCon- figuration		x
Set Rotary Table Con- figuration	ML_SetRotaryTableConfi- guration	mlpiRobotSetRotaryTable- Configuration	SetRotaryTableConfi- guration		
Set Belt Error Reaction	ML_SetBeltErrorReaction	mlpiRobotSetBeltErrorRe- action	SetBeltErrorReaction		x
Set Belt Parameter	ML_SetBeltParameter			x	
Set Safe Zone	ML_GroupSetSafeZone	mlpiRobotSetSafeZone	SetSafeZone		x

V1 Robot Control V1

V2 Robot Control V2

Tab.2-5: List of all configuration commands

List of all information functions

Reference description	PLC	MLPI core	MLPI COM MLPI Java	V1	V2
Read Pos	MC_GroupReadActualPosi- tion	mlpiReadPos	ReadPos		x
Switch On Path	ML_SwitchOnPath			x	x
Read Cyclic Position	ML_RobotReadCyclicPos	mlpiRobotReadCyclicPos			x
Move Point	ML_RobotMovePoint	mlpiRobotMovePoint	MovePoint		x
Transform Point	ML_RobotTransformPoint	mlpiTransformPoint	TransformPoint		x

Functional Description

Reference description	PLC	MLPI core	MLPI COM MLPI Java	V1	V2
Set Performance Enable	ML_RobotSetPerformanceEnable	mlpiRobotSetPerformanceEnable	SetPerformanceEnable		x
Get Performance	ML_RobotGetPerformance	mlpiRobotGetPerformanceResult	GetPerformanceResult		x
Get Mechanic	ML_RobotGetMechanic	mlpiRobotGetMechanicalData	GetMechanicData		x
Get Trafo Values	ML_RobotGetTrafoValues	mlpiRobotGetTrafoValues	GetTrafoValues		x
Get Belt Configuration	ML_RobotGetBeltConfiguration	mlpiRobotGetBeltConfiguration	GetBeltConfiguration		x
Get Belt Desync Relative Configuration	ML_RobotGetBeltDesyncRelConfiguration	mlpiRobotGetBeltDesyncRelConfiguration	GetBeltDesyncRelConfiguration		x
Get Belt Error Reaction	ML_RobotGetBeltErrorreaction	mlpiRobotGetBeltErrorReaction	GetBeltErrorReaction		x

V1 Robot Control V1
V2 Robot Control V2
Tab.2-6: List of all information functions

List of all teach-in commands and functions

Reference description	PLC	MLPI core	MLPI COM MLPI Java	V1	V2
Write Point	ML_GroupWritePoint	mlpiRobotTeachInWritePoint			x
Read Point	ML_GroupReadPoint	mlpiRobotTeachInReadPoint			x
Read Next Point	ML_GroupReadNextPoint	mlpiRobotTeachInReadNextPoint			x
Delete Points	ML_GroupDeletePoints	mlpiRobotTeachInDeletePoints			x
Save Point File	ML_GroupSavePointFile	mlpiRobotTeachInSavePointFile			x
Load Point File	ML_GroupLoadPointFile	mlpiRobotTeachInLoadPointFile			x
Delete Point File	ML_GroupDeletePointFile	mlpiRobotTeachInDeletePointFile			x

V1 Robot Control V1
V2 Robot Control V2
Tab.2-7: List of all teach-in commands and functions

2.5 Motion Types

2.5.1 Motion Types, Overview

The MLC Robot Control supports kinematics motions in three motion types:

- Linear (straight line in the space), [Linear Motion Type on page 32](#)

Functional Description

- Circular (circle segment in the space), [Circular Motion Type on page 33](#)
- Point to point (PTP), [Point to Point Motion Type on page 34](#)



With interpolation, orientation motions are controlled fully synchronously.

Furthermore, "[Absolute Positioning on page 34](#)" is distinguished from "[Relative Positioning on page 34](#)".

Motion types with absolute and relative positioning

	Absolute	Relative
Linear (straight line in space)	Move Linear Absolute	Move Linear Relative
Circular (circle segment in the space)	Move Circular Absolute	Move Circular Relative
Point to point (PTP)	Move Direct Absolute	Move Direct Relative

Tab.2-8: Overview of motion types

2.5.2 Linear Motion Type

The kinematics travels to the target point on a straight line.

The straight line is geometrically defined by two points. The control knows point P1 as it is the last point moved to. Therefore, it is sufficient to specify the target point P2.

The single axis motions result from the path and the inverse kinematics transformations.

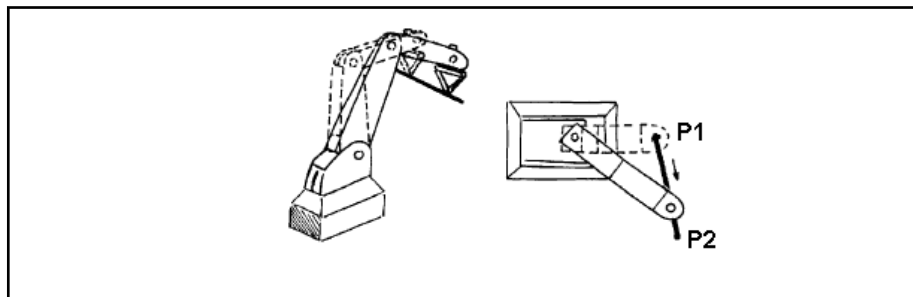


Fig.2-13: Linear interpolation method

The following motion commands may be used for this motion type:

- [Move Linear Absolute on page 29](#)
- [Move Linear Relative on page 29](#)

Orientation motions

Orientation motions are carried out synchronously to the spatial motion. This means, that the orientation is changed by x % when the kinematics traveled x % of the spatial distance. Therefore, acceleration, traveling with a constant velocity and deceleration on the spatial path are time-synchronous to the orientation motion.

Furthermore, velocity, acceleration and jerk on the path are limited in such a way that the single-axis values for velocity, acceleration and jerk of the orientation axes (A-0-0032, A-0-0033, A-0-0034, A-0-0036) are not exceeded. For this reason, the programmed maximum values of the Move Linear command may not be reached.

Functional Description

If only an orientation motion is commanded without a spatial motion using a Move Linear command, this motion is carried out according to the single-axis limit values of the orientation axes.

For this kind of motion, it is advisable to use the PTP motion type as the values for the velocity profile may then be influenced by a transfer parameter (see [Point to Point Motion Type on page 34](#)).

2.5.3 Circular Motion Type

The kinematics approaches the target point on a circle segment in the space.

A circle is geometrically defined by three points. In addition to target point P2, specify an intermediate point "PZ" so that the control can correctly calculate the circle segment.

The control knows point P1 as it is the last point traveled to. Therefore, it is sufficient to specify the intermediate point PZ and the target point P2.

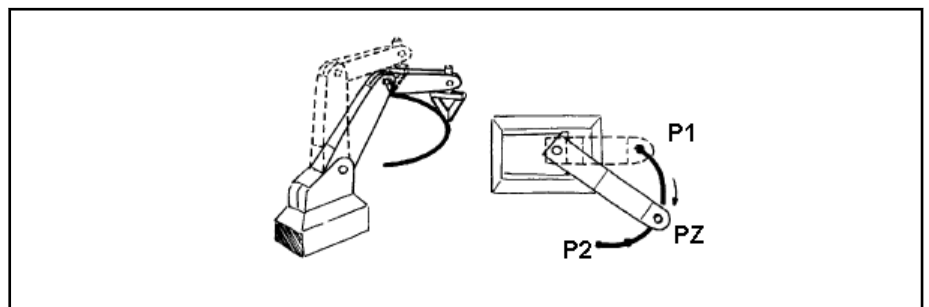


Fig.2-14: Circular interpolation type

The following motion commands may be used for this motion type:

- [Move Circular Absolute on page 29](#)
- [Move Circular Relative on page 29](#)

Orientation motions

With orientation motions, only the orientation of the end point is considered. The orientation of the intermediate point is ignored.

Orientation motions are carried out fully synchronously to the spatial motion. This means, that the orientation is changed by x % when the kinematics traveled x % of the spatial distance. Therefore, acceleration, traveling with a constant velocity and deceleration on the spatial path are time-synchronous to the orientation motion.

Furthermore, velocity, acceleration and jerk on the path are limited in such a way that the single-axis values for velocity, acceleration and jerk of the orientation axes (A-0-0032, A-0-0033, A-0-0034, A-0-0036) are not exceeded. For this reason, the set maximum values of the Move Circular command may not be reached.

If only an orientation motion is commanded without a spatial motion using a Move Circular command, this motion is carried out according to the single-axis limit values of the orientation axes.

For this kind of motion, it is advisable to use the PTP motion type as the values for the velocity profile may then be influenced by a transfer parameter (see [Point to Point Motion Type on page 34](#)).

Degenerate circles

Degenerate circles are created if:

- Start, intermediate, and end points are located on a straight line
- Start and end point are identical
- Start and intermediate point are identical
- Intermediate and end point are identical

Functional Description

In these cases, the kinematics travels on a straight line between the start and end point instead of in circle. The intermediate point is ignored.



The intermediate point is also ignored if the start, intermediate and end points are located on a straight line and the intermediate point is situated behind the end point.

In this case the intermediate point is not reached.

2.5.4 Point to Point Motion Type

With this motion type the kinematics moves to the commanded position as fast as possible.

Every individual axis travels from the start point to the end point on the shortest path.

The velocity profile of the motion is determined by the motion limit values (A-0-0032, A-0-0033, A-0-0034, A-0-0036) of the single axes.

The path in the space and the path velocity are irrelevant. It is guaranteed that the motions of all single axes are completed at the same time. All motion stages (acceleration, constant velocity, deceleration) of all axes are started and completed at the same time. In general, this is referred to as fully synchronous PTP.

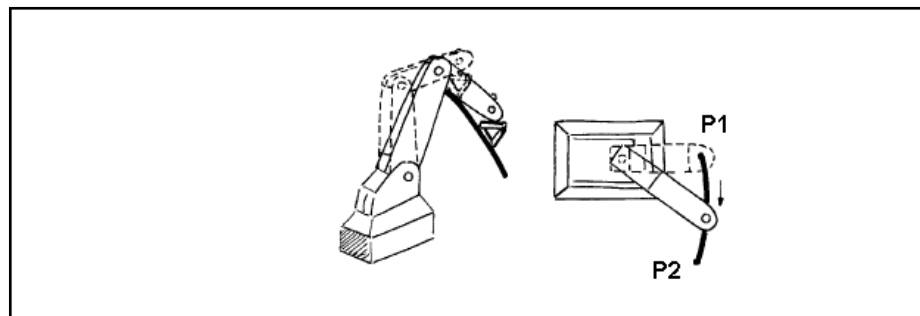


Fig.2-15: Direct interpolation type

The following motion commands may be used for this motion type:

- [Move Direct Absolute on page 29](#)
- [Move Direct Relative on page 29](#)

2.5.5 Absolute Positioning

Absolute positioning means specifying a position. This specified position is approached absolutely in the space.

2.5.6 Relative Positioning

Relative positioning means specifying a distance. This specified distance is moved in relation to the last commanded position. That means the last commanded position is added to the distance to move. This results in the new absolute end position.

2.6 Manual Mode

2.6.1 Manual Mode, General Information

To move a kinematics in manual mode, use one of the following command types:

- [Single-step Operation on page 36](#)
- [Continuous Operation on page 36](#)

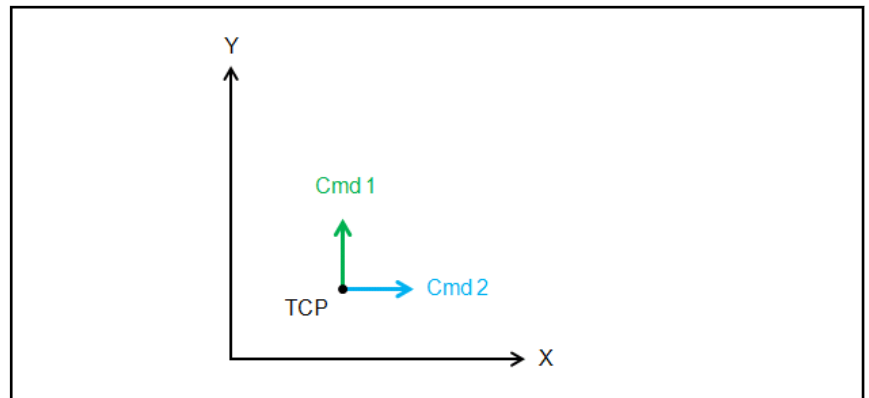
Functional Description

Manual mode supports the following motion types (see [Motion Types on page 31](#)) using the specified coordinate system:

- **PTP (ACS):**
With the ACS coordinate system use the PTP motion type
- **Linear (BCS, MCS, PCS):**
With the BCS, MCS or PCS coordinate systems use the Linear motion type

The commands in manual mode have the following properties:

- It is possible to execute several manual mode commands in **different** directions at the same time

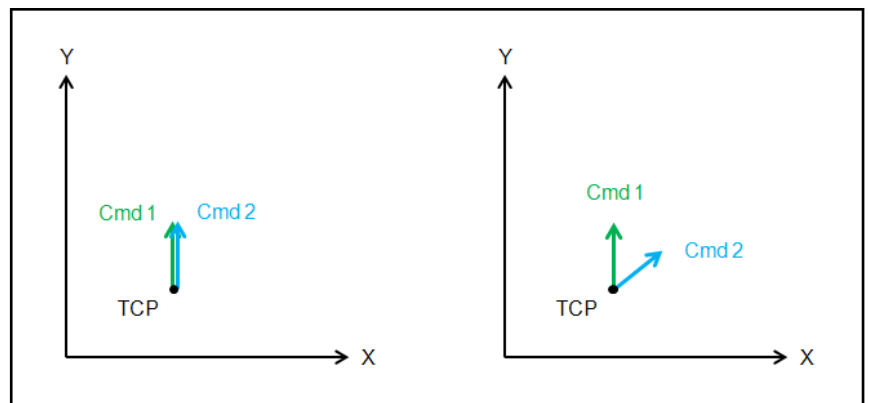


Cmd 1 Manual mode command in Y-direction

Cmd 2 Manual mode command in X-direction

Fig.2-16: Several manual mode commands in different directions

- It is not possible to execute several manual mode commands in the **same or partially same** direction at the same time



Cmd 1 Manual mode command in Y-direction

Cmd 2 Manual mode command in Y-direction and partially in Y-direction

Fig.2-17: Several manual mode commands in the same or partially same direction

- It is not possible to execute several manual mode commands with **differing coordinate systems** (e.g. MCS and BCS) at the same time
- It is not possible to execute several manual mode commands with **differing command types** (e.g. single-step and continuous operation) at the same time

Functional Description

2.6.2 Single-step Operation

To move a kinematics by a certain step width in manual mode, use the command [Jog Step on page 29](#).

The step width is the specified distance in the specified coordinate system.

2.6.3 Continuous Operation

To move a kinematics continuously into a certain direction in manual mode, use the command [Jog Cont on page 29](#).

While this command is enabled, the kinematics travels into the specified direction in the specified coordinate system.

If the command is terminated, the kinematics stops with the specified maximum values.

2.7 Teach-In

2.7.1 Teach-In, General Information

Use the teach-in function to save kinematics positions as points and reuse them.

The point names have the following conventions:

- Names have a maximum length of 20 characters
- Names may not contain any special characters " \ ; , *"
- A differentiation is made between the use of upper and lower case

Use the following functions to save, delete or load points in a kinematics:

- [Save Point on page 37](#)
- [Load Point on page 38](#)
- [Load Next Point on page 38](#)
- [Delete Points on page 38](#)

Points in a kinematics are not saved remanently, i.e. if the kinematics is deleted or if the control is turned off, the points are deleted.

In order to save points from a kinematics in a point file or load points from a point file in a kinematics, use the following commands:

- [Save Point File on page 38](#)
- [Load Point File on page 39](#)

In a point file, points are saved remanently. The point files are stored in the directory "<User-Partition> / PointFile" on the Compact Flash card.

To delete a point file, use the following command:

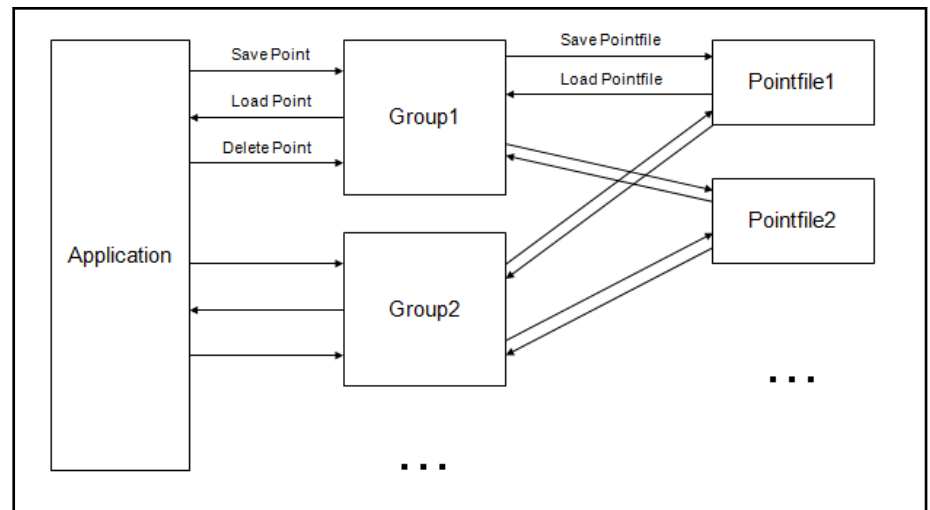
- [Delete Point File on page 39](#)



Points from any point file may be loaded into any kinematics.

Make sure that the selected kinematics is able to travel to the loaded points.

Functional Description



Application	User application
Group	Kinematics
Pointfile	Point file
Save Point	Function "Save point"
Load Point	Function "Load point" or "Load next point"
Delete Point	Function "Delete points"
Save Pointfile	Command "Save point file"
Load Pointfile	Command "Load point file"

Fig. 2-18: Overview of the "Teach points" functionality

2.7.2 Structure of the Point File

The point file is in ".txt" format.

One data set is saved per line.

A data set consists of the following main parts:

- Point name
- Coordinate values
- Belt values
- Coordinate system

The individual main parts of a data set are separated by a semicolon ";".

The main parts "coordinate values" and "belt values" each consist of 16 values. The values are separated by a comma ",".

Example:

A data set from the point file Dummy.txt

```
Point1;100.00000000,20.00000000,0.00000000,0.00000000,0.00000000
00,0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,0.00
000000,0.00000000,0.00000000,0.00000000,0.00000000,0.00000000
0,;0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,0.000
00000,0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,
0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,;PCS1;
```

2.7.3 Save Point

The function [Write Point on page 31](#) saves a point to a kinematics.

Use the option "Overwrite" to specify the behavior of the system if the point already exists in the kinematics:

- **Active:** The point existing in the kinematics is overwritten

Functional Description

- **Inactive:** The point existing in the kinematics is not overwritten
The option "Save belt values" specifies if the belt values are saved:
- **Active:** The belt values are saved to the kinematics
- **Inactive:** The belt values of the kinematics are assigned the value "0.0"

2.7.4 Load Point

The function [Read Point on page 31](#) loads a point from a kinematics.

2.7.5 Load Next Point

The function [Read Next Point on page 31](#) loads the next point from a kinematics.

If the specified point name exists in the kinematics, the next point in the kinematics is loaded.

If an empty string is specified as point name, the first point in the kinematics is loaded.

If the specified point name is the last point in a kinematics, an empty string is returned.

The points saved in the kinematics are sorted by character. It is irrelevant in what order the points are saved to the kinematics.

Example:

Sorted point names of a kinematics

1. Point1
2. Point10
3. Point2
4. point1

If point name "Point1" is specified, point "Point10" is loaded.

2.7.6 Delete Points

The function [Delete Points on page 31](#) deletes points of a kinematics.

The character "*" serves as a wildcard as the last character.

Example:

Point names of a kinematics

- Point11
- Point22
- point3

If point name "Point11" is specified, point "Point11" is deleted.

If point name "Point*" is specified, points "Point11" and "Point22" are deleted.

If point name "*" is specified, all points are deleted.

2.7.7 Save Point File

Execute the command [Save Point File on page 31](#) to save points from a kinematics in a point file.

When saving the point file, there are the following options:

- The selected point file is created if it does not exist yet
- If the selected point file already exists, there are the following options

- **New**
The contents of the existing point file are deleted.
The points of the kinematics are entered to the point file.
- **Add**
The contents of the existing point file remain unchanged.
Points that are not contained in the existing point file are added from the kinematics of the point file.
- **Add and overwrite**
Points contained in the existing point file and the kinematics file are overwritten with values from the kinematics.
The contents of the other points in the existing point file remain unchanged.
The other points of the kinematics are added to the point file.

2.7.8 Load Point File

Execute the command [Load Point File on page 31](#) to load points from a point file to a kinematics.

When loading the point file, there are the following options:

- **New**
The contents of the kinematics are deleted.
The points of the point file are entered to the kinematics.
- **Add**
The contents of the kinematics remain unchanged.
Points that are not contained in the kinematics are added from the point file.
- **Add and overwrite**
Points contained in the kinematics and the existing point file are overwritten with values from the point file.
The contents of the other points in the kinematics remain unchanged.
The other points of the existing point file are added to the kinematics.

2.7.9 Delete Point File

The function [Delete Point File on page 31](#) deletes a point file.

2.8 Blending

2.8.1 Blending, General Information

During coordinated kinematics movement, there are two ways of traveling to the end point of a motion command:

- **Traveling directly to the end point (TO):**
End point P1 is traveled to directly, the velocity is slowed to zero. Only then, the next motion command (P1→P2) is started.

Functional Description

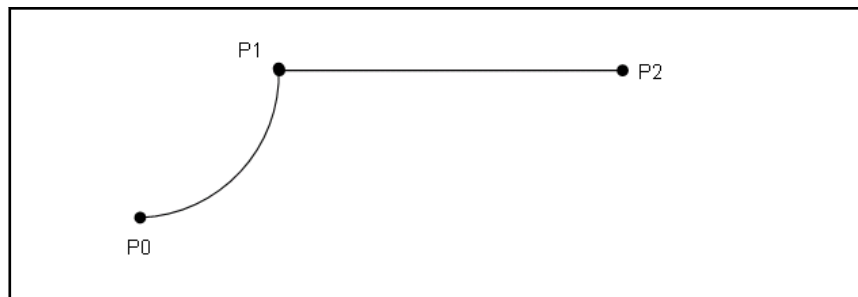


Fig.2-19: TO

- **Blending the end point (VIA):**

The end point P1 is blended with the next motion command. The kinematics does not reach the end point and leaves the programmed path to generate a continuous motion.

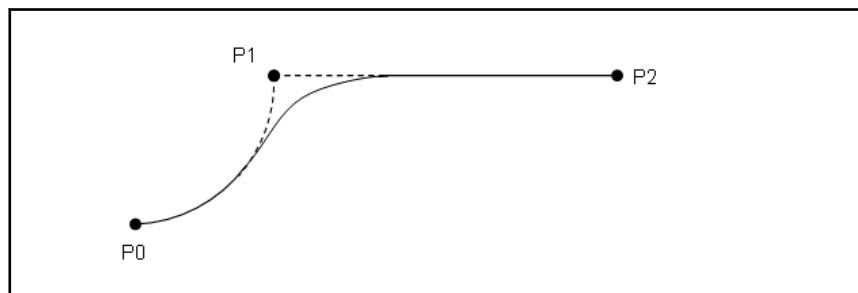


Fig.2-20: VIA

There are the following reasons to use blending:

- A continuous motion preserves the mechanical parts
- A continuous motion reduces vibrations at the gripper and the load
- The process cycle time is reduced
- It is easier to implement a constant velocity curve required by the application, e.g. to apply adhesive, for painting, etc.

2.8.2 Blending Radius

The programmed radius R_{prog}

The blending radius R specifies at what distance to the end point of a motion command the blending process begins.

With the PTP motion type, R_{prog} results from the programmed percentage multiplied by the distance in the ACS coordinate system.

With the Linear and Circular motion types, R_{prog} is specified as a spatial distance.

2.8.3 Slope Types

The selected slope type determines the behavior in the blending range.

Functional Description

Slope types	Advantages	Disadvantages
Block_Slope (early start)	Quick reactions are possible	Path velocity and acceleration result from the superimposed motion The path results from the superimposed motion The path does not depend on the maximum values of the velocity profile
Cont_Slope (spline)	Constant path velocity The path depends on the maximum values of the velocity profile	No quick reactions possible due to spline calculation

Tab.2-9: Overview of slope types

2.8.4 Block_Slope

The "Block_Slope" slope type implements blending by an early start of the next motion command.

Due to the early start, the motion commands blend. This results in a path with a path velocity and path acceleration in the blending range.

This path depends on the maximum values of the velocity profile of the motion commands.

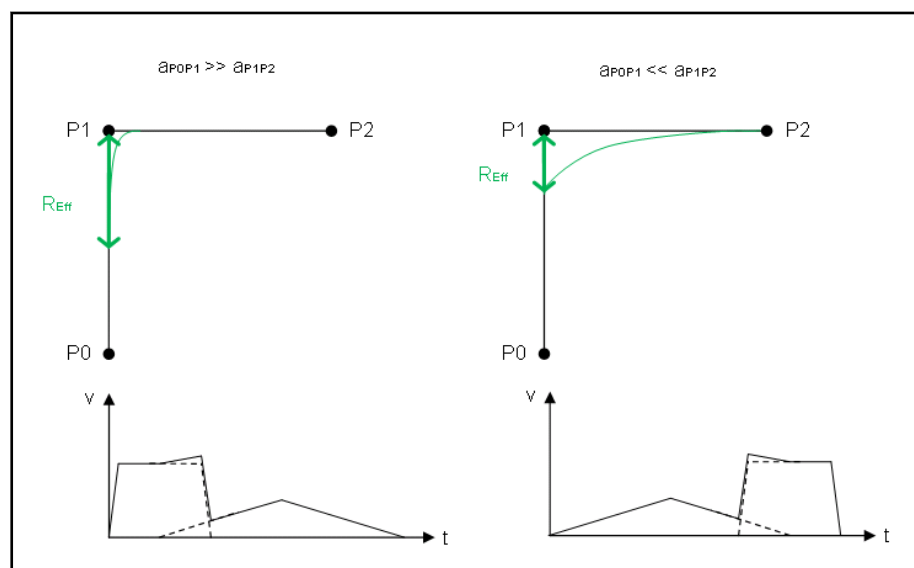


Fig.2-21: Path depending on the maximum values of the velocity profile

The effective radius R_{Eff}

With Block_Slope, the effective radius R_{Eff} is determined by the selected option in parameter K-0-0099, Kinematic configuration:

- **Blending without limitation:**

R_{Eff} is not limited. Therefore, R_{Eff} corresponds to the programmed radius R_{Prog} .

Functional Description

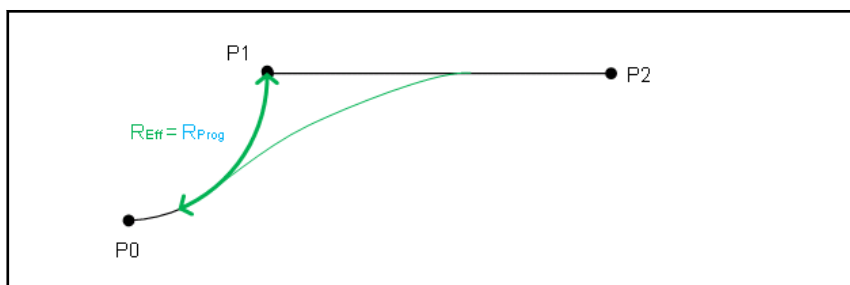


Fig.2-22: Blending without limitation

- **Blending with limitation to half the path distance:**

R_{Eff} is limited to the minimum of R_{Prog} , half the path distance from P0 to P1, and half the path distance from P1 to P2.

$$R_{Eff} = \min \left(R_{Prog}, \frac{1}{2} * \overline{P0P1}, \frac{1}{2} * \overline{P1P2} \right)$$

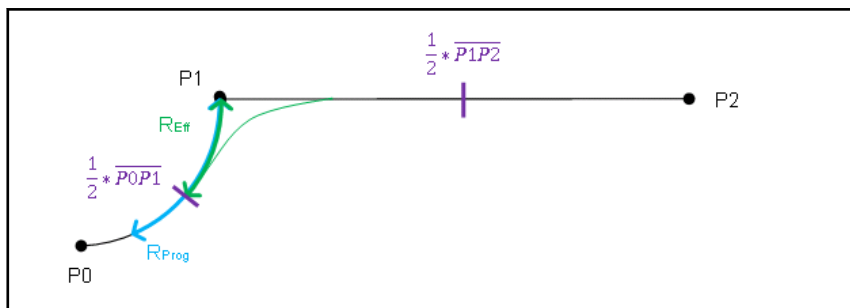
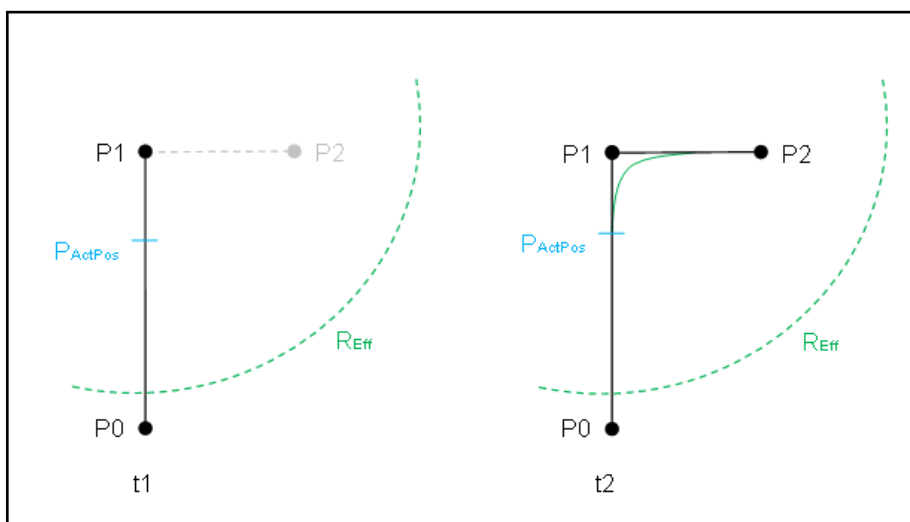


Fig.2-23: Blending with limitation to half the path distance

Late blending

If a motion command whose end point is traveled to using VIA is already active and another motion command is issued, both motion commands are blended late.

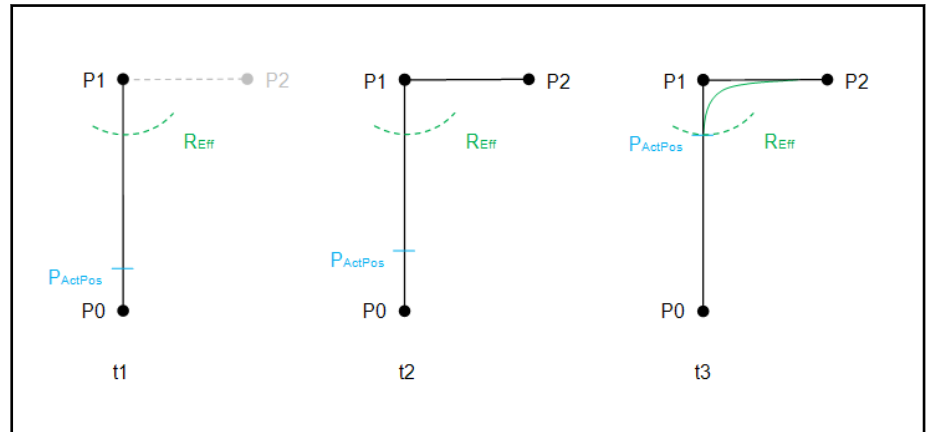
With late blending using Block_Slope, keep in mind that the second motion command needs to be prepared.



t1 At time t1, the motion command to P2 is issued
 t2 At time t2, the motion command to P2 is prepared
 Fig.2-24: Late blending with Block_Slope

Functional Description

If the second motion command is issued while the current position is not yet in the blending range R_{Eff} , blending only starts when R_{Eff} is entered.



- t1 At time t1, the motion command to P2 is issued
t2 At time t2, the motion command to P2 is prepared
t3 At time t3, the current position has entered R_{Eff}

Fig.2-25:

Maximum number of superimposed motions

The maximum number of superimposed motions depends on the option chosen in parameter K-0-0099:

- A maximum of 2 motion commands may be superimposed
- An unlimited number of motion commands may be superimposed

2.8.5 Cont_Slope

The "Cont_Slope" slope type implements blending by combining two motion commands using a spline.

The spline guarantees a constant velocity curve. The spline is assigned to the first of the two motion commands to be blended and adopts its maximum values of the velocity profile.

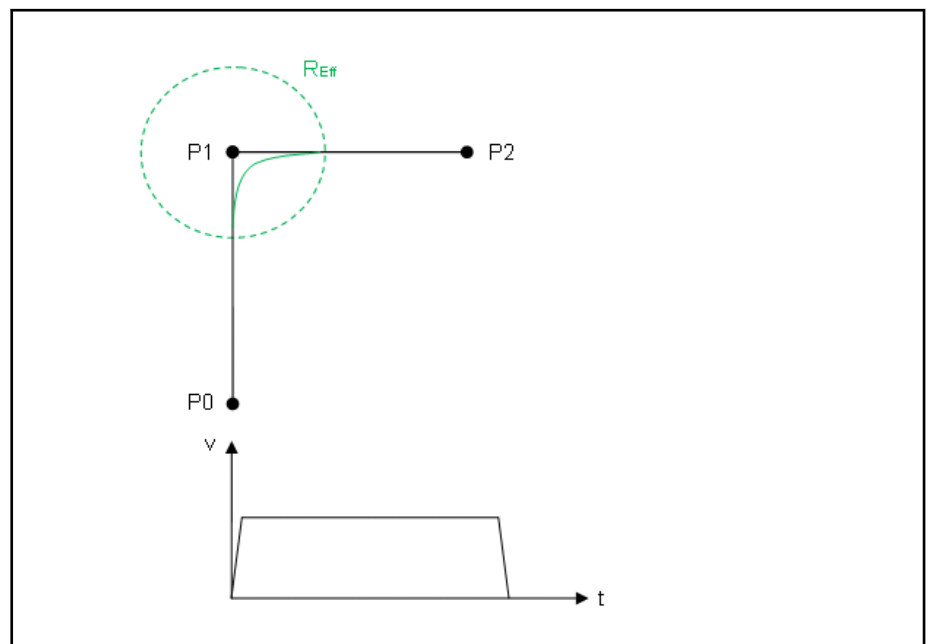


Fig.2-26: Blending with Cont_Slope

Functional Description

The effective radius R_{Eff} With Cont_Slope, the effective radius R_{Eff} corresponds to the programmed radius R_{Prog} .

R_{Eff} limitation with TO, VIA If a motion command is blended whose end point is traveled to using TO, R_{Eff} is not limited.

If a motion command is blended whose end point is traveled to using VIA, R_{Eff} is limited to half of the motion command distance. R_{Eff} is limited as an end point traveled to using VIA may also be blended.

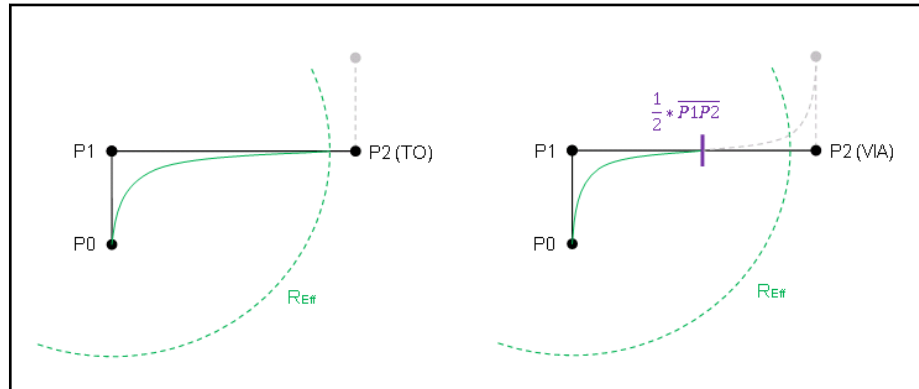
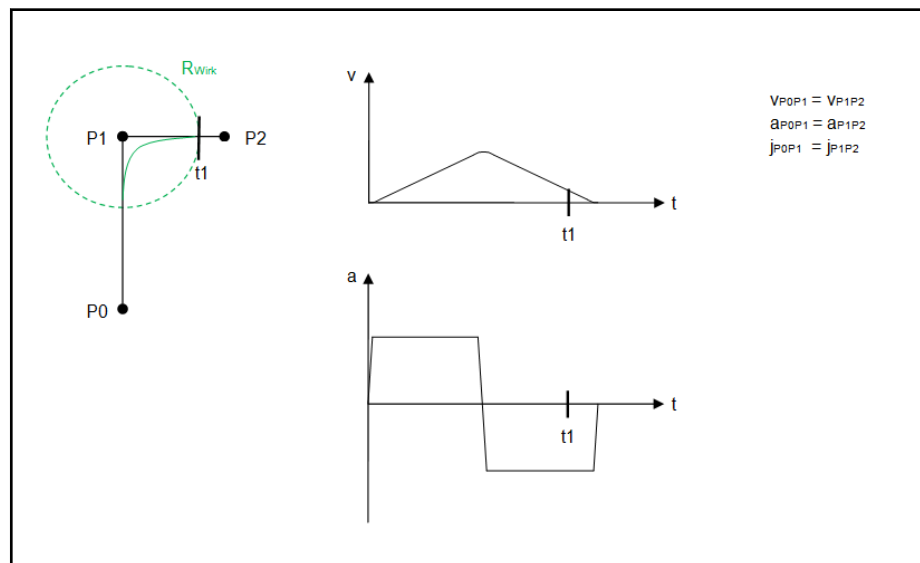


Fig.2-27: R_{Eff} limitation with TO, VIA

Transition accelerations If both motion commands have the same maximum values for the velocity profile, blending can have a transition acceleration at the transition point of the two commands.

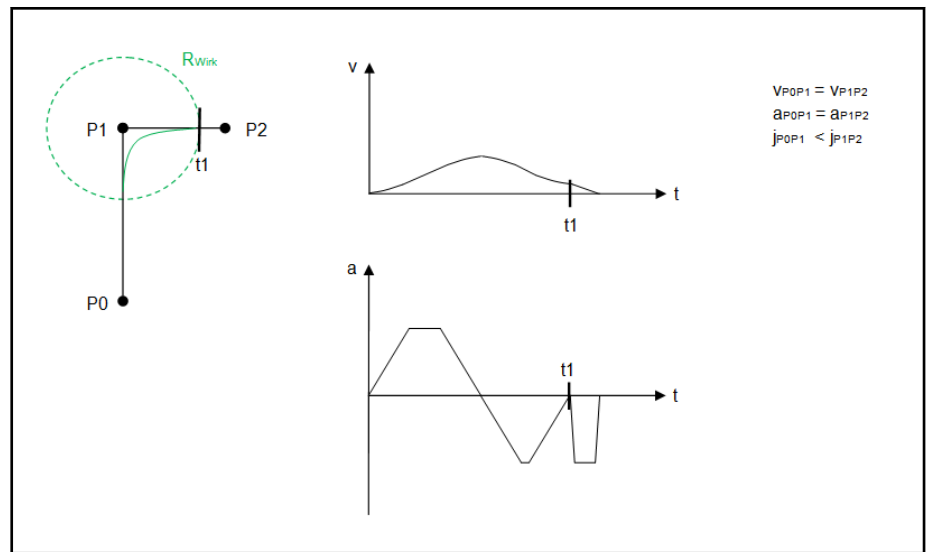


t1 Command transition

Fig.2-28: Transition acceleration - identical maximum values for the velocity profile

If both motion commands have different maximum values for the velocity profile, acceleration is set to zero at command transition.

Functional Description



t1 Command transition

Fig.2-29: Zero transition acceleration - different maximum values for jerk j

Late blending

If a motion command whose end point is traveled to using VIA is already active and another motion command is issued, both motion commands are blended late.

With late blending using Cont_Slope, keep in mind that the second motion command and the spline need to be prepared. This preparation time has to be compensated using a lead time set in parameter K-0-0250. The lead time has to be adapted to the respective application during commissioning.

The starting point of the spline results from the lead time multiplied by the maximum path velocity and added to the current position.

$$P_{StartSpline} = (t_{LeadTime} * v_{Max}) + P_{ActPos}$$

At time t1, the motion command to P2 is issued.

At time t2, the motion command to P2 is prepared.

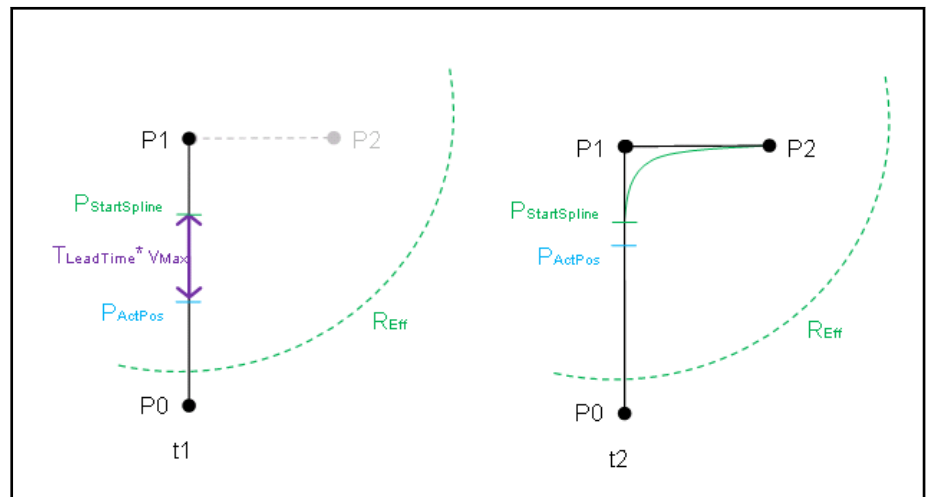


Fig.2-30: Late blending with Cont_Slope

If the starting point of the spline is already exceeded when the second motion command and the spline are prepared, blending with Cont_Slope has failed (see switching to Block_Slope).

Functional Description

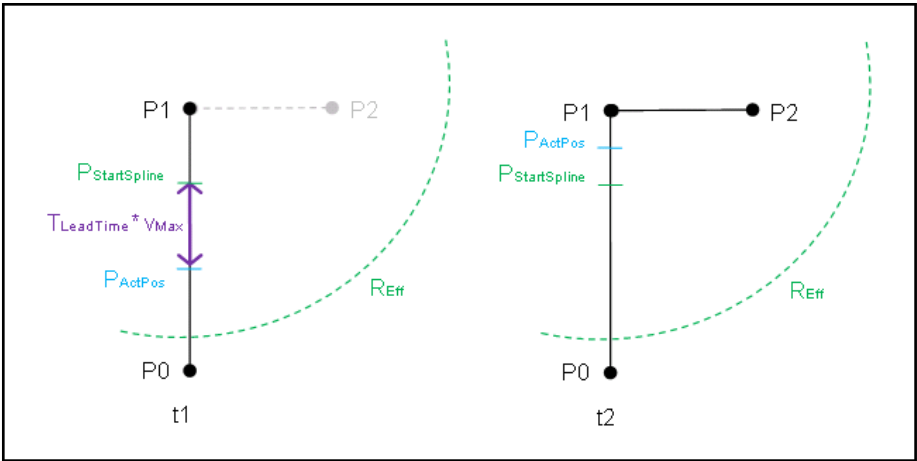


Fig.2-31: Late blending with Cont_Slope failed

- Switching to Block_Slope
- If blending with Cont_Slope failed, the following behavior depends on the option selected in parameter K-0-0099, Kinematic configuration:

 - The end point is traveled to using TO
 - Blending is executed with the slope type Block_Slope
- Spline calculation
- Parameter K-0-0021 (Kinematic status) indicates with a signal bit if a spline is being calculated. Another bit indicates if the spline is valid.

2.8.6 Blending Possibilities with Various Motion Types

The following tables indicate what motion types may be blended. If blending is not possible, the end point is traveled to using "TO".

Block_Slope

		2. Command		
		PTP	Linear	Circular
1. Command	PTP	x	-	-
	Linear	-	x	x
	Circular	-	x	x

Tab.2-10: Blending possibilities with Block_Slope

Cont_Slope

		2. Command		
		PTP	Linear	Circular
1. Command	PTP	-	-	-
	Linear	-	x	x
	Circular	-	x	x

Tab.2-11: Blending possibilities with Cont_Slope

2.9 Stopping, Interrupting, Continuing Motions

2.9.1 Stopping Motions

The command [Stop on page 29](#) stops all motions of a kinematics.

Functional Description

The stopping process has the following properties:

- The kinematics is stopped in a controlled way
- All active motion commands are stopped on the programmed path using the values specified for deceleration and jerk



If the specified values for acceleration and jerk are less than the respective values of the motion command, the values of the motion command are used for the stopping process

- All commands in the block buffer are aborted
- Any belt synchronization is desynchronized using the specified deceleration and jerk



If the specified deceleration or jerk is 0, the deceleration or jerk of the belt-specific error reaction are used (see [Belt-specific error reaction on page 50](#)).

- The contents of the block buffer are emptied

The following diagram illustrates the velocity curve in case of stop during a synchronization movement.

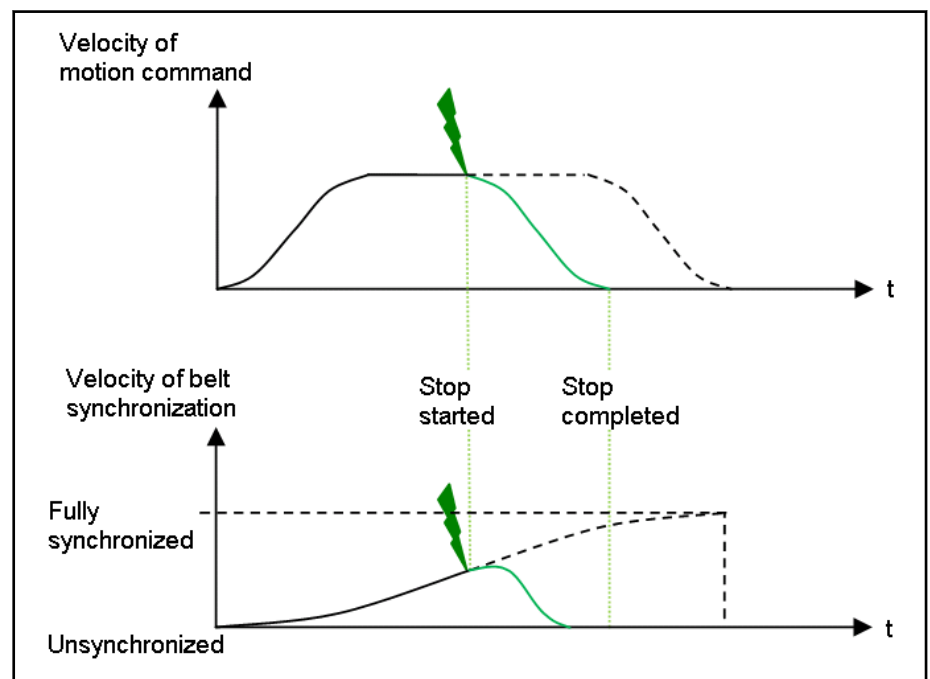


Fig.2-32: Velocity curve in case of stop

During the stopping process, the state machine of the kinematics is in the state "Stopping". If the stopping process is completed, the state machine is set to the state "Standstill Complete".

In the state "Stopping", no command may be issued.

The stopping process is terminated when:

1. The kinematics is in standstill
2. Stopping was terminated by the user

Functional Description

2.9.2 Interrupting Motions

The command [Interrupt on page 29](#) interrupts all motions of a kinematics.

The interrupt process has the following properties:

- The kinematics is stopped in a controlled way
- All active motion commands are stopped on their paths using the values specified for deceleration and jerk



If the specified values for acceleration and jerk are less than the respective values of the motion command, the values of the motion command are used for the stopping process

- Any existing belt synchronization is maintained
- The contents of the block buffer remain unchanged

The following diagram illustrates the velocity curve in case of interrupt during a synchronization movement.

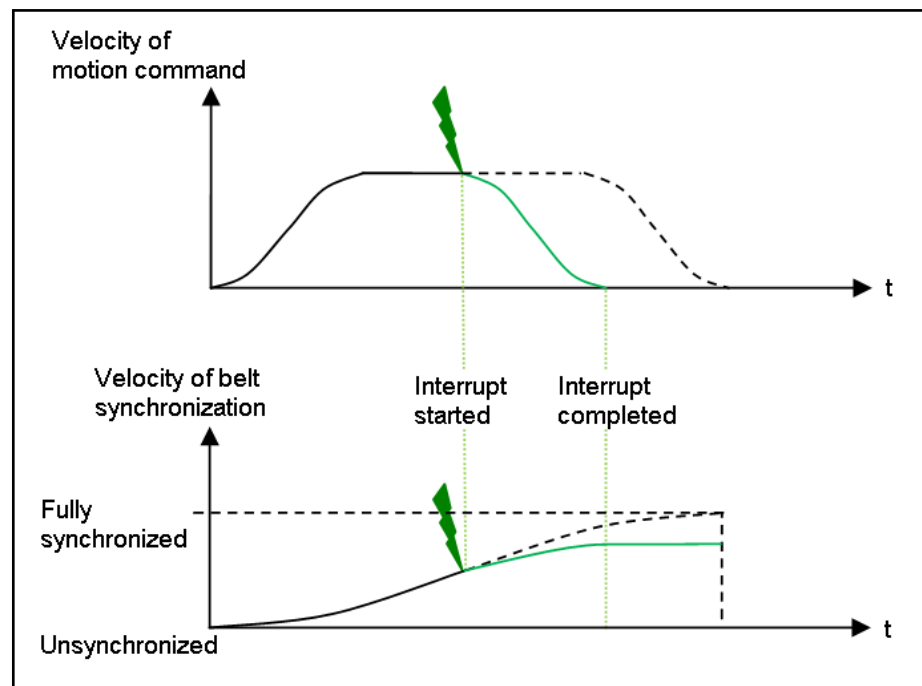


Fig.2-33: Velocity curve in case of interrupt

The state machine maintains its current state.

Additional commands may be issued during the interrupt process.

The interrupt process is terminated when:

1. The motion is continued using the command [Continue on page 29](#)
2. The motion is stopped using the command [Stop on page 29](#)

2.9.3 Continuing Motions

To restart a kinematics motion interrupted by the command [Interrupt on page 29](#), execute the command [Continue on page 29](#).

The continue process has the following properties:

- All motion commands are continued with their original deceleration and jerk
- The interrupt process is aborted

The state machine maintains its current state.

Additional commands may be issued during the continue process.

The continue process is terminated when:

1. The interrupt process is terminated
2. All motion commands continued their motions

2.9.4 Waiting

Use the command [Wait on page 29](#) to wait for a specified time between two other commands. The wait command delays execution of all following commands by the specified time. Therefore, it is possible to add other commands to the command buffer immediately after issuing the wait command. These added commands are started with a delay.



Two motion commands that are separated by a wait command are not blended.

2.10 Belt-synchronous Motion

2.10.1 Belt-synchronous Motion, General Information

With the "belt-synchronous" function, the kinematics motion can be synchronized to an assembly or conveyor belt with respect to position and orientation.

In this case, the belt can travel either forward or backward, change velocity or stop.

The belt has to be straight and may be positioned anywhere in the space.

A belt is represented by a PCS ([Product Coordinate System \(PCS\) on page 22](#)).

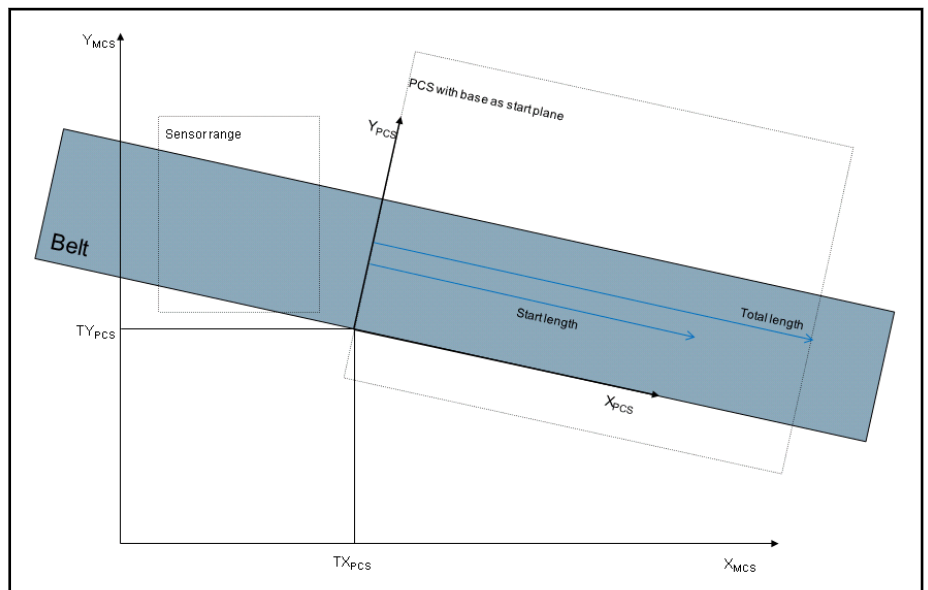


Fig.2-34: Overview belt-synchronous

Functional Description

The sensor range shown in the figure, can also be implemented as an individual product coordinate system.

The kinematics internally captures the belt motion.

Current belt values can be read via parameter K-0-0100 (Current position, Cartesian Machine Coordinate System) or the KinData interface.

The maximum number of available belts per kinematics is 16 minus the kinematic axes (positioning/orientation/auxiliary axes).

2.10.2 Belt Configuration

The belt configuration consists of the following:

- [Creating a belt in K-0-0008 on page 50](#)
- [Basic configuration on page 50](#)
- [Belt-specific error reaction on page 50](#)

Creating a belt in K-0-0008

A belt has to be created in parameter K-0-0008 (Axis configuration list), see [Adding a New Kinematics on page 65](#). Only then can it be used.

Basic configuration

A belt has to be configured once before belt-synchronous traveling.

For configuration use the command [Set Belt Configuration on page 30](#).

A PCS that is already configured is overwritten by the command [Set Belt Configuration on page 30](#). Therefore, product coordinate systems that do not represent belts have to be configured later than the belts.

Read out transformations for product coordinate systems that refer to a belt using the command [Get Trafo Values on page 30](#) and the respective PCS number. Read out the respective belt configuration using the command [Get Belt Configuration on page 30](#) and the respective axis reference.

The basic configuration consists of the following parts:

- **Spatial position:** This is defined by a [Product Coordinate System \(PCS\) on page 22](#). The position of the Y/Z-plane defines the start plane. The position of the X-axis defines the belt direction. The belts are numbered continuously, beginning with PCS1 (e.g. belt 4 corresponds to PCS 4).
- **Start Length:** Defines the last possible start of the synchronization motion.
- **Total Length:** Defines the end of the belt-synchronous working area.



The belt-synchronous working area is limited by the start plane and "Total Length". A kinematics can travel belt-synchronously only in this area.

"Start Length" in the belt-synchronous working area defines the last possible start of the synchronization motion. Choose this length so that the complete belt-synchronous motion including the desynchronization motion until "Total Length" is completed.

For spatial rotation of the belt only the degrees of freedom are supported that are also supported by the transformation, e.g. 3-axis delta transformation mechanically only supports rotation around the Z axis. The belt can therefore only be rotated around the Z axis.

Belt-specific error reaction

In addition to the basic configuration described above, it is possible to specify a belt-specific error reaction for each configured belt.

Use the command ["Set Belt Error Reaction on page 30"](#) to configure the error reaction.

Read out the configured error reaction using "[Get Belt Error Reaction on page 30](#)".



If no error reaction is defined with "Set Belt Error Reaction", "Desynchronization using limit values" is used as the belt error reaction. Therefore, in case of error the kinematics desynchronizes the belt using the specified values for deceleration (1000 mm/s²) and jerk (10000 mm/s³).

2.10.3 Belt Values and Points on the Belt

The position of a point on the belt is defined by its coordinates and the so-called belt value.

The belt value defines the absolute position of the point on the belt using any coordinate system at the time of point recording.

Internally, the recorded belt value is set off against the current belt value. This way it is possible to track points on the belt. For the displacement of a point to a point behind the start plane, the following general mapping rule is applied:

$$P_{x(n)}' = P_{x(n-1)\_PCS(k)} - B_{P(n)\_PCS(k)} + B_{ref}$$

$$P_{n\_PCS(k)}'(P_{x(n)}', P_y, \dots / B_{ref})$$

$P_{x(n)}'$	X-value of the point at time n moved to the current belt position
$P_{x(n-1)\_PCS(k)}$	X-value of the point at time (n-1) in product coordinate system k
B_{ref}	Belt value at the time of point recording
$B_{P(n)\_PCS(k)}$	Belt value of point P at the time in product coordinate system k
$P_{n\_PCS(k)}$	Point behind the start plane at time n in the product coordinate system k

Fig.2-35: Mapping of point P at time n-1 to a point behind the start plane at time n

Points are automatically tracked by the control during synchronization and synchronous traveling. In addition, it is possible to track specified points on the belt using the function [Move Point on page 30](#), e.g. for display purposes.

If the end point of a synchronizing or fully synchronous motion command is located in front of the start plane, the kinematics waits until the end point passes the start plane and then travels to the end point synchronously on the belt.



Different coordinates have to be used for synchronization and desynchronization.

For example, it is not possible to use (100 / 100 / 0) with belt value 10 for synchronization and next (100 / 100 / 0) with belt value 0 for desynchronization.

When the kinematics prepares the block, it calculates a traversing command with distance "0". The result is that the command is not processed, but marked as done right away.

As the belt is always aligned along the X-coordinate of a belt PCS, only the X-coordinate of the point is moved in the belt PCS to move the point on the belt.

The belt has a modulo value that is independent of the belt axis. By default, the belt value is modulated at 10000.

Functional Description

In parameter K-0-0099 (Kinematic configuration) the belt modulo value may be set to equal the axis modulo value, as an option.



The "Total Length" always has to be less or equal to half the belt modulo value. This condition guarantees safe detection of the belt modulo overflow.

2.10.4 Belt Operation

Running operation is divided into the following phases:

- [Synchronization on page 52](#)
- [Traveling fully synchronously on page 53](#)
- [Desynchronization on page 53](#)

The following overview lists possible motion commands in the individual phases:

Reference description	Synchronization	Traveling fully synchronously	Desynchronization
Move Direct Absolute	x	x	x
Move Direct Relative		x	x
Move Linear Absolute	x	x	x
Move Linear Relative		x	x
Move Circular Absolute	x	x	x
Move Circular Relative		x	x
Synchronization with Limits	x		
Desynchronization with Limits			x

Tab.2-12: Overview of the command possibilities for belt-synchronous operation

Synchronization

For synchronization using [Move Linear Absolute on page 29](#), [Move Direct Absolute on page 29](#) or [Move Circular Absolute on page 29](#), simply enter a corresponding belt value into the commanded point at the belt axis position.

As a rule, synchronization only takes place when the programmed target point (the commanded point is moved internally with the belt value) is inside the belt-synchronous working area. If this is not the case, the kinematics waits at the last commanded end position.

When the commanded point is reached, the kinematics is fully synchronous, i.e. the belt velocity is determined by the kinematics using the block distance. The greater the block distance the smaller the exceedances of the block limit values during the synchronization command. In reverse, the kinematics creates the belt velocity in jumps with very small distances, e.g. belt distance close to "0".



Points that are still outside the start plane when the command is issued, are internally moved to the start plane.

If the synchronization point moved to the start plane corresponds to the waiting position of the kinematics, the belt velocity is generated in jumps as described above.

In this case, the synchronization distance approaches "0".

Functional Description

For this reason, synchronization with maximum values for acceleration and jerk [Synchronization with Limits on page 29](#) is possible.

Here, no specific synchronization end point is specified, but the maximum values for acceleration and jerk are used to create the respective belt velocity in belt direction in a controlled way.

If the TCP is outside the belt-synchronous working area, synchronization is aborted. The end position on the belt is only known after synchronization. In a second step, the position on the belt may then be traveled to synchronously.



If the selected maximum values for the synchronization mode "Synchronization with Limits" are very small, the synchronization time increases. With a lot of belt dynamics, the kinematics may not be able to synchronize due to the small maximum values.

Traveling fully synchronously

Fully synchronous traveling using [Move Linear Absolute on page 29](#), [Move Direct Absolute on page 29](#) or [Move Circular Absolute on page 29](#) works in the same way as described above in "Synchronization".

In addition, it is possible to travel by a relative distance on the belt using [Move Linear Relative on page 29](#), [Move Direct Relative on page 29](#) or [Move Circular Relative on page 29](#).

Instead of a concrete end point a distance is specified that is to be traveled in relation to the belt. To implement this behavior, specify a belt value in addition to the commanded distance. The specified belt value is added to the PCS X-coordinate of the respective belt, i.e. if the belt value is "50", the kinematics travels by "50" in belt direction.

Desynchronization

For desynchronization using [Move Linear Absolute on page 29](#), [Move Direct Absolute on page 29](#) or [Move Circular Absolute on page 29](#), enter the belt value "0" at the respective location.

Enter the belt value "1000000" at the position of the belt axis and issue the command [Move Linear Relative on page 29](#), [Move Direct Relative on page 29](#) or [Move Circular Relative on page 29](#) for relative desynchronization. This means, that the kinematic travels from its current position by the distance specified in the point definition and is desynchronized afterwards.

With this procedure, the absolute spatial end point will only be known when half the distance was traveled.

Furthermore, desynchronization may be executed using the command [Desynchronization with Limits on page 29](#). Here, specify maximum values for jerk and deceleration instead of a concrete desynchronization end point.

These maximum values are used to decrease the respective belt velocity in a controlled way.

2.10.5 Blending and Belt Synchronization

Blending of the individual motion blocks is guaranteed in all cases as described in [Blending on page 39](#). The only exception is the command [Synchronization with Limits on page 29](#).

Due to the belt dynamics, it is not possible to predict the exact termination of the synchronization motion (end point). Therefore, the start point of the following motion will only be known and the following block can only be started, when full synchronization with the respective belt is reached and "Synchronization with Limits" is terminated.

In this case, the kinematics stops for one cycle on the belt, i.e. for one cycle the kinematics travels only at belt velocity. Belt-synchronous traveling is not linked to a certain type of blending but independent of the motion criteria.

Functional Description

2.10.6 Summary

A belt is represented by a **PCS** ([Product Coordinate System \(PCS\) on page 22](#)) which may be offset and rotated in relation to the **MCS** ([Machine Coordinate System \(MCS\) on page 22](#)).

For an application example, see [Application example 2: Belt Synchronization on page 316](#).

A point on the belt is defined by its coordinates and the belt value. The belt value defines the absolute position of the point on the belt using any coordinate system at the time of point recording.

Blending individual motion blocks during belt-synchronous traveling is guaranteed in all cases.

The only exception is [Synchronization with Limits on page 29](#). Due to the belt dynamics, it is not possible to predict the exact termination of the synchronization motion (end point).

Each belt has a belt-specific error reaction, see [Belt-specific Error Reaction on page 60](#).

2.11 Switching on Path

Functional description

In many use cases it is required to capture path-dependent events and to trigger switching processes depending on them.

In the following, the functional principle is described with and without lead time.

Functional Description

Functional Principle

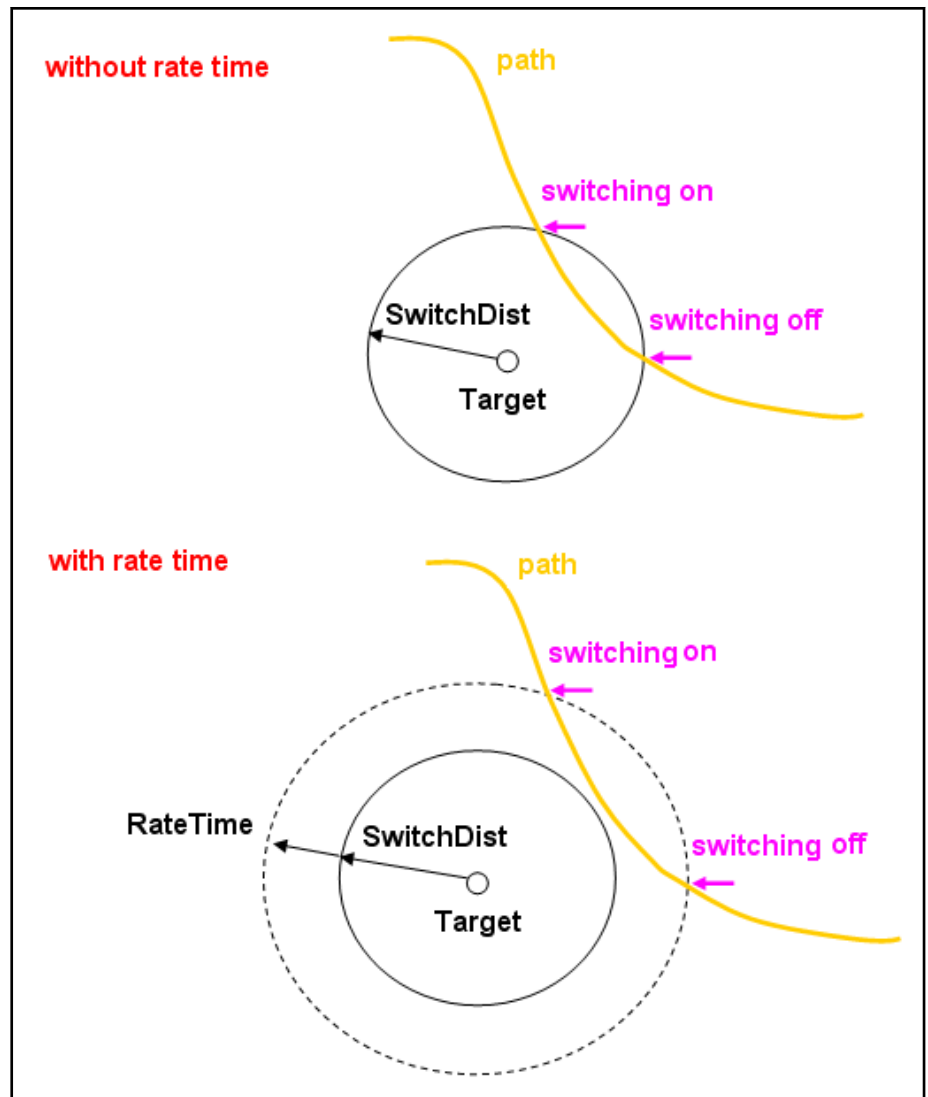


Fig.2-36: Functional Principle

The switching point is specified by "Target" and "SwitchDist" (sphere around "Target"). A rate time can be set via "RateTime".

This time increases the sphere around "Target" depending on the current kinematic velocity.

If the kinematics can reach the sphere around "Target" with the current velocity in the specified "RateTime", "InRange" is TRUE.

Functional Description

Example

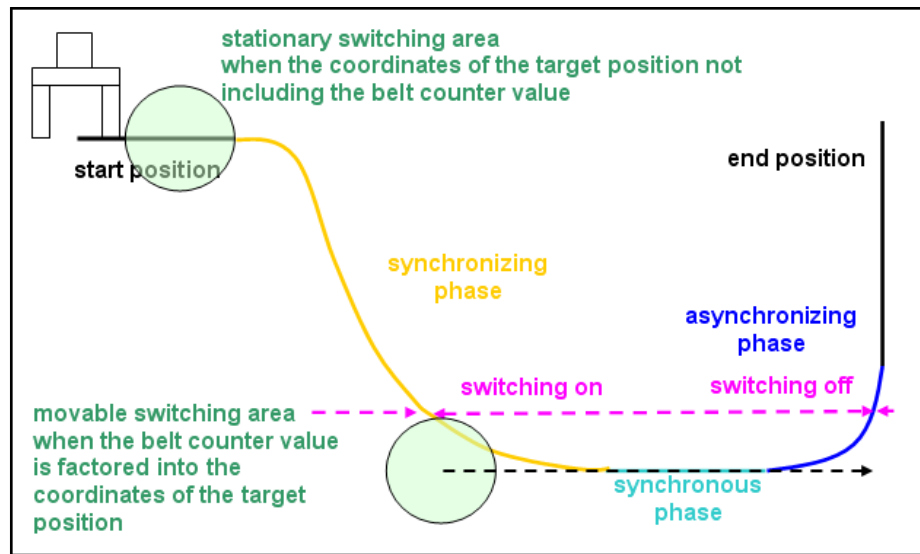


Fig.2-37: Example

If the coordinates of "Target" are specified without the belt values, the switching range of the function block is a static sphere.

If a belt value is specified, the sphere is guided in belt direction around the target point depending on the current belt counter value.

Time response

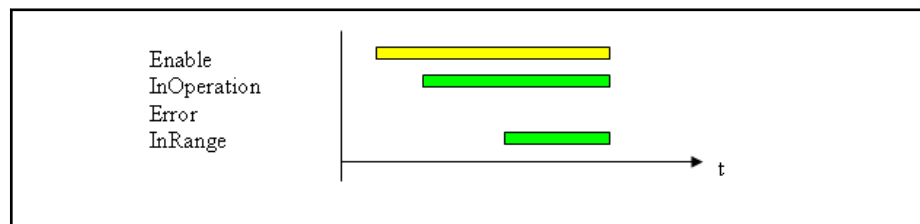


Fig.2-38: Time response

The function block [ML_SwitchOnPath](#) on page 162 starts to process its logic in a state-controlled way with a rising edge at "Enable".

After all prepared commands have been executed, the function block switches to the "InOperation" state. Changes at "SwitchDist" and "RateTime" are applied as long as "Enable" is set to TRUE.

The "InRange" output is set if the kinematic position is in the radius of the sphere and if the "RateTime" is met. Otherwise, it is reset.

If problems occur, the "Error" state is assumed and the outputs "InRange" and "InOperation" are reset. The outputs "ErrorID" and "ErrorIdent" contain more detailed information on the error.

If "Enable" is reset, "InRange", "InOperation" and "Error" are reset and the function block waits in its initial state at a rising edge at "Enable".

Switching point

If the TCP of the kinematics is on a position smaller than or equal to "SwitchDist", the function block state is set to "InRange".

Lead time

"RateTime" determines the number of milliseconds to be set before reaching the actual switching position of the "InRange" output. The current velocity is read from the Global Variable List and the remaining time up to the switching point is calculated (without considering acceleration, deceleration and jerk).

If the remaining time is smaller or equal to the lead time "RateTime", the "InRange" output is immediately set. Otherwise, the current position is read in again and it is calculated again.

Functional Description

Application notes The function block only operates correctly if realistic values are specified for the inputs "SwitchDist" and "RateTime":

- With a PLC cycle time of the motion task of 10 ms, it makes no sense to set lead times of less than 10 ms.
- "SwitchDist" during high velocities has to be high enough to switch on time.

At a velocity of 5000 mm/s and a cycle time of 10 ms, the function block can calculate the remaining distances and remaining times every 50 mm up to the switching point. Therefore, the distance and the lead time have to be high enough or the cycle time has to be reduced.

2.12 Traversing Range Limits

The kinematics supports various traversing range limits.

The following figure shows an overview of all possible traversing range limits and their effectiveness.

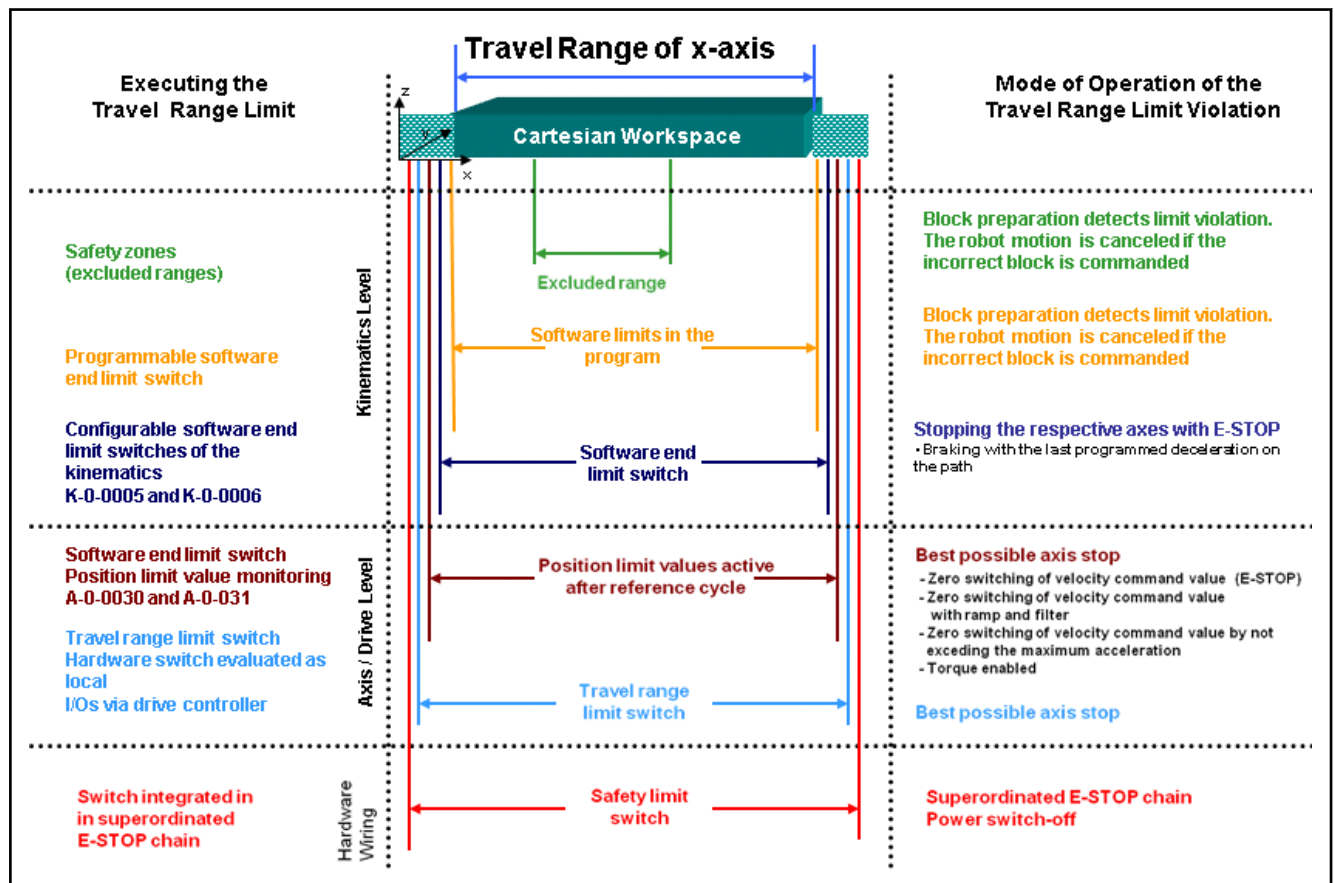


Fig.2-39: Overview of traversing range limits

The Robot Control V1 concept "Configurable safe zones" K-0-0156/K-0-0157 safe zone1 min/max point to K-0-0218/ K-0-0219 safe zone32 min/max point was replaced in Robot Control V2 by the concept "Programmable safe zones".

Programmable safe zones

Programmable safe zones are defined by the command [Set Safe Zone on page 30](#).

32 safe zones are available.

Functional Description

A safe zone is determined by two points in the machine coordinate system that form a cuboid. The area **inside** this cuboid constitutes the safe zone. The limits of the safe zone, such as the minimum point S1 (0,0,0), are also part of the safe zone.

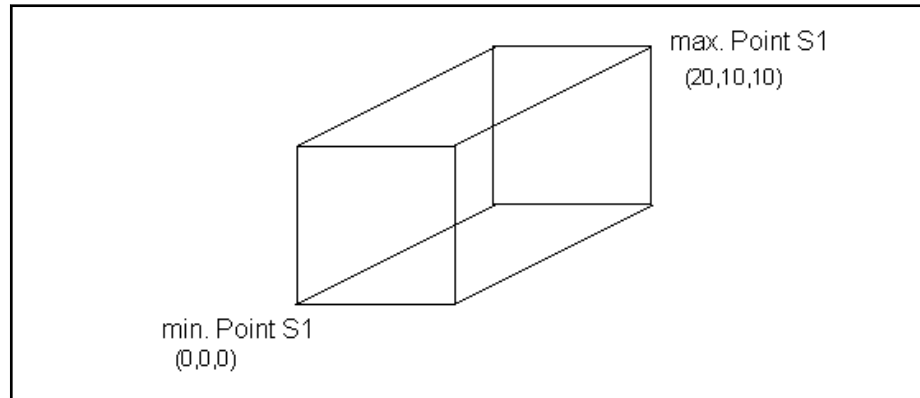


Fig.2-40: Maximum and minimum point of a safe zone

The safe zone is checked during block preparation when a new motion command is issued. The system checks if the linear distance between starting point (A) and end point (B) of the motion has a point of intersection or contact with one side of the safe zone.

If the block preparation detects a violation of a safe zone, error message F12C0433, Workspace violation is returned.

The contact check between the straight line and the safe zone is not carried out if the starting point of the straight line is inside the safe zone. For example, if the TCP position is located on the boundary of the safe zone, the TCP of the kinematics is inside the defined safe zone. Therefore, the kinematics can travel to a point outside the safe zone without a safe zone violation being reported.

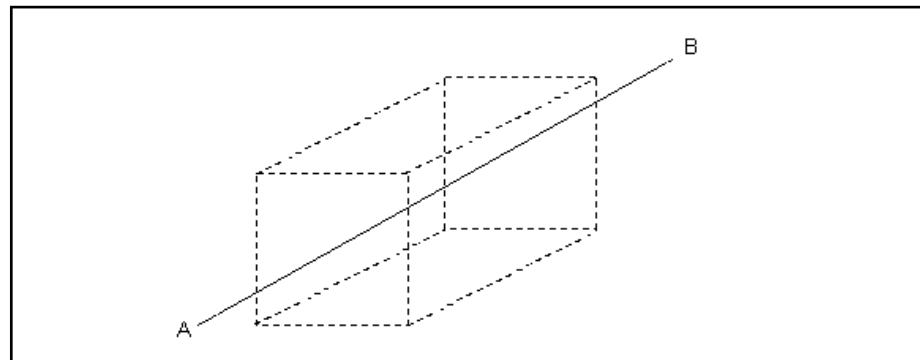


Fig.2-41: Safe zone with motion from A to B

Regarding the effectiveness of the safe zone, the following has to be considered:

- Each kinematics only considers its individual safe zones.
- The safe zones consider only the TCP of the kinematics, not the mechanics.
- In case of the PTP interpolation, the monitoring of the safe zones is strictly restricted since the distance to be traveled is not defined. Deviations from the linear distance AB are not detected.
- In case of a CIRCULAR interpolation, the monitoring of the safe zones is strictly restricted. The safe zone check is limited to the linear distance

Functional Description

between the starting and intermediate point as well as between intermediate and end point. The circle segment is not checked.

- When blending with VIA or APPROX, deviations from the linear distance AB are not detected. Deviations due to blending radii are not considered.
- The maximum point always has to be larger than the minimum point, since the safe zone check otherwise becomes inactive.
- Since the safe zones are checked in the block preparation, safe zones cannot be dynamically adapted. If the safe zones are changed, they are only effective for non-prepared motion commands. We therefore recommend programming the programmable safe zones prior to starting the motion command.

Programmable software limit switches

Programmable software limit switches are defined by the command [Set Safe Zone on page 30](#).

They work in the same way as safe zones, with one exception: the minimum point and maximum point form a cuboid that represents the permissible area of a kinematics.

Error F12C0885, Workspace out of range is returned.

Configurable software limit switches

Configurable software limit switches are set using parameters K-0-0005, Software limit switch, positive, MCS and K-0-0006, Software limit switch, negative, MCS.

They act like programmable software limit switches. In addition, the commanded kinematic position is checked cyclically. If it is outside the configurable software limit switch, error message F12C0885, Workspace out of range is returned.

Outside the configurable software limit switch, only such position commands cause errors that move the kinematics further away from the working area.

Each coordinate of the machine coordinate system (MCS) is checked individually:

For example, if the kinematics is above the configured software limit switch on the X- and Y-axis, only motions in the direction of lower X- and Y-values are permitted. Motions in the direction of smaller X-values but greater Y-values result in an error.

2.13 Errors and Error Reactions

2.13.1 Errors and Error Reactions, General Information

Several messages, warnings and errors may be returned.

(Also refer to [Diagnostics, General Information on page 351](#)).

The following is supported:

Error abbreviation	Description	Error reaction
A0xxxxxx	Message	None
E0xxxxxx	Warning	None
F0xxxxxx	F0 error	None
Fyxxxxxx (y > 0)	Error with error reaction in the kinematics	Error reaction in the kinematics

Tab.2-13: Diagnostics and error reactions

Functional Description

The error reaction of the kinematics consists of the following parts:

- [Error Reaction of the Commanded Motion on page 60](#)
- [Belt-specific Error Reaction on page 60](#)

The error reaction of the kinematics is completed if both parts of the error reaction are completed.

While a kinematics executes an error reaction, it is in state "Errorstop Stopping" (see [State Diagram of a Kinematics on page 26](#)). No kinematics commands can be executed in this state.

When the error reaction is completed, all uncompleted motion commands are aborted (see [Motion commands on page 28](#)).

2.13.2 Error Reaction of the Commanded Motion

The error reaction of the kinematics is stopping on the programmed path, if possible. The values for deceleration and jerk can be configured in parameter "K-0-0223, Selection of error reaction":

- **Use of the command values:** The maximum values specified at the respective command are used for stopping (see [Motion commands on page 28](#))
- **Use of the emergency deceleration and the emergency jerk:** Emergency halt deceleration (K-0-0224) and Emergency halt jerk (K-0-0225) are used

The following applies: The greater values for emergency deceleration and commanded deceleration and/or emergency jerk and commanded jerk are used.

2.13.3 Belt-specific Error Reaction

In addition to the error reaction of the overall kinematics, a specific error reaction for all configured belts may be enabled. This is configured via [Set Belt Error Reaction on page 30](#). If a belt error reaction is configured, it will always be carried out in the direction of the belt and independently of the kinematics error reaction. All in all there are three possible reactions for each belt:

Belt-specific error reaction options	Description
None	Only the kinematics error reaction is active. In case of an error, synchronization with the respective belt is maintained, if any.
Desynchronization over time	In case an error occurs, desynchronization takes places using the configured time.
Desynchronization using limit values	In case of an error, desynchronization is carried out using the configured maximum values for deceleration and jerk

Tab.2-14: Belt-specific error reaction options



If a belt error reaction is configured, it will always be carried out in the direction of the belt and independently of the kinematics error reaction.

The duration of the belt error reaction may vary according to the selected parameters.

2.14 Cyclical Command Value Specification

2.14.1 Cyclical Command Value Specification, General Information

In robotics it is currently possible, to move the Tool Center Point (TCP) block by block to points in the space (block operation). Furthermore, it is possible to specify spatial command position values cyclically (in every motion cycle) (cyclical operation). It is possible to switch flexibly between the two operation modes without stopping the kinematics.

Using cyclical command value specification, a kinematics can be cyclically assigned a new position, velocity or distance command value.

Use cams, FlexProfile or other kinematics to generate the command value.

2.14.2 Start and Operation of the Cyclical Command Value Specification

The cyclical command value specification is started using [Open Cyclic Position on page 29](#).

Blending the cyclical command value range is possible (see [fig. 2-43 "Schematic illustration of the cyclical command value specification with blending" on page 62](#)) if a travel command is still active when the cyclical channel is opened and both commands do not explicitly prohibit blending with TO. In this case, the cyclical command value specification has to start at point P1 when blending starts at point P1.

Furthermore, [fig. 2-42 "Schematic illustration of the cyclical command value specification without blending" on page 61](#) explains cyclical command value specification without blending.

Then, once per motion cycle a new kinematics command value **has** to be set using the function [Write Cyclic Position on page 29](#).

The command value may be set in any coordinate system.

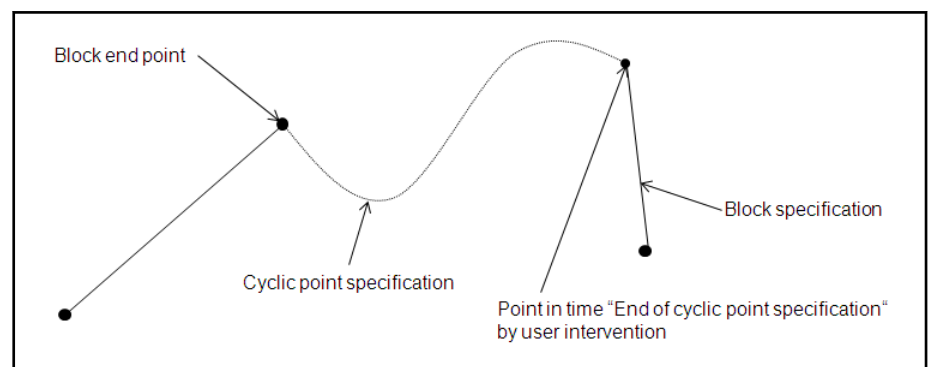


Fig. 2-42: Schematic illustration of the cyclical command value specification without blending

Functional Description

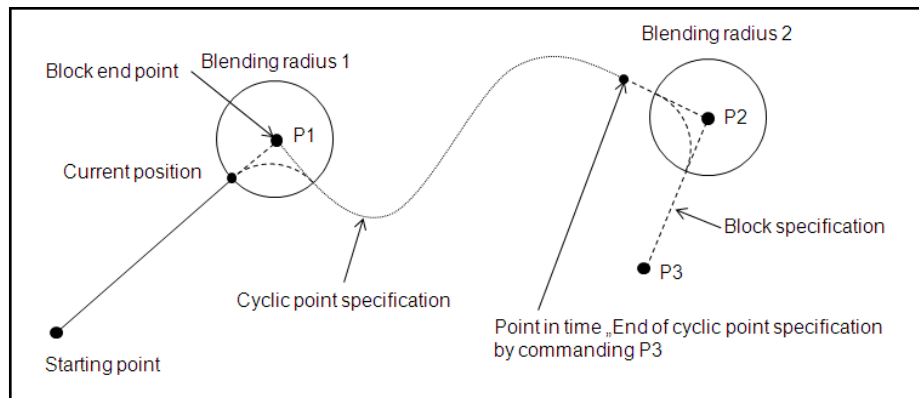


Fig.2-43: Schematic illustration of the cyclical command value specification with blending

2.14.3 Cyclical Command Value Specification

In the cyclical command value channel, the command value may be specified in three different formats:

Absolute position command value mode

The cyclic command value specification is an absolute position.

Relative position command value mode

The cyclic command value specification is a relative distance to the last commanded position in the last cycle.

Velocity command value mode

The cyclic command value specification is a velocity. The specified velocity is internally converted into a position difference that is added relatively to the last commanded position.

The cyclical command value specification is to be initialized with the last commanded block preparation position, which may be read out with the function [Read Cyclic Position on page 30](#).

Based on this, the command values can be specified cyclically (see [List of all motion commands and functions on page 29](#)).

NOTICE

The kinematics follows the command value!

Specified command value jumps may cause errors if the kinematics is not able to follow the command value, e.g. due to insufficient axis dynamics.

2.14.4 Termination of Cyclic Command Value Specification

The cyclical command value specification is terminated when a new travel command is issued.

This travel command may also be blended, see [fig. 2-43 "Schematic illustration of the cyclical command value specification with blending" on page 62](#). The velocity reached with the cyclical command value specification is then decreased in the direction of the last, cyclically specified motion vector with the previously defined maximum values. The entire distance required to decrease the last cyclically increased velocity is calculated. With this distance, a point in the space may be determined that is the starting point for the next travel command, see point P2 in [fig. 2-43 "Schematic illustration of the cyclical command value specification with blending" on page 62](#). Therefore, blending the following travel commands is always possible using early command start.



If the kinematics is already decelerating due to the specified command values and if the jerk and deceleration values are greater than the values for opening the cyclical channel, the previously defined values may be adapted under certain circumstances.

In particular if the selected jerk is very small, it is not sufficient to decrease the great current deceleration and reach velocity "0". As a result, the kinematics would travel backwards.

By adapting the jerk internally, it is guaranteed that any deceleration when leaving command value specification is always reduced.

2.15 Performance Measurement

The performance measurement of the kinematics allows for the measurement of the duration and number of calls of specific areas of the kinematic core. This information may be used to optimize applications.

The performance measurement is started and stopped by function [Set Performance Enable on page 30](#).

When a new measurement starts, the data recorded before is deleted.

Read the results using function [Get Performance on page 30](#), which returns a list of performance measurements.

Each list entry has the following structure:

Name	Unit	Description
name	Text	Name of the performance measurement
count	Number	Number of measured values
min	µs	Minimum measured value
max	µs	Maximum measured value
avg	µs	Average measured value

Tab.2-15: Performance measurement entries

The following measurements are available (excerpt):

Name of the entry	Description
Shared memory	In the PLC there is a Shared Memory for each kinematics KinData[] - Kinematic Data in the Shared Memory on page 109 This shared memory is operated in each motion cycle
Adapt	Actual value detection of the kinematics
Execute	Command value generation of the kinematics
SetCmd	Commands and functions issued by the user to the core (see Robot Control, Commands and Functions, Overview on page 27)
GetCmd	Reading status information of commands (see Command State on page 28)
prepare	Block preparation
MoveExecute	Cyclical execution of a move command

Tab.2-16: Available measurements (excerpt)

3 Interface Functions

3.1 Menu Structure of the Kinematics

3.1.1 Menu Item: Robot

By selecting the "Robot" node in the project, the menu item **Robot** is enabled in the IndraWorks main menu.

This menu item contains the following sub-menu items:

- **Add** - Use this item to add a kinematics or the "Global Data" node if it does not exist yet
- **Parameters** - Use this item to call the parameter export and import for all kinematics
- **Export** - Use this item to export the kinematic configuration
- **Import** - Use this item to import the exported kinematic configuration
- **Compare** - Use this item to compare two projects

The same sub-menu items are also available in the context menu of the "Robot" node.

Here, there are the following additional sub-menu items:

- **Find element** - Use this menu item to search within the kinematic configuration
- **New view** - Use this menu item to display the kinematic configuration in a separate Project Explorer window

3.1.2 Adding a New Kinematics

Go to the menu item **Add** in the "Robot" node to create a new kinematics.

A wizard guides the user through creating a new kinematics:

Step 1: Entering device name, comments and the author

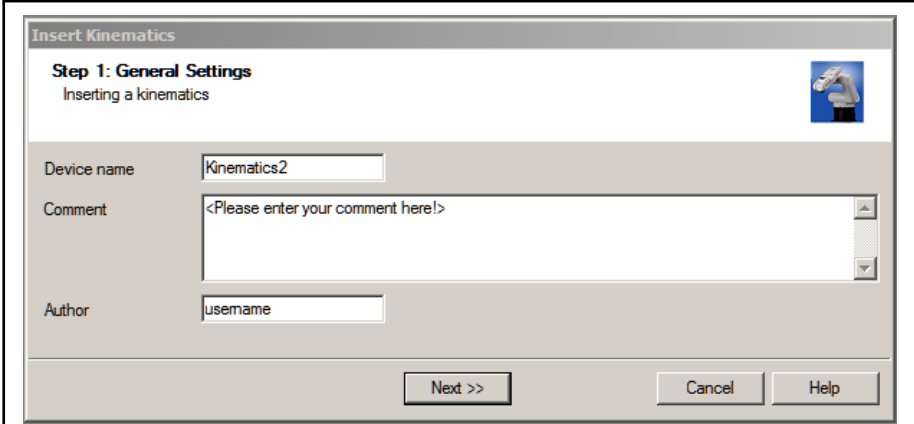
The image shows a software wizard window titled "Insert Kinematics". The current step is "Step 1: General Settings" with the subtitle "Inserting a kinematics". There is a small icon of a horse in the top right corner. The window contains three input fields: "Device name" with the text "Kinematics2", "Comment" with the placeholder text "<Please enter your comment here!>", and "Author" with the text "username". At the bottom right, there are three buttons: "Next >>", "Cancel", and "Help".

Fig. 3-1: Creating the kinematics, step 1

Step 2: Entering the Robot Control version, the kinematic number and the kinematic type

Interface Functions

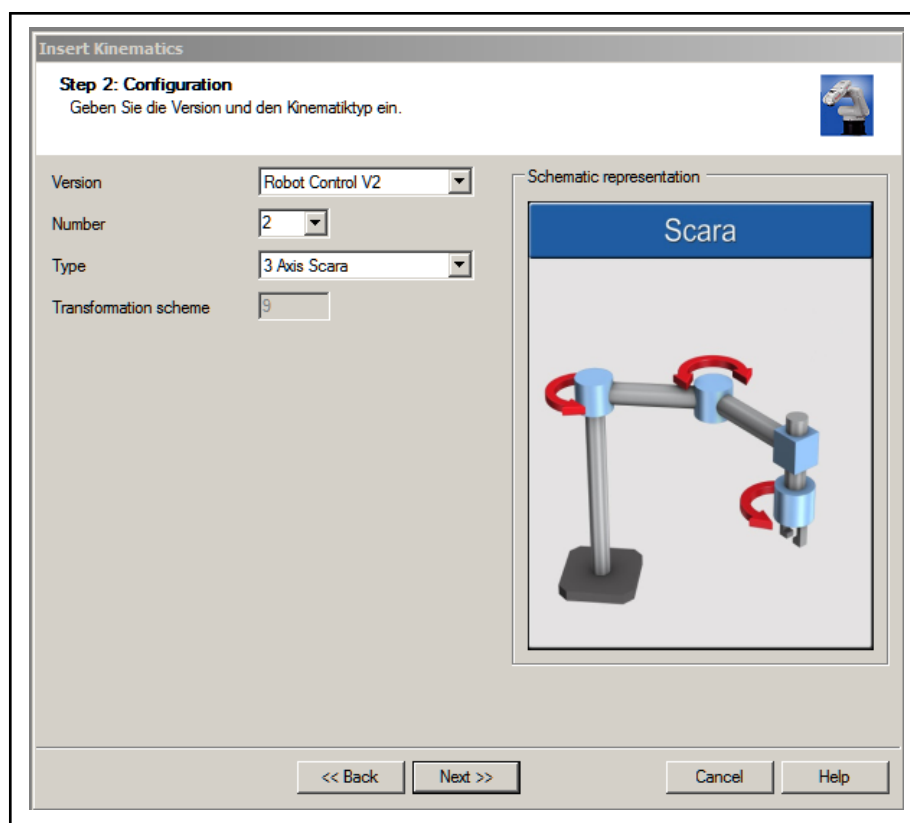


Fig.3-2: Creating the kinematics, step 2

Step 3: Assigning axes

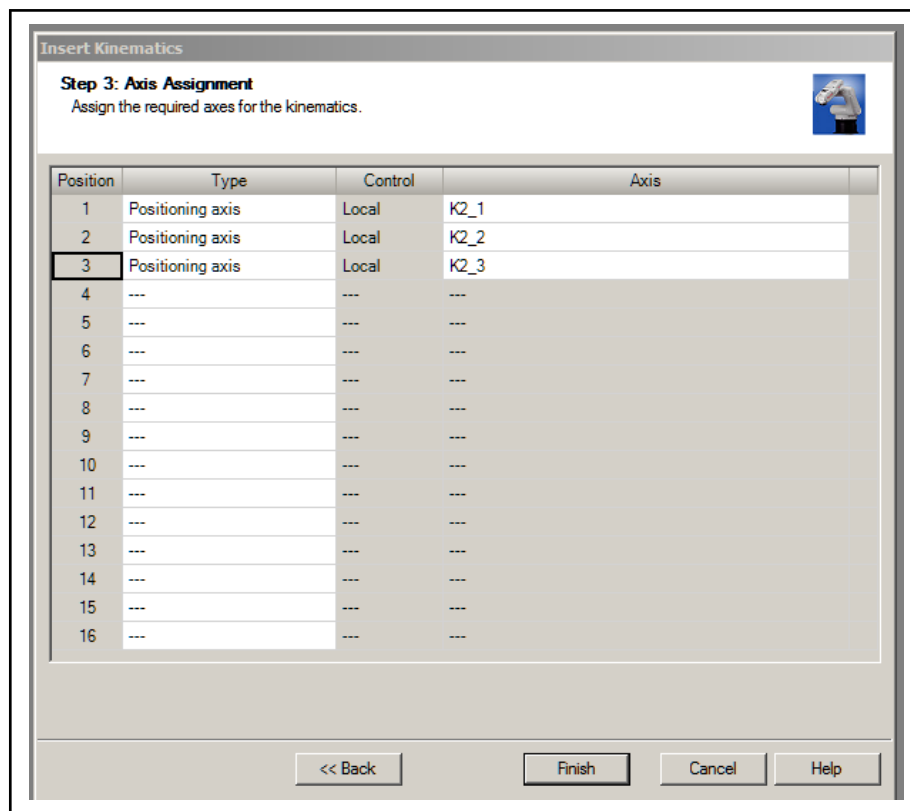


Fig.3-3: Creating the kinematics, step 3

The entered axes (positioning axes, orientation axes and belt axes) are applied to parameter K-0-0008 (Axis configuration list).

After the kinematics has been completed successfully, a new menu item (kinematic name) is created under the Robot node.

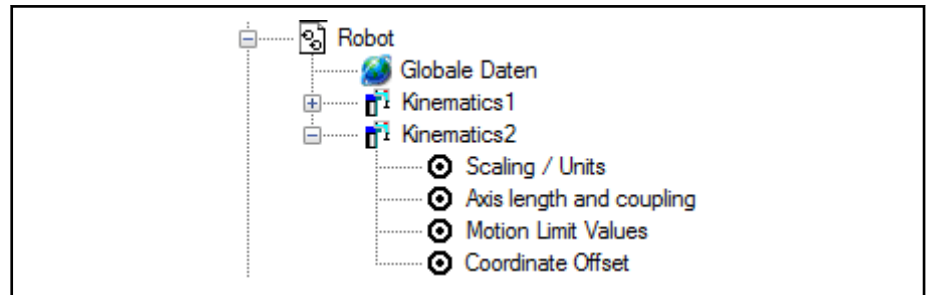


Fig. 3-4: New menu item "Kinematics2"

3.1.3 Menu item: Kinematics

Select a kinematics node in the Project Explorer. For this kinematics, a new menu item (e.g. **Kinematics2**) is enabled in the IndraWorks main menu.

The name of the menu item is the name of the selected kinematics.

This menu item contains a subset of the context menu entries of the kinematics.

3.2 Kinematics - Context Menu

3.2.1 Kinematics - Context Menu and/or Kinematics Menu Item, General Information

The context menu of a kinematics and the main menu item **Kinematics** contain the following items:

- **Initial commissioning** - Commissioning cycle of the kinematics, see [Set-up on page 68](#)
- **Parameter** - Edits parameters
- **Diagnostics** - Diagnostic functions of the kinematics, see [Diagnostics on page 68](#)
- **Export...** - Saves the robot elements in a file
- **Import...** - Imports robot elements from a file
- **Compare...** - Compares and merges objects

The following menu items are **only** provided in the context menu of the kinematics:

- **Delete** - Deletes the selected kinematics
- **Rename** - Renames the selected kinematics
- **Find element...** - Searches for elements in the project
- **Print preview...** - Provides a preview of the kinematic data
- **Print...** - Prints the current kinematic data
- **New view** - Displays the kinematics in an individual window
- **Properties...** - Provides information on the kinematics (such as name or type)

Interface Functions

3.2.2 Setup

For initial commissioning of a kinematics, complete the following dialogs using a wizard:

- [Scaling/Units on page 70](#)
- [Axis Lengths and Coupling on page 70](#)
- [Motion Limit Values on page 70](#)
- [Coordinate Offset on page 71](#)

3.2.3 Diagnostics

This dialog box graphically displays the status information of the respective kinematics. The upper part of the window displays general information on the kinematics.

The middle part displays the kinematic-specific information on all axes of a kinematics in a table. The information comprises the current position, the programmed end point (only in automatic mode) and the lagging of the individual axes.

The lower part of the window displays the kinematic status in detail.

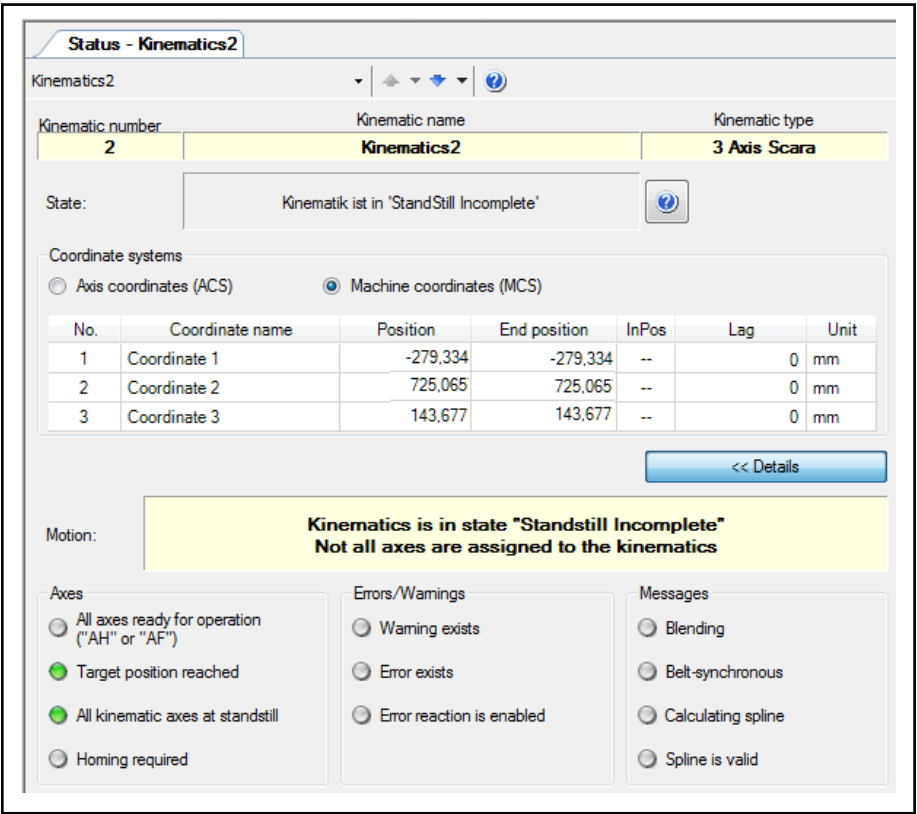


Fig.3-5: Diagnostics of a kinematics

Coordinate system

The setting under "Coordinate system" specifies the coordinate system in which the axis values or belt values are to be displayed.

There are two settings:

- **Axis coordinate system (ACS)** = Displays the positions of the single kinematic axes

Interface Functions

Coordinate systems						
☒ Axis coordinates (ACS) ☐ Machine coordinates (MCS)						
Axis No.	Axis name	Position	End position	InPos	Lag	Unit
4	vAxis4	98,959	98,959	☐	0	Deg
5	vAxis5	27,759	27,759	☐	0	Deg
6	vAxis6	16,959	16,959	☐	0	Deg

Fig.3-6: Display of the axis coordinates (ACS)

- **Machine coordinates (MCS)** = Displays the spatial coordinates based on the machine zero point

Coordinate systems						
☐ Axis coordinates (ACS) ☒ Machine coordinates (MCS)						
No.	Coordinate name	Position	End position	InPos	Lag	Unit
1	Coordinate 1	-279,334	-279,334	☐	0	mm
2	Coordinate 2	725,065	725,065	☐	0	mm
3	Coordinate 3	143,677	143,677	☐	0	mm

Fig.3-7: Display of the machine coordinates (MCS)

List of axis coordinates

The list of coordinates contains the following:

- Number of the coordinate
- Name of the coordinate
- Current position of the coordinate
- End point of the coordinate
- Lag of the coordinate

If the "InPos" checkbox is ticked, this indicates that the respective coordinate is "in position".

3.3 Kinematics – Dialogs

3.3.1 Overview on dialogs

When the kinematics folder is opened in the Project Explorer, individual dialogs are provided.

These dialogs are disabled at first and enabled when the project is switched to online mode.

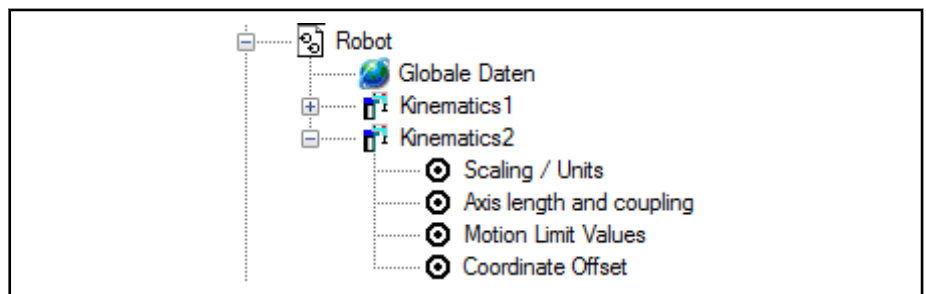


Fig.3-8: Kinematics dialogs (Robot Control V2)

- For scaling/units, see [Scaling/Units on page 70](#)
- For axis length and coupling, see [Axis Lengths and Coupling on page 70](#)
- For motion limit values, see [Motion Limit Values on page 70](#)
- For the coordinate offset, see [Coordinate Offset on page 71](#)

Interface Functions

3.3.2 Scaling/Units

Use this dialog to toggle between the units millimeter [mm] and inch [in].

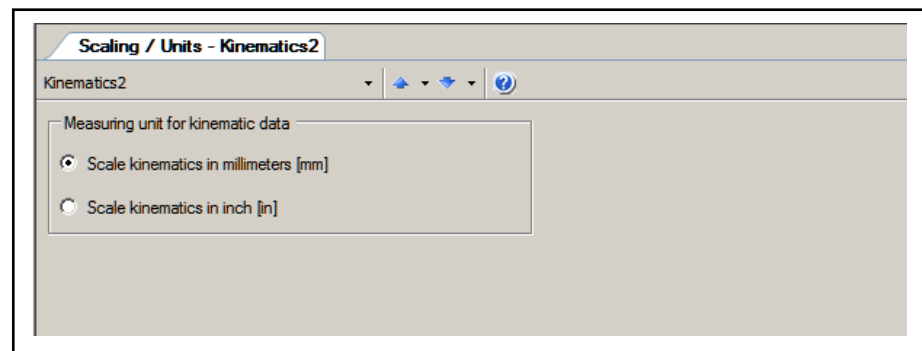


Fig.3-9: Toggling units

The following parameter is evaluated:

- K-0-0054, Measuring unit for kinematic data

3.3.3 Axis Lengths and Coupling

Set the transformation lengths and transformation factors required to transform axis coordinates into Cartesian coordinates in this dialog.

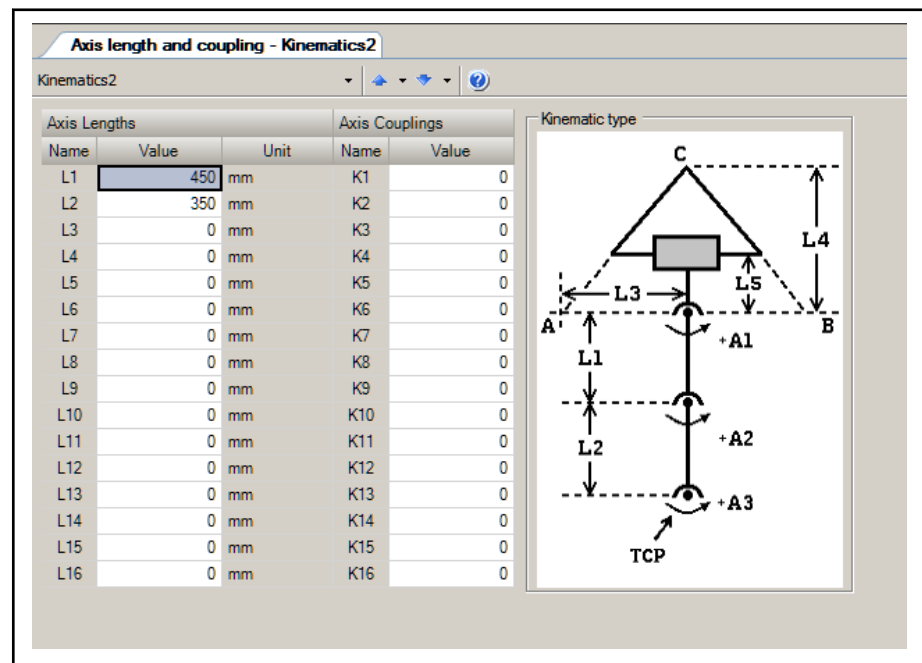


Fig.3-10: Defining axis lengths and axis couplings

The following parameters are evaluated:

- K-0-0040, Transformation, lengths of the mechanics
- K-0-0041, Transformation, factors

3.3.4 Motion Limit Values

The following limit values can be entered:

- Maximum path velocity
- Maximum path acceleration (bipolar)
- Maximum path jerk (bipolar)

Interface Functions

- Software limit switch, positive, MCS
- Software limit switch, negative, MCS

The dialog box provides an overview on the preset motion limit values of the kinematics.

Motion Limit Values - Kinematics2			
Kinematics2			
Maximum path velocity:	200	mm/s	
Maximum path acceleration:	0	mm/s²	
Maximum path jerk:	0	mm/s³	
Software limit switch for machine coordinates (MCS)			
	Positive		Negative
Coordinate 1:	999999 mm		-999999 mm
Coordinate 2:	999999 mm		-999999 mm
Coordinate 3:	999999 mm		-999999 mm

Fig.3-11: "Motion Limit Values" dialog

The following parameters are evaluated:

- K-0-0003, Maximum path velocity
- K-0-0034, Maximum path acceleration (bipolar)
- K-0-0035, Maximum path jerk (bipolar)
- K-0-0005, Software limit switch, positive, MCS
- K-0-0006, Software limit switch, negative, MCS

3.3.5 Coordinate Offset

Use this dialog to rotate and move the base coordinate system of a kinematics in relation to the machine coordinate system.

The values entered here represent the spatial position and rotation of the kinematics.

The dialog box provides an overview on the preset coordinate offset of the kinematics.

Interface Functions

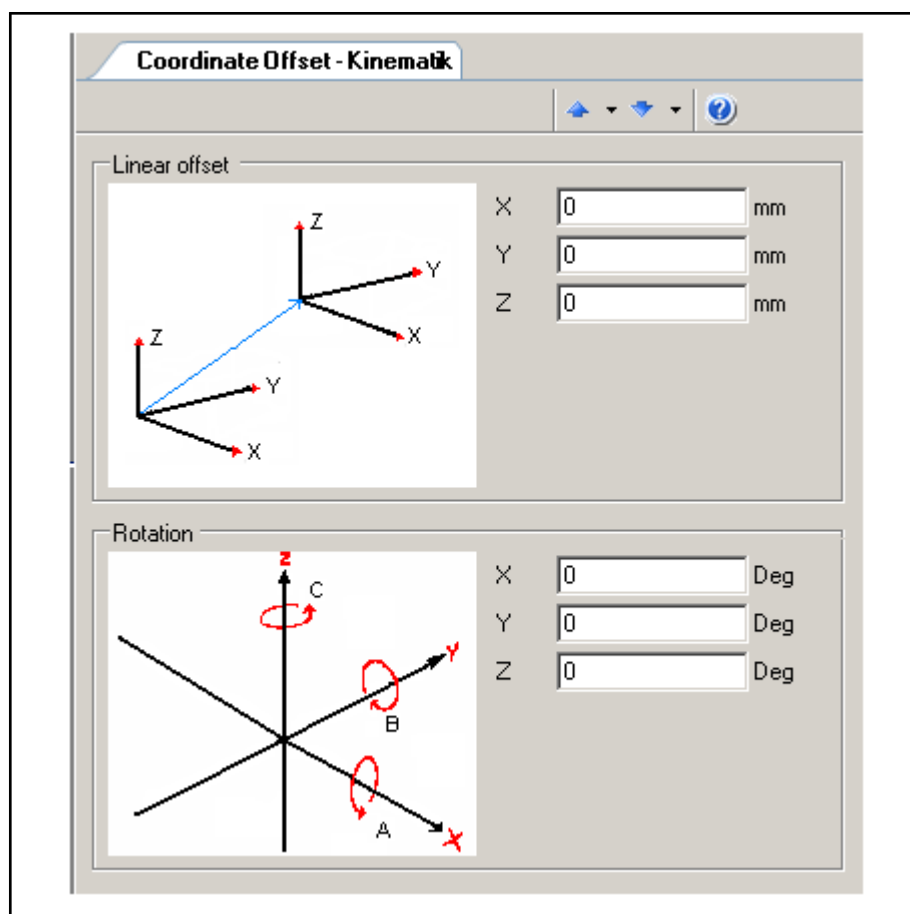


Fig.3-12: Overview on the coordinate offset

The following parameter is evaluated:

- K-0-0042, Offset of MCS system

4 Transformation Schemes

4.1 Transformation Schemes, Overview

The ACS/BCS transformation converts positions between axis coordinate systems (ACS) and base coordinate systems (BCS) (see also [ACS/BCS Transformation on page 23](#)).

Basic requirements

The basic requirements to perform the ACS/BCS transformation correctly are:

- **Home position:** Exact setting of the axis zero points
- **Adjustment:** Exact adjustment of the axis angles
- **Scaling:** Correct scaling of the axes
- **Direction of rotation:** Correct direction of rotation of the axes



If one of these basic requirements is not met, correct motion is not guaranteed.

This can result in incorrect or unexpected motions.

Parameters involved

The following parameters are involved in the configuration of an ACS/BCS transformation:

- **K-0-0008:** Axis configuration list
- **K-0-0031:** Transformation scheme
- **K-0-0040:** Transformation, lengths of the mechanics
- **K-0-0041:** Transformation, factors

Transformation scheme	Name
0	Cartesian on page 74
7	2-Axis Delta with 1 Optional Orientation Axis on page 75
9	3-Axis Scara with 1 Optional Orientation Axis on page 77
10	4-Axis Scara with 1 Optional Orientation Axis on page 79
12	2-Axis Parallelogram on page 81
13	Inverted 2-Axis Parallelogram on page 82
18	Belt Kinematics on page 83
19	2-Axis Parallelogram V1 on page 84
20	3-Axis Delta with 1 Optional Orientation Axis on page 85
21	2-Axis Parallelogram V2 on page 87
38	3-Axis Twisting Crane on page 88
39	2-Axis Flex Stamper on page 89
40	3-Axis Delta with Linear Axes and 1 Optional Orientation Axis on page 91
45	2-Axis Delta Slide on page 94
46	3-Axis Parallelogram V3 on page 95
61	4-Axis Speedpicker on page 96

Transformation Schemes

Transformation scheme	Name
100	User-defined Transformation on page 98
111	Extended User-defined Transformation on page 100

Tab.4-1: Overview of the ACS/BCS transformation schemes

4.2 Transformation Schemes

4.2.1 Cartesian

Home position

The Cartesian (transformation scheme 0) carries out a one-to-one transformation between the axis coordinate system (ACS) and the base coordinate system (BCS).

For this reason, [Move direct motions on page 29](#) are identical to [Move Linear motions on page 29](#) in relation to the path motion.

To use this transformation, enter the transformation number 0 into parameter "K-0-0031, Transformation scheme".

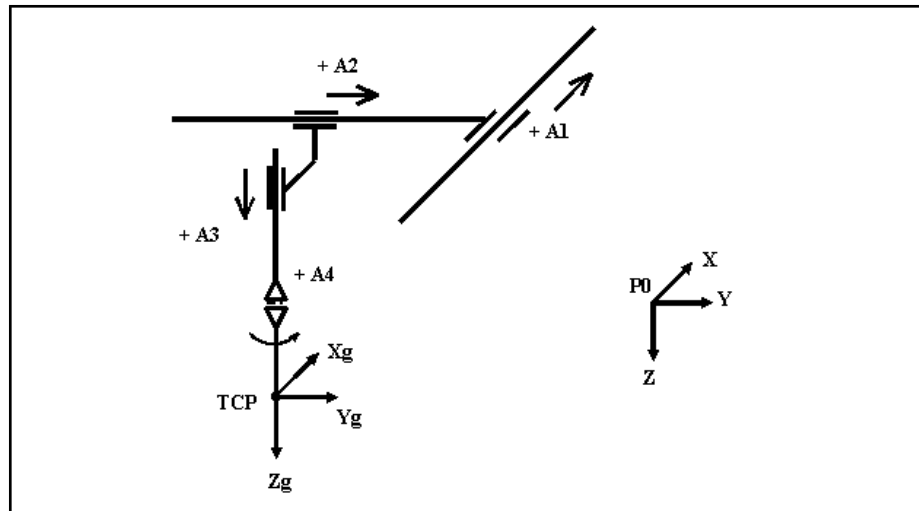


Fig.4-1: Schematic view of the Cartesian

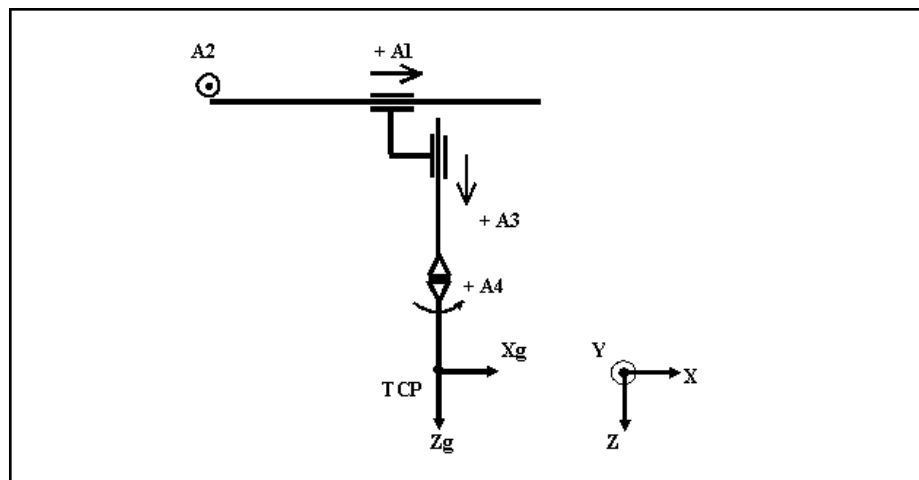


Fig.4-2: Side view: a typical X-Y-Z-C Cartesian system

Parameter settings

Transformation Schemes

Axis name	Axis type	Scaling
A1:	Positioning axis	Translatory absolute
A2:	Positioning axis (optional) Orientation axis (optional)	Translatory absolute rotary
A3:	Positioning axis (optional) Orientation axis (optional)	Translatory absolute rotary
A4:	Orientation axis (optional)	rotary
A5:	Orientation axis (optional)	
A6:	Orientation axis (optional)	
A7:	Orientation axis (optional)	
A8:	Orientation axis (optional)	
A9:	Orientation axis (optional)	
A10:	Orientation axis (optional)	
A11:	Orientation axis (optional)	
A12:	Orientation axis (optional)	
A13:	Orientation axis (optional)	
A14:	Orientation axis (optional)	
A15:	Orientation axis (optional)	
A16:	Orientation axis (optional)	

Tab.4-2: K-0-0008, Axis configuration list, axis assignment

The first orientation axis always rotates round the X-coordinate.

The second orientation axis always rotates round the Y-coordinate.

The third orientation axis always rotates round the Z-coordinate.

Any additional orientation axes are auxiliary axes.

Axis directions of rotation and zero positions

With the Cartesian, the zero position of all axes is the origin of the base coordinate system.

The travel direction for up to 3 positioning axes always forms a right-handed Cartesian coordinate system.

The zero points and travel directions of the orientation axes are those of the right-handed Cartesian coordinate system.

4.2.2 2-Axis Delta with 1 Optional Orientation Axis

To use this transformation, enter the transformation number 7 into parameter "K-0-0031, Transformation scheme".

Transformation Schemes

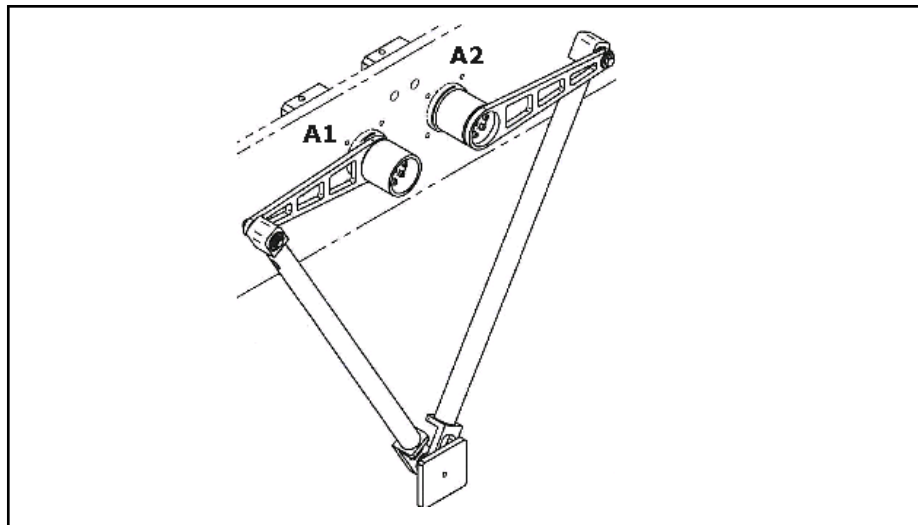


Fig.4-3: Robot type 07, 2-axis delta

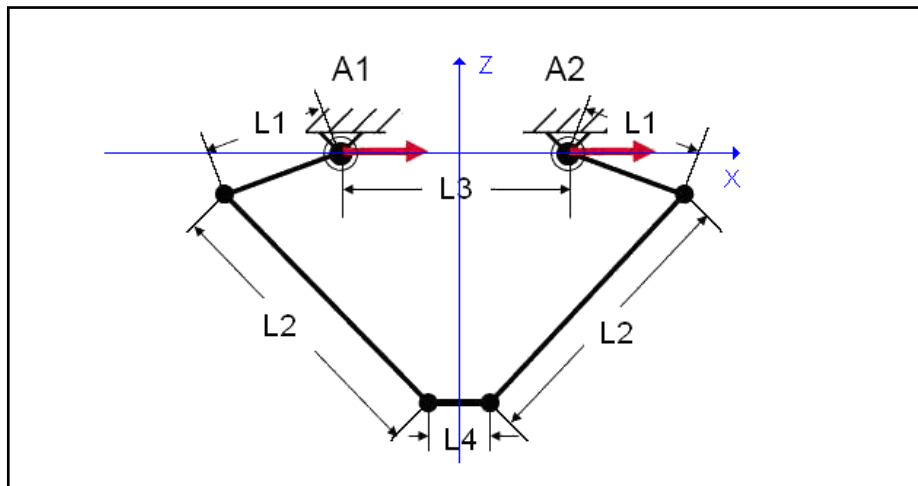


Fig.4-4: Schematic view of 2-axis delta (X/Z view)

Axis directions of rotation and zero positions

fig. 4-3 "Robot type 07, 2-axis delta" on page 76 shows a bipod robot in 3D with the axis positions $A1=180^\circ$ and $A2 = 0^\circ$.

In zero position, both positioning axes A1 and A2 are horizontal towards the X-axis (see red arrows in fig. 4-4 "Schematic view of 2-axis delta (X/Z view)" on page 76).

- The third axis rotates around the Z-axis. The zero point for the third axis (orientation axis) is in the positive direction on the x-axis.
- The arm length L_1 is measured from the rotary center point to the center point of the rod retainer
- The rod length L_2 is measured from the reference points on both sides
- L_3 is the distance of the rotary center point of both arms
- L_4 is the length of the tool plate measured at the rod mountings
- The direction of rotation of axes A1 and A2 is positive when rotating counterclockwise
- The Cartesian zero point of the bipod robot is in the center of both arm rotation points A1 and A2

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+ 90 .. 270 degrees
A2:	Positioning axis	Rotary absolute	+/- 90 degree
A3:	Orientation axis	Rotary abs./modulo	Optional

Tab.4-3: K-0-0008, Axis configuration list, axis assignment

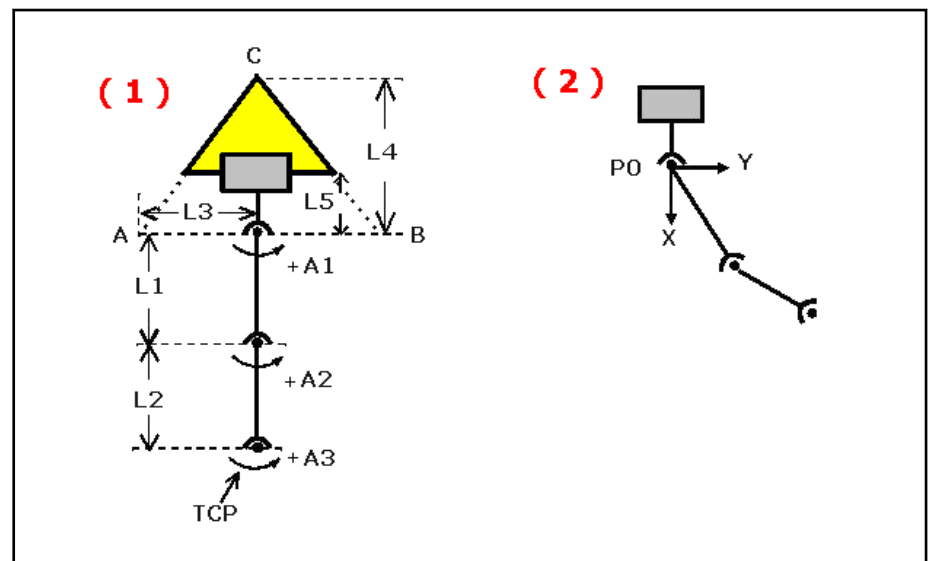
Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of rod to the tool plate	mm or inch (see K-0-0054)	>0
L3:	Distance of the two rotation points of the arms	mm or inch (see K-0-0054)	>0
L4:	Distance of rods at base plate	mm or inch (see K-0-0054)	>0

Tab.4-4: K-0-0040, Transformation lengths, axis configuration, axis lengths

K-0-0041 (Transformation, factors) is not used.

4.2.3 3-Axis Scara with 1 Optional Orientation Axis

To use this transformation, enter the transformation number 9 into parameter "K-0-0031, Transformation scheme".



(1)	Top view with zero positions for all axes; yellow: prohibited area
(2)	Zero point reference of all axes

Fig.4-5: Schematic view of 3-axis Scara with 1 optional orientation axis

Axis directions of rotation and zero positions

fig. 4-5 "Schematic view of 3-axis Scara with 1 optional orientation axis" on page 77: (1) displays the zero positions of all axes in a top view.

- If the robot is in the position shown in the figures, all axes in the machine coordinates (MCS) have to display zero.
This can be ensured by setting the actual values of the reference point.
- The figures also show arm lengths L_1 , L_2 and the positive direction of rotation of the axes.

Transformation Schemes

- A prohibited area can be defined with the lengths L_3 , L_4 , L_5 where the tool center point (TCP) may not enter. This prevents collisions when the travel ranges of axes 1 and 2 are so large that the robot could run into its own support. In ①, this prohibited area is shown in yellow.
- If the length $L_3=0$, range monitoring is not enabled.
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and L_1 , L_2 may not be equal to zero.
- The transformation can consider a mechanical coupling of axes 1 and 2 with axis 3.

Such a coupling exists if axis 3 moves along when axes 1 and 2 move and the measuring system for the third axis does not detect this motion. This is generally the case if the motor for the third axis is not directly on the axis, but instead on its support.

Set this coupling using the parameter K-0-0041 (see above).



The maximum travel range of the axes 1 and 2 is +/-179 degrees.
The travel range for the third axis is unlimited.

The respective base coordinate system (BCS) is identified in [fig. 4-5 "Schematic view of 3-axis Scara with 1 optional orientation axis" on page 77 \(2\)](#) by the two main axes (x, y).

The zero point for the base coordinates is at the base point P_0 of the robot.

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Orientation axis (optional)	rotary	Optional

Tab.4-5: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or in (see K-0-0054)	>0
L2:	Length of the second arm	mm or in (see K-0-0054)	>0
L3:	Distance from the zero point to the intersection with the straight line AC	mm or in (see K-0-0054)	>=0
L4:	Distance from zero point to point C of the triangle	mm or in (see K-0-0054)	>=0
L5:	Distance from zero point to the line AB	mm or in (see K-0-0054)	>=0

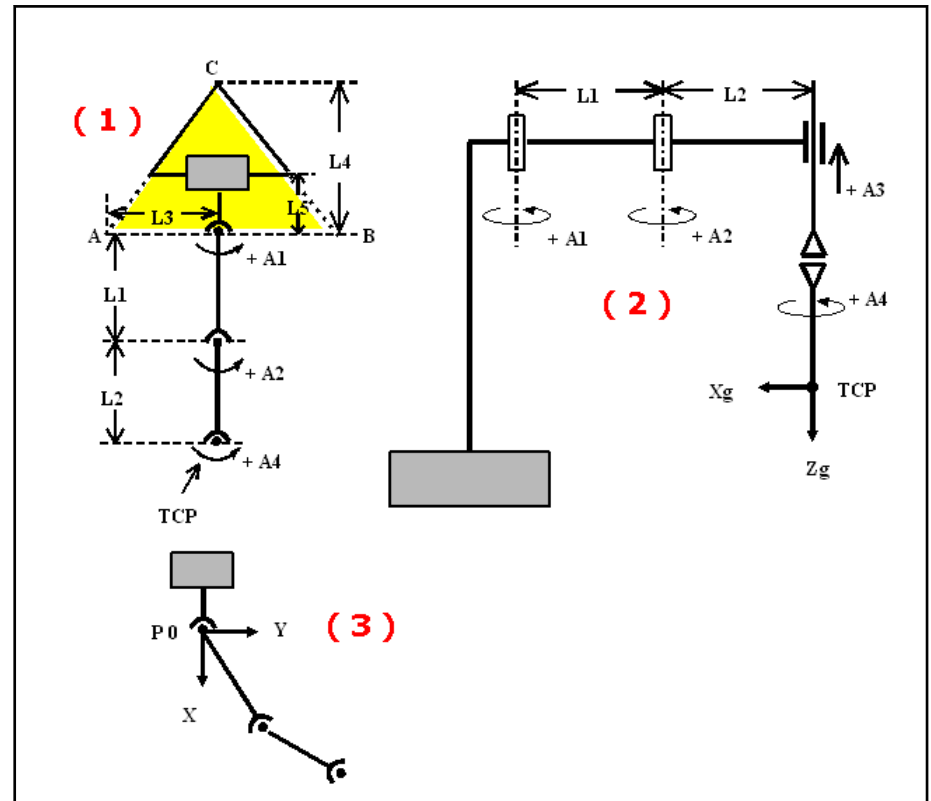
Tab.4-6: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Mechanical coupling between the axes 1, 2 and 3	Factor	Optional

Tab.4-7: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.4 4-Axis Scara with 1 Optional Orientation Axis

To use this transformation, enter the transformation number 10 into parameter "K-0-0031, Transformation scheme".



- (1) Top view with zero positions for all axes; yellow: prohibited area
 (2) Side view with zero positions for all axes
 (3) Zero point reference of all axes

Fig. 4-6: Schematic view of 4-axis Scara with 1 optional orientation axis

Axis directions of rotation and zero positions

fig. 4-6 "Schematic view of 4-axis Scara with 1 optional orientation axis" on page 79: (1) displays the zero positions of all axes in top view and (2) in side view.

- If the robot is in the position shown in the figures, all axes in the machine coordinates (MCS) have to display zero.
This can be ensured by setting the actual values of the reference point.
- In the zero position, axes 1, 2 and 4 point in forward direction.
- The zero point for the third axis can be freely chosen.
- The figures also show the arm lengths L_1 , L_2 and the positive directions of rotations of the axes.
- A prohibited area can be defined with the lengths L_3 , L_4 , L_5 where the tool center point (TCP) may not enter.
- This prevents collisions when the travel ranges of axes 1 and 2 are so large that the robot could run into its own support.
In ①, this prohibited area is shown in yellow.
- If the length $L_3=0$, range monitoring is not enabled.
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and L_1 , L_2 may not be equal to zero.

Transformation Schemes

- The transformation can consider a mechanical coupling of the axes 1 and 2 with axis 4.
Such a coupling exists if axis 4 moves along when axes 1 and 2 travel and the measuring system of axis 4 does not detect this motion. This is generally the case if the motor for the fourth axis is not directly on the axis, but instead on its support.
Set this coupling using the parameter K-0-0041 (see above).
- The transformation also consider a mechanical coupling of the fourth axis with the third axis.
Such a coupling exists if the third coordinate (Z-axis) changes when axis 4 travels and the measuring system of the third axis does not detect this motion.
This is for example the case if the axes 3 and 4 are driven by a ball screw.

The coupling factor is defined as follows:

$$K_{43} = \Delta z / \Delta A_4$$

ΔA_4 is the axis angle detected by the measuring system and Δz is the change in the coordinate.



The maximum travel range of axes 1 and 2 is ± 179 degrees.

The travel range for axes 3 and 4 is unlimited.

The respective base coordinate system is identified by the 3 main axes (X, Y, Z).

The zero point for the base coordinates is at the base point P_0 of the robot.

The transformation can also be used for a 2- or 3-axis Scara.

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Positioning axis (optional)	Translatory absolute	Optional
A4:	Orientation axis (optional)	rotary	Optional

Tab.4-8: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Distance from the zero point to the intersection with the straight line AC	mm or inch (see K-0-0054)	Optional, see ① >=0
L4:	Distance from zero point to point C of the triangle	mm or inch (see K-0-0054)	Optional, see ① >=0
L5:	Distance from zero point to the line AB	mm or inch (see K-0-0054)	Optional, see ① >=0

Tab.4-9: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation Schemes

Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Mechanical coupling between axes the 1, 2 and 4	Factor	Optional
K2:	Mechanical coupling between motors 4 and 3 (distance/lathe spindle)	mm/degree	Optional

Tab.4-10: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.5 2-Axis Parallelogram

To use this transformation, enter the transformation number 12 into parameter "K-0-0031, Transformation scheme".

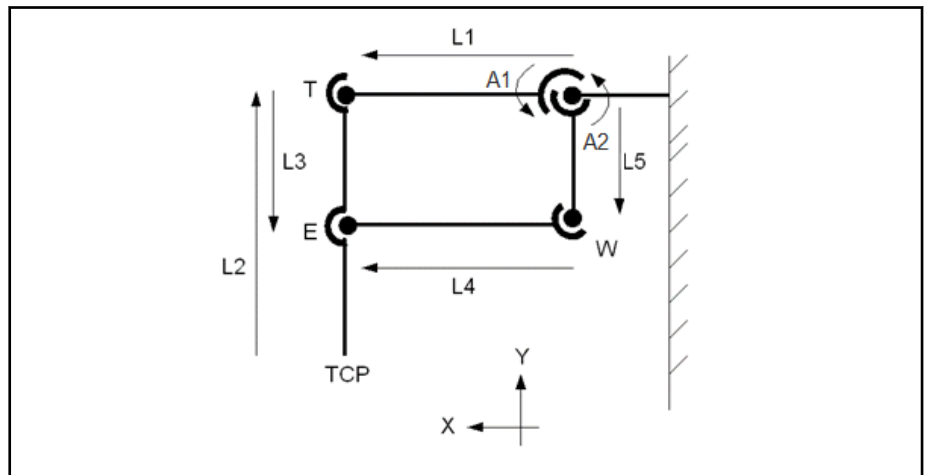


Fig.4-7: Schematic view of 2-axis parallelogram

Axis directions of rotation and zero positions

fig. 4-7 "Schematic view of 2-axis parallelogram" on page 81 shows

- The schematic kinematics representation for all axes in zero position
- The arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes in direction of the arrows
- The travel range of the axes has to be limited mechanically so that the angle between both arms with the length name L_1 and L_5 is never equal to 180° . Otherwise, two different mathematical solutions are available depending on the mechanics. This can result in undesired motions
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as a positive number and unequal to zero
- The respective base coordinate system is identified by the two main axes (X, Y)
- The base coordinate zero point is located in the intersection of the two length specifications L_1 and L_5

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees

Tab.4-11: K-0-0008, Axis configuration list, axis assignment

Transformation Schemes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Length of the third arm	mm or inch (see K-0-0054)	>0
L4:	Length of the fourth arm	mm or inch (see K-0-0054)	>0
L5:	Length of the fifth arm	mm or inch (see K-0-0054)	>0

Tab.4-12: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.6 Inverted 2-Axis Parallelogram

To use this transformation, enter the transformation number 13 into parameter "K-0-0031, Transformation scheme".

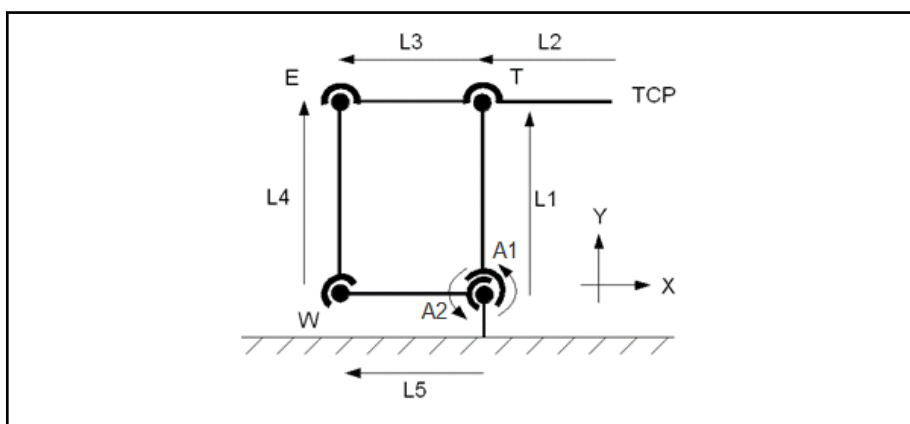


Fig.4-8: Schematic view of 2-axis parallelogram

Axis directions of rotation and zero positions

fig. 4-8 "Schematic view of 2-axis parallelogram" on page 82 shows

- The schematic kinematic representation for axes in zero position.
- The arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes in direction of the arrows.
- The traversing range of the axes has to be limited mechanically so that the angle between both arms with the length name L_1 and L_5 is never equal to 180° . For this case, there are two different mathematical solutions, which may lead to undesired motions
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and unequal to zero.
- The respective base coordinate system is identified by the two main axes (X, Y).
- The base coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees

Tab.4-13: K-0-0008, Axis configuration list, axis assignment

Transformation Schemes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Length of the third arm	mm or inch (see K-0-0054)	>0
L4:	Length of the fourth arm	mm or inch (see K-0-0054)	>0
L5:	Length of the fifth arm	mm or inch (see K-0-0054)	>0

Tab.4-14: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.7 Belt Kinematics

To use this transformation, enter the transformation number 18 into parameter "K-0-0031, Transformation scheme".

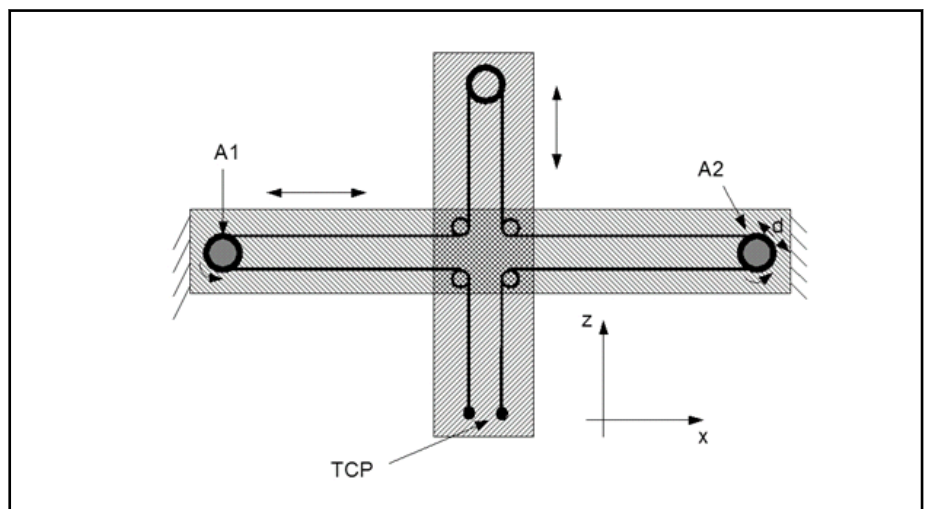


Fig.4-9: Schematic representation of the belt kinematics

Axis directions of rotation and zero positions

- The upper figure shows the schematic kinematic representation for axes in zero position. Consequences for the application: the axes have to be traveled to their desired zero position. Subsequently, they have to be referenced.
- The positive directions of rotation of the axes are shown in the figures by the direction of the arrows.
- The set travel range limits of the axes may not exceed the mechanical travel ranges.
- The respective base coordinate system is identified by the two main axes (X, Z).
- The gear of both axes has to set so that the radius ($d/2$) is entered in mm on the load side ($S-0-0121 = 1$, $S-0-0122 = d/2$).

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	Gear: the radius ($d/2$) has to be entered at load side
A2:	Positioning axis	Rotary absolute	Gear: the radius ($d/2$) has to be entered at load side

Tab.4-15: K-0-0008, Axis configuration list, axis assignment

Transformation Schemes

4.2.8 2-Axis Parallelogram V1

To use this transformation, enter the transformation number 19 into parameter "K-0-0031, Transformation scheme".

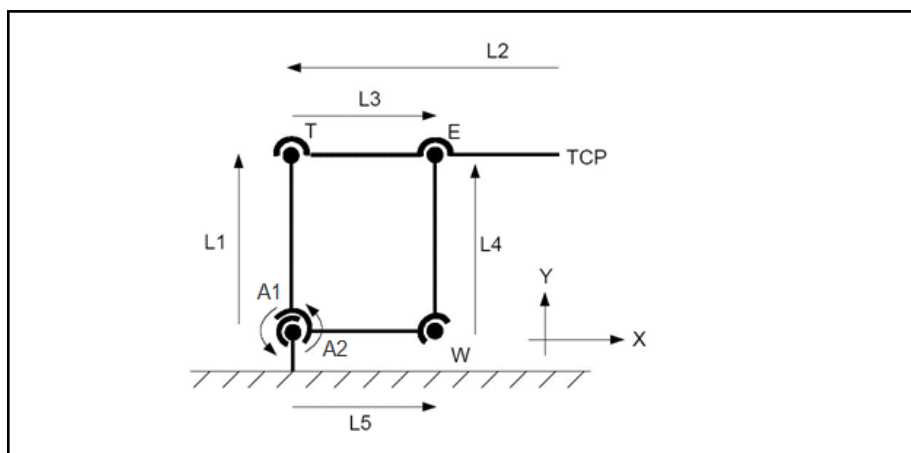


Fig.4-10: Schematic view of 2-axis parallelogram V1

Axis directions of rotation and zero positions

fig. 4-10 "Schematic view of 2-axis parallelogram V1" on page 84 shows

- The schematic kinematic representation for axes in zero position.
- The arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes in direction of the arrows.
- The travel range of the axes has to be limited mechanically so that the angle between both arms with the length name L_1 and L_5 is never equal to 180° . Otherwise, two different mathematical solutions are available depending on the mechanics. This can result in undesired motions.
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and unequal to zero.
- The respective base coordinate system is identified by the two main axes (X, Y).
- The base coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees

Tab.4-16: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Length of the third arm	mm or inch (see K-0-0054)	>0
L4:	Length of the fourth arm	mm or inch (see K-0-0054)	>0
L5:	Length of the fifth arm	mm or inch (see K-0-0054)	>0

Tab.4-17: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.9 3-Axis Delta with 1 Optional Orientation Axis

To use this transformation, enter the transformation number 20 into parameter "K-0-0031, Transformation scheme".

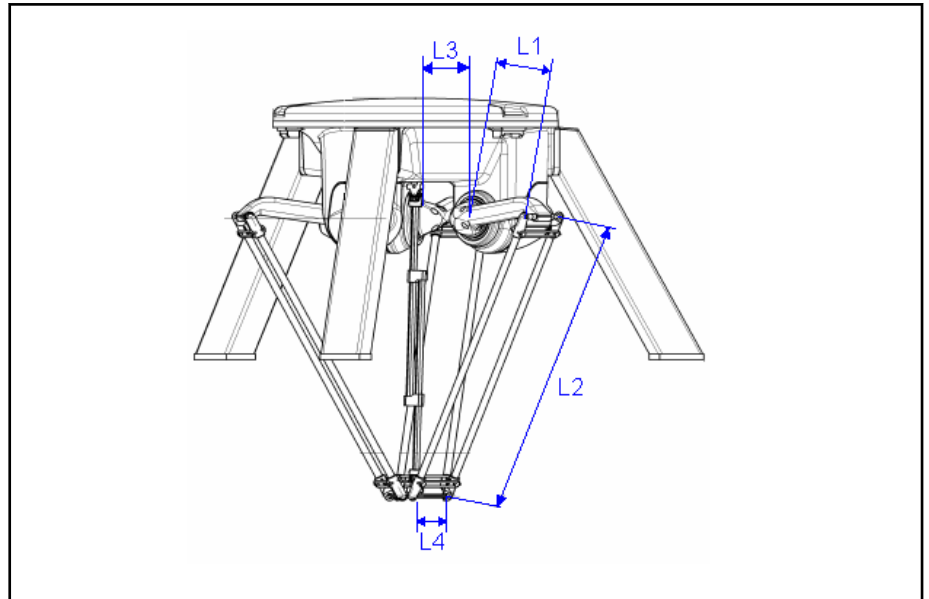


Fig.4-11: Schematic view of 3-axis delta with 1 optional orientation axis

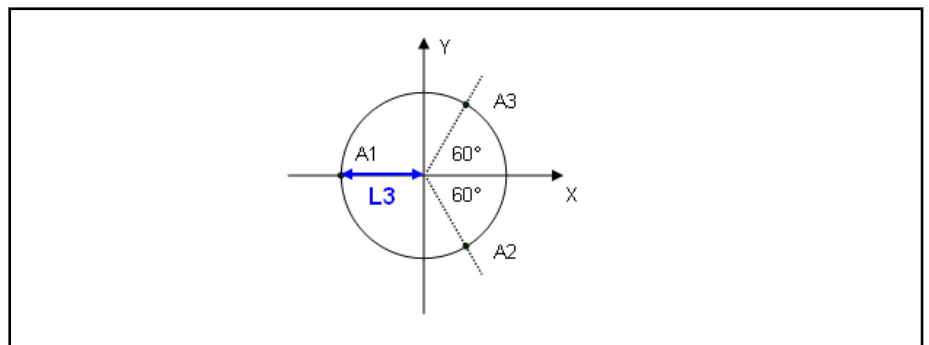


Fig.4-12: Radius L3 of the motor base and the coordinate system (X-Y view)

The three axes A1 to A3 are all displaced by 120° from each other.

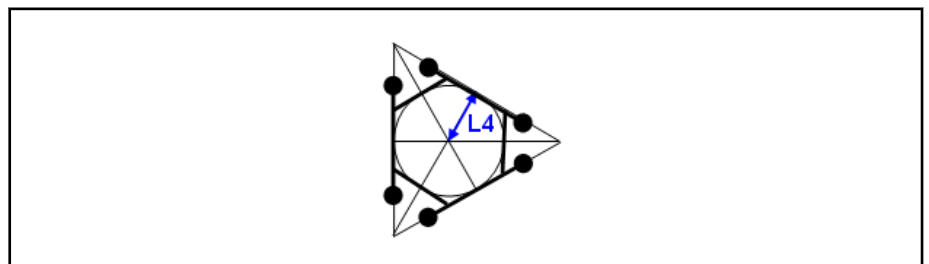
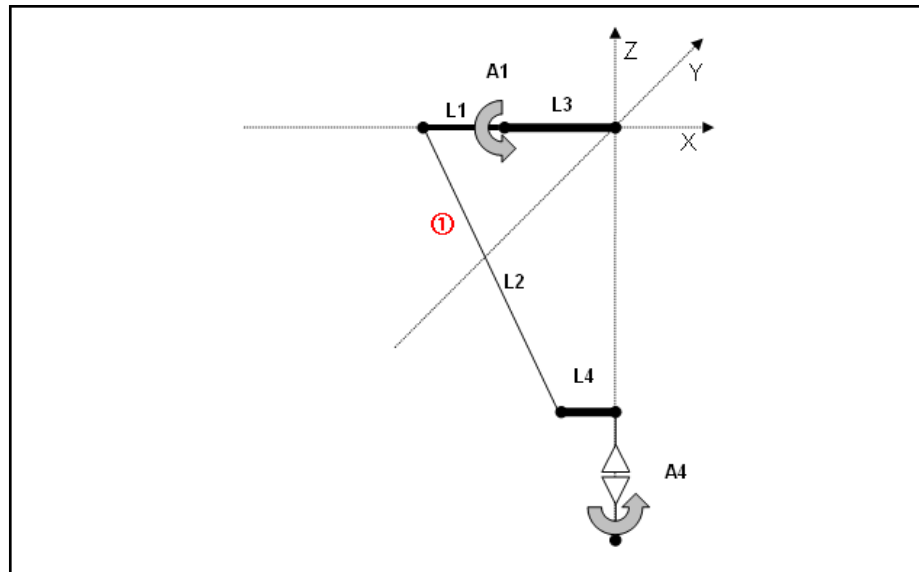


Fig.4-13: Radius L4 of the tool plate

Transformation Schemes



① Connection rod

Fig. 4-14: Representation of the zero position of the first axis

The zero position of the orientation axis A4 is in positive X-direction.

Axis directions of rotation and zero positions

- fig. 4-11 "Schematic view of 3-axis delta with 1 optional orientation axis" on page 85 shows a delta robot at zero position of all axes, i.e. if the robot is in the position shown in the figures, the machine coordinates (MCS) for all of the axes have to be zero.



This can be ensured by setting the actual values of the reference point accordingly.

- In the zero position, the axes 1, 2 and 3 (positioning axes) are horizontal.
- The zero point for the fourth axis (orientation axis) is in the positive direction of the x-axis.
- The arm length L_1 is measured from the rotary center point to the center point of the rod retainer.
- The rod length L_2 is measured from the reference points on both sides.
- L_3 is the radius of the perimeter formed by the three motor spindles (or gears) measured at the rotary center points of the arms (see fig. 4-12 "Radius L_3 of the motor base and the coordinate system (X-Y view)" on page 85).
- L_4 is the radius of the incircle of the equilateral triangle on tool plate. The equilateral triangle is defined by the reference points of the rods. (see fig. 4-13 "Radius L_4 of the tool plate" on page 85).
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down into z-direction. The orientation axis A4 (rotation along the z-axis) rotates positively counterclockwise.
- The Cartesian zero point for the delta robot is at the center point of the perimeter defined by the three motor spindles (see also fig. 4-12 "Radius L_3 of the motor base and the coordinate system (X-Y view)" on page 85).

Parameter settings

Transformation Schemes

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 90 degree
A2:	Positioning axis	Rotary absolute	+/- 90 degree
A3:	Positioning axis	Rotary absolute	+/- 90 degree
A4:	Orientation axis (optional)	Rotary abs./modulo	Optional

Tab.4-18: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of rod to the tool plate	mm or inch (see K-0-0054)	>0
L3:	Radius of the circle defined by the three rotation points of the arms	mm or inch (see K-0-0054)	>0
L4:	Radius of the base plate	mm or inch (see K-0-0054)	>0

Tab.4-19: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.10 2-Axis Parallelogram V2

To use this transformation, enter the transformation number 21 into parameter "K-0-0031, Transformation scheme".

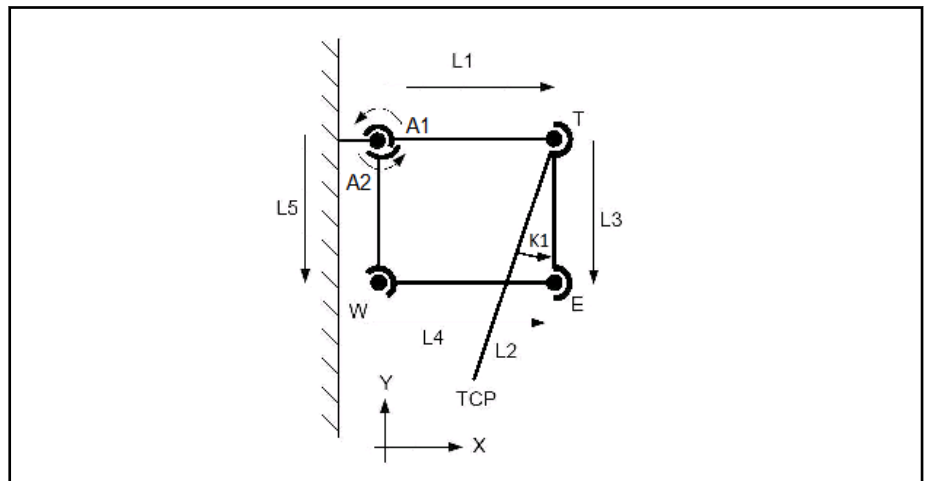


Fig.4-15: Schematic view of 2-axis parallelogram V2

Axis directions of rotation and zero positions

fig. 4-15 "Schematic view of 2-axis parallelogram V2" on page 87 shows

- The schematic kinematics representation for all axes in zero position
- The arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes in direction of the arrows
- The traversing range of the axes has to be limited mechanically so that the angle between both arms with the length name L_1 and L_5 is never equal to 180° . For this case, there are two different mathematical solutions depending on the mechanical parameters, which may lead to undesired motions
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 , K_1 always have to be entered as positive numbers and unequal to zero
- The respective base coordinate system is identified by the two main axes (X, Y)

Transformation Schemes

- The base coordinate zero point is located in the intersection of the two length specifications L_1 and L_5

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees

Tab.4-20: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Length of the third arm	mm or inch (see K-0-0054)	>0
L4:	Length of the fourth arm	mm or inch (see K-0-0054)	>0
L5:	Length of the fifth arm	mm or inch (see K-0-0054)	>0

Tab.4-21: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Angle between T, E and T, TCP	Degrees	>0

Tab.4-22: K-0-0041, Transformation factors, axis configuration, axis couplings

4.2.11 3-Axis Twisting Crane

To use this transformation, enter the transformation number 38 into parameter "K-0-0031, Transformation scheme".

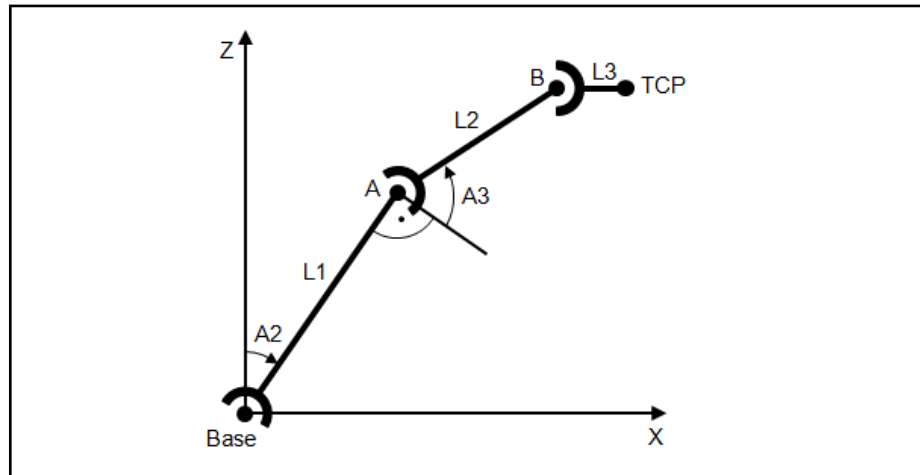


Fig.4-16: Schematic side view of 3-axis twisting crane

Transformation Schemes

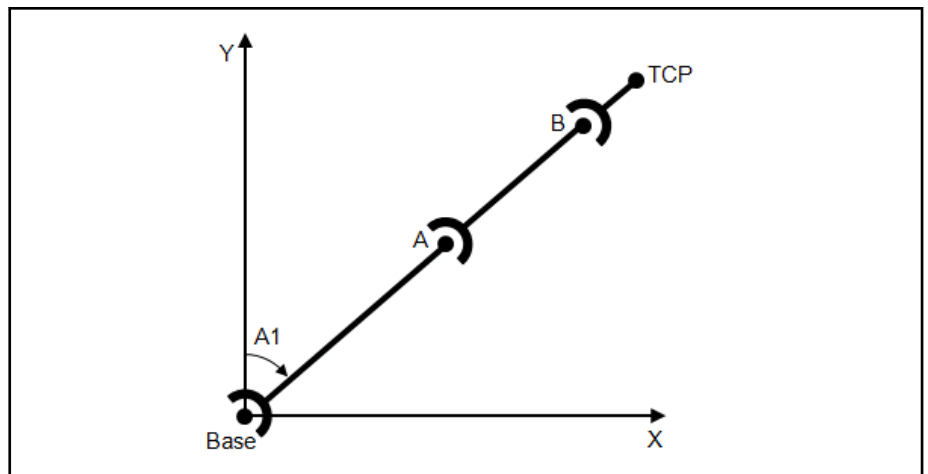


Fig.4-17: Schematic top view of 3-axis twisting crane

Axis directions of rotation and zero positions

The schematic views show the 3-axis twisting crane from the side and from above. The schematic views show the following:

- The direction of rotation and zero point of axes A1, A2, and A3
- Arm length L_1 as length between Base and A
- Arm length L_2 as length between points A and B
- Arm length L_3 as length between points B and TCP
- The third arm (B, TCP) is kept in parallel to the X/Y plane mechanically
- The respective base coordinate system is identified by the 3 main axes (X, Y, Z)
- The base coordinate zero point is in the assembly point of axes A1 and A2

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	$-180^\circ < x < 180^\circ$
A3:	Positioning axis	Rotary absolute	$-90^\circ < x < 90^\circ$

Tab.4-23: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Arm length Base, A	mm or inch (see K-0-0054)	>0
L2:	Arm length A, B	mm or inch (see K-0-0054)	>0
L3:	Arm length B, C	mm or inch (see K-0-0054)	≥ 0

Tab.4-24: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.12 2-Axis Flex Stamper

To use this transformation, enter the transformation number 39 into parameter "K-0-0031, Transformation scheme".

Transformation Schemes

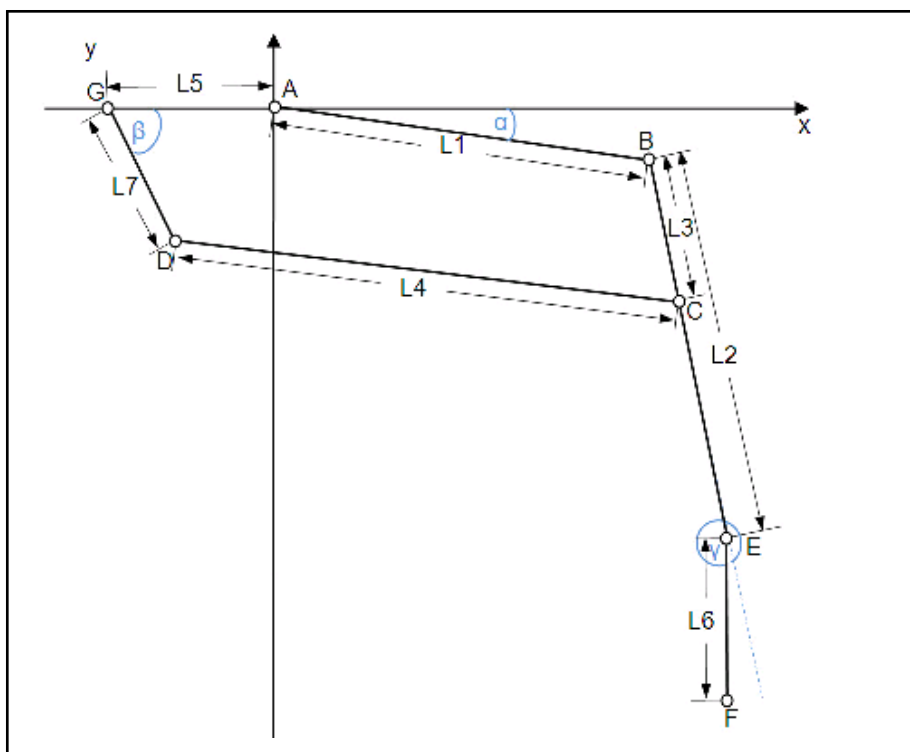


Fig.4-18: Schematic view of 2-axis flex stamper

Axis directions of rotation and zero positions

fig. 4-18 "Schematic view of 2-axis flex stamper" on page 90 shows

- The schematic kinematics representation in position axis 1= α , axis 2 = β , axis 3 = γ
- The arm lengths L_1 , L_2 , L_3 , L_4 , L_5 , L_6 , L_7
- The respective base coordinate system is identified by the two main axes (X, Y)
- The base coordinate zero point is in the assembly point of axis A1

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	$0^\circ < x < 360^\circ$
A2:	Positioning axis	Rotary absolute	$0^\circ < x < 360^\circ$
A3:	Orientation axis	Rotary absolute	$0^\circ < x < 360^\circ$

Tab.4-25: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Arm length A, B	mm or inch (see K-0-0054)	>0
L2:	Arm length B, E	mm or inch (see K-0-0054)	>0
L3:	Arm length B, C	mm or inch (see K-0-0054)	>0 <L2
L4:	Arm length D, C	mm or inch (see K-0-0054)	>0
L5:	Positioning difference of axis 1 and axis 2 (A, G)	mm or inch (see K-0-0054)	>=0

Transformation Schemes

L6:	Arm length E, F	mm or inch (see K-0-0054)	>0
L7:	Arm length G, D	mm or inch (see K-0-0054)	>0

Tab.4-26: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration
K1:	0: the angle of axis 3 is automatically selected in such a way that arm E,F is in parallel to the Y-axis 1: the angle of axis 3 can be specified as orientation around the Z-axis by the user

Tab.4-27: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.13 3-Axis Delta with Linear Axes and 1 Optional Orientation Axis

To use this transformation, enter the transformation number 40 into parameter "K-0-0031, Transformation scheme".

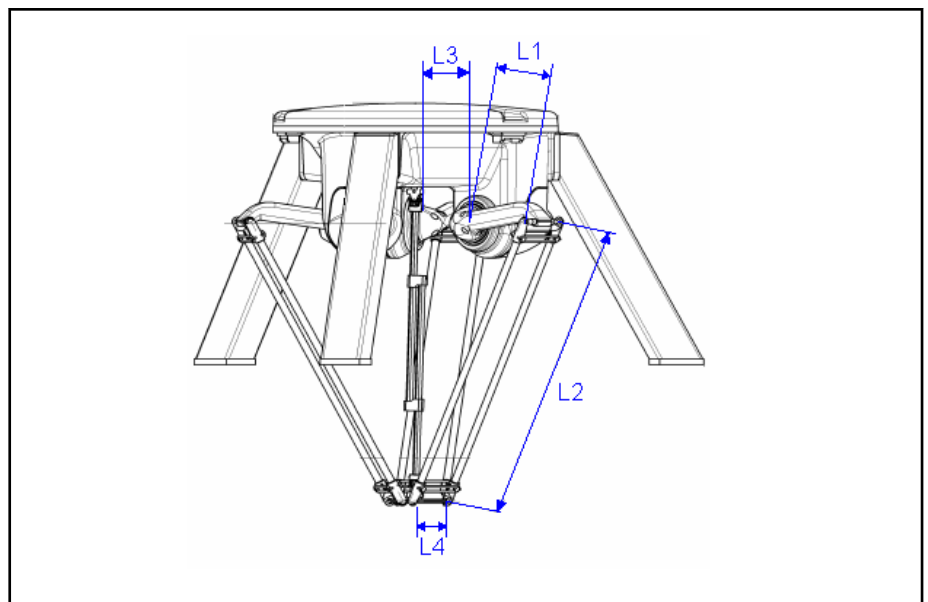


Fig.4-19: Schematic view of 3-axis delta with 1 optional orientation axis

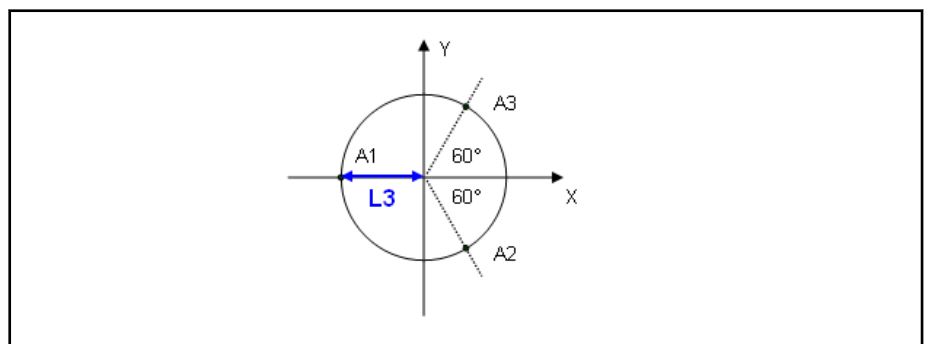


Fig.4-20: Radius L3 of the joint base and the coordinate system (X-Y view)

The three joints A1 to A3 are all displaced by 120° from each other.

Transformation Schemes

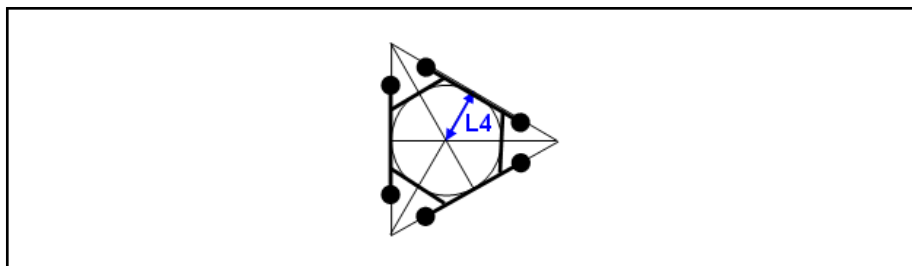
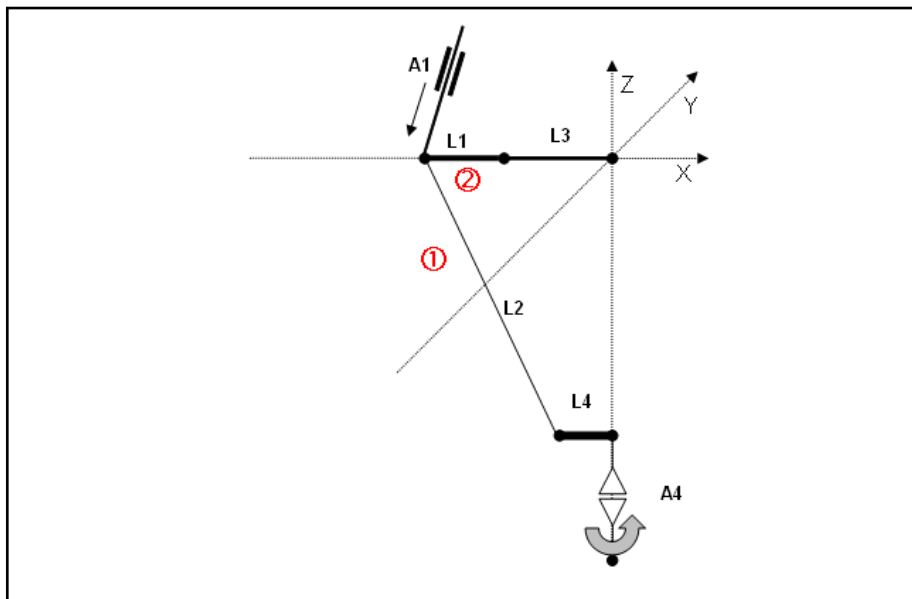


Fig. 4-21: Radius L4 of the tool plate



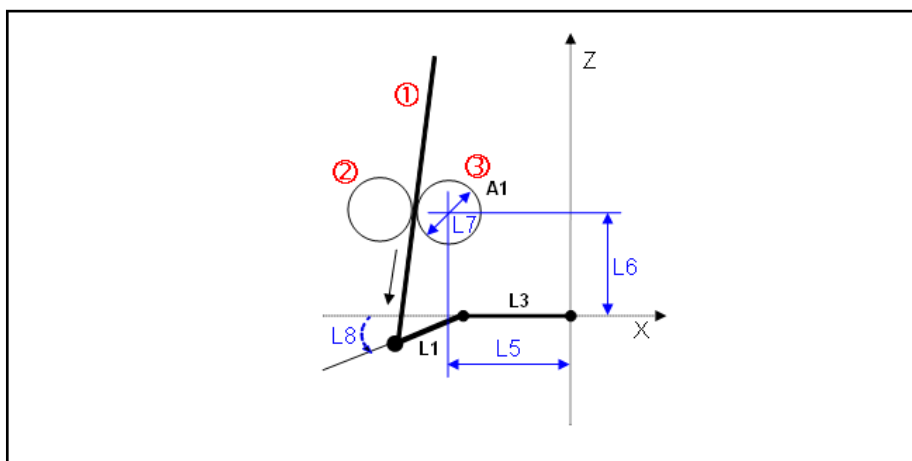
- ① Connection rod
② Arm

Fig. 4-22: Representation of the zero position

The zero position of the orientation axis A4 should be in positive x-direction.

- K-0-0040 (Transformation, lengths of the mechanics), transformation lengths, axis configuration

Axis length entries L1 to L8:



- ① Rack
② Pressure roller
③ Drive roller

Fig. 4-23: Advanced Settings

Transformation Schemes

Axis directions of rotation and zero positions

fig. 4-19 "Schematic view of 3-axis delta with 1 optional orientation axis" on page 91 shows a delta robot at zero position of all axes, i.e. if the robot is in the position shown in the figures, the machine coordinates (MCS) for all of the axes have to be zero.



This can be ensured by setting the actual values of the reference point accordingly.

- In zero position, the arms are horizontal
- The zero point for the 4th axis (orientation axis) is in the positive direction of the X-axis
- The arm length L_1 is measured from the rotary center point to the center point of the rod retainer
- The rod length L_2 is measured from the reference points on both sides
- L_3 is the radius of the perimeter formed by the three motor spindles (or gears), measured at the rotary center points of the arms (see fig. 4-20 "Radius L_3 of the joint base and the coordinate system (X-Y view)" on page 91)
- L_4 is the radius of the incircle of the equilateral triangle on tool plate. The equilateral triangle is defined by the reference points of the rods. (see fig. 4-21 "Radius L_4 of the tool plate" on page 92)
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down into z-direction. The orientation axis A4 (rotation along the Z-axis) rotates positively counterclockwise
- The Cartesian zero point for the Delta robot is at the center point of the perimeter defined by the three motor spindles (see also fig. 4-20 "Radius L_3 of the joint base and the coordinate system (X-Y view)" on page 91).

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Translatory absolute	
A2:	Positioning axis	Translatory absolute	
A3:	Positioning axis	Translatory absolute	
A4:	Orientation axis (optional)	Rotary abs./modulo	Optional

Tab.4-28: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of rod to the tool plate	mm or inch (see K-0-0054)	>0
L3:	Radius of the circle defined by the three rotation points of the arms	mm or inch (see K-0-0054)	>0
L4:	Radius of the base plate	mm or inch (see K-0-0054)	>0
L5:	Horizontal distance from the rack drive roller to the origin of the coordinate system	mm or inch (see K-0-0054)	>0
L6:	Vertical distance from the drive roller to the x-y plane	mm or inch (see K-0-0054)	>0

Transformation Schemes

L7:	Diameter of the drive roller	mm or inch (see K-0-0054)	>0
L8:	Phase correction	Degrees	-45 ... +45

Tab.4-29: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.14 2-Axis Delta Slide

To use this transformation, enter the transformation number 45 into parameter "K-0-0031, Transformation scheme".

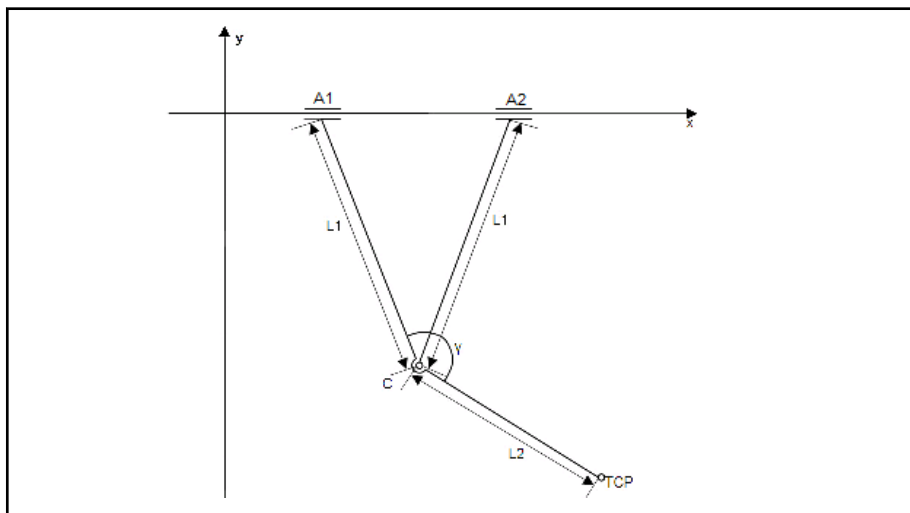


Fig.4-24: Schematic view of 2-axis delta slide

Axis directions of rotation and zero positions

fig. 4-24 "Schematic view of 2-axis delta slide" on page 94 shows

- The schematic representation of the kinematics
- The arm lengths L_1 , L_2
- The angle gamma at point C
- The positive directions of the axes in positive X-direction
- Length L_1 always has to be entered as a positive number and unequal to ZERO
- Length L_2 always has to be entered as a positive number
- The angle gamma (K_1) always has to be between 0 and 360 degrees
- The respective base coordinate system is identified by the two main axes (X, Y)

Parameter settings

Axis name	Axis type	Scaling
A1:	Positioning axis	Translatory absolute
A2:	Positioning axis	Translatory absolute

Tab.4-30: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arms A, C and B, C	mm or inch (see K-0-0054)	>0
L2:	Length of arm C, TCP	mm or inch (see K-0-0054)	>=0

Tab.4-31: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation Schemes

Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Angle gamma	Degrees	$0 \leq \gamma < 360$

Tab.4-32: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.15 3-Axis Parallelogram V3

To use this transformation, enter the transformation number 46 into parameter "K-0-0031, Transformation scheme".

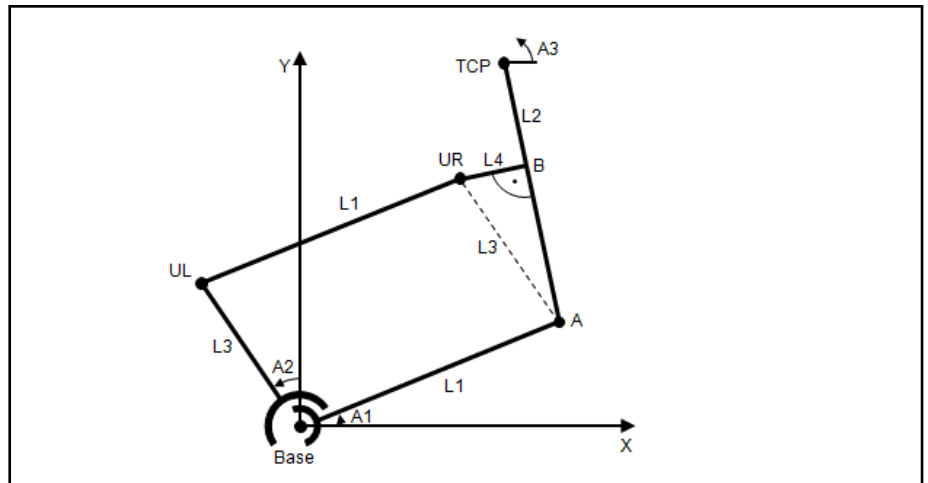


Fig.4-25: Schematic view of 3-axis parallelogram V3

Axis directions of rotation and zero positions

The schematic view contains the following data:

- Arm length L_1 between base and A and between UL and UR
- Arm length L_2 between A and TCP
- Arm length L_3 between Base and UL and between A and UR
- Arm length L_4 between B and UR
- Zero position of axes A1, A2, and A3 at the beginning of the arrows
- The positive directions of rotation of axes A1, A2, and A3 in direction of the arrows
- The lengths L_1 , L_2 , L_3 always have to be positive and unequal to zero
- Length L_4 always has to be positive
- Length L_4 has to be less than length L_3
- The respective base coordinate system is identified by the two main axes (X, Y)
- The base coordinate zero point is in the suspension point of axes A1 and A2

Parameter settings

Axis name	Axis type	Scaling
A1:	Positioning axis	Rotary absolute
A2:	Positioning axis	Rotary absolute
A3:	Orientation axis	Rotary absolute

Tab.4-33: K-0-0008, Axis configuration list, axis assignment

Transformation Schemes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second arm	mm or inch (see K-0-0054)	>0
L3:	Length of the third arm	mm or inch (see K-0-0054)	>0
L4:	Length of the fourth arm	mm or inch (see K-0-0054)	>=0

Tab.4-34: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration
K1:	Mechanical coupling between axes A2 and A3

Tab.4-35: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.16 4-Axis Speedpicker

To use this transformation, enter the transformation number 61 into parameter "K-0-0031, Transformation scheme".

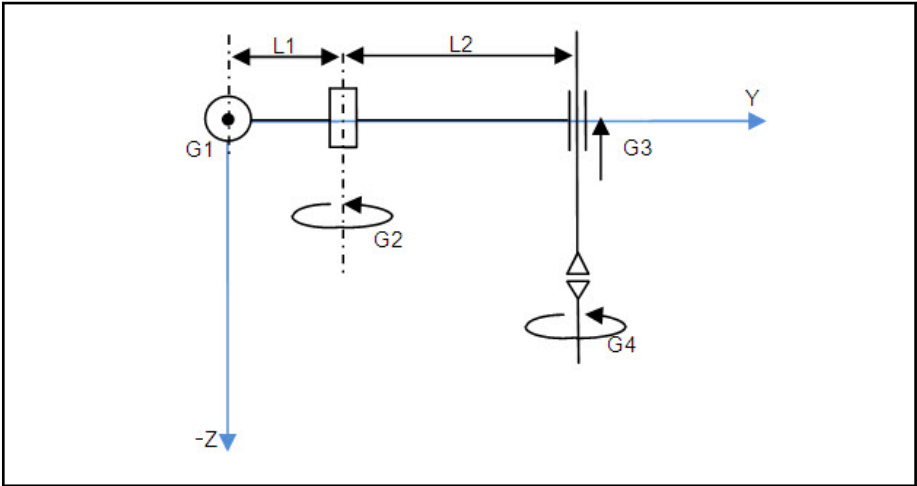


Fig.4-26: Schematic view of 4-axis speedpicker

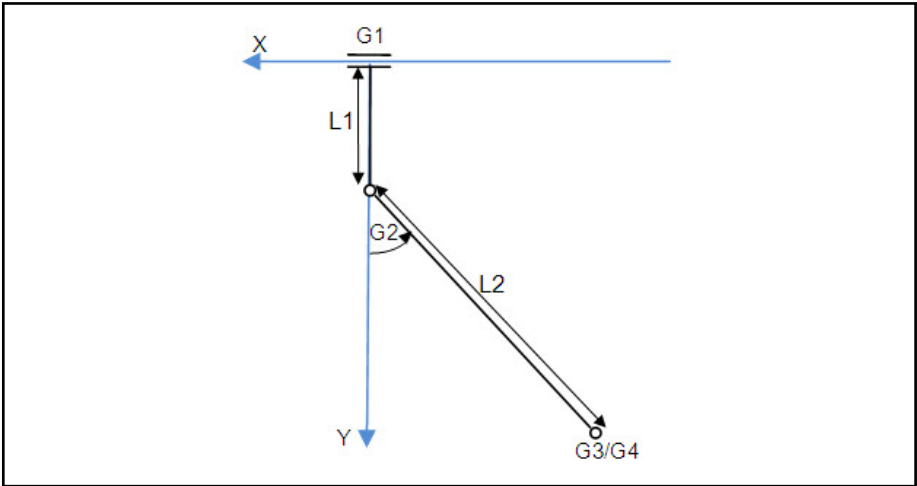


Fig.4-27: Schematic view of 4-axis speedpicker (from above)

Axis directions of rotation and zero positions

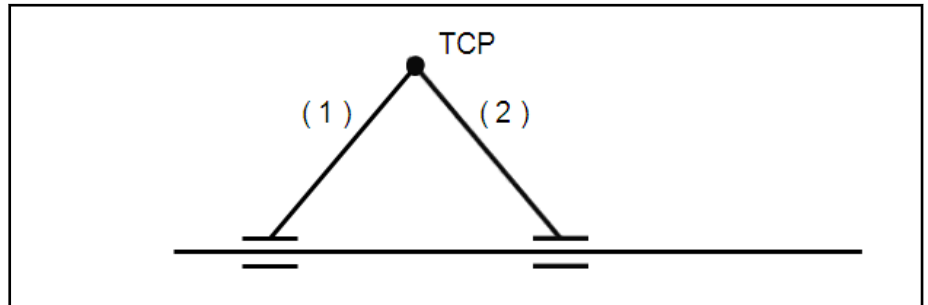
fig. 4-26 "Schematic view of 4-axis speedpicker" on page 96 and fig. 4-27 "Schematic view of 4-axis speedpicker (from above)" on page 96 show

Transformation Schemes

- The schematic representation of the kinematics
- The arm lengths L_1, L_2
- The respective base coordinate system is identified by the two main axes (X, Y)

Arm behavior with PTP motions

The coupling factor K_3 controls the arm behavior with PTP motions if the time is not specified in axis coordinates. The speedpicker can have the following arm behavior for the same TCP:



- (1) Left-armed
(2) Right-armed

Fig. 4-28: Left-/right-armed speedpicker

If the target point of a PTP motion is not specified in axis coordinates, the arm can travel to the left or right to reach the target point.

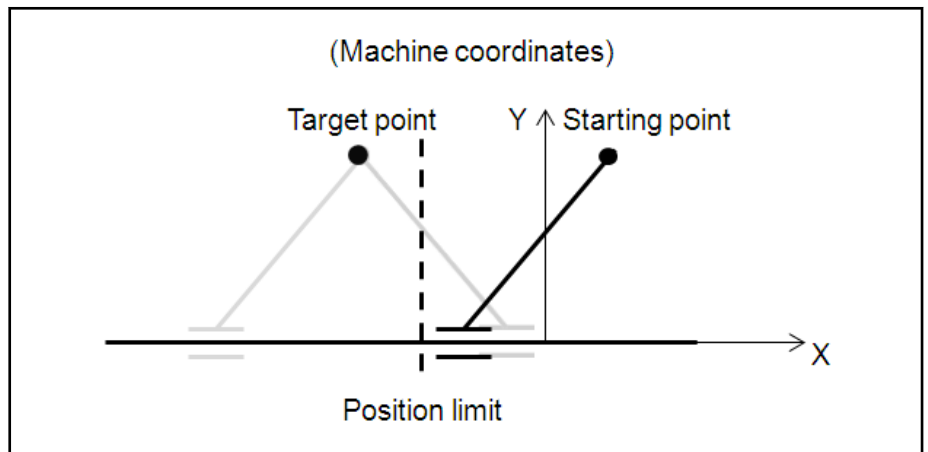


Fig. 4-29: Target point

The value in parameter K_3 controls the behavior:

- 0: arm behavior is retained. In this case, left-arm operation is used to approach the target point.
- 1: axis 1 (X-Slide) is to move only a little. In this case, the position difference from axis 1 between the start and target point is smaller if right-arm operation is used to approach the target point. Therefore, this solution is used
- 2: axis 2 (arm) is to move only a little. In this case, the position difference from axis 2 between the start and target point is smaller if left-arm operation is used to approach the target point. Therefore, this solution is used
- 3: The position limit K_4 determines the arm behavior. In this case, the target point is to the left of the position limit. Therefore, right-arm operation is used to approach the target point

Parameter settings

Transformation Schemes

Axis name	Axis type	Scaling
A1:	Positioning axis	Translatory absolute
A2:	Positioning axis	Rotary absolute
A3:	Positioning axis	Translatory absolute
A4:	Orientation axis	Rotary absolute

Tab.4-36: K-0-0008, Axis configuration list, axis assignment

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Arm length between G1 and G2	mm or inch (see K-0-0054)	≥ 0
L2:	Arm length between G2 and G3/G4	mm or inch (see K-0-0054)	> 0

Tab.4-37: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit
K1:	Mechanical coupling between axis 4 and 3 (distance/lathe spindle)	mm/degree
K3:	Arm behavior with PTP motions 0: In case of ambiguous solutions, the solution with the same arm behavior as the starting point is used 1: axis 1 is to move only a little: If with PTP motions the target positions are not specified in axis coordinates and the solutions are ambiguous, the solution that generates smaller motions of axis 1 is used 2: axis 2 is to move only a little: If with PTP motions the target positions are not specified in axis coordinates and the solutions are ambiguous, the solution that generates smaller motions of axis 2 is used 3: position limit: If with PTP motions the target point is not specified in axis coordinates, left-arm operation is used to approach the target point if the X-coordinate in the base coordinate system is greater or equal to the position limit (K4). If with PTP motions the target point is not specified in axis coordinates, right-arm operation is used to approach the target point if the X-coordinate in the base coordinate system is less than the position limit (K4). This setting is required if certain positions can only be reached with one arm behavior	
K4:	Position limit (only active at K3 = 3)	mm or inch (see K-0-0054)

Tab.4-38: K-0-0041, Transformation factors, axis configuration, axis coupling

4.3 User-defined Transformation

It often occurs in the application that the transformations provided by the system do not fit the kinematics. Therefore, the system features a user-defined transformation type. For this purpose, enter 100 as transformation scheme in parameter "K-0-0031, Transformation scheme". This type implements the rules for conversion between base and axis coordinates in the PLC.

There are two transformation directions:

- **Forward transformation:** forward transformation converts positions from ACS to BCS

Transformation Schemes

- **Inverse transformation:** inverse transformation converts positions from BCS to ACS
- Methods to be implemented**
- To define a transformation, the interface MC_KIN_REF has to be implemented by a function block.
- This interface contains three methods. These methods do not need to be called within the PLC. The system calls them automatically.
- **Initialization:** method for transformation initialization. Here, all required checks of the boundary conditions and precalculations have to be executed for the transformation to work without errors
- These include:*
- Check of the required axis number for the transformation in parameter "K-0-0008, Axis configuration list"
 - Check if the axis scaling is correct for the transformation of the configured axes in parameter "K-0-0008, Axis configuration list"
 - Check of parameters "K-0-0040, Transformation, lengths of the mechanics" and "K-0-0041, Transformation, factors"
 - Precalculation of values used in forward and inverse transformation
- The content of parameters "K-0-0008, Axis configuration list", "K-0-0040, Transformation, lengths of the mechanics" and "K-0-0041, Transformation, factors" is transmitted to the method. If initialization cannot be completed successfully, the function returns a value not equal to zero.
- **Forward:** method that converts coordinates from the axis coordinate system into coordinates of the base coordinate system ([Transformations on page 23](#)).
- For this purpose, the method receives two references to the coordinates of the axis coordinate system and the coordinates of the base coordinate system.
- From the coordinates of the axis coordinate system the method calculates the coordinates of the base coordinate system.
- If a point cannot be transformed, the method has to return a value not equal to zero.
- **Inverse:** method that converts coordinates from the base coordinate system into coordinates of the axis coordinate system ([Transformations, Overview on page 23](#)).
- For this purpose, the method receives two references to the coordinates of the base coordinate system and the coordinates of the axis coordinate system.
- From the coordinates of the base coordinate system the method calculates the coordinates of the axis coordinate system.
- If a point cannot be transformed, the method has to return a value not equal to zero.

Method error handling

If a floating point error occurs with one of the interface functions (methods), one of the following errors is reported.

However, such errors already have to be considered in the transformation mathematics: For example, always check if a divisor is zero. If yes, report an error and do not execute the division.

- F12C0600, Invalid operation at PLC function call

Transformation Schemes

- F12C0601, Denormalized operand at PLC function call
- F12C0602, Zero division at PLC function call
- F12C0603, Overflow at PLC function call
- F12C0604, Underflow at PLC function call
- F12C0605, Precision error at PLC function call

If an interface function (method) is to be called that was not registered, the following error is reported:

- F12C0498, Transformation in PLC has not been registered

If an interface function cannot be executed successfully (return value of the function is not zero), the following error is reported:

- F12C0499, User defined transformation error in PLC

Selecting a transformation

If a function block is implemented that represents a transformation (the function block implements interface [MC_KIN_REF on page 232](#)), this function block can be registered in the system with function block [MC_SetKinTransform on page 131](#).

Registration requires an instance of the transformation function block and the kinematics that the instance is to apply to.

Transformation registration has to be carried out in the configuration phase of the control (P2).

If during switching to the control operating state (BB) in the kinematics a user-defined transformation is defined but not registered, error F12C0498 "Transformation in PLC has not been registered" is reported.

If the same transformation is to be used in several kinematics, a separate instance of the specific transformation function block has to be used for each kinematics.

Use case

For a use case for the creation and application of user-defined transformations, see [Application Example 3: User-Defined Transformation on page 317](#).

4.4 Extended User-defined Transformation

It often occurs in the application that the transformations provided by the system do not fit the kinematics. For this case, the system features an extended user-defined transformation type. For this purpose, enter 111 as transformation scheme in parameter "K-0-0031, Transformation scheme".

The extended user-defined transformation type provides an extended interface for transformation definition.

This interface offers the following advantages to the simple user-defined transformation:

- Increased calculation accuracy by using the LREAL data type
- Resolving ambiguities of a transformation
- A mechanical kinematics view using mechanical information (see [Mechanical Information on page 103](#))



For the above reasons, it is recommended to use transformation type 111 (extended user-defined transformation) instead of transformation type 100 (user-defined transformation).

Transformation Schemes

The most important difference between the simple and the extended user-defined transformation is the representation of a point.

The coordinates have fixed positions for X, Y, Z, A, B, C in exactly this order.

Non-existent degrees of freedom are no longer omitted, as it was done with the simple user-defined transformation.

A transformation reports the supported degrees of freedom to the system using the interface method [MC_KIN_REF_EXT](#), [GetUsedAxes](#) on page 234.

Points have the coordinate values with the specified degrees of freedom. This applies to ACS points as well as to BCS points.

Example:

Scara with two positioning axes and one orientation axis

A Scara robot is equipped with 2 X and Y positioning axes and one Z orientation axis (C coordinate). The interface method returns `ML_KIN_USED_AXES.X_AXIS` OR `ML_KIN_USED_AXES.Y_AXIS` OR `ML_KIN_USED_AXES.Z_AXIS` (coordinates 1, 2 and 6).

ACS and BCS points contain:

- The X coordinate and/or motor 1 at index 1
- the Y coordinate and/or motor 2 at index 2
- the C coordinate and/or motor 3 at index 6

The transformation has to process these coordinates.

Methods to be implemented

To define an extended transformation, the interface [MC_KIN_REF_EXT](#) on page 234 has to be implemented by a function block.

This interface contains six methods. These methods do not need to be called within the PLC. The system calls them automatically.

- **GetUsedAxes:** This method defines the degrees of freedom of the transformation

If a transformation has a degree of freedom around an axis, the respective value of data type [ML_KIN_USED_AXES](#) on page 231 has to be returned

If there are multiple degrees of freedom, multiple values have to be linked with an OR operation

- **Initialization:** method for transformation initialization. Here, all required checks of the boundary conditions and precalculations have to be executed for the transformation to work without errors

These include:

- Check of the required axis number for the transformation in parameter "K-0-0008, Axis configuration list"
- Check if the axis scaling is correct for the transformation of the configured axes in parameter "K-0-0008, Axis configuration list"
- Check of parameters "K-0-0040, Transformation, lengths of the mechanics" and "K-0-0041, Transformation, factors"
- Precalculation of values used in forward and inverse transformation

The content of parameters "K-0-0008, Axis configuration list", "K-0-0040, Transformation, lengths of the mechanics" and "K-0-0041, Transformation, factors" is transmitted to the method.

Transformation Schemes

If the initialization cannot be carried out successfully, the method has to return an MLC error code

- **DetermineMemory:** This method resolves ambiguities of the transformation. For this purpose, a reference to an ACS point is transmitted to the method. This point has the same ambiguous solution to be used in the next inverse transformation. Additionally, the method contains information whether the transformation may select one of the ambiguous solutions independently in the next inverse transformation.
- **Forward:** method that converts coordinates from the axis coordinate system into coordinates of the base coordinate system ([Transformations on page 23](#)).

For this purpose, the method receives two references to the coordinates of the axis coordinate system and the coordinates of the base coordinate system.

From the coordinates of the axis coordinate system the method calculates the coordinates of the base coordinate system.

If a point cannot be transformed, the method has to return an MLC error code.

- **Inverse:** method that converts coordinates from the base coordinate system into coordinates of the axis coordinate system ([Transformations, Overview on page 23](#)).

For this purpose, the method receives two references to the coordinates of the base coordinate system and the coordinates of the axis coordinate system.

From the coordinates of the base coordinate system the method calculates the coordinates of the axis coordinate system.

If a point cannot be transformed, the method has to return an MLC error code.

- **GetMechanic:** This method obtains mechanical information for a point in the ACS from the transformation.

For this purpose the method features

- A reference to a point in the ACS coordinate system
- A reference to a structure where the mechanical information is to be stored
- The number of free elements in this structure
- A reference to the value indicating the number of elements used by the structure
- And information if only changed values are to be returned

The auxiliary function [ML_AddMechanicData on page 135](#) fills in the structure

Method error handling

If a floating point error occurs at one of the methods, the error is reported automatically.

However, such errors already have to be considered in the transformation mathematics: For example, always check if a divisor is zero. If yes, report an error and do not execute the division.

- F12C0600: Invalid operation at PLC function call
- F12C0601: Denormalized operand at PLC function call
- F12C0602: Zero division at PLC function call

- F12C0603: Overflow at PLC function call
- F12C0604: Underflow at PLC function call
- F12C0605: Precision error at PLC function call

In contrast to the simple user-defined transformation, the extended user-defined transformation can return an MLC error code as return value of the functions. This facilitates diagnostics. An MLC error code has the following structure: 16#Fxxxxxxx. If an interface function was executed without error, a code with the structure 16#0xxxxxxx is returned.

The most important error codes for a transformation:

- F22B0021: Axis angles out of definition range
- F12C0479: WC zero position is singular
- F12C047A: Too close to stand
- F22B0022: Point can not reached by transformation
- F12C008A: Transformation parameter does not fit to robot type
- F22B0023: In saved range of transformation
- F22B0024: Axis has wrong scaling for transformation
- F22B0025: Too less axes for Transformation

Selecting a transformation

If a function block is implemented that represents a transformation (the function block implements interface [MC_KIN_REF_EXT on page 234](#)), this function block can be registered in the system with function block [MC_SetKinTransformExt on page 132](#).

Registration requires an instance of the transformation function block and the kinematics that the instance is to apply to.

Transformation registration has to be carried out in the configuration phase of the control (P2).

If during phase switching to the control operating state (BB) in the kinematics a user-defined transformation is defined but not registered, error "F12C0498: Transformation in PLC has not been registered" is reported.

If the same transformation is to be used in several kinematics, a separate instance of the specific transformation function block has to be used for each kinematics.

Use case

For a use case for the creation and application of user-defined transformations, see [Application Example 4: Extended User-Defined Transformation on page 321](#).

4.5 Mechanical Information

Kinematics configuration includes several mechanical data about the system. This includes the arm lengths of the kinematics or its spatial position, for example.

This data is not only required for system operation, but also to create a generic visualization of the system.

For this purpose, the system returns the mechanical information in compressed form.

The mechanical information can be read out using [Get Mechanic on page 30](#).

The following data are read:

- **ObjectID:** Text name of the object
- **Statement:** Visualization instruction to be applied to the object
- **X, Y, Z:** data for the statement

Transformation Schemes

The following statements may be used:

- **Create rod:** The object with the ObjectID is a cylinder with length 1 and radius 1
The center point of the circular area of the cylinder is in the zero point and the cylinder points in positive X direction. X, Y, Z are ignored
- **Create sphere:** The object with the ObjectID is a sphere with length 1 and radius 1
The center point of the sphere is in the zero point. X, Y, Z are ignored
- **Change position:** The new position of the object with the ObjectID is (X,Y,Z)
- **Change size:** The new size of the object with the ObjectID is (X,Y,Z)
- **Change direction:** The new direction of the object with the ObjectID is (X,Y,Z)

[Get Mechanic on page 30](#) may specify in addition if all data or only changed data is to be returned (updateOnly).

This can greatly reduce the amount of data to be transmitted.

Usually, all data is requested once, afterwards only changed data is transmitted.

Example [Get Mechanic on page 30](#) returns the following data:

Example of mechanical data

ObjectID	Statement	x	y	z	Description
TCP	Sphere				There is a sphere called "TCP"
S1	Rod				There is a rod called "S1"
S2	Rod				There is a rod called "S2"
K12	Sphere				There is a sphere called "K12"
TCP	Size	3	3	3	Change the size of "TCP" to (3,3,3)
S1	Size	100	2	2	Change the size of "S1" to (100,2,2)
S2	Size	100	2	2	Change the size of "S2" to (100,2,2)
S1	Direction	70.7	70.7	0	Change the direction of "S1" to (70.7,70.7,0)
S2	Direction	-70.7	-70.7	0	Change the direction of "S2" to (-70.7,-70.7,0)
S2	Position	70.7	70.7	0	Change the position of "S2" to (70.7,70.7,0)
K12	Position	70.7	70.7	0	Change the position of "K12" to (70.7,70.7,0)
TCP	Position	141.4	0	0	Change the position of "TCP" to (141.4,3,3)

Tab.4-39: Example of mechanical data

This data can be visualized as follows:

Transformation Schemes

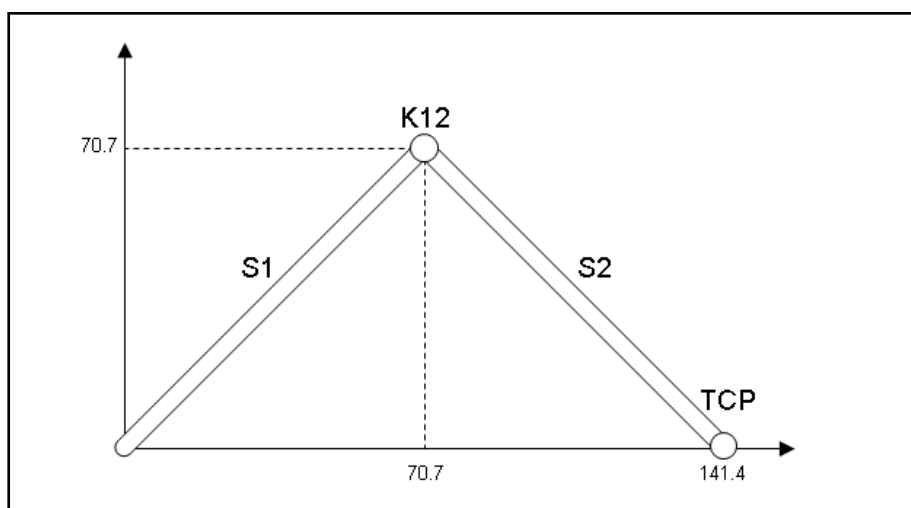


Fig.4-30: Visualized data of GetMechanic

If the function [Get Mechanic on page 30](#) is called with "updateOnly", it only returns data of variable statements:

Example of mechanical data

ObjectID	Statement	x	y	z	Description
S1	Direction	70.7	70.7	0	Change the direction of "S1" to (70.7,70.7,0)
S2	Direction	-70.7	-70.7	0	Change the direction of "S2" to (-70.7,-70.7,0)
S2	Position	70.7	70.7	0	Change the position of "S2" to (70.7,70.7,0)
K12	Position	70.7	70.7	0	Change the position of "K12" to (70.7,70.7,0)
TCP	Position	141.4	0	0	Change the position of "TCP" to (141.4,3,3)

Tab.4-40: Example of mechanical data

5 ML_Robot.library

5.1 Overview

The ML_Robot library contains the relevant data types, functions and function blocks for motion control (Coordinated Motion) required to implement the kinematic functions of the IndraMotion MLC system.

Some parts of the ML_Robot library deviate for each control type.

Therefore, the name of the "ML_Robot" library is extended by the precise reference, e.g. "ML_Robot_L25" for IndraMotion MLC, control CML25.



This documentation only contains the interface description for the ML_Robot library.

For the description of the range of interface functions, see [Functional Description on page 19](#).

Library structure:

- Data types of the ML_Robot library, Data Types ([Data Types, Overview on page 229](#)):

Data types

- **Config:** Function block data types for configuration of a kinematics
- **Info:** Function block data types for information acquisition of a kinematics
- **Jog:** Function block data types for manual operation (Jog)
- **Movement:** Function block data types for kinematics movement
- **Parameters:** Function block data types for parameter access to a kinematics
- **RCL:** Function block data types for kinematics program control

- Definitions of global variables ([Global Variables on page 108](#)):

Global Variable List

- Shared Memory (**GVL_Robot**)
- Global constants (**GVL_RobotConstant**)
- Direct variables

- Internal (Do not use!): Internal definitions
- Function blocks for free use ([Function Blocks, Overview on page 112](#)):

Kinematics

- **Config:** Function blocks for configuration of a kinematics
- **Info:** Function blocks for kinematics information
- **Movement:** Function blocks for movements of a kinematics
- **Parameters:** Function blocks for parameter access to a kinematics
- **RCL:** Function blocks for program control of a kinematics using Robot Control Language (RCL), only to be used in Robot-Control V1
- **TeachIn:** Function blocks for saving and reusing kinematic positions as points, only to be used in Robot-Control V2

- **Project information:** Functions for information access about this library

ML_Robot.library

5.2 Global Variables

5.2.1 Overview

When generating a new kinematics in IndraWorks, the device name and the corresponding drives are queried.

A kinematic number is automatically assigned to the kinematics.

Using the automatically generated kinematic number

MB_MIN_KIN_NUMBER . . MB_MAX_KIN_NUMBER

(see [Global Constants on page 109](#))

it is possible to access indexed kinematic data via

- **Function variable access** (see [DV_Kin\[\] - Kinematic Data via Function Variable Access on page 110](#))
- **Shared memory access** (see [KinData\[\] - Kinematic Data in the Shared Memory on page 109](#))



Kinematic data in shared memory is updated consistently and cyclically in the Motion mode "operating mode". Due to the required update time, its scope is limited.

Kinematic data via function variable access (also "direct variables") is updated irrespectively of the Motion mode at the time of access.

Kinematic data on shared memory and function variables may be distributed over several Motion cycles (command value generation, actual value provision), depending on the task properties of the POU. Therefore, it is not necessarily consistent.

Part of the kinematic data is available in the shared memory and via the function variable access.

Since kinematic data in the shared memory is cyclically exchanged irrespective of its use, further redundant access via direct variables should be avoided due to performance reasons.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable. We recommend avoiding multiple access to a function variable and to save the operating date to a local PLC variable, if required.

Kinematic data on shared memory and function variable access may be distributed over several Motion cycles (command value generation, actual value provision), depending on the task properties of the POU. Therefore, it is not necessarily consistent.



A function variable is not part of the process image of the PLC task. The simplified, blocking access to a function variable is an implicit function call that is executed at the time of call-up. Multiple access to a function variable can cause different results in a PLC program (POU) as the operating date can be modified due to task changes.

5.2.2 Global Constants

The list of global constants "GVL_RobotConstant" defines the valid constants for the respective control hardware.

Library	Range
ML_Robot_yyy ¹⁾	Global Variable List

Tab.5-1: Library assignment

Variable name	Type	Description	MLC		
			L25	L45	L65
MB_MIN_KIN_NUMBER	INT	Smallest kinematic number	1		
MB_MAX_KIN_NUMBER	INT	Greatest kinematic number	4	16	16
MB_NUMBER_OF_KIN	INT	Maximum number of kinematics	4	16	16
MB_MAX_KIN_AXIS_NUMBER	INT	Maximum number of axes per kinematics	16		
MB_MAX_COORDINATES_NUMBER	INT	Maximum number of coordinates in a Cartesian point	6		
ML_MAX_KIN_ARRAY_DIMENISINS	INT	Maximum number of dimensions for a point array	5		
ML_MAX_IDX_ARRAY_USINT	UDINT				
KIN_OBJ_2_OBJECT	Array [0..16] of OB- JECTS		[0,1,2,3,4,5,6,7,8 ,9, 10,11,12,13,14,1 5,16]		

Tab.5-2: Global constants of the ML_Robot library

5.2.3 KinData[] - Kinematic Data in the Shared Memory

Brief description Kinematic data in shared memory is declared as the global array "KinData[]" of the "ML_KINEMATICSDATA_SM" data structure. The definition is part of the Global Variable List "GVL_Robot" in the "ML_Robot" library.

Definition of the global array KinData[]

VAR_GLOBAL

```
KinData AT %MBaaa: ARRAY [MB_MIN_KIN_NUMBER..MB_MAX_KIN_NUMBER] OF ML_KINEMATICSDATA_SM;
END_VAR
```



Please find the addresses "%MBaaa" of KinData in the library manager of ML_Robot_yyy¹⁾, GVL_Base, documentation.

Also refer to

- [ML_KINEMATICSDATA_SM on page 237](#)

Functional description The read-only access to data in the shared memory can be carried out via the following equivalent statements (ST).

¹⁾ yyy: L25, L45 or L65

ML_Robot.library

Read-only access to data in the shared memory:

```
MyVar:= KinData[MyKinName.GroupNo].Kin_InPosition;
MyVar:= KinData[Index].Kin_InPosition;
MyVar:= KinData[1].Kin_InPosition;
```

The write access to data in the shared memory can be carried out via the following equivalent statements (ST).

Writing end Access to data in the shared memory:

```
KinData[MyKinName.GroupNo].stJogCmd_q:= MyVar;
KinData[Index].stJogCmd_q:= MyVar;
KinData[1].stJogCmd_q:= MyVar;
```

The IntelliSense of a variable in the shared memory contains the parameter ident number and if necessary the bit, access possibilities (RD or RW), parameter names and bit description.

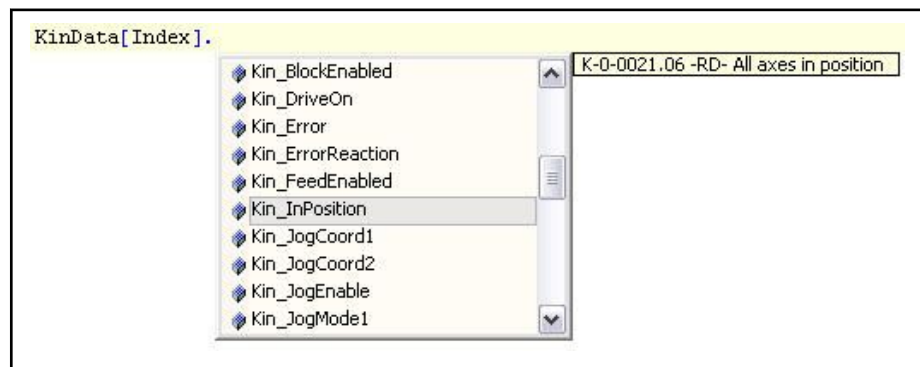


Fig.5-1: IntelliSense of a variable in the shared memory



Variables of a kinematics in the shared memory can also be accessed by using references via a user-defined name.

User-defined Naming by referencing:

```
VAR
  MyKinName: REFERENCE TO ML_KINEMATICS_DATA_SM:= KinData[MyKinName.GroupNo];
END_VAR
```

```
MyVar:= MyKinName.Kin_InPosition;
MyKinName.stJogCmd_q:= MyVar;
```

5.2.4 DV_Kin[] - Kinematic Data via Function Variable Access

Kinematic data via function variable access is declared as a global array "DV_Kin[]" of the function block "ML_DirectVarKin".

The definition is part of the Global Variable List "GVL_Robot" in the "ML_Robot" library.

The function block "ML_DirectVarKin" provides access to direct variables using methods. The GET and SET properties of the respective method define the access option. Executing the write-access SET on a function variable depends on the motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarKin" is not intended for direct use in the user program.

Extract from GVL_Robot

```
VAR_GLOBAL
...
  DV_Kin: ARRAY [MB_MIN_KIN_NUMBER..MB_MAX_KIN_NUMBER] OF ML_DirectVarKin;
...
END_VAR
```

Functional description Read-only access to a direct variable can be carried out via the following equivalent statements (ST).

Read-only access to a function variable

```
MyVar := DV_Kin[MyKinName.GroupNo].K_0_0022;
MyVar := DV_Kin[Index].K_0_0022;
MyVar := DV_Kin[1].K_0_0022;
```

Write access to a function variable can be carried out via the following equivalent statements (ST).

Write access to a function variable

```
DV_Kin[MyKinName.GroupNo].K_0_0241 := MyVar;
DV_Kin[Index].K_0_0241 := MyVar;
DV_Kin[1].K_0_0241 := MyVar;
```

The IntelliSense of a function variable contains the parameter ID, the access possibilities (READ or READ/WRITE) and the parameter name.

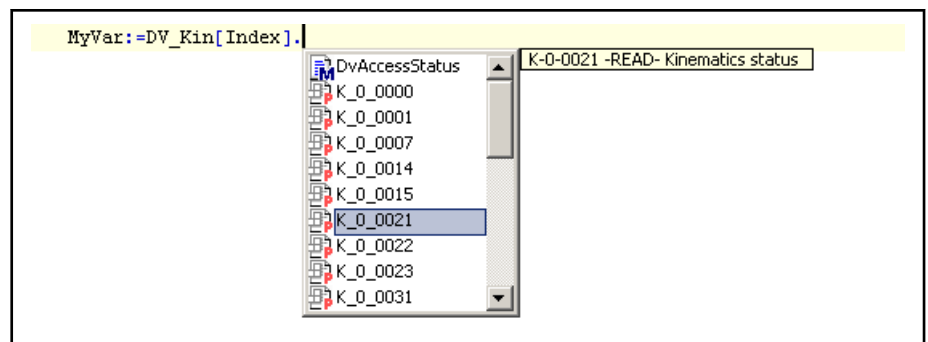


Fig. 5-2: Function variable access using the IntelliSense functionality



Although access to function variables looks like simple variable access, it is a complex function call that takes a lot more time than variable access.



Direct variables of a kinematic can also be accessed by using references via a user-defined designation.

Access via REFERENCE TO

```
VAR
  MyKinName: REFERENCE TO ML_DirectVarKin := DV_Kin[MyKinName.GroupNo];
END_VAR

MyVar := MyKinName.K_0_0022;
MyKinName.K_0_0241 := MyVar;
```

Status of parameter values in the case of PLC access using direct variables

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

ML_Robot.library

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
K_Status:= DV_Kin[Index].DvAccessStatus(FP_K_0_0015);
```

Declaration files for Global Constants

- [Accessing Kinematic Data on page 189](#)

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: sercos error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

5.3 Function Blocks

5.3.1 Function Blocks, Overview



All function blocks that are only valid for Robot Control V1 have to be removed from Robot Control V2 programs.

If this is not done and a function block is called that is not valid for Robot Control V2, this function block returns the respective error message and the program may be interrupted.

The following function blocks are supported by the respective kinematic type:

Function block	Robot Control V1	Robot Control V2
Config		
MC_GroupReset on page 128	x	x
MC_SetCartesianTransform on page 129		x
MC_SetKinTransform on page 131	x	x
MC_SetKinTransformExt on page 132		x
ML_AddMechanicData on page 135		x
MC_UngroupAllAxes on page 133	x	x
ML_GroupAllAxes on page 135	x	x
ML_GroupSetSafeZone on page 137		x
ML_KinSetDistanceBlendOption on page 138	x	
ML_SetBeltConfiguration on page 140		x

Function block	Robot Control V1	Robot Control V2
ML_SetBeltErrorReaction on page 142		x
ML_SetBeltDesyncRelConfiguration on page 139		x
ML_SetBeltParameter on page 143	x	
ML_SetCylindricTransform on page 145		x
ML_SetRotaryTableConfiguration on page 147		x
Info		
MC_GroupReadActualPosition on page 148		x
ML_KinGetDistanceBlendOption on page 149	x	
ML_RobotGetBeltDesyncRelConfiguration on page 151		x
ML_RobotGetBeltConfiguration on page 150		x
ML_RobotGetBeltErrorReaction on page 152		x
ML_RobotGetMechanic on page 153		x
ML_RobotGetPerformance on page 155		x
ML_RobotGetTrafoValues on page 156		x
ML_RobotMovePoint on page 157		x
ML_RobotReadCyclicPos on page 158		x
ML_RobotSetPerformanceEnable on page 159		x
ML_RobotTransformPoint on page 160		x
ML_SwitchOnPath on page 162	x	x
Movement		
MC_GroupContinue on page 163		x
MC_GroupInterrupt on page 164		x
MC_GroupStop on page 165		x
MC_MoveCircularAbsolute on page 166	x	x
MC_MoveCircularRelative on page 169		x
MC_MoveDirectAbsolute on page 171	x	x
MC_MoveDirectRelative on page 173		x
MC_MoveLinearAbsolute on page 175	x	x
MC_MoveLinearRelative on page 177	x	x
ML_BeltDeSynchronisationWithLimits on page 179		x
ML_BeltSynchronisationWithLimits on page 180		x
ML_GroupJogCont on page 181		x
ML_GroupJogStep on page 183		x
ML_GroupWait on page 186		x
ML_RobotOpenCyclicPos on page 187		x
ML_RobotWriteCyclicPos on page 188		x

ML_Robot.library

Function block	Robot Control V1	Robot Control V2
Parameters		
MB_GroupReadListParameter on page 190	x	x
MB_GroupReadParameter on page 191	x	x
MB_GroupReadRealParameter on page 192	x	x
MB_GroupReadSercosAttribute on page 193	x	x
MB_GroupReadSercosDataStatus on page 194	x	x
MB_GroupReadStringParameter on page 195	x	x
MB_GroupWriteListParameter on page 196	x	x
MB_GroupWriteParameter on page 198	x	x
MB_GroupWriteRealParameter on page 199	x	x
MB_GroupWriteStringParameter on page 199	x	x
RCL\ RCL program		
MB_IRDProgSelect on page 201	x	
MB_IRDProgStart on page 202	x	
MB_IRDProgStop on page 202	x	
ML_RoCoDir on page 203	x	
ML_RoCoGetPoint on page 206	x	
ML_RoCoGetPointList on page 209	x	
ML_RoCoSetPoint on page 211	x	
RCL\ RCL user interface		
ML_KinGetUserBit on page 215	x	
ML_KinGetUserBit_DWORD on page 215	x	
ML_KinGetUserDInt on page 216	x	
ML_KinGetUserReal on page 217	x	
ML_KinSetUserBit on page 218	x	
ML_KinSetUserDInt on page 219	x	
ML_KinSetUserPoint on page 220	x	
ML_KinSetUserReal on page 220	x	
TeachIn		
ML_GroupDeletePoints on page 222		x
ML_GroupLoadPointFile on page 223		x
ML_GroupDeletePointFile on page 221		x
ML_GroupReadNextPoint on page 224		x
ML_GroupReadPoint on page 225		x

Function block	Robot Control V1	Robot Control V2
ML_GroupSavePointFile on page 226		x
ML_GroupWritePoint on page 228		x

Tab.5-3: Type of robot and supported function blocks

5.3.2 Standard Interfaces at Function Blocks

Motivation

The majority of the function blocks disposes of an input for the activation and an output for displaying the correct processing.

In addition, an output is often necessary which indicates the processing time.

Additional outputs are defined for the indication of errors.

A uniform naming convention as well as an identical standard interface behavior, increases understanding, shortens the time needed for familiarization and reduces the workload of the support.

Function Block Types

With function blocks, it is possible to encapsulate complex tasks so they can be reused and to address them via defined interfaces. For this, the processing can be either state-controlled or edge-controlled. Furthermore, a difference is made between tasks which can be processed completely and tasks which require continuous intervention once they are started.

State-controlled	If a function block always repeats its task after switch-on whenever it has reached a defined state, this FB processes in a "state-controlled" way.
Edge-controlled	If a function block fulfills its task exactly once after being switched on, this FB processes in an "edge-controlled" way.
Terminating processing	If a block can definitively process its order and afterwards stop working, this is called "definitive processing".
Permanent processing	If a block can never conclude its order definitively but is continuously in engagement, this is defined as "continuous processing".

In combination, there are four FB types possible:

Control	Processing	Example
State-controlled (Enable)	Terminating (Done)	Permanent reading of a parameter. (if the reading is completed, the function block has completed its task. It repeats this task as long as the control input (Enable) remains set.); e.g. MB_ReadParameter.
State-controlled (Enable)	Permanent (In....)	Terminable control. (If the control is active, it processes until the control input (Enable) is reset and thus deactivates the control.)
Edge-controlled (Execute)	Terminating (Done)	One-time writing of a parameter. (If the writing is completed, the FB has completed its task. A new edge is required at the control input (Execute) in order to start new writing.) e.g. MB_WriteParameter
Edge-controlled (Execute)	Permanent (In....)	Non-terminable control. (Once the control is started via the control input (Execute), it cannot be deactivated via the interfaces of this FB anymore.) e.g. MC_MoveVelocity

Tab.5-4: Overview on FB types

ML_Robot.library

Function Block Types and their Interfaces

Inputs In order to mark whether a function block processes state-controlled or edge-controlled, two different variable names are used for the inputs for the activation of the function block.

- Enable = state-controlled
- Execute = edge-controlled

Outputs In order to mark whether a function block processes in a terminating way or permanent, different variable names are used for the outputs which indicate the processing state.



- Done = Terminating processing
- In..... = continuous processing (e.g., InOperation, InSync, InVelocity, InGear, InTorque...)

The optional "Active" output can be used with terminating processing and permanent processing FBs behaving differently!

All outputs for the error identification (Error, ErrorID, ErrorIdent) can be found with terminating processing and permanent processing FBs behaving identically.

Standardized identifiers

Identifier	Description
Inputs	
Enable	Enable input of state-controlled FBs
Execute	Enable input of edge-controlled FBs. Other FBs can interrupt this.
ExecuteLock	Enable input of edge-controlled FBs. As long as this input is TRUE, the FB cannot be interrupted by others.
Outputs	
Active	Output marking the processing time
Done	Processing completed successfully and the data outputs are valid.
In.....	Output signalizes that the FB is processing its task and that the data outputs are valid.
Shutdown (optional)	The output signalizes that the FB is currently reaching a defined final state (e.g., it stops axes, releases resources, ...). The FB must be called as long as necessary for "Shutdown" to become FALSE.
CommandAborted	Output indicates that the FB was interrupted (e.g. by another FB).
Error	Process completed with error
ErrorID	Output for rough error classification
ErrorIdent	Output for detailed error classification

Tab.5-5: Overview on standardized identifiers

Inputs and Outputs of State-controlled Function Blocks, Terminating Processing

Control/ Processing	I/O	Variable name	Description
State-con- trolled/termi- nating	E	Enable	The input variables are registered with a positive edge at "Enable". New input values do not become effective until there is the next positive edge at "Enable". "Enable" has to remain TRUE while the function block is processed! If "Enable" is disabled, processing is interrupted and the "Done", "Active", "CommandAborted", and "Error" outputs are set to FALSE. If required, "Shutdown" signalizes that further signals are necessary for the FB to reach its final state defined.
	A	Done	If "Done" is TRUE, the function block has successfully completed its order and is then in a final state. Data outputs are valid now. "Active", "Error" and "CommandAborted" are FALSE! "Done" remains TRUE for exactly one PLC cycle. If "Enable" is still TRUE, the FB starts processing again.
	A	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediary state. Possible preprocessing are not marked with this output! "Done", "Error" and "CommandAborted" are FALSE!
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time when "Enable" is deactivated in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	CommandAborted	If "CommandAborted" is TRUE, the function block has been interrupted and is now in a final state. "Done", "Active" and "Error" are FALSE! "CommandAborted" remains TRUE until control input "Enable" is disabled. An additional reset input is not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block has been interrupted due to an error and is now in a final state. "Error" remains TRUE until the control input "Enable" is disabled. An additional reset input is not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab.5-6: I/O of state-controlled function blocks, terminating processing



If specific inputs must be applied not only with an edge at "Enable" but cyclically as long as the function block is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names.



If a function block is provided with "Shutdown", the behavior of this FB when "Enable" is set to TRUE while "Shutdown" remains set (retrigger), must be precisely defined.

ML_Robot.library

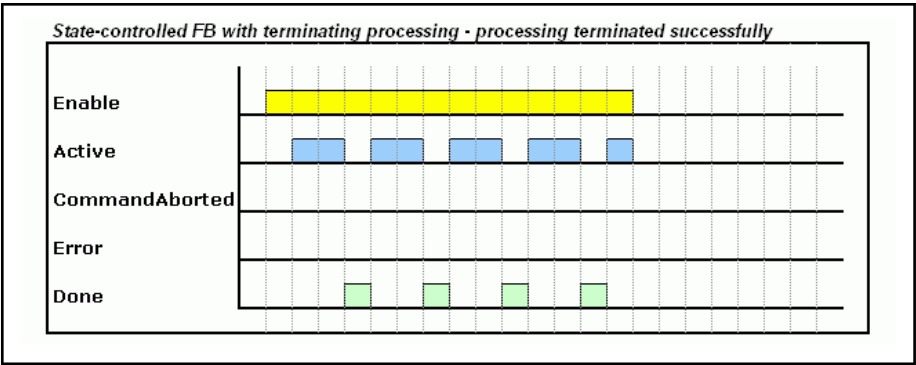


Fig.5-3: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing successfully completed

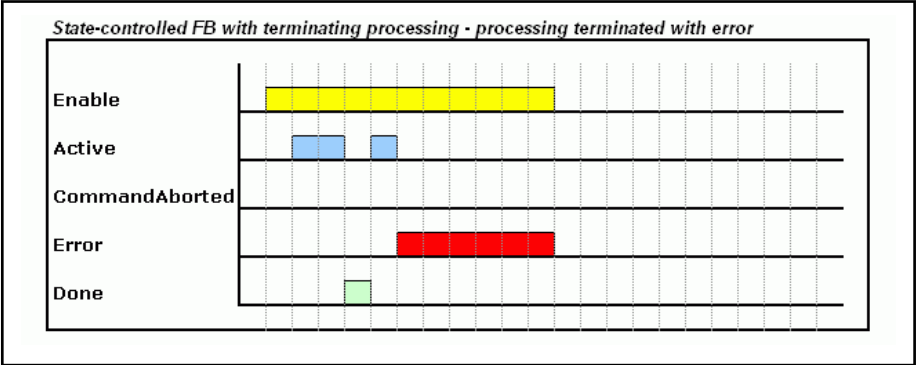


Fig.5-4: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing completed with error

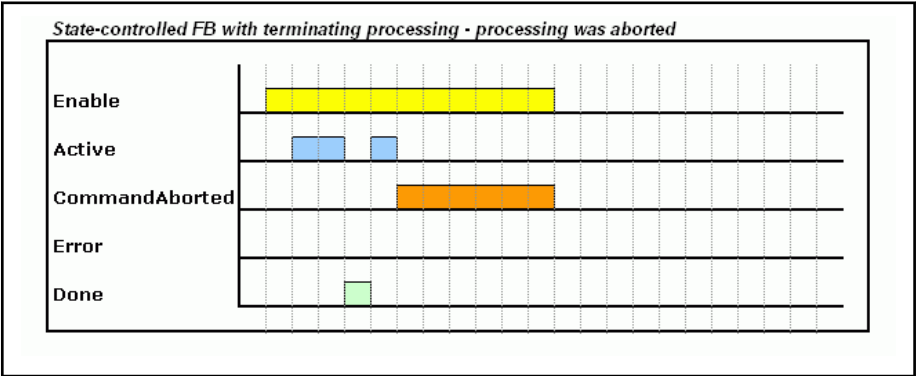


Fig.5-5: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing interrupted

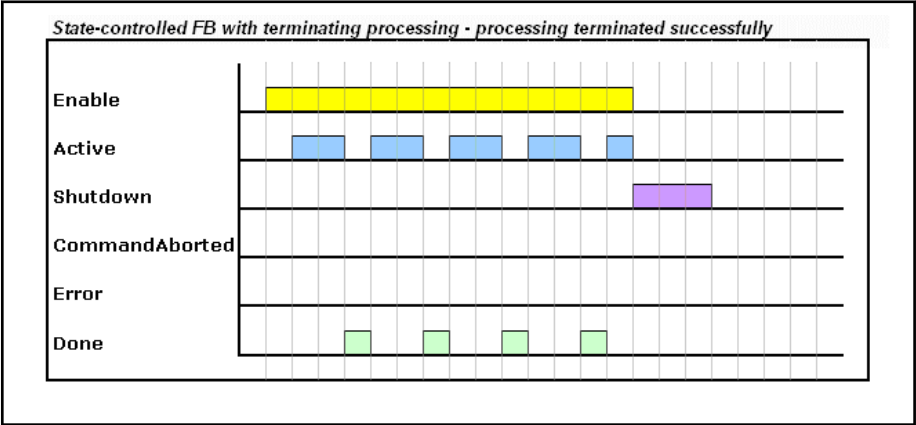


Fig.5-6: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing successfully completed.

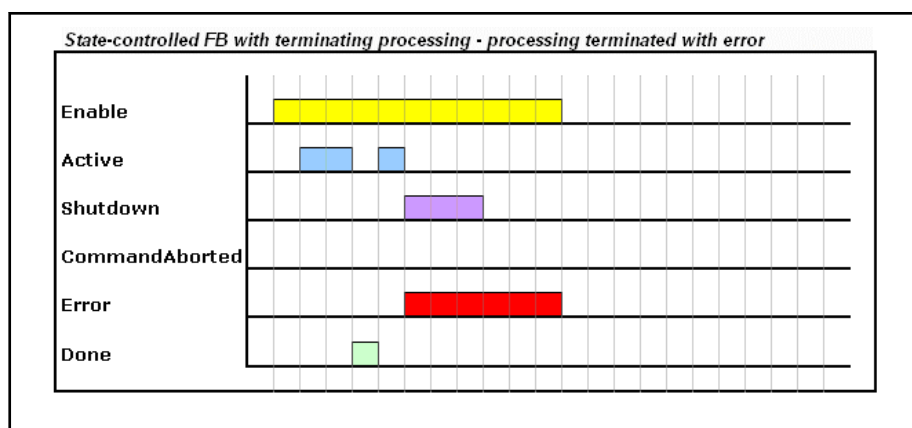


Fig.5-7: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing completed with errors.

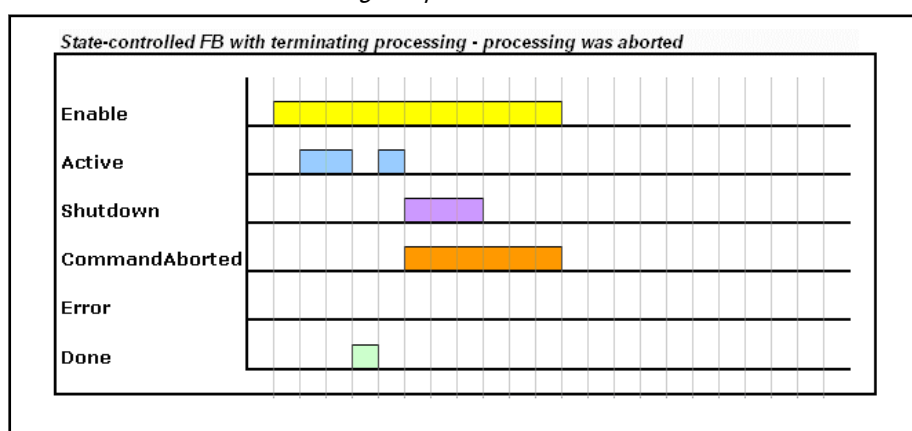


Fig.5-8: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing interrupted.

ML_Robot.library

Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing

Control/ Processing	I/O	Name	Description
Edge-controlled/terminating	E	Execute	The input variables are registered with a positive edge at "Execute". New input values do not become effective until there is the next positive edge at "Execute". The change of edges at "Execute" is sufficient to start the FB. The state of "Execute" is irrelevant for further processing. With a new edge change during the processing, the previous task is discarded, the inputs are taken over again and the task is continued with the new values (retriggering, subsequent triggering). If a "Shutdown" output is available, retriggering is perhaps not supported. The FB documentation brings clarity.
	A	Done	If "Done" is TRUE, the function block has successfully completed its order and is then in a final state. Data outputs are valid now. "Active", "Error" and "CommandAborted" are FALSE! If control input "Execute" is FALSE when the order is completed, "Done" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Done" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Done". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediary state. Possible preprocessing are not marked with this output! "Done", "Error", "Shutdown" and "CommandAborted" are FALSE!
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time after having completed its task (signalized via "Done") in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	CommandAborted	If "CommandAborted" is TRUE, the function block has been interrupted and is now in a final state. "Done", "Active" and "Error" are FALSE! If control input "Execute" is FALSE when the order is completed, "CommandAborted" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "CommandAborted" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block has been interrupted due to an error and is now in a final state. If control input "Execute" is FALSE when the error occurs, "Error" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Error" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab.5-7: I/O of state-controlled function blocks, terminating processing



If certain inputs are applied not only with the 0/1 edge at "Execute", but cyclically as long as the FB is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this FB when an edge is redetected at "Execute" while "Shutdown" remains set (retrigger), must be precisely defined.

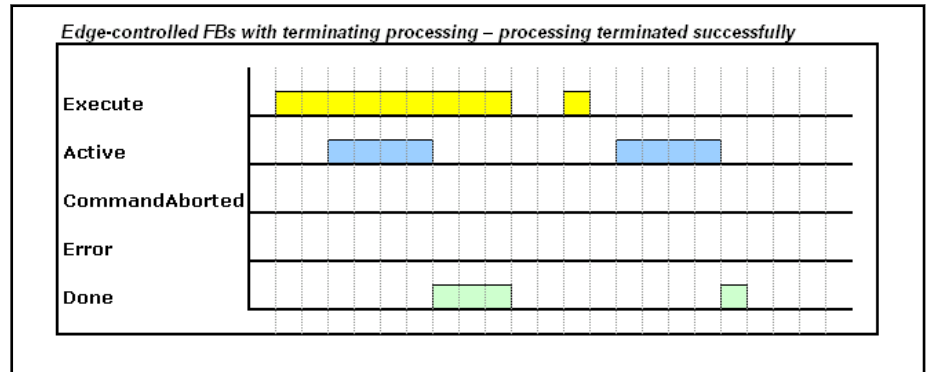


Fig. 5-9: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing successfully completed

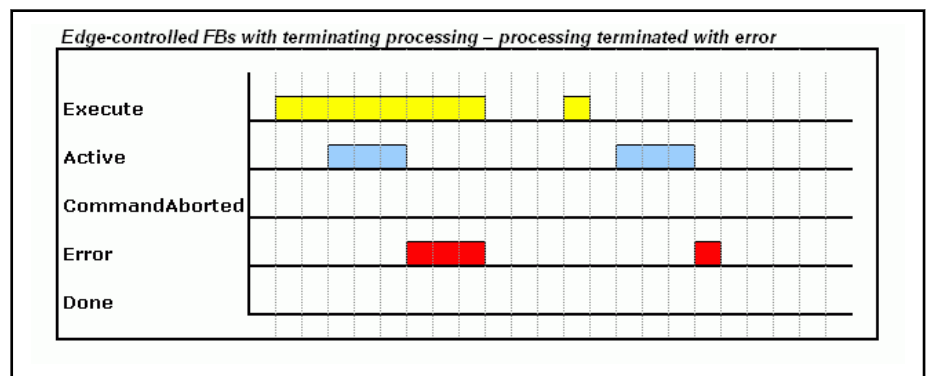


Fig. 5-10: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing completed with error

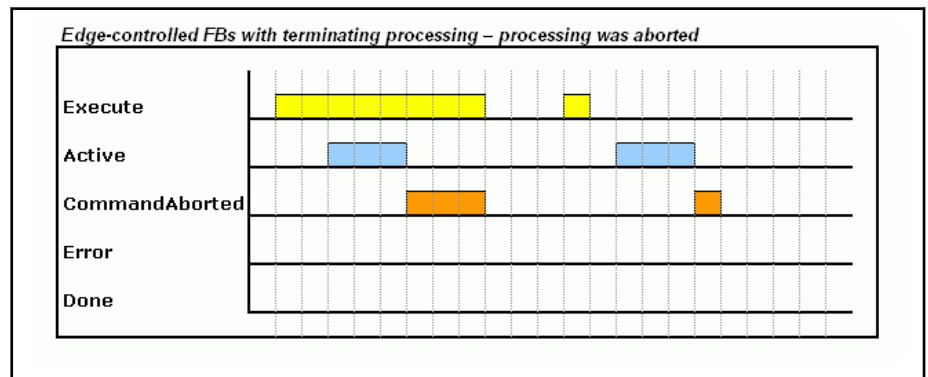


Fig. 5-11: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing interrupted

ML_Robot.library

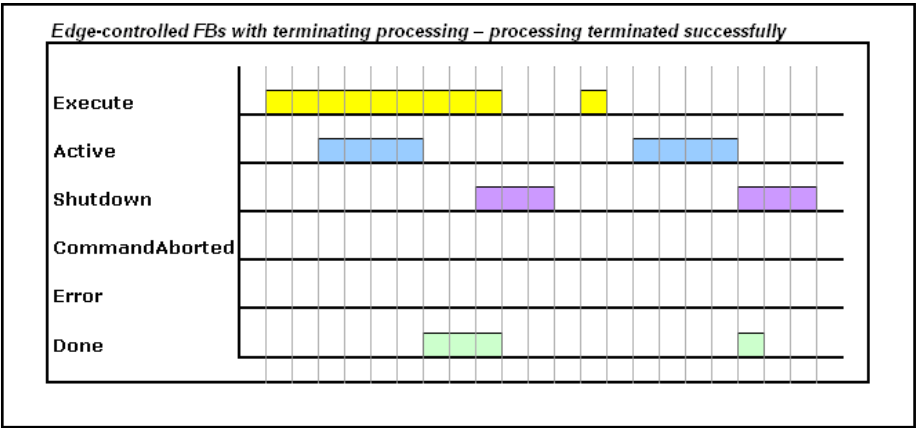


Fig.5-12: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing successfully completed

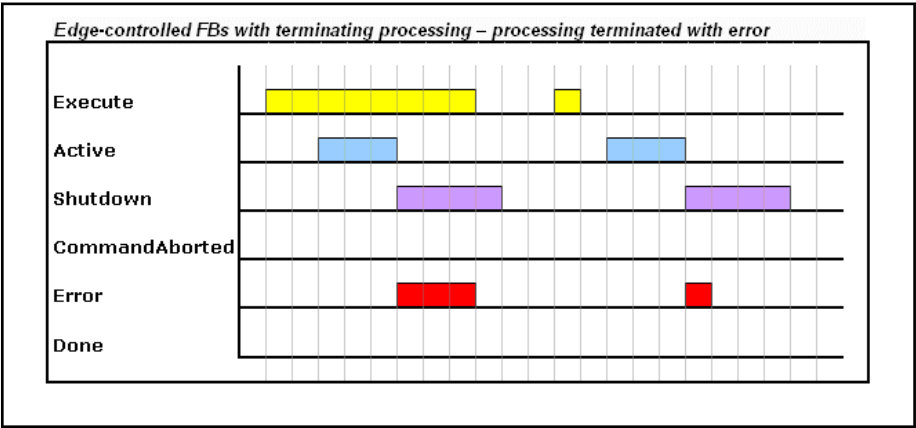


Fig.5-13: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing completed with error

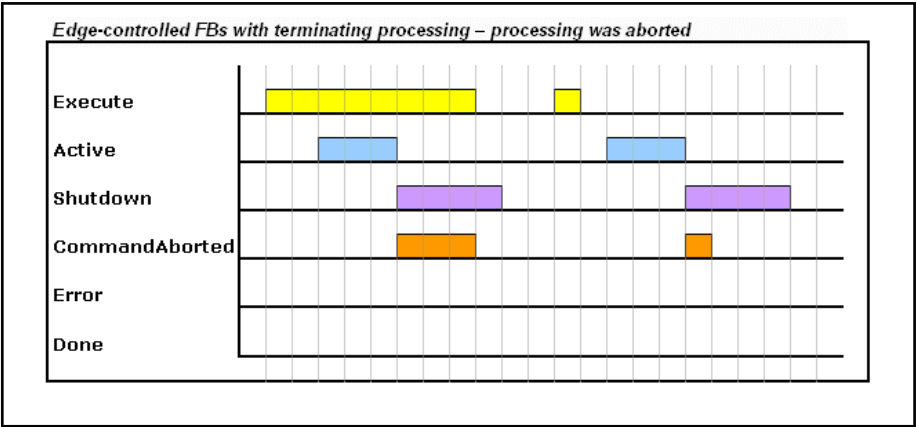


Fig.5-14: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing interrupted

Inputs and Outputs of State-controlled Function Blocks, Permanent Processing

Control/ Processing	I/O	Variable name	Description
State-controlled/ permanent	E	Enable	The input variables are registered with a positive edge at "Enable". New input values do not become effective until there is the next positive edge at "Enable". "Enable" has to remain TRUE while the function block is processed! If "Enable" is disabled, processing is interrupted and the "In...", "Active", "CommandAborted" and "Error" outputs are set to FALSE If required, "Shutdown" signalizes that further signals are necessary for the FB to reach its final state defined.
	A	In....	If "In...." is TRUE, the function block has reached its goal but is still in engagement in order to "keep" what it has achieved and is thus in a continuous final state. Data outputs are valid now. Since the FB remains in engagement, "Active" must also remain TRUE. "Error" and "CommandAborted" are FALSE! As long as "Enable" is TRUE, "In...." remains TRUE as well.
	A	Active	If "Active" is TRUE, the function block processes its actual task. Possible pre-processing are not marked with this output! Since the FB remains in continuous engagement, "Active" remains TRUE until the FB is switched off via control input "Enable" or until it is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" must be FALSE.
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time when "Enable" is deactivated in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	CommandAborted	If "CommandAborted" is TRUE, the function block has been interrupted and is now in a final state. "In...", "Active" and "Error" are FALSE! "CommandAborted" remains TRUE until the control input "Enable" is disabled. An additional reset input is not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block was canceled due to an error and is now in a final state. "Error" remains TRUE until control input "Enable" is disabled. An additional reset input is not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab.5-8: I/O of state-controlled function blocks, terminating processing



If specific inputs need to be applied not only with an edge at "Enable" but cyclically while the function block is being processed, this has to be explicitly documented. It is not sufficient to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this function block when "Enable" is set to TRUE while "Shutdown" remains set (retrigger), has to be precisely defined.

ML_Robot.library

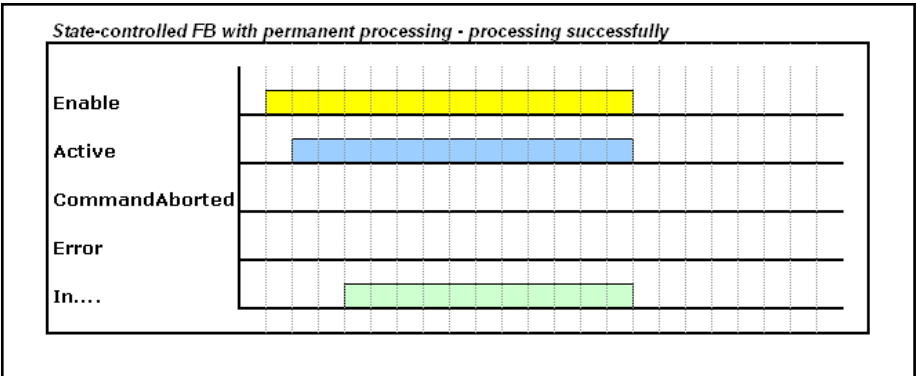


Fig.5-15: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing successful

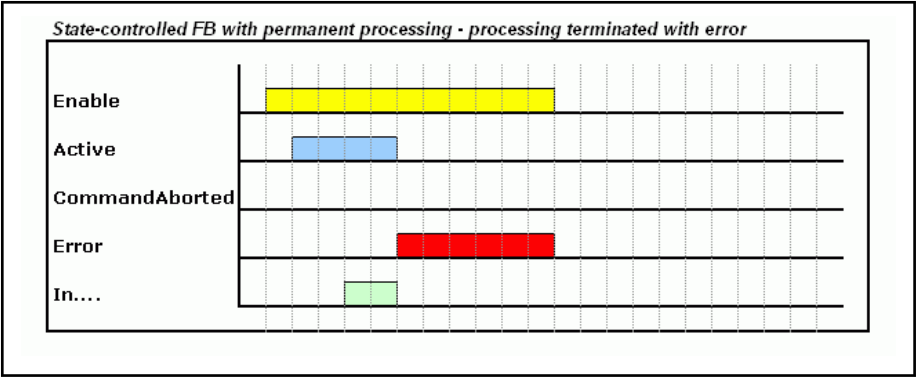


Fig.5-16: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing aborted with errors.

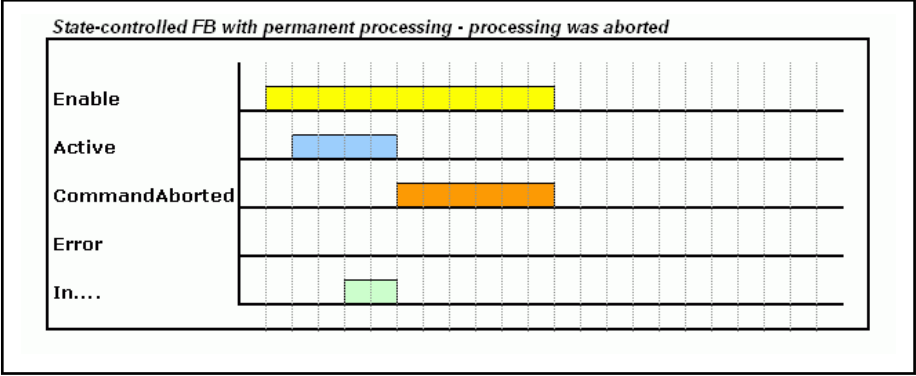


Fig.5-17: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing interrupted

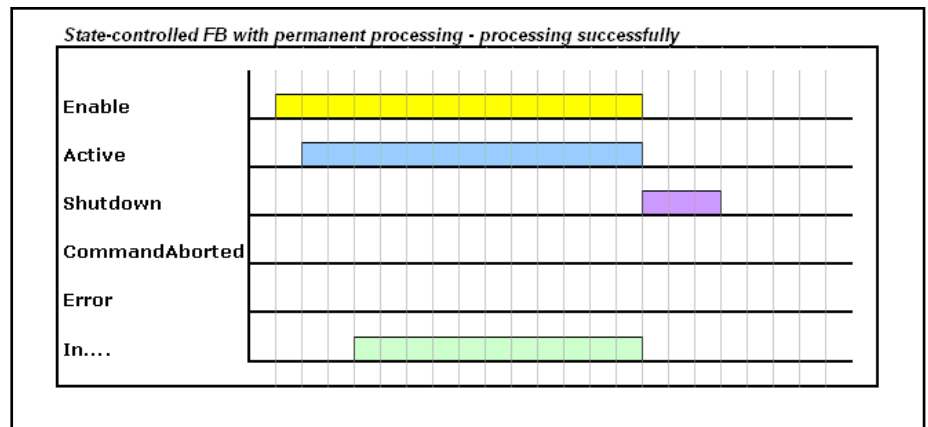


Fig.5-18: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing successful

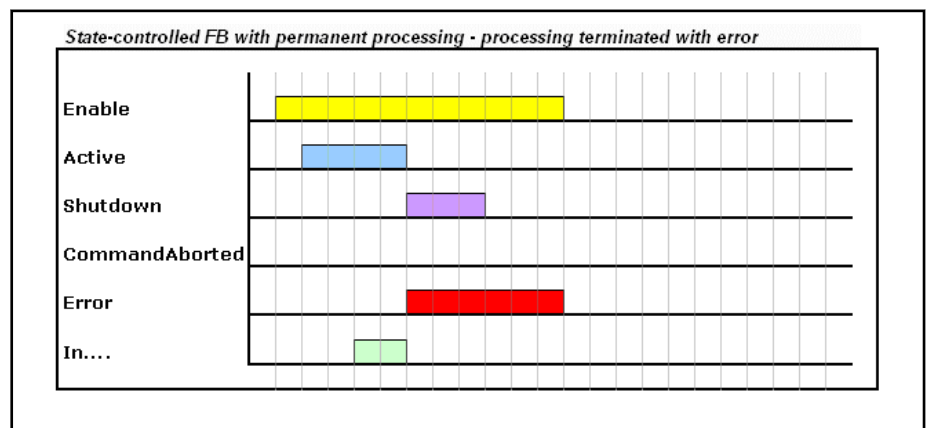


Fig.5-19: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing aborted with errors.

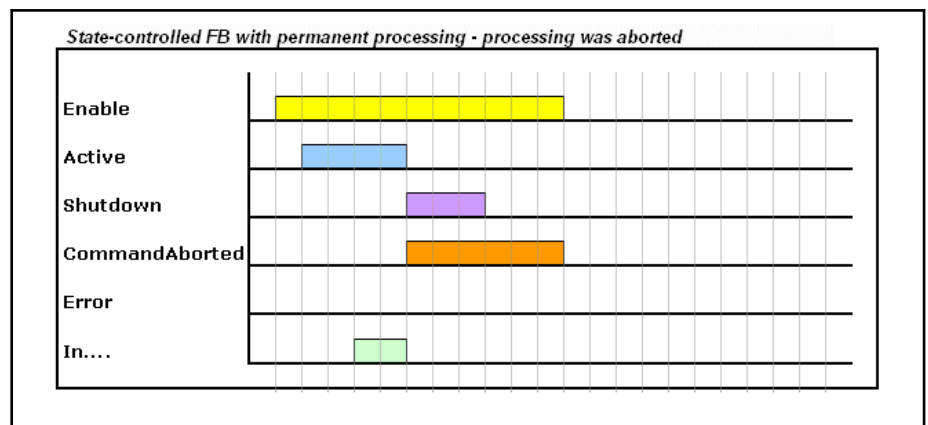


Fig.5-20: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing interrupted

ML_Robot.library

Inputs and Outputs of Edge-controlled Function Blocks, Permanent Processing

Control/ Processing	I/O	Variable name	Description
Edge-controlled/ permanent	E	Execute	The input variables are registered with a positive edge at "Execute". New input values do not become effective until there is the next positive edge at "Execute". The edge change at "Execute" is sufficient to start the function block. The state of "Execute" is irrelevant for further processing. With a new edge change during processing, the previous task is discarded, the inputs are applied again and the task is continued with the new values (retriggering, subsequent triggering). If the FB is provided with a "Shutdown" output, retriggering is perhaps not allowed.
	A	In....	If "In...." is TRUE, the function block has reached its goal but is still active in order to keep the results and is thus in a permanent final state. Data outputs are valid now. Since the FB remains in engagement, "Active" must also remain TRUE. This even applies when "In...." drops again. "Error" and "CommandAborted" are FALSE! If control input "Execute" is FALSE when the order is completed, "In..." remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "In..." remains TRUE until "Execute" is disabled.
	A	Active	If "Active" is TRUE, the function block processes its actual task. Possible pre-processing are not marked with this output! Since the FB remains in continuous engagement, "Active" remains TRUE until the FB is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" as well as "Shutdown" must be FALSE.
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time after having aborted its task ("CommandAborted" or "Error" = TRUE) in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	CommandAborted	If "CommandAborted" is TRUE, the function block has been interrupted and is now in a final state. "In...", "Active" and "Error" are FALSE! If control input "Execute" is FALSE when the order is completed, "CommandAborted" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "CommandAborted" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block has been interrupted due to an error and is now in a final state. If control input "Execute" is FALSE when the error occurs, "Error" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Error" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab.5-9: I/O of edge-controlled function blocks, permanent processing



If certain inputs are applied not only with the 0/1 edge at "Execute", but cyclically as long as the FB is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this FB when an edge is redetected at "Execute" while "Shutdown" remains set (retrigger), must be precisely defined.

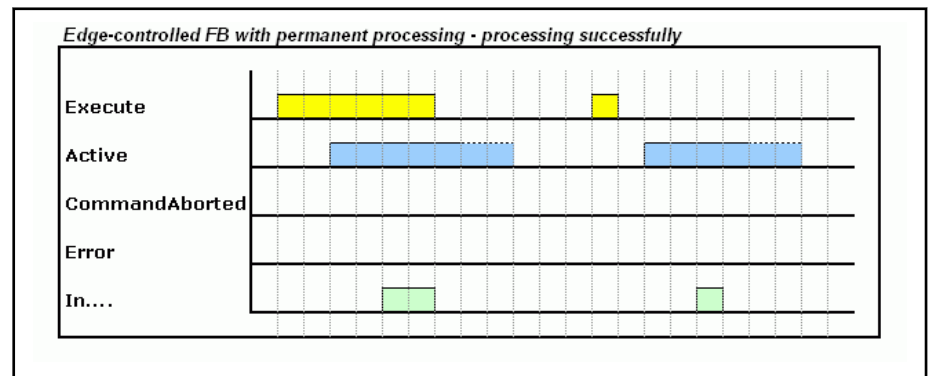


Fig. 5-21: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing successful

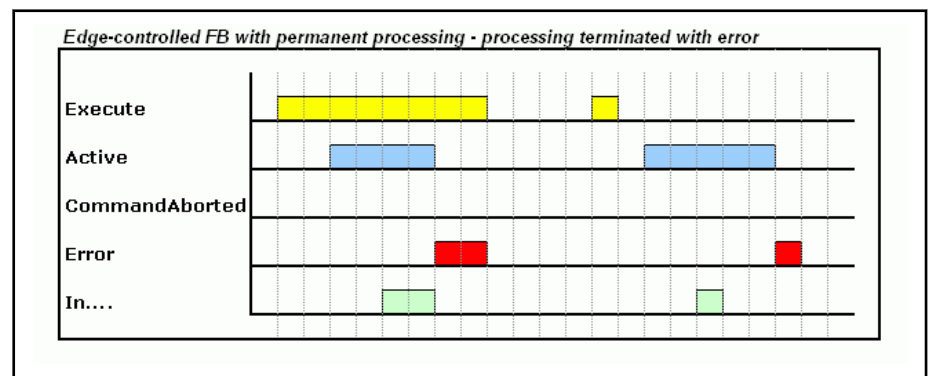


Fig. 5-22: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing aborted with errors.

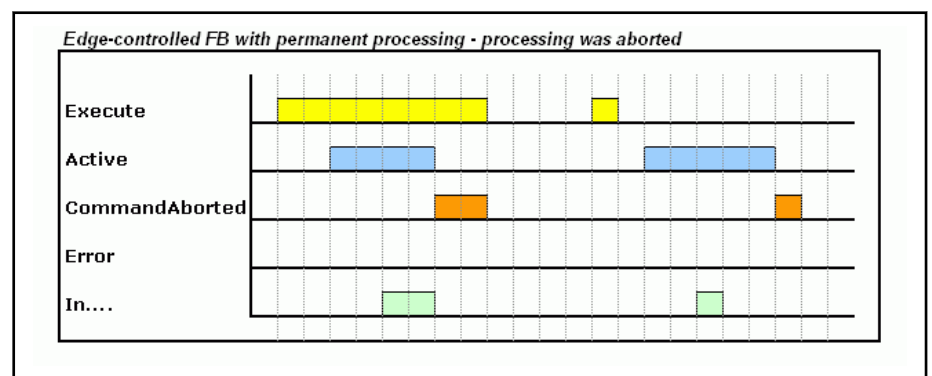


Fig. 5-23: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing interrupted

ML_Robot.library

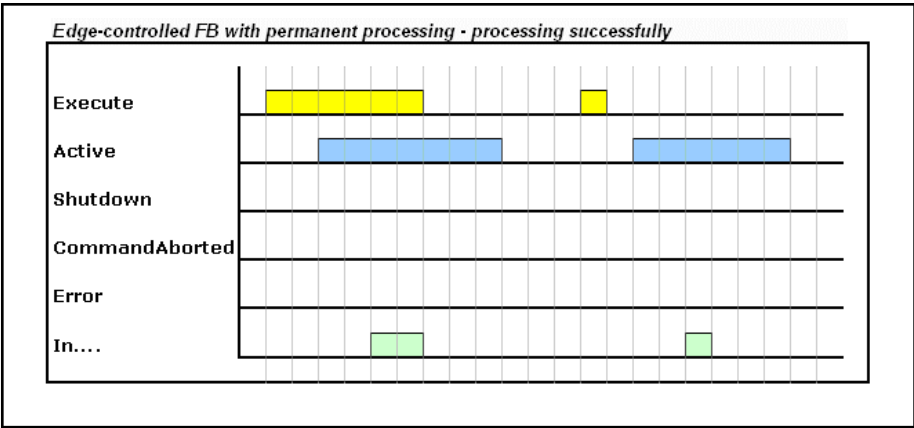


Fig.5-24: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing successful

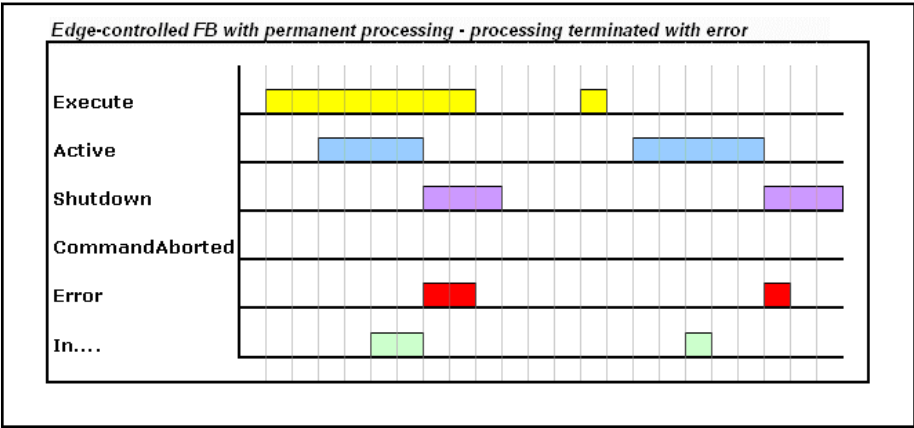


Fig.5-25: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing aborted with errors.

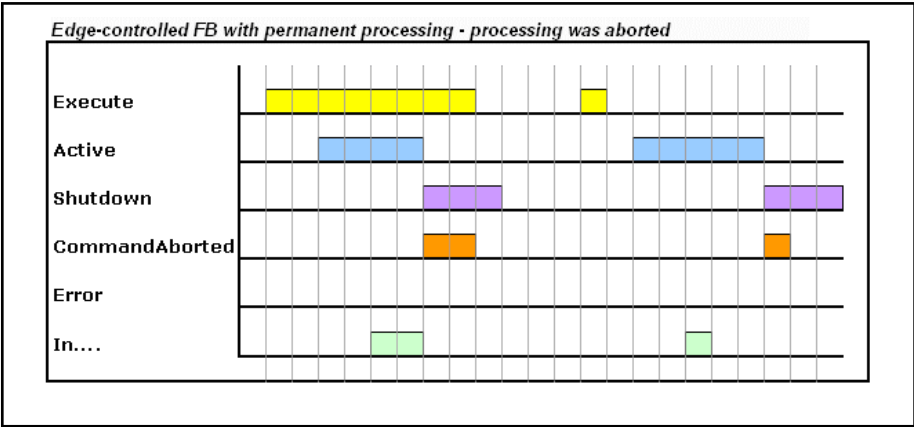


Fig.5-26: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing interrupted

5.3.3 Config Function Blocks

MC_GroupReset

Brief description The function block "MC_GroupReset" resets all reported errors of a kinematics.
The function block influences the state machine of the kinematics.
(Refer to [State Diagram of a Kinematics on page 26](#)).



If there is no error in the kinematics of this function block, MC_GroupReset is not enabled.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-10: Library assignment

Interface description

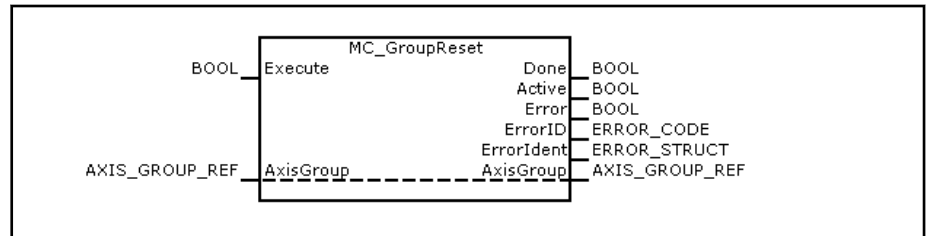


Fig.5-27: FB MC_GroupReset

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	The kinematics reset was successful
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-11: Interface of the MC_GroupReset function block

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

MC_SetCartesianTransform

Brief description

The function block "MC_SetCartesianTransform" defines a Cartesian product coordinate system.

(Refer to [Product Coordinate System \(PCS\)](#), [page 22](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-12: Library assignment

ML_Robot.library

Interface description

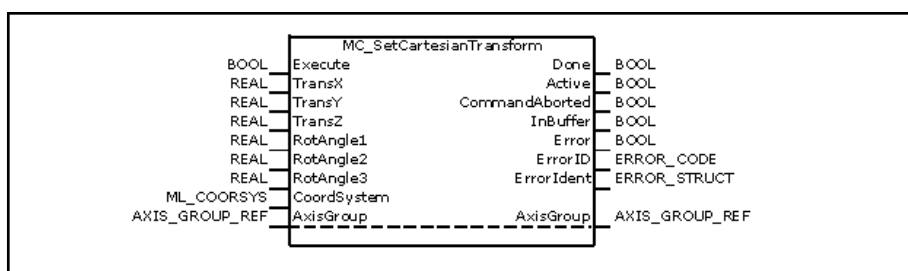


Fig.5-28: Function block MC_SetCartesianTransform

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	TransX	REAL	X position of the PCS in MCS
	TransY	REAL	Y position of the PCS in MCS
	TransZ	REAL	Z position of the PCS in MCS
	RotAngle1	REAL	Rotation of the PCS around the x-axis in MCS
	RotAngle2	REAL	Rotation of the PCS around the y-axis in MCS
	RotAngle3	REAL	Rotation of the PCS around the z-axis in MCS
	CoordSystem	ML_COORDSYS	Product coordinate system (PCS) that the values are to be set for (Refer to ML_COORDSYS , page 244).
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-13: Function block MC_SetCartesianTransform

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
TransX	REAL			0.0	Rising edge at "Execute"
TransY	REAL			0.0	Rising edge at "Execute"
TransZ	REAL			0.0	Rising edge at "Execute"
RotAngle1	REAL			0.0	Rising edge at "Execute"
RotAngle2	REAL			0.0	Rising edge at "Execute"

Name	Type	Min. value	Max. value	Default value	Effective
RotAngle3	REAL			0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS	CS_PRODU CT_COOR- DINATES_1	CS_PRODU CT_COOR- DI- NATES_16	CS_PRODUCT_CO- ORDINATES_1	Rising edge at "Execute"

Tab.5-14: Input behavior of MC_SetCartesianTransform

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-15: Error numbers of the function block MC_SetCartesianTransform

MC_SetKinTransform

Brief description The function block "MC_SetKinTransform" registers a user-defined transformation for a kinematics.
(Refer to [User-defined Transformation, page 98](#)).



This function block has to be executed in the parameterization mode (P2) before switching to the operating mode (P4/BB). Otherwise, it cannot be enabled.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-16: Library assignment

Interface description

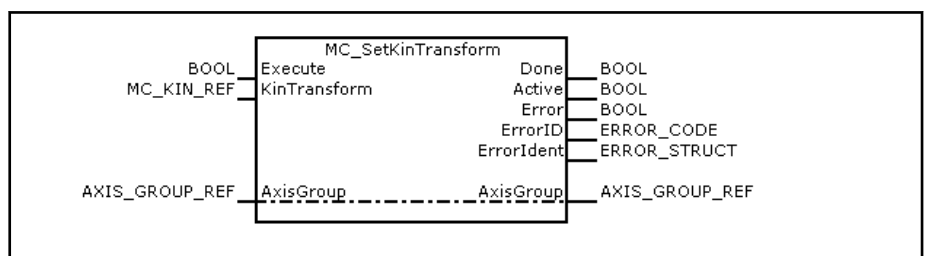


Fig.5-29: MC_SetKinTransform function block

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	KinTransform	MC_KIN_REF	Instance of a function block that implements the transformation interface This transformation has to be registered with the system (Refer to MC_KIN_REF , page 232).
VAR_OUTPUT	Done	BOOL	Registering of the transformation completed
	Active	BOOL	Registering of the transformation active
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-17: MC_SetKinTransform function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
KinTransform	MC_KIN_REF on page 232				Rising edge at "Execute"

Tab.5-18: Input behavior of the MC_SetKinTransform

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#), page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-19: Error number of the function block MC_SetKinTransform

MC_SetKinTransformExt

Brief description The function block "MC_SetKinTransformExt" registers an extended user-defined transformation for a kinematics.
(Refer to [Extended User-defined Transformation](#), page 100).



This function can only be used for Robot Control V2.

This function block has to be executed in the parameterization mode (P2) before switching to the operating mode (P4/BB). Otherwise, it cannot be enabled.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-20: Library assignment

Interface description

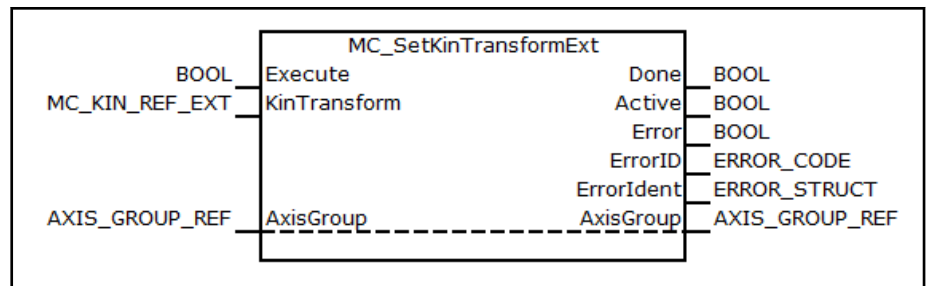


Fig. 5-30: MC_SetKinTransformExt function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	KinTransform	MC_KIN_REF_EXT	Instance of a function block that implements the transformation interface This transformation has to be registered with the system (Refer to MC_KIN_REF_EXT , page 234).
VAR_OUTPUT	Done	BOOL	Registering of the transformation completed
	Active	BOOL	Registering of the transformation active
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-21: MC_SetKinTransformExt function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
KinTransform	MC_KIN_REF_EXT on page 234				Rising edge at "Execute"

Tab. 5-22: Input behavior of MC_SetKinTransformExt

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab. 5-23: Error number of the function block MC_SetKinTransform

MC_UngroupAllAxes

Brief description The Coordinated Motion function block "MC_UngroupAllAxes" removes all axes of a kinematics.

ML_Robot.library

Therefore, this function block directly influences the state machine of the kinematics.
(Refer to [State Diagram of a Kinematics on page 26](#)).

Library	Range
ML_Robot	Kinematics/Config

Tab.5-24: Library assignment

Interface description

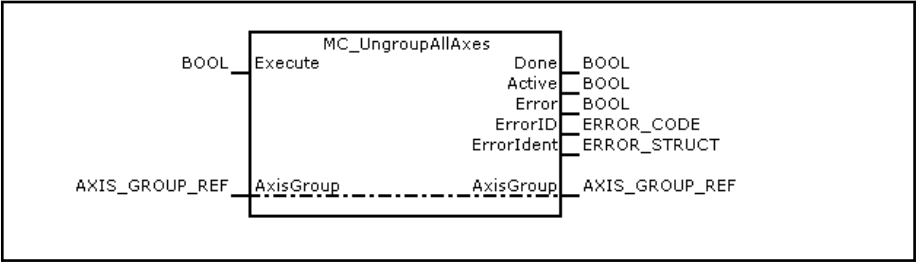


Fig.5-31: MC_UngroupAllAxes function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
VAR_OUTPUT	Done	BOOL	Processing completed
	Active	BOOL	Processing active
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Error ID
	ErrorIdent	ERROR_STRUCT	For the error structure with further error subdivision, see F_RELATED_TABLE error codes, on page 134 .
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-25: MC_UngroupAllAxes function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab.5-26: Input behavior of ML_UngroupAllAxes

F_RELATED_TABLE error codes The function block generates the following error messages in Additional1/Additional2 for F_RELATED_TABLE 16#0170:

ErrorID	Additional1	Additional2	Description
ACCESS_ERROR 16#04	16#00000002	16#00	GroupRef is not within the valid range
ACCESS_ERROR 16#04	16#00000011	16#00	sercos phase 2 is not active

Tab.5-27: Generated error numbers of the ML_UngroupAllAxes function block for F_RELATED_TABLE

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR 16#06	16#F0290402	16#00	Invalid kinematic number

Tab.5-28: Generated error numbers of the MC_UngroupAllAxes function block for MLC_TABLE

ML_AddMechanicData

Brief description The function "ML_AddMechanicData" adds an entry to the ML_KIN_MECHANIC_DATA structure.
(Refer to [Mechanical Information](#), page 103).

Library	Range
ML_Robot	Kinematics/Config

Tab.5-29: Library assignment

Interface description

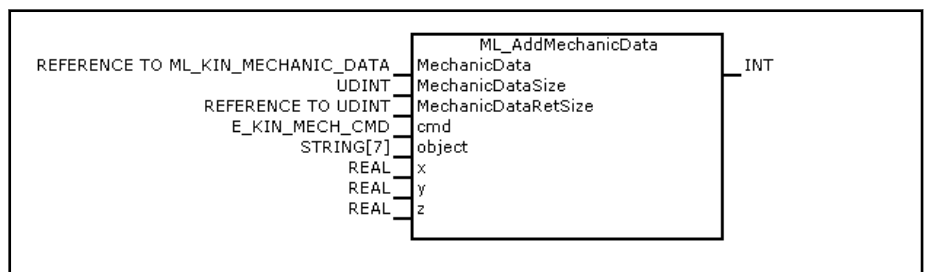


Fig.5-32: ML_AddMechanicData function

	Name	Type	Comment
VAR_INPUT	MechanicData	REFERENCE TO ML_KIN_MECHANIC_DATA	Structure to be filled in ML_KIN_MECHANIC_DATA on page 240
	MechanicDataSize	UDINT	Number of all elements in MechanicData
	MechanicDataRetSize	REFERENCE TO UDINT	Number of all used elements in MechanicData
	cmd	E_KIN_MECH_CMD	Command for the object name E_KIN_MECH_CMD on page 237
	object	STRING[7]	Object name
	x	REAL	X value
	y	REAL	Y value
	z	REAL	Z value
RETURN	ML_AddMechanicData	INT	Return value of the function

Tab.5-30: ML_AddMechanicData function

ML_GroupAllAxes

Brief description The Coordinated Motion function block "ML_GroupAllAxes" adds all axes of a kinematics to the kinematics.
Therefore, this function block directly influences the state machine of the kinematics.
(Refer to [State Diagram of a Kinematics](#) on page 26).

ML_Robot.library

To determine the axes of a kinematics, the function block reads parameter K-0-0008 of the selected kinematics and adds all positioning and orientation axes to the kinematics.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-31: Library assignment

Interface description

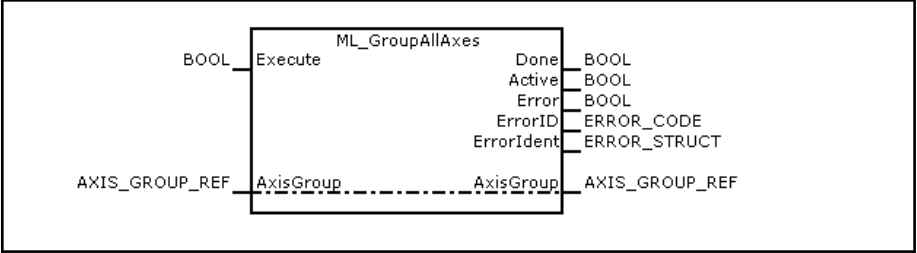


Fig.5-33: ML_GroupAllAxes function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
VAR_OUTPUT	Done	BOOL	Processing completed
	Active	BOOL	Processing active
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Error ID
	ErrorIdent	ERROR_STRUCT	For the error structure with further error subdivision, see F_Related_Table error codes, on page 136 .
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-32: ML_GroupAllAxes function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab.5-33: Input behavior of the ML_GroupAllAxes function block

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

F_Related_Table error codes

The function block generates the following error messages in Additional1/Additional2 for F_RELATED_TABLE 16#0170:

ErrorID	Additional1	Additional2	Description
ACCESS_ERROR 16#04	16#00000002	16#00	GroupRef is not within the valid range
ACCESS_ERROR 16#04	16#00000011	16#00	sercos phase 2 is not active

Tab.5-34: Generated error numbers of the ML_GroupAllAxes function block for F_RELATED_TABLE

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR 16#06	16#F0290402	16#00	Invalid kinematic number

Tab.5-35: Generated error numbers of the ML_GroupAllAxes function block for MLC_TABLE

ML_GroupSetSafeZone

Brief description The function block "ML_GroupSetSafeZone" configures a safe zone of a kinematics.
(Refer to [Traversing Range Limits on page 57](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-36: Library assignment

Interface description

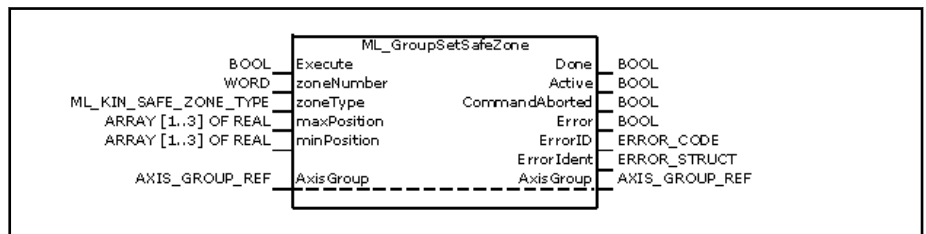


Fig.5-34: Function block ML_GroupSetSafeZone

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	zoneNumber	WORD	Number of the safe zone
	zoneType	ML_KIN_SAFE_ZONE_TYPE	Type of the safe zone (See ML_KIN_SAFE_ZONE_TYPE on page 231)
	maxPosition	ARRAY [1..3] OF REAL	Maximum point in the machine coordinate system of the safe zone
	minPosition	ARRAY [1..3] OF REAL	Minimum point in the machine coordinate system of the safe zone
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-37: Function block ML_GroupSetSafeZone

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
zoneNumber	WORD	0	31	0	Rising edge at "Execute"
zoneType	ML_KIN_SAFE_ZONE_TYPE			ML_KIN_SAFE_ZONE_DEACTIVATE	Rising edge at "Execute"
maxPosition	ARRAY [1..3] OF REAL			[3(0)]	Rising edge at "Execute"
minPosition	ARRAY [1..3] OF REAL			[3(0)]	Rising edge at "Execute"

Tab.5-38: Input behavior of ML_GroupSetSafeZone

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_KinSetDistanceBlendOption

Brief description The function "ML_KinSetDistanceBlendOption" enables the blending mode. The new mode has an effect on all subsequent commands in the block buffer.



This function can only be used for Robot Control V1.
The function overwrites the value set by SPC_FCT: 60.

Library	Range
ML_Robot_yyy	Kinematics/Config

Tab.5-39: Library assignment

Interface description

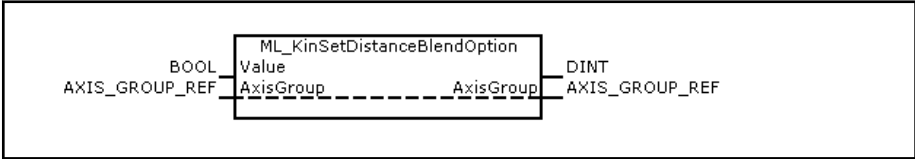


Fig.5-35: ML_KinSetDistanceBlendOption function

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
VAR_INPUT	Value	BOOL	FALSE: The blending radius is limited to half the distance of the total distance (default). TRUE: The blending radius can reach the entire distance of the shorter distance if possible.
Function value	ML_KinSetDistanceBlendOption	DINT	It indicates whether the function was executed successfully: 0 - Valid 1 - Other errors -2 - Error: Value in AxisGroup is not a valid kinematics

Tab.5-40: FUN ML_KinSetDistanceBlendOption interface

Functional description

This function corresponds - in the RCL program - to the special function SPC_FCT: 60, Set blending mode.

The programmed blending radius R is considered in both modes.

If **Value** = FALSE,

the effective blending radius R_w results from the minimum of the programmed radius R, half the distance s_1 and half the distance s_2 .

$$R_w = \min \left(R, \frac{1}{2}s_1, \frac{1}{2}s_2 \right)$$

Fig.5-36: Effective radius when blending with blending mode 0

If **Value** = TRUE,

the effective blending radius R_w results from the minimum of the programmed radius R, the total distance s_1 and the total distance s_2 , if this is possible.

$$R_w = (R, n_1 \cdot s_1, n_2 \cdot s_2)$$

Fig.5-37: Effective radius when blending with blending mode 1

$n_{(1,2)} = 1$ or $1/2$

s_1 = distance 1

s_2 = distance 2

$n_{(1,2)}$ is automatically selected by the control by means of the programmed traveling commands.

If the travel command is a "MOVE TO" command, $n = 1$ for this distance.

If the travel command is a "MOVE VIA" command, n depends on the specified command.

ML_SetBeltDesyncRelConfiguration

Brief description

The ML_SetBeltDesyncRelConfiguration function block configures relative desynchronization of a belt.

(Refer to [Belt-synchronous Motion on page 49](#)).



This function block can only be used for Robot Control V2.

ML_Robot.library

Library	Range
ML_Robot	Kinematics/Config

Tab.5-41: Library assignment

Interface description

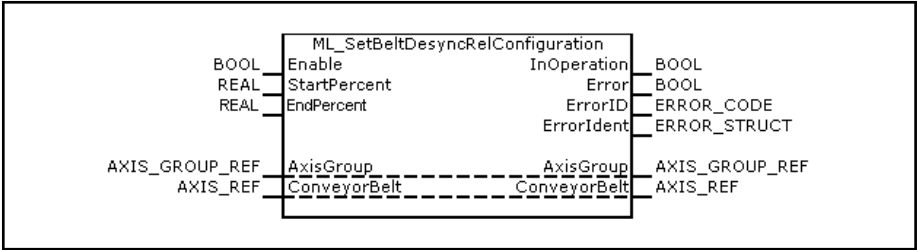


Fig.5-38: Function block ML_SetBeltDesyncRelConfiguration

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	StartPercent	REAL	Starting percentage for relative desynchronization
	EndPercent	REAL	Ending percentage for relative desynchronization
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-42: Function block ML_SetBeltDesyncRelConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
StartPercent	REAL	0.0	1.0	0.0	Continuous
EndPercent	REAL	0.0	1.0	0.5	Continuous

Tab.5-43: Input behavior of ML_SetBeltDesyncRelConfiguration

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ML_SetBeltConfiguration

Brief description

The function block "ML_SetBeltConfiguration" configures a belt.
This function block has to be executed once before the belt-synchronous traveling.
The belt parameters may only be set if the kinematics is neither synchronizing nor traveling belt-synchronously.
(Refer to [Belt-synchronous Motion on page 49](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-44: Library assignment

Interface description

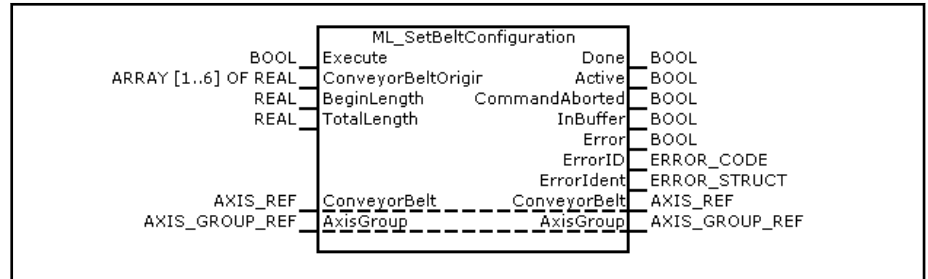


Fig.5-39: Function block ML_SetBeltConfiguration

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	ConveyorBeltOrigin	ARRAY [1..6] OF REAL	Defines the belt position and rotation in relation to the MCS
	BeginLength	REAL	Defines the length of the late start of a belt-synchronous motion from the starting position
	TotalLength	REAL	Defines the length of the belt-synchronous travel ranges from the starting position
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-45: Function block ML_SetBeltConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
ConveyorBeltOrigin	ARRAY [1..6] OF REAL	[6(0.0)]		[6(0.0)]	Rising edge at "Execute"
BeginLength	REAL	> 0.0	<TotalLength>	0.0	Rising edge at "Execute"
TotalLength	REAL	> 0.0		0.0	Rising edge at "Execute"

Tab.5-46: Input behavior of ML_SetBeltConfiguration

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120.](#)

ML_Robot.library

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#0003	16#F0220201	16#0000	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-47: Error numbers of the function block ML_SetBeltConfiguration

ML_SetBeltErrorReaction

Brief description The function block "ML_SetBeltErrorReaction" configures the error reaction of a belt.
 (Refer to [Belt-synchronous Motion, page 49](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-48: Library assignment

Interface description

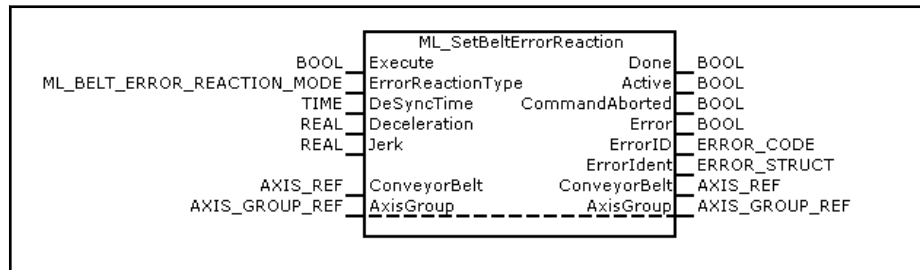


Fig.5-40: Function block ML_SetBeltErrorReaction

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	ErrorReactionType	ML_BELT_ERROR_REACTION_MODE	Defines the error reaction type (Refer to ML_BELT_ERROR_REACTION_MODE, page 230).
	DeSyncTime	TIME	Time for desynchronization
	Deceleration	REAL	Maximum path deceleration for desynchronization
	Jerk	REAL	Maximum path jerk for desynchronization
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-49: Function block ML_SetBeltErrorReaction

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
ErrorReactionType	ML_BELT_ERROR_REACTION_MODE			ML_BELT_ERROR_NO_SPECIFIC_REACTION	Rising edge at "Execute"
DeSyncTime	TIME	>T#0ms		T#0ms	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"

Tab.5-50: Input behavior of ML_SetBeltErrorReaction

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#0003	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-51: Error numbers of the function block ML_SetBeltErrorReaction

ML_SetBeltParameter

Brief description The function block "ML_SetBeltParameter" sets the parameters of a belt. This function block has to be executed once before the belt-synchronous traveling.
The belt parameters may only be set if the kinematics is neither synchronizing nor traveling belt-synchronously.
(Refer to [Belt-synchronous Motion on page 49](#)).



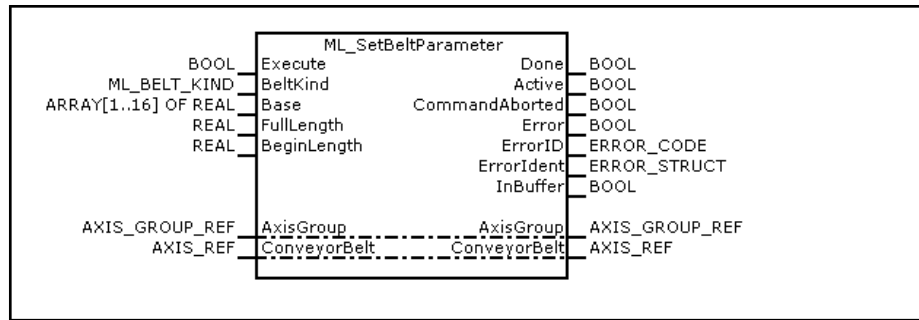
This function block can only be used for Robot Control V1.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-52: Library assignment

ML_Robot.library

Interface description

Fig.5-41: *ML_SetBeltParameter function block*

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	BeltKind	ML_BELT_KIND	Defines the belt synchronization algorithm (Refer to ML_BELT_KIND on page 230)
	Base	ARRAY [1..16] OF REAL	Defines the starting position from where belt-synchronous traveling can begin
	FullLength	REAL	Defines the length of the belt-synchronous travel ranges from the starting position
	BeginLength	REAL	Defines the length of the late start of a belt-synchronous motion from the starting position
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-53: *ML_SetBeltParameter function block*

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
BeltKind	ML_BELT_KIND			BK_4	Rising edge at "Execute"
Base	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
FullLength	REAL	>0.0		0.0	Rising edge at "Execute"
BeginLength	REAL	>0.0	<FullLength>	0.0	Rising edge at "Execute"

Tab.5-54: *Input behavior of the ML_SetBeltParameter*

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-55: Error numbers of the function block ML_SetBeltParameter

ML_SetCylindricTransform

Brief description The function block "MC_SetCylindricTransform" defines a cylindrical product coordinate system.
(Refer to [Product Coordinate System \(PCS\), page 22](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-56: Library assignment

Interface description

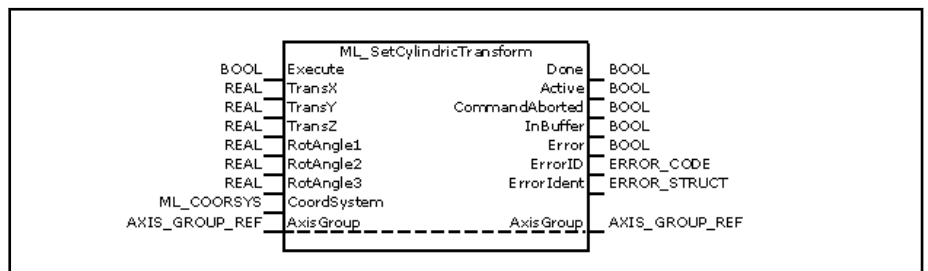


Fig.5-42: Function block ML_SetCylindricTransform

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	TransX	REAL	X position of the PCS in MCS
	TransY	REAL	Y position of the PCS in MCS
	TransZ	REAL	Z position of the PCS in MCS

ML_Robot.library

	Name	Type	Comment
	RotAngle1	REAL	Rotation of the PCS around the x-axis in MCS
	RotAngle2	REAL	Rotation of the PCS around the y-axis in MCS
	RotAngle3	REAL	Rotation of the PCS around the z-axis in MCS
	CoordSystem	ML_COORDSYS	Product coordinate system that the values are to be defined for See ML_COORDSYS , page 244
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-57: Function block ML_SetCylindricTransform

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
TransX	REAL			0.0	Rising edge at "Execute"
TransY	REAL			0.0	Rising edge at "Execute"
TransZ	REAL			0.0	Rising edge at "Execute"
RotAngle1	REAL			0.0	Rising edge at "Execute"
RotAngle2	REAL			0.0	Rising edge at "Execute"
RotAngle3	REAL			0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS	CS_PRODU CT_COOR- DINATES_1	CS_PRODU CT_COOR- DI- NATES_16	CS_PRODUCT_CO- ORDINATES_1	Rising edge at "Execute"

Tab.5-58: Input behavior of ML_SetCylindricTransform

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#0003	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-59: Error numbers of the function block ML_SetCylindricTransform

ML_SetRotaryTableConfiguration

Brief description The function block "ML_SetRotaryTableConfiguration" configures a rotary table.

(Refer to [Belt-synchronous Motion on page 49](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Config

Tab.5-60: Library assignment

Interface description

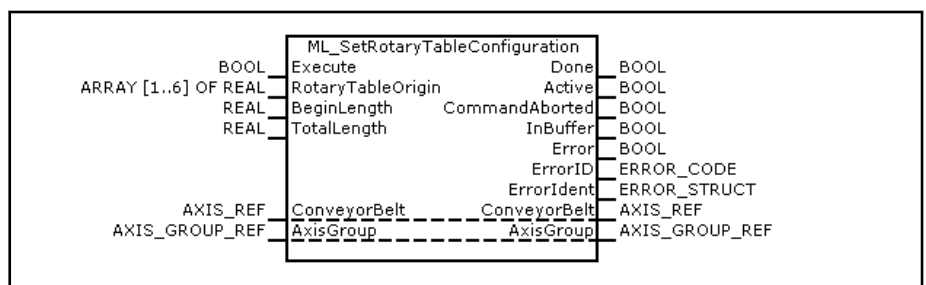


Fig.5-43: Function block ML_SetRotaryTableConfiguration

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	RotaryTableOrigin	ARRAY [1..6] OF REAL	Defines the position and rotation of the rotary table in relation to the MCS
	BeginLength	REAL	Defines the length of the late start of a belt-synchronous motion from the starting position
	TotalLength	REAL	Defines the length of the belt-synchronous travel ranges from the starting position
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-61: Function block ML_SetRotaryTableConfiguration

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
ConveyorBeltOrigin	ARRAY [1..6] OF REAL			[6(0)]	Rising edge at "Execute"
BeginLength	REAL	> 0.0		0.0	Rising edge at "Execute"
TotalLength	REAL	> 0.0		0.0	Rising edge at "Execute"

Tab.5-62: Input behavior of ML_SetRotaryTableConfiguration

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#0003	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is in- correct

Tab.5-63: Error numbers of the function block ML_SetRotaryTableConfiguration

5.3.4 Info Function Blocks and Functions

MC_GroupReadActualPosition

Brief description The function block "MC_GroupReadActualPosition" reads the current position of the kinematics.

 A modification of the kinematics at the input of the function block requires another 0/1 transition at "Enable".

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-64: Library assignment

Interface description

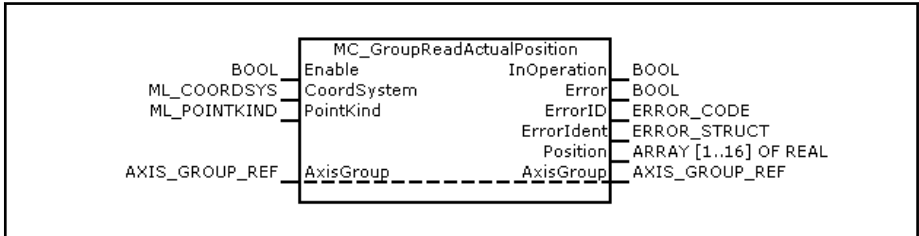


Fig.5-44: Function block MC_GroupReadActualPosition

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the position value as long as "Enable" is TRUE.
	CoordSystem	ML_COORDSYS	Coordinate system of the point see ML_COORDSYS on page 244
	PointKind	ML_POINTKIND	Point type regarding chronological sequence: Block preparation/block processing see ML_POINTKIND on page 240
VAR_OUTPUT	InOperation	BOOL	The valid value is available in its task
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Position	ARRAY [1..16] OF REAL	Current spatial position of the kinematics
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-65: Interface of MC_GroupReadActualPosition

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
CoordSystem	ML_COORDSYS	CS_PRODUCT_COORDINATES_1	CS_PRODUCT_COORDINATES_16	CS_PRODUCT_COORDINATES_1	Continuous if "Enable" = TRUE
PointKind	ML_POINTKIND	POINTKIND_PREPARE	POINTKIND_EXECUTE	POINTKIND_PREPARE	Continuous if "Enable" = TRUE

Tab.5-66: Input behavior of GroupReadActualPosition

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_KinGetDistanceBlendOption

Brief description The function "ML_KinGetDistanceBlendOption" returns the blending mode set.



This function can only be used for Robot Control V1.

ML_Robot.library

Library	Range
ML_Robot	Kinematics/Info

Tab.5-67: Library assignment

Interface description

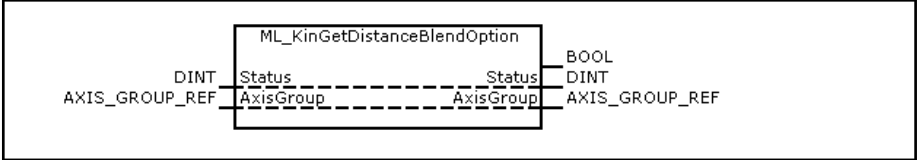


Fig.5-45: ML_KinGetDistanceBlendOption function

	Name	Type	Comment
VAR_IN_OUT	Status	DINT	It indicates whether the function was executed successfully: 0 - Valid 1 - Other errors -2 - Error: Value in AxisGroup is not a valid kinematics
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value	ML_KinGetDis- tanceBlendOp- tion	BOOL	FALSE: The blending radius is limited to half the distance of the total distance (default). TRUE: The blending radius can reach the entire distance of the shorter distance if possible.

Tab.5-68: FUN ML_KinGetDistanceBlendOption interface

Functional description Using the function, the blending mode of the kinematics set in the block preparation can be queried in the PLC program.

 This function corresponds - in the RCL program - to the special function SPC_FCT: 59, Query blending mode.

ML_RobotGetBeltConfiguration

Brief description The function block "ML_RobotGetBeltConfiguration" returns the current configuration of a belt.

 (Refer to [Belt-synchronous Motion on page 49](#)).

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-69: Library assignment

Interface description

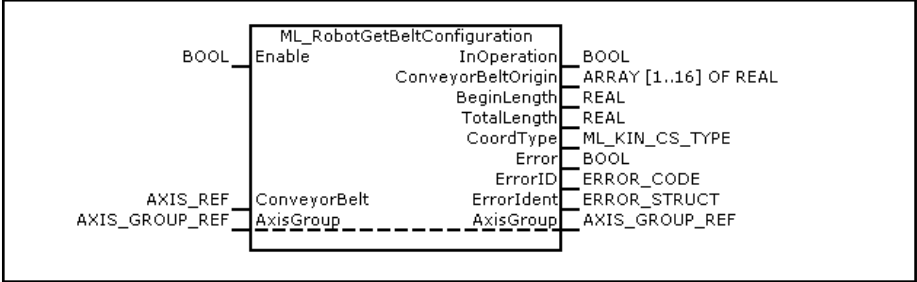


Fig.5-46: Function block ML_RobotGetBeltConfiguration

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	ConveyorBeltOrigin	ARRAY [1..16] OF REAL	Defines the belt position and rotation in relation to the MCS
	BeginLength	REAL	Defines the length of the late start of a belt-synchronous motion from the starting position
	TotalLength	REAL	Defines the length of the belt-synchronous travel ranges from the starting position
	CoordType	ML_KIN_CS_TYPE	Type of coordinate system (See ML_KIN_CS_TYPE on page 239)
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt

Tab.5-70: Function block *ML_RobotGetBeltConfiguration*

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab.5-71: Input behavior of *ML_RobotGetBeltConfiguration*

Error handling

The function block generates the following types of error messages:

[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-72: Error numbers of the function block *ML_RobotGetBeltConfiguration*

ML_RobotGetBeltDesyncRelConfiguration

Brief description

The function block "ML_RobotGetBeltDesyncRelConfiguration" reads the relative desynchronization configuration of a belt.

(Refer to [Belt-synchronous Motion](#), page 49).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-73: Library assignment

ML_Robot.library

Interface description

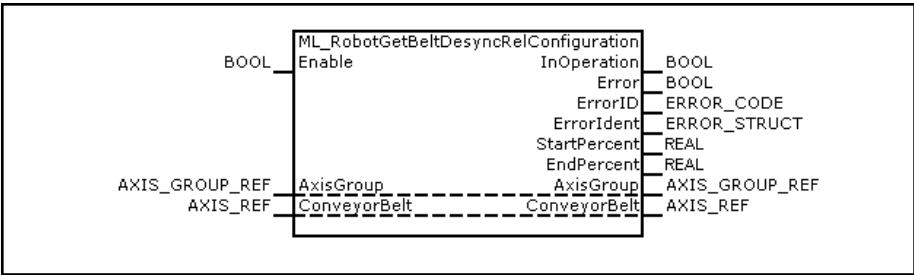


Fig.5-47: Function block ML_RobotGetBeltDesyncRelConfiguration

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	StartPercent	REAL	Starting percentage for relative desynchronization
	EndPercent	REAL	Ending percentage for relative desynchronization
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-74: Function block ML_RobotGetBeltDesyncRelConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab.5-75: Input behavior of ML_RobotGetBeltDesyncRelConfiguration

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_RobotGetBeltErrorReaction

Brief description The function block "ML_RobotGetBeltErrorReaction" returns the configured belt error reaction of the robot.
(Refer to [Mechanical Information, page 103](#)).

This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-76: Library assignment

Interface description

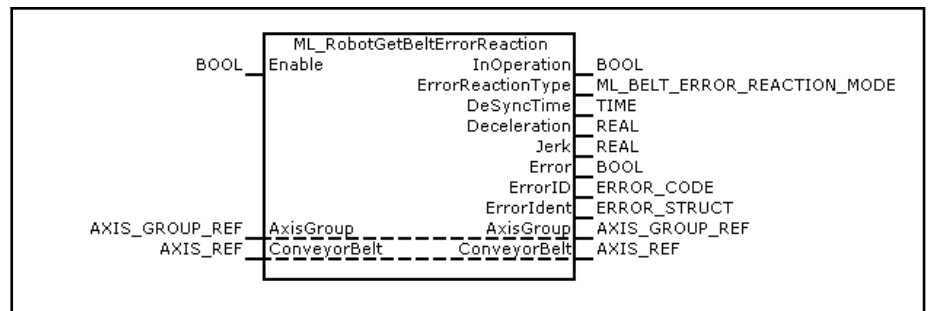


Fig. 5-48: Function block ML_RobotGetBeltErrorReaction

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	ErrorReactionType	ML_BELT_ERROR_REACTION_MODE	Set error reaction
	DeSyncTime	TIME	Set time for desynchronization
	Deceleration	REAL	Set maximum path deceleration
	Jerk	REAL	Set maximum path jerk
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Reference to the belt

Tab. 5-77: Function block ML_RobotGetBeltErrorReaction

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-78: Input behavior of ML_RobotGetBeltErrorReaction

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab. 5-79: Error numbers of the function block ML_RobotGetMechanic

ML_RobotGetMechanic

Brief description The function block "ML_RobotGetMechanic" returns mechanical kinematic data.
(Refer to [Mechanical Information on page 103](#)).

ML_Robot.library



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-80: Library assignment

Interface description

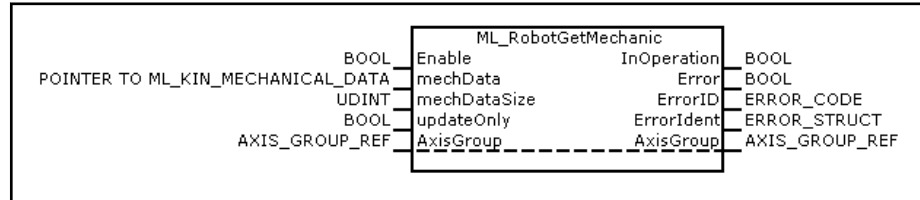


Fig.5-49: Function block ML_RobotGetMechanic

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	mechData	POINTER TO ML_KIN_MECHANICAL_DATA	Pointer to the array of mechanical data to be filled
	mechDataSize	UDINT	Number of available elements in the array of mechanical data
	updateOnly	BOOL	Request an update or the complete mechanical data
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	mechDataCount	UDINT	Number of data read in the array of mechanical data
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-81: Function block ML_RobotGetMechanic

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
mechData	POINTER TO ML_KIN_MECHANICAL_DATA			NULL	Continuous if "Enable" = TRUE
mechDataSize	UDINT			0	Continuous if "Enable" = TRUE
updateOnly	BOOL			FALSE	Continuous if "Enable" = TRUE

Tab.5-82: Input behavior of ML_RobotGetMechanic

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling

The function block generates the following types of error messages:

[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-83: Error numbers of the function block ML_RobotGetMechanic

ML_RobotGetPerformance

Brief description The function block "ML_RobotGetPerformance" returns performance data of the kinematics.

(Refer to [Performance Measurement on page 63](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-84: Library assignment

Interface description

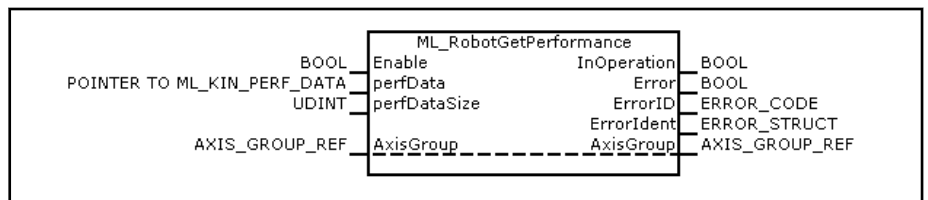


Fig. 5-50: Function block ML_RobotGetPerformance

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	perfData	POINTER TO ML_KIN_PERF_DATA	Pointer to the array of performance data to be filled
	perfDataSize	UDINT	Number of available elements in the array of performance data
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	mechDataCount	UDINT	Number of data read in the array of performance data
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-85: Function block ML_RobotGetPerformance

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
perfData	POINTER TO ML_KIN_PERF_DATA			NULL	Continuous if "Enable" = TRUE
perfDataSize	UDINT			0	Continuous if "Enable" = TRUE

Tab.5-86: Input behavior of ML_RobotGetPerformance

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-87: Error numbers of the function block ML_RobotGetPerformance

ML_RobotGetTrafoValues

Brief description

The function block "ML_RobotGetTrafoValues" returns the current configuration of a product coordinate system or of the basic coordinate system.
(Refer to [Coordinate Systems on page 19](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-88: Library assignment

Interface description

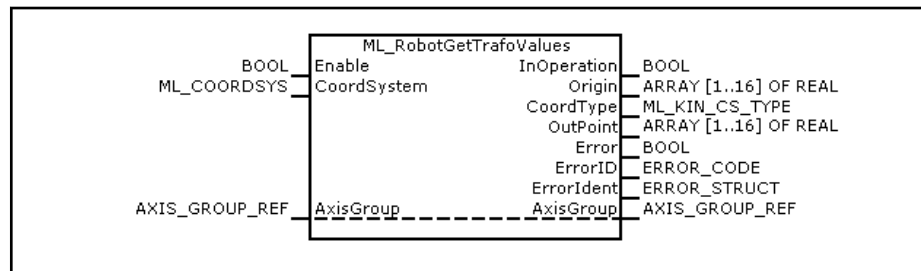


Fig.5-51: Function block ML_RobotGetTrafoValues

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	CoordSystem	ML_COORDSYS	Coordinate system of the point
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Origin	ARRAY [1..16] OF REAL	Position and orientation of the coordinate system

	Name	Type	Comment
	CoordType	ML_KIN_CS_TYPE	Type of coordinate system
	OutPoint	ARRAY [1..16] OF REAL	Current spatial position of the point
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-89: Function block ML_RobotGetTrafoValues

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
CoordSystem	ML_COORDSYS			CS_MACHINE_COOR-DINATES	Continuous if "Enable" = TRUE

Tab.5-90: Input behavior of ML_RobotGetTrafoValues

Error handling

The function block generates the following types of error messages:

[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-91: Error numbers of the function block ML_RobotGetTrafoValues

ML_RobotMovePoint

Brief description

The function block "ML_RobotMovePoint" returns the current position of a point in the space.

(Refer to [Belt-synchronous Motion on page 49](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-92: Library assignment

Interface description

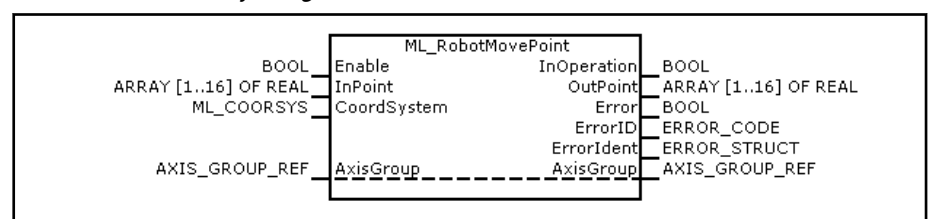


Fig.5-52: Function block ML_RobotMovePoint

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	InPoint	ARRAY [1..16] OF REAL	Point in the space whose current position is to be requested
	CoordSystem	ML_COORDSYS	Coordinate system of the point
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	OutPoint	ARRAY [1..16] OF REAL	Current spatial position of the point
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-93: Function block ML_RobotMovePoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
InPoint	ARRAY [1..16] OF REAL			[16(0)]	Continuous if "Enable" = TRUE
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Continuous if "Enable" = TRUE

Tab.5-94: Input behavior of ML_RobotMovePoint

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing](#) on page. 123

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
			48
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-95: Error numbers of the function block ML_RobotMovePoint

ML_RobotReadCyclicPos

Brief description The function block "ML_RobotReadCyclicPos" reads the current cyclic position.
 (Refer to [Cyclical Command Value Specification](#) on page 61).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-96: Library assignment

Interface description

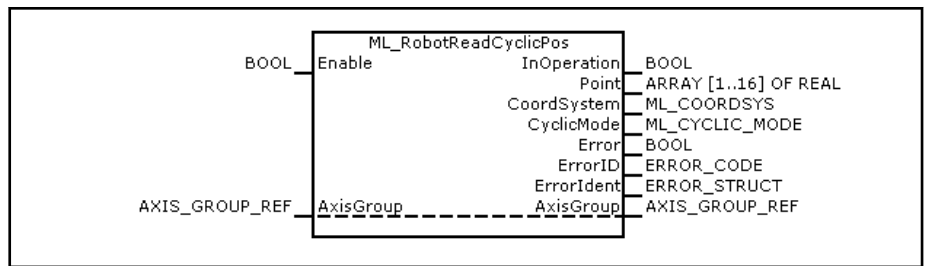


Fig.5-53: Function block ML_RobotReadCyclicPos

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Point	ARRAY [1..16] OF REAL	Last cyclically commanded position
	CoordSystem	ML_COORDSYS	Coordinate system of the last cyclically commanded position
	CyclicMode	ML_CYCLIC_MODE	Type of cyclical command value specification
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-97: Function block ML_RobotReadCyclicPos

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab.5-98: Input behavior of ML_RobotReadCyclicPos

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-99: Error numbers of the function block ML_RobotReadCyclicPos

ML_RobotSetPerformanceEnable

Brief description The function block "ML_RobotSetPerformanceEnable" changes the activation state of the performance measurement.
(Refer to [Performance Measurement on page 63](#)).



This function block can only be used for Robot Control V2.

ML_Robot.library

Library	Range
ML_Robot	Kinematics/Info

Tab.5-100: Library assignment

Interface description

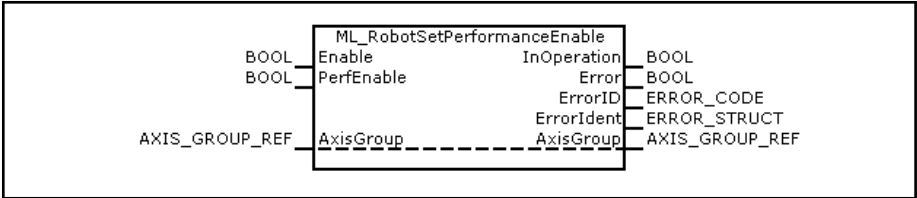


Fig.5-54: Function block ML_RobotSetPerformanceEnable

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	PerfEnable	BOOL	New activation state of the performance measurement
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-101: Function block ML_RobotSetPerformanceEnable

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PerfEnable	BOOL			FALSE	Continuous if "Enable" = TRUE

Tab.5-102: Input behavior of ML_RobotSetPerformanceEnable

- Signal-time diagramRefer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)
- Error handlingThe function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is in-correct

Tab.5-103: Error numbers of the function block ML_RobotSetPerformanceEnable

ML_RobotTransformPoint

- Brief descriptionThe function block "ML_RobotTransformPoint" transforms a point from a co-ordinate system to another coordinate system.
(Refer to [Coordinate Systems on page 19](#)).

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Info

Tab.5-104: Library assignment

Interface description

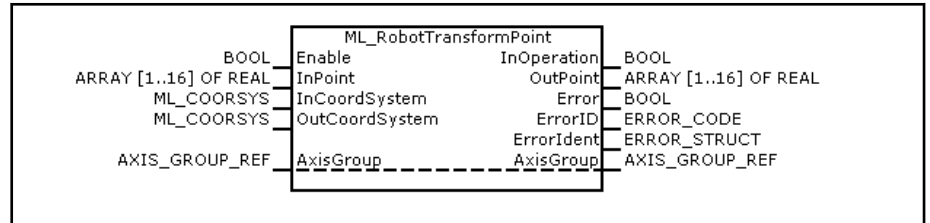


Fig.5-55: Function block ML_RobotTransformPoint

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	InPoint	ARRAY [1..16] OF REAL	Point in space that is to provide the current position
	InCoordSystem	ML_COORDSYS	Coordinate system of the point
	OutCoordSystem	ML_COORDSYS	Coordinate system the point is to be transformed to
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	OutPoint	ARRAY [1..16] OF REAL	Current spatial position of the point
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-105: Function block ML_RobotTransformPoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
InPoint	ARRAY [1..16] OF REAL			[16(0)]	Continuous if "Enable" = TRUE
InCoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Continuous if "Enable" = TRUE
OutCoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Continuous if "Enable" = TRUE

Tab.5-106: Input behavior of ML_RobotTransformPoint

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing](#) on page. 123

Error handling

The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_Robot.library

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-107: Error numbers of the function block ML_RobotTransformPoint

ML_SwitchOnPath

Brief description With the function block "ML_SwitchOnPath" it is possible to switch on one path.

During a traverse movement, PLC outputs can be controlled in a position-accurate way and with a temporal lead.

(Refer to [Belt-synchronous Motion on page 49](#)).

Library	Range
ML_Robot	Kinematics/Info

Tab.5-108: Library assignment

Interface description

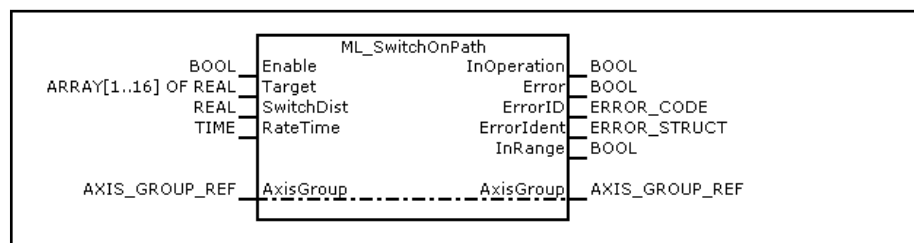


Fig.5-56: ML_SwitchOnPath function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	Target	ARRAY [1..16] OF REAL	Target position for switching (world coordinate position)
	SwitchDist	REAL	Distance/radius from the target position to the switching position
	RateTime	TIME	Lead time
VAR_OUTPUT	InOperation	BOOL	Processing is executed
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InRange	BOOL	Kinematics in switching range
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-109: ML_SwitchOnPath function block

Name	Type	Default value	Effective
Enable	BOOL	FALSE	Continuous
Target	ARRAY [1..16] OF REAL	16(0.0)	Continuous

Name	Type	Default value	Effective
SwitchDist	REAL	16(0.0)	Continuous
RateTime	TIME	0.0	Continuous

Tab.5-110: Input behavior of the ML_SwitchOnPath

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page 123](#)

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect

Tab.5-111: Error numbers of the function block ML_SwitchOnPath

5.3.5 Movement Function Blocks

MC_GroupContinue

Brief description The function block "MC_GroupContinue" restarts a kinematics motion that was interrupted by the function block [MC_GroupInterrupt on page 164](#).
(See [Continuing Motions on page 48](#))



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-112: Library assignment

Interface description

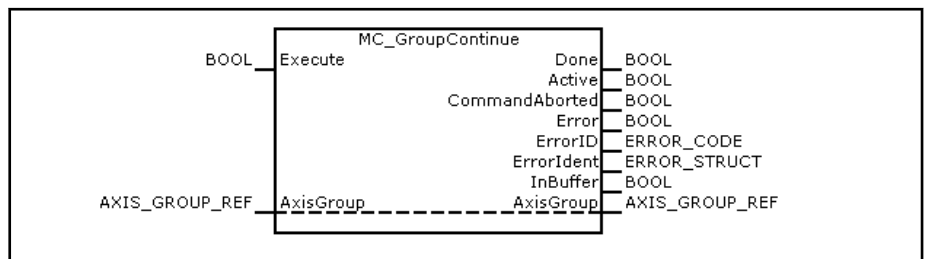


Fig.5-57: Function block MC_GroupContinue

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge
VAR_OUTPUT	Done	BOOL	All interrupted motions are restarted
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command was aborted by another function block
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause

ML_Robot.library

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-113: Function block MC_GroupContinue

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab.5-114: Input behavior of MC_GroupContinue

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

MC_GroupInterrupt

Brief description

The function block "MC_GroupInterrupt" interrupts the movement of a kinematics.

(Refer to [Interrupting Motions, page 48](#)).

The motion can be resumed with the function block [MC_GroupContinue on page 163](#).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-115: Library assignment

Interface description

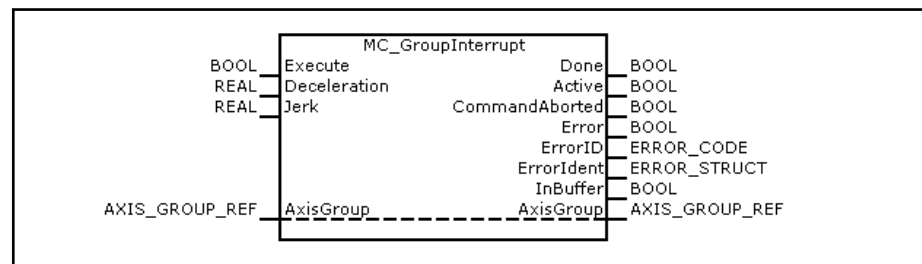


Fig.5-58: Function block MC_GroupInterrupt

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
VAR_OUTPUT	Done	BOOL	Velocity 0 is reached
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command was aborted by another function block
	Error	BOOL	Indicates that an error occurred.

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-116: Function block MC_GroupInterrupt

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"

Tab.5-117: Input behavior of MC_GroupInterrupt

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

MC_GroupStop

Brief description The function block "MC_GroupStop" stops a kinematics in a controlled way. (Refer to [Stopping Motions, page 46](#)).
Other commands can only be initiated when the "Done" output is set at the function block and when there is a falling edge at the "Execute" input.



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-118: Library assignment

Interface description

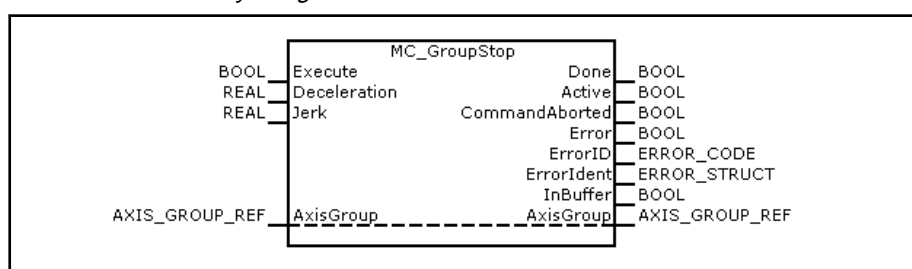


Fig.5-59: Function block MC_GroupStop

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk

ML_Robot.library

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	Velocity 0 is reached
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command was aborted by another function block
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-119: Function block MC_GroupStop

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	≥ 0.0		0.0	Rising edge at "Execute"

Tab.5-120: Input behavior of MC_GroupStop

Signal-time diagram

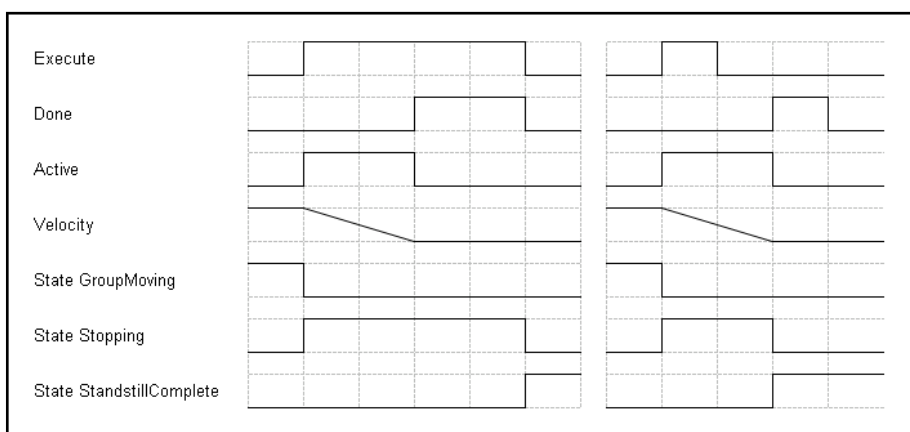


Fig.5-60: Time diagram of MC_GroupStop

Error handling

The function block generates the following types of error messages:

[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

MC_MoveCircularAbsolute

Brief description

The function block "MC_MoveCircularAbsolute" moves a kinematics circularly from the last commanded TCP position via an intermediate point to a specified absolute target position in the given coordinate system.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-121: Library assignment

Interface description

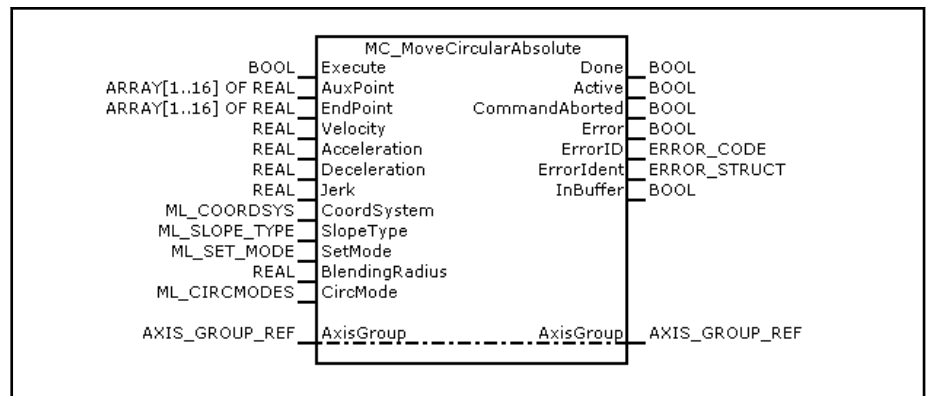


Fig.5-61: MC_MoveCircularAbsolute function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	AuxPoint	ARRAY [1..16] OF REAL	Intermediate position of the motion
	EndPoint	ARRAY [1..16] OF REAL	Target position of the motion
	Velocity	REAL	Maximum path velocity
	Acceleration	REAL	Maximum path acceleration
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius
	CircMode	ML_CIRCMODES	Type of circular motion
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-122: MC_MoveCircularAbsolute function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
AuxPoint	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
EndPoint	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"
CircMode	ML_CIRCMODES			CM_BORDER	Rising edge at "Execute"

Tab.5-123: Input behavior of the MC_MoveCircularAbsolute

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#0000	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#0000	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#0000	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the function block ML_GroupAllAxes on page 135

Tab.5-124: Error numbers of the function block MC_MoveCircularAbsolute

MC_MoveCircularRelative

Brief description

The function block "MC_MoveCircularRelative" moves a kinematics circularly from the last commanded TCP position via an intermediate point to a specified absolute target position in the given coordinate system.

The intermediate and target points are specified in relation to the starting point.



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-125: Library assignment

Interface description

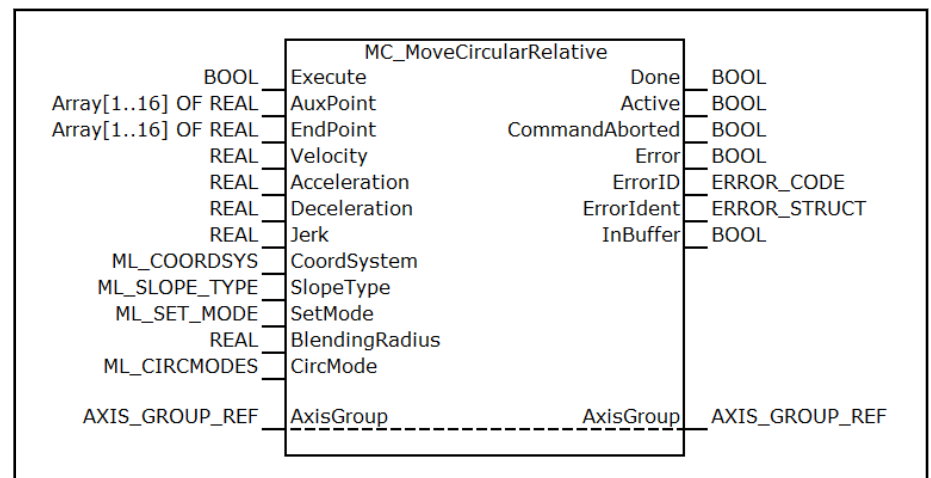


Fig.5-62: Function block MC_MoveCircularRelative

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	AuxPoint	ARRAY [1..16] OF REAL	Intermediate position of the motion. Specification relative to the starting point
	EndPoint	ARRAY [1..16] OF REAL	Target position of the motion. Specification relative to the starting point
	Velocity	REAL	Maximum path velocity
	Acceleration	REAL	Maximum path acceleration
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius
	CircMode	ML_CIRCmodes	Type of circular motion
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed

ML_Robot.library

	Name	Type	Comment
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-126: Function block MC_MoveCircularRelative

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
AuxPoint	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
EndPoint	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"
CircMode	ML_CIRCMODES			CM_BORDER	Rising edge at "Execute"

Tab.5-127: Input behavior of MC_MoveCircularRelative

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#0000	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#0000	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#0000	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the function block ML_GroupAllAxes on page 135

Tab.5-128: Error numbers of the function block MC_MoveCircularRelative

MC_MoveDirectAbsolute

Brief description The function block "MC_MoveDirectAbsolute" moves a kinematics point by point from the last commanded TCP position to a specified absolute target position in the given coordinate system.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-129: Library assignment

Interface description

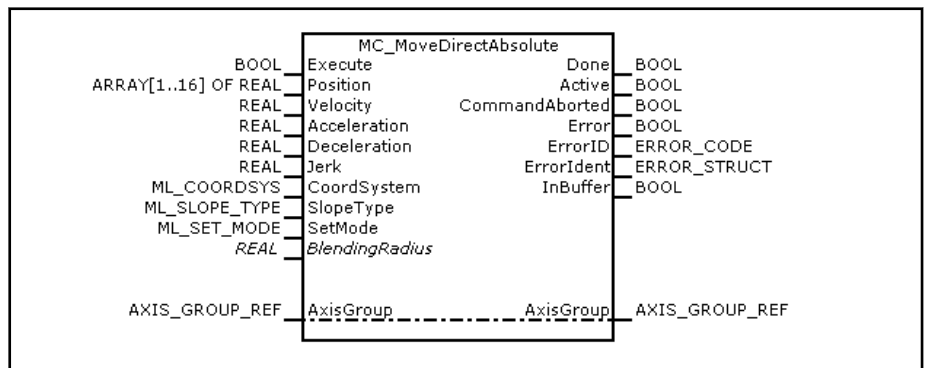


Fig.5-63: MC_MoveDirectAbsolute function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Position	ARRAY [1..16] OF REAL	Target position of the motion
	Velocity	REAL	Percentage of the maximum axis velocity (axis parameters A-0-0032 and A-0-0033)
	Acceleration	REAL	Percentage of the maximum axis acceleration (axis parameter A-0-0034)
	Deceleration	REAL	Percentage of the maximum axis deceleration (axis parameter A-0-0034)

ML_Robot.library

	Name	Type	Comment
	Jerk	REAL	Percentage of the maximum axis jerk (axis parameter A-0-0036)
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius in per cent of A-0-0230 the axes involved
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-130: MC_MoveDirectAbsolute function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Position	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	Real	>=0.0		100.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0	1000.0	0.0	Rising edge at "Execute"

Tab.5-131: Input behavior of the MC_MoveDirectAbsolute

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:

[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-132: Error numbers of the function block MC_MoveDirectAbsolute

MC_MoveDirectRelative

Brief description The function block "MC_MoveDirectRelative" moves a kinematics point by point from the last commanded TCP position to a specified relative target position in the given coordinate system.



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-133: Library assignment

Interface description

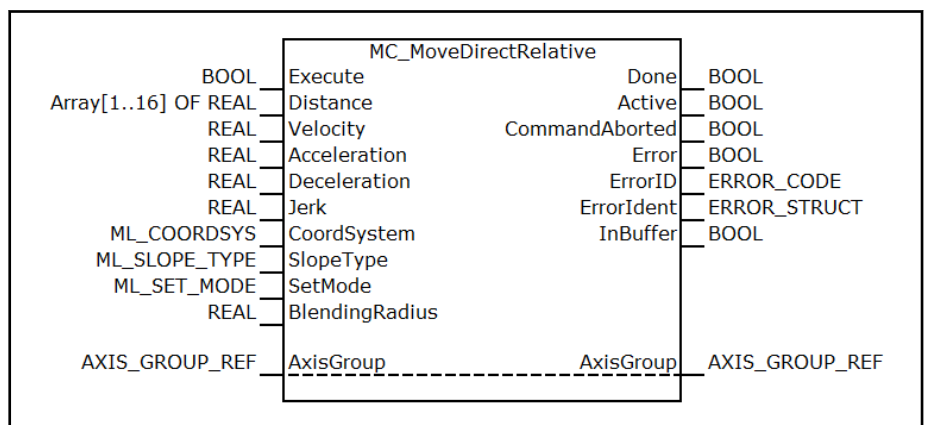


Fig.5-64: Function block MC_MoveDirectRelative

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Distance	ARRAY [1..16] OF REAL	Target position of the motion in relation to the starting point of the motion
	Velocity	REAL	Percentage of the maximum axis velocity (axis parameters A-0-0032 and A-0-0033)

ML_Robot.library

	Name	Type	Comment
	Acceleration	REAL	Percentage of the maximum axis acceleration (axis parameter A-0-0034)
	Deceleration	REAL	Percentage of the maximum axis deceleration (axis parameter A-0-0034)
	Jerk	REAL	Percentage of the maximum axis jerk (axis parameter A-0-0036)
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius in per cent of A-0-0230 the axes involved
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-134: Function block MC_MoveDirectRelative

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Position	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	Real	>=0.0		100.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0	1000.0	0.0	Rising edge at "Execute"

Tab.5-135: Input behavior of MC_MoveDirectRelative

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-136: Error numbers of the function block MC_MoveDirectRelative

MC_MoveLinearAbsolute

Brief description The Coordinated Motion function block "MC_MoveLinearAbsolute" moves a kinematics linearly from the last commanded TCP position to a specified absolute target position in the given coordinate system.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-137: Library assignment

Interface description

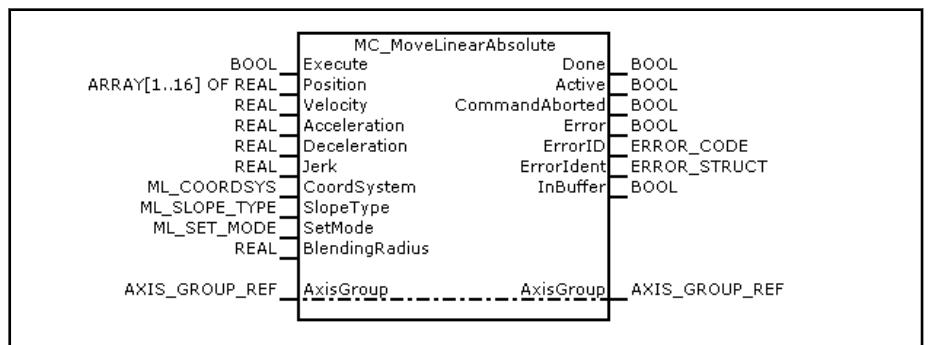


Fig.5-65: MC_MoveLinearAbsolute function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Position	ARRAY [1..16] OF REAL	Absolute target position of the motion
	Velocity	REAL	Maximum path velocity
	Acceleration	REAL	Maximum path acceleration

ML_Robot.library

	Name	Type	Comment
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-138: MC_MoveLinearAbsolute function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Position	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"

Tab.5-139: Input behavior of the MC_MoveLinearAbsolute

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:

MLC_TABLE, ERROR_TABLE 16#0030 , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-140: Error numbers of the function block MC_MoveLinearAbsolute

MC_MoveLinearRelative

Brief description The function block "MC_MoveLinearRelative" moves a kinematics linearly from the last commanded TCP position to a specified relative target position in the given coordinate system.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-141: Library assignment

Interface description

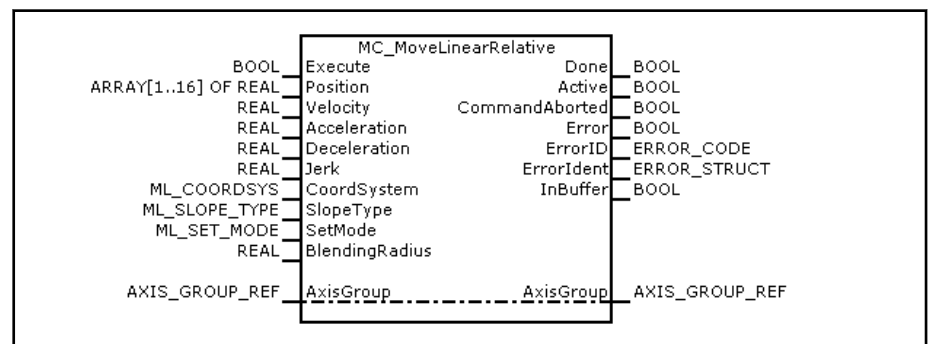


Fig.5-66: MC_MoveLinearRelative function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Position	ARRAY [1..16] OF REAL	Relative target position of the motion
	Velocity	REAL	Maximum path velocity
	Acceleration	REAL	Maximum path acceleration
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion

ML_Robot.library

	Name	Type	Comment
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
	BlendingRadius	REAL	Blending radius
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-142: MC_MoveLinearRelative function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Position	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"

Tab.5-143: Input behavior of the MC_MoveLinearRelative

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#0000	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#0000	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#0000	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-144: Error numbers of the function block MC_MoveLinearRelative

ML_BeltDeSynchronisationWithLimits

Brief description The function block "ML_BeltDeSynchronisationWithLimits" carries out de-synchronization from a belt using motion limit values.
(Refer to [Belt-synchronous Motion](#), page 49).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-145: Library assignment

Interface description

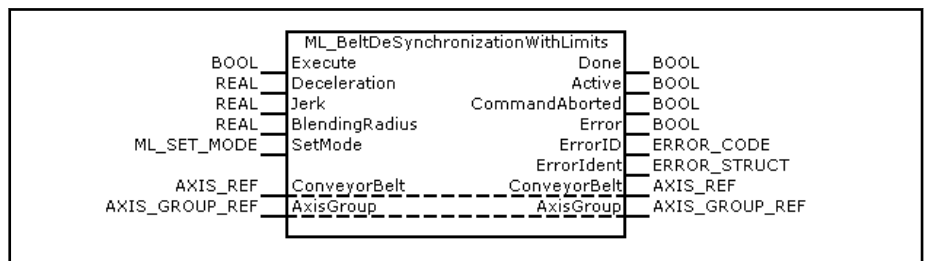


Fig.5-67: Function block ML_BeltDeSynchronisationWithLimits

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	BlendingRadius	REAL	Blending radius
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
VAR_OUTPUT	Done	BOOL	Registering of the transformation completed
	Active	BOOL	Registering of the transformation active
	Error	BOOL	Indicates that an error occurred in the function block instance

ML_Robot.library

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-146: Function block ML_BeltDeSynchronisationWithLimits

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"

Tab.5-147: Input behavior of ML_BeltDeSynchronisationWithLimits

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_BeltSynchronisationWithLimits

Brief description The function block "ML_BeltSynchronisationWithLimits" carries out synchronization with a belt using motion limit values.
(Refer to [Belt-synchronous Motion](#), page 49).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-148: Library assignment

Interface description

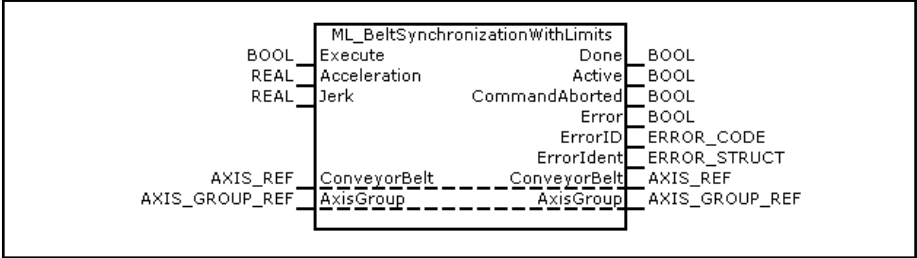


Fig.5-68: Function block ML_BeltSynchronisationWithLimits

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Acceleration	REAL	Maximum path acceleration
	Jerk	REAL	Maximum path jerk
VAR_OUTPUT	Done	BOOL	Registering of the transformation completed
	Active	BOOL	Registering of the transformation active
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab.5-149: Function block ML_BeltSynchronisationWithLimits

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"

Tab.5-150: Input behavior of ML_BeltSynchronisationWithLimits

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_GroupJogCont

Brief description The Coordinated Motion function block ML_GroupJogCont moves a kinematics continuously into one direction.

A falling edge at the "Enable" input stops the motion.

For more information, see: [Manual Mode on page 34](#)



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-151: Library assignment

ML_Robot.library

Interface description

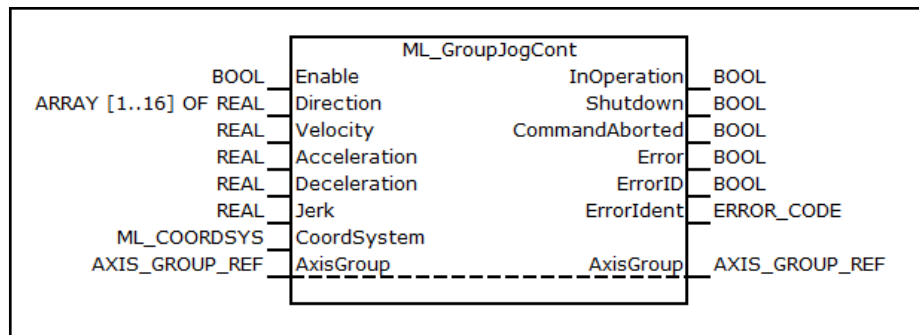


Fig.5-69: Function block ML_GroupJogCont

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Rising edge: start of the motion Falling edge: end of the motion (starts deceleration)
	Direction	ARRAY [1..16] OF REAL	Direction vector of the motion
	Velocity	REAL	Maximum path velocity CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s or in/s
	Acceleration	REAL	Maximum path acceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s² or in/s²
	Deceleration	REAL	Maximum path deceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s² or in/s²
	Jerk	REAL	Maximum path jerk CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s³ or in/s³
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
VAR_OUTPUT	InOperation	BOOL	Motion is executed
	Shutdown	BOOL	Motion is stopped
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-152: Function block ML_GroupJogCont

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Direction	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Enable"
Velocity	REAL	1.0		0.0	Rising edge at "Enable"
Acceleration	REAL	1.0		0.0	Rising edge at "Enable"
Deceleration	REAL	1.0		0.0	Rising edge at "Enable"
Jerk	REAL	>=0.0		0.0	Rising edge at "Enable"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Enable"

Tab.5-153: Input behavior of MC_GroupJogCont

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Terminating Processing on page 117](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-154: Error numbers of the function block ML_GroupJogCont

ML_GroupJogStep

Brief description The Coordinated Motion function block "ML_GroupJogStep" moves a kinematics by one step into a certain direction.

For more information, see: [Manual Mode on page 34](#)

ML_Robot.library



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-155: Library assignment

Interface description

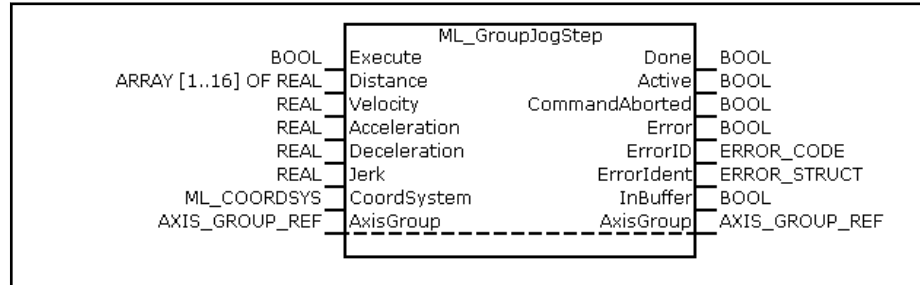


Fig.5-70: Function block ML_GroupJogStep

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Distance	ARRAY [1..16] OF REAL	Relative target position of the motion
	Velocity	REAL	Maximum path velocity CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s or in/s
	Acceleration	REAL	Maximum path acceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s² or in/s²
	Deceleration	REAL	Maximum path deceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s² or in/s²
	Jerk	REAL	Maximum path jerk CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s³ or in/s³
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-156: Function block ML_GroupJogStep

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Distance	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
Velocity	REAL	1.0		0.0	Rising edge at "Execute"
Acceleration	REAL	1.0		0.0	Rising edge at "Execute"
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"

Tab.5-157: Input behavior of ML_GroupJogStep

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR 16#03	16#F0220201	16#00	Invalid control number Cause: Control number of the kinematics to be commanded is incorrect
RESOURCE_ERROR 16#03	16#F0220202	16#00	Invalid axis number Cause: Axis number of the kinematics to be commanded is incorrect
STATE_MACHINE_ERROR 16#05	16#F0290405	16#00	Command in 'Standstill Incomplete' denied Cause: Axes have not been grouped before to the kinematics with the ML_GroupAllAxes function block

Tab.5-158: Error numbers of the function block ML_GroupJogStep

ML_Robot.library

ML_GroupWait

Brief description Due to the Coordinated Motion function block "ML_GroupWait" the kinematics waits for the specified time at the last commanded position.
(Refer to [Waiting](#), page 49).

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-159: Library assignment

Interface description

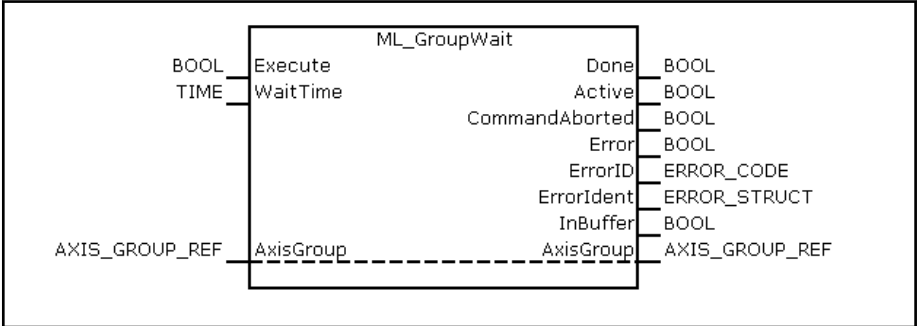


Fig.5-71: Function block ML_GroupWait

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled.
	WaitTime	TIME	Time the robot waits in the last commanded position
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-160: Function block ML_GroupWait

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
WaitTime	TIME	0.0		0.0	Rising edge at "Execute"

Tab.5-161: Input behavior of ML_GroupWait

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_RobotOpenCyclicPos

Brief description The function block "ML_RobotOpenCyclicPos" opens a cyclic command value channel.

(Refer to [Cyclical Command Value Specification](#), page 61).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-162: Library assignment

Interface description

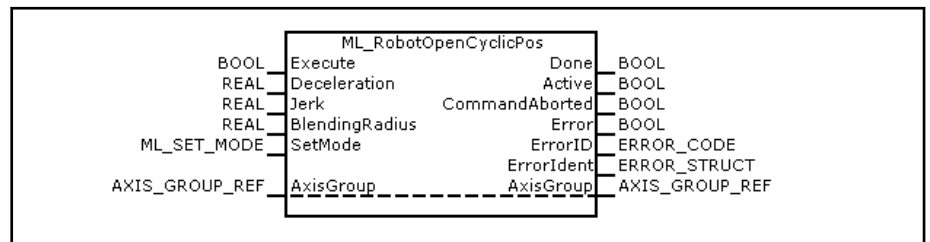


Fig.5-72: Function block ML_RobotOpenCyclicPos

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	BlendingRadius	REAL	Blending radius
	SetMode	ML_SET_MODE	Blending type (VIA, TO)
VAR_OUTPUT	Done	BOOL	"Opening a cyclic command value channel" completed
	Active	BOOL	"Opening a cyclic command value channel" enabled
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-163: Function block ML_RobotOpenCyclicPos

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Deceleration	REAL	1.0		0.0	Rising edge at "Execute"
Jerk	REAL	>=0.0		0.0	Rising edge at "Execute"

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
BlendingRadius	REAL	0.0		0.0	Rising edge at "Execute"
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"

Tab.5-164: Input behavior of ML_RobotOpenCyclicPos

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_RobotWriteCyclicPos

Brief description The function block "ML_RobotWriteCyclicPos" writes a cyclic command value.

(Refer to [Cyclical Command Value Specification on page 61](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/Movement

Tab.5-165: Library assignment

Interface description

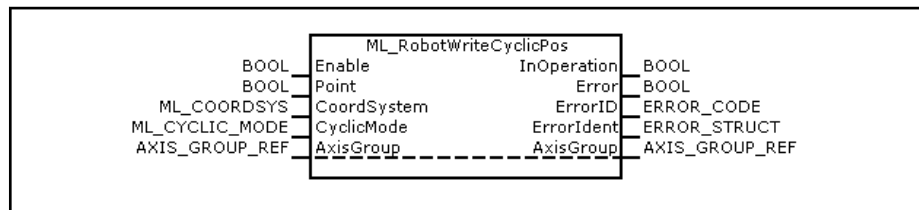


Fig.5-73: Function block ML_RobotWriteCyclicPos

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Point	ARRAY [1..16] OF REAL	Cyclic command position
	CoordSystem	ML_COORDSYS	Coordinate system of the cyclic command position
	CyclicMode	ML_CYCLIC_MODE	Type of command value specification
VAR_OUTPUT	InOperation	BOOL	"Writing a cyclic command value" completed
	Active	BOOL	"Writing a cyclic command value" enabled
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-166: Function block ML_RobotWriteCyclicPos

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Point	ARRAY [1..16] OF REAL			[16(0.0)]	Rising edge at "Execute"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Execute"
CyclicMode	ML_CYCLIC_MODE			CY-CLIC_MODE_POS_ABS	Rising edge at "Execute"

Tab.5-167: Input behavior of ML_RobotWriteCyclicPos

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing](#) on page. 123

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

5.3.6 Parameter Function Blocks

Accessing Kinematic Data

Structure of and access to parameters of the PLC program

The kinematic data can be accessed via read (MB_GroupRead***) and sometimes write (MB_GroupWrite***) parameter access:

- K-parameters (kinematics parameters) that relate to a kinematics

The structure of all IndraMotion parameters is identical:

Example: K-0-0001, with "K" as a parameter type identifier which is followed by the group number "0" and the four-digit parameter number "0001".

Based on the parameter number, access to these parameters is possible in the PLC program.

If the parameter number is known:

Program:

```
VAR_GLOBAL CONSTANT
  FP_K_0_0000: MB_IDN:= 184683593728 ; // Dummy K Parameter
  FP_K_0_0001: MB_IDN:= 184683593729 ; // Logical kinematics number
  FP_K_0_0002: MB_IDN:= 184683593730 ; // Kinematics name
  FP_K_0_0003: MB_IDN:= 184683593731 ; // Maximum path speed
  //...
  FP_K_0_2301: MB_IDN:= 184683596029 ; // PLC integer input from RCL
  FP_K_0_2302: MB_IDN:= 184683596030 ; // PLC real input from RCL
END_VAR
```

As an alternative to the constant, the number can also be attached to the input "ParameterNumber" of the respective function block.

Offsets der Parameter

Parameter	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000
A-0-0000	4294967296	16#1 0000 0000
C-0-0000	8589934592	16#2 0000 0000
K-0-0000	184683593728	16#2B 0000 0000
M-0-0000	193273528320	16#2D 0000 0000

ML_Robot.library

Parameter	Decimal	Hexadecimal
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab.5-168: Parameter offsets

Access to the K-parameters:

- Simultaneously to the read/write parameter blocks for A and C-parameters, there are group read/write parameter blocks for K-parameters.

The address identifier is automatically generated by IndraWorks MLC (kinematics name/kinematic number), type AXIS_GROUP_REF, RIL_CommonTypes.library.

Example: MB_GroupReadParameter

Function block reference

```
Kin_001:AXIS_GROUP_REF:=(CntrlNo:=LOCAL_CNTRL,GroupNo:=KINEMATIC_1); //Kinematik Kin_001, Nr. 1
```

MB_GroupReadListParameter

Brief description The function block "MB_GroupReadListParameter" returns the value of a parameter specified by the user.

A pointer addressing a memory space of no more than 64 kB is transferred to the function block as defined **by the user**. In addition, the required use data length is transferred (maximum number of readable bytes).

The return value is the number of bytes actually read.



The data type of the elements of the parameter to be read and the data type of the elements of the ARRAY have to match.



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-169: Library assignment

Interface description

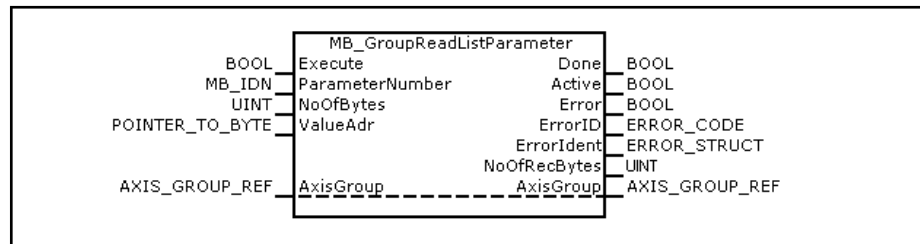


Fig.5-74: FB MB_GroupReadListParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute".
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
	NoOfBytes	UINT	Maximum number of data to be read in bytes

	Name	Type	Comment
	ValueAdr	POINTER TO BYTE	It points to the initial address from where the data read is to be stored (maximum NoOfBytes).
VAR_OUTPUT	Done	BOOL	Valid data is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	NoOf RecBytes	UINT	Number of bytes actually read

Tab.5-170: Interface of FB MB_GroupReadListParameter

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

In addition, the following can be displayed:

- F00D1205, Error, cannot handle a single parameter
- F90D1204, Error, no memory available

MB_GroupReadParameter

Brief description The function block "MB_GroupReadParameter" returns the value of a parameter specified by the user.

The return value is represented as DINT number.



Any change, e.g. of the parameter number or the kinematics at the input of the function block, requires another 0/1 transition at "Enable".



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-171: Library assignment

Interface description

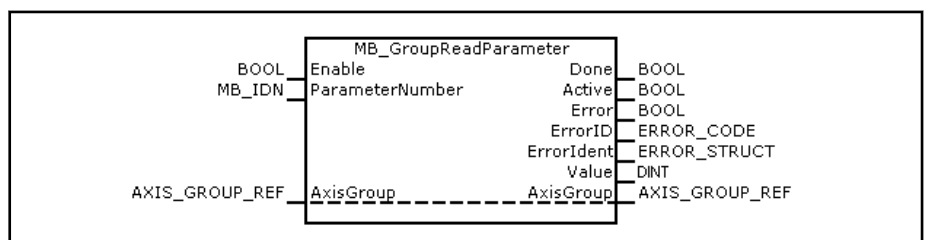


Fig.5-75: FB MB_GroupReadParameter

ML_Robot.library

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	DINT	4 byte value, represented as DINT

Tab.5-172: Interface of FB MB_GroupReadParameter

- Signal-time diagram
- Refer to [Inputs and Outputs of State-controlled Function Blocks, Terminating Processing on page 117](#).
- Error handling
- The function block generates the following types of error messages:
- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
 - [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

MB_GroupReadRealParameter

Brief description

The function block "MB_GroupReadRealParameter" returns the value of a parameter specified by the user.
The return value is represented as REAL number.



Any change, e.g. of the parameter number or the kinematics at the input of the function block, requires another 0/1 transition at "Enable".



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-173: Library assignment

Interface description

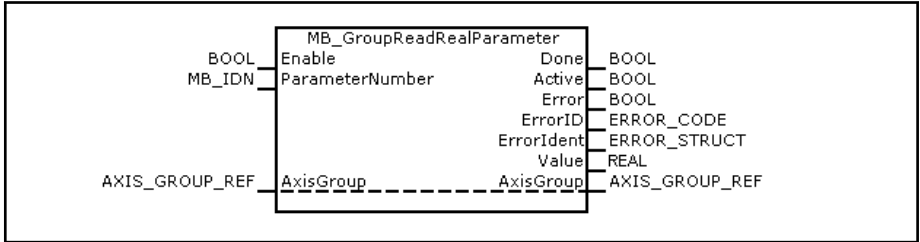


Fig.5-76: FB MB_GroupReadRealParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	REAL	4 byte value represented as REAL number

Tab.5-174: Interface of the MB_GroupReadRealParameter FB

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Terminating Processing on page 117](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

MB_GroupReadSercosAttribute

Brief description The function block "MB_GroupReadSercosAttribute" enables reading the sercos attributes of a K-parameter.
The return value is represented as a DWORD.



Any change, e.g. of the parameter number or the kinematics at the input of the function block, requires another 0/1 transition at "Enable".



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-175: Library assignment

Interface description

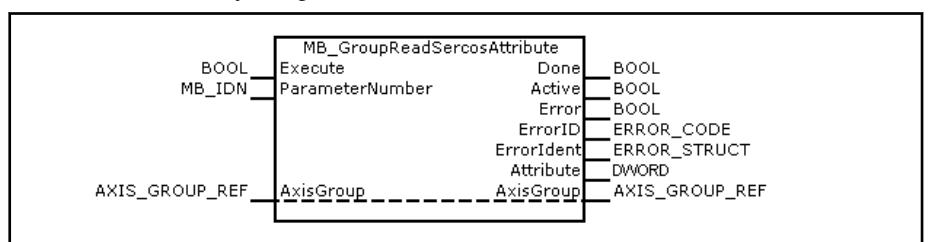


Fig.5-77: FB MB_GroupReadSercosAttribute

ML_Robot.library

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute".
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	DWORD	4 byte value, contains sercos attributes of the K-parameter

Tab.5-176: Interface of the MB_GroupReadSercosAttribute function block

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

In addition, the following can be displayed:

- F00D1230, Error ReadSERCOSAttribute, no memory available

MB_GroupReadSercosDataStatus

Brief description The function block "MB_GroupReadSercosDataStatus" returns the status of a K-parameter specified by the user.

The return value is represented as an ENUM value (INT number).



Any change, e.g. of the parameter number or the kinematics at the input of the function block, requires another 0/1 transition at "Enable".



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-177: Library assignment

Interface description

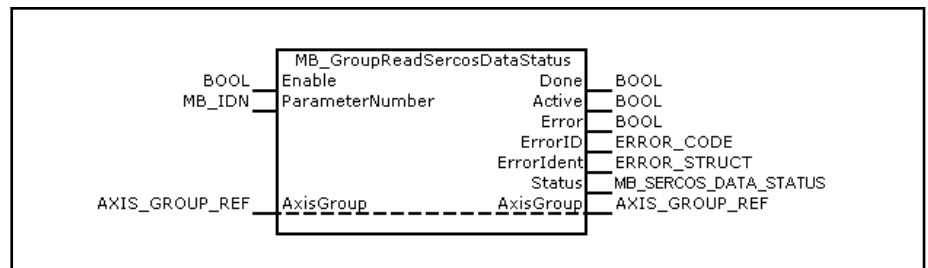


Fig.5-78: FB MB_GroupReadSercosDataStatus

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Status	MB_SERCOS_DATA_STATUS	Status of the selected parameter

Tab.5-178: Interface of FB MB_GroupReadSercosDataStatus

Brief description With this function block and a [MB_GroupWriteParameter on page 198](#), it is possible to trigger drive commands and to track their processing:

- 0 - Command not set and not released
- 1 - Command set and released
- 3 - Command processed correctly
- 5 - Command processing interrupted
- 7 - Command processing running
- 15 - Error, command processing impossible
- 256 - Value of the parameter beyond its minimum/maximum range

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Terminating Processing on page 117](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

MB_GroupReadStringParameter

Brief description The function block "MB_GroupReadStringParameter" returns the value of a parameter specified by the user.

The return value is displayed as a character string (STRING) according to with its actual length with a maximum of 255 characters.

ML_Robot.library

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-179: Library assignment

Interface description

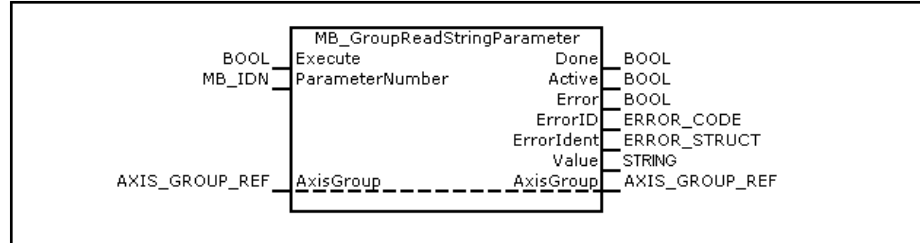


Fig.5-79: FB MB_GroupReadStringParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	It starts reading the text of the parameter with a rising edge
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	STRING	Character string with a maximum of 255 characters

Tab.5-180: Interface of the MB_GroupReadStringParameter function block

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

In addition, the following can be displayed:

- F00D1218, Error, no memory available
- F00D1219, Error, cannot handle a single parameter
- F00D1220, Unsupported selection of sercos element
- F00D1221, Timeout on parameter access

MB_GroupWriteListParameter

Brief description

The function block "MB_GroupWriteListParameter" writes memory contents having a defined length (NoOfBytes) and provided with the starting address (ValueAdr) to a parameter specified by the user.



The specified number of bytes must exactly match the data volume to be transferred!

If this number is too high, data waste is transferred. If it is too small, data is missing in the target parameter!



The data type of the elements of the parameter to be written and the data type of the elements of the ARRAY have to match.



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-181: Library assignment

Interface description

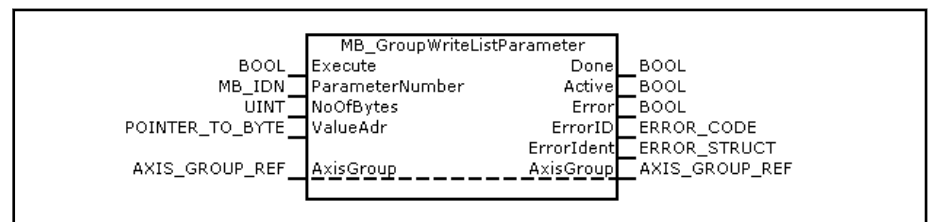


Fig.5-80: FB MB_GroupWriteListParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
	NoOfBytes	UINT	Number of bytes to be written
	ValueAdr	POINTER TO BYTE	Pointer to the initial address of the memory in which the data to be written is saved.
VAR_OUTPUT	Done	BOOL	The value was successfully written to the control/drive
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-182: Interface of FB MB_GroupWriteListParameter

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

ML_Robot.library

In addition, the following can be displayed:

- F00D1207, Error, data greater 64KBytes
- F00D1210, Error, cannot handle a single parameter
- F00D1211, Error ReadSERCOSDataStatus, no memory available
- F90D1209, Error, no memory available

MB_GroupWriteParameter

Brief description The function block "MB_GroupWriteParameter" writes a value to a parameter specified by the user.



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-183: Library assignment

Interface description

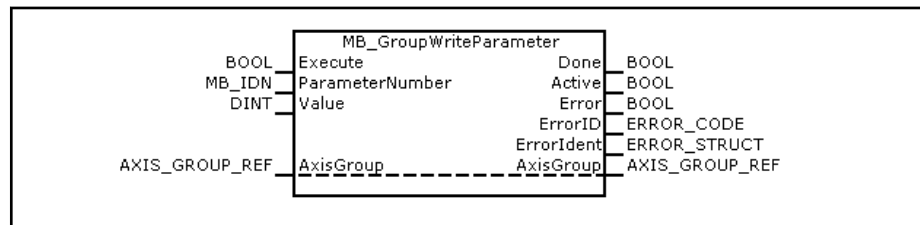


Fig.5-81: FB MB_GroupWriteParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
	Value	DINT	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-184: Interface of the MB_GroupWriteParameter FB

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

MB_GroupWriteRealParameter

Brief description The function block "MB_GroupWriteRealParameter" writes a value to a parameter specified by the user.



Only for K-parameters!

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-185: Library assignment

Interface description

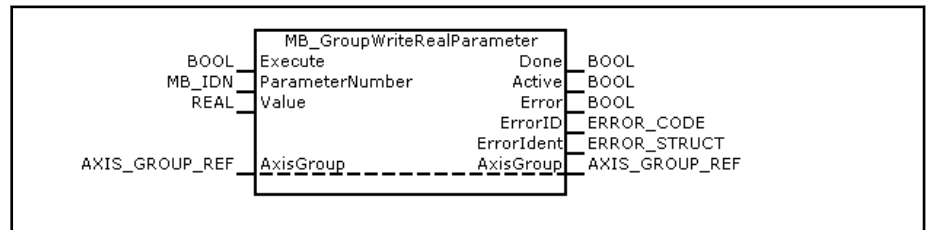


Fig.5-82: FB MB_GroupWriteRealParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
	Value	REAL	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-186: Interface of the function block MB_GroupWriteRealParameter

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

MB_GroupWriteStringParameter

Brief description The function block "MB_GroupWriteStringParameter" writes a character string with up to 255 characters to a parameter specified by the user.



Only for K-parameters!

ML_Robot.library

Library	Range
ML_Robot	Kinematics/Parameters

Tab.5-187: Library assignment

Interface description

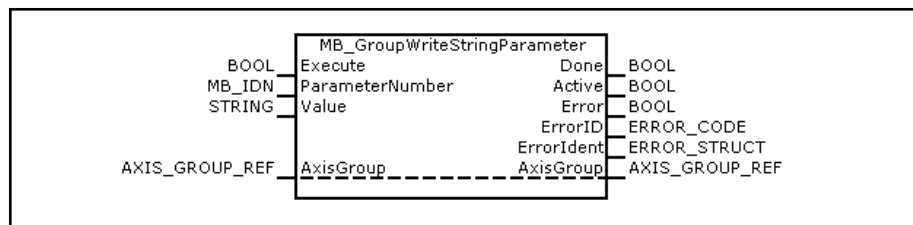


Fig.5-83: FB MB_GroupWriteStringParameter

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Contains information on the concrete kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the desired K-parameter (see Accessing Kinematic Data on page 189)
	Value	STRING	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-188: Interface of FB MB_GroupWriteStringParameter

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010 on page 360](#), error when accessing the parameter (e.g. parameter is not supported)
- [MLC_TABLE, ERROR_TABLE 16#0030 on page 351](#), error when accessing the axis/control

In addition, the following can be displayed:

ErrorIdent / Additional1	Description
F00D1222	Data larger than 255 byte
F00D1223	Error while reading the attribute
F00D1224	No memory available
F00D1225	Data of single parameter requested
F00D1226	Error read attribute
F00D1227	Error when writing the date

Tab.5-189: Error numbers, caused by MB_GroupWriteStringParameter

5.3.7 RCL Program Function Blocks

MB_IRDProgSelect

Brief description With the function block "MB_IRDProgSelect", an IRD program (Industrial Robot Data) is selected.
The state machine of the kinematics switches to the state Programmed Motion, Program Selected.



This function block can only be used for Robot Control V1.



Before MB_IRDProgSelect can be executed, all configured axes have to be assigned to the group (also refer to ML_PLCopen, MC_AddAxisToGroup and ML_GroupAllAxes).

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-190: Library assignment

Interface description

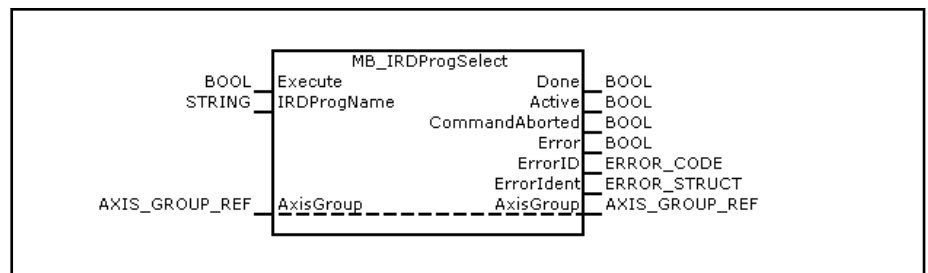


Fig.5-84: FB MB_IRDProgSelect

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	IRDProgName	STRING[8]	Name of the IRD program without extension (.ird).
VAR_OUTPUT	Done	BOOL	The IRD program has been successfully selected.
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandA-borted	BOOL	Command aborted by a following command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-191: Interface of the MB_IRDProgSelect FB

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_Robot.library

MB_IRDProgStart

Brief description The function block "MB_IRDProgStart" starts a selected IRD program.
The state machine of the kinematics switches to the state Programmed Motion, Program Running.



This function block can only be used for Robot Control V1.



Before applying the function block, [MB_IRDProgSelect](#) on [page 201](#) has to be called.

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-192: Library assignment

Interface description

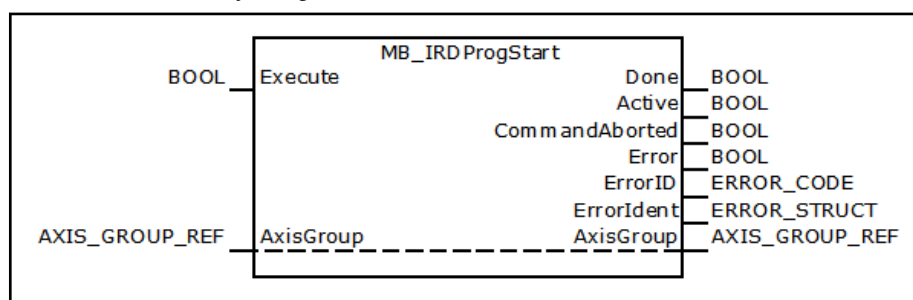


Fig.5-85: FB MB_IRDProgStart

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	The selected IRD program has been started successfully.
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-193: Interface of the MB_IRDProgStart FB

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on [page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

MB_IRDProgStop

Brief description The function block "MB_IRDProgStop" stops and ends a running IRD program.
The state machine of the kinematics switches to the state Programmed Motion, Program Stopping.



This function block can only be used for Robot Control V1.



After the execution of the FB MB_IRDProgStop, no IRD program is selected anymore!

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-194: Library assignment

Interface description

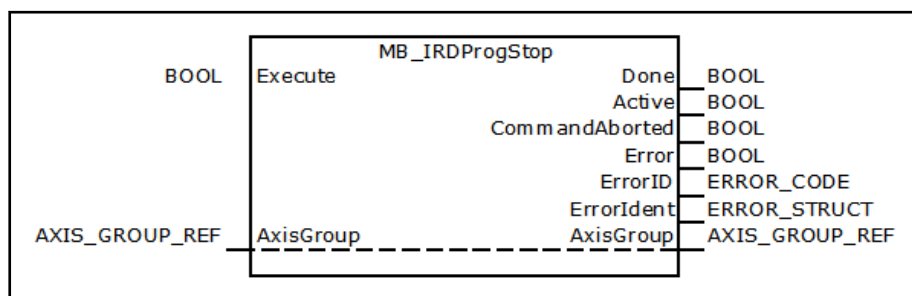


Fig.5-86: FB MB_IRDProgStop

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	The IRD program has been successfully stopped and ended.
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandA-borted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-195: Interface of the MB_IRDProgStop FB

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_RoCoDir

Brief description

The function block "ML_RoCoDir" returns a list of file information.
The user can preselect file information using a search mask.



This function block can only be used for Robot Control V1.

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-196: Library assignment

ML_Robot.library

Interface description

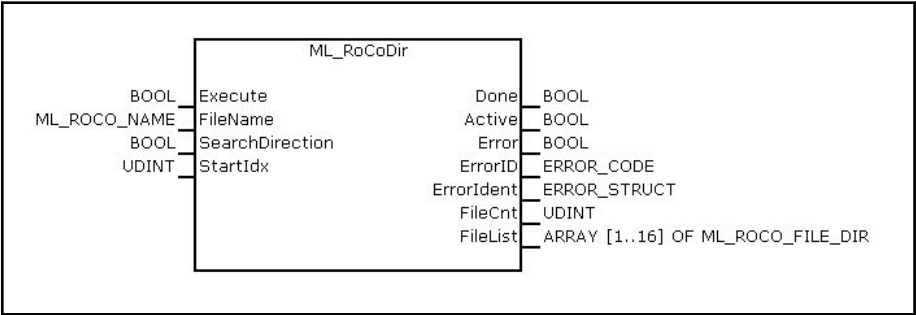


Fig.5-87: Interface of the FB ML_RoCoDir

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (once, edge-controlled)
	FileName	ML_ROCO_NAME	File name is only read at rising edge (ML_ROCO_NAME on page 247)
	SearchDirection	BOOL	If FALSE, the files are listed in ascending order. If TRUE, the files are listed in descending order.
	StartIdx	UDINT	Index from which the files should be listed.
VAR_OUTPUT	Done	BOOL	Processing completed without errors, output data is valid
	Active	BOOL	Processing not yet completed, output data invalid
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	FileCnt	UDINT	Number of invalid files in the output FileList
	FileList	AR- RAY[1...16]ML_RO- CO_FILE_DIR	List with found files (ML_ROCO_FILE_DIR on page 247)

Tab.5-197: Interface of FB ML_RoCoDir

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL	-	-	FALSE	Continuous
FileName	ML_ROCO_NAME	``	n.def.	``.``	Rising edge at "Execute"
SearchDirection	BOOL	-	-	FALSE	Rising edge at "Execute"
StartIdx	UDINT	1	4294967295	1	Rising edge at "Execute"

Functional description The function block starts to search for the file name "FileName" with a rising edge of the "Execute" input of the Robot Control (RoCo) file system.

Placeholders are allowed for the file names, but only one for file name and extension.

"?" is not a valid placeholder character.

Valid examples are

```
*.pkt  
MyPoints.*  
*.*
```

Invalid names such as

```
My*.pkt  
My??ints.pkt  
Mypoints.p*
```

If the function block is executed successfully, the "FileCnt" output returns the number of files found.

The file information can be found in the "FileList" array.

If the "FileCnt" output has a value of 16, more files may exist. These can be read out by calling the block again with a modified "StartIdx".

To read out all file information, reading continues until the value of "FileCnt" is less than 16.

This is shown in the following example:

Program:

```
VAR  
  fb: ML_RoCoDir;  
  ex: BOOL;  
  FileName: STRING(44) := '.*.*';  
  StartIndex: UDINT := 1;  
  Done: BOOL;  
  Active: BOOL;  
  Error: BOOL;  
  FileCnt: UDINT := 1;  
  BigFileList: ARRAY [1..160] OF ML_ROCO_FILE_DIR;  
  Direction: BOOL := FALSE; (* List files in ascending order *)  
  fbState: DWORD := 0;  
  old_ex: BOOL;  
  index: UDINT;  
  i: UDINT;  
END_VAR;  
  
PROGRAM getAllFiles  
  (* getAllFiles must be called every loop until fbState is >= 10 *)  
  
  CASE fbState OF  
    0:  
      index := 1;  
      StartIndex := 1;  
      fb(Execute:= FALSE);  
      fbState := 1;  
  
    1:  
      fb(Execute:= TRUE,  
        FileName:= FileName,  
        SearchDirection:= Direction,  
        StartIdx:= StartIndex,  
        Done=> Done,  
        Active=> Active,  
        Error=> Error,  
        ErrorID=> ,  
        ErrorIdent=> ,  
        FileCnt=> FileCnt,  
        FileList=> BigFileList);  
  
      IF (Done = TRUE) THEN  
        (* Copy files information from function block to global list *)  
        FOR i := 1 TO FileCnt DO  
          BigFileList[index] := fb.FileList[i];  
          index := index + 1;  
          IF (index > 160) THEN  
            (* Global array full *)  
            fbState := 20;  
            EXIT;  
          END_IF  
        END_FOR  
      END_IF  
    20:  
      EXIT;  
  END_CASE  
END_PROGRAM
```

ML_Robot.library

```

    (* Increment start index for next loop *)
    StartIndex := StartIndex + FileCnt;
    (* Check if no more files exist *)
    IF (FileCnt < 16) THEN
        fbState := 10;
    END_IF
    (* Reset function block for next loop *)
    fb(Execute:= FALSE);
END_IF

IF (Error = TRUE) THEN
    (* Error occurred *)
    fbState := 30;
END_IF

10:
    (* Read all files *)
    ;
20:
    (* Read all files, but global buffer full, more filea available *)
    ;
30:
    (* Function block had an error *)
    ;
END_CASE
END_PROGRAM
```

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_RoCoGetPoint

Brief description The function block "ML_RoCoGetPoint" returns information on a point to be selected.
The name of the point file and the point have to be specified. The indices of each dimension can be specified for arrays.

 This function block can only be used for Robot Control V1.

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-198: Library assignment

Interface description

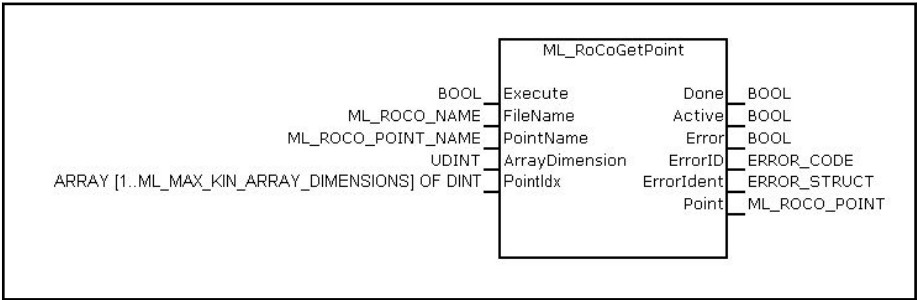


Fig.5-88: Interface of the function block ML_RoCoGetPoint

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (once, edge-controlled)
	FileName	ML_ROCO_NAME	File name is only read at rising edge (ML_ROCO_NAME on page 247)

	Name	Type	Comment
	PointName	ML_ROCO_POINT_NAME	Point name is only read at a rising edge (ML_ROCO_POINT_NAME on page 248)
	ArrayDimension	UDINT	Number of dimensions (only for arrays, to be set to 0 for the individual points)
	PointIdx	ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT	Index of each dimension (only for arrays; this input is not evaluated for individual points)
VAR_OUTPUT	Done	BOOL	Processing completed without errors, output data is valid
	Active	BOOL	Processing not yet completed, output data invalid
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Point	ML_ROCO_POINT	In case of valid execution, the values of the found point (ML_ROCO_POINT on page 248)

Tab.5-199: Interface of the ML_RoCoGetPoint function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL	-	-	FALSE	Continuous
FileName	ML_ROCO_NAME	''	n.def.	''	Rising edge at "Execute"
PointName	ML_ROCO_POINT_NAME	''	n.def.	''	Rising edge at "Execute"
ArrayDimension	UDINT	1	4294967295	0	Rising edge at "Execute"
PointIdx	ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT	-2147483648	2147483647	0	Rising edge at "Execute"

Tab.5-200: Min/max/default values of the function block ML_RoCoGetPoint

Functional description

With a rising edge at the "Execute" input, the function block starts to search for the specified file name "FileName" in the Robot Control V1 file system.

When the file was found, the specified point name ("PointName") is searched for. If this is also found, the values are written to the Point structure at the output. This is only possible if the point exists. If this is not the case, an error is generated.

In case of arrays, the number of array dimensions and the indices of the point have to be transferred. First, it is checked whether the point with its indices is within valid limits.

The following example explains the writing:

Program:

```
VAR
fb3: ML_RoCoGetPoint;
  FileName:      ML_ROCO_NAME := '';
  PointName:     ML_ROCO_POINT_NAME := '';
  ArrayDimension: UDINT := 1;
```

ML_Robot.library

```

Done: BOOL;
Active: BOOL;
Error: BOOL;
Point: ML_ROCO_POINT;
PointIdx: ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT;
END_VAR;

PROGRAM GetPoint
(* Reset function block *)
fb3(Execute:= FALSE);

FileName := 'Beta.PKT';
ArrayDimension := 0;
PointName := 'A';

(* Reading a single point *)
Fb3(Execute:= TRUE,
  FileName:= FileName,
  PointName:= PointName,
  ArrayDimension:= ArrayDimension,
  PointIdx:= PointIdx,
  Done=> Done,
  Active=> Active,
  Error=> Error,
  ErrorID=> ,
  ErrorIdent=> ,
  Point=> Point);
IF (Done = TRUE) THEN
  (* Point successfully read *)
  ;
END_IF

IF (Error = TRUE) THEN
  (* Error occurred *)
  ;
END_IF

(*Reading a point in an array (B[1][2]) *)
PointName := 'B';
ArrayDimension := 2;
PointIdx[1] := 1;
PointIdx[2] := 2;

fb3(Execute:= FALSE);
fb3(Execute:= TRUE,
  FileName:= FileName,
  PointName:= PointName,
  ArrayDimension:= ArrayDimension,
  PointIdx:= PointIdx,
  Done=> Done,
  Active=> Active,
  Error=> Error,
  ErrorID=> ,
  ErrorIdent=> ,
  Point=> Point);

IF (Done = TRUE) THEN
  (* Point successfully read *)
  ;
END_IF

IF (Error = TRUE) THEN
  (* Error occurred *)
  ;
END_IF
END_PROGRAM

```

At the "Point" output, the values "PointName", "PointValue" and "PointDef" are entered if a point was found. If there is one single point, the values "ArrayDimension" and "ArrayIndicesLimits" are preset to 0. If there are points within arrays, these values are assigned the actual values for the array "Dimension" or with the indices of each dimension.

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_RoCoGetPointList

Brief description The function block "ML_RoCoGetPointList" returns a list of point information. The user can select the point table by specifying the file name.



This function block can only be used for Robot Control V1.

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-201: Library assignment

Interface description

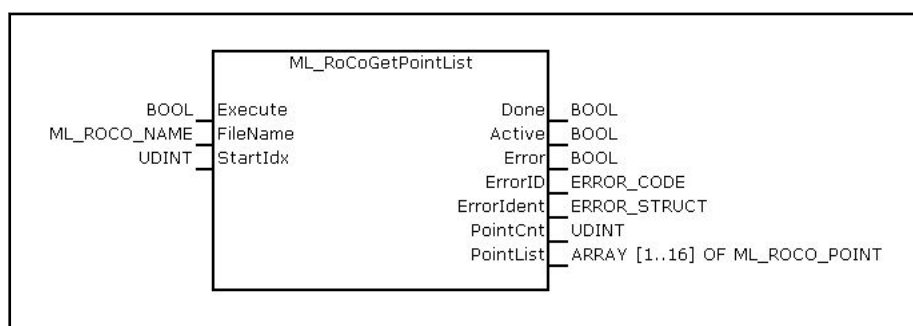


Fig.5-89: Interface of the function block ML_RoCoGetPointList

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (once, edge-controlled)
	FileName	ML_ROCO_NAME	File name, is only read at a rising edge (ML_ROCO_NAME on page 247)
	StartIdx	UDINT	Index from which points are to be listed
VAR_OUTPUT	Done	BOOL	Processing completed without errors, output data is valid
	Active	BOOL	Processing not yet completed, output data invalid
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	PointCnt	UDINT	Number of valid points in the PointList
	PointList	ARRAY[1...16] ML_ROCO_POINT	List with found points (ML_ROCO_POINT on page 248)

Tab.5-202: Interface of the ML_RoCoGetPointList function block

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL	-	-	FALSE	Continuous
FileName	ML_ROCO_NAME	``	n.def.	``	Rising edge at "Execute"
StartIdx	UDINT	1	4294967295	1	Rising edge at "Execute"

Functional description

With a rising edge at the Execute input, the function block starts to search for the specified file name "FileName" in the Robot Control V1 file system. If the file was found, the points in the file are read out starting from "StartIdx". The read points are entered into the "PointList" array and the number is displayed at the "PointCnt" output.

Up to 16 points may be read. If there are more than 16 points in the file, increase the StartIdx to read the additional points.

The following example explains the reading:

Program:

```

VAR
Fb1: ML_RoCoGetPointList;
  FileName: ML_ROCO_NAME := '';
  StartIndex: UDINT := 1;
  Done: BOOL;
  Active: BOOL;
  Error: BOOL;
  PointCnt: UDINT := 1;
  BigPointList: ARRAY [1..160] OF ML_ROCO_POINT;
  fbState: DWORD := 0;
  old_ex: BOOL;
  index: UDINT;
  i: UDINT;
END_VAR;

PROGRAM getAllFiles
(* getAllFiles must be called every loop until fdState is >= 10 *)
CASE fbState OF
0:
  index := 1;
  StartIndex := 1;
  fb1(Execute:= FALSE);
  fbState := 1;
1:
  FileName := 'alpha.PKT';
  fb1(Execute:= TRUE,
    FileName:= FileName,
    StartIdx:= StartIndex,
    Done=> Done,
    Active=> Active,
    Error=> Error,
    ErrorID=> ,
    ErrorIdent=> ,
    PointCnt=> PointCnt,
    PointList=> );
  IF (Done = TRUE) THEN
    (* Copy files information from function block to global list *)
    FOR i := 1 TO PointCnt DO
      BigPointList[index] := fb1.PointList[i];
      index := index + 1;
      IF (index > 160) THEN
        (* Global array full *)
        fbState := 20;
        EXIT;
      END_IF
    END_FOR
    (* Increment start index for next loop *)
    StartIndex := StartIndex + PointCnt;
    (* Check if no more files exist *)
    IF (PointCnt < 16) THEN
      fbState := 10;
    END_IF
    (* Reset function block for next loop *)
  
```

```
        fb1(Execute:= FALSE);
    END_IF

    IF (Error = TRUE) THEN
        (* Error occurred *)
        fbState := 30;
    END_IF
10:  (* Read all Points *)
    ;
20:  (* Read all Points, but global buffer full, more Points available *)
    ;
30:  (* Function block had an error *)
    ;
END_CASE
END_PROGRAM
```

At the "PointList" output, the values "PointName", "PointValue" and "Point-Def" are entered for individual points. The value in ArrayDimension is fixed to 0.

If it is a point array, the number of array dimensions is specified in "ArrayDimension" and the start and end indices of all dimensions are set in "ArrayIndices". The point with the smallest index of each dimension is returned in "PointValue" in this case. Therefore, it is possible to read all individual points and to determine the size of all arrays.

All values of an array can then be read using the ML_GetPoint function block.

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_RoCoSetPoint

Brief description The function block "ML_RoCoSetPoint" writes a point specifying the point file and the point name.

The indices of each dimension can be specified for arrays.

 This function block can only be used for Robot Control V1.

Library	Range
ML_Robot	Kinematics/RCL/RCL program

Tab.5-203: Library assignment

Interface description

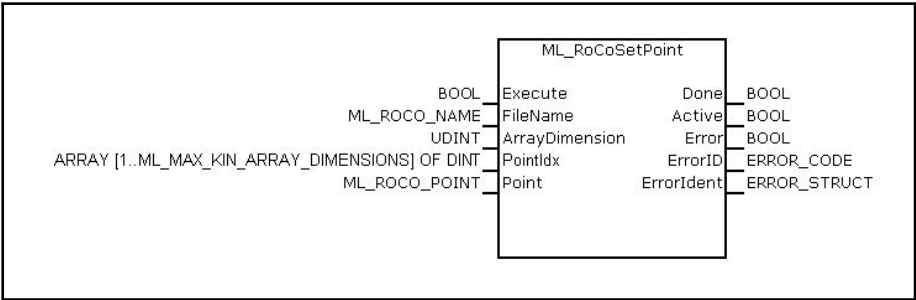


Fig.5-90: Interface of the function block ML_RoCoSetPoint

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (once, edge-controlled)
	FileName	ML_ROCO_NAME	File name is only read at rising edge (ML_ROCO_NAME on page 247)
	ArrayDimension	UDINT	Number of dimensions (only for arrays, to be set to 0 for the individual points)
	PointIdx	ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT	Index of each dimension (only for arrays; this input is not evaluated for individual points)
	Point	ML_ROCO_POINT	Point structure with name and value. (ML_ROCO_POINT on page 248)
VAR_OUTPUT	Done	BOOL	Processing completed without errors, output data is valid
	Active	BOOL	Processing not yet completed, output data invalid
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab.5-204: Interface of the ML_RoCoSetPoint function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL	-	-	FALSE	Continuous
FileName	ML_ROCO_NAME	''	n.def.	''	Rising edge at "Execute"
ArrayDimension	UDINT	0	4294967295	0	Rising edge at "Execute"
ArrayIdx	ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT			0	Rising edge at "Execute"

Tab.5-205: Min/max/default values of the function block ML_RoCoSetPoint

Functional description

With a rising edge at the "Execute" input, the function block starts to search for the specified file name "FileName" in the Robot Control V1 file system. If the file was found, the specified point name is searched for. If this is also found, the transferred values are also written in the point table. This is only possible if the point exists, which means that no new points can be created.

In case of arrays, the array dimensions and the indices of the point have to be transferred. First, it is checked whether the point with its indices is within valid limits.

The following example explains the writing:

Program:

```
VAR
fb2: ML_RoCoSetPoint;
  FileName: ML_ROCO_NAME := '';
  ArrayDimension: UDINT := 1;
  Done: BOOL;
```

```

Active: BOOL;
Error: BOOL;
PointCnt: UDINT := 1;
Point: ML_ROCO_POINT;
PointIdx: ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSIONS] OF DINT;
END_VAR;

PROGRAM getAllFiles
(* Reset function block *)
fb2(Execute:= FALSE);

FileName := 'Beta.PKT';
Point.PointName := 'A';
Point.PointValue[1] := 111.0;
Point.PointValue[2] := 222.0;
Point.PointValue[3] := 333.0;
Point.PointValue[4] := 444.0;
Point.PointValue[5] := 4.0;
Point.PointValue[6] := 5.0;
Point.PointValue[7] := 6.0;
Point.PointValue[8] := 7.0;
Point.PointValue[9] := 8.0;
Point.PointValue[10] := 9.0;
Point.PointValue[11] := 10.0;
Point.PointValue[12] := 11.0;
Point.PointValue[13] := 12.0;
Point.PointValue[14] := 13.0;
Point.PointValue[15] := 14.0;
Point.PointValue[16] := 15.0;
ArrayDimension := 0;

(* Writing a single point *)
fb2(Execute:= TRUE,
    FileName:= FileName,
    ArrayDimension:= ArrayDimension,
    PointIdx:= PointIdx,
    Point:= Point,
    Done=> Done,
    Active=> Active,
    Error=> Error,
    ErrorID=> ,
    ErrorIdent=> );

(* Writing a point in an array (B[1][2]) *)
Point.PointName := 'B';
ArrayDimension := 2;
PointIdx[1] := 1;
PointIdx[2] := 2;

fb2(Execute:= FALSE);
fb2(Execute:= TRUE,
    FileName:= FileName,
    ArrayDimension:= ArrayDimension,
    PointIdx:= PointIdx,
    Point:= Point,
    Done=> Done,
    Active=> Active,
    Error=> Error,
    ErrorID=> ,
    ErrorIdent=> );

IF (Done = TRUE) THEN
    (* Point successfully written *)
;
END_IF

IF (Error = TRUE) THEN
    (* Error occurred *)
;
END_IF
END_PROGRAM

```

All values of an array can be called using the ML_RoCoGetPoint block.

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling

The function block generates the following types of error messages:

ML_Robot.library

[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

5.3.8 RCL User Interface Function Blocks

RCL User Interface, Overview

The folder functions allow access to the "User Data" of the kinematics. They expand access via shared memory (ML_KINEMATICSDATA_SM).



This function can only be used for Robot Control V1.

Extract of the ML_KINEMATICSDATA_SM structure

```

TYPE ML_KINEMATICSDATA_SM :
STRUCT
  //....
  dwUserBits_i      :DWORD;                // -RD- User data, PLC input / RoCo output
  adiUserDInt_i     :ARRAY[1..4] OF DINT;   // -RD- User data, PLC input / RoCo output
  arUserReals_i     :ARRAY[1..4] OF REAL;    // -RD- User data, PLC input / RoCo output

  stJogCmd_q       :ML_KIN_JOG;            // -RW- Command interface for jog mode

  dwUserBits_q      :DWORD;                // -RW- User data, PLC output / RoCo input
  adiUserDInt_q     :ARRAY[1..4] OF DINT;   // -RW- User data, PLC output / RoCo input
  arUserReals_q     :ARRAY[1..4] OF REAL;    // -RW- User data, PLC output / RoCo input
  arUserPoints_q    :ARRAY[1..4] OF ML_ARR_COORDS; // -RW- User data, PLC output / RoCo input
END_STRUCT
END_TYPE

```

ML_KINEMATICSDATA_SM	Function	index
dwUserBits_i	ML_KinGetUserBit	1...32
	ML_KinGetUserBit	33...192
dwUserBits_i	ML_KinGetUserBit_DWord	1
	ML_KinGetUserBit_DWord	2...6
adiUserDInt_i[1..4]	ML_KinGetUserDInt	1...4
	ML_KinGetUserDInt	5...50
arUserReals_i[1..4]	ML_KinGetUserReal	1...4
	ML_KinGetUserReal	5...50
dwUserBits_q	ML_KinSetUserBit	1...32
	ML_KinSetUserBit	33...192
adiUserDInt_q[1..4]	ML_KinSetUserDInt	1...4
	ML_KinSetUserDInt	5...50
arUserReals_q[1..4]	ML_KinSetUserReal	1...4
	ML_KinSetUserReal	5...50
arUserPoints_q[1..4]	ML_KinSetUserPoint	1...4
	ML_KinSetUserPoint	5...50

Fig.5-91: Extension of the shared memory area by function access

Functions

- [ML_KinGetUserBit](#) on page 215
- [ML_KinGetUserBit_DWORD](#) on page 215
- [ML_KinGetUserDInt](#) on page 216
- [ML_KinGetUserReal](#) on page 217
- [ML_KinSetUserBit](#) on page 218

- [ML_KinSetUserDInt on page 219](#)
- [ML_KinSetUserPoint on page 220](#)
- [ML_KinSetUserReal on page 220](#)

ML_KinGetUserBit

Brief description The function "ML_KinGetUserBit" reads a user bit from the extended KinData interface.



This function can only be used for Robot Control V1.



The status **MUST** be verified, as the return value is not valid in case of error!

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-206: Library assignment

Interface description

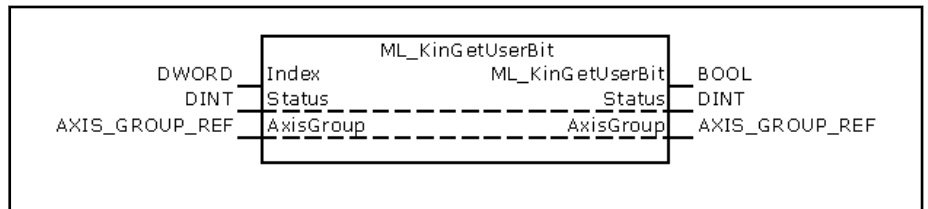


Fig.5-92: FUN ML_KinGetUserBit

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index bit to be read (1...192)
VAR_IN_OUT	Status	DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		BOOL	Value of bit to be read

Tab.5-207: Interface of FUN ML_KinGetUserBit

Implementation example:
bValue := ML_KinGetUserBit (180, diStatus, Kin1);
 The value of the 180th bit of kinematics Kin1 is returned in bValue.
 If the value in diStatus = 0, bValue is valid.

ML_KinGetUserBit_DWORD

Brief description The function "ML_KinGetUserBit_DWORD" reads a user bit array with 32 bits from the extended KinData interface.



This function can only be used for Robot Control V1.



The status **MUST** be verified, as the return value is not valid in case of error!

ML_Robot.library

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-208: Library assignment

Interface description

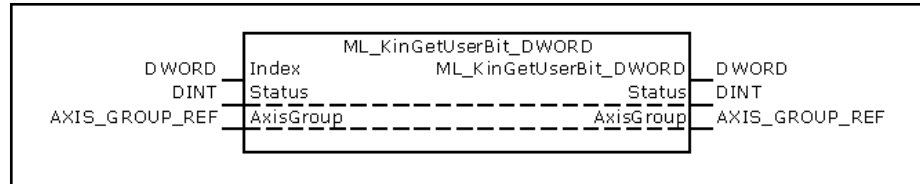


Fig.5-93: FUN ML_KinGetUserBit_DWORD

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of bit array to be read (1...6) 1 - bits 1...32 2 - bits 33...64 3 - bits 65...96 4 - bits 97...128 5 - bits 129...160 6 - Bits 161...192
VAR_IN_OUT	Status	DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DWORD	Value of bit array to be read

Tab.5-209: Interface of FUN ML_KinGetUserBit_DWORD

Implementation example: `dwValue := ML_KinGetUserBit_DWORD ( 2, diStatus, Kin1 );`
The value of the 2nd bit array of kinematics Kin1 is returned in dwValue.
If the value in diStatus = 0, dwValue is valid.

ML_KinGetUserDInt

Brief description The function "ML_KinGetUserDInt" reads a user integer from the extended KinData interface.



This function can only be used for Robot Control V1.



The status MUST be verified, as the return value is not valid in case of error!

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-210: Library assignment

Interface description

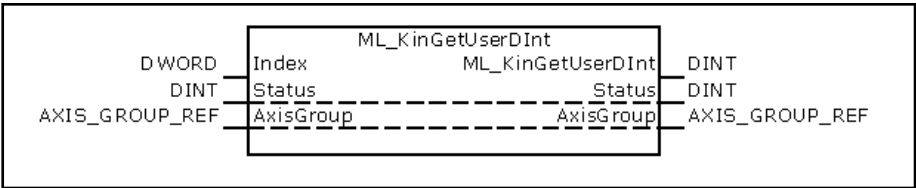


Fig.5-94: FUN ML_KinGetUserDInt

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of integer to be read (1...50)
VAR_IN_OUT	Status	DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DINT	Value of integer to be read

Tab.5-211: Interface of FUN ML_KinGetUserDInt

Implementation example:

diValue := ML_KinGetUserDInt(36, diStatus, Kin2);
The value of the 36th integer of kinematics Kin2 is returned in diValue.
If the value in diStatus = 0, diValue is valid.

ML_KinGetUserReal

Brief description

The function "ML_KinGetUserReal" reads a user real from the extended Kin-Data interface.



This function can only be used for Robot Control V1.



The status MUST be verified, as the return value is not valid in case of error!

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-212: Library assignment

Interface description

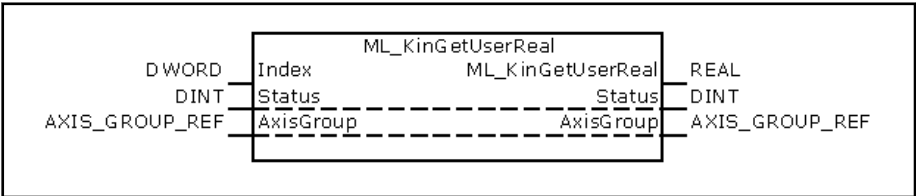


Fig.5-95: FUN ML_KinGetUserReal

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of a real to be read (1...50)
VAR_IN_OUT	Status	DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		REAL	Value of real to be read

Tab.5-213: Interface of FUN ML_KinGetUserReal

Implementation example: `rValue := ML_KinGetUserReal( 42, diStatus, Kin1 );`
The value of the 42nd real value of kinematics Kin1 is returned in rValue.
If the value in diStatus = 0, rValue is valid.

ML_KinSetUserBit

Brief description The function "ML_KinSetUserBit" sets a user bit from the extended KinData interface.



This function can only be used for Robot Control V1.



It is ESSENTIAL that the function value is verified! Otherwise, the value is not set in case of error.

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-214: Library assignment

Interface description

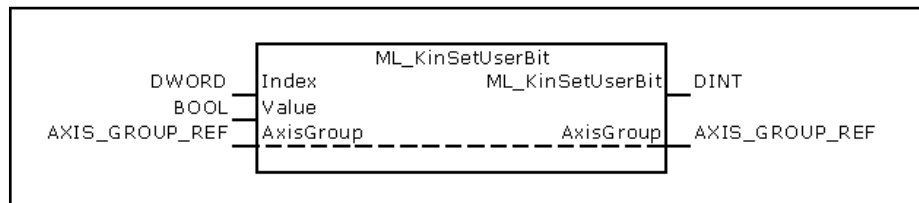


Fig.5-96: FUN ML_KinSetUserBit

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of bit to be read (1...192)
	Value	BOOL	Value of bit to be written

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.

Tab.5-215: Interface of FUN ML_KinSetUserBit

Implementation example: `diStatus := ML_KinSetUserBit ( 102, TRUE, Kin2 );`
The transmitted value is TRUE and written to the 102nd bit of kinematics Kin2.
If the value is in diStatus = 0, the value is set successfully.

ML_KinSetUserDInt

Brief description The function "ML_KinSetUserDInt" sets a user integer from the extended Kin-Data interface.



This function can only be used for Robot Control V1.



It is ESSENTIAL that the function value is verified! Otherwise, the value is not set in case of error.

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-216: Library assignment

Interface description

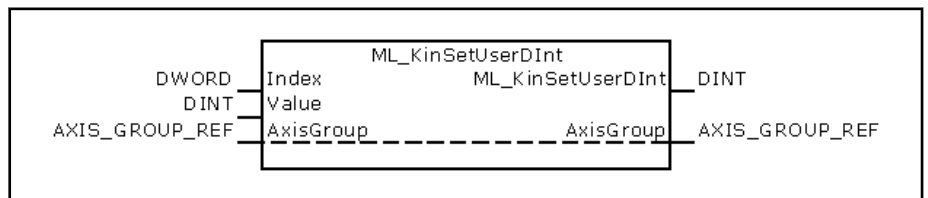


Fig.5-97: FUN ML_KinSetUserDInt

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of integer to be read (1...50)
	Value	DINT	Value of integer to be written
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.

Tab.5-217: Interface of FUN ML_KinSetUserDInt

ML_Robot.library

Implementation example: `diStatus := ML_KinSetUserDInt ( 47, 1357, Kin1 );`
 The transmitted value 1357 is written to the 47th integer of kinematics Kin1.
 If the value is in `diStatus = 0`, the value is set successfully.

ML_KinSetUserPoint

Brief description The function "ML_KinSetUserPoint" sets a user point from the extended Kin-Data interface.



This function can only be used for Robot Control V1.



It is ESSENTIAL that the function value is verified! Otherwise, the value is not set in case of error.

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-218: Library assignment

Interface description

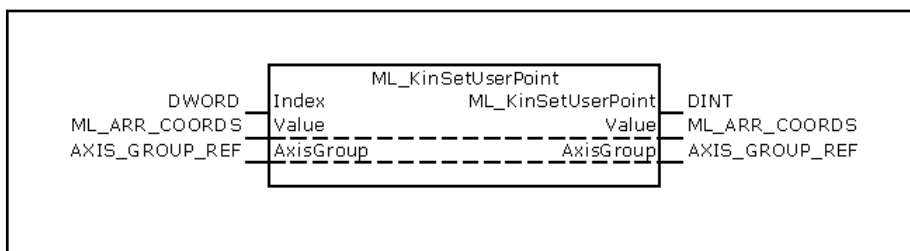


Fig.5-98: FUN ML_KinSetUserPoint

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of point to be written (1...50)
VAR_IN_OUT	Value	ML_ARR_COORDS	Value of point to be written
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range Or: Real value of ML_ARR_COORDS in value does not have a valid floating point format (error "Impermissible decimal number") -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.

Tab.5-219: Interface of FUN ML_KinSetUserPoint

Implementation example: `diStatus := ML_KinSetUserPoint ( 48, point, Kin1 );`
 The value transmitted in point is written to the 48th point of kinematics Kin1.
 If the value is in `diStatus = 0`, the value is set successfully.

ML_KinSetUserReal

Brief description The function "ML_KinSetUserReal" sets a user real from the extended KinData interface.



This function can only be used for Robot Control V1.



It is ESSENTIAL that the function value is verified! Otherwise, the value is not set in case of error.

Library	Range
ML_Robot	Kinematics/RCL user interface

Tab.5-220: Library assignment

Interface description

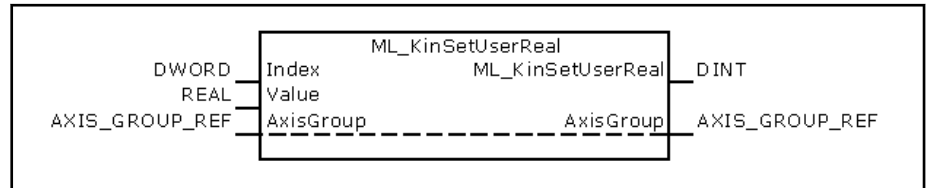


Fig.5-99: FUN ML_KinSetUserReal

	Name	Type	Comment
VAR_INPUT	Index	DWORD	Index of real to be read (1...50)
	Value	REAL	Value of real to be written
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the kinematics
Function value		DINT	It indicates whether the function was executed successfully: 0 - Valid -1 - Error: Index is outside the valid range Or: Real value in value does not have a valid floating point format (error "Impermissible decimal number") -2 - Error: Value in AxisGroup is not a valid kinematics >0 - Internal error This value MUST be verified.

Tab.5-221: Interface of FUN ML_KinSetUserReal

Implementation example:

```
diStatus := KinSetUserReal ( 4, 22.5, Kin2 );
```

The transmitted value 22.5 is written to the 4th real of kinematics Kin2.

If the value is in diStatus = 0, the value is set successfully.

5.3.9 TeachIn Function Blocks

ML_GroupDeletePointFile

Brief description

The function block "ML_GroupDeletePointFile" deletes a point file of a kinematics.

(Refer to [Teach-In](#), page 36).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-222: Library assignment

ML_Robot.library

Interface description

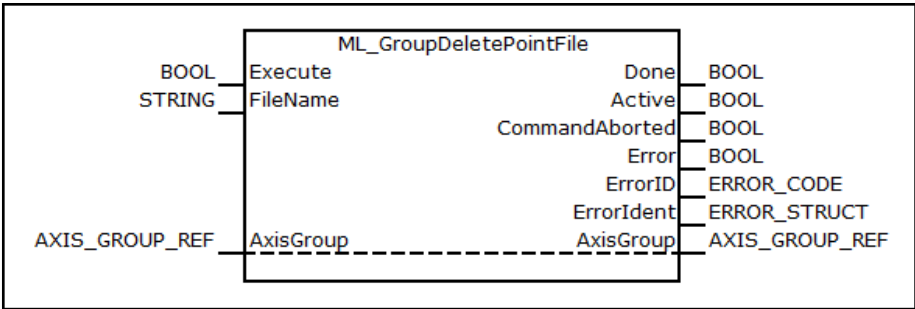


Fig.5-100: ML_GroupDeletePointFile function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	FileName	String	Name of the file to be saved
VAR_OUTPUT	Done	BOOL	Deletion of the file completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-223: ML_GroupDeletePointFile function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	String			"	Rising edge at "Execute"

Tab.5-224: Input behavior of ML_GroupDeletePointFile

Signal-time diagram Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing on page 120](#).

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_GroupDeletePoints

Brief description The function block "ML_GroupDeletePoints" deletes teach-in points of a kinematics.
(Refer to [Teach-In, page 36](#)).

This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-225: Library assignment

Interface description

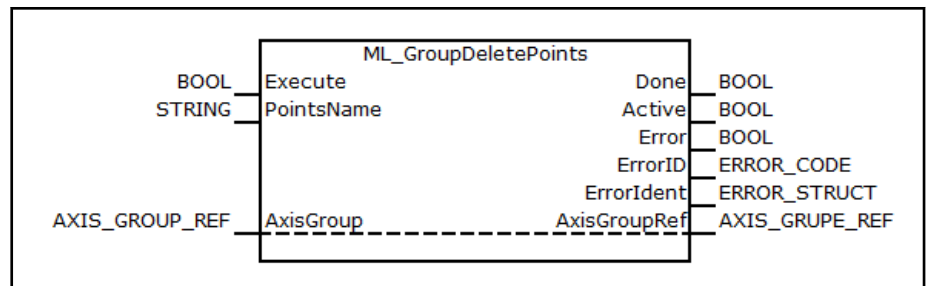


Fig.5-101: Function block ML_GroupDeletePoints

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	PointsName	STRING(19)	Point(s) that are to be deleted Name may contain the wildcard ""
VAR_OUTPUT	Done	BOOL	Deletion completed
	Active	BOOL	Deletion in progress
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-226: Function block ML_GroupDeletePoints

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
PointsName	STRING(19)			"	Rising edge of "Execute"

Tab.5-227: Input behavior of ML_GroupDeletePoints

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030 , page 351](#) - error when accessing the axis/the control.

ML_GroupLoadPointFile

Brief description

The function block "ML_GroupLoadPointFile" loads a point file of a kinematics.

(Refer to [Teach-In, page 36](#)).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-228: Library assignment

ML_Robot.library

Interface description

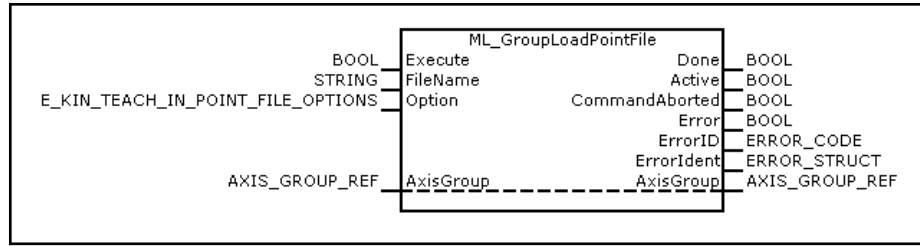


Fig.5-102: Function block ML_GroupLoadPointFile

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	FileName	String	Name of the file to be loaded
	Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS	Options for loading the file See E_KIN_TEACH_IN_POINT_FILE_OPTIONS , page 242
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-229: Function block ML_GroupLoadPointFile

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	String			"	Rising edge at "Execute"
Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS			E_GROUP_TEACH_IN_NEW	Rising edge at "Execute"

Tab.5-230: Input behavior of ML_GroupLoadPointFile

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_GroupReadNextPoint

Brief description

The function block "ML_GroupReadNextPoint" searches for the next teach-in point and returns the data of the next kinematics point.
(Refer to [Teach-In](#), page 36).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-231: Library assignment

Interface description

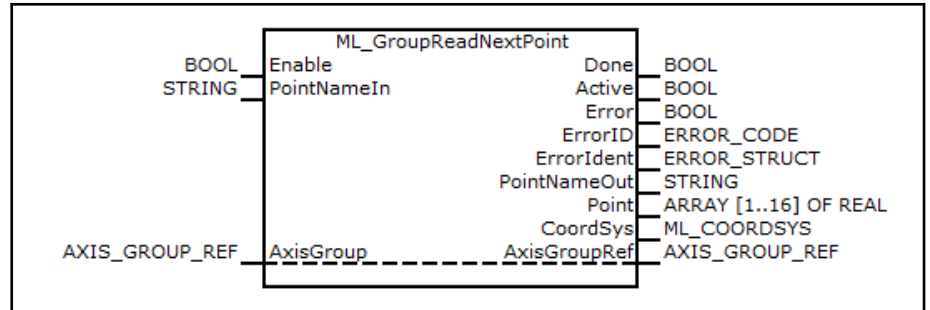


Fig.5-103: Function block ML_GroupReadNextPoint

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	PointNameIn	STRING(19)	Name of the point
VAR_OUTPUT	Done	BOOL	Reading the point completed
	Active	BOOL	Reading the point in progress
	PointNameOut	STRING(19)	Name of the next point
	CoordSys	ML_COORDSYS	Coordinate system of the next point (Refer to ML_COORDSYS , page 244).
	Point	ARRAY [1..16] OF REAL	Coordinates of the next point
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-232: Function block ML_GroupReadNextPoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PointNameIn	STRING(19)			"	Continuous

Tab.5-233: Input behavior of ML_GroupReadNextPoint

Signal-time diagram

Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing on page. 123](#)

Error handling

The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_GroupReadPoint

Brief description

The function block "ML_GroupReadPoint" returns the teach-in data of a kinematics point.
(Refer to [Teach-In](#), page 36).

ML_Robot.library

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-234: Library assignment

Interface description

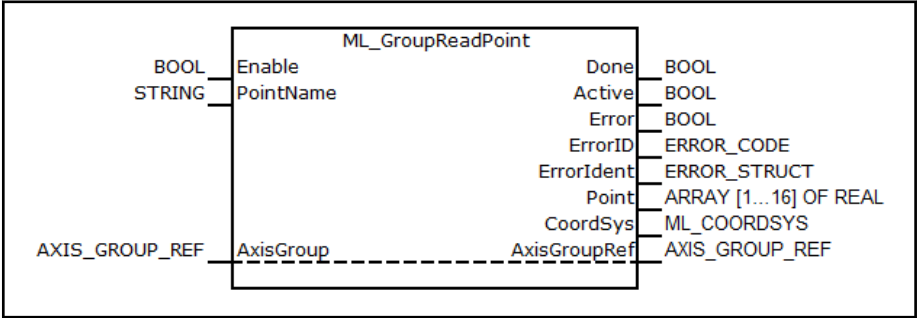


Fig.5-104: Function block ML_GroupReadPoint

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	PointName	STRING(19)	Name of the point
VAR_OUTPUT	Done	BOOL	Reading the point completed
	Active	BOOL	Reading the point in progress
	CoordSys	ML_COORDSYS	Coordinate system of a point (See ML_COORDSYS on page 244)
	Point	ARRAY [1..16] OF REAL	Coordinates of the point
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-235: Function block ML_GroupReadPoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PointName	STRING(19)			"	Continuous

Tab.5-236: Input behavior of ML_GroupReadPoint

- Signal-time diagram
- Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing](#) on page. 123
- Error handling
- The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 351](#) - error when accessing the axis/the control.

ML_GroupSavePointFile

Brief description The function block "ML_GroupSavePointFile" saves a point file of a kinematics.

(Refer to [Teach-In](#) on page 36).



This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-237: Library assignment

Interface description

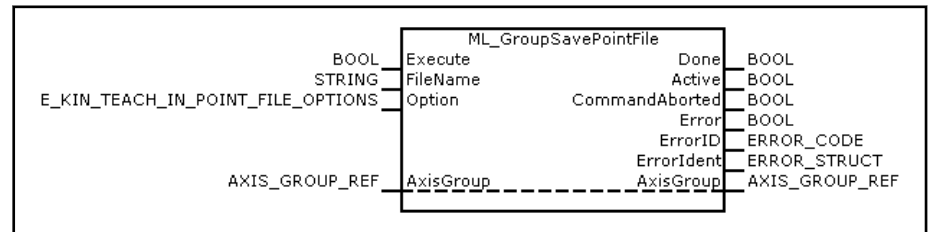


Fig.5-105: Function block ML_GroupSavePointFile

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	FileName	String	Name of the file to be saved
	Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS	Options for saving the file See E_KIN_TEACH_IN_POINT_FILE_OPTIONS , page 242
VAR_OUTPUT	Done	BOOL	File was saved successfully
	Active	BOOL	Function block is active
	CommandAborted	BOOL	Command aborted by another function block.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-238: Function block ML_GroupSavePointFile

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	String			"	Rising edge at "Execute"
Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS			E_GROUP_TEACH_IN_NEW	Rising edge at "Execute"

Tab.5-239: Input behavior of ML_GroupSavePointFile

Signal-time diagram

Refer to [Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing](#) on page 120.

Error handling

The function block generates the following types of error messages:
[MLC_TABLE](#), [ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

ML_Robot.library

ML_GroupWritePoint

Brief description The function block "ML_GroupWritePoint" teaches a point of a kinematics.
(Refer to [Teach-In](#), page 36).

 This function block can only be used for Robot Control V2.

Library	Range
ML_Robot	Kinematics/TeachIn

Tab.5-240: Library assignment

Interface description

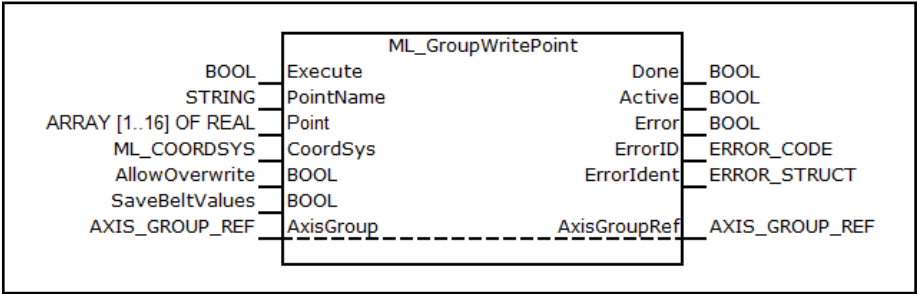


Fig.5-106: Function block ML_GroupWritePoint

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	PointName	STRING(19)	Name of the point
	Point	ARRAY [1..16] OF REAL	Coordinates of the point
	CoordSys	ML_COORDSYS	Coordinate system of a point (Refer to ML_COORDSYS , page 244).
	AllowOverwrite	BOOL	Permits overwriting an existing point
	SaveBeltValues	BOOL	Saves the belt values of the point
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Definition in progress
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab.5-241: Function block ML_GroupWritePoint

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
PointName	STRING(19)			"	Rising edge of "Execute"
Point	ARRAY [1..16] OF REAL			16(0.0)	Rising edge of "Execute"

Name	Type	Min. value	Max. value	Default value	Effective
CoordSys	ML_COORDSYS			CS_MACHINE_COOR- DINATES	Rising edge of "Execute" Continuous
AllowOverwrite	BOOL			FALSE	Rising edge of "Execute"
SaveBeltValues	BOOL			FALSE	Rising edge of "Execute"

Tab.5-242: Input behavior of ML_GroupWritePoint

Signal-time diagram Refer to [Inputs and Outputs of State-controlled Function Blocks, Permanent Processing](#) on page. 123

Error handling The function block generates the following types of error messages:
[MLC_TABLE, ERROR_TABLE 16#0030](#) , page 351 - error when accessing the axis/the control.

5.4 Data Types

5.4.1 Data Types, Overview

The following data types support the function blocks of the "ML_Robot" library:

Configuration *Data types supporting the function blocks "Folder Configuration"*

- [ML_BELT_ERROR_REACTION_MODE](#) on page 230
- [ML_BELT_KIND](#) on page 230
- [ML_KIN_AXES_CONFIG_DATA](#) on page 230
- [ML_KIN_AXIS_CONFIG](#) on page 231
- [ML_KIN_SAFE_ZONE_TYPE](#) on page 231
- [ML_KIN_USED_AXES](#) on page 231
- [MC_KIN_REF](#) on page 232
- [MC_KIN_REF_EXT](#) on page 234

Info *Data types supporting the function blocks "Folder Info"*

- [E_KIN_MECH_CMD](#) on page 237
- [ML_KINEMATICSDATA_SM](#) on page 237
- [ML_KIN_CS_TYPE](#) on page 239
- [ML_KIN_MECHANIC_DATA](#) on page 240
- [ML_KIN_PERF_DATA](#) on page 240
- [ML_POINTKIND](#) on page 240

Jog *Data types supporting the function blocks "Folder Jog"*

- [ML_COORD_BITS](#) on page 241
- [ML_KIN_E_COORD_SYS](#) on page 241
- [ML_KIN_E_JOG_VEL](#) on page 241
- [ML_KIN_JOG](#) on page 242

Movement *Data types supporting the function blocks "Folder Movement"*

- [ML_ARR_COORDS](#) on page 243
- [ML_ARR_PLACEMENT](#) on page 244

ML_Robot.library

- [ML_CIRCMODES](#) on page 244
- [ML_COORDSYS](#) on page 244
- [ML_CYCLIC_MODE](#) on page 245
- [ML_SET_MODE](#) on page 246
- [ML_SLOPE_TYPE](#) on page 246
- RCL *Data types supporting the function blocks "Folder RCL"*
 - [ML_POINT_TYPE](#) on page 246
 - [ML_ROCO_FILE_DIR](#) on page 247
 - [ML_ROCO_NAME](#) on page 247
 - [ML_ROCO_POINT](#) on page 248
 - [ML_ROCO_POINT_NAME](#) on page 248
- TeachIn *Data types supporting the function blocks "Folder TeachIn"*
 - [E_KIN_TEACH_IN_POINT_FILE_OPTIONS](#) on page 242

5.4.2 Config Data Types

ML_BELT_ERROR_REACTION_MODE

Brief description The enumeration type "ML_BELT_ERROR_REACTION_MODE" defines the error reaction in case of belt synchronization:
The enumeration type is used by the following function block:
[ML_SetBeltErrorReaction](#) on page 142

Element	Description
ML_BELT_ERROR_NO_SPECIFIC_REACTION	No error reaction, the robot continues to move in synchronization with the belt
ML_BELT_ERROR_DESYNC_OVER_TIME	Desynchronization using time
ML_BELT_ERROR_DESYNC_WITH_LIMITS	Desynchronization using limit values

Tab.5-243: Elements of the ML_BELT_ERROR_REACTION_MODE enumeration type

ML_BELT_KIND

Type definitions The "ML_BELT_KIND" enumeration type defines the belt type of a belt.
The following function block uses this enumeration type to configure the belt drive:
[ML_SetBeltParameter](#) on page 143

Element	Description
BK_4	Belt type 4 (with synchronization and desynchronization)

Tab.5-244: Definition of the elements of the enumeration type ML_BELT_KIND

For details, refer to [Belt Configuration](#) on page 50

ML_KIN_AXES_CONFIG_DATA

The axis configuration of a kinematics is stored in the structure "ML_KIN_AXES_CONFIG_DATA".

The following interfaces use this structure:

- [MC_KIN_REF](#) on page 232
- [MC_KIN_REF_EXT](#) on page 234

Structure element	Type	Description
Configuration	ML_KIN_AXIS_CONFIG	Axis configuration Refer to ML_KIN_AXIS_CONFIG , page 231
ControlNumber	WORD	Control number of the axis
AxisNumber	WORD	Axis number of the axis

Tab.5-245: *ML_KIN_AXES_CONFIG_DATA data structure*

ML_KIN_AXIS_CONFIG

Brief description The following structure uses enumeration type "ML_KIN_AXIS_CONFIG" for the configuration of an axis in a kinematics.

[ML_KIN_AXES_CONFIG_DATA on page 230](#)

Element	Description
ML_KIN_AXIS_TYPE_NONE	Axis is not used in the kinematics
ML_KIN_AXIS_TYPE_POSITION	Axis is used as positioning axis of the kinematics
ML_KIN_AXIS_TYPE_ORIENTATION	Axis is used as orientation axis of the kinematics
ML_KIN_AXIS_TYPE_BELT	Axis is used as belt axis of the kinematics

Tab.5-246: *Definition of the elements of the enumeration type ML_KIN_AXIS_CONFIG*

ML_KIN_SAFE_ZONE_TYPE

Type definitions The enumeration type "ML_KIN_SAFE_ZONE_TYPE" is used by function block [ML_GroupSetSafeZone on page 137](#).

It defines the configuration of a safe zone:

Element	Description
ML_KIN_SAFE_ZONE_DEACTIVATE	Deactivated
ML_KIN_SAFE_ZONE_INSIDE_OK	Software limit switch
ML_KIN_SAFE_ZONE_OUTSIDE_OK	Safe zone

Tab.5-247: *Definition of the elements of the enumeration type ML_KIN_SAFE_ZONE_TYPE*

ML_KIN_USED_AXES

Type definitions The "ML_KIN_USED_AXES" enumeration type defines the degrees of freedom of a kinematics.

Use this enumeration type to define a kinematics:

[Extended User-defined Transformation on page 100](#)

Element	Description
X_AXIS	Degree of freedom on the X-axis
Y_AXIS	Degree of freedom on the Y-axis
Z_AXIS	Degree of freedom on the Z-axis
A_AXIS	Freedom of rotation around the X-axis

ML_Robot.library

Element	Description
B_AXIS	Freedom of rotation around the Y-axis
C_AXIS	Freedom of rotation around the Z-axis

Tab.5-248: Definition of the elements of the enumeration type
ML_KIN_USED_AXES

MC_KIN_REF

MC_KIN_REF, Overview

The MC_KIN_REF interface defines transformation methods.

A function block has to implement this interface in order to register a transformation using the function block

[MC_SetKinTransform on page 131](#)

Method	Return value	Arguments	Description
Initialization	INT	AxesConfigData TransformationLengths TransformationFactors	Initialization functions of a transformation
Forward	INT	OriginalCoordinates JointCoordinates	Forward position kinematics
Inverse	INT	OriginalCoordinates JointCoordinates	Inverse position kinematics

Tab.5-249: MC_KIN_REF interface

MC_KIN_REF, Initialization

This method is called by the system if a transformation is to be initialized. Initialization takes place during phase switching from P2 to BB.

Precalculations and checks on the information can be carried out in this function.

Argument	Type	Description
AxesConfigData	REFERENCE TO ARRAY [1..16] OF ML_KIN_AXES_CONFIG_DATA	K-0-0008, Kinematic configuration ML_KIN_AXES_CONFIG_DATA on page 230
TransformationLengths	REFERENCE TO ARRAY [1..16] OF REAL	K-0-0040, Transformation, lengths of the kinematics
TransformationFactors	REFERENCE TO ARRAY [1..16] OF REAL	K-0-0041, Transformation, factors of the kinematics

Tab.5-250: Arguments of the interface function "Initialization"

If the initialization cannot be completed successfully, the function has to return a value not equal to 0.

Possible floating point errors (such as division by zero) are intercepted by the system and output as errors (see [Error Handling with Interface Functions on page 233](#)).

MC_KIN_REF, Forward

The system calls this method if conversions are to be made from the axis coordinate system (ACS) to the base coordinate system (BCS). This method is called cyclically in the operating mode.

Argument	Type	Description
OriginalCoordinates	REFERENCE TO ARRAY [1..16] OF REAL	Base coordinate system
JointCoordinates	REFERENCE TO ARRAY [1..16] OF REAL	Axis coordinate system

Tab.5-251: Arguments of the interface function "Forward"

If a point cannot be transformed, the function has to return a value not equal to 0.

Possible floating point errors (such as division by zero) are intercepted by the system and output as errors (see [Error Handling with Interface Functions on page 233](#)).

MC_KIN_REF, Inverse

The system calls this method if conversions are to be made from the base coordinate system (BCS) to the axis coordinate system (ACS). This method is called cyclically in the operating mode.

Argument	Type	Description
OriginalCoordinates	REFERENCE TO ARRAY [1..16] OF REAL	Base coordinate system
JointCoordinates	REFERENCE TO ARRAY [1..16] OF REAL	Axis coordinate system

Tab.5-252: Arguments of the interface function "Inverse"

If a point cannot be transformed, the function has to return a value not equal to 0.

Possible floating point errors (such as division by zero) are intercepted by the system and output as errors (see [Error Handling with Interface Functions on page 233](#)).

Error Handling with Interface Functions

If a floating point error occurs with one of the interface functions (methods), one of the following errors is reported.

However, such errors already have to be considered in the transformation mathematics: For example, always check if a divisor is zero. If yes, report an error and do not execute the division.

- F12C0600, Invalid operation at PLC function call
- F12C0601, Denormalized operand at PLC function call
- F12C0602, Zero division at PLC function call
- F12C0603, Overflow at PLC function call
- F12C0604, Underflow at PLC function call
- F12C0605, Precision error at PLC function call

If an interface function (method) is to be called that was not registered, the following error is reported:

- F12C0498, Transformation in PLC has not been registered

ML_Robot.library

If an interface function cannot be executed successfully (return value of the function is not zero), the following error is reported:

- F12C0499, User defined transformation error in PLC

MC_KIN_REF_EXT

MC_KIN_REF_EXT, Overview

The MC_KIN_REF_EXT interface defines methods of an extended transformation.

A function block has to implement this interface in order to register an extended transformation using function block

[MC_SetKinTransformExt on page 132](#)

Method	Return value	Arguments	Description
GetUsedAxes	UINT		Degrees of freedom of a transformation
DetermineMemory	UINT	ACS TrafoChoise	Method to resolve ambiguities
Initialization	UDINT	AxesConfigData TransformationLengths TransformationFactors	Initialization functions of a transformation
Forward	UDINT	BCS ACS	Forward position kinematics
Inverse	UDINT	BCS ACS	Inverse position kinematics
GetMechanic	UDINT	ACS MechanicData MechanicDataSize MechanicDataRetSize UpdateOnly	Method for mechanical information of a transformation

Tab.5-253: MC_KIN_REF interface

MC_KIN_REF_EXT, GetUsedAxes

The system calls this method to query the possible degrees of freedom of the transformation. If the transformation supports one of the degrees of freedom, it has to return ML_KIN_USED_AXES type values that are linked by the OR operator.

(see [ML_KIN_USED_AXES on page 231](#))

MC_KIN_REF_EXT, DetermineMemory

The system calls this method to enable the transformation to resolve ambiguities in the next inverse transformation.

Argument	Type	Description
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the axis coordinate system
TrafoChoise	BOOL	In the next inverse transformation, the transformation independently selects the ambiguous solution to be used

Tab.5-254: Arguments of the interface function *DetermineMemory*

If the method cannot be completed successfully, an MLC error code has to be returned:

(see [Error Handling with Interface Functions, page 236](#)).

MC_KIN_REF_EXT, Initialization

This method is called by the system if a transformation is to be initialized. Initialization takes place during phase switching from P2 to BB.

Precalculations and checks on the information can be carried out in this function.

Argument	Type	Description
AxesConfigData	REFERENCE TO ARRAY [1..16] OF ML_KIN_AXES_CONFIG_DATA	K-0-0008, Kinematic configuration ML_KIN_AXES_CONFIG_DATA on page 230
TransformationLengths	REFERENCE TO ARRAY [1..16] OF LREAL	K-0-0040, Transformation, lengths of the kinematics
TransformationFactors	REFERENCE TO ARRAY [1..16] OF LREAL	K-0-0041, Transformation, factors of the kinematics

Tab.5-255: Arguments of the interface function *"Initialization"*

If the initialization cannot be completed successfully, an MLC error code has to be returned:

(see [Error Handling with Interface Functions, page 236](#)).

MC_KIN_REF_EXT, Forward

The system calls this method if conversions are to be made from the axis coordinate system (ACS) to the base coordinate system (BCS). This method is called cyclically in the operating mode.

Argument	Type	Description
BCS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the base coordinate system
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the axis coordinate system

Tab.5-256: Arguments of the interface function *"Forward"*

If a point cannot be transformed, an MLC error code has to be returned:

(see [Error Handling with Interface Functions, page 236](#)).

MC_KIN_REF_EXT, Inverse

The system calls this method if conversions are to be made from the base coordinate system (BCS) to the axis coordinate system (ACS). This method is called cyclically in the operating mode.

ML_Robot.library

Argument	Type	Description
OriginalCoordinates	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the base coordinate system
JointCoordinates	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the axis coordinate system

Tab.5-257: Arguments of the interface function "Inverse"

If a point cannot be transformed, an MLC error code has to be returned:
(see [Error Handling with Interface Functions, page 236](#)).

MC_KIN_REF_EXT, GetMechanic

The system calls this method to request the mechanical information of a point in the axis coordinate system (ACS). The auxiliary function [ML_AddMechanicData on page 135](#) enters the mechanical information.

Argument	Type	Description
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in the axis coordinate system
MechanicData	REFERENCE TO ML_KIN_MECHANIC_DATA	ML_KIN_MECHANIC_DATA on page 240 , Mechanical data to be entered
MechanicDataSize	UDINT	Number of elements in MechanicData
MechanicDataRetSize	REFERENCE TO UDINT	Number of entered elements in MechanicData
UpdateOnly	BOOL	Return only modifiable entries

Tab.5-258: Arguments of the interface function "Inverse"

If the method cannot be completed successfully, an MLC error code has to be returned:

(see [Error Handling with Interface Functions, page 236](#)).

Error Handling with Interface Functions

If a floating point error occurs at one of the methods, the error is reported automatically.

However, such errors already have to be considered in the transformation mathematics: For example, always check if a divisor is zero. If yes, report an error and do not execute the division.

- F12C0600: Invalid operation at PLC function call
- F12C0601: Denormalized operand at PLC function call
- F12C0602: Zero division at PLC function call
- F12C0603: Overflow at PLC function call
- F12C0604: Underflow at PLC function call
- F12C0605: Precision error at PLC function call

In contrast to the simple user-defined transformation, the extended user-defined transformation can return an MLC error code as return value of the functions. This facilitates diagnostics. An MLC error code has the following structure: 16#Fxxxxxxx. If an interface function was executed without error, a code with the structure 16#0xxxxxxx is returned.

The most important error codes for a transformation:

- F22B0021: Axis angles out of definition range
- F12C0479: WC zero position is singular

- F12C047A: Too close to stand
- F22B0022: Point can not reached by transformation
- F12C008A: Transformation parameter does not fit to robot type
- F22B0023: In saved range of transformation
- F22B0024: Axis has wrong scaling for transformation
- F22B0025: Too less axes for Transformation

5.4.3 Info Data Types

E_KIN_MECH_CMD

Brief description The enumeration type "E_KIN_MECH_CMD" defines how a mechanical part of a kinematics is to be generated and/or modified.

The enumeration type is used by the following structure:

[ML_KIN_MECHANIC_DATA on page 240](#)

Element	Description
KIN_MECH_CMD_CREATE_ROD	• Create a rod • The rod points from the zero point in positive X direction • The rod has length 1 and radius 1
KIN_MECH_CMD_CREATE_SPHERE	• Create a sphere • The sphere is at the zero point • The sphere has a radius of 1
KIN_MECH_CMD_POSITION	Change the position of an object to (x,y,z)
KIN_MECH_CMD_DIRECTION	Change the direction of an object to (x,y,z)
KIN_MECH_CMD_SIZE	Change the size of an object to (x,y,z)

Tab.5-259: Elements of enumeration type E_KIN_MECH_CMD

ML_KINEMATICS_DATA_SM

Brief description An instance of the structure "ML_KINEMATICS_DATA_SM" contains information on the corresponding kinematics that can be accessed via **shared memory**.

Library	Range
ML_Robot	Data Types/Info

Tab.5-260: Library assignment

The respective kinematics types support the following structural elements in ML_KINEMATICS_DATA_SM:

Structure element	Description	Access	RobotControl	
			V1	V2
uiCoordSum_i	K-0-0007, Axis number of kinematics	read	x	x
uiUpdateCount_i	Update counter (odd: update starts, even: update completed)	read	x	x
arPositionJointCoord_i	A-0-0100, Actual position value	read	x	x

ML_Robot.library

Structure element	Description	Access	RobotControl	
			V1	V2
arPositionOriginal-Coord_i	K-0-0100, Current position, Cartesian Machine Coordinate System	read	x	x
arPositionWorkpiece-Coord_i	K-0-0105, Actual position, Cartesian Workpiece Coordinates	read	x	x
strToolName_i	K-0-0140, Tool name	read	x	
arToolCoord_i	K-0-0106, Coordinate values of tool	read	x	
strWorkpieceName_i	K-0-0141, Workpiece name	read	x	
arWorkpieceCoord_i	K-0-0107, Coordinate values of workpiece	read	x	
K-0-0021, Kinematic status				
Kin_OpMode_Auto	K-0-0021.00, automatic	read	x	
Kin_OpMode_Manual	K-0-0021.01, manual	read	x	
Kin_JogCoord1	K-0-0021.03, bit 1 of jogging coordinate system	read	x	
Kin_JogCoord2	K-0-0021.04, bit 2 of jogging coordinate system	read	x	
Kin_StandStill	K-0-0021.05, all axes of the kinematics are stopped	read	x	x
Kin_InPosition	K-0-0021.06, the set end position is reached	read	x	x
Kin_BeltSynchronous	K-0-0021.07, belt-synchronous	read	x	x
Kin_Warning	K-0-0021.08, warning	read	x	x
Kin_Error	K-0-0021.09, error	read	x	x
Kin_NotHomed	K-0-0021.10, homing required; not all axes are referenced	read	x	x
Kin_Blending	K-0-0021.11, blending enabled	read	x	x
Kin_TrafoValuesInvalid	K-0-0021.12, transformed machine coordinates (MCS) are invalid	read	x	
Kin_JogEnabled	K-0-0021.13, jogging enabled	read	x	
Kin_DrivesOn	K-0-0021.14, all axes are operational (AH or AF)	read	x	x
Kin_FeedEnabled	K-0-0021.15, feed enabled, traversing motions enabled	read	x	
Kin_BlockEnabled	K-0-0021.16, block processing enabled, processing of the next block enabled	read	x	
Kin_ErrorReaction	K-0-0021.17, error reaction enabled	read	x	x
Kin_JogMode1	K-0-0021.19, jog modes, bit 1	read	x	
Kin_JogMode2	K-0-0021.20, jog modes, bit 2	read	x	
Kin_ProgSelected	K-0-0021.21, program selected	read	x	
Kin_ResetActive	K-0-0021.23, reset is active	read	x	
Kin_ProgEnded	K-0-0021.24, program terminated	read	x	
Kin_ProgStarted	K-0-0021.25, program started	read	x	
Kin_ProgPaused	K-0-0021.26, program: PAUSE command enabled	read	x	

Structure element	Description	Access	RobotControl	
			V1	V2
Kis_SplineCalcActive	K-0-0021.28, calculating spline for blending motion	read	x	x
Kis_SplineValid	K-0-0021.29, spline is valid	read	x	x
K-0-0022 Extended kinematic status				
Kis_Errorstop	K-0-0022.00, kinematics is in "Errorstop"	read	x	x
Kis_Stopping	K-0-0022.01, kinematics is in "Stopping"	read	x	x
Kis_Incomplete	K-0-0022.03, not all axes are assigned to the kinematics (incomplete)	read	x	x
Kis_ProgrammedMotion	K-0-0022.05, kinematics is in "Programmed Motion"	read	x	
Kis_StandStill	K-0-0022.07, kinematics is in "StandStill"	read	x	x
Kis_ProgramSelected	K-0-0022.08, program is selected	read	x	x
Kis_Running	K-0-0022.09, kinematics is running	read	x	
Kis_ManualMotion	K-0-0022.10, kinematics is in jog mode KinData[].stJogCom_q.enJogCmdVel ≠ KIN_JOG_STOP	read	x	
Kis_ProgramPaused	K-0-0022.11, kinematics is in "Pause"	read	x	
Kis_GroupMoving	K-0-0022.12, kinematics is in "GroupMoving"	read	x	x
arProgrEndPos_i	K-0-2200, Programmed position 1 (active); K-0-2201, Programmed position 2 (active)	read	x	
wProgrEndPosType_i	K-0-2202, Type of actively programmed position	read	x	
wlpoType_i	K-0-2203, Interpolation type	read	x	
strQLLFileName_i	K-0-2204, Currently active QLL file name	read	x	
dwQLLLnNumber_i	K-0-2205, Currently active line number	read	x	
dwUserBits_i	User data: PLC input, RCL output	read	x	
adiUserDInt_i	User data: PLC input, RCL output	read	x	
arUserReals_i	User data: PLC input, RCL output	read	x	
stJogCmd_q	Jog command (see ML_KIN_JOG on page 242)	write	x	
dwUserBits_q	User data: PLC output, RCL input	write	x	
adiUserDInt_q	User data: PLC output, RCL input	write	x	
arUserReals_q	User data: PLC output, RCL input	write	x	
arUserPoints_q	User data: PLC output, RCL input	write	x	

x Structural elements supported by robot type
Tab.5-261: Structural elements in ML_KINEMATICS_DATA_SM
See also:

- [KinData\[\] - Kinematic Data in the Shared Memory on page 109](#)

ML_KIN_CS_TYPE

Brief description The enumeration type "ML_KIN_CS_TYPE" defines the transformation type.

ML_Robot.library

Element	Description
CS_TYPE_CARTESIAN	Cartesian transformation
CS_TYPE_CYLINDRIC	Cylindrical transformation

Tab.5-262: Elements of enumeration type ML_KIN_CS_TYPE

ML_KIN_MECHANIC_DATA

Brief description Structure "ML_KIN_MECHANIC_DATA" defines the mechanical structure of a robot.

This structure is used by:

- Function block [ML_RobotGetMechanic](#) on page 153
- Function block [ML_AddMechanicData](#) on page 135

Structure element	Type	Description
cmd	E_KIN_MECH_CMD	Command to a mechanical partial object Refer to E_KIN_MECH_CMD , page 237
object	STRING[7]	Identifier of the mechanical partial object
x	REAL	Data for the command to a mechanical partial object
y	REAL	Data for the command to a mechanical partial object
z	REAL	Data for the command to a mechanical partial object

Tab.5-263: Data structure ML_KIN_MECHANIC_DATA

ML_KIN_PERF_DATA

Brief description The structure "ML_KIN_PERF_DATA" defines the data of the performance measurement.

The following function block uses this structure:

[ML_RobotGetPerformance](#) on page 155

Structure element	Type	Description
name	STRING(16)	Name of the performance measurement
count	UDINT	Number of measurements
fMin	REAL	Minimum duration in μ s
fMax	REAL	Maximum duration in μ s
fAvg	REAL	Average duration in μ s

Tab.5-264: Data structure ML_KIN_PERF_DATA

ML_POINTKIND

Brief description The enumeration type "ML_POINTKIND" defines the point type regarding the chronological sequence.

Element	Description
POINTKIND_PREPARE	Point refers to block preparation
POINTKIND_EXECUTE	Point refers to block processing

Tab.5-265: Definition of the elements of the enumeration type ML_POINTKIND

5.4.4 Jog Data Types

ML_COORD_BITS

Short description The structure contains a bit for every coordinate. That structure is used to activate the jog mode for this coordinate in one direction.

Library	Range
ML_Robot	Data types/ Robot types

Tab.5-266: Library assignment

Type definition

```

TYPE ML_COORD_BITS :
STRUCT
  Coord_01 : BOOL; // 01. coordinate or axis of the robot
  Coord_02 : BOOL; // 02. coordinate or axis of the robot
  // ....
  Coord_16 : BOOL; // 16. coordinate or axis of the robot
END_STRUCT
END_TYPE

```

ML_KIN_E_COORD_SYS

Brief description The enumeration type contains the jog coordinate systems that can be selected.

Library	Range
ML_Robot	Data Types/Robot Types

Tab.5-267: Library assignment

Program:

```

TYPE ML_KIN_E_COORD_SYS : ( KIN_CORD_SYS_JOINT_COORD := 0, (*joint coordinates*)
  KIN_CORD_SYS_ORIGIN_COORD := 1, (*world coordinates*)
  KIN_CORD_SYS_WORKPIECE_COORD := 2, (*workpiece coordinates*)
  KIN_CORD_SYS_GRIPPER_COORD := 3 (*gripper coordinates*)
);
END_TYPE

```

ML_KIN_E_JOG_VEL

Short description The enumeration type contains the jog velocities that can be selected.

Library	Range
ML_Robot	Data types/ Robot types

Tab.5-268: Library assignment

Program:

```

TYPE ML_KIN_E_JOG_VEL :
(
  KIN_JOG_STOP := 0, (*Stop*)
  KIN_JOG_STEP_SMALL := 1, (*jogging, small steps*)
)

```

ML_Robot.library

```

KIN_JOG_STEP_LARGE := 2, (*jogging, large steps*)
KIN_JOG_CONT_SLOW  := 4, (*jogging, slow continuous motion*)
KIN_JOG_CONT_FAST  := 8  (*jogging, fast continuous motion*)
);
END_TYPE

```

ML_KIN_JOG

Brief description Using this structure, the command interface for jogging can be read and assigned by the PLC.

See also: [ML_KINEMATICSDATA_SM on page 237](#)

Library	Range
ML_Robot	Data Types/Robot Types

Tab.5-269: Library assignment

Program:

```

TYPE ML_KIN_JOG :
STRUCT
enJogCmdCoordSys:ML_KIN_E_COORD_SYS;(*Command: Selection jog coordinate system*)
enJogCmdVel      :ML_KIN_E_JOG_VEL;  (*Command: jog velocity*)
wJogCmdPositiv   :ML_COORD_BITS;     (*Command: bit string jogging pos. direction*)
wJogCmdNegativ   :ML_COORD_BITS;     (*Command: bit string jogging neg. direction*)
END_STRUCT
END_TYPE

```

Structure element	Type	Default value	Description
enJogCmdCoordSys	ML_KIN_E_COORD_SYS	0	contains the selection of (jogging) coordinate systems
enJogCmdVel	ML_KIN_E_JOG_VEL	0	contains the selection of (jogging) velocities
wJogCmdPositiv	ML_COORD_BITS	0	contains one bit per coordinate with which the coordinate can be activated in the positive direction.
wJogCmdNegativ	ML_COORD_BITS	0	contains one bit per coordinate with which the coordinate can be activated in the negative direction.

Tab.5-270: Interface description of the data structure ML_KIN_JOG

5.4.5 TeachIn Data Types

E_KIN_TEACH_IN_POINT_FILE_OPTIONS

Brief description The enumeration type "E_KIN_TEACH_IN_POINT_FILE_OPTIONS" contains the available options for loading and saving point files.

The enumeration type is used by the following function blocks:

- [ML_GroupLoadPointFile on page 223](#)
- [ML_GroupSavePointFile on page 226](#)

Library	Range
ML_Robot	Data Types/Robot Types

Tab.5-271: Library assignment

Type definition

```

TYPE E_KIN_TEACH_IN_POINT_FILE_OPTIONS :
(
  E_GROUP_TEACH_IN_NEW           := 0, (*new points*)
  E_GROUP_TEACH_IN_APPEND        := 1, (*append points*)
  E_GROUP_TEACH_IN_APPEND_OVERWRITE := 2, (*append points and override existing points*)
);
END_TYPE

```

When the point file is loaded, the options have the following effects:

- **New**
The contents of the kinematics are deleted.
The points of the point file are entered to the kinematics.
- **Add**
The contents of the kinematics remain unchanged.
Points that are not contained in the kinematics are added from the point file.
- **Add and overwrite**
Points contained in the kinematics and the existing point file are overwritten with values from the point file.
The contents of the other points in the kinematics remain unchanged.
The other points of the existing point file are added to the kinematics.

When the point file is saved, the options have the following effects:

- The selected point file is created if it does not exist yet
- If the selected point file already exists, there are the following options
 - **New**
The contents of the existing point file are deleted.
The points of the kinematics are entered to the point file.
 - **Add**
The contents of the existing point file remain unchanged.
Points that are not contained in the existing point file are added from the kinematics of the point file.
 - **Add and overwrite**
Points contained in the existing point file and the kinematics file are overwritten with values from the kinematics.
The contents of the other points in the existing point file remain unchanged.
The other points of the kinematics are added to the point file.

5.4.6 Movement Data Types

ML_ARR_COORDS

Kurzbeschreibung The ARRAY saves the coordinates of a kinematic position in joint, world or workpiece coordinates.

Library	Range
ML_Robot	Data types/ Robot types

Tab.5-272: Library assignment

ML_Robot.library

Program:

```

TYPE (* 1..max. number of the kinematic coordinates *)
ML_ARR_COORDS:ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL;
END_TYPE

```

ML_ARR_PLACEMENT

Short description The ARRAY saves the coordinate of a position, e.g. a workpiece or a tool.

Library	Range
ML_Robot	Data types/ Robot types

Tab.5-273: Library assignment

Program:

```

TYPE (* 1..max. sum of coordinates for a placement *)
ML_ARR_PLACEMENT :ARRAY [1..MB_MAX_KIN_COORDINATES_NUMBER] OF REAL;
END_TYPE

```

ML_CIRCMODES

Type definitions The enumeration type "ML_CIRCMODES" defines the interpretation of the "AuxPoint" input:

The following function block uses this enumeration type to define the circle segment to be traveled:

[MC_MoveCircularAbsolute on page 166](#)

Element	Description
CM_BOARDER	The "AuxPoint" input defines a point to be traveled through on the circle segment

Tab.5-274: Elements of the enumeration type ML_CIRCMODES

ML_COORDSYS

Brief description The enumeration type "ML_COORDSYS" defines in which coordinate system the points have to be specified.

The enumeration type is used by the following function blocks of the "ML_Robot" library to define the coordinate system:

- [MC_GroupReadActualPosition on page 148](#)
- [MC_MoveCircularAbsolute on page 166](#)
- [MC_MoveCircularRelative on page 169](#)
- [MC_MoveDirectAbsolute on page 171](#)
- [MC_MoveDirectRelative on page 173](#)
- [MC_MoveLinearAbsolute on page 175](#)
- [MC_MoveLinearRelative on page 177](#)
- [MC_SetCartesianTransform on page 129](#)
- [ML_GroupJogCont on page 181](#)
- [ML_GroupJogStep on page 183](#)
- [ML_GroupReadNextPoint on page 224](#)

- [ML_GroupReadPoint on page 225](#)
- [ML_GroupWritePoint on page 228](#)
- [ML_RobotGetTrafoValues on page 156](#)
- [ML_RobotMovePoint on page 157](#)
- [ML_RobotReadCyclicPos on page 158](#)
- [ML_RobotTransformPoint on page 160](#)
- [ML_RobotWriteCyclicPos on page 188](#)

Element	Description
CS_MACHINE_COORDINATES	Machine coordinate system
CS_AXES_COORDINATES	Axis coordinate system
CS_BASE_COORDINATES	Base coordinate system
CS_GRIPPER_COORDINATES	Gripper coordinate system
CS_PRODUCT_COORDINATES_1	Product coordinate system 1
CS_PRODUCT_COORDINATES_2	Product coordinate system 2
CS_PRODUCT_COORDINATES_3	Product coordinate system 3
CS_PRODUCT_COORDINATES_4	Product coordinate system 4
CS_PRODUCT_COORDINATES_5	Product coordinate system 5
CS_PRODUCT_COORDINATES_6	Product coordinate system 6
CS_PRODUCT_COORDINATES_7	Product coordinate system 7
CS_PRODUCT_COORDINATES_8	Product coordinate system 8
CS_PRODUCT_COORDINATES_9	Product coordinate system 9
CS_PRODUCT_COORDINATES_10	Product coordinate system 10
CS_PRODUCT_COORDINATES_11	Product coordinate system 11
CS_PRODUCT_COORDINATES_12	Product coordinate system 12
CS_PRODUCT_COORDINATES_13	Product coordinate system 13
CS_PRODUCT_COORDINATES_14	Product coordinate system 14
CS_PRODUCT_COORDINATES_15	Product coordinate system 15
CS_PRODUCT_COORDINATES_16	Product coordinate system 16

Tab.5-275: Elements of the enumeration type ML_COORDSYS

ML_CYCLIC_MODE

Brief description The enumeration type "ML_CYCLIC_MODE" defines how the cyclic command value specification is to take place.

The enumeration type is used by function block

[ML_RobotWriteCyclicPos on page 188](#)

ML_Robot.library

Element	Description
CYCLIC_MODE_POS_ABS	The cyclic command value specification is an absolute position.
CYCLIC_MODE_POS_REL	The cyclic command value specification is a relative delta for the last commanded position in each cycle
CYCLIC_MODE_VEL	The cyclic command value specification is a velocity.

Tab.5-276: Definition of the elements of the enumeration type ML_CY-CLIC_MODE

ML_SET_MODE

Brief description The enumeration type "ML_SET_MODE" defines the blending type, i.e. whether the motion ends at the end point or is blended.

The enumeration type is used by function blocks

- [MC_MoveLinearAbsolute on page 175](#)
- [MC_MoveLinearRelative on page 177](#)
- [MC_MoveDirectAbsolute on page 171](#)
- [MC_MoveCircularAbsolute on page 166](#)

Element	Description
SM_TO	The motion ends at the end point with a velocity of 0
SM_VIA	The motion can be blended

Tab.5-277: Definition of the elements of the enumeration type ML_SET_MODE

ML_SLOPE_TYPE

Brief description The enumeration type "ML_SLOPE_TYPE" defines the slope type of the motion.

The enumeration type is used by the following function blocks:

- [MC_MoveLinearAbsolute on page 175](#)
- [MC_MoveLinearRelative on page 177](#)
- [MC_MoveDirectAbsolute on page 171](#)
- [MC_MoveDirectRelative on page 173](#)
- [MC_MoveCircularAbsolute on page 166](#)
- [MC_MoveCircularRelative on page 169](#)

Element	Description
ST_BLOCK_SLOPE	The spatial deviation in the blending range depends on the velocity
ST_PROGRAM_SLOPE	The spatial deviation in the blending range does not depend on the velocity
ST_CONTINUOUS_SLOPE	The spatial deviation in the blending range does not depend on the velocity and the continuity of the path acceleration is maintained

Tab.5-278: Elements of the enumeration type ML_SLOPE_TYPE

5.4.7 RCL Data Types

ML_POINT_TYPE

Short description The enumeration type contains the point coordinate systems that can be selected.

Library	Range
ML_Robot	Data types/Robot types

Tab.5-279: Library assignment

Program:

```

TYPE ML_KIN_E_JOG_VEL :
(
  ML_ROCO_WORLD_COORD := 0, (*point in world coordinates*)
  ML_ROCO_JOINT_COORD := 1 (*point in joint coordinates*)
);
END_TYPE

```

ML_ROCO_FILE_DIR

Short description This data structure contains the properties of a file.

Library	Range
ML_Robot	Data types/Robot types

Tab.5-280: Library assignment

Structure element	Type	Default value	Description
FileName	ML_ROCO_NAME	''	File name including file extension
FileSize	UDINT	0	File size
FileDate	STRING(15)	''	File date and time
FileAttrib	STRING(4)	''	File attribute: 'H' = Hidden = file is hidden 'W' = Write ' = read and write 'R' = Read = read-only 'D' = Delete = file can be deleted
KinNbr	DINT	0	Kinematics number

Tab.5-281: Interface description data structure ML_ROCO_FILE_DIR

ML_ROCO_NAME

Short description This data type is the general name definition in the kinematics field.
It is also used for the tool and workpiece names as well as for the file names including the extension.

Library	Range
ML_Robot	Data types/Robot types

Tab.5-282: Library assignment

ML_Robot.library

Program:

```

TYPE (* 12 char (blanks filled) + zero + alignment (len=16) *)
  ML_ROCO_NAME : STRING(15);
END_TYPE

```

ML_ROCO_POINT**Short description** This data structure contains the properties of a point.

Library	Library
ML_Robot	DataTypes/Robot Types

Tab.5-283: Library assignment

Structure element	Type	Default value	Description
PointName	ML_ROCO_POINT_NAME	``	Point name
NbrOfValues	USINT	3	Number of coordinates per point
PointValue	ML_ARR_COORDS		Point value
PointDef	DWORD	0	Specification, whether or not the point value is defined. Each bit of this value indicates whether the value is or is not defined. The value FALSE means that the value is not defined. TRUE means that the value is defined. That means PointDef.0 -> PointValue[1] PointDef.1 -> PointValue[2]
KinematicNbr	UINT	1	Number of the kinematic in which the point is used
PointType	ML_POINT_TYPE	ML_ROCO_WORLD_COORD	Type of point ML_ROCO_WORLD_COORD = point in WC, ML_ROCO_JOINT_COORD = Point in JC
ArrayDimension	DINT	0	Number of dimensions of an array. Always 0 for single points
ArrayIndicesLimits	ARRAY [1..ML_MAX_KIN_ARRAY_DIMENSION, 1..2] OF DINT	0	Start and end index for every array dimension

Tab.5-284: Interface description data structure ML_ROCO_POINT

ML_ROCO_POINT_NAME**Short description** This data type contains the name of a point.

Library	Range
ML_Robot	Data types/Robot types

Tab.5-285: Library assignment

Program:

```

TYPE
  ML_ROCO_POINT_NAME : STRING(12); (* 12 char *)
END_TYPE

```


6 Kinematics Interface

6.1 Kinematics Interface, Introduction

The **kinematics interface** combines and extends the Robot Control functionalities. It is the counterpart to the **axis interface** and is designed for easy and comfortable operation of the kinematics. This makes it possible to develop well structured application programs for Robot Control applications within a short time.



The kinematics interface may be used for both Robot Control V1 and Robot Control V2.

However, certain operation modes are only permissible for one Robot Control version.

For details, refer to [Operation Modes on page 290](#)

System requirements

The kinematics interface requires the following hardware and software:

- Software: IndraMotion MLC 13VRS
- Firmware: IndraMotion ML* 13VRS
- *CML control:*
 - IndraControl L25 or
 - IndraControl L45 or
 - IndraControl L65
- Template: Axis interface example project or GAT

Initial Commissioning

A requirement for the kinematics interface is that the axis interface is integrated and adapted to coordinated motions (Robot Control). If this was already done, the following steps may be skipped.

If the project was created using the axis interface example project, GAT^{compact}, GAT with central state machine or GAT with decentral state machine, the following steps have to be carried out.

Axis interface example project

1. Rename the structure "TE_AXIS_CONTROL_Type01" to "TE_AXIS_CONTROL_TYPE02" and extend it by the structure variable "CoordModeCtrl" of type "MB_AXIS_COORDINATED"
CoordModeCtrl: MB_AXIS_COORDINATED;
2. Rename the structure "TE_AXIS_STATUS_Type01" to "TE_AXIS_STATUS_TYPE02"
TE_AXIS_STATUS_TYPE02
3. The declaration for both structure variables "arAxisCtrl_gb" and "arAxisStatus_gb" has to be changed to the corresponding type02 structures
arAxisCtrl_gb: ARRAY [] OF TE_AXIS_CONTROL_TYPE02;
arAxisStatus_gb: ARRAY [] OF TE_AXIS_STATUS_TYPE02;
4. Change the basic class of the function block "TE_Axisinterface" to "MB_AxisInterfaceType02"
TE_AxisInterface EXTENDS MB_AxisInterfaceType02

Kinematics Interface

- In method "TE_Axisinterface.mInitExtensions()" initialize the additional input "CoordModeCtrl" of the function block "MB_AxisInterfaceType02"
CoordModeCtrl REF= arAxisCtrl_gb[index].CoordModeCtrl;



The structures TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 are contained in the folder: "AxisInterface\Type".

The structure variables arAxisCtrl_gb, arAxisStatus_gb are contained in the Global Variable List "UserVarGlobal".

GATcompact:

- Rename the structure "TE_AXIS_CONTROL_Type01" to "TE_AXIS_CONTROL_TYPE02" and extend it by the structure variable "CoordModeCtrl" of type "MB_AXIS_COORDINATED"
CoordModeCtrl: MB_AXIS_COORDINATED;
- Rename the structure "TE_AXIS_STATUS_Type01" to "TE_AXIS_STATUS_TYPE02"
TE_AXIS_STATUS_TYPE02
- The declaration for both structure variables "arAxisCtrl_gb" and "arAxisStatus_gb" has to be changed to the corresponding type02 structures
arAxisCtrl_gb: ARRAY [] OF TE_AXIS_CONTROL_TYPE02;
arAxisStatus_gb: ARRAY [] OF TE_AXIS_STATUS_TYPE02;
- Change the basic class of the function block "TE_Axisinterface" to "MB_AxisInterfaceType02"
TE_AxisInterface EXTENDS MB_AxisInterfaceType02
- In method "TE_Axisinterface.mInitExtensions()" initialize the additional input "CoordModeCtrl" of the function block "MB_AxisInterfaceType02"
CoordModeCtrl REF= arAxisCtrl_gb[].CoordModeCtrl;
- Adjust the "TE_GAT_AXIS" references to the new type02 structures.
CTRL: REFERENCE TO TE_AXIS_CONTROL_TYPE02
Status: REFERENCE TO TE_AXIS_STATUS_TYPE02



The structures TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 are contained in the folder: "GATcompact\AxisInterface\Type".

The structure variables arAxisCtrl_gb, arAxisStatus_gb are contained in the Global Variable List "UserVarGlobal".

The structure TE_GAT_AXIS is in the folder "GATcompact"

GAT with central state machine

- Rename the structure "TE_AXIS_CONTROL_Type01" to "TE_AXIS_CONTROL_TYPE02" and extend it by the structure variable "CoordModeCtrl" of type "MB_AXIS_COORDINATED"
CoordModeCtrl: MB_AXIS_COORDINATED;

Kinematics Interface

2. Rename the structure "TE_AXIS_STATUS_Type01" to "TE_AXIS_STATUS_TYPE02"
TE_AXIS_STATUS_TYPE02
3. Adjust the basic classes of the structures "TE_AXISINTERFACE_CONTROL" and "TE_AXISINTERFACE_STATUS" accordingly (Type01 -> Type02)
TE_AXISINTERFACE_CONTROL EXTENDS TE_AXIS_CONTROL_TYPE02
TE_AXISINTERFACE_STATUS EXTENDS TE_AXIS_STATUS_TYPE02
4. Change the basic class of the function block "TE_Axisinterface" to "MB_AxisInterfaceType02"
TE_AxisInterface EXTENDS MB_AxisInterfaceType02
5. In method "TE_Axisinterface.mInitExtensions()" initialize the additional input "CoordModeCtrl" of the function block "MB_AxisInterfaceType02"
CoordModeCtrl REF= arAxisCtrl_gb[].CoordModeCtrl;



The structures TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 are contained in the folder: "GAT\AxisInterface\Type".

The structures TE_AXISINTERFACE_CONTROL and TE_AXISINTERFACE_STATUS are contained in the folder: "GAT\AxisInterface\Assignments"

GAT with a decentral state machine

1. Rename the structure "TE_AXIS_CONTROL_Type01" to "TE_AXIS_CONTROL_TYPE02" and extend it by the structure variable "CoordModeCtrl" of type "MB_AXIS_COORDINATED"
CoordModeCtrl: MB_AXIS_COORDINATED;
2. Rename the structure "TE_AXIS_STATUS_Type01" to "TE_AXIS_STATUS_TYPE02"
TE_AXIS_STATUS_TYPE02
3. The declaration for both structure variables "arAxisCtrl_gb" and "arAxisStatus_gb" has to be changed to the corresponding type02 structures
arAxisCtrl_gb: ARRAY [] OF TE_AXIS_CONTROL_TYPE02;
arAxisStatus_gb: ARRAY [] OF TE_AXIS_STATUS_TYPE02;
4. Change the basic class of the function block "TE_Axisinterface" to "MB_AxisInterfaceType02"
TE_AxisInterface EXTENDS MB_AxisInterfaceType02
5. In method "TE_Axisinterface.mInitExtensions()" initialize the additional input "CoordModeCtrl" of the function block "MB_AxisInterfaceType02"
CoordModeCtrl REF= arAxisCtrl_gb[].CoordModeCtrl;
6. Adjust the types of "arAxisCtrl" and "arAxisStatus" to type02 structures.

Kinematics Interface

```

arAxisCtrl: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_CON-
TROL_TYPE02;
arAxisStatus: ARRAY [1..uiMAX_AXIS_INDEX] OF TE_AXIS_STA-
TUS_TYPE02;

```

- Adjust the "TE_GAT_AXIS" references to the new type02 structures.

CTRL: REFERENCE TO TE_AXIS_CONTROL_TYPE02

Status: REFERENCE TO TE_AXIS_STATUS_TYPE02



The structures TE_AXIS_CONTROL_TYPE02, TE_AXIS_STATUS_TYPE02 are contained in the folder: "GAT\AxisInterface\Type".

The structure variables arAxisCtrl_gb, arAxisStatus_gb are in the Global Variable List "Global_GAT_Axis_Interfaces".

The types arAxisCtrl and arAxisStatus are in "GAT\AxisInterface\TE_AxisInitAllAxisType01"

The structure TE_GAT_AXIS is in the folder "GAT\MachineUnits"

Integration of the kinematics interface

Carry out the following steps to integrate the kinematics interface:

- Start the GAT Wizard by clicking on .
- Select the "Plug-ins" node
- Select the function "Add Plug-in"
- Select the "KinInterface" plug-in and confirm import with **Apply**

6.2 ML_KinTech.library

6.2.1 ML_KinTech.library, Introduction and Overview

ML_KinTech provides the following function blocks:

Function block	Description
ML_KinematicsInitType01 on page 280	Used to initialize the kinematics interface for all kinematics. The function block only needs to be called once when the program is started and/or each time the phase is switched from the parameterization mode to the operating mode
ML_KinematicsInterfaceType01 on page 283	Used to configure the kinematics interface for all kinematics. This function block has to be called cyclically as long as the operating mode is active
ML_KinJogV1 on page 285	Used to jog individual axes or axis groups of a kinematics with Robot Control V1
ML_KinJog on page 287	Used to jog individual axes or axis groups of a kinematics with Robot Control V2

Tab.6-1: ML_KinTech function blocks

The kinematics interface is provided as a plug-in for the IndraMotion programming template axis interface. For this reason, its structure and mode of operation is similar to the axis interface. The axis in the axis interface is replaced by the kinematics in the kinematics interface.

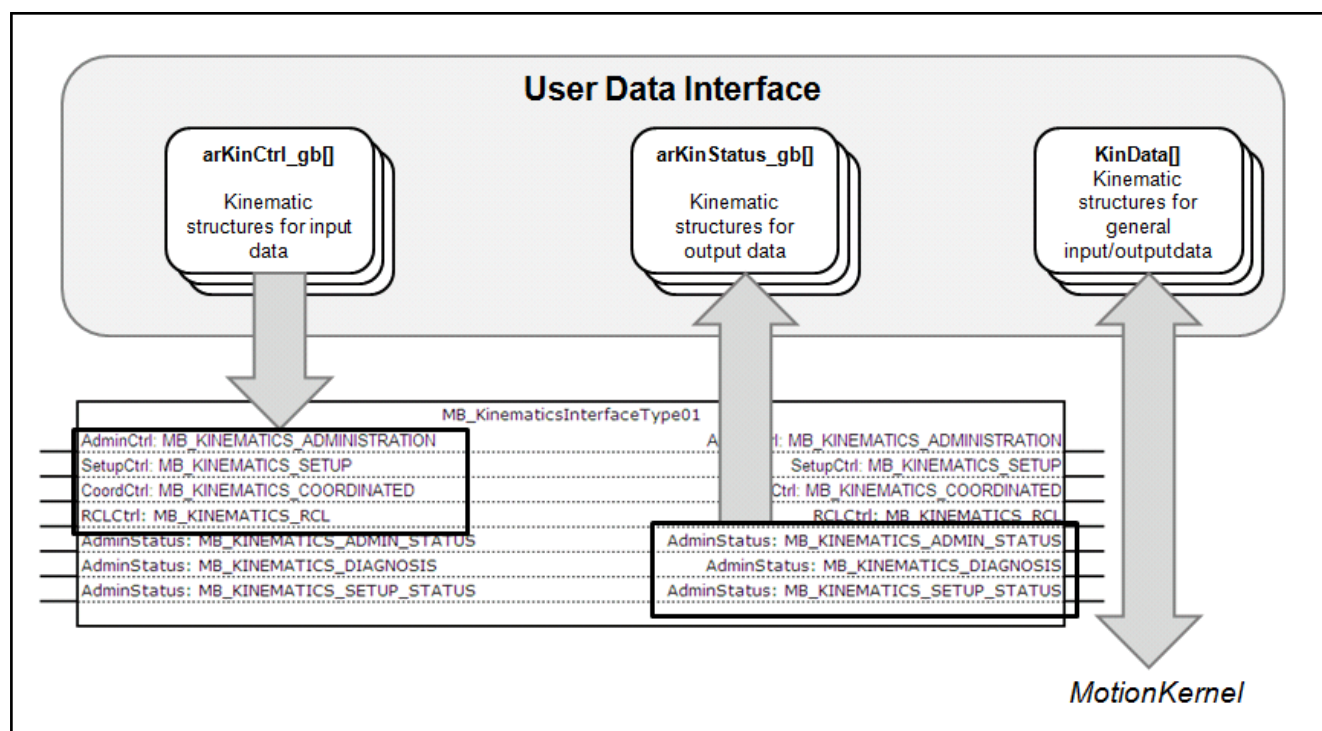


Fig.6-1: Kinematics interface - data structure of the interface

The following table shows the user interface for the axis interface data structure:

User interface	Type	Description
arKinCtrl[]_gb	MB_KINEMATICS_CONTROL_TYPE01 (user extension: TE_KINEMATICS_CONTROL_TYPE01) MB_KINEMATICS_CONTROL_TYPE02 (user extension: TE_KINEMATICS_CONTROL_TYPE02)	Control structure, including command values and variables to activate the operation modes
arKinStatus[]_gb	MB_KINEMATICS_STATUS_TYPE01 (user extension: TE_KINEMATICS_CONTROL_TYPE01) MB_KINEMATICS_STATUS_TYPE02 (user extension: TE_KINEMATICS_CONTROL_TYPE02)	Status structure, including diagnostic information and acknowledgements for the operation modes
KinData[]		Status and control structure including actual values and status bits (processed by MotionControl in the firmware)

Tab.6-2: Kinematics interface - data structure of the user interface

6.2.2 Kinematics Interface Structures

Kinematics Interface Structures, Overview

The kinematics interface provides a data interface for easy kinematics operation. The data structures of the kinematics interface are described below.



The kinematics interface can be extended by the user.

In order to enable these extensions, user structures have to be defined in the user project that extend the library structures.

The extended structures are identified by the prefix "TE_".

For example, if a structure is called

TE_KINEMATICS_ADMIN_STATUS,

it is an extended structure of

MB_KINEMATICS_ADMIN_STATUS.

Information on the structure can be found under the name MB_KINEMATICS_ADMIN_STATUS.

MB_KINEMATICS_CONTROL_TYPE01

MB_KINEMATICS_CONTROL_TYPE01, Overview

MB_KINEMATICS_CONTROL_TYPE01 on page 256		
	MB_KINEMATICS_ADMINISTRATION on page 257	
		MB_KINEMATICS_MODE_UNION on page 260
		MB_KINEMATICS_CONVEYOR_BELT on page 261
		MB_KINEMATICS_CONVEYOR_BELT_V1 on page 263
		MB_KINEMATICS_RCL_POINT on page 264
		MB_KINEMATICS_RCL_PROG on page 266
	MB_KINEMATICS_SETUP on page 268	
	MB_KINEMATICS_COORDINATED on page 270	
	MB_KINEMATICS_RCL on page 272	

Tab.6-3: MB_KINEMATICS_CONTROL_TYPE01, structural setup

MB_KINEMATICS_CONTROL_TYPE01

Brief description The arKinCtrl_gb[] instance accesses the MB_KINEMATICS_CONTROL_TYPE01 data structure.

MB_KINEMATICS_CONTROL_TYPE01

```
1 TYPE MB_KINEMATICS_CONTROL_TYPE01 :
2   STRUCT
3     Admin: MB_KINEMATICS_ADMINISTRATION; (* Administration of the kinematics | Verwaltung der Kinematik *)
4
5     SetupMode: MB_KINEMATICS_SETUP; (* Jog/Homing functionality | Tipp/Einrichtfunktionen *)
6
7     CoordMode: MB_KINEMATICS_COORDINATED; (* operating mode coordinated motion (linear, circular, direct; absolute, relative) |
8                                             Koordinierbetriebsart (Linear, Zirkular, FTP) *)
9
10    RCLMode: MB_KINEMATICS_RCL; (* operating mode robot control language (RCL) |
11                                RCL-Betriebsart*)
12  END_STRUCT
13 END_TYPE
```

Fig.6-2: Elements of MB_KINEMATICS_CONTROL_TYPE01

- Admin
- Access to kinematics management attributes
- SetupMode
- Access to attributes of the jog/setup function
- CoordMode
- Access to attributes of the positioning modes
- RCLMode
- Access to attributes of the RCL operation mode

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_ADMINISTRATION

Brief description The MB_KINEMATICS_ADMINISTRATION structure is a substructure of "MB_KINEMATICS_CONTROL_TYPE01" and contains control variables and command values for kinematics management.

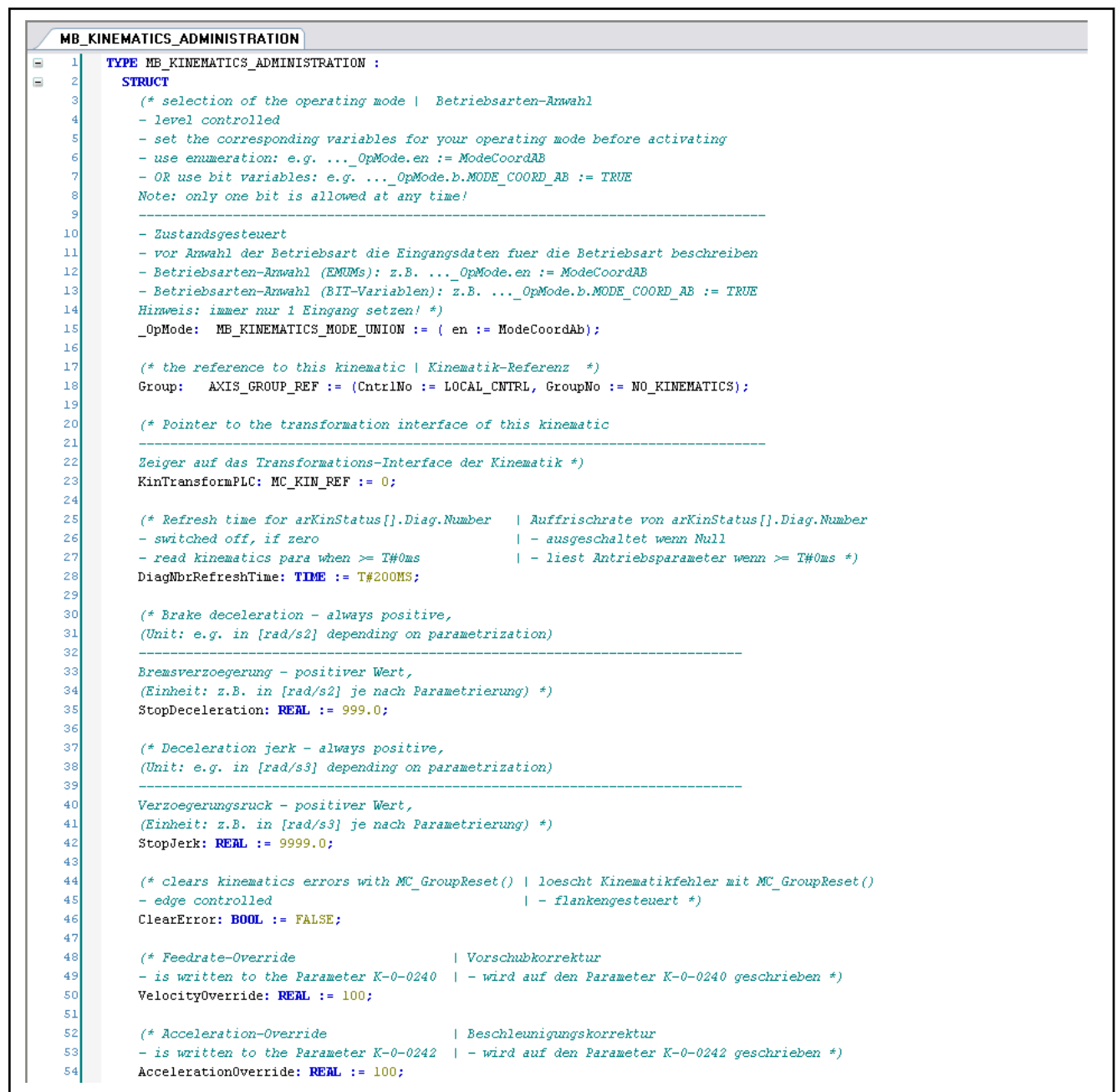


Fig. 6-3: MB_KINEMATICS_ADMINISTRATION (part 1)

Kinematics Interface

```

56      (* Deceleration-Override          | Verzögerungskorrektur
57      - is written to the Parameter K-0-0243 | - wird auf den Parameter K-0-0243 geschrieben *)
58      DecelerationOverride: REAL := 100;
59
60      (* Enable the cyclic check and transmission of changed inputs for the actual operating mode
61      - relevant for the values in arKinCtrl_gb with note "Note: cyclic checked AND transmitted on change"
62      - level controlled
63      -----
64      Zuschalten des bedarfsweisen Scannens von Eingangsdaten in der aktuellen Betriebsart
65      - betrifft Groessen in arKinCtrl_gb mit Anm. "Hinweis: wird permanent zyklisch verarbeitet"
66      - Zustandsgesteuert *)
67      EnableCyclicScanning: BOOL := TRUE;
68
69      (* Switch to parameter level 1, parameter A-0-0024 bit 3
70      - TRUE: activate parameter level 1
71      - FALSE: deactivate parameter level 1
72      -----
73      Umschalten der Achsen in den Parametriermodus 1, Parameter A-0-0024 Bit 3
74      - TRUE: aktiviert den Parametriermodus
75      - FALSE: deaktiviert den Parametriermodus *)
76      ParamLevel: BOOL := FALSE;
77
78      (* Restart the same OpMode again, e.g. ModeCoordPosLinAbs with same values
79      - edge controlled
80      -----
81      Neustart der gleichen Betriebsart, z.B. ModeCoordPosLinAbs mit gleichen Werten
82      - flankengesteuert *)
83      RetriggerOpMode: BOOL := FALSE;
84
85      (* Configuration of a conveyor belt for Robot-Control V1
86      -----
87      Konfiguration eines Transportbandes für Robot-Control V1 *)
88      ConveyorBeltV1: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF MB_KINEMATICS_CONVEYOR_BELT_V1;
89
90      (* Configuration of a conveyor belt for Robot-Control V2
91      -----
92      Konfiguration eines Transportbandes für Robot-Control V2 *)
93      ConveyorBelt: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF MB_KINEMATICS_CONVEYOR_BELT;
94
95      (* Handling of RCL points | Handhabung von RCL-Punkten
96      - set a single point      | - Setzen eines einzelnen Punktes
97      - read a single point     | - Lesen eines einzelnen Punktes
98      - read a points list      | - Lesen einer Punkteliste
99      - read a program list     | - Lesen einer Programmliste
100     - ONLY Robot-Control V1    | - NUR mit Robot-Control V1 zulässig *)
101     RCLPoint: MB_KINEMATICS_RCL_POINT;
102
103     (* Handling of RCL programs | Handhabung von RCL-Programmen
104     - load a program           | - Laden eines Programms
105     - save a program           | - Speichern eines Programms
106     - compile a program        | - Kompilieren eines Programms
107     - ONLY Robot-Control V1    | - NUR mit Robot-Control V1 zulässig*)
108     RCLProg: MB_KINEMATICS_RCL_PROG;
109
110     (* Activities after changing operation mode
111     - NoAction: execute the latest block and command the next block
112     - StopMovement: stop the latest block and start the next block
113     - AbortMovement: abort the latest block and blend to the next block
114     -----
115     Verhalten nach einem Betriebsartenwechsel
116     - NoAction: führe den aktuellen Satz aus und kommandiere den nächsten Satz
117     - StopMovement: stoppe den aktuellen Satz und starte den nächsten Satz
118     - AbortMovement: unterbreche den aktuellen Block und verschleife auf den nächsten Satz *)
119     ChangeOpMode: MB_KINEMATICS_ADMIN_CHANGE_OPMODE := NoAction;
120     END_STRUCT
121     END_TYPE

```

Fig.6-4: MB_KINEMATICS_ADMINISTRATION (part 2)

Kinematics Interface

OpMode	Selecting the operation mode For more information on operation mode switching, see Overview, page 290  Setting several bits in <code>arKinCtrl_gb[].Admin._OpMode.b</code> causes an error.
Group	This element contains information on the control and kinematics number and serves as a reference for the internally used function blocks.
KinTransformPLC	With this element, it is possible to register user-defined transformations. If a transformation function block is created, it is registered in the PLC.  Registration of user-defined transformations is only possible in parameterization mode P2. The attempt to register a transformation in operating mode BB returns no result.  If the kinematics interface is started with a user-defined transformation for the first time, the control returns the error message "Transformation not registered". This is due to the IMC interface trying to switch to operating mode. When the transformation is registered, the error may be acknowledged.
DiagNbrRefreshTime	This element controls the update rate of <code>arKinStatus_gb[].Diag</code>.
StopDeceleration	The value of this element indicates the deceleration applied for stopping the kinematics.  Only permissible for Robot Control V2.
StopJerk	The value of this element indicates the deceleration jerk applied for stopping the kinematics.  Only permissible for Robot Control V2.
ClearError	This edge-controlled element deletes kinematics and axis errors. A rising edge also calls the function block "MC_GroupReset".
VelocityOverride	This element indicates the feed correction. The value is internally written to parameter K-0-0240.
AccelerationOverride	This element indicates the acceleration correction. The value is internally written to parameter K-0-0242.
DecelerationOverride	This element indicates the deceleration correction. The value is internally written to parameter K-0-0243.
EnableCyclicScanning	This state-controlled element controls the cyclical reading of <code>arKinCtrl_gb[].CoordMode.Point</code>. If this element is set, a new motion is started with each new position and thereby a new block is entered into the block buffer.
ParamLevel	This edge-controlled element switches all axes belonging to the kinematics to parameterization mode. By resetting the element, the axes are reset to their default state.  If the parameterization mode is active, "PM" is displayed in the project tree with the respective axes.
RetriggerOpMode	This edge-controlled element restarts the operation mode currently in use. If it is set, the kinematics interface internally switches to "ModeCoordExternalFB" and then back to the original operation mode. Therefore, relative trav-

Kinematics Interface

	eling of the kinematics is possible several times without changing the position values.
ConveyorBeltV1	Access to the attributes to configure belts for Robot Control V1 For details, refer to MB_KINEMATICS_CONVEYOR_BELT_V1 on page 263  Only permissible for Robot Control V1. A maximum of 4 belts per kinematics may be configured.
ConveyorBelt	Access to the attributes to configure belts for Robot Control V2 For details, refer to MB_KINEMATICS_CONVEYOR_BELT on page 261  Only permissible for Robot Control V2. A maximum of 16 belts per kinematics may be configured.
RCLPoint	Access to attributes for handling RCL points For details, refer to MB_KINEMATICS_RCL_POINT on page 264  Only permissible for Robot Control V1.
RCLProg	Access to attributes for handling RCL programs For details, refer to MB_KINEMATICS_RCL_PROG on page 266  Only permissible for Robot Control V1.
ChangeOpMode	The behavior after a change in operation mode can be defined with this element. <i>The following values are possible:</i> • NoAction - in case of a change in operation mode the motion commanded previously is completed • StopMovement - in case of a change in operation mode the current motion is stopped with the function block "MB_IRDProgStop" (Robot Control V1) or "MC_GroupStop" (Robot-Control V2)
Library	ML_KinTech.compiled-library
System	IndraMotion MLC • MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256 MB_KINEMATICS_MODE_UNION
Brief description	The union MB_KINEMATICS_MODE_UNION is a union data type used by the _OpMode element (MB_KINEMATICS_ADMINISTRATION). The union contains the enumeration type MB_KINEMATICS_MODE and the bit structure MB_KINEMATICS_MODE_BITS _OpMode may be controlled by both data types as they are on the same storage location.

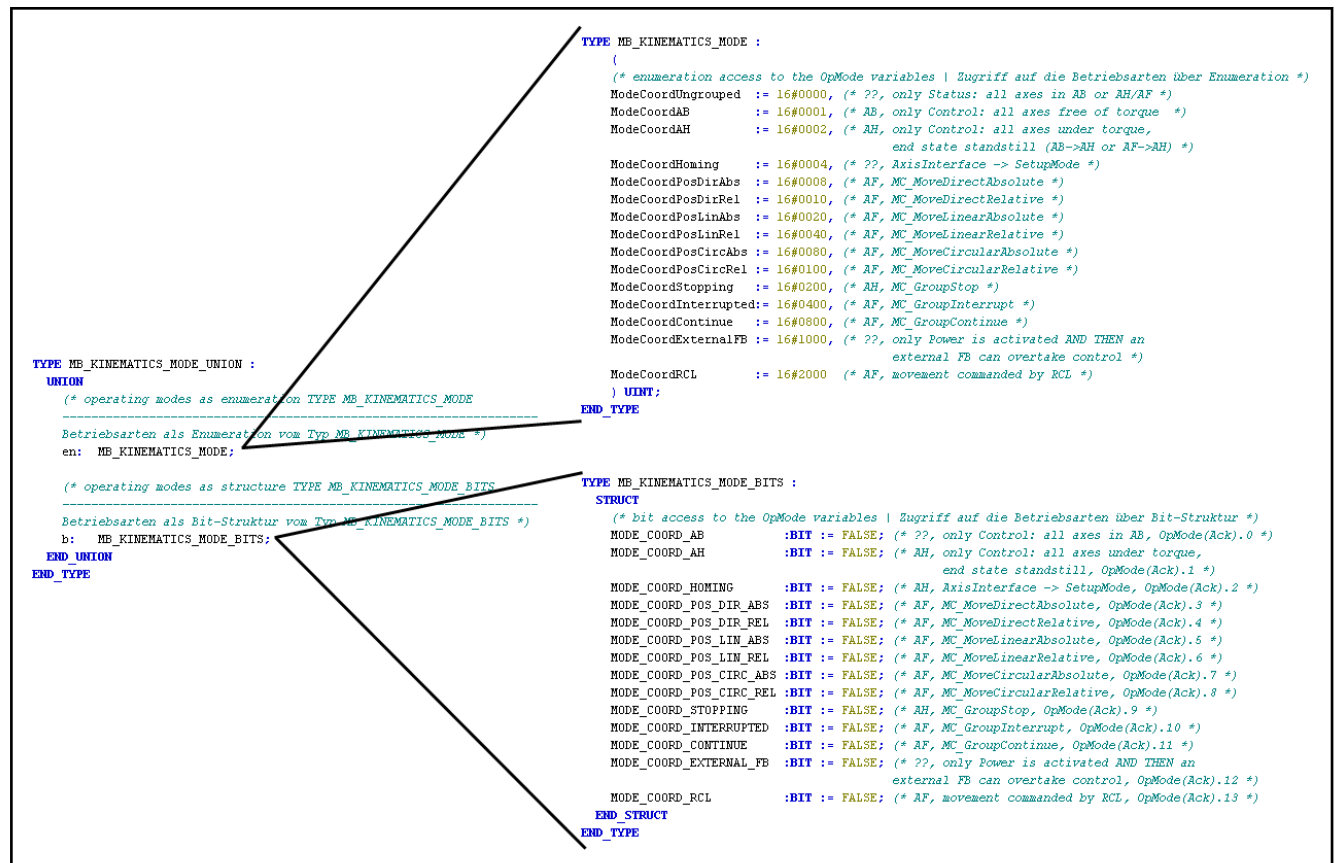


Fig. 6-5: The MB_KINEMATICS_MODE_UNION data type with MB_KINEMATICS_MODE and MB_KINEMATICS_MODE_BITS

- en Access to the operation mode via an enumeration value
- b Access to the operation mode via a bit structure

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_CONVEYOR_BELT

Brief description The structure MB_KINEMATICS_CONVEYOR_BELT is used to configure a belt with Robot Control V2.

The elements contained in this structure correspond to the inputs of the function blocks

- ML_SetBeltConfiguration,
- ML_SetBeltErrorReaction.

For more detailed information, see the respective documentation.

The following figure shows the structure elements.



Only permissible for Robot Control V2.

Kinematics Interface

```

MB_KINEMATICS_CONVEYOR_BELT
1  TYPE MB_KINEMATICS_CONVEYOR_BELT :
2  STRUCT
3      (* Execute belt configuration      | Ausführen der Bandkonfiguration
4      - edge controlled                 | - flankengesteuert *)
5      Execute: BOOL := FALSE;
6
7      (* Defines the position and rotation of the belt relative to the WCS
8      -----
9      Definiert die Position und Verdrehung des Bandes relativ zum WCS *)
10     ConveyorBeltOrigin: ARRAY [1..6] OF REAL := [6(0.0)];
11
12     (* Defines the length of the belt synchronous traverse from the starting
13     position
14     -----
15     Definiert die Länge des bandsynchronen Verfahrbereichs ab der Startposition *)
16     TotalLength: REAL := 0.0;
17
18     (* Defines the length of the latest starting for band synchronous movement
19     from the start position
20     -----
21     Definiert die Länge des spätesten Starten einer bandsynchronen Bewegung ab
22     der Startposition *)
23     BeginLength: REAL := 0.0;
24
25     (* Referenz of the belt axis
26     -----
27     Referenz der Bandachse *)
28     Axis: AXIS_REF;
29
30     (* Defines the type of the error reaction
31     -----
32     Definiert den Typ der Fehlerreaktion *)
33     ErrorReactionType: ML_BELT_ERROR_REACTION_MODE := ML_BELT_ERROR_NO_SPECIFIC_REACTION;
34
35     (* Defines the time for the desynchronization
36     -----
37     Definiert die Zeit zum absynchronisieren *)
38     ErrorReactionDesyncTime: REAL := 0.0;
39
40     (* Defines the deceleration for the desynchronization
41     -----
42     Definiert die Bremsverzögerung zum absynchronisieren *)
43     ErrorReactionDeceleration: REAL := 0.0;
44
45     (* Defines the jerk for the desynchronization
46     -----
47     Definiert den Bremsruck zum absynchronisieren *)
48     ErrorReactionJerk: REAL := 0.0;
49 END_STRUCT
50 END_TYPE

```

Fig.6-6: MB_KINEMATICS_CONVEYOR_BELT

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01](#), Overview on page 256

MB_KINEMATICS_CONVEYOR_BELT_V1

Brief description The structure MB_KINEMATICS_CONVEYOR_BELT_V1 is used to configure a belt with Robot Control V1.

The elements contained in this structure correspond to the inputs of the function block

- ML_SetBeltParameter



Only permissible for Robot Control V1.

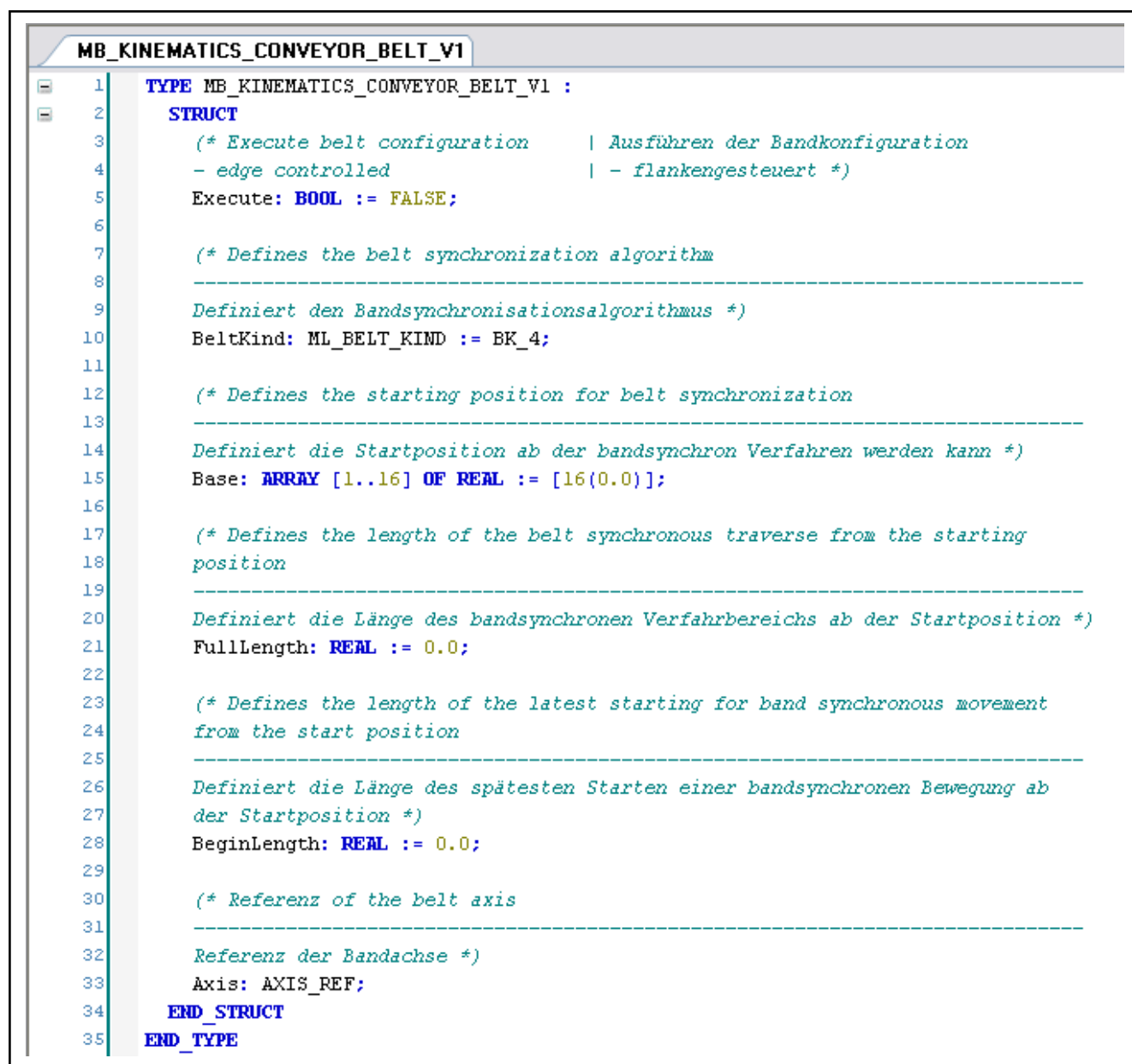


Fig. 6-7: Elements of the MB_KINEMATICS_CONVEYOR_BELT_V1 structure

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01](#), Overview on page 256

Kinematics Interface

MB_KINEMATICS_RCL_POINT

Brief description *The structure MB_KINEMATICS_RCL_POINT is used for the following*

- Writing an individual point (ML_RoCoSetPoint)
- Reading an individual point (ML_RoCoGetPoint)
- Reading a point list (ML_RoCoGetPointList)
- Reading a program list (ML_RoCoDir)



Only permissible for Robot Control V1.

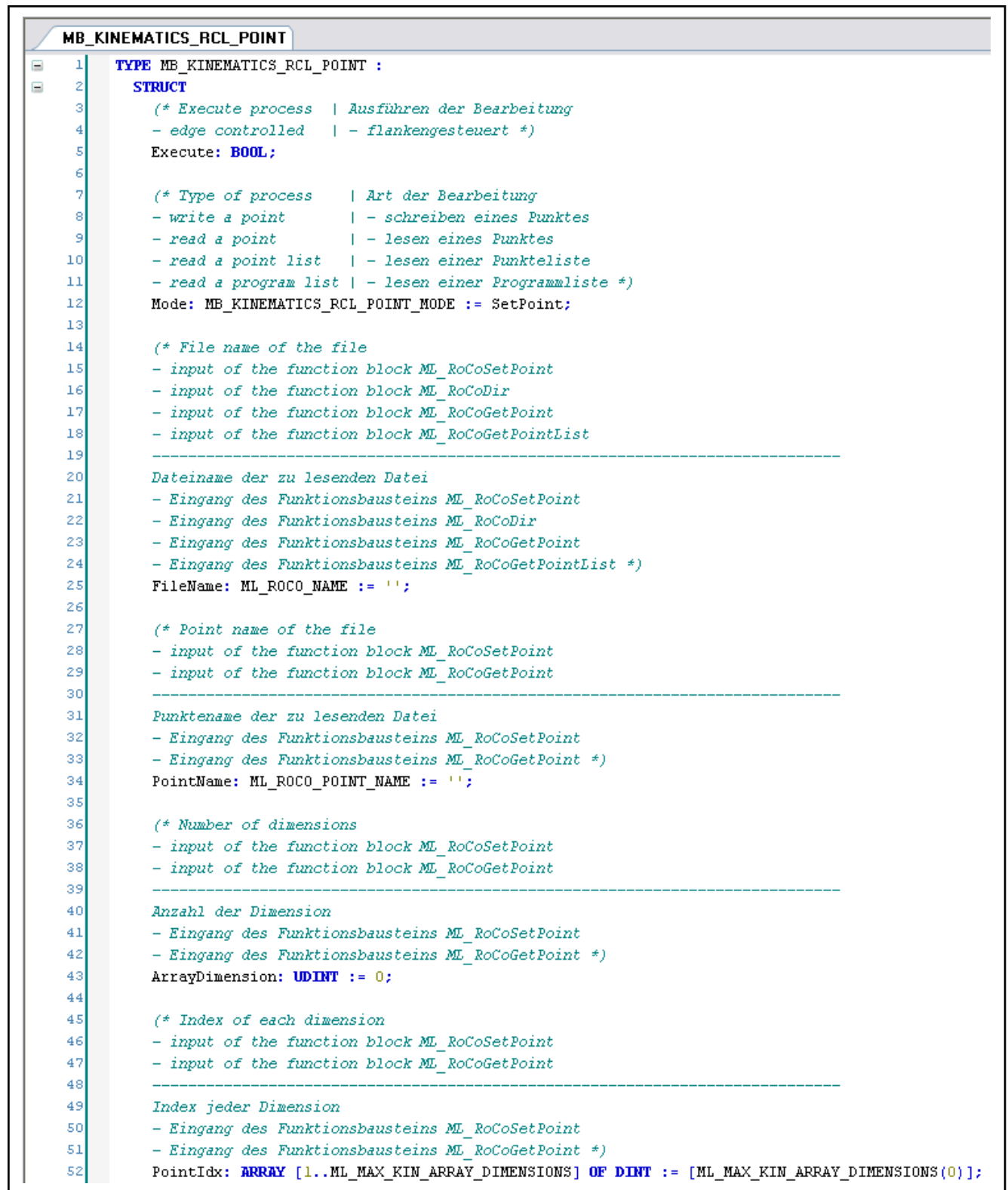


Fig.6-8: Elements of the MB_KINEMATICS_RCL_POINT structure (part 1)

Kinematics Interface

```

54      (* Index of each dimension
55      - input of the function block ML_RoCoDir
56      - input of the function block ML_RoCoGetPointList
57      -----
58      Index jeder Dimension
59      - Eingang des Funktionsbausteins ML_RoCoDir
60      - Eingang des Funktionsbausteins ML_RoCoGetPointList *)
61      StartIdx: UDINT := 1;
62
63      (* Point structure with name and value
64      - input of the function block ML_RoCoSetPoint
65      -----
66      Struktur eines Punktes mit Namen und Werte
67      - Eingang des Funktionsbausteins ML_RoCoSetPoint *)
68      Point: ML_ROCO_POINT;
69
70      (* Direction of the read-sequence(low to high, high to low)
71      - input of the function block ML_RoCoDir
72      -----
73      Reihenfolge, in welcher gelesen wird (aufsteigend, absteigend)
74      - Eingang des Funktionsbausteins ML_RoCoDir *)
75      SearchDirection: BOOL := FALSE;
76  END_STRUCT
77  END_TYPE

```

Fig.6-9: Elements of the MB_KINEMATICS_RCL_POINT structure (part 2)

Mode This element defines the type of access.

The following enumerations are possible:

- SetPoint: Writing an individual point
- GetPoint: Reading an individual point
- GetPointList: Reading a point list
- GetProgList: Reading a program list

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01](#), Overview on page 256

MB_KINEMATICS_RCL_PROG

Brief description *The structure MB_KINEMATICS_RCL_PROG is used for the following*

- Loading a RCL program from the Compact Flash card to the internal memory of the control
- Storing a RCL program from the internal memory of the control to the Compact Flash card
- Compiling a RCL program in the control

The parameters K-0-1060, K-0-1061 and K-0-1062 are used for these actions:



Only permissible for Robot Control V1.

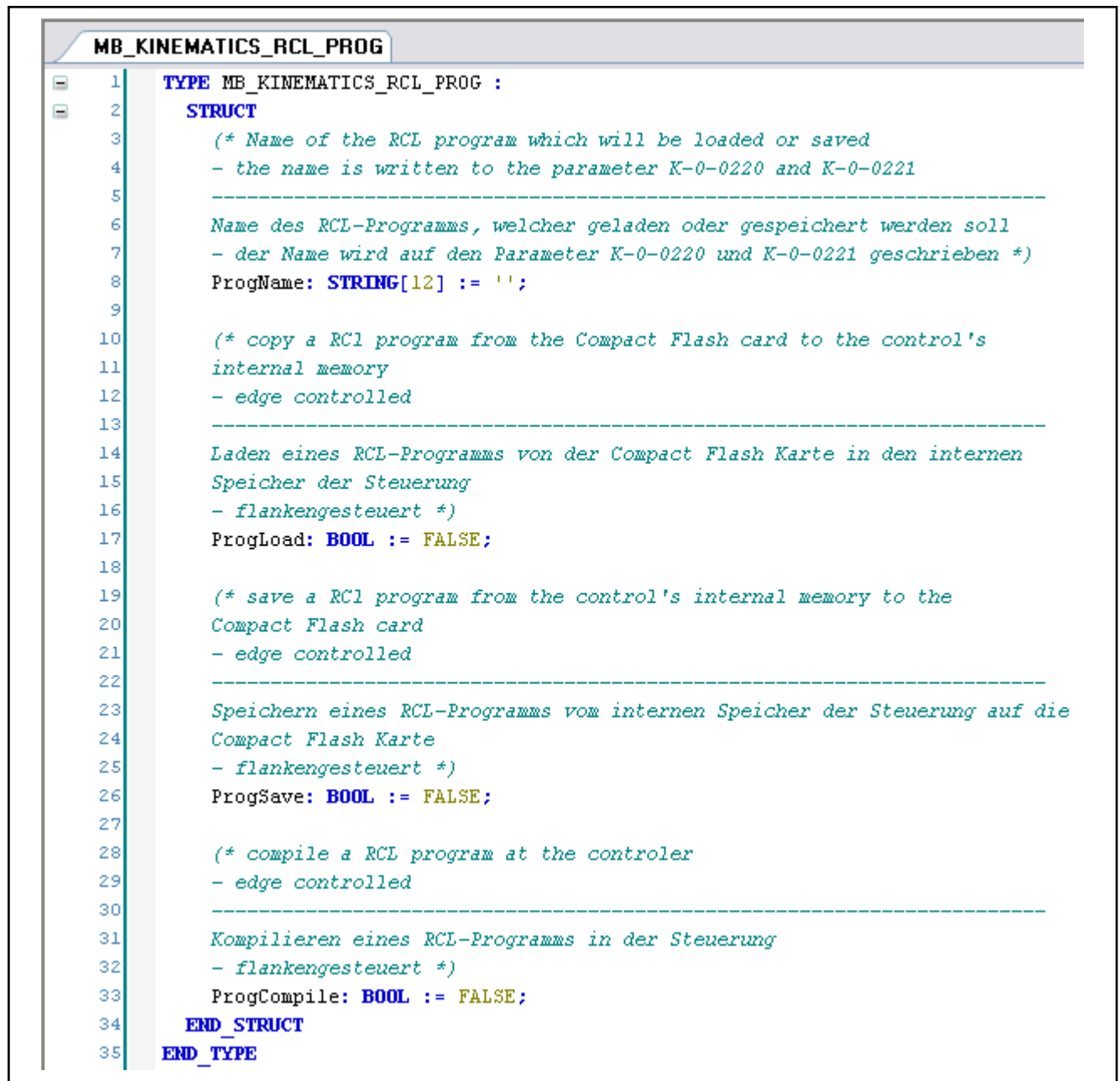


Fig.6-10: MB_KINEMATICS_RCL_PROG

ProgName	This element contains the name of the RCL program to be read or stored. The name is internally written to the parameters K-0-0220 and K-0-0221.
ProgLoad	This element triggers a command which copies the file(s) specified in parameter K-0-0220 from the Compact Flash card to the internal memory of the control. Parameter K-0-1060 is internally used for this purpose.
ProgSave	This element triggers a command which copies the file(s) specified in parameter K-0-0220 from the internal memory of the control to the Compact Flash card. Parameter K-0-1061 is internally used for this purpose.
ProgCompile	This element compiles the QLL file specified in parameter K-0-0221. Parameter K-0-1062 is internally used for this purpose.
Library	ML_KinTech.compiled-library
System	IndraMotion MLC

Kinematics Interface

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_SETUP

Brief description The MB_KINEMATICS_SETUP structure is a substructure of MB_KINEMATICS_CONTROL_TYPE01 and contains control variables to jog kinematics.



Fig.6-11: MB_KINEMATICS_SETUP

- Enable** This element activates the jog/setup mode.
- CoordSystem** This element defines the coordinate system for jogging.

The following values are possible for Robot Control V1:

- CS_MACHINE_COORDINATES
(internally corresponds to KIN_CORD_SYS_ORIGIN_COORD)
- CS_AXES_COORDINATES
(internally corresponds to KIN_CORD_SYS_JOINT_COORD)
- PRODUCT_COORDINATES_1
(internally corresponds to KIN_CORD_SYS_WORKPIECE_COORD)
- CS_GRIPPER_COORDINATES
(internally corresponds to KIN_CORD_SYS_GRIPPER_COORD)

The following values are possible for Robot Control V2:

- CS_MACHINE_COORDINATES
- CS_AXES_COORDINATES
- CS_BASE_COORDINATES
- CS_PRODUCT_COORDINATES_1
- CS_PRODUCT_COORDINATES_2
- ...
- CS_PRODUCT_COORDINATES_16

JogMode	This element defines the jogging type (small/large incremental motion, slow/fast continuous motion).
JogPlus	This element contains bits for all kinematics axes. Setting the respective bit triggers a traversing motion in positive direction according to the specifications.
JogMinus	This element contains bits for all kinematics axes. Setting the respective bit triggers a traversing motion in negative direction according to the specifications.
DistanceSmall	This element contains the short distance for incremental jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0080 and K-0-0080.
DistanceLarge	This element contains the large distance for incremental jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0081 and K-0-0081.
VelocitySlow	This element contains the slow velocity for continuous jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0082 and K-0-0082.
VelocityFast	This element contains the fast velocity for continuous jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0083 and K-0-0083.
Acceleration	This element contains the acceleration for incremental and continuous jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0084 and K-0-0084.
Deceleration	This element contains the deceleration for incremental and continuous jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0085 and K-0-0085.
Jerk	This element contains the maximum acceleration and deceleration jerk for incremental and continuous jogging in the axis and Cartesian coordinate system. If Robot Control V1 is used, this value is written to the parameters A-0-0086/87 and K-0-0086/87.
Library	ML_KinTech.compiled-library

Kinematics Interface

System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_COORDINATED

Brief description The MB_KINEMATICS_COORDINATED structure is a substructure of MB_KINEMATICS_CONTROL_TYPE01 and contains control variables and command values for the positioning modes.

The elements contained in this structure correspond to the inputs of the function blocks:

- ML_MoveLinearAbsolute
- ML_MoveLinearRelative
- ML_MoveDirectAbsolute
- ML_MoveDirectRelative
- ML_MoveCircularAbsolute
- ML_MoveCircularRelative

For more detailed information, see the respective documentation.

Kinematics Interface

The following figure shows the MB_KINEMATICS_COORDINATED structure:

MB_KINEMATICS_COORDINATED	
1	TYPE MB_KINEMATICS_COORDINATED:
2	STRUCT
3	(* position for an absolute move (Unit: e.g. in [degree] depending on
4	parametrization) (negative or positive)
5	- start moving with positionings opmodes
6	Note: cyclic checked AND transmitted on change (EnableCyclicScanning = TRUE)
7	-----
8	Zielposition, absolut oder relativ (Einheit: z.B. in [Grad] je nach Parametrierung)
9	(negativ oder positiv)
10	- Freigabe mit den Positinierbetriebsarten
11	Hinweis: wird permanent zyklisch verarbeitet (EnableCyclicScanning = TRUE) *)
12	Point: ARRAY [1..16] OF REAL := [16(0.0)];
13	-----
14	(* maximum velocity (Unit: e.g. in [rpm] depending on parametrization)
15	(always positive)
16	- start moving with positionings opmodes
17	-----
18	Positioniergeschwindigkeit, (Einheit: z.B. in [U/min] je nach Parametrierung)
19	(positiv eingeben)
20	- Freigabe mit Positinierbetriebsarten *)
21	Velocity: REAL := 10.0;
22	-----
23	(* Value of the acceleration (Unit: e.g. in [rad/s ²] depending on
24	parametrization) (always positive)
25	- start moving with positionings opmodes
26	-----
27	Beschleunigung, (Einheit: z.B. in [rad/s ²] je nach Parametrierung) (immer positiv)
28	- Freigabe mit Positinierbetriebsarten *)
29	Acceleration: REAL := 10.0;
30	-----
31	(* Value of the deceleration (Unit: e.g. in [rad/s ²] depending on
32	parametrization) (always positive)
33	- start moving with Positinierbetriebsarten
34	-----
35	Verzögerung, (Einheit: z.B. in [rad/s ²] je nach Parametrierung) (immer positiv)
36	- Freigabe mit Positinierbetriebsarten *)
37	Deceleration: REAL := 10.0;
38	-----
39	(* Value of the jerk (Unit: e.g. in [rad/s ³] depending on parametrization)
40	(always positive)
41	- start moving with positionings opmodes
42	-----
43	Ruck, (Einheit: z.B. in [rad/s ³] je nach Parametrierung) (immer positiv)
44	- Freigabe mit Positinierbetriebsarten *)
45	Jerk: REAL := 100.0;
46	-----
47	(* defines in which coordinate system the points have to be specified
48	(machine coordinates or joint coordinates)
49	- start moving with positionings opmodes
50	-----
51	Koordinatensystem (Maschinen- oder Achs-Koordinaten)
52	- Freigabe mit Positinierbetriebsarten *)
53	CoordSystem: ML_COORDSYS := CS_MACHINE_COORDINATES;
54	-----
55	(* Define the slope type of the motion (BlockSlope, ProgramSlope or ContinuousSlope)
56	- start moving with positionings opmodes
57	-----
58	Slope-Art der Bewegung (BlockSlope, ProgramSlope or ContinuousSlope)
59	- Freigabe mit Positinierbetriebsarten *)
60	SlopeType: ML_SLOPE_TYPE := ST_BLOCK_SLOPE;
61	-----
62	(* Defines whether the motion ends at the end point or is blended (to or via)
63	- start moving with positionings opmodes
64	-----
65	definiert ob die Bewegung am Endpunkt endet oder überschliffen wird (to oder via)
66	- Freigabe mit Positinierbetriebsarten *)
67	SetMode: ML_SET_MODE := SM_T0;
68	-----
69	(* Defines the blending radius (Unit: [%] from the parameter A-0-0230)
70	- start moving with positionings opmodes
71	-----
72	Überschleifradius (Unit: [%] vom Parameter A-0-0230)
73	- Freigabe mit Positinierbetriebsarten *)
74	BlendingRadius: REAL := 0.0;
75	-----
76	(* position for an circular move (Unit: e.g. in [degree] depending on
77	parametrization) (negative or positive)
78	- start moving with circular positionings opmodes
79	-----
80	Zielposition, zirkular (Einheit: z.B. in [Grad] je nach Parametrierung)
81	(negativ oder positiv)
82	- Freigabe mit Positinierbetriebsarten "Zirkular" *)
83	CircularAuxPoint: ARRAY [1..16] OF REAL := [16(0.0)];
84	-----

Fig.6-12: MB_KINEMATICS_COORDINATED (part 1)

Kinematics Interface

```
85      (* Defines the interpretation of the "AuxPoint" input
86      - start moving with circular positionings opmodes
87      -----
88      Definiert die Interpretation des Eingangs "AuxPoint"
89      - Freigabe mit Positinerbetriebsart "Zircular" *)
90      CircularCircMode: ML_CIRCmodes := CM_BOARDER;
91  END_STRUCT
92  END_TYPE
```

Fig.6-13: MB_KINEMATICS_COORDINATED (part 2)

Library ML_KinTech.compiled-library
System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_RCL

Brief description The MB_KINEMATICS_RCL structure is a substructure of MB_KINEMATICS_CONTROL_TYPE01 and contains control variables and command values for the positioning modes.

The elements contained in this structure correspond to the inputs of the function block "MB_IRDProgSelect". For more detailed information, see the respective documentation.

 Only permissible for Robot Control V1.

The following figure shows the MB_KINEMATICS_RCL structure:

```
MB_KINEMATICS_RCL
1  TYPE MB_KINEMATICS_RCL :
2  STRUCT
3      (* Name of the IRD program without extension
4      -----
5      Name des IRD-Programms ohne Dateinamen-Erweiterung *)
6      ProgName: STRING[12] := '';
7  END_STRUCT
8  END_TYPE
```

Fig.6-14: MB_KINEMATICS_COORDINATED

Library ML_KinTech.compiled-library
System IndraMotion MLC

- [MB_KINEMATICS_CONTROL_TYPE01, Overview on page 256](#)

MB_KINEMATICS_STATUS_TYPE01

MB_KINEMATICS_STATUS_TYPE01, Overview

MB_KINEMATICS_STATUS_TYPE01 on page 273	
MB_KINEMATICS_ADMIN_STATUS on page 273	
MB_KINEMATICS_RCL_STATUS on page 276	
MB_KINEMATICS_DIAGNOSIS on page 278	
MB_KINEMATICS_SETUP_STATUS on page 280	

Tab.6-4: MB_KINEMATICS_STATUS_TYPE01, structural setup

MB_KINEMATICS_STATUS_TYPE01

Brief description The arKinStatus_gb[] instance accesses the MB_KINEMATICS_STATUS_TYPE01 structure.

It contains the following elements:

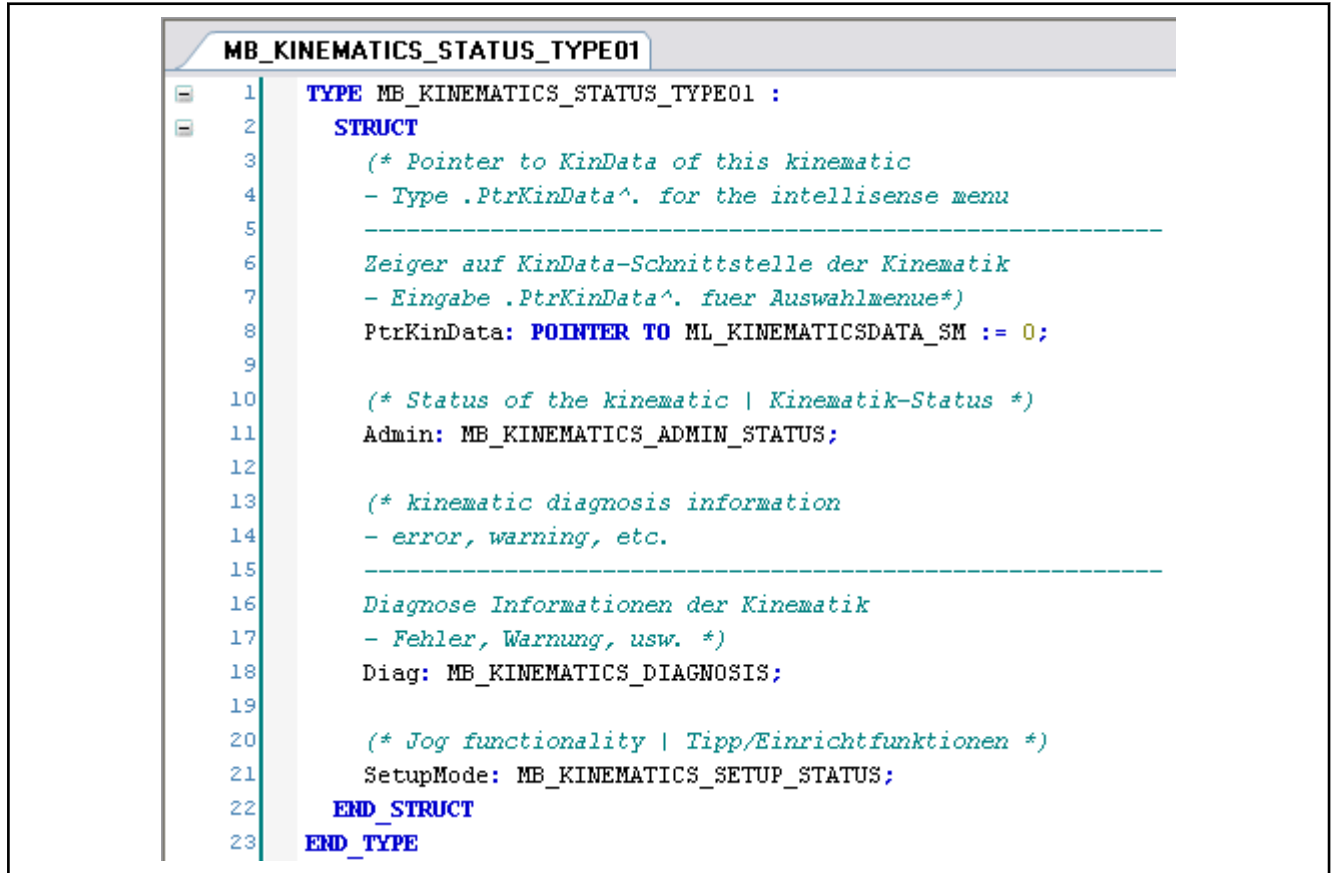


Fig. 6-15: MB_KINEMATICS_STATUS_TYPE01

PtrKinData	Access to the KinData interface of the kinematics
Admin	Access to status attributes of the kinematics management
Diag	Access to diagnostic outputs
SetupMode	Access to status attributes of the jog/setup function
Library	ML_KinTech.compiled-library
System	IndraMotion MLC
	• MB_KINEMATICS_STATUS_TYPE01, Overview on page 272

MB_KINEMATICS_ADMIN_STATUS

Brief description The MB_KINEMATICS_ADMIN_STATUS structure is a substructure of MB_KINEMATICS_STATUS_TYPE01 and contains the output status for kinematics management.

Kinematics Interface

The following figure shows the MB_KINEMATICS_ADMIN_STATUS structure:



Fig.6-16: MB_KINEMATICS_ADMIN_STATUS

Kinematics Interface

_OpModeAck	This element returns the operation mode status.
MODE_COORD_AB ... AH	These elements solely serve as placeholders and are only permissible in the control structure.
MODE_COORD_HOMING ... RCL	These elements return the status of the operation mode in bits. If none of the bits is set, this indicates the status "ModeCoordUngrouped".
Active	This element indicates, if the structures arKinCtrl_gb[] and arKinStatus_gb[] are valid (= TRUE) or not (= FALSE). It is set during initialization of the kinematics interface.
Name	This element indicates the kinematics names. It is set during initialization of the kinematics interface.
TrafoNb	This element indicates the transformation number of the kinematics. It is set during initialization of the kinematics interface.
CmdDone	This element indicates if the command issued last was completed successfully.
 <i>Only valid for the following operation modes</i>	
• ModeCoordContinue • ModeCoordInterrupted • ModeCoordPosDirAbs • ModeCoordPosDirRel • ModeCoordPosLinAbs • ModeCoordPosLinRel • ModeCoordPosCircAbs • ModeCoordPosCircRel • ModeCoordStopping • ModeCoordRCL	
InBuffer	This element indicates if the command issued last was successfully entered into the block buffer.
 <i>Only valid for the following operation modes</i>	
• ModeCoordContinue • ModeCoordInterrupted • ModeCoordPosDirAbs • ModeCoordPosDirRel • ModeCoordPosLinAbs • ModeCoordPosLinRel • ModeCoordPosCircAbs • ModeCoordPosCircRel • ModeCoordStopping	
ParamLevelAck	This element indicates if all axes belonging to the kinematics are in parameterization mode 1.
SetConveyorBeltDone	This element indicates if the configuration of the belt was completed successfully.
KinTransformPLCDone	This element indicates if the user-defined transformation was registered successfully.
RCL	Access to status attributes for handling RCL points and programs

Kinematics Interface

For details, refer to [MB_KINEMATICS_RCL_STATUS](#) on page 276

Library ML_KinTech.compiled-library

System IndraMotion MLC

- [MB_KINEMATICS_STATUS_TYPE01](#), Overview on page 272

MB_KINEMATICS_RCL_STATUS

Brief description MB_KINEMATICS_RCL_STATUS represents the status structures of MB_KINEMATICS_RCL_POINT and MB_KINEMATICS_RCL_PROG.

MB_KINEMATICS_RCL_STATUS	
1	TYPE MB_KINEMATICS_RCL_STATUS :
2	STRUCT
3	<i>(* Status of the latest handling of RCL points</i>
4	<i>- TRUE: processing was successful</i>
5	<i>- FALSE: processing failed OR was not startet</i>
6	-----
7	<i>Status der letzten Handhabung von RCL-Punkten</i>
8	<i>- TRUE: Bearbeitung war erfolgreich</i>
9	<i>- FALSE: Bearbeitung ist fehlgeschlagen oder wurde</i>
10	<i>nicht gestartet *)</i>
11	PointModeDone: BOOL := FALSE ;
12	
13	<i>(* Shows the latest read points list</i>
14	-----
15	<i>Zeigt die zuletzt gelesene Punktliste *)</i>
16	PointList: ARRAY [1..16] OF ML_ROCO_POINT;
17	
18	<i>(* Shows the latest read program list</i>
19	-----
20	<i>Zeigt die zuletzt gelesene Programmliste *)</i>
21	ProgList: ARRAY [1..16] OF ML_ROCO_FILE_DIR;
22	
23	<i>(* Shows the latest read point</i>
24	-----
25	<i>Zeigt den zuletzt gelesenen Punkte *)</i>
26	Point: ML_ROCO_POINT;
27	
28	<i>(* Status of K-0-0220 and K-0-0221</i>
29	<i>- TRUE: set was successful</i>
30	<i>- FALSE: set failed OR was not startet</i>
31	-----
32	<i>Status des Setzens der Parameter K-0-0220 und</i>
33	<i>K-0-0221</i>
34	<i>- TRUE: Setzen war erfolgreich</i>
35	<i>- FALSE: Setzen ist fehlgeschlagen oder wurde nicht</i>
36	<i>gestartet *)</i>
37	ProgNameAck: BOOL := FALSE ;
38	
39	<i>(* Status of the copying from the Compact Flash card</i>
40	<i>to the control's internal memory</i>
41	<i>- TRUE: copying was successful</i>
42	<i>- FALSE: copying failed OR was not startet</i>
43	-----
44	<i>Status des Kopiervorgangs eines RCL-Programms von</i>
45	<i>der Compact Flash Karte auf den internen Speicher</i>
46	<i>der Steuerung</i>
47	<i>- TRUE: Kopiervorgang war erfolgreich</i>
48	<i>- FALSE: Kopiervorgang ist fehlgeschlagen oder wurde nicht</i>
49	<i>gestartet *)</i>
50	ProgLoadAck: BOOL := FALSE ;
51	

Fig. 6-17: MB_KINEMATICS_RCL_STATUS data types (part 1)

Kinematics Interface

```

52      (* Status of the copying from the control's internal
53      memory to the Compact Flash card
54      - TRUE: copying was successful
55      - FALSE: copying failed OR was not startet
56      -----
57      Status des Kopiervorgangs eines RCL-Programms vom
58      internen Speicher der Steuerung auf der Compact Flash
59      Karte
60      - TRUE: Kopiervorgang war erfolgreich
61      - FALSE: Kopiervorgang ist fehlgeschlagen oder wurde nicht
62      gestartet *)
63      ProgSaveAck: BOOL := FALSE;
64
65      (* Status of the compile process of a RCL program
66      - TRUE: compile process was successful
67      - FALSE: compile process failed OR was not startet
68      -----
69      Status des Kompiliervorgangs eines RCL-Programms
70      - TRUE: Kompiliervorgangs war erfolgreich
71      - FALSE: Kompiliervorgangs ist fehlgeschlagen oder wurde nicht
72      gestartet *)
73      ProgCompileAck: BOOL := FALSE;
74      END_STRUCT
75      END_TYPE

```

Fig.6-18: MB_KINEMATICS_RCL_STATUS data types (part 2)

PointModeDone	This element indicates if the last command to write or read a point or a point/program list was successful (= TRUE) or not (= FALSE).
PointList	This element displays the point list read last.
ProgList	This element displays the program list read last.
Point	This element displays the point read last.
ProgNameAck	This element indicates whether writing the name specified in ar-KinCtrl_gb[].RCLMode.ProgName was successful (= TRUE) or not (= FALSE).
ProgLoadAck	This element indicates whether loading the program specified in ar-KinCtrl_gb[].RCLMode.ProgName to the internal memory of the control was successful (= TRUE) or not (= FALSE).
ProgSaveAck	This element indicates whether saving the program specified in ar-KinCtrl_gb[].RCLMode.ProgName to the Compact Flash card was successful (= TRUE) or not (= FALSE).
ProgCompileAck	This element indicates whether compiling the program specified in ar-KinCtrl_gb[].RCLMode.ProgName in the control was successful (= TRUE) or not (= FALSE).
Library	ML_KinTech.compiled-library
System	IndraMotion MLC
	• MB_KINEMATICS_STATUS_TYPE01, Overview on page 272
MB_KINEMATICS_DIAGNOSIS	
Brief description	The MB_KINEMATICS_DIAGNOSIS structure is a substructure of MB_KINEMATICS_STATUS_TYPE01 and contains warning and error outputs.

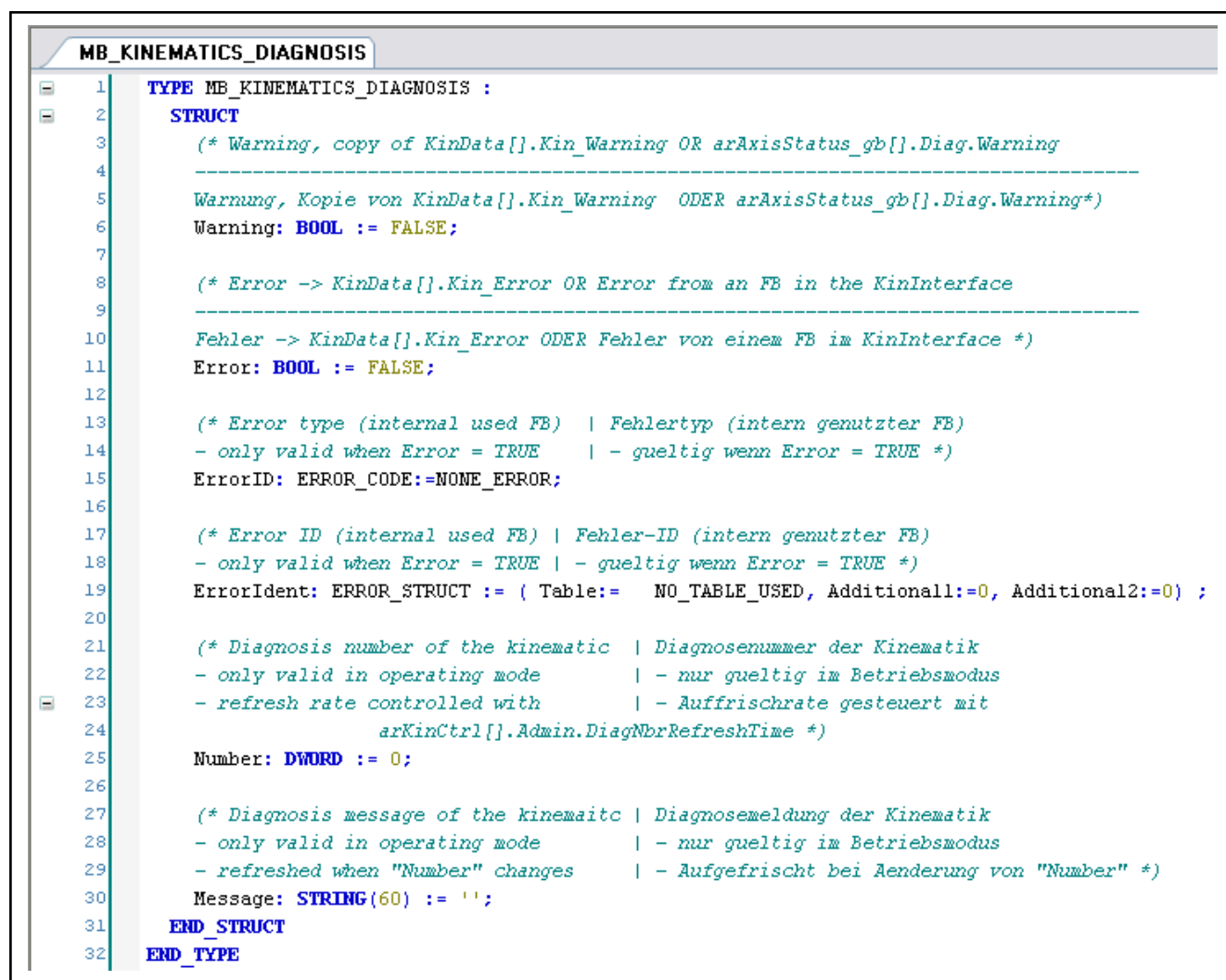


Fig.6-19: MB_KINEMATICS_DIAGNOSIS

arKinCtrl_gb[].Admin.DiagNbrRefreshTime controls the update rate of the element "Number". As soon as element "Number" changes, the text "Message" is read again. arKinCtrl_gb[].Admin.DiagNbrRefreshTime also turns off reading diagnostic information (DiagNbrRefreshTime := T#0ms).

Warning	This element is a copy of KinData[].Kin_Warning or arAxisStatus_gb[].Diag.Warning and indicates that the kinematics or its axes report a warning.
Error	This element indicates that the kinematics, its axes or an internal function block of the kinematics interface report an error.
ErrorID	This element returns the error type of the kinematics, one of its axes or an internal function block.
ErrorID	This element returns the error ID of the kinematics, one of its axes or an internal function block.
Number	This element returns the diagnostic number of the kinematics, one of its axes or an internal function block.
Message	This element returns the diagnostic message of the kinematics, one of its axes or an internal function block.
Library	ML_KinTech.compiled-library

Kinematics Interface

- SystemIndraMotion MLC
- [MB_KINEMATICS_STATUS_TYPE01, Overview on page 272](#)

MB_KINEMATICS_SETUP_STATUS

Brief description

The MB_KINEMATICS_SETUP_STATUS structure is a substructure of MB_KINEMATICS_STATUS_TYPE01 and contains control variables to jog kinematics.

```
MB_KINEMATICS_SETUP_STATUS
1  TYPE MB_KINEMATICS_SETUP_STATUS :
2  STRUCT
3      (*Status enable gog mode | Status der Tipp-/Einrichtbetriebsfreigabe*)
4      EnableAck: BOOL := FALSE;
5  END_STRUCT
6  END_TYPE
```

Fig.6-20: MB_KINEMATICS_SETUP_STATUS

- EnableAck
- This element indicates whether the jog/setup mode is active (= TRUE) or not (= FALSE).
- Library
- ML_KinTech.compiled-library
- System
- IndraMotion MLC
- [MB_KINEMATICS_STATUS_TYPE01, Overview on page 272](#)

6.2.3 Kinematics Interface Function Blocks

ML_KinematicsInitType01

Brief description

The function block ML_KinematicsInitType01 is used to initialize the kinematics interface for all kinematics. The function block only needs to be called once when the program is started and/or each time the phase is switched from the parameterization mode to the operating mode.

Interface description

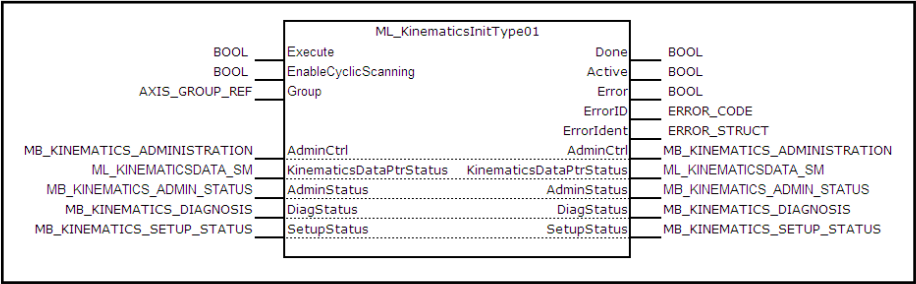


Fig.6-21: Function block ML_KinematicsInitType01

I/O type	Name	Data type	Comment
VAR_INPUT	Execute	BOOL	Starting initialization with a positive edge
	EnableCyclicScanning	BOOL	Cyclic testing and transmission of the changed inputs for the current operation mode
	Group	BOOL	Kinematics to be initialized in the kinematics interface

Kinematics Interface

I/O type	Name	Data type	Comment
VAR_OUTPUT	Done	BOOL	Set when the function block has completed processing.
	Active	BOOL	Set when the function block is active (not in idle mode).
	Error	BOOL	It indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Short note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed information on the error
VAR_IN_OUT	AdminCtrl	MB_KINEMATICS_ADMINISTRATION	Axis management arKinCtrl_gb[].Admin
	KinematicsDataPtrStatus	ML_KINEMATICS_DATA_SM	Pointer to KinData of this kinematics arKinStatus_gb[].PtrAxisData
	AdminStatus	MB_KINEMATICS_ADMIN_STATUS	Status management of the kinematics arKinStatus_gb[].Admin
	DiagStatus	MB_KINEMATICS_DIAGNOSIS	Kinematics diagnostic information arKinStatus_gb[].Diag
	SetupStatus	MB_KINEMATICS_SETUP_STATUS	Status of the kinematics jog/setup function arKinStatus_gb[].SetupMode

Tab.6-5: ML_KinematicsInitType01 interface variables



It is not possible to transfer the complete structures "MB_Kinematics_Control_Type01" and "MB_Kinematics_Status_Type01" to the function block instead of the individual structure elements "AdminCtrl", "KinematicsDataPtrStatus", etc. Therefore, no specific extensions to the existing code may be added in the user application. As a consequence, the required elements of "MB_Kinematics_Control_Type01" and "MB_Kinematics_Status_Type01" are transferred as separate inputs.

For more detailed information, see [chapter 6.2.7 "Kinematics Interface User Extension" on page 313](#).



The function block ML_KinematicsInitType01 may only be used on the IndraMotion MLC control.

Functional Description

The function block ML_KinematicsInitType01 initializes the following structure elements with default values:

Structure element	Default
AdminCtrl._OpMode	ModeCoordAB
AdminCtrl.ClearError	FALSE
AdminCtrl.EnableCyclicScanning	EnableCyclicScanning input
KinematicsDataPtrStatus	ADR(KinData[Group.GroupNo])
AdminCtrl.Group	Input Group
AdminStatus._OpModeAck	ModeCoordUngrouped

Kinematics Interface

Structure element	Default
AdminStatus.Active	FALSE
AdminStatus.Name	' '
AdminStatus.TrafoNb	999
AdminStatus.CmdDone	FALSE
AdminStatus.InBuffer	FALSE
AdminStatus.ParamLevelAck	FALSE
AdminStatus.SetConveyorBeltDone[1]	FALSE
AdminStatus.SetConveyorBeltDone[2]	FALSE
AdminStatus.SetConveyorBeltDone[3]	FALSE
AdminStatus.SetConveyorBeltDone[4]	FALSE
AdminStatus.KinTransformPLCDone	FALSE
AdminStatus.RCL.PointModeDone	FALSE
AdminStatus.RCL.ProgNameAck	FALSE
AdminStatus.RCL.ProgLoadAck	FALSE
AdminStatus.RCL.ProgSaveAck	FALSE
AdminStatus.RCL.ProgCompileAck	FALSE
DiagStatus.Error	FALSE
DiagStatus.ErrorID	NONE_ERROR
DiagStatus.ErrorIdent.Table	NO_TABLE_USED
DiagStatus.ErrorIdent.Additional1	0
DiagStatus.ErrorIdent.Additional2	0
DiagStatus.Number	0
DiagStatus.Message	'reading not activated'
SetupStatus.EnableAck	FALSE

Tab.6-6: Structure elements initialized by ML_KinematicsInitType01

The function block ML_KinematicsInitType01 reads the following parameters and writes the values to the following structure elements:

K-parameters (name)	Structure element
K-0-0031 (Transformation scheme)	AdminStatus.TrafoNb
K-0-0002 (kinematics name)	AdminStatus.Name
K-0-0008 (axis configuration list)	only required for internal purposes

Tab.6-7: Parameters read by ML_KinematicsInitType01

EnableCyclicScanning: The "EnableCyclicScanning" input enables cyclic testing and transmission of modified inputs for the current operation mode. This is relevant for values in arKinCtrl_gb[] with the comment: "Cyclically tested and transmitted in case of change." The default value is TRUE.

Group: The "Group" input indicates which kinematics is to be initialized.

Error handling The function block generates the following error messages in DiagStatus.ErrorIdent.Additional1/Additional2 for the table "F_RELATED_TABLE", 16#0170:

ErrorID	Additional1	Additional2	Description
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Invalid function block state
INPUT_INVALID_ERROR, (16#0001)	16#2600	16#0000	invalid kinematics number "Group.GroupNo"
INPUT_INVALID_ERROR, (16#0001)	16#2600	16#0001	Invalid control number "Group.CntrlNo". It has to be set to LOCAL_CNTRL

Tab.6-8: ML_KinematicsInitType01 error codes

ML_KinematicsInterfaceType01

Brief description The function block **ML_KinematicsInterfaceType01** is used to configure the kinematics interface for all kinematics. This function block has to be called cyclically as long as the user is in the operating mode.

Interface description

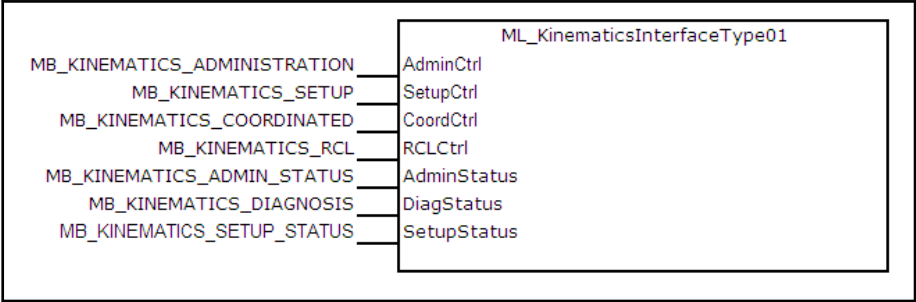


Fig.6-22: Function block ML_KinematicsInterfaceType01

Kinematics Interface

I/O type	Name	Data type	Comment
VAR_INPUT	AdminCtrl	REFERENCE TO MB_KINEMATICS_ADMINISTRATION	Axis management arKinCtrl_gb[].Admin
	SetupCtrl	REFERENCE TO MB_KINEMATICS_SETUP	Jog/setup operation arKinCtrl_gb[].SetupMode
	CoordCtrl	REFERENCE TO MB_KINEMATICS_COORDINATED	Operation mode coordinated motion arKinCtrl_gb[].CoordMode
	RCLCtrl	REFERENCE TO MB_KINEMATICS_RCL	Operation modes RCL arKinCtrl_gb[].RCLMode
	AdminStatus	REFERENCE TO MB_KINEMATICS_ADMIN_STATUS	Status management of the axis arKinStatus_gb[].Admin
	DiagStatus	REFERENCE TO MB_KINEMATICS_DIAGNOSIS	Status diagnostics of the axis arKinStatus_gb[].Diag
	SetupStatus	REFERENCE TO MB_KINEMATICS_SETUP_STATUS	Jog/setup operation status arKinStatus_gb[].SetupMode

Tab.6-9: ML_KinematicsInterfaceType01 interface variables



It is not possible to transfer the complete structures "MB_Kinematics_Control_Type01" and "MB_Kinematics_Status_Type01" to the function block instead of the individual structure elements "AdminCtrl", "SetupCtrl", etc. Therefore, no specific extensions to the existing code may be added in the user application. As a consequence, the required elements of "MB_Kinematics_Control_Type01" and "MB_Kinematics_Status_Type01" are transferred as separate inputs.

For more detailed information, see [chapter 6.2.7 "Kinematics Interface User Extension" on page 313](#).

To optimize performance, the structures are connected with REFERENCE TO instead of as VAR_IN_OUT as VAR_INPUT. In this way, the inputs only have to be initialized once. In this way, if the function block is called cyclically, the structures do not have to be transferred.



The function block ML_KinematicsInterfaceType01 may only be used on the IndraMotion MLC control.

Error handling

The function block generates the following error messages in DiagStatus.ErrorIdent.Additional1/Additional2 for the table "F_RELATED_TABLE", 16#0170:

ErrorID	Additional1	Additional2	Description
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0000	Invalid internal state
STATE_MACHINE_ERROR, (16#0005)	16#0006	16#0001	Invalid function block state

Kinematics Interface

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR, (16#0003)	16#2600	16#0001	Invalid control number "Group.CntrlNo". It has to be set to LOCAL_CNTRL
DEVICE_ERROR, (16#0008)	16#2601	16#0000	Kinematics reports an error in the KinData structure
INPUT_INVALID_ERROR, (16#0001)	16#2602	Number of Op-Mode	Invalid operation mode selected
INPUT_INVALID_ERROR, (16#0001)	16#2603	Number of Op-Mode	Invalid, unsupported operation mode enabled
INPUT_RANGE_ERROR, (16#0006)	16#2604	16#0000	At least one input frequency (AdminCtrl, AdminStatus, etc.) is not set
ACCESS_ERROR, (16#0004)	16#2605	16#0000	It is not possible and necessary to reference the axis/axes
ACCESS_ERROR, (16#0004)	16#2605	Number of the axis to be referenced	It is not possible to reference the axis/axes
RESOURCE_ERROR, (16#0003)	16#2606	16#0000	Operation mode was changed during setup mode
OTHER_ERROR, (16#7FFE)	16#2607	16#0000	It was not possible to group kinematics
INPUT_INVALID_ERROR, (16#0001)	16#2609	16#0000	Invalid coordinate system selected
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	Invalid Robot Control version

Tab.6-10: ML_KinematicsInterfaceType01 error codes

ML_KinJogV1

Brief description The function block **ML_KinJogV1** is used to jog individual axes or axis groups of a kinematics. This function block has to be called cyclically.



Only permissible for Robot Control V1.

Interface description

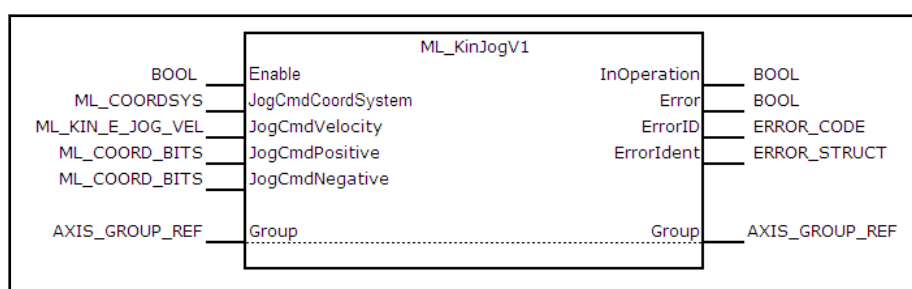


Fig.6-23: ML_KinJogV1 function block

Kinematics Interface

I/O type	Name	Data type	Comment
VAR_INPUT	Enable	BOOL	Processing enabled for the function block
	JogCmdCoordSystem	ML_COORDSYS	Specifying the jogging coordinate system
	JogCmdVelocity	ML_KIN_E_JOG_VEL	Specifying the jogging velocity
	JogCmdPositive	ML_COORD_BITS	Contains one bit per coordinate with which the coordinates can be jogged in the positive direction
	JogCmdNegative	ML_COORD_BITS	Contains one bit per coordinate with which the coordinates can be jogged in the negative direction
VAR_OUTPUT	InOperation	BOOL	The function blocks is busy with its actual task
	Error	BOOL	It indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Short note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed information on the error
VAR_IN_OUT	Group	BOOL	Reference to the kinematics to be jogged

Tab.6-11: ML_KinJogV1 interface variables

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
JogCmdCoordSystem	ML_COORDSYS			KIN_CORD_SYS_JOINT_COORD	Continuous
JogCmdVelocity	ML_KIN_E_JOG_VEL			KIN_JOG_STOP	Continuous
JogCmdPositive	ML_COORD_BITS			16(FALSE)	Continuous
JogCmdNegative	ML_COORD_BITS			16(FALSE)	Continuous

Tab.6-12: ML_KinJogV1 input behavior



The function block ML_KinJogV1 may only be used on the IndraMotion MLC control.

Functional description	With the ML_KinJog function block it is possible to jog a kinematics with Robot Control V1. Jogging is carried out via the KinData interface KinData[].stJogCmd_q. If the function block was started successfully, it automatically groups the axes belonging to the kinematics.
JogCmdCoordSystem	This input indicates the coordinate system where jogging is to take place. <i>The input can assume the following values:</i> • KIN_CORD_SYS_JOINT_COORD: Axis coordinate system • KIN_CORD_SYS_ORIGIN_COORD: World coordinate system • KIN_CORD_SYS_WORKPIECE_COORD: Workpiece coordinate system • KIN_CORD_SYS_GRIPPER_COORD: Gripper coordinate system
JogCmdVelocity	This input indicates the jogging type (incrementally slow or fast, continuously slow or fast).

The input can assume the following values:

- KIN_JOG_STOP
Jogging disabled
- KIN_JOG_STEP_SMALL
incremental jogging with small steps
- KIN_JOG_STEP_LARGE
incremental jogging with large steps
- KIN_JOG_CONT_SLOW
continuous jogging with slow velocity
- KIN_JOG_CONT_LARGE
continuous jogging with fast velocity

JogCmdPositive This input enables the traversing motion in positive direction for the respective axis.

JogCmdNegative This input enables the traversing motion in negative direction for the respective axis.

Error handling The function block generates the following error messages in ErrorIdent. Additional1/Additional2 using the table "F_RELATED_TABLE", 16#0170:

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	Invalid Robot Control version

Tab.6-13: ML_KinJogV1 error codes

ML_KinJog

Brief description The function block **ML_KinJog** is used to jog individual axes or axis groups of a kinematics. This function block has to be called cyclically.



Only permissible for Robot Control V2.

Interface description

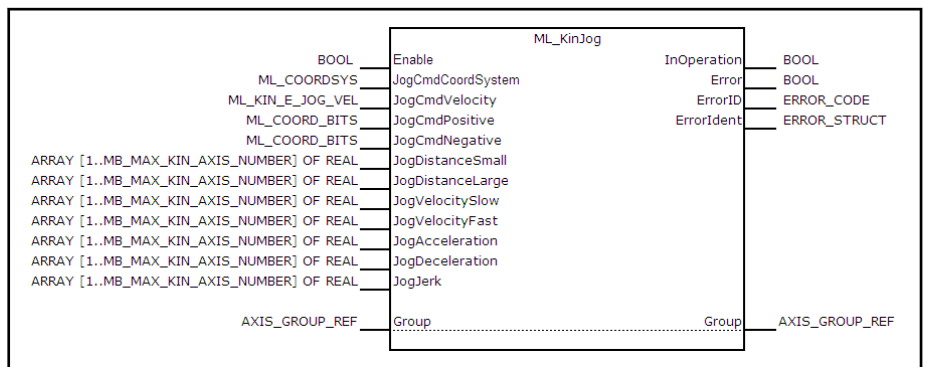


Fig.6-24: ML_KinJog function block

Kinematics Interface

I/O type	Name	Data type	Comment
VAR_INPUT	Enable	BOOL	Processing enabled for the function block
	JogCmdCoordSystem	ML_COORDSYS	Specifying the jogging coordinate system
	JogCmdVelocity	ML_KIN_E_JOG_VEL	Specifying the jogging velocity
	JogCmdPositive	ML_COORD_BITS	Contains one bit per coordinate with which the coordinates can be jogged in the positive direction
	JogCmdNegative	ML_COORD_BITS	Contains one bit per coordinate with which the coordinates can be jogged in the negative direction
	JogDistanceSmall	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the short jogging distance for every axis
	JogDistanceLarge	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the long jogging distance for every axis
	JogVelocitySlow	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the slow jogging velocity for every axis
	JogVelocityFast	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the fast jogging velocity for every axis
	JogAcceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the acceleration for the jogging mode for every axis
	JogDeceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the deceleration for the jogging mode for every axis
	JogJerk	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specifies the jerk for the jogging mode for every axis
VAR_OUTPUT	InOperation	BOOL	The function block is busy with its actual task
	Error	BOOL	It indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Short note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed information on the error
VAR_IN_OUT	Group	BOOL	Reference to the kinematics to be jogged

Tab.6-14: ML_KinJog interface variables

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
JogCmdCoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Continuous

Kinematics Interface

Name	Type	Min. value	Max. value	Default value	Effective
JogCmdVelocity	ML_KIN_E_JOG_VEL			KIN_JOG_STOP	Continuous
JogCmdPositive	ML_COORD_BITS			16(FALSE)	Continuous
JogCmdNegative	ML_COORD_BITS			16(FALSE)	Continuous
JogDistanceSmall	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		1.0	Continuous
JogDistanceLarge	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		10.0	Continuous
JogVelocitySlow	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		10.0	Continuous
JogVelocityFast	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		25.0	Continuous
JogAcceleration	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		370.0	Continuous
JogDeceleration	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		370.0	Continuous
JogJerk	ARRAY [1..MB_MAX_KIN_NUMBER] OF REAL	>0.0		0.0	Continuous

Tab.6-15: ML_KinJog input behavior



The function block ML_KinJog may only be used on the IndraMotion MLC control.

- Functional description** With the ML_KinJog function block it is possible to jog a kinematics with Robot Control V2. Jogging in a Cartesian coordinate system corresponds to a linear traversing motion while jogging in the axis coordinate system corresponds to a PTP traversing motion. If the function block was started successfully, it automatically groups the axes belonging to the kinematics.
- JogCmdCoordSystem** This input indicates the coordinate system where jogging is to take place.
The input can assume the following values:
- CS_MACHINE_COORDINATES: World coordinate system
 - CS_AXES_COORDINATES: Axis coordinate system
 - CS_BASE_COORDINATES: Base coordinate system
 - CS_PRODUCT_COORDINATES_1: Product coordinate system 1
 - CS_PRODUCT_COORDINATES_2: Product coordinate system 2
 - ...
 - CS_PRODUCT_COORDINATES_16: Product coordinate system 16
- JogCmdVelocity** This input indicates the jogging type (incrementally small or large, continuously slow or fast).

Kinematics Interface

The input can assume the following values:

- KIN_JOG_STOP
Jogging disabled
- KIN_JOG_STEP_SMALL
incremental jogging with small steps
- KIN_JOG_STEP_LARGE
incremental jogging with large steps
- KIN_JOG_CONT_SLOW
continuous jogging with slow velocity
- KIN_JOG_CONT_LARGE
continuous jogging with fast velocity

JogCmdPositive This input enables the traversing motion in positive direction for the respective axis.

JogCmdNegative This input enables the traversing motion in negative direction for the respective axis.

JogDistanceSmall, JogDistanceLarge, JogVelocitySlow, JogVelocityFast, JogAcceleration, JogDeceleration, JogJerk These inputs define the jogging distance, jogging velocity, jogging acceleration and the jogging jerk for incremental and continuous jogging. Their units depend on "JogCmdCoordSystem" and correspond to the kinematics units (in case of a Cartesian coordinate system) or the axis units (in case of the axis coordinate system).



Continuous jogging is based on the issuance of several traversing commands while the respective bit of the input "JogCmdPositive" or "JogCmdNegative" is set. If the kinematics moves to the border of the working area, a premature error message may be displayed, stating that the commanded point can not be reached by the kinematics. In this case, acknowledge the error and travel to the desired point at a lower speed or by incremental jogging.

Error handling The function block generates the following error messages in ErrorIdent.Additional1/Additional2 using the table "F_RELATED_TABLE", 16#0170:

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	Invalid Robot Control version

Tab.6-16: ML_KinJog error codes

6.2.4 Operation Modes

Overview

There are two methods to activate an operation mode:

1. Selection of Enum values

Assigning a value of TYPE MB_KINEMATICS_MODE:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAH;
```

2. Using the bit access

Setting a bit using the bit access functionality.

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AH:= TRUE;
```

Kinematics Interface



Only 1 bit at a time may be set in `arKinCtrl_gb[]`. `Admin._OpMode`. Setting several bits causes an error in the kinematics interface.

Before an operation mode can be activated, values must be assigned to every attribute. All attributes are provided with default values. Some are provided with values not equal to zero while others are defined as 0. Due to the special demands, a value has to be assigned to them.



Only the attributes (e.g. point, velocity) used or whose default value was changed need to be defined before the current change of operation mode is carried out.

The following section describes the operation modes supported by the kinematics interface and the attributes that can be assigned to them.



The kinematics interface internally uses the axis interface to carry out certain switches (e.g. grouping/degrouping all axes, referencing axes).

Kinematics Ready, Drives Ready

If this operation mode is enabled, the kinematics is degrouped and all axes belonging to it are set to AB (drive ready). In this operation mode the kinematics interface is disabled and the kinematics is in the PLCOpen state "StandStill incomplete". The following command activates the operation mode:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAB;
```

or

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AB := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	Admin.StopDeceleration	REAL	999	Yes
	Admin.StopJerk	REAL	9999	Yes
arKinStatus_gb[]	Admin.MODE_COORD_AB	BOOL		Not applicable

Tab.6-17: Kinematics ready, drives ready attributes



If the operation mode "ModeCoordAB" is selected, the kinematics interface returns the operation mode "ModeCoordUngrouped" in the status structure!

Kinematics Ready, Drives Halt

If this operation mode is enabled, the kinematics is degrouped and all axes belonging to it are set to AH (drive halt). In this operation mode the kinematics interface is disabled and the kinematics is in the PLCOpen state "StandStill incomplete". The axes are switched to the PLCOpen state "StandStill",

Kinematics Interface

i.e. they can be commanded using the axis interface or separate PLCOpen function blocks. The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordAH;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_AH:= TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	Admin.StopDeceleration	REAL	999.0	Yes
	Admin.StopJerk	REAL	9999.0	Yes
arKinStatus_gb[]	Admin.MODE_COORD_AH	BOOL		Not applicable

Tab.6-18: Kinematics ready, drives halt attributes



If the operation mode "ModeCoordAH" is selected, the kinematics interface returns the operation mode "ModeCoordUngrouped" in the status structure!

Kinematics Homing

If this operation mode is enabled, all axes belonging to the kinematics are referenced. Homing is carried out via the SetupMode of the axis interface. The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordHoming;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_HOMING := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	Admin.HomingSequence	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	MB_MAX_KIN_AXIS_NUMBER(0.0)	Yes
arKinStatus_gb[]	Admin.MODE_COORD_HOMING	BOOL		Not applicable
	Admin.HomingAck	BOOL		Not applicable

Tab.6-19: Kinematics homing attributes



If the kinematics is already grouped when the operation mode "ModeCoordHoming" is selected, renewed homing is not possible and the kinematics interface reports an error.

Absolute PTP Positioning

Activating this operation mode causes an absolute PTP motion to a previously specified target position. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosDirAbs;  
or  
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_DIR_ABS :=  
TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100	No
	CoordCoordModeCtrl.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_DIR_ABS	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-20: Absolute PTP positioning attributes

Relative PTP Positioning

Activating this operation mode causes a relative PTP motion by a previously specified position difference. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosDirRel;  
or  
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_DIR_REL :=  
TRUE;
```

Kinematics Interface

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100	No
	CoordCoordModeCtrl.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_DIR_REL	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-21: Relative PTP positioning attributes

Absolute Linear Positioning

Activating this operation mode causes an absolute linear motion to a previously specified target position. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosLinAbs;
or
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_C00RD_POS_LIN_ABS := TRUE;
```


The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_LIN_ABS	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-22: Absolute linear positioning attributes

Relative Linear Positioning

Activating this operation mode causes a relative linear motion by a previously specified position difference. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosLinRel;
or
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_LIN_REL := TRUE;
```

Kinematics Interface

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_LIN_REL	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-23: Relative linear positioning attributes

Absolute Circular Positioning

Activating this operation mode causes an absolute circular motion to a previously specified target position around a specified intermediate point. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosCircAbs;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_C00RD_POS_CIRC_ABS := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.AuxPoint	ARRAY [1..16] OF REAL	16(0.0)	No
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
	CoordMode.CircularCircMode	ML_CIRCMODES	CM_BOARDER	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_CIRC_ABS	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-24: Absolute circular positioning attributes

Relative Circular Positioning

Activating this operation mode causes a relative circular motion by a previously specified position difference and by a specified intermediate position difference. The kinematics is grouped and switches to the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosCircRel;
or
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_CIRC_REL := TRUE;
```

Kinematics Interface

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.CircularAuxPoint	ARRAY [1..16] OF REAL	16(0.0)	No
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Velocity	REAL	10.0	No
	CoordMode.Acceleration	REAL	10.0	No
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
	CoordMode.CircularCircMode	ML_CIRCMODES	CM_BOARDER	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_CIRC_REL	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-25: Relative circular positioning attributes

Kinematics Stopping

Activating this operation mode stops the kinematics while maintaining the grouping. The kinematics switches to the PLCOpen state "Stopping". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordStopping;
or
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_STOPPING := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	AdminCtrl.StopDeceleration	REAL	999.0	Yes
	AdminCtrl.StopJerk	REAL	9999.0	Yes

Kinematics Interface

Element	Name	Type	Default	Cyclically scanned
arKinStatus_gb[]	Admin.MODE_COORD_STOP-PING	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-26: Kinematics stopping attributes



The operation mode "ModeCoordStopping" is available for Robot Control V2 only. If it is selected with Robot Control V1, the kinematics interface reports an error.

"Interrupted" Operation Mode

Activating this operation mode interrupts the kinematics motion while maintaining the grouping. The kinematics remains in the PLCOpen state "Group-Moving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordInterrupted;
or
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_INTERRUPTED := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
arKinStatus_gb[]	Admin.MODE_COORD_INTERRUPTED	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-27: Operation mode "Interrupted" attributes



The operation mode "ModeCoordInterrupted" is available for Robot Control V2 only. If it is selected with Robot Control V1, the kinematics interface reports an error.

"Continue" Operation Mode

Activating this operation mode starts a motion that was previously interrupted. The kinematics is in the PLCOpen state "GroupMoving". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordContinue;
```

Kinematics Interface

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_CONTINUE :=
TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
arKinStatus_gb[]	Admin.MODE_COORD_CONTINUE	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab.6-28: Operation mode "Continue" attributes



The operation mode "ModeCoordContinue" is available for Robot Control V2 only. If it is selected with Robot Control V1, the kinematics interface reports an error.

"External Function Block" Operation Mode

Activating this operation mode groups the kinematics and waits for an external motion command. The kinematics switches to the PLCOpen state "Stand-Still Complete". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordExternalFB;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_EXTERNAL_FB :=
TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
arKinStatus_gb[]	Admin.MODE_COORD_EXTERNAL_FB	BOOL		Not applicable

Tab.6-29: Attributes, operation mode "external function block"

"RCL" Operation Mode

Activating this operation mode groups the kinematics and starts the RCL program specified in arKinCtrl_gb[].RCLMode.ProgName. The kinematics switches to the PLCOpen state "ProgrammedMotion". The following command activates the operation mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordRCL;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_RCL := TRUE;
```

The following table contains the attributes supported by this operation mode:

Element	Name	Type	Default	Cyclically scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	RCLCtrl.ProgName	STRING	' '	No
arKinStatus_gb[]	Admin.MODE_COORD_RCL	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable

Tab.6-30: Operation mode "RCL" attributes



The operation mode "ModeCoordRCL" is available for Robot Control V1 only. If it is selected with Robot Control V2, the kinematics interface reports an error.

6.2.5 Global Variables

The ML_KinTech library contains the Global Variable List **Global_MB_KinInterfaceVars**.

The library defines the following constants:

1. arK8ConfigList_gb:

This array constant contains information on the axes assigned to the kinematics:

- uiNumberOfKinAxis - total number of axes
- uiNumberOfMoveAxis - number of positioning and orientation axes
- uiNumberOfBeltAxis - number of belt axes
- arAxisType - axis types (positioning axis, orientation axis, belt axis or no axis)
- arAxisNumber - axis numbers

2. arRocoKernel_gb:

This array constant contains information on the Robot Control version of the kinematics

- 1 - Robot Control V1
- 2 - Robot Control V2

3. arpAxisCtrlAdmin_gb:

This pointer constant points to the AdminCtrl structure of the axis interface ("MB_AXIS_ADMINISTRATION"). It enables parallel access to the axis interface both from the kinematics interface and from a user application

4. arpAxisCtrlCoordMode_gb:

This pointer constant points to the CoordModeCtrl structure of the axis interface ("MB_AXIS_COORDINATED"). It enables parallel access to the axis interface both from the kinematics interface and from a user application

5. arpAxisStatusAdmin_gb:

This pointer constant points to the AdminStatus structure of the axis interface ("MB_AXIS_ADMIN_STATUS"). It enables parallel access to the

Kinematics Interface

axis interface both from the kinematics interface and from a user application

6. arpAxisStatusDiag_gb:

This pointer constant points to the DiagStatus structure of the axis interface ("MB_AXIS_DIAGNOSIS"). It enables parallel access to the axis interface both from the kinematics interface and from a user application

6.2.6 First Steps, Kinematics Interface

Overview

Like the programming template "axis interface example project" the kinematics interface provides an equivalent example project that is automatically generated when the plug-in is integrated. Please find this example project in the tab **Logic** ► **Application** ► **AxisInterface** ► **Plug-ins** ► **KinInterface** .

Program for the Kinematics Interface

The program **TE_KinInterfaceType01Prog** has the following tasks:

- Initializing the kinematics interface:**
 When reaching the parameterization mode (phase 2), the kinematics interface is initialized for all configured kinematics using the **TE_KinematicsInitAllKinematicsType01** function block. When operation mode (Phase 4) is reached, initialization is executed again in order to recognize changes in the parameterization.
- Cyclic call of the kinematics interface:**
 Following successful initialization (indicated by "TE_KinInterfaceType01Prog.InitDone") the kinematics interface is called cyclically. For this purpose, the function block "TE_KinematicsInterface" is called. It is derived from the function block "MB_KinInterfaceType01" and therefore enables user extensions.
- User extensions of the kinematics interface:**
 The function block "TE_KinematicsInterface" allows the user to add elements to the structures of the kinematics interface and thereby extend their application in an optimal way. For more details, see [chapter 6.2.7 "Kinematics Interface User Extension" on page 313](#)

Kinematics Interface Visualizations

Overview

The kinematics interface contains visualization masks to provide a ready-made, simple interface to set and control the kinematics.

It contains the following visualizations:

Visualization	Description
SystemOverview	
KinSystem_Overview_All	Complete overview of all defined kinematics, including simple diagnostics, status and coordinates
KinematicsModes	
Kinematics_Overview	Displays the current operation mode and status of the kinematics interface in addition to navigation buttons to all other visualizations.
KinHoming_mode	Commanding and monitoring the homing mode
KinPosition_mode	Commanding and monitoring the positioning modes

Kinematics Interface

Visualization	Description
• KinRCL_mode	Commanding and monitoring the RCL operation mode
Utilities	
• ConveyorBelt	Configuring one or several belts
• RCLPoint	Setting and reading a RCL point, reading an RCL point list and program list
• RCLProg	Reading, saving and compiling a RCL program
• Setup	Commanding and monitoring the jogging mode

Tab.6-31: Kinematics interface visualizations

The following global variables are used to control and access system information within the visualizations:

- arKinCtrl_gb[]
- arKinStatus_gb[]
- KinData[]
- VisuKinematicsNo

System Overview Visualization

With the visualization **KinSystem_Overview_All** all kinematics configured in the project can be integrated quickly. The default kinematics displayed has no reference. The following explains how this reference has to be set. Additional kinematics can be added for display in the offline mode.

The visualization provides an overall system status and enables clearing errors. Individual kinematics control elements return the kinematic name and the kinematic number, certain diagnoses and the current position as world coordinates together with the operation mode status displays.

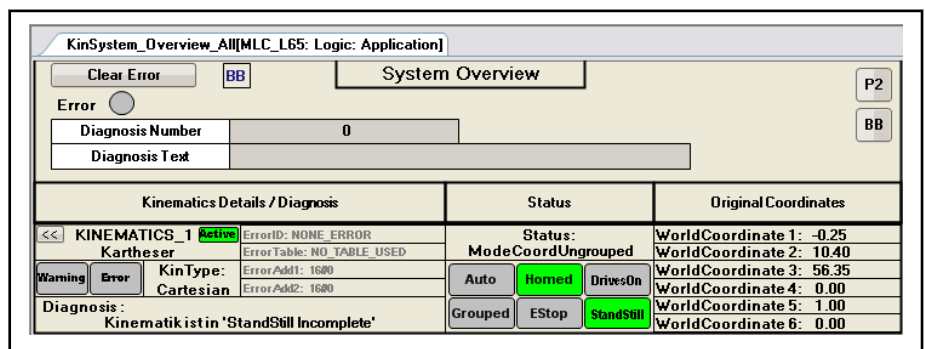


Fig. 6-25: KinSystem_Overview_All display

Clear Error This button triggers "Clear Error" via the IMC interface.



No kinematic errors or axis errors are cleared. Those have to be cleared separately in the respective kinematic or axis overviews.

P2 This button can be used to switch to parameterization mode P2.

BB This button can be used to switch to operating mode BB.

Adding a kinematics to the system overview

Add a kinematics to the system overview display in the same manner as a new visualization: select the respective element and specify the kinematic reference. In the following, two approaches to add a new kinematics to the system overview are described:

Kinematics Interface

Option 1: Creating a kinematics visualization with a frame

1. Open the respective IndraWorks project and select the tab **Logic ▶ Application ▶ AxisInterface ▶ Plug-ins ▶ KinInterface ▶ Visualizations ▶ SystemOverview**.
2. With IndraLogic in offline mode double-click on the visualization "Kin-System_Overview_All".
3. Click on the button **Frame**  in the **ToolBox** window and generate an outline below the last kinematics in the correct size.

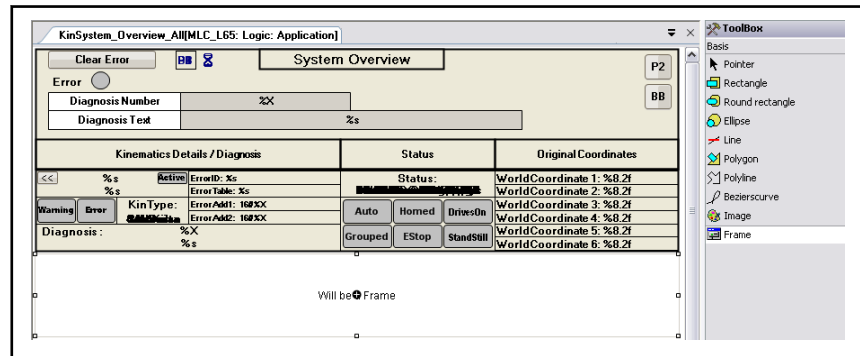


Fig.6-26: Adding a kinematics to the project

4. Right-click on the frame and activate "Frame selection". Select the **Kin-System_Overview** element from the visualization selection window. A completely new system overview for the kinematics is displayed.
5. Open the properties window for the newly inserted frame and enter the name of the kinematics under **Referenced visualizations ▶ Referenced visualizations ▶ KinSystem_Overview ▶ m_Input_KinRef**. Adapt the reference of the first kinematics in the same way.
The system overview is now available for the created kinematics.

Option 2: Copying an existing kinematics visualization

1. Open the respective IndraWorks project and select the tab **Logic ▶ Application ▶ AxisInterface ▶ Plug-ins ▶ KinInterface ▶ Visualizations ▶ SystemOverview**.
2. With IndraLogic in offline mode double-click on the visualization "Kin-System_Overview_All".
3. Copy the existing kinematics visualization and insert it as often as desired.
4. Open the properties window for the newly inserted kinematics visualizations and enter the name of the kinematics under **Referenced visualizations ▶ Referenced visualizations ▶ KinSystem_Overview ▶ m_Input_KinRef**. Adapt the reference of the first kinematics in the same way.
The system overview is now available for the specified kinematics.



One of the options described above has to be repeated for all kinematics added to the project.

Navigation system overview

It is possible to go to the visualization of the individual kinematics "Kinematics_Overview" from the kinematics overview using . This button is on the

Kinematics Interface

left in the column "Kinematics Details / Diagnosis". All other visualizations can be accessed from the individual kinematics display.

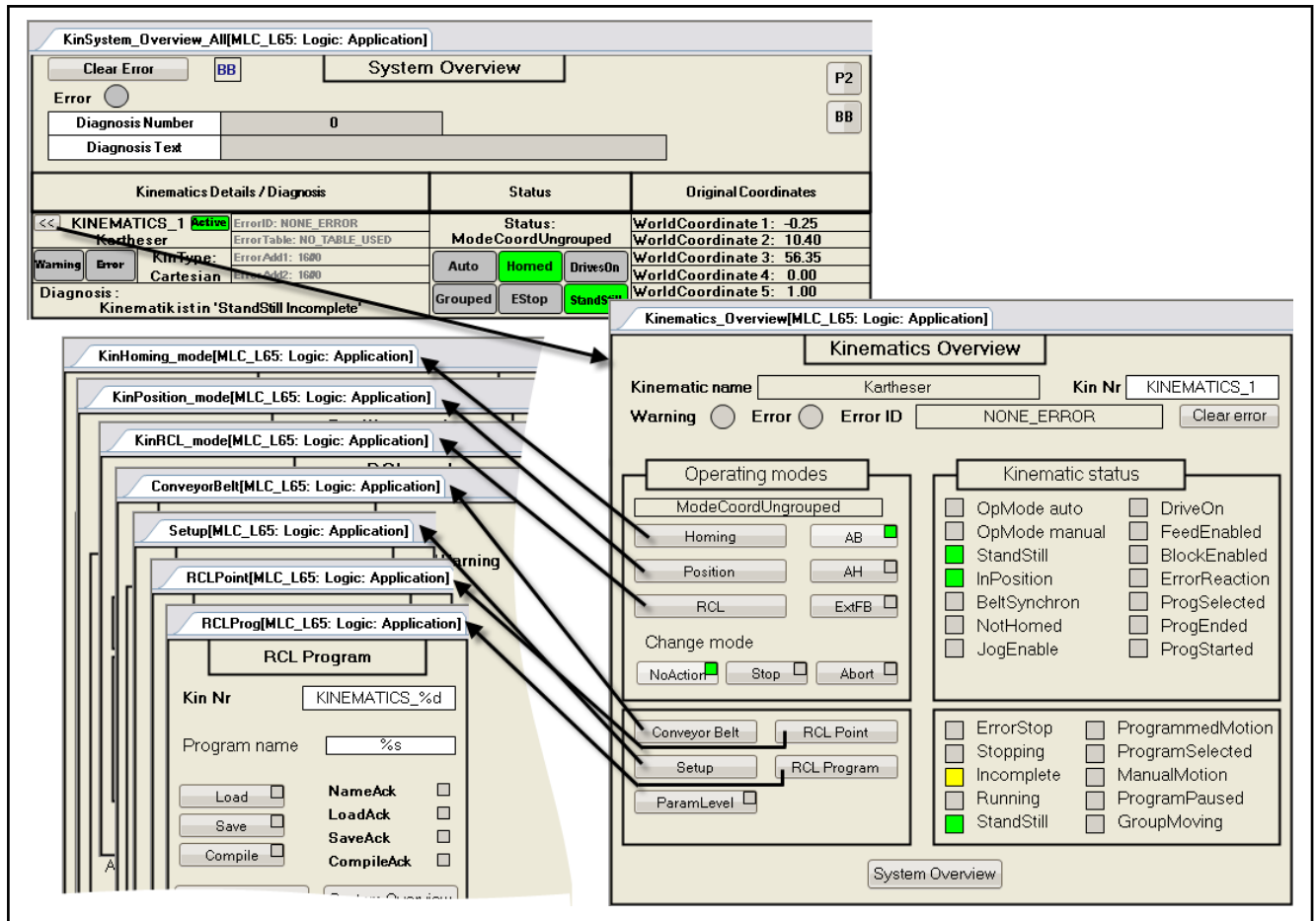


Fig.6-27: Navigation system overview



It is possible to switch between various kinematics in every visualization by entering the kinematic number in the field **Kin Nr**.

Kinematics Overview Visualization

The **Kinematics_Overview** visualization clearly displays the most important functions of the kinematics interface and the KinData interface diagnostics.

Kinematics Interface

Kinematics Overview

Kinematic name: %s Kin Nr: KINEMATICS_%d

Warning ☐ Error ☐ Error ID: %s

Operating modes

%s

 ☐

 ☐

 ☐

Change mode

☐ ☐

☐

SetupMode ACTIVE

Kinematic status

☐ OpMode auto	☐ DriveOn
☐ OpMode manual	☐ FeedEnabled
☐ StandStill	☐ BlockEnabled
☐ InPosition	☐ ErrorReaction
☐ BeltSynchron	☐ ProgSelected
☐ NotHomed	☐ ProgEnded
☐ JogEnable	☐ ProgStarted

☐ ErrorStop	☐ ProgrammedMotion
☐ Stopping	☐ ProgramSelected
☐ Incomplete	☐ ManualMotion
☐ Running	☐ ProgramPaused
☐ StandStill	☐ GroupMoving

Fig.6-28: Kinematics overview visualization

Use the buttons **Homing**, **Position** and **RCL** to change the visualization of the respective operation mode where special values for the respective operation mode can be entered. Use the buttons **AB**, **AH** and **ExtFB** to switch directly to the respective operation mode. Use the buttons **NoAction** and **Stop** to define the reaction following a change in operation mode.

Use the buttons **Conveyor Belt**, **Setup**, **RCL Point** and **RCL Program** to open the respective visualizations stored in "Utilities" for belt configuration, kinematics jogging, RCL point editing and RCL program handling. The message "SetupMode ACTIVE" indicates that the jogging mode is enabled.

The right part of the visualization displays diagnostic data of the KinData interface and current PLCOpen state of the kinematics.



Depending on the Robot Control version used, certain display elements are hidden. Therefore, only kinematics interface functions that are available in the respective Robot Control version can be selected.

Referencing Overview (Homing) Visualization

The **KinHoming_mode** visualization is used to reference kinematic axes.

Fig. 6-29: Referencing overview visualization

The order of the individual axes may be chosen at random. By default, all axes assigned to the kinematics are entered according to their configuration order. Axis number 0 is the abort condition for the homing sequence.

The right part of the visualization displays the reference status of all specified axes and the kinematics. If the axis or kinematics is referenced, the display element turns green.



The display element for "Kinematics homed" only turns green when the homing sequence was processed without errors, even if all axes are already referenced.

Positioning Overview (Position) Visualization

The visualization **KinPosition_mode** positions the kinematics.

Kinematics Interface

KinPosition_mode[MLC_L65: Logic: Application]

Position mode

Kinematic name Kin Nr

Warning ☐ Error ☐ Error ID

Command values

Point

[1] [2] [3]
 [4] [5] [8]
 [7] [8] [9]
 [10] [11] [12]
 [13] [14] [15]
 [16]

Velocity
 Acceleration
 Deceleration
 Jerk
 BlendRadius

AuxPoint

[1] [2] [3]
 [4] [5] [8]
 [7] [8] [9]
 [10] [11] [12]
 [13] [14] [15]
 [16]

Actual values

OrigCoord 1
 OrigCoord 2
 OrigCoord 3
 OrigCoord 4
 OrigCoord 5
 OrigCoord 6
 OrigCoord 7
 OrigCoord 8
 OrigCoord 9
 OrigCoord 10
 OrigCoord 11
 OrigCoord 12
 OrigCoord 13
 OrigCoord 14
 OrigCoord 15
 OrigCoord 16
 Velocity
 Acceleration
 Jerk

CoordSystem

SlopeType

SetMode

☐ Direct Abs ☐ Linear Abs ☐ Circular Abs
☐ Direct Rel ☐ Linear Rel ☐ Circular Rel
☐ Stop ☐ Interrupt ☐ Continue
☐ EnableCyclicScanning ☐ Retrigger OpMode

Aktual coordination system
 Actual slope type
 Actual set mode

☐ InBuffer ☐ Stopping
☐ CmdDone ☐ ErrorStop
☐ InPosition ☐ GroupMoving
☐ BeltSynchron
 Actual operating mode

Fig.6-30: Positioning overview visualization

The following may be entered or selected

- Position
- Velocity
- Acceleration
- Deceleration
- Jerk
- Blending radius
- Intermediate position
- Coordinate system
- Slope type of the motion
- Blending type

Use the buttons **DirectAbs**, **LinearAbs**, etc. to select the respective positioning mode. For more details, see the operation mode description.

Kinematics Interface

The right part of the visualization displays the current position as world coordinates, the parameterized boundary conditions, important status elements of the kinematics interface and the KinData interface, and the current operation mode.



Depending on the Robot Control version used, certain display elements are hidden. Therefore, only kinematics interface functions that are available in the respective Robot Control version can be selected.

Robot Control Language (RCL) Visualization

Use the visualization **KinRCL_mode** to select and start a RCL program.

Fig. 6-31: Robot Control Language visualization

Specify the RCL program name in the field "ProgName" to start the program via the **Start** button. Stop the RCL program by changing the operation mode via the buttons **AB** or **AH**.

The right part of the visualization displays important status elements of the kinematics interface and the KinData interface as well as the current operation mode.



Specify the RCL program name in the field "ProgName" **without** stating its file name extension (*.ird).



Only permissible for Robot Control V1.

Belt Configuration (ConveyorBelt) Visualization

Use the **ConveyorBelt** visualization to configure all belts belonging to the kinematics.

Kinematics Interface

Fig.6-32: Belt configuration visualization

Specify the belt configuration and the belt error reaction to parameterize the respective belt. Multiply the belt configuration overview according to the number of belts. This is done in the same way as adding a kinematics to the system overview (see [chapter "System Overview Visualization" on page 303](#)).

Indicate a consecutive belt number as a referenced visualization. Via this belt number the kinematics interface structure `arKinCtrl_gb[].Admin.ConveyorBelt` is internally accessed.



The belt is configured internally with the function block "ML_SetBeltParameter" (Robot Control V1) or "ML_SetBeltConfiguration" (Robot Control V2). For more detailed information, see the respective documentation.



Depending on the Robot Control version used, certain display elements are hidden. Therefore, only kinematics interface functions that are available in the respective Robot Control version can be selected.

Jog/Setup Mode Visualization

With the **Setup** visualization the kinematics may be jogged.

The screenshot shows a graphical user interface for setting up kinematics. At the top, a title bar reads "Setup[MLC_L65: Logic: Application]". Below this is a "Setup" section. The "Kin Nr" field is set to "KINEMATICS_%d". The "CoordSystem" section has a text input field with "%s" and a grid of buttons: MCS, ACS, TCS, WpCS, BCS, PCS1, PCS2, and PCS3. The "JogMode" section has buttons for "Step small", "Cont slow", "Step large", and "Cont fast". Below these are another text input field with "%s" and a grid of buttons for axis movement: 1+, 2+, 3+, 4+, 5+, 6+, 1-, 2-, 3-, 4-, 5-, and 6-. At the bottom, there are checkboxes for "Enable" and "EnableAck", and two buttons: "Kin Overview" and "System Overview".

Fig. 6-33: Jog/setup operation visualization

Define the coordinate system where jogging is to take place with the buttons in "CoordSystem". Specify the step width or the traversing velocity in "Jog-Mode" in the same way. Use the buttons 1+ to 6- to move the first 6 axes of the kinematics in the positive and negative direction according to the settings.



Belt axes do not belong to the axis group and can therefore not be jogged.



Depending on the Robot Control version used, certain display elements are hidden. Therefore, only kinematics interface functions that are available in the respective Robot Control version can be selected.

Robot Control Language Point (RCLPoint) Visualization

Use the **RCLPoint** visualization to read or write a RCL point and/or to read a RCL point list or program list.

Kinematics Interface

Fig. 6-34: Robot Control Language point visualization

It is possible to read out a point (**GetPoint**), a point list (**GetPointList**) or a program list (**GetProgList**) via the values to be specified in this visualization. In order to set a point (**SetPoint**), set the kinematics interface structure element `arKinCtrl_gb[] .Admin.RCLPoint.Point` in the user program.



The file name has to be specified **with** the file name extension (*.pkt) in the field "FileName".



Only permissible for Robot Control V1.

Robot Control Language Program (RCLProg) Visualization

*It is possible to copy a RCL file via the **RCLProg** visualization:*

- From the Compact Flash card to the internal memory of the control
 - From the internal memory of the control to the Compact Flash card
- or

*It is possible to compile a RCL file via the **RCLProg** visualization:*

- On the control

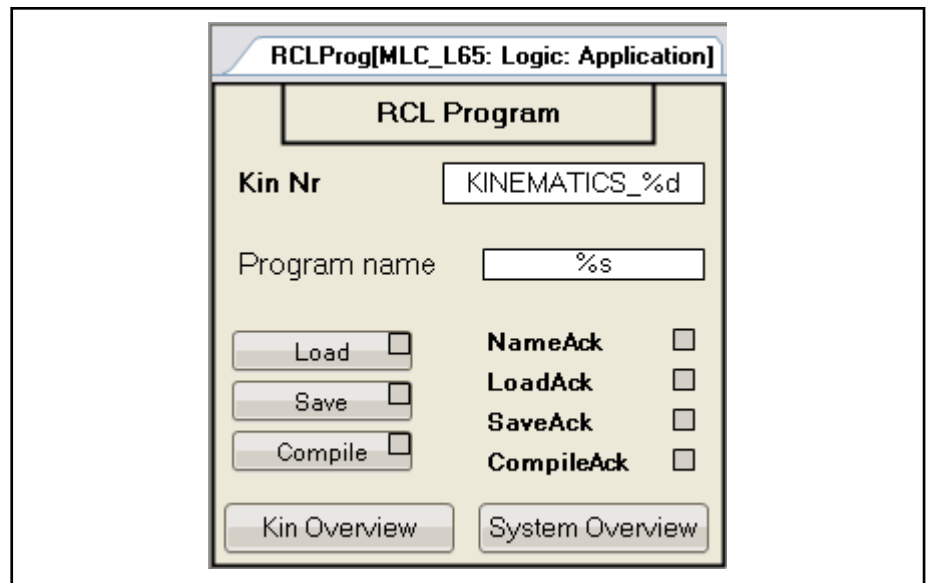


Fig. 6-35: Robot Control Language program visualization



The file name has to be specified **with** the file name extension (e.g. *.qll) in the field "Program name".



Only permissible for Robot Control V1.

6.2.7 Kinematics Interface User Extension

Overview

Notes regarding the implementation of user extensions

The `arKinCtrl_gb[ ]` and `arKinStatus_gb[ ]` structures can be extended to adapt the kinematics interface to special applications.

This section describes how to extend the kinematics interface by adding a new functionality. We assume that the extension structures prepared in the kinematics interface are used. The following only describes the required changes in the POU's and structures.

The user extensions were implemented using the object-oriented extensions from IndraLogic 2G.

Please note the following particularities:

- The function block **TE_KinematicsInterface** is derived from the basic function block "MB_KinInterfaceType01". The basic function block or its methods/actions can be called via the keyword "SUPER". At several locations, the `mSetError` method of the `MB_KinInterfaceType01` function block is called to enter errors via `SUPER^.mSetError(...)`; into the diagnostic system for example.
- Within the function block **TE_KinematicsInterface**, the data of `arKinCtrl_gb` can be accessed using the inputs `AdminCtrlExt`, `CoordModeCtrlExt`, etc. and the data of `arAxisStatus_gb` can be accessed using `AdminStatusExt`, `DiagStatusExt`, etc.



The inputs `AdminCtrl` (without `*Ext`) etc. belong to the basic function block and are not to be used as no user extensions are available there.

Kinematics Interface

- The inputs of the function block `TE_KinematicsInterface` are defined as "REFERENCE TO". In the "mInitExtension" method, the references are initialized once and do not have to be transferred anymore when the function block is cyclically called.

Extending the Kinematics Interface Structure

To extend the structures `TE_KINEMATICS_CONTROL_TYPE01` and `TE_KINEMATICS_STATUS_TYPE01` by additional substructures, the following steps are required in addition to the procedure described below:

1. Enter new elements in `TE_KINEMATICS_CONTROL_TYPE01` and/or `TE_KINEMATICS_STATUS_TYPE01`
2. Add the additional inputs at the function block `TE_KinematicsInterface` as REFERENCE TO
3. Initialize the references in the method "mInitExtension" of the function block `TE_KinematicsInterface`



To extend the substructures `TE_KINEMATICS_ADMINISTRATION`, `TE_KINEMATICS_COORDINATED`, etc. only step 1 of the procedure described above is required for the respective structure. In this structure, the extension has to be added below (* add your extensions below here | Erweiterungen hier einfüegen*). Then, the newly created input or output is available via the respective *Ext structure element.

Extending the Function Block `TE_KinematicsInterface`

As a last step in the user extension procedure, extend the `TE_KinematicsInterface` function block in such a way that the program code of the user extension may be inserted.

1. In the project tree, right-click to select **`TE_KinematicsInterface`** and **Add ► Action**.
2. Specify a name for the new action that matches the extension
3. Insert the program code into the new action.
Use the inputs with the suffix *Ext (e.g. `AdminCtrlExt`, `DiagStatusExt`, etc.) to access the structures `arKinCtrl_gb[]` and `arKinStatus_gb[]`.
4. Call the new action in the `TE_KinematicsInterface` function block.
5. Compile the project again and check it for programming errors
6. Load the project in the control.

The new input and output elements are now part of the kinematics interface structure and can be viewed using the `UserVarGlobal` variables.

7 Application Examples

7.1 Application Examples, Overview

In the following, the Robot Control application is explained, using different examples.

7.2 Application example 1: Using Coordinate Systems

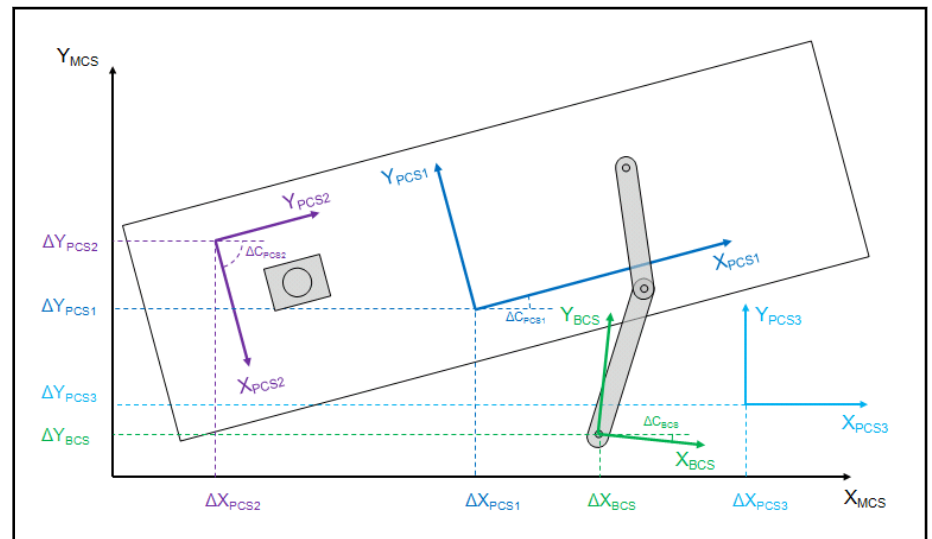


Fig. 7-1: Application example: Coordinate systems

PCS1: Coordinate system of the belt

A PCS in the direction of the belt is defined to transport a product in sync with a belt.

Moving and rotating the PCS1 to the MCS is defined via [Set Belt Configuration, page 30](#) with $(\Delta X_{PCS1}, \Delta Y_{PCS1}, 0, 0, 0, \Delta C_{PCS1})$.

PCS2: Coordinate system of the camera

To facilitate recording of product positions, a PCS is defined as camera coordinate system.

Moving and rotating the PCS2 to the MCS is defined via [Set Cartesian Transformation, page 30](#) with $(\Delta X_{PCS2}, \Delta Y_{PCS2}, 0, 0, 0, \Delta C_{PCS2})$.

PCS3: Coordinate system of the pallet

To facilitate the positioning of products on a pallet, a PCS is defined as pallet coordinate system.

Moving and rotating the PCS3 to the MCS is defined via [Set Cartesian Transformation, page 30](#) with $(\Delta X_{PCS3}, \Delta Y_{PCS3}, 0, 0, 0, \Delta C_{PCS3})$.

BCS: Coordinate system of the robot

The robot is defined as scara (9) in parameter K-0-0031.

Moving and rotating the BSC to the MCS can be defined via parameter K-0-0042 with $(\Delta X_{BCS}, \Delta Y_{BCS}, 0, 0, 0, \Delta C_{BCS})$.

Procedure

The camera records products that are picked up from the belt and are to be placed on the pallet.

The camera records the product and returns the position in PCS2.

The position in PCS2 (X_{PCS2}, Y_{PCS2}) is assigned the belt value (B1) of the belt for the belt-synchronous motion at the time of recording $(X_{PCS2}, Y_{PCS2}, B1)$.

This position is commanded using "[Move Linear Absolute, page 29](#)".

Application Examples

It is waited until the product has reached the belt-synchronous operating range. Subsequently, the position is traversed (including synchronizing to the belt).

If the product has been picked up, a position in the pallet can be defined directly in PCS3 (X_{PCS3} , Y_{PCS3} , 0) using "Move Linear Absolute".

As this point is commanded without a belt value, a desynchronization from the belt takes place automatically.

The required conversions between the coordinate systems are executed automatically.

7.3 Application example 2: Belt Synchronization

Initial situation

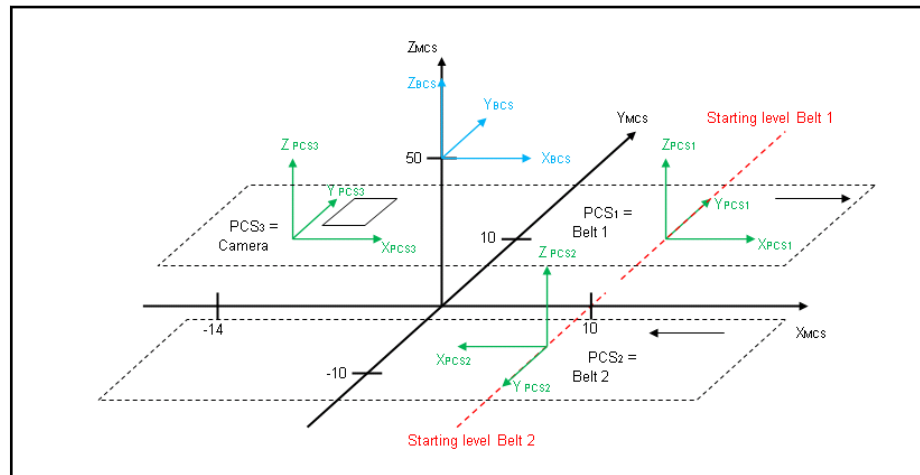


Fig. 7-2: Application example: Belt-synchronous

A camera records products that were picked up from a belt and have been placed on a second belt.

The camera returns the (X, Y) positions of the project in an individual product coordinate system (PCS_3).

Both belts represent two more product coordinate systems (PCS_1 and PCS_2).

Moving and rotating the camera coordinate system (PCS_3) around the **MCS** is defined using the command [Set Cartesian Transformation, page 30](#) (-14, 10, 0, 0, 0, 0) $_{PCS3}$.

The first three specified values stand for the coordinate system shift in X, Y and Z direction. The last three values refer to the rotation around the X, Y or Z axis.

Both belts are configured, rotated and moved around the **MCS** using the command [Set Belt Configuration, page 30](#) (10, 10, 0, 0, 0, 0) $_{PCS1}$, (10, -10, 0, 0, 0, 180) $_{PCS2}$.

The initial plane shown in the figure is also specified using this command.

Only once the recorded product has passed this level, the product is in the belt-synchronous operating range and can be synchronized. The initial planes of both belts do not mandatorily have to be identical!

For this example, a delta kinematics is defined via parameter K-0-0031 = 20 and has been moved in the positive Z direction using parameter K-0-0042 (0, 0, 50, 0, 0, 0) $_{BCS}$.

Application Examples

The product position returned by the camera (X_{PCS3}, Y_{PCS3}) has to be assigned the current belt value at the time of the camera recording in order to establish a time reference.

The current belt value is provided by parameter K-0-0100 at the position of the respective axis.

The resulting point can be traversed using a motion command (e.g. [Move Linear Absolute](#), page 29).

In this case, the MLC waits until the product has passed the initial plane and thus reached the belt-synchronous operating range. At this point in time, synchronization is started and the point is traversed.

The required calculations to determine the current product position in relation to the belt is automatically executed by the MLC and can be read out by means of the function [Move Point](#), page 30.

The kinematics remains belt-synchronous until another motion command with a belt value of 0 (for absolute motions) or 1,000,000 (for relative motions) is traversed.

In this case, desynchronization from the belt takes place.

Calculation example

General mapping rule for belt synchronization:

$$B_{act} - B_{product} + X_{PCS1} > 0$$

B_{act}	Current belt value
$B_{product}$	Belt value at the time of camera recording
X_{PCS1}	X-position of the product in the PCS_1

Fig. 7-3: Mapping rule for belt synchronization

A delta kinematics with 3 positioning axes, 1 orientation axis and 2 belt axes.

For example, the camera recorded a product with position (10, 10, 0, 0, 35, 0)_{PCS3}.

From the belt view, i.e. the product coordinate system 1, the product is at position (-14, 10, 0, 0, 35, 0)_{PCS1}.

For another calculation point in time, the belt has a current belt value of 40, resulting in the following rule:

$$40 - 35 + (-14) = -9 < 0$$

This means that the kinematics waits for 10 belt value before synchronizing to this point.

After these 10 belt values, the mapping rule is fulfilled and the product has been run through the initial plane.

7.4 Application Example 3: User-Defined Transformation

This example implements the 2-axis delta slide transformation (transformation scheme 45) as user-defined transformation.

This is only an example: If the transformation scheme 45 is required, it can be used directly without implementing the transformation scheme in the PLC.

The same basic conditions as specified in [2-Axis Delta Slide](#), page 94 apply for the configuration.

Implementing the function block

First, a function block has to be implemented that implements the [MC_KIN_REF](#) interface.

Application Examples

Declaration

```

FUNCTION_BLOCK DeltaSlide IMPLEMENTS MC_KIN_REF
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    L1: REAL := 0;
    L2: REAL := 0;
    Gamma: REAL := 0;
    Ymin, Ymax: REAL := 0;
    GRAD_RAD_FAKTOR : REAL := 0.01745329251994329; // (2*PI)/360
END_VAR

```

Implementing the initialization method

In this example, some variables are declared that access three interface methods.

The initialization method executes the following checks and calculations:

- Checking the lengths 1 and 2 in parameter "K-0-0040, Transformation, lengths of the mechanics"
- Checking factor 1 in parameter "K-0-0041, Transformation, factors"
- Check if at least 2 positioning axes have been configured in parameter "K-0-0008, Axis configuration list"
- Check if both positioning axes are scaled translatory absolute
- Calculate the minimum and maximum Y-value in the basic coordinate system

Initialization

```

METHOD Initialisation : INT
VAR_INPUT
    AxesConfigData : REFERENCE TO ARRAY [1..16] OF ML_KIN_AXES_CONFIG_DATA;
    TransformationLengths : REFERENCE TO ARRAY [1..16] OF REAL;
    TransformationFactors : REFERENCE TO ARRAY [1..16] OF REAL;
END_VAR
VAR
    Axis1_A_0_0021: DWORD;
    Axis2_A_0_0021: DWORD;
END_VAR

// arm lengths ok?
IF (TransformationLengths[1] > 0.0) THEN
    L1 := TransformationLengths[1];
ELSE
    Initialisation := 1;
    RETURN;
END_IF

IF (TransformationLengths[2] >= 0.0) THEN
    L2 := TransformationLengths[2];
ELSE
    Initialisation := 2;
    RETURN;
END_IF

IF ((TransformationFactors[1] >= 0.0) AND
    (TransformationFactors[1] < 360.0)) THEN
    Gamma := TransformationFactors[1];
ELSE
    Initialisation := 3;
    RETURN;
END_IF

IF ((AxesConfigData[1].Configuration <> ML_KIN_AXIS_TYPE_POSITION) OR
    (AxesConfigData[2].Configuration <> ML_KIN_AXIS_TYPE_POSITION)) THEN
    Initialisation := 4;
    RETURN;
END_IF

Axis1_A_0_0021 := DV_Axis[AxesConfigData[1].AxisNumber].A_0_0021;

```



```

Axis2_A_0_0021 := DV_Axis[AxesConfigData[1].AxisNumber].A_0_0021;

//Axes must be translatory
IF (Axis1_A_0_0021.29 = TRUE) THEN
    Initialisation := 5;
    RETURN;
END_IF
IF (Axis2_A_0_0021.29 = TRUE) THEN
    Initialisation := 5;
    RETURN;
END_IF

//Axes must be absolute
IF (Axis1_A_0_0021.28 = TRUE) THEN
    Initialisation := 5;
    RETURN;
END_IF
IF (Axis2_A_0_0021.28 = TRUE) THEN
    Initialisation := 5;
    RETURN;
END_IF

Ymin := - L1 - (COS((180 - Gamma)* GRAD_RAD_FAKTOR)*L2);
Ymax := SIN((180 - Gamma) * GRAD_RAD_FAKTOR)*L2;

    Initialisation := 0;

```

Implementing the forward transformation

The forward transformation calculates coordinates in the basic coordinate system from coordinates in the axis coordinate system.

Please note errors where the axes are too far apart or the axis positions are inverted.

Forward transformation

```

METHOD Forward : INT
VAR_INPUT
    OriginalCoordinates : REFERENCE TO ARRAY [1..16] OF REAL;
    JointCoordinates    : REFERENCE TO ARRAY [1..16] OF REAL;
END_VAR
VAR
    Alpha : REAL;      //d__Angle_AC_X
    Beta  : REAL;      //d__Angle_CTCP_X
    Gamma_Temp : REAL; //d__Angle_L1_L2_Temp;
    Cx     : REAL;
    Cy     : REAL;
END_VAR

IF (JointCoordinates[2] - JointCoordinates[1]) < 0.0 THEN
    Forward := 1; // arms wrong
    RETURN;
END_IF

Cx := (JointCoordinates[1] + JointCoordinates[2]) / 2;

IF (L1*L1 - (Cx - JointCoordinates[1])*(Cx - JointCoordinates[1])) <= 0.0 THEN
    Forward := 2; //arms too short
    RETURN;
END_IF

Cy := - Sqrt(L1*L1 - (Cx - JointCoordinates[1])*(Cx - JointCoordinates[1]));
Alpha := ACOS((Cx - JointCoordinates[1])/L1);

Gamma_Temp := 180 - Gamma; (* angle correction *)
Beta := Alpha - (Gamma_Temp*GRAD_RAD_FAKTOR);

//make new base coordinates
OriginalCoordinates[1] := Cx + (COS(Beta)*L2);
OriginalCoordinates[2] := Cy - (SIN(Beta)*L2);

Forward := 0;

```

Implementing the backward transformation

The backward transformation calculates coordinates in the axis coordinate system from coordinates in the basic coordinate system.

Please note errors where the point could not be reached.

Application Examples

Backward transformation

```

METHOD Inverse : INT
VAR_INPUT
    OriginalCoordinates : REFERENCE TO ARRAY [1..16] OF REAL;
    JointCoordinates    : REFERENCE TO ARRAY [1..16] OF REAL;
END_VAR
VAR
    ATPC, AB : REAL;
    Beta  : REAL; //d__Angle_ATCP_X
    Delta : REAL; //d__Angle_AC_ATCP
    Alpha : REAL; //d__Angle_AC_X
    Ax, Bx : REAL;
    TCPx, TCPy : REAL;
    GRAD_RAD_FAKTOR : REAL := 0.01745329251994329; // (2*PI)/360
END_VAR

IF (Ymin > OriginalCoordinates[2] OR Ymax <= OriginalCoordinates[2]) THEN
    Inverse := 1; // unreachable
    RETURN;
END_IF

TCPx := OriginalCoordinates[1];
TCPy := OriginalCoordinates[2];

ATPC := SQRT((L1*L1) + (L2*L2) - (2 * L1 * L2 * COS(Gamma* GRAD_RAD_FAKTOR))); // Law of cosines
Ax := TCPx - SQRT(ATPC*ATPC - TCPy*TCPy);

Beta := ATAN2(-TCPy, (TCPx-Ax));
Delta := ACOS( -((L2*L2) - (L1*L1) - (ATPC * ATPC)) / (2 * L1 * ATPC) ); // Law of cosines
Alpha := Beta + Delta;
AB := 2 * L1 * COS(Alpha);

Bx := Ax + AB;

JointCoordinates[1] := Ax;
JointCoordinates[2] := Bx;

Inverse := 0;

```

Registering the transformation at a kinematics

It is assumed that a kinematics by the name of "K1" exists. For this kinematics, a "DeltaSlide" transformation is to be registered.

Note that the **instance** myDeltaSlide is registered in the kinematics, not the function block type DeltaSlide.

For all other kinematics, an individual **instance** of the DeltaSlide function block has to be used.

Use

```

PROGRAM PlcProg
VAR
    myDeltaSlide : DeltaSlide;
    setKinTrafo : MC_SetKinTransform;
    initTransformation : BOOL;
END_VAR

IF(initTransformation) THEN
    setKinTrafo.KinTransform := myDeltaSlide;
    setKinTrafo.Execute := TRUE;
    setKinTrafo.AxisGroup := K1;
    IF(setKinTrafo.Done) THEN
        initTransformation := FALSE;
        setKinTrafo.Execute := FALSE;
        setKinTrafo.AxisGroup := K1;
    END_IF
    IF(setKinTrafo.Error) THEN
        initTransformation := FALSE;
    END_IF
END_IF

```

7.5 Application Example 4: Extended User-Defined Transformation

This example implements the 3-axis Scara transformation (transformation scheme 9) without the orientation axis as an extended user-defined transformation.

This is only an example: If the transformation scheme 9 is required, it can be used directly without implementing the transformation scheme in the PLC.

The same basic conditions as specified in [3-Axis Scara with 1 Optional Orientation Axis, page 77](#) apply for the configuration.

Implementing the function block

First, a function block has to be implemented that implements the MC_KIN_REF_EXT interface.

Declaration

```
FUNCTION_BLOCK Scara IMPLEMENTS MC_KIN_REF_EXT
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    c_boLeftArm: BOOL := FALSE;
    dL1 : LREAL := 0.0;
    dL2 : LREAL := 0.0;
    FLT_EPSILON: LREAL := 1.192092896E-07;
    GRAD_RAD_FAKTOR: LREAL := 0.01745329251994329; (* (2*PI)/360 *)
    RAD_GRAD_FAKTOR: LREAL := (1.0/GRAD_RAD_FAKTOR);
    PI_GANZ : LREAL := 3.1415926535897928;
END_VAR
```

In this example, some variables are declared that the interface methods access.

Implementing the GetUsedAxes method

The GetUsedAxes method returns the degrees of freedom of the transformation. In this example, the degrees of freedom of the transformation are in the X/Y plane.

GetUsedAxes method

```
METHOD GetUsedAxes : UINT
GetUsedAxes := ML_KIN_USED_AXES.X_AXIS OR ML_KIN_USED_AXES.Y_AXIS;
```

Implementing the initialization method

The initialization method executes the following checks and calculations:

- Checking the lengths 1 and 2 in parameter "K-0-0040, Transformation, lengths of the mechanics"
- Check if at least 2 positioning axes have been configured in parameter "K-0-0008, Axis configuration list"
- Check if the scaling of both positioning axes is rotary absolute

Initialization

```
METHOD Initialisation : UDINT
VAR_INPUT
    AxesConfigData : REFERENCE TO ARRAY [1..16] OF ML_KIN_AXES_CONFIG_DATA;
    TransformationLengths : REFERENCE TO ARRAY [1..16] OF LREAL;
    TransformationFactors : REFERENCE TO ARRAY [1..16] OF LREAL;
END_VAR
VAR
    axisScalingA1: DWORD;
    axisScalingA2: DWORD;
END_VAR

IF (TransformationLengths[1] > 0.0) THEN
    dL1 := TransformationLengths[1];
ELSE
    Initialisation := 16#F12C008A; // trafo values wrong
    RETURN;
```

Application Examples

```

END_IF

IF (TransformationLengths[2] > 0.0) THEN
    dL2 := TransformationLengths[2];
ELSE
    Initialisation := 16#F12C008A; // trafo values wrong
    RETURN;
END_IF

IF (AxesConfigData[1].Configuration <> ML_KIN_AXIS_CONFIG.ML_KIN_AXIS_TYPE_POSITION) OR
    (AxesConfigData[2].Configuration <> ML_KIN_AXIS_CONFIG.ML_KIN_AXIS_TYPE_POSITION) THEN
    Initialisation := 16#F22B0025; // too less axes
    RETURN;
END_IF

axisScalingA1 := DV_Axis[AxesConfigData[1].AxisNumber].A_0_0021;
axisScalingA2 := DV_Axis[AxesConfigData[2].AxisNumber].A_0_0021;

IF axisScalingA1.28 OR NOT(axisScalingA1.29) OR // axis 1 must be absolute rotatory
    axisScalingA2.28 OR NOT(axisScalingA2.29) THEN // axis 2 must be absolute rotatory
    Initialisation := 16#F22B0024; // axis has wrong scaling
    RETURN;
END_IF

```

Implementing the DetermineMemory method

The DetermineMemory method defines the arm behavior of the Scara robot. It calculates from the axis coordinates if the kinematics is right- or left-armed. The "TrafoChoise" input is ignored. If the value is "TRUE", the arm behavior can be used with the next inverse transformation, for example, where the first axis moves only a little. This can result in less time-consuming motions, if axis 2 is more dynamic than axis 1. However, this is not implemented here.

DetermineMemory method

```

METHOD DetermineMemory : UDINT
VAR_INPUT
    ACS : REFERENCE TO ARRAY [1..16] OF LREAL;
    TrafoChoise : BOOL;
END_VAR

IF( ACS[2] > FLT_EPSILON ) THEN
    c_boLeftArm := FALSE;
END_IF

IF( ACS[2] < -FLT_EPSILON ) THEN
    c_boLeftArm := TRUE;
END_IF

```

Implementing the forward transformation

The forward transformation calculates coordinates in the basic coordinate system from coordinates in the axis coordinate system.

Note the error cases!

Forward transformation

```

METHOD Forward : UDINT
VAR_INPUT
    BCS : REFERENCE TO ARRAY [1..16] OF LREAL;
    ACS : REFERENCE TO ARRAY [1..16] OF LREAL;
END_VAR

VAR
    dPHI1 : LREAL;
    dPHI2 : LREAL;
    dPHI11 : LREAL;
    dPHI12 : LREAL;
END_VAR

dPHI1 := ACS[1];
dPHI2 := ACS[2];

IF ( (ABS(dPHI1) > 179.0)
    OR (ABS(dPHI2) > 179.0) ) THEN
    Forward := 16#F22B0021; // Angles out of range
    RETURN;
END_IF

dPHI11 := dPHI1 * GRAD_RAD_FAKTOR;
dPHI12 := ( dPHI2 * GRAD_RAD_FAKTOR ) + dPHI11;

```

```
BCS[1] := ( dL1 * COS( dPHI11 ) ) + ( dL2 * COS( dPHI12 ) );
BCS[2] := ( dL1 * SIN( dPHI11 ) ) + ( dL2 * SIN( dPHI12 ) );
```

Implementing the backward transformation

The backward transformation calculates coordinates in the axis coordinate system from coordinates in the basic coordinate system.

Note the error cases!

Backward transformation

```
METHOD Inverse : UDINT
VAR_INPUT
    BCS : REFERENCE TO ARRAY [1..16] OF LREAL;
    ACS : REFERENCE TO ARRAY [1..16] OF LREAL;
END_VAR
VAR
    dLengthTCP : LREAL; //length of the distance from the TCP to the origin
    dGamma : LREAL; //angle of the triangle; leads to phi2
    dBeta : LREAL; //angle of the triangle; leads to phi1
    dTheta : LREAL; //pitch angle of the distance from the TCP to the origin; leads to phi1
    dTemp : LREAL; //temporary variable
    dPHI1 : LREAL; //angle of the motor in degrees
    dPHI2 : LREAL; //angle of the motor in degrees
    dphi11 : LREAL; //angle of the motor in radian; d_phi12 = d__phi1 + d__phi2
    dphi21 : LREAL; //angle of the motor in radian; d_phi12 = d__phi1 + d__phi2
END_VAR

( *****
  Get Length of TCP-Origin
  ***** )
dLengthTCP := SQRT(BCS[1]*BCS[1] + BCS[2]*BCS[2]);
IF ( (dLengthTCP - FLT_EPSILON) > (dL1 + dL2) ) THEN
    Inverse := 16#F22B0022; // too far away
    RETURN;
END_IF

IF ( dLengthTCP > (dL1 + dL2) ) THEN
    dLengthTCP := dL1 + dL2;
END

( *****
  Calc d__Gamma
  ***** )
dTemp := ( (dL1 * dL1) + (dL2 * dL2) - (dLengthTCP * dLengthTCP) )
        / ( 2.0 * dL1 * dL2 );
IF ( ABS(dTemp) > 1.0 ) THEN
    Inverse := 16#F12C0479; // singular position: acos is not defined for |x| > 1, catch possible failure
    RETURN;
END_IF
dGamma := ACOS(dTemp);

( *****
  Calc d__Beta
  ***** )
IF (dLengthTCP < FLT_EPSILON) THEN
    Inverse := 16#F12C0479; // too near to base point
    RETURN;
END_IF

dTemp := ( (dL1 * dL1) + (dLengthTCP * dLengthTCP) - (dL2 * dL2) )
        / ( 2.0 * dL1 * dLengthTCP );
IF ( ABS(dTemp) > 1.0 ) THEN
    Inverse := 16#F12C0479; // singular position: acos is not defined for |x| > 1, catch possible failure
    RETURN;
END_IF
dBeta := ACOS(dTemp);

( *****
  Calc d__Theta
  ***** )
dTheta := atan2( BCS[2], BCS[1] );

( *****
  Set the angles considering to the arm
  ***** )
IF ( c_boLeftArm ) THEN
    //left arm
    dphi1 := dBeta + dTheta;
```

Application Examples

```

IF ( dphi1 > PI_GANZ) THEN
  dphi1 := dphi1 - ( 2.0 * PI_GANZ );
END_IF

dphi2 := dGamma - PI_GANZ;
IF(dphi2 > 0.0) THEN      //prevent the change of the arm
  dphi2 := -FLT_EPSILON; //may occur of inaccuracy of the calculation
END_IF

ELSE
  //right arm
  dphi1 := dTheta - dBeta;
  IF ( dphi1 < -PI_GANZ ) THEN
    dphi1 := dphi1 + ( 2.0 * PI_GANZ );
  END_IF

  dphi2 := PI_GANZ - dGamma;
  IF(dphi2 < 0.0) THEN      //prevent the change of the arm
    dphi2 := +FLT_EPSILON; //may occur of inaccuracy of the calculation
  END_IF
END_IF

( *****
  Transform angles to Grad
  ***** )
dPHI1 := dphi1 * RAD_GRAD_FAKTOR;
dPHI2 := dphi2 * RAD_GRAD_FAKTOR;

( *****
  Check angles
  ***** )
IF ( (ABS(dPHI1) > 179.0)
    OR (ABS(dPHI2) > 179.0) ) THEN
  Inverse := 16#F22B0021; // Angles out of range
  RETURN;
END_IF

ACS[1] := dPHI1;
ACS[2] := dPHI2;

```

GetMechanic method

The GetMechanic method calculates the mechanical data belonging to a point in the axis coordinate system. In case of the Scara, the mechanical view consists of 3 spheres (BASE, MID, TCP) and 2 cylinders (BASE to MID, MID to TCP). Note that the object name "TCP" is used by most visualizations to capture the path. The transformation length 2 (K-0-0040) provides the sizes of the rods and spheres. For simplification the function ML_AddMechanicData from the ML_Robot library is used.

GetMechanic method

```

METHOD GetMechanic : UDINT
VAR_INPUT
  ACS      : REFERENCE TO ARRAY [1..16] OF LREAL;
  MechanicData : REFERENCE TO ML_KIN_MECHANIC_DATA;
  MechanicDataSize : UDINT;
  MechanicDataRetSize : REFERENCE TO UDINT;
  UpdateOnly : BOOL;
END_VAR
VAR
  dphi11 : LREAL;           //angle of the motor in radian
  dphi12 : LREAL;           //angle of the motor in radian; d_phi12 = dphi1 + dphi2
  TCPx : LREAL;             //TCP point
  TCPy : LREAL;             //TCP point
  MIDx : LREAL;             //MID point
  MIDy : LREAL;             //MID point
  MID2TCPx : LREAL;
  MID2TCPy : LREAL;

  fRodRadius : LREAL := dL2 / 100.0;
  fSphereSize : LREAL := fRodRadius * 4.0;
  xMID2TCP : INT;
END_VAR

( *****
  compute angles
  ***** )
dphi11 := ACS[1] * GRAD_RAD_FAKTOR;
dphi12 := ( ACS[2] * GRAD_RAD_FAKTOR ) + dphi11;

```

```
(
*****
Get the MID
*****
)
MIDx := dL1 * COS( dphi11 );
MIDy := dL1 * SIN( dphi11 );

(
*****
Get the TCP
*****
)
TCPx := ( dL1 * COS( dphi11 ) ) + ( dL2 * COS( dphi12 ) );
TCPy := ( dL1 * SIN( dphi11 ) ) + ( dL2 * SIN( dphi12 ) );

MID2TCPx := TCPx - MIDx;
MID2TCPy := TCPy - MIDy;

IF NOT(UpdateOnly) THEN
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_CREATE_SPHERE, 'BASE', 0.0, 0.0, 0.0);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_CREATE_SPHERE, 'MID', 0.0, 0.0, 0.0);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_CREATE_SPHERE, 'TCP', 0.0, 0.0, 0.0);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_CREATE_ROD, 'ROD1', 0.0, 0.0, 0.0);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_CREATE_ROD, 'ROD2', 0.0, 0.0, 0.0);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_SIZE, 'BASE', fSphereSize, fSphereSize, fSphereSize);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_SIZE, 'MID', fSphereSize, fSphereSize, fSphereSize);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_SIZE, 'TCP', fSphereSize, fSphereSize, fSphereSize);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_SIZE, 'ROD1', fRodRadius, fRodRadius, dL1);
    ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
        KIN_MECH_CMD_SIZE, 'ROD2', fRodRadius, fRodRadius, dL2);
END_IF

ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
    KIN_MECH_CMD_POSITION, 'TCP', TCPx, TCPy, 0.0);
ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
    KIN_MECH_CMD_POSITION, 'MID', MIDx, MIDy, 0.0);
ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
    KIN_MECH_CMD_POSITION, 'ROD2', MIDx, MIDy, 0.0);
ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
    KIN_MECH_CMD_DIRECTION, 'ROD1', MIDx, MIDy, 0.0);
ML_AddMechanicData(MechanicData, MechanicDataSize, MechanicDataRetSize,
    KIN_MECH_CMD_DIRECTION, 'ROD2', MID2TCPx, MID2TCPy, 0.0);
```

Registering the transformation at a kinematics

It is assumed that a kinematics by the name of "K1" exists. For this kinematics, a "Scara" transformation is to be registered.

Note that the **instance** myScara is registered in the kinematics, not the function block type Scara.

For all other kinematics, an individual **instance** of the Scara function block has to be used.

Use

```
PROGRAM PlcProg
VAR
    myDeltaSlide      : DeltaSlide;
    setKinTrafo        : MC_SetKinTransform;
    initTransformation : BOOL;
END_VAR

IF(initTransformation) THEN
    setKinTrafo.KinTransform := myDeltaSlide;
    setKinTrafo.Execute      := TRUE;
    setKinTrafo(AxisGroup := K1);
    IF(setKinTrafo.Done) THEN
        initTransformation := FALSE;
        setKinTrafo.Execute := FALSE;
        setKinTrafo(AxisGroup := K1);
    END_IF
    IF(setKinTrafo.Error) THEN
        initTransformation := FALSE;
```

Application Examples

END_IF
END_IF

7.6 Sequential Programming with the Kinematics Interface

It is possible to implement a motion sequence with the kinematics interface if certain boundary conditions are observed. The following explains how this is done on the basis of an example.

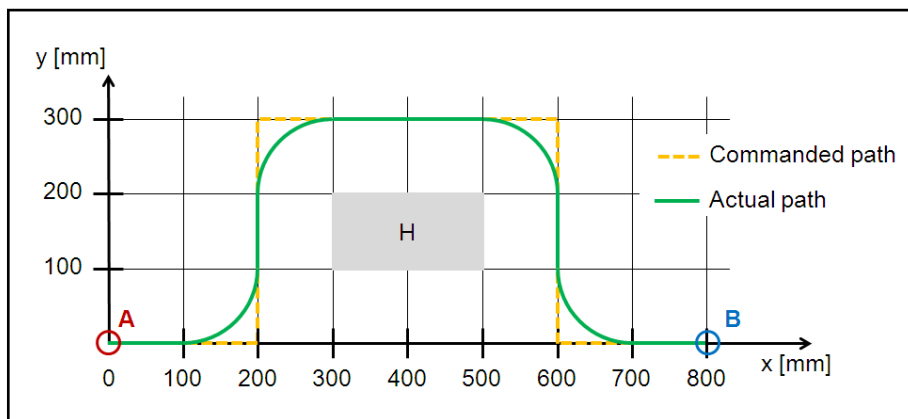


Fig.7-4: Sequential programming with the kinematics interface - motion path

A kinematics (a Cartesian in this case) is to travel from point A to point B around an obstacle H. For this purpose, the path illustrated in the figure above is specified.

The resulting positions are:

- A(0,0)
- P1(200,0)
- P2(200,300)
- P3(600,300)
- P4(600,0)
- B(800,0)

The positions are to be approached by linear, absolute blending motions. To implement this with the kinematics interface, the following program lines are required. The limit values for velocity, acceleration, deceleration and jerk were chosen freely.

Declaration

```

VAR
  uiState:      UINT := STATE_IDLE;
  bStartMove:   BOOL := FALSE;
  rVelocity:    REAL := 0.0;
  rAcceleration: REAL := 0.0;
  rDeceleration: REAL := 0.0;
  rJerk:        REAL := 0.0;
  arPoint1:     ARRAY[1..16] OF REAL := [200.0, 0.0, 14(0.0)];
  arPoint2:     ARRAY[1..16] OF REAL := [200.0, 300.0, 14(0.0)];
  arPoint3:     ARRAY[1..16] OF REAL := [600.0, 300.0, 14(0.0)];
  arPoint4:     ARRAY[1..16] OF REAL := [600.0, 0.0, 14(0.0)];
  arPointB:     ARRAY[1..16] OF REAL := [800.0, 0.0, 14(0.0)];
END_VAR

VAR CONSTANT
  STATE_IDLE:  UINT := 1;
  STATE_MOVE1: UINT := 2;
  STATE_MOVE2: UINT := 3;
  STATE_MOVE3: UINT := 4;
  STATE_MOVE4: UINT := 5;
  STATE_MOVE5: UINT := 6;
END_VAR

```


Program part

```
CASE uiState OF
STATE_IDLE:
  IF bStartMove THEN
    bStartMove := FALSE;
    uiState    := STATE_MOVE1;
  END_IF

STATE_MOVE1:
  (* Specify behavior after an operation mode change; always has to be
    set to "NoAction" for blending motions *)
  arKinCtrl_gb[1].Admin.ChangeOpMode := NoAction;

  (* arPoint1 = specify position P1*)
  arKinCtrl_gb[1].CoordMode.Point := arPoint1;

  (* specify limit values *)
  arKinCtrl_gb[1].CoordMode.Velocity    := 100;
  arKinCtrl_gb[1].CoordMode.Acceleration := 1000;
  arKinCtrl_gb[1].CoordMode.Deceleration := 1000;
  arKinCtrl_gb[1].CoordMode.Jerk       := 10000;

  (* specify coordinate system *)
  arKinCtrl_gb[1].CoordMode.CoordSystem := CS_MACHINE_COORDINATES;

  (* specify SetMode *)
  arKinCtrl_gb[1].CoordMode.SetMode := SM_VIA;

  (* specify SlopeType *)
  arKinCtrl_gb[1].CoordMode.SlopeType := ST_CONTINUOUS_SLOPE;

  (* specify blending radius *)
  arKinCtrl_gb[1].CoordMode.BlendingRadius := 100;

  (* Send linear traversing command;
    Block is set in block buffer *)
  arKinCtrl_gb[1].Admin._OpMode.en := ModeCoordPosLinAbs;

  (* before the next step: query "InBuffer" output *)
  IF arKinStatus_gb[1].Admin.InBuffer THEN
    uiState := STATE_MOVE2;
  END_IF

STATE_MOVE2:
  (* arPoint2 = specify position P2*)
  arKinCtrl_gb[1].CoordMode.Point := arPoint2;

  (* only input structures have to be described which
    have changed *)
  arKinCtrl_gb[1].CoordMode.Velocity := 20;

  (* a linear traversing command does not have to be send
    as this is automatically executed by changing
    the position value *)

  (* before the next step: query "InBuffer" output *)
  IF arKinStatus_gb[1].Admin.InBuffer THEN
    uiState := STATE_MOVE3;
  END_IF

STATE_MOVE3:
  (* arPoint3 = specify position P3*)
  arKinCtrl_gb[1].CoordMode.Point := arPoint3;

  (* before the next step: query "InBuffer" output *)
  IF arKinStatus_gb[1].Admin.InBuffer THEN
    uiState := STATE_MOVE4;
  END_IF

STATE_MOVE4:
  (* arPoint4 = specify position P4*)
  arKinCtrl_gb[1].CoordMode.Point := arPoint4;

  (* before the next step: query "InBuffer" output *)
  IF arKinStatus_gb[1].Admin.InBuffer THEN
    uiState := STATE_MOVE5;
  END_IF

STATE_MOVE5:
  (* arPointB = specify position B *)
  arKinCtrl_gb[1].CoordMode.Point := arPointB;
```

Application Examples

```
(* End *)
uiState := STATE_IDLE;
```

```
END_CASE
```

Block preparation, i.e. processing the SFC, is significantly faster than executing the motion. Therefore, the SFC may be in STATE_IDLE while the kinematics is still moving.



The status outputs of the kinematics interface always refer to the motion command issued **last**. In this example this is the motion to point B. Therefore, there is no way to evaluate the status of prior motions.

7.7 Registering a User-defined Transformation

If a user-defined transformation is used, i.e. transformation number 100, the transformation has to be registered in the PLC prior to switching phases to BB. Registration may be carried out easily via the kinematics interface. For this purpose, the transformation function block simply has to be connected to the "Admin.KinTransform" input of the kinematics interface. The following command starts transformation registration ("fbTransformation" corresponds to the transformation function block instance):

```
arKinCtrl_gb[1].Admin.KinTransformPLC := fbTransformation;
```



Registration is only possible in parameterization mode **P2**. If a transformation is connected to the kinematics interface input in another phase, registration only starts when phase 2 is reached.



If the kinematics interface is registered with a user-defined transformation for the first time, the control returns the error message "Transformation not registered". This is due to the IMC interface trying to switch to operating mode. When the transformation is registered, the error may be acknowledged.

The status output `arKinStatus_gb[1].Admin.KinTransformPLCDone` indicates that the transformation registration in the PLC was successful.

8 Parameter

8.1 Parameter, General Information

The IndraMotion MLC parameter system supports the implementation of kinematics on the control.

- Limit values and axis-specific data are stored in the A-parameters of the respective axis.
- The following [Control-Related Parameters, page 332](#) are created to manage all kinematic data of one control.
- The following [Kinematic-Related Parameters, page 334](#) are created to manage data of a kinematic in the control. Each parameter exists once in each kinematics.

8.2 Axis-Relevant Parameters

8.2.1 Axis-Relevant Parameters, General Information

Limit values and axis-relevant data to support a Robot Control V2 kinematic are stored in A-parameters of the respective axis.

8.2.2 A-0-0028, Travel range limit switch

Function This parameter defines the signal behavior of the travel range limit switch inputs and the behavior of the drive with regard to exceeded travel ranges.

Structure

Bit	Function
0	Signal behavior travel range limit switch inputs (real axes, only) 0: not inverted, 24 V $\Rightarrow$ travel range is exceeded, 1: inverted, 0 V $\Rightarrow$ travel range is exceeded.
1	Activation of travel range limit switch (real axes, only) 0: no, 1: yes.
2	Handling when travel range exceeded 0: as error (acc. to error reaction), 1: as fatal warning (shutdown).
15..3	reserved

8.2.3 A-0-0030, Position limit value, positive

Function An axis's permitted travel range can be defined by the position limit values. The maximum allowed positive actual position value (positive traveling direction) is entered in this parameter.

The positive limit value is only active if all position data refers to the axis zero point, i.e. the axis is referenced "A-0-0021, Axis status, bit 10".

Use If the "A-0-2200, Command position" of the axis is set outside the positive position limit value, the error bit in "A-0-0021, Axis status, bit 9" is TRUE and generates the error "F2112053, Command target position outside of travel range".

If the "A-0-0100, Actual position value" exceeds the "A-0-0030, Position limit value, positive" or the "A-0-0031, Position limit value, negative", the axis sets

Parameter

the error bit in the "A-0-0021, Axis status, bit 9" and generates the error "F2116029, Positive position limit value exceeded" or "F2116030, Negative position limit value exceeded".

See also Functional Description: "Position Limit Values".

8.2.4 A-0-0031, Position limit value, negative

Function An axis's permitted travel range can be defined by the position limit values. The maximum allowed negative actual position value (negative travel direction) is entered in this parameter.

The negative limit value is only active if all position data refers to the axis zero point, i.e. the axis is referenced "A-0-0021, Axis status, bit 10".

Use If the "A-0-2200, Command position" of the axis is set outside the negative position limit value, the error bit in "A-0-0021, Axis status, bit 9" is TRUE and generates the error "F2112053, Command target position outside of travel range".

If the "A-0-0100, Position value" exceeds the "A-0-0030, Position limit value, positive" or the "A-0-0031, Position limit value, negative", the axis sets the error bit in the "A-0-0021, Axis status, bit 9" and generates the error "F2116029, Positive position limit value exceeded" or "F2116030, Negative position limit value exceeded".

See also Functional Description: "Position Limit Values".

8.2.5 A-0-0032, Velocity limit value, positive

Function The parameter describes the maximum admissible velocity in the positive direction.

Use If the "A-0-0102, Actual velocity value" exceeds the absolute limit value "A-0-0032, Velocity limit value, positive" or "A-0-0033, Velocity limit value, negative", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112063, Velocity greater than limit A-0-0032 / A-0-0033".

The entered value is used as the maximum value for velocity variables in the positive direction.

See also Functional Description "Velocity Limitation".

8.2.6 A-0-0033, Velocity limit value, negative

Function The parameter describes the maximum admissible velocity in the negative direction.

Use If the "A-0-0102, Actual velocity value" exceeds the absolute limit value "A-0-0032, Velocity limit value, positive" or "A-0-0033, Velocity limit value, negative", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112063, Velocity greater than limit A-0-0032 / A-0-0033".

The entered value is used as the maximum value for velocity variables in the negative direction.

See also Functional Description "Velocity Limitation".

8.2.7 A-0-0034, Acceleration limit value, bipolar

Function The parameter describes the maximum admissible symmetric acceleration in both directions (acceleration and deceleration).

Use If the parameter "A-0-0103, Actual acceleration value" exceeds the parameter "A-0-0034, Acceleration limit value, bipolar", the axis sets the warning bit

Parameter

"A-0-0021, Axis status, bit 8" and generates the warning "E0112039, Maximum acceleration exceeded".

8.2.8 A-0-0036, Jerk limit value, bipolar

Function The parameter describes the maximum admissible acceleration change for each time interval (jerk) symmetrical in both directions (acceleration and deceleration).

This limit value counteracts the commanded jerk in the parameter "A-0-0216, Jerk".



The parameter limits the acceleration change for each time interval during "Drive hold" and for commands automatically generating position command values for themselves.

A value of "0" switches off the jerk limit value.

It is influenced by the parameter "A-0-0053, Scaling type for acceleration data".

8.2.9 A-0-0038, Torque/force limit value, bipolar

Function This parameter can be used to limit the torque/force command value to maximum admissible values to protect mechanical components in the drive train.

The parameter acts in a bipolar way. That means that positive and negative command values are limited to the entered value. The limit affects the total of all torque command values. That is the total of the command value parts of the velocity controller output and the command value parts from the acceleration-dependent feedforwards ("S-0-0348, Acceleration feedforward gain"; velocity control loop: "P-0-1126, Acceleration feedforward").

The unit of "A-0-0038" depends on the scaling set.

See also Parameter Description "A-0-0050, Scaling type for torque/force data"



The effective limit value is displayed in "P-0-0049, Effective torque/force command value". This value can be limited via "A-0-0038" due to a smaller value in "P-0-0109, Peak torque/force limitation" or due to a current limitation by a respective amplifier or motor load.

If the drive should be provided with a lower torque during stationary mode (constant speed) than during accelerations, this can be achieved with the respective values of "S-0-0082, Torque/force limit value, positive" and "S-0-0083, Torque/force limit value, negative" in connection with the acceleration feedforward (S-0-0348 or P-0-1126).

The torque/force limit value, bipolar also affects the "C1300 command: Travel to fixed stop".

The value of "A-0-0038" is a criterion for acknowledging "C1300".

See also Functional Description "Torque/Force Control".

8.2.10 A-0-0224, Rapid stop deceleration

Function In this parameter, the maximum deceleration value at which the axis is set to standstill at

- Emergency stop with ramp is set.

Parameter

The parameter unit is specified by "A-0-0053, Scaling type for acceleration data".

See also Functional Description: "Error Reaction"

8.2.11 A-0-0225, Rapid stop jerk

Function In this parameter, the maximum jerk value at which the axis
 • Emergency stop with ramp
 is set.

The parameter unit is specified by "A-0-0053, Scaling type for acceleration data".

See also Functional Description: "Error Reaction".



This parameter is only in effect if "Interpolation in the control" was selected for the axis.

8.2.12 A-0-0240, Feedrate override

Function The factor used to modify the axis velocity is set in this parameter.

A value of 100% means no modification.

If the factor is 0%, the warning "E0112055, Feedrate override A-0-0240 = 0" is output.

8.2.13 A-0-0241, Feedrate override control word

Function This parameter influences the feedrate override without modifying the Feedrate override parameter A-0-0240.

Structure

Bit	Designation / Function
0	Reserved
1	Feedrate override = 0
15-2	Reserved

Use Bit 1:

When bit 1 is set, a feedrate override of 0 becomes effective without modifying the actual A-0-0240 parameter value.



Feedrate override = 0 means "Halt", but in this case without the warning "E0112055, Feedrate override A-0-0240 = 0"

8.3 Control-Related Parameters

8.3.1 Control-Related Parameters, General Information

The following C-parameters are created to manage **all** kinematic data of one control.

8.3.2 C-0-0123, Maximum number of kinematics

Function This parameter indicates the maximum number of kinematics.

8.3.3 C-0-2480, Configured kinematic number

Function This parameter indicates the number of the currently configured kinematics.

8.3.4 C-0-2482, List of non-linked axes

- Function** This list parameter displays the axes that are not assigned to a kinematics yet.
If no kinematics is configured, this list corresponds to parameter "C-0-0483, List of configured axes".
The axes are assigned to the kinematics using parameter "K-0-0008, Axis configuration list".



An axis can only be assigned to one kinematics.

8.3.5 C-0-2483, List of configured kinematics

- Function** This parameter indicates the configured kinematics of the system.
- Use** Example:
3 kinematics (kinematics 1, 2 and 3) are configured in the system.
C-0-2483 indicates:

Element	Value	Comment
1	1	Kinematics 1 is configured
2	2	Kinematics 2 is configured
3	3	Kinematics 3 is configured

8.3.6 C-0-2484, Kinematic configuration list

- Function** This parameter defines the kinematics system. It configures the kinematics. Only K-parameters of kinematics configured in this parameter can be accessed.
The axes are assigned to the kinematics via the parameter "K-0-0008, Axes configuration list" (example K01: K-0-0008).

- Structure** Valid values of a kinematic configuration:

Value	Comment
0	No kinematic mechanism configured
1	Robot Control V1 kinematics configuration
2	Robot Control V2 kinematics configuration

- Use** If Robot Control V1 is used, the list may not have any gaps with unconfigured kinematics. This is checked when switching to phase 4.

If Robot Control V2 is used, gaps are allowed.

Otherwise, parameter "C-0-2484, Kinematics configuration list" is entered into parameter "C-0-0111, List of all invalid C-parameters" and an incorrect configuration is displayed (F0160024, Error in cross-check of C-parameters).

Writing on the parameters "C-0-2484, Configuration list of the kinematics" updates the parameters "C-0-2480, Configured number of kinematics" and "C-0-2483, List of the configured kinematics" .

Example A (Robot Control V1):

Three kinematics are to be configured (version 04V12 and higher).

C-0-2484 displays:

Parameter

Element	Value	Comment
1	1	Kinematics 1 is configured
2	1	Kinematics 2 is configured
3	1	Kinematics 3 is configured
4	0	No kinematics 4 used/required

Example B (Robot Control V2):

Three kinematics are to be configured (version 13V10 and higher).

C-0-2484 displays:

Element	Value	Comment
3	0	No kinematics used/required
4	2	Kinematics 3 is configured
5	0	No kinematics used/required

8.4 Kinematic-Related Parameters

8.4.1 Kinematic-Related Parameters, General Information

The following K-parameters are created to manage the data of a Robot Control V2 kinematics in the control. Each parameter exists once in each kinematics.

8.4.2 K-0-0001, Logic kinematic number

Function This parameter shows the actual logical kinematics number.

8.4.3 K-0-0002, Kinematic name

Function The name of the kinematics (e.g. Robi_1) is written in this parameter.

A global variable of type AXIS_GROUP_REF is automatically created with this name.



The following restrictions apply:

- maximum of 12 characters,
- A...Z, a...z, 0..9, '_'
- first character must be a letter,
- "Robi_1" is identical to "ROBI_1".

8.4.4 K-0-0003, Maximum path velocity

Function Maximum programmable path velocity in machine coordinates (MCS) at the Tool Center Point (TCP).

Depending on parameter K-0-0054, the entry is made in mm/s or in/s.

The parameter limits the maximum path velocity of the kinematics in the Linear and Circular motion types.

No limit is set in the Motion type PTP.

When the parameter value is reached, the warning E02C08CF is displayed. It is deleted when the value is undershot.

8.4.5 K-0-0005, Software limit switch, positive, MCS

Function This parameter determines the values of the positive software limit switch in machine coordinates (MCS).
Depending on the parameter K-0-0054, the entry is made in millimeters [mm] or inches [in] or degrees.

8.4.6 K-0-0006, Software limit switch, negative, MCS

Function This parameter determines the values of the negative software limit switch in machine coordinates (MCS).
Depending on the parameter K-0-0054, the entry is made in millimeters [mm] or inches [in] or degrees.

8.4.7 K-0-0007, Axis number of kinematics

Function This parameter contains the actual number of axes of the kinematics.
The value results from "K-0-0008, Axes configuration list".
It contains the sum of all there configured axes and belts.
Parameter "K-0-0070, Belt number of kinematics" contains only the sum of the belts.

8.4.8 K-0-0008, Axis configuration list

Function This parameter configures the axes for the kinematics.

Structure The parameter is a list parameter with the following triple structure:

- Element 1: Axis configuration of kinematic axis 1
- Element 2: Control number
- Element 3: Assigned axis for kinematic axis 1
- Element 4: Axis configuration of kinematic axis 2
- Element 5: Control number
- Element 6: Assigned axis for kinematic axis 2
- ...

The element number must be even.

Value	Comment
0	No axis configured
8	Positioning axis configured
16	Orientation axis configured
32	Belt axis configured

Tab.8-1: Valid values of an axis configuration

Value	Comment
0	Individual control
1-64	Control number in C2C link

Tab.8-2: Valid values for the control number

Use Structure of the triple list

The list may not contain any gaps.

One positioning axis has to be at least assigned to a kinematics.

Parameter

Orientation axes have to be located after the positioning axes.

Belt axes have to be located after positioning axes and orientation axes.

Based on the set parameter "K-0-0031 Transformation scheme," a different minimum number of positioning and orientation axes are required (see desired transformation description).

A maximum of four belt axes may be defined.

The triple list may configure a maximum total of 16 axes.

Assignment assistance

Each axis may be assigned to at most one kinematics as positioning or orientation axis.

Parameter "C-0-2482, List of non-linked axes ", lists the axes not yet assigned to a kinematics.

Kinematic axes configured as belt axes do not affect the parameter "C-0-2482, List of non-linked axes."

Belt axes can be selected from "C-0-0483, List of configured axes" and can be assigned to several kinematics.

Influence on other parameters

The number of triple entries in the list determines the parameter "K-0-0007, Axis number of kinematics."

The number of configured belt axes in the list determines the parameter "K-0-0070, Belt number of kinematics."



If "A-0-0059, Scaling type for position data " is set to modulo format for kinematic axes, it is also checked whether the "A-0-0045, Modulo value" is greater than zero and an integer.

With a Robot Control V2 kinematics, the system additionally checks if the scaling of the assigned axes is permissible.

If this is not the case, parameter "K-0-0008, Axis configuration list" is entered into parameter "K-0-0011, List of all invalid K-parameters" and an incorrect configuration is displayed (F0160040, Error in cross-check of K-parameters and/or F22B0024, Axis scaling is incorrect for the transformation).

Example:

The kinematics uses the transformation scheme 10 (SCARA) with four axes.

Axes 1, 2 and 3 are positioning axes.

Axis 4 is an orientation axis.

K-0-0008 displays:

Element	Value	Comment
1	8	Kinematic axis 1: Positioning axis
2	0	Control number 0: Individual control
3	1	Axis with logic axis number 1 assigned
4	8	Kinematic axis 2: Positioning axis
5	0	Control number 0: Individual control
6	2	Axis with logic axis number 2 assigned
7	8	Kinematic axis 3: Positioning axis

Element	Value	Comment
8	0	Control number 0: Individual control
9	3	Axis with logic axis number 3 assigned
10	16	Kinematic axis 4: Orientation axis
11	0	Control number 0: Individual control
12	4	Axis with logic axis number 4 assigned

The following axis types are available for positioning and orientation axes (see C-0-0484):

- Real axis
- Virtual axis
- controller axis.

For belt axes, the following are also available:

- encoder axis and
- link axis (cross link sercos II).

8.4.9 K-0-0009, Osci: list of all K-parameters that can be oscilloscoped

Function This parameter contains the list of all K parameters which can be oscilloscoped.

8.4.10 K-0-0010, List of all K-parameters

Function This list includes all K-parameters. The entry is made in ID number format. The entries are sorted by description and their relation and not by ID number or alphabetically.

K-parameters are stored in the order of these entries.

The ID number format is as follows:

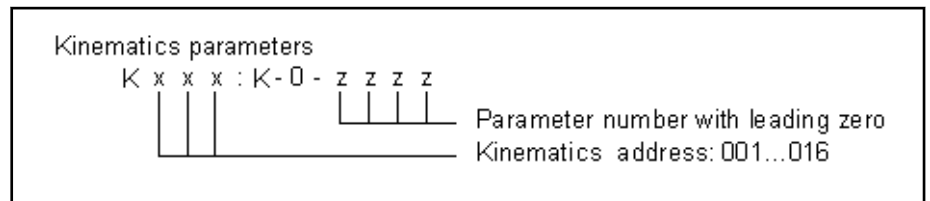


Fig. 8-1: K-parameter structure

Example:

K001: K-0-0002 - Kinematics 1; parameter K-0-0002.

8.4.11 K-0-0011, List of all invalid K-parameters

Function This list displays all invalid K-parameters in the result

- of a checksum test, e.g. after a firmware update, or
- a plausibility test, e.g. in the result of an illegal operation mode.

The entry is made in ID number format.

Parameter

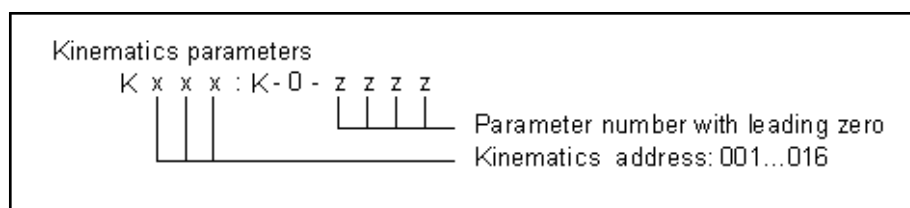


Fig.8-2: K-parameter structure

Use K001: K-0-0008 - Kinematics 1; parameter K-0-0008.

8.4.12 K-0-0012, Checksum of all K-parameters

Function When reading this parameter the checksum is generated by all K parameter values the IDNs of which are listed in parameter "K-0-0013, IDN list of checksum K parameter".

8.4.13 K-0-0013, IDN list of checksum K-parameters

Function The parameter's ID numbers whose checksum is to be formed are entered in this parameter.

The checksum formation is triggered when the "K-0-0012, Checksum of all C-parameters" is read out.

In the default state, no ID numbers are entered in K-0-0013.

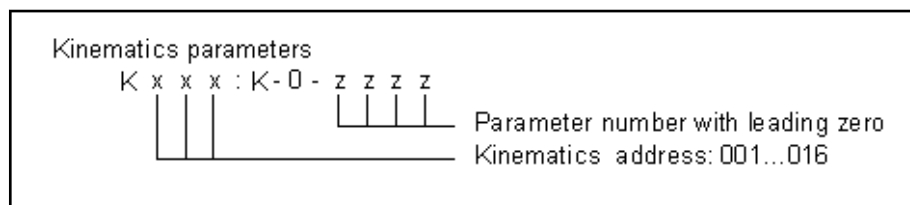


Fig.8-3: Structure of a K-parameter

Use K001: K-0-0008 - cinematics 1; parameter K-0-0008.

8.4.14 K-0-0016, Project identification number

Function IndraWorks entered a series of unique characters in this parameter to identify worldwide the project.

This series of characters is one to one for each project.

This parameter has no influence on the functions of the Motion Logic Control.



This parameter is for internal purposes only.

8.4.15 K-0-0019, Kinematic type

Function This parameter indicates the kinematic type.

Structure

Bit	Content	Bit name
2..0	001: Robot Control V1 010: Robot Control V2	Kinematic type
31..3	Reserved	

Tab.8-3: K-0-0019 Kinematic type

8.4.16 K-0-0020, Kinematic diagnostic message

Function These parameters contain the current diagnostic message for the kinematic (message, warning or error).

Parameter

If bit 8 or bit 9 of the "A-0-0021, Kinematic status" parameter is set to 1, this parameter contains either the warning text (bit 8) or the error text (bit 9).

From the perspective of the PLC, this information is available to the kinematic "*" as

- KinData[*].dwKinStatus_i.8= TRUE, diagnostic message in K-0-0020 is a warning text or
- KinData[*].dwKinStatus_i.8= TRUE, diagnostic message in K-0-0020 is an error text.



If no error or warning has been reported in the kinematics, the current kinematic state is displayed as text.

8.4.17 K-0-0021, Kinematic status

Function This parameter describes the current status of the kinematics.
This is internal system information that is triggered via PLC function blocks for example. This is not information on the state of the kinematics state machine.

Structure

Bit	Content	Bit name
5	All kinematic axes are in standstill. (Belt axes are not considered)	Kin_StandStill
6	Command position = actual position The programmed end position is reached.	Kin_InPosition
7	Belt-synchronous	Kin_BeltSynchronous
8	Warning	Kin_Warning
9	Error	Kin_Error
10	Homing required; not all axes are referenced.	Kin_NotHomed
11	Blending	Kin_Blending
14	All axes are ready for operation (AH or AF)	Kin_DrivesOn
17	Error reaction enabled	Kin_ErrorReaction
28	Spline is calculated for blending motion	
29	Spline is valid	

Tab.8-4: K-0-0021 Kinematic status

Use **Bit 5: All kinematic axes are in standstill**

This bit signals that the axes positioned by the kinematics are in standstill (A-0-0021, bit 5). The standstill window A-0-0220 is taken into consideration).

Bit 6: The programmed end position is reached

No active Motion command available.

Bit 7: Belt-synchronous

Partial or fully synchronized traversing to a belt.

Bit 8: Warning

A warning is reported.

Bit 9: Error

An error is reported.

Parameter

Bit 10: Homing required

At least one kinematic axis is not referenced.

Bit 11: Blending

Blending is being executed.

Bit 14: All axes ready for operation

All axes of the kinematics are ready for operation (AH or AF).

Bit 17: Error reaction active

An error reaction is active.

Bit 28: Spline is calculated for blending motion

Calculating spline for blending motion.

Bit 29: Spline is valid

The spline calculated for the blending motion is valid. The bit remains active until the spline has been traversed.

8.4.18 K-0-0022, Extended kinematic status

Function This parameter uses several status bits to describe the current state of the kinematic state machine . Each state is assigned to a status bit combination in the parameter K-0-0022.

	Kinematic state machine is in
Bit 0: Kis_Errorstop	"Errorstop" or "Errorstop Stopping"
Bit 1: Kis_Stopping	"Errorstop Stopping" or "Stopping"
Bit 3: Kis_Incomplete	"Standstill incomplete"
Bit 7: Kis_StandStill	"Standstill Complete" or "Standstill Incomplete"
Bit 10: Kis_ManualMotion	"Manual Motion"
Bit 12: Kis_GroupMoving	"Group Moving"

Structure**8.4.19 K-0-0023, Kinematic diagnostic number**

Function If bits 8 or 9 of parameter "K-0-0021, Kinematic status" are set, the current diagnostic number can be read this parameter.

8.4.20 K-0-0031, Transformation scheme**Function**

The transformation scheme is set in this parameter.

More settings for the transformation scheme can be selected in the parameters K-0-0040 and K-0-0041.

8.4.21 K-0-0034, Maximum path acceleration (bipolar)

Function Maximum programmable path acceleration in machine coordinates (MCS) at the Tool Center Point (TCP).

Depending on the parameter K-0-0054, the entry is made in mm/s or in/s.

The parameters limits the maximum path acceleration of the kinematics in the Motion types Linear and Circular.

No limit is set in the Motion type PTP.

If the parameter value is reached, the warning E02C08CE is reported and re-set if fallen below.

8.4.22 K-0-0035, Maximum path jerk (bipolar)

- Function** Maximum programmable path jerk in machine coordinates (MCS) at the Tool Center Point (TCP).
Depending on parameter K-0-0054, the entry is made in mm/s³ or in/s³.
The parameters limits the maximum path jerk of the kinematics in the Motion types Linear and Circular. No Motion takes place in the Motion type PTP.
If the parameter value is reached, the warning E02C08CD is reported and re-set if fallen below.

8.4.23 K-0-0040, Transformation, lengths of the mechanics

- Function** In this parameters, the lengths for the transformation scheme in parameter K-0-0031 are configured.
The meaning of the individual values depends on the set K-0-0031, Transformation scheme.

8.4.24 K-0-0041, Transformation, factors

- Function** This parameter configures the factors for the transformation scheme in parameter K-0-0031.
The meaning of the individual values depends on the transformation scheme set in K-0-0031.

8.4.25 K-0-0042, Offset of MCS system

- Function** In this parameter, the offset and the rotation of the basic coordinate system (BCS) with regard to the machine coordinate system (MCS) is selected.
The values are always specified in the machine coordinate system.

- Structure** This parameter is a list parameter with six values:

Element	Description
1	Displacement in Cartesian x-direction [in millimeters or inches (see K-0-0054)]
2	Displacement in Cartesian y-direction [in millimeters or inches (see K-0-0054)]
3	Displacement in Cartesian z-direction [in millimeters or inches (see K-0-0054)]
4	Rotation along the Cartesian x-axis [in degrees] (see note)
5	Rotation along the Cartesian y-axis [in degrees] (see note)
6	Rotation along the Cartesian z-axis [in degrees] (see note)

- Use** The first three values indicate the position of the coordinate system origin from the perspective of the machine coordinate system. The elements 4 to 6 indicate the rotation of the coordinate system.



The rotation of the basic coordinate system is only possible in degrees of freedom supported by the transformation scheme.

For example, the scara transformation scheme only contains the degree of freedom by the Z-axis as orientation. Only a rotation of the basic coordinate system by the Z-axis is possible.

Parameter

8.4.26 K-0-0050, Distance units MCS

Function The distance and orientation units for each element of a list parameter in the machine coordinate system (MCS) are separated by a command and saved in this text parameter.

- Only legible entries are automatically determined.
- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the transformation type K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm or inch (see K-0-0054)
 - For alpha, beta, gamma: Degrees

Use *Examples*

1. Cartesian robot with X, Y, Z, C
K-0-0050 = mm,mm,mm,degree or inch,inch,inch,degree (see K-0-0054)
2. Scara with lifting and rotating spindle (X, Y, Z, C)
K-0-0050 = mm,mm,mm,degree or inch,inch,inch,degree (see K-0-0054)
3. Cartesian robot with X, Y, Z, C, B
K-0-0050 = mm,mm,mm,degree, degree or inch,inch,inch,degree, degree (see K-0-0054)
4. Cartesian robot with X, Y, C
K-0-0050 = mm,mm,degree or inch,inch,degree (see K-0-0054)
5. Delta with C-axis and 2 belts
K-0-0050 = mm,mm,mm,degree or inch,inch,inch,degree (see K-0-0054)



The belts are not included in the lists for the machine coordinate system (MCS).

8.4.27 K-0-0051, Velocity units MCS

Function The velocity units for each element of a list parameter in the world coordinate system are separated with a command and saved in this text parameter.

- Only legible entries are automatically determined.
- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the transformation type K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/s or in/s (see K-0-0054)
 - For alpha, beta, gamma: Degree/s

Use *Examples*

1. Cartesian robot with X, Y, Z, C
K-0-0051 = mm/s, mm/s, mm/s, degree/s or inch/s, inch/s, inch/s, degree/s (see K-0-0054)
2. Scara with lifting and rotating spindle (X, Y, Z, C)
K-0-0051 = mm/s, mm/s, mm/s, degree/s or inch/s, inch/s, inch/s, degree/s (see K-0-0054)

3. Cartesian robot with X, Y, Z, C, B
K-0-0051 = mm/s, mm/s, mm/s, degree/s, degree/s or inch/s, inch/s, inch/s, degree/s, degree/s (see K-0-0054)
4. Cartesian robot with X, Y, C
K-0-0051 = mm/s, mm/s, degree/s or inch/s, inch/s, degree/s (see K-0-0054)
5. Delta with C-axis and 2 belts
K-0-0051 = mm/s, mm/s, mm/s, degree/s or inch/s, inch/s, inch/s, degree/s (see K-0-0054)



The belts are not included in the lists for the machine coordinate system (MCS).

8.4.28 K-0-0052, Acceleration units MCS

Function The acceleration or deceleration units for each element of a list parameter in the world coordinate system are separated with a command and saved in this text parameter.

- Only legible entries are automatically determined.
- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the transformation type K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/s² or inch/s²
 - For alpha, beta, gamma: degree/s²

Use *Examples*

1. Cartesian robot with X, Y, Z, C
K-0-0052 = mm/s², mm/s², mm/s², degree/s² or inch/s², inch/s², inch/s², degree/s² (see K-0-0054)
2. Scara with lifting and rotating spindle (X, Y, Z, C)
K-0-0052 = mm/s², mm/s², mm/s², degree/s² or inch/s², inch/s², inch/s², degree/s² (see K-0-0054)
3. Cartesian robot with X, Y, Z, C, B
K-0-0052 = mm/s², mm/s², mm/s², degree/s², degree/s² or inch/s², inch/s², inch/s², degree/s², degree/s² (see K-0-0054)
4. Cartesian robot with X, Y, C
K-0-0052 = mm/s², mm/s², degree/s² or inch/s², inch/s², degree/s² (see K-0-0054)
5. Delta with C-axis and 2 belts
K-0-0052 = mm/s², mm/s², mm/s², degree/s², or inch/s², inch/s², inch/s², degree/s² (see K-0-0054)



The belts are not included in the lists for the machine coordinate system (MCS).

8.4.29 K-0-0053, Jerk units MCS

Function The jerk units for each element of a list parameter in the world coordinate system are separated with a command and saved in this text parameter.

Parameter

- Only legible entries are automatically determined.
- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the transformation type K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/s³ or inch/s³
 - For alpha, beta, gamma: degree/s³

Use Examples

1. Cartesian robot with X, Y, Z, C
K-0-0053 = mm/s³, mm/s³, mm/s³, degree/s³ or inch/s³, inch/s³, inch/s³, degree/s³ (see K-0-0054)
2. Scara with lifting and rotating spindle (X, Y, Z, C)
K-0-0053 = mm/s³, mm/s³, mm/s³, degree/s³ or inch/s³, inch/s³, inch/s³, degree/s³ (see K-0-0054)
3. Cartesian robot with X, Y, Z, C, B
K-0-0053 = mm/s³, mm/s³, mm/s³, degree/s³, degree/s³ or inch/s³, inch/s³, inch/s³, degree/s³, degree/s³ (see K-0-0054)
4. Cartesian robot with X, Y, C
K-0-0053 = mm/s³, mm/s³, degree/s³ or inch/s³, inch/s³, degree/s³ (see K-0-0054)
5. Delta with C-axis and 2 belts
K-0-0053 = mm/s³, mm/s³, mm/s³, degree/s³ or inch/s³, inch/s³, inch/s³, degree/s³ (see K-0-0054)



The belts are not included in the lists for the machine coordinate system (MCS).

8.4.30 K-0-0054, Measuring unit for kinematic data

Function This parameter determines the scaling unit of the kinematics.

Structure The following settings can be made:

Bit	Name/function
0	Unit 0: millimeter [mm] 1: inch [in]
15..1	Reserved

8.4.31 K-0-0070, Belt number of kinematics

Function This parameter contains the current number of belts of the kinematics.
The value results from "K-0-0008, Axes configuration list". It only contains the sum of there configured belts.
The parameter "K-0-0007, Axis number of kinematics" contains the sum of all axes and belts.

8.4.32 K-0-0096, Peak value of command buffer size

Function This parameter shows the maximum value of the command buffer size that has been reached. By writing smaller values, the value can be reset.

8.4.33 K-0-0097, Maximum command buffer size

Function This parameters specifies the maximum number of commands that can simultaneously be saved in the block buffer.

8.4.34 K-0-0098, Current command buffer size

Function This parameter shows the current number of commands in the block buffer.

8.4.35 K-0-0099, Kinematic configuration

Function This parameter configures the behavior of the kinematics.
Structure

Bit	Name/function
0	Enables automatic switching from CONT_SLOPE to BLOCK_SLOPE
1	Enables the axis modulo of each belt axis as belt modulo
2	Enables the complete radius to blend with BLOCK_SLOPE
3	Enables the option to be synchronous with multiple belts at the same time
4	Enables the option that more than two commands can simultaneously be active in BLOCK_SLOPE
5..31	Reserved

Use

Bit 0: Enables automatic switching from CONT_SLOPE to BLOCK_SLOPE
If the automatic switching is active, it is switched to the blending method BLOCK_SLOPE if blending with CONT_SLOPE failed.

Bit 1: Enables the axis modulo of each belt axis as belt modulo
If bit 1 is set, the axis modulo of each belt axis is used as belt modulo value.
Caution: The belt operating are can be max. belt modulo/2.

Bit 2: Enables the complete radius for blending with BLOCK_SLOPE
If bit 2 is set, the blending radius for BLOCK_SLOPE is not limited anymore to the minimum of radius and path distance/2 of the shorter block.

Bit 3: Enables the option to be synchronous with multiple belts at the same time
If bit 3 is set, multiple belts can simultaneously be synchronous. The belts have to be positioned linearly independent. I.e., in an angle of 90 degrees.

Bit 4: Enables the option that more than two commands can simultaneously be active in BLOCK_SLOPE
If bit 4 is set, more than two commands can simultaneously be active in BLOCK_SLOPE. Thus, more than 2 commands are blended. A third command is also started when it is in the blending radius and the first command has not been completed. Note that the resulting path is not very defined due to blending of more than two commands. E.g., two countercurrent blocks can be canceled.

8.4.36 K-0-0100, Current position, Cartesian Machine Coordinate System

Function The parameter indicates the current position of the kinematics in the machine coordinaten system (MCS).

Parameter

The current position consists of a list of Cartesian coordinates, the orientation and the belt counters, depending on the transformation scheme (K-0-0031) and the axis configuration (K-0-0008).

8.4.37 K-0-0102, Current path velocity

- Function** This parameter provides the current path velocity of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.
- The unit is mm/s or in/s (see K-0-0054).

8.4.38 K-0-0103, Current path acceleration

- Function** This parameter provides the current path acceleration of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.
- The unit is mm/s² or in/s².

8.4.39 K-0-0104, Current path jerk

- Function** This parameter provides the current path jerk of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.
- The unit is mm/s³ or in/s³.

8.4.40 K-0-0150, Interpolated command position

- Function** The parameter indicates the current interpolated command position of the kinematics in the machine coordinate system (MCS).
- Depending on the transformation scheme (K-0-0031) and the axis configuration (K-0-0008), the current position consists of a list of Cartesian world coordinates, the orientation, and the belt counters.

8.4.41 K-0-0152, Interpolated path velocity

- Function** This parameter provides the current interpolated path velocity of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.
- The unit is mm/s or in/s (see K-0-0054).

8.4.42 K-0-0153, Interpolated path acceleration

- Function** This parameter provides the current interpolated path acceleration of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.
- The unit is mm/s² or in/s².

8.4.43 K-0-0154, Interpolated path jerk

- Function** This parameter provides the current interpolated path jerk of the kinematics at the TCP.
- Only the positioning axes and not the orientation and belt axes are considered.

The unit is mm/s³ or in/s³.

8.4.44 K-0-0223, Selection of error reaction

Function This parameter can be used to determine the behavior of the kinematics in case of an error.

Structure

Bit	Name/function
0	0: The limit values of the active command are used 1: The limit values of the parameters K-0-0224, Emergency deceleration, and K-0-0225, Emergency jerk, are used
1..31	Reserved

Use This parameters is only available from Robot Control V2.

Bit0= 0: The limit values of the active command are used

In case of an error, the kinematics stops with the limit values of the active command. If a different command becomes active during an error reaction (e.g. by blending), the limit values of the new active command are used.

Bit0= 1: The limit values of the parameters K-0-0224, Emergency deceleration, and K-0-0225, Emergency jerk, are used

In case of an error, the kinematics stops with the limit values of the parameters K-0-0224, Emergency deceleration, and K-0-0225, Emergency jerk. These limit values are only used if they are higher than currently active command. The emergency deceleration and the emergency jerk are considered separately: If the programmed deceleration is lower than the emergency deceleration, the emergency deceleration is used. If the programmed jerk is higher than the emergency jerk, the programmed jerk is used.

8.4.45 K-0-0224, Emergency halt deceleration

Function The parameter determines the emergency deceleration. The parameter is only applied if bit 0 is set in the parameter K-0-0223, Selection of error reaction.

8.4.46 K-0-0225, Emergency halt jerk

Function The parameter determines the emergency jerk. The parameter is only applied if bit 0 is set in the parameter K-0-0223, Selection of error reaction.

8.4.47 K-0-0240, Feedrate override

Function

The command velocity at runtime can be changed percentaged to the programmed value using this parameter.

A value of 100% indicates that the programmed velocity is used to traverse the axes.

The percentage value can be set in a range from 0% to 600%.



In case of 0%, the axes stop without changing the state of the kinematic state machine.

8.4.48 K-0-0241, Feedrate override control word

Function This parameter controls the feed rate override.

Parameter

Structure

Bit	Name/function
0	Reserved
1	Bit 1 = 0: Feed rate override in K-0-0240 is active Bit 1 = 1: Feed Halt, the value in K-0-0240 is retained.
15..2	Reserved

Use **Bit 1:**

Feed halt becomes active by setting bit 1.

8.4.49 K-0-0242, Acceleration override

Function

The acceleration at runtime can be changed in percentage to the programmed value using this parameter.

A value of 100% means that the axes travel at the specified (programmed) acceleration.

The percentage can be set at a range from 1% to 600%.



Setting of K-0-0242 if active in the block preparation, thus, not immediately.

8.4.50 K-0-0243, Deceleration override

Function

The deceleration at runtime can be changed in percentage to the programmed value using this parameter.

A value of 100% means that the axes travel at the specified (programmed) deceleration.

The percentage can be set at a range from 1% to 600%.



Setting of K-0-0243 if active in the block preparation, thus, not immediately.

Smaller values in K-0-0243 result in deceleration distances that are too long.

8.4.51 K-0-0250, Continuous slope lead time

Function This parameter specifies the lead time to calculate the blending motion for CONT_SLOPE.

Use The lead distance results from the lead time and the path velocity.

If late blending is enabled due to a late issuing of a traversing command, the current position is saved.

If the turn off point (lead distance + saved current position) is exceeded at the end of the blending motion calculation, an error is output.

If the turn off point is not exceeded, the blending motion (of the blending radius is big enough) is started at the turn off point.

8.4.52 K-0-0815, Odometer

Function This parameter shows the traversed distance of the "Tool Center Points" (TCP) in kilometers.

8.4.53 K-0-1000, Command: Load basic kinematic parameters

Function This parameter is used as a command for loading the basic parameters of a kinematics.

Use *From IndraWorks MLC:*

- Use the button "Load basic parameters" in the **context menu of the kinematics** ► **Parameter handling** ► **Load Basic Parameters**.

From the field bus:

- Enter the binary value "0000000000000011" (=16#0003) into parameter (K-0-1000).

This activates the loading procedure of the basic parameters of the kinematics.

- After the loading procedure, the MLC automatically resets the parameter to "16#00".

8.4.54 K-0-2100, List of all K-parameters to be backed up

Function All K-parameters that have to be considering during parameter backup are listed in this list. The entry is made in ID number format.

The entries are sorted by description and their relation and not by ID number or alphabetically.

K-parameters are stored in the order of these entries.

The ID number format is as follows:

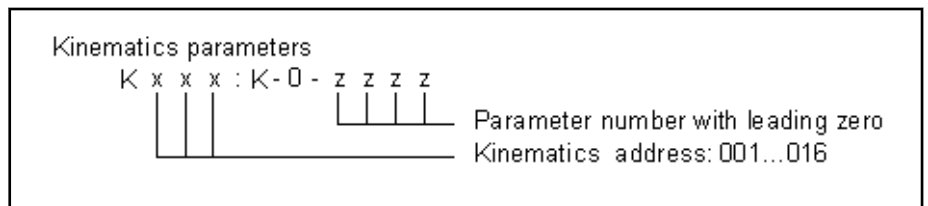


Fig. 8-4: K-parameter structure

Example:

K001: K-0-0002 - Kinematics 1; parameter K-0-0002.

9 Diagnostics

9.1 Diagnostics, General Information

The IndraMotion MLC diagnostic system supports working with kinematics on the control.

The following tables contain essential error messages of IndraMotion MLC.

All messages, warnings and errors are described and contained in

Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnostics

DOK-XLCMLC-DIAG****V13-RERS-EN-P, R911336358

9.2 MLC_TABLE, ERROR_TABLE 16#0030

MLC_TABLE - ERROR_TABLE, Table number 16#0030

The entries in the following table match the MLC troubleshooting guide (diagnostics).

ErrorIdent / Additional1		Diagnostics
F0220200	F2229200	General drive error
F0220201		Invalid control number
F0220202		Invalid axis number
F0220203	F2229203	Invalid command type
F0220204	F2229204	Command not supported by axis
F0220205	F2229205	Command rejected, no power
F0220206	F2229206	Command rejected while 'Stopping'
F0220207	F2229207	Command rejected while 'Homing'
F0220208	F2229208	Invalid command number
F0220210	F2229210	Reset still 'ErrorStop'
F0220211	F2229211	Command rejected while 'Diskrete Motion'
F0220212	F2229212	Command rejected while 'Continuous Motion'
F0220213	F2229213	Command rejected while 'Synchronized Motion'
F0220214	F2229214	Command rejected while 'ErrorStop'
F0220216	F2229216	No power available
F0220217	F2229217	Command rejected while 'StandStill' and 'PowerOn'
F0220218	F2229218	Command rejected while 'StandStill'
F0220223	F2229223	Velocity below minimum
F0220224	F2229224	Velocity above maximum
F0220225	F2229225	Acceleration below minimum
F0220226	F2229226	Acceleration above maximum
F0220227	F2229227	Deceleration below minimum
F0220228	F2229228	Deceleration above maximum
F0220229	F2229229	Jerk below minimal

Diagnostics

ErrorIdent / Additional1		Diagnostics
F0220230	F2229230	Jerk above maximum
F2220231		Invalid master axis data
	F2229231	Invalid direction
F0220232	F2229232	Invalid master control number
F0220233	F2229233	Invalid master axis number
F0220234	F2229234	Invalid CamTable ID
	F2229235	Error in MotionProfile
F0220263	F2229263	RatioNumerator below minimum
F0220264	F2229264	RatioNumerator above maximum
F0220265	F2229265	RatioDenominator below minimum
F0220266	F2229266	RatioDenominator above maximum
F0220267	F2229267	MasterFineAdjust below minimum
F0220268	F2229268	MasterFineAdjust above maximum
F0220269	F2229269	Invalid SyncMode
F0220270	F2229270	Invalid StartMode
F0220271	F2229271	Cam distance below minimum value
F0220272	F2229272	Cam distance above maximum value
F0220273	F2229273	Command rejected while 'Parameter Mode'
F0220274	F2229274	Axis not referenced
F0220275	F2229275	Command aborted
F0220276	F2229276	Control error
F0220277	F2229277	Direct master axis change
F0220278	F2229278	Command 'Disabled axis' active
F0220279	F2229279	Command 'parking axis' active
F0220280	F2229280	Command rejected while 'Coordinated Motion'
F0220281	F2229281	Invalid float data
F0220282	F2229282	MotionProfile, SetSelection below minimum value
F0220283	F2229283	MotionProfile, SetSelection above maximum value
	F2229284	Cam table is write-protected
	F2229285	CAM table, usage counter out of range
	F2229286	Choice relative positioning above max. value
F0220287		Unreferenced cam table object
	F2229287	Can not assign axis to kinematics
F0220288		ChangeCamTable: above maximum value
F0220289		ChangeCamTable: below minimum value
F0220290		ChangeCamTable: invalid ID

Diagnostics

ErrorIdent / Additional1	Diagnostics
F0220291	Unreferenced MotionProfile object
F0220292	ChangeMotionProfileSet: SetNumber below minimum value
F0220293	ChangeMotionProfileSet: SetNumber above maximum value
F0220294	ChangeMotionProfileSet: Selected profile write-protected
F0220295	ChangeMotionProfileStep, invalid StepNumber
	F2229296 Absolute entry point in FlexProfile not found
	F2229297 Absolute entry in FlexProfile not possible
	F2229298 Master velocity of FlexProfile is negative

Tab.9-1: Diagnostics of the Motion function blocks (axis state) I



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1	Diagnostics
F0220300	F2229300 Invalid command channel
F0220301	F2229301 Command rejected while 'Controlled Motion'
	F2229310 Error in calculation of FlexProfile step
	F2229311 FlexProfile: Invalid switching position
	F2229312 FlexProfile: Invalid FlexProfile startmode
	F2229313 FlexProfile: Invalid FlexProfile slave axis offset
	F2229314 Sum of dist. not 0 or mod. value, cyclic exec. not pos.
	F2229315 Invalid FlexProfile: Execution not possible
F0220316	ChangeFlexProfile: Length of FlexProfile lists inconsist.
F0220317	ChangeFlexProfile: No steps in profile
F0220318	ChangeFlexProfile: Unknown master
F0220319	ChangeFlexProfile: Motion law unknown
F0220320	ChangeFlexProfile: Step mode unknown
F0220321	ChangeFlexProfile: Step range 0 or negative
F0220322	ChangeFlexProfile: Invalid comb. of boundary value types
F0220323	ChangeFlexProfile: Invalid boundary values
F0220324	ChangeFlexProfile: Invalid Hub
F0220325	ChangeFlexProfile: Invalid value for turnaround point disp
F0220326	ChangeFlexProfile: Invalid slope
F0220327	ChangeFlexProfile: Invalid Cam table
F0220328	ChangeFlexProfile: Motion law + selected step mode
F0220329	ChangeFlexProfile: Velocity jump in profile

Diagnostics

ErrorIdent / Additional1		Diagnostics
F0220330		ChangeFlexProfile: Acceleration jump in profile
F0220331		ChangeFlexProfile: Jerk jump in profile
F0220332		ChangeFlexProfile: Velocity limit invalid
F0220333		ChangeFlexProfile: Acceleration limit invalid
F0220334		Unreferenced FlexProfile object
F0220335	F2229335	FlexProfile: Set is write-protected
	F2229336	FlexProfile: Invalid step number
	F2229337	FlexProfile: Set number out of range
	F2229338	ChangeFlexProfile: Invalid nominal master velocity
	F2229339	ChangeFlexProfile: invalid operation mode
F0220341	F2229341	FlexEvent: Set is write-protected
	F2229342	FlexEvent: Set number out of range
F0220343		ChangeFlexEvent: Length of FlexProfile lists inconsistent
F0220344		ChangeFlexEvent: Step Number assignment is invalid
F0220345		ChangeFlexEvent: Trigger-Mode is invalid
F0220346		ChangeFlexEvent: Action-Mode is invalid
	F2229347	FlexProfile: Invalid. Single execution not possible
	F2229348	FlexProfile: Invalid. Cyclic execution not possible
F0220349		ChangeFlexProfile: Jerk limit invalid
F0220350		ChangeFlexProfile: Limit too low
	F2229351	Synchronisation velocity below minimum
	F2229352	Synchronisation velocity above maximum
	F2229353	Synchronisation acceleration below minimum
	F2229354	Synchronisation acceleration above maximum
F0220355		ChangeFlexProfile: Invalid sequence of Flex-Steps
F2220356		Invalid filter time const., additive master position
F2220357		Invalid filter time const., angle offset begin of profile
F2220358		Invalid filter time const., additive slave position
	F2229359	FlexProfile: Invalid master axis offset
F0220360	F2229360	Command rejected while 'Controlled Motion'
F0220361		ChangeFlexProfile: Invalid master axis source
	F2229361	Command not supported by axis (Configuration)
	F2229362	Command not supported by axis (Functional Package)
	F2229363	MotionProfile invalid

Tab.9-2: Diagnostics of the Motion function blocks (axis state) II

Diagnostics



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1	Diagnostics
F0220501	Wrong data type
F0220502	ProbeModule out of range
F0220503	ProbeInput out of range
F0220504	ExpectWindowPosEdgeMode out of range
F0220505	ExpectWindowNegEdgeMode out of range
F0220506	ExpectWindowPosEdgeStart out of range
F0220507	ExpectWindowPosEdgeEnd out of range
F0220508	ExpectWindowNegEdgeStart out of range
F0220509	ExpectWindowNegEdgeEnd out of range
F0220510	ExpectWindowPosEdgeSetPoint out of range
F0220511	ExpectWindowNegEdgeSetPoint out of range
F0220512	MarkerFailureEnable out of range
F0220513	MaxNumberOfMarkerFailure out of range
F0220514	Interpolation out of range
F0220515	TimeStampEnable out of range
F0220516	DeadTimePosEdge out of range
F0220517	DeadTimeNegEdge out of range
F0220518	ProbeData out of range
F0220519	PositionEvalMode out of range
F0220520	DiffEvalMode out of range
F0220521	TOUCHPROBE_REF is not valid or wrong configuration
F0220522	AXIS_REF is not valid or wrong configuration of axis

Tab.9-3: Diagnostics of the Motion function blocks (axis state) III



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1	Diagnostics
F0220600	Path / file name unknown
F0220601	Unsupported file system
F0220602	Input 'StartIdx' has an invalid value
F0220603	Invalid state of state machine

Diagnostics

ErrorIdent / Additional1	Diagnostics
F0220604	Error in file structure
F0220605	Input 'ArrayDimension' has an invalid value
F0220606	Point not found
F0220607	Wrong number of coordinates per point
F0220608	Wrong number of kinematics
F0220609	Wrong point type, point in world or joint coordinates
F2220610	Invalid master axis

Tab.9-4: Diagnostics of the Motion function blocks (axis state) IV



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1	Diagnostics
F00D1002	Cyclic position channel already open
F00D1003	Cyclic force channel already open
F00D1004	Cyclic velocity channel already open
F00D1005	Cyclic analog channel already open
F00D100A	Error while reading C-0-0483
F00D100B	Error while reading C-0-0484
F00D100C	Error while opening cyclic channel
F00D100D	Error while opening cyclic channel
F00D1014	No reference for opening cyclic channel available
	F20D1015 Error while closing cyclic channel
F00D1016	Wrong #-address of axis data
F00D1017	Wrong #-address of axis data
F00D1018	Wrong #-address of axis data
F00D1019	Error while closing cyclic channel
	F20D1025 Error while closing cyclic channel
F00D1027	Error while closing cyclic channel
	F20D1029 Error opening cyclic channel
F00D102A	ListParameter requested, ReadParameter
	F20D102B Error opening cyclic channel (READ-ACCESS)
	F20D102C Error reading access (DirectReadReal)
	F20D102D Error reading access (DirectReadWord)
	F20D102E Error reading access (DirectReadDword)
F00D1030	Error while closing cyclic channel

Diagnostics

ErrorIdent / Additional1	Diagnostics
	F20D1034 Error write access (DirectWriteWord)
	F20D1036 Axis has not been found (ML_AXISDATA)
	F20D1037 Invalid write/read access to ML_AXISDATA
F00D103B	Invalid control number
F00D103C	Invalid axis number
F00D1205	Error, cannot handle a single parameter
F00D1207	Error, data greater 64KBytes
F00D1210	Error, cannot handle a single parameter
F00D1211	Error ReadSERCOSDataStatus, no memory available
F00D1218	Error, no memory available
F00D1219	Error, cannot handle a single parameter
F00D1220	Unsupported selection of sercos element
F00D1221	Timeout on parameter access
F00D1222	Too many data to transmit
F00D1224	Error, no memory available
F00D1225	Error, cannot handle a single parameter
F00D1227	Error while writing data
F00D1230	Error ReadSERCOSAttribute, no memory available

Tab.9-5: Diagnostics MB_ReadxxxParameter / MB_WritexxxParameter / direct access

ErrorIdent / Additional1	MB_ReadxxxParameter / MB_WritexxxParameter / direct access
F90D1021	Error ReadParameter, no memory available
F90D1023	Error WriteParameter
F90D1024	Error WriteParameter, no memory available
F90D1032	No access to axes (1)
F90D1033	No access to axes (2)
F90D1038	Error getting MessageQueue
F90D1039	Thread could not be started
F90D103A	Left endless While-Loop
F90D1204	Error, no memory available
F90D1209	Error, no memory available

Tab.9-6: Fatal system errors of MB_ReadxxxParameter / MB_WritexxxParameter / direct access

Diagnostics



Depending on the pre-selected axis reaction, the error **F*0Dxxxx** appears as

- F00Dxxxx, if no error reaction by the axis
- F20Dxxxx, if "axis stop", best shutdown possible
- F90Dxxxx, no error reaction to the drive; torque disable

ErrorIdent / Additional1	Diagnostics	
F0290001		Transformation not valid
F0290400	F2299400	General kinematics error
F0290401	F2299401	Invalid control number
F0290402	F2299402	Invalid kinematics number
F0290403	F2299403	Invalid command type
F0290404	F2299404	Command not supported by kinematics
F0290405	F2299405	Command rejected while 'StandStill Incomplete'
F0290406	F2299406	Command rejected while 'PM Program Stopping'
F0290407	F2299407	Command rejected while 'StandStill Complete'
F0290408	F2299408	Invalid command number
F0290409	F2299409	Command not supported in this state
F0290410	F2299410	Reset still 'ErrorStop'
F0290411	F2299411	Command rejected while 'PM Program Running'
F0290412	F2299412	Error while selecting RCL program
F0290413	F2299413	Command rejected while 'Parameter Mode'
F0290414	F2299414	Command rejected while 'ErrorStop'
F0290415	F2299415	Command aborted
F0290416	F2299416	Switching to 'StandStill' not possible
F0290417	F2299417	RCL process does not exist
F0290501	F2299501	Two PLC-MotionCommands per motion cycle not allowed
F0290502	F2299502	Command rejected - buffer full
F0290503	F2299503	Invalid motion command
F0290505	F2299505	Synchronisation to more than one belt not possible

Tab.9-7: Kinematic state diagnostics of the MLC_Table I



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

Diagnostics

ErrorIdent / Additional1		Diagnostics
F0290501	F2299501	Two PLC-MotionCommands per motion cycle not allowed
F0290502	F2299502	Command rejected - buffer full
F0290503	F2299503	Invalid motion command
F0290505	F2299505	Synchronisation to more than one belt not possible
	F2299506	Slope type is not supported
	F2299507	Invalid set mode
	F2299508	Invalid coordinate system
	F2299509	Belt config not allowed temporary
	F2299510	Belt not configured
	F2299511	Belt synchronization error
	F2299512	Invalid belt synchronisation mode
	F2299513	Bad belt error reaction mode
	F2299514	Belt begin length invalid
	F2299515	Belt begin length greater total length
	F2299516	Wrong PCS index
	F2299517	Wrong CS selected
	F2299518	The TCP is not within belt working area
	F2299528	TeachIn: No name given
	F2299529	TeachIn: Name too long
	F229952A	TeachIn: Wrong name
	F229952B	TeachIn: Point already exist
	F229952C	TeachIn: Point doesn't exist
	F229952D	TeachIn: Wrong option
	F229952E	Tacheln: File corrupted
	F229952F	TeachIn: Wrong coordinate
	F2299530	TeachIn: Wrong coordinate system
	F2299531	TeachIn: File could not be opened
	F2299532	Values for relative desynchronisation invalid
	F2299533	Command buffer full
	F2299534	Transformation did not supported freedom degree

Tab.9-8: Kinematic state diagnostics of the MLC_ Table II



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

Diagnostics

ErrorIdent / Additional1	Diagnostics
F9299519	JCS not set
F929951A	BCS not set
F929951B	WCS not set
F929951C	PCS not set
F929951D	Trafo manager not set
F929951E	Preshape execute error
F929951F	Shape execute error
F9299520	Postshape execute error
F9299521	Move not allowed for this cmd
F9299522	No robot set
F9299523	Belt no time set
F9299524	Desync init failed
F9299525	lpo cycle time zero
F9299526	lpo distance below zero
F9299527	lpo too much elements

Tab.9-9: Kinematic state diagnostics of the MLC_Table III, Fatal System Errors



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

9.3 SERCOS_TABLE, ERROR_TABLE 16#0010

SERCOS_TABLE - ERROR_TABLE, Table number 16#0010

IndraLogic XLC /IndraMotion MLC extend the four-digit sercos error codes as eight-digit MLC_Error codes as follows:

Supplements are "F" for error, "0" as severity (severity of the error) and "2D" as a sign for the origin. Then, the four sercos digits follow.

Severity 0 signifies no reaction at the axes/drives.

The eight-digit code is used at the IndraWorks surface (error memory).

At the block outputs, the table number "ErrorIdent/Table", 16#0030 (MLC_TABLE) or 16#0010 (SERCOS_TABLE) determines which presentation is used.

Diagnostics

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
	General error	
0000	sercos: No error in the service channel	F02D0000
0001	sercos: Service channel not open	F02D0001
0003	Max. count of parallel parameter requests exceeded	F02D0003
0007	Parameter access to remote control not allowed	F02D0007
0009	sercos: Invalid access to closing the service channel	F02D0009
00A0	sercos: Parameter service channel is not open	F02D00A0
00B0	sercos: Parameter element does not exist	F02D00B0
00C0	sercos: Selected drive does not exist	F02D00C0
	Element 1 (ident number)	
1001	sercos: No IDN	F02D1001
1009	sercos: Invalid access to element 1	F02D1009
	Element 2 (name)	
2001	sercos: No name	F02D2001
2002	sercos: Name transmission too short	F02D2002
2003	sercos: Name transmission too long	F02D2003
2004	sercos: Name cannot be changed (read only)	F02D2004
2005	sercos: Name is write-protected at this time	F02D2005
	Element 3 (attribute)	
3002	sercos: Attribute transmission too short	F02D3002
3003	sercos: Attribute transmission too long	F02D3003
3004	sercos: Attribute cannot be changed (read only)	F02D3004
3005	sercos: Attribute is write-protected at this time	F02D3005
	Element 4 (unit)	
4001	sercos: No units	F02D4001
4002	sercos: Unit transmission too short	F02D4002
4003	sercos: Unit transmission too long	F02D4003
4004	sercos: Unit cannot be changed (read only)	F02D4004
4005	sercos: Unit is write-protected at this time	F02D4005
	Element 5 (minimum input value)	
5001	sercos: No minimum input value	F02D5001
5002	sercos: Min. input value transmission too short	F02D5002
5003	sercos: Min. input value transmission too long	F02D5003

Diagnostics

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
5004	sercos: Min. input value cannot be changed (read only)	F02D5004
5005	sercos: Min. input value is write-protected at this time	F02D5005
	Element 6 (maximum input value)	
6001	sercos: No maximum input value	F02D6001
6002	sercos: Max. input value transmission too short	F02D6002
6003	sercos: Max. input value transmission too long	F02D6003
6004	sercos: Max. input value cannot be changed (read only)	F02D6004
6005	sercos: Max. input value is write-protected at this time	F02D6005
	Element 7 (operation date)	
7001	sercos: No operation data	F02D7001
7002	sercos: Operation data transmission too short	F02D7002
7003	sercos: Operation data transmission too long	F02D7003
7004	sercos: Operation data cannot be changed (read only)	F02D7004
7005	sercos: Operation data is write-protected at this time	F02D7005
7006	sercos: Operation data is smaller than the min input value	F02D7006
7007	sercos: Operation data is greater than the max input value	F02D7007
7008	sercos: Invalid operation data	F02D7008
7009	sercos: Operation data write-protected by a password	F02D7009
700A	sercos: Operation data is write-protected (cycl. conf)	F02D700A
700B	sercos: Invalid indirect addr., (data container, list)	F02D700B
700C	sercos: Operation data is write-protected (other settings)	F02D700C
700D	sercos: Invalid floating point number	F02D700D
700E	sercos: Reserved	F02D700E
700F	sercos: Operation data is write-protected at OM	F02D700F
7010	sercos: Procedure command already active	F02D7010
7011	sercos: Procedure command not interruptible	F02D7011
7012	sercos: Procedure command at this time not executable	F02D7012
7013	sercos: Procedure command not executable	F02D7013
710C	sercos: Data exceed maximal range	F02D710C
8001	sercos: Access temporary not possible	F02D8001
8096	sercos: Device deactivated	F02D8096
8097	sercos: Device does not exist	F02D8097

Tab.9-10: sercos errors and their use in IndraLogic XLC / IndraMotion MLC

10 Appendix

10.1 Transformations, Comparing Robot Control V1 to Robot Control V2

The following table compares the transformations of Robot Control V1 to the transformations of Robot Control V2.

Transformation scheme	Name	V1	V2
0	Cartesian on page 74	X	X
7	2-Axis Delta with 1 Optional Orientation Axis on page 75	X	X
9	3-Axis Scara with 1 Optional Orientation Axis on page 77	X	X
10	4-Axis Scara with 1 Optional Orientation Axis on page 79	X	X
12	2-Axis Parallelogram on page 81	X	X
13	Inverted 2-Axis Parallelogram on page 82	X	X
18	Belt Kinematics on page 83	X	X
19	2-Axis Parallelogram V1 on page 84	X	X
20	3-Axis Delta with 1 Optional Orientation Axis on page 85	X	X
21	2-Axis Parallelogram V2 on page 87	X	X
26	Robot with offset wrist with 6 (7) axes	X	
27	Robot with offset wrist with 6 axes + 3 linear axes	X	
29	2-axis delta with rotary and translatory axis	X	
30	Adabot	X	
32	2-axis delta with asymmetric TCP	X	
38	3-Axis Twisting Crane on page 88		X
39	2-Axis Flex Stamper on page 89		X
40	3-Axis Delta with Linear Axes and 1 Optional Orientation Axis on page 91		X
45	2-Axis Delta Slide on page 94		X
46	3-Axis Parallelogram V3 on page 95		X
61	4-Axis Speedpicker on page 96		X
100	User-defined Transformation on page 98	X	X
111	Extended User-defined Transformation on page 100		X

V1 Robot Control V1
V2 Robot Control V2
Tab. 10-1: Overview of mechanic transformations

10.2 Overview: Kinematic parameters



1. Only an IndraMotion MLC supports kinematic functions.
2. "+" indicates that Robot Control V1 and/or V2 supports these parameters; "-" indicates that Robot Control V1 and/or V2 does not support them.

Appendix

Axis-specific parameters Limit values and axis-specific data are stored in the A-parameters of the respective axis.

Ident. No.	Parameter name	V1	V2
A-0-0028	Travel range limit switch	+	+
A-0-0030	Position limit value, positive	+	+
A-0-0031	Position limit value, negative	+	+
A-0-0032	Velocity limit value, positive	+	+
A-0-0033	Velocity limit value, negative	+	+
A-0-0034	Acceleration limit value, bipolar	+	+
A-0-0036	Jerk limit value, bipolar	+	+
A-0-0038	Torque/force limit value, bipolar	+	+
A-0-0080	Jogging, short distance	+	-
A-0-0081	Jogging, long distance	+	-
A-0-0082	Jogging, low velocity	+	-
A-0-0083	Jogging, high velocity	+	-
A-0-0084	Jogging, acceleration	+	-
A-0-0085	Jogging, deceleration	+	-
A-0-0086	Jogging, acceleration jerk	+	-
A-0-0087	Jogging, deceleration jerk	+	-
A-0-0224	Rapid stop deceleration	+	+
A-0-0225	Rapid stop jerk	+	+
A-0-0230	Blending distance PTP	+	-
A-0-0240	Feedrate override	+	+
A-0-0241	Feedrate override control word	+	+

V1 Robot Control V1
V2 Robot Control V2
Tab. 10-2: Overview of available A-parameters

Control-specific parameters The following C-parameters are created to manage **all** kinematic data of one control.

Ident. No.	Parameter name	V1	V2
C-0-0123	Maximum number of kinematics	+	+
C-0-2480	Configured kinematic number	+	+
C-0-2482	List of non-linked axes	+	+
C-0-2483	List of configured kinematics	+	+
C-0-2484	Kinematic configuration list	+	+

V1 Robot Control V1
V2 Robot Control V2
Tab. 10-3: Overview of available C-parameters

Parameters of one kinematics The following K-parameters are created to manage the data of **one** kinematics in the control. Each parameter exists once in each kinematics.

Appendix

Ident. No.	Parameter name	V1	V2
K-0-0000	Dummy, K-parameter	+	+
K-0-0001	Logic kinematic number	+	+
K-0-0002	Kinematic name	+	+
K-0-0003	Maximum path velocity	+	+
K-0-0004	Reserved	+	-
K-0-0005	Software limit switch, positive, MCS	+	+
K-0-0006	Software limit switch, negative, MCS	+	+
K-0-0007	Axis number of kinematics	+	+
K-0-0008	Axis configuration list	+	+
K-0-0009	Osci: list of all K-parameters that can be oscilloscoped	+	+
K-0-0010	List of all K-parameters	+	+
K-0-0011	List of all invalid K-parameters	+	+
K-0-0012	Checksum of all K-parameters	+	+
K-0-0013	IDN list of checksum K-parameters	+	+
K-0-0014	Path acceleration, change time	+	-
K-0-0015	Path deceleration, change time	+	-
K-0-0016	Project identification number	+	+
K-0-0019	Kinematic type	+	+
K-0-0020	Kinematic diagnostic message	+	+
K-0-0021	Kinematic status	+	+
K-0-0022	Extended kinematic status	+	+
K-0-0023	Kinematic diagnostic number	+	+
K-0-0031	Transformation scheme	+	+
K-0-0032	Referencing of axes, command configuration	+	-
K-0-0033	IPOS, update at coordinate change	+	-
K-0-0034	Maximum path acceleration (bipolar)	-	+
K-0-0035	Maximum path jerk (bipolar)	-	+
K-0-0036	MCS axis monitoring	+	-
K-0-0040	Transformation, lengths of the mechanics	+	+
K-0-0041	Transformation, factors	+	+
K-0-0042	Offset of MCS system	+	+
K-0-0043	Assignment list of workpiece coordinates	+	-
K-0-0045	Tool selection	+	-
K-0-0050	Distance units MCS	+	+
K-0-0051	Velocity units MCS	+	+
K-0-0052	Acceleration units MCS	+	+

Appendix

Ident. No.	Parameter name	V1	V2
K-0-0053	Jerk units MCS	+	+
K-0-0054	Measuring unit for kinematic data	+	+
K-0-0070	Belt number of kinematics	+	+
K-0-0071	Coupling factors, belt 1	+	-
K-0-0072	Coupling factors, belt 2	+	-
K-0-0073	Coupling factors, belt 3	+	-
K-0-0074	Coupling factors, belt 4	+	-
K-0-0080	Jogging, distance short in MCS	+	-
K-0-0081	Jogging, distance long in MCS	+	-
K-0-0082	Jogging, velocity low in MCS	+	-
K-0-0083	Jogging, velocity high in MCS	+	-
K-0-0084	Jogging, acceleration in MCS	+	-
K-0-0085	Jogging, deceleration in MCS	+	-
K-0-0086	Jogging, acceleration jerk in MCS	+	-
K-0-0087	Jogging, deceleration jerk in MCS	+	-
K-0-0096	Peak value of command buffer size	-	+
K-0-0097	Maximum command buffer size	-	+
K-0-0098	Current command buffer size	-	+
K-0-0099	Kinematic configuration	+	+
K-0-0100	Current position, Cartesian Machine Coordinate System	+	+
K-0-0102	Current path velocity	+	+
K-0-0103	Current path acceleration	+	+
K-0-0104	Current path jerk	+	+
K-0-0105	Actual position, Cartesian Workpiece Coordinates	+	-
K-0-0106	Coordinate values of tool	+	-
K-0-0107	Coordinate values of workpiece	+	-
K-0-0121 ...	Coordinate name C1 ...	+	-
K-0-0136	Coordinate name C16	+	-
K-0-0140	Tool name	+	-
K-0-0141	Workpiece name	+	-
K-0-0150	Interpolated command position	+	+
K-0-0152	Interpolated path velocity	+	+
K-0-0153	Interpolated path acceleration	+	+
K-0-0154	Interpolated path jerk	+	+
K-0-0155	Safe zones on/off	+	-
K-0-0156	Safe zone 1 min. point	+	-

Appendix

Ident. No.	Parameter name	V1	V2
K-0-0157 ...	Safe zone 1 max. point ...	+	-
K-0-0218	Safe zone 32 min. point	+	-
K-0-0219	Safe zone 32 max. point	+	-
K-0-0220	File transfer name	+	-
K-0-0221	File name to be compiled	+	-
K-0-0223	Selection of error reaction	-	+
K-0-0224	Emergency halt deceleration	-	+
K-0-0225	Emergency halt jerk	-	+
K-0-0240	Feedrate override	+	+
K-0-0241	Feedrate override control word	+	+
K-0-0242	Acceleration override	+	+
K-0-0243	Deceleration override	+	+
K-0-0250	Continuous slope lead time	+	+
K-0-0815	Odometer	-	+
K-0-1000	Command: Load basic kinematic parameters	+	+
K-0-1060	Command: Load file from CF	+	-
K-0-1061	Command: Save file to CF	+	-
K-0-1062	Command: Compile file to MLC	+	-
K-0-2100	List of all K-parameters to be backed up	+	+
K-0-2200	Programmed position 1 (active)	+	-
K-0-2201	Programmed position 2 (active)	+	-
K-0-2202	Type of actively programmed position	+	-
K-0-2203	Interpolation type	+	-
K-0-2204	Currently active QLL file name	+	-
K-0-2205	Currently active line number	+	-
K-0-2300	PLC bit input (RCL)	+	-
K-0-2301	PLC integer input (RCL)	+	-
K-0-2302	PLC floating point input (RCL)	+	-

V1 Robot Control V1
V2 Robot Control V2
Tab. 10-4: Overview of available K-parameters

10.3 Switching from Robot Control V1 to Robot Control V2

10.3.1 Converting RCL Commands

A RCL programming language contains all key words required for kinematics commands.

The following sections explain what PLCopen function blocks replace these key words and how the keyword functionality can be reproduced using PLCopen function blocks.

Appendix

Variable Declaration The required variables are declared in the declaration part of the function block or program. The following illustrates example variable declarations using RCL and PLCopen.

RCL	PLCopen
INTEGER: iValue	iValue: DINT; iValue: INT;
REAL: rValue	rValue: REAL;
BINARY: bValue	bValue: BOOL;
CHAR: strValue	strValue: STRING(255);
POINT: arPoint	arPoint: ARRAY [1..16] OF REAL; (uses the machine coordinate system)
JC_POINT: arPoint	arPoint: ARRAY [1..16] OF REAL; (uses the axis coordinate system)
BELT	Not required
TEXT: strValue	strValue: STRING(80);
ARRAY [1..7] INTEGER: arValue	arValue: ARRAY [1..7] OF DINT; arValue: ARRAY[1..7] OF INT;
WC_FRAME	Not required
FILE: strValue	Uses the function blocks of the SysFile library
INPUT BINARY: 1 = ibValue	VAR_INPUT ibValue: BOOL; END_VAR
OUTPUT BINARY: 1 := obValue	VAR_OUTPUT obValue: BOOL; END_VAR
PUBLIC ... EXTERNAL ...	Uses the global variable declaration
CONST: iValue=1, rValue=2.75	VAR CONSTANT iValue: UINT := 1; rValue: REAL := 2.75; END_VAR

The prefix "DEF" which may be used for POINT and JC_POINT variables causes the point to be saved in a point file. This is not required if PLCopen function blocks are used, as the commanded points cannot be saved to a point file automatically.

Functions RCL and PLCopen functions have largely the same syntax or may be mapped using simple function blocks. The following table matches the RCL functions to equivalent PLCopen functions.

RCL	PLCopen
SIN(0.5)	SIN(0.5) -> 0.479426
COS(0.5)	COS(0.5) -> 0.877583
ATAN(0.5)	ATAN(0.5) -> 0.463648

Appendix

SQRT(49)	SQRT(49) -> 7
ABS(-5)	ABS(-5) -> 5
TRUNC(1.9)	TRUNC(1.9) -> 1
ORD	No function available
CHR(34)	No function available
ROUND(7.81)	REAL_TO_INT(7.81) -> 8
13 MOD 5	13 MOD 5 -> 3
END_OF_FILE	Uses function block SysFileEOF
ASSIGN	No function available
INT_ASC	No function available
ASC_INT	No function available
SIZEOF(arValue)	SIZEOF(arValue) -> result depends on the type of array elements arValue: ARRAY[1..5] OF INT; -> result = 10 arValue: ARRAY[1..5] OF DINT; -> result = 20
LIMIT_MIN, ..._MAX, ..._OFF	Corresponds to software limit switches Robot Control V1: No function available Robot Control V2: ML_GroupSetSafeZone
JC(...), WC(...)	Corresponds to a transformation of coordinates Robot Control V1: No function available Robot Control V2: ML_RobotTransformPoint

Special functions

The RCL special functions implement functions without reserved elements in the RCL language. They are an extension of the RCL language and have to be declared before they are called, similar to variables. The following table lists the PLCopen function blocks that cover the special functions of the extended RCL language.

RCL	PLCopen
SPC_FCT:1=IOL	ML_SwitchOnPath
SPC_FCT:2=PRO	No equivalent function block available (can be mapped with ML_SwitchOnPath)
SPC_FCT:21=belt_kind	Robot Control V1: Corresponds to the "BeltKind" input of the ML_SetBeltParameter function block Robot Control V2: No equivalent function block available
SPC_FCT:23=clock_date	ML_GetSystemTimeUTC or IL_SysTimeDate
SPC_FCT:24=sys_count	No equivalent function block available
SPC_FCT:28=belt_set	No equivalent function block available

Appendix

RCL	PLCopen
SPC_FCT:53=belt_area	Corresponds to inputs "BeginLength" and "TotalLength" or "FullLength" of the function block ML_SetBeltParameter (Robot Control V1) or of the function block ML_SetBeltConfiguration (Robot Control V2)
SPC_FCT:59=getBlendMode	Robot Control V1: ML_KinGetDistanceBlendOption Robot Control V2: No equivalent function block available
SPC_FCT:60=setBlendMode	Robot Control V1: ML_KinSetDistanceBlendOption Robot Control V2: No equivalent function block available

Motion boundary conditions

Each motion command requires certain boundary conditions (velocity, acceleration, coordinate system, etc.).

In the RCL program, these values are defined by the respective additional information. These values are indicated by the letters "V" (path velocity), "A" (path acceleration), etc.

Below, an example PTP and linear motion illustrates the relations between the boundary conditions in the RCL program to the inputs of the PLCopen function blocks and their effects:

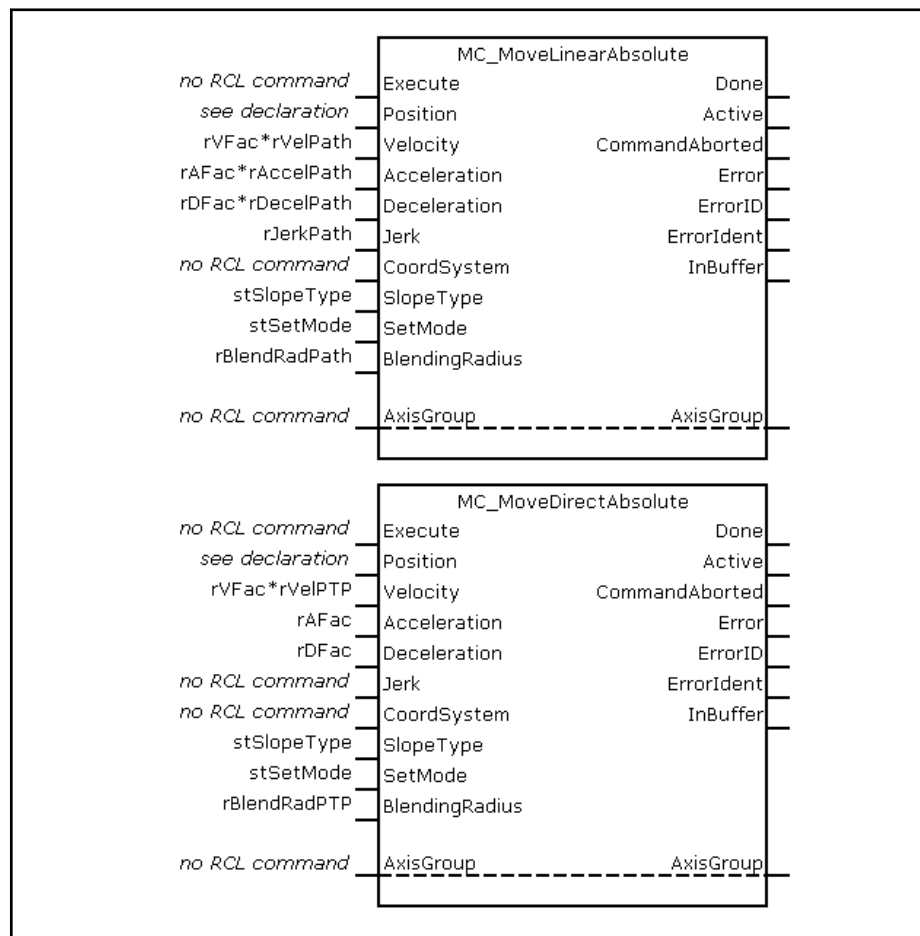


Fig. 10-1: Boundary conditions in the RCL program and inputs of the PLCopen function blocks

Appendix

RCL	PLCopen
V	rVelPath
A	rAccelPath, rDecelPath
J	rJerkPath
R	rBlendRadPath
VFACTOR	rVFac
AFACTOR	rAFac
DFACTOR	rDFac
V_PTP	rVelPTP
R_PTP	rBlendRadPTP
BLOCK_..., PROGR_..., CONT_SLOPE	stSlopeType
TO / EXACT, VIA / APPROX	stSetMode
WITH	No explicit input (see below)
UNTIL	No explicit input (see below)

The coordinate system the kinematics is to travel in, is specified implicitly in the RCL program by the target point.

If the target point is declared as "POINT", this equals CS_MACHINE_COORDINATES at the input of "CoordSystem" of the PLCopen function blocks.

However, if the target point is declared as "JC_POINT", this equals CS_AXES_COORDINATES at the input of "CoordSystem" of the PLCopen function blocks.

The target point is specified in the RCL program together with the motion command which is represented by the PLCopen function block input "Position" (absolute motion command) or "Distance"(relative motion command).

With the RCL command **WITH** a motion command may be assigned other boundary conditions that only apply to this command. With the PLCopen function blocks, this is implied by different values at each function block.

If an RCL command has the abort condition **UNTIL**, the motion is only carried out while the condition is met. With PLCopen function blocks, this is only possible if the function blocks MC_GroupInterrupt and MC_GroupContinue are used in addition to the motion command.

For details, refer to [Converting RCL Programs on page 372](#)

Other key words

In addition to the keywords mentioned, the RCL language contains other commands that are required or useful for a motion. The following table lists the PLCopen function blocks that cover these additional keywords of the extended RCL language.

RCL	PLCopen
MOVE ...	MC_MoveDirectAbsolute MC_MoveLinearAbsolute MC_MoveCircularAbsolute
MOVE_REL ...	MC_MoveDirectRelative (only Robot Control V2) MC_MoveLinearRelative MC_MoveCircularRelative (only Robot Control V2)

Appendix

RCL	PLCopen
PAUSE	No equivalent function block available
HALT	No equivalent function block available
POS	Robot Control V1: uses the KinData interface Robot Control V2: MC_GroupReadActualPosition
SYNC	No equivalent function block available
SYNCHRON	Corresponds to a belt value $\neq 0$ in the target point
SYNCHRON_END	Corresponds to a belt value $= 0$ in the target point
WAIT ...	Robot Control V1: TON / TOF Robot Control V2: ML_GroupWait
WAIT UNTIL ...	No equivalent function block available

10.3.2 Converting RCL Programs

In RCL programs a motion sequence is programmed sequentially, i.e. the program is prepared and executed line by line. As the PLCopen function blocks are called cyclically, they require special handling.

Using the interactions of the activation input "Execute" and the status outputs "Done" and "InBuffer", it is possible to imitate sequential programming.

The following typical use cases illustrate how different RCL sequences can be implemented by PLCopen function blocks.

Blending motion with various velocities

RCL program example

```

BEGIN
BLOCK_SLOPE
  V = rVelocity
  A = rAcceleration
  J = rJerk
  R = rRadius

  MOVE LINEAR VIA arPosition
  MOVE LINEAR WITH V = rNewVelocity TO arNextPosition
PROGRAM_END

```

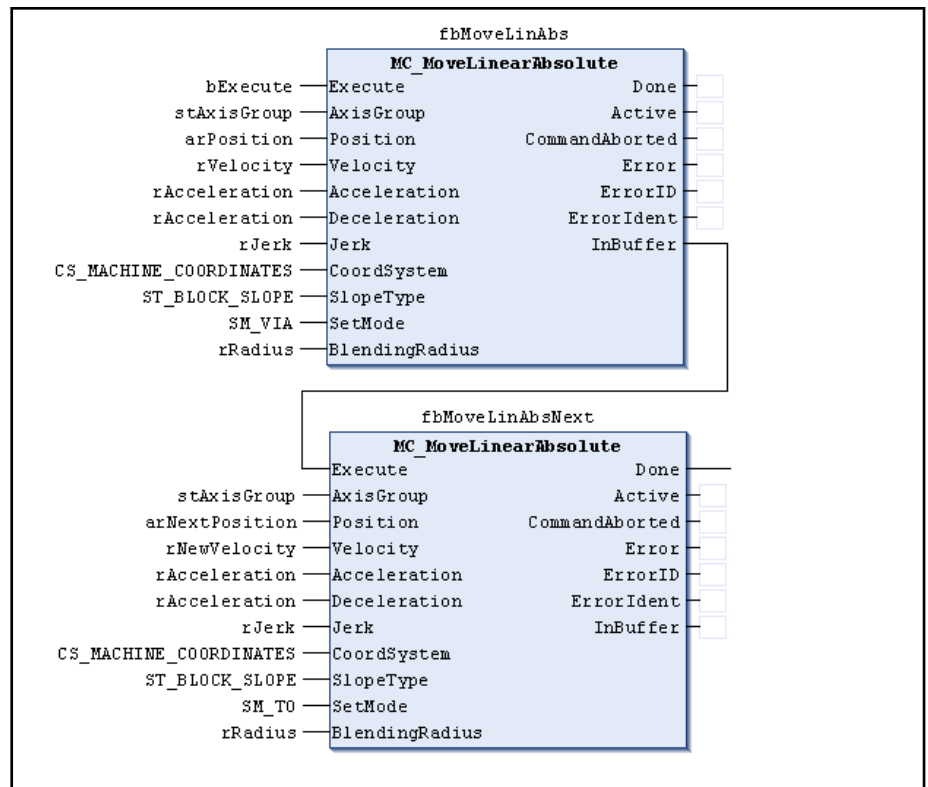


Fig. 10-2: PLCopen program example

After starting the RCL program, the first programmed motion sequence is started automatically and the following motion is written into the block buffer as preparation. If PLCopen function blocks are used, the first motion block is started by setting its "Execute" input. All following motion blocks can be started with the outputs of the previous function block.

Whether a motion is blended depends on which PLCopen function block output is used to activate the following function block. As shown by the example, the output "InBuffer" of the instance "fbMoveLinAbs" is connected to the input of the following function block. Therefore, the second motion command is issued as soon as the first motion command is written to the block buffer. However, if a non-blending motion is required, i.e. if the target point is to be traveled to directly (SetMode = TO) it is not mandatory to use the output "InBuffer". It is also possible to use the output "Done" as a condition for the next step.

The coordinate system used for traveling is defined indirectly by the point in the RCL program. In the example, it is of type POINT, therefore the machine coordinate system is used. Using PLCopen function blocks, the coordinate system has to be specified at input "CoordSystem" and corresponds to "CS_MACHINE_COORDINATES".

It is also possible to implement the motion shown in the example with only one function block instance. For this purpose, the output "InBuffer" or "Done" has to be linked to the input "Execute" of the same instance using the respective logic, so it can be restarted by a positive edge at the output. However, if the "InBuffer" output is used, it is not possible to diagnose the first motion command as the outputs of the function block are reset when it is restarted and therefore apply to the motion started last!

Appendix

Motion with an abort condition (only Robot Control V2)

RCL program example

```

BEGIN
BLOCK_SLOPE
V = rVelocity
A = rAcceleration
J = rJerk
R = rRadius

MOVE_LINEAR TO arPosition UNTIL bAbort = FALSE
PROGRAM_END

```

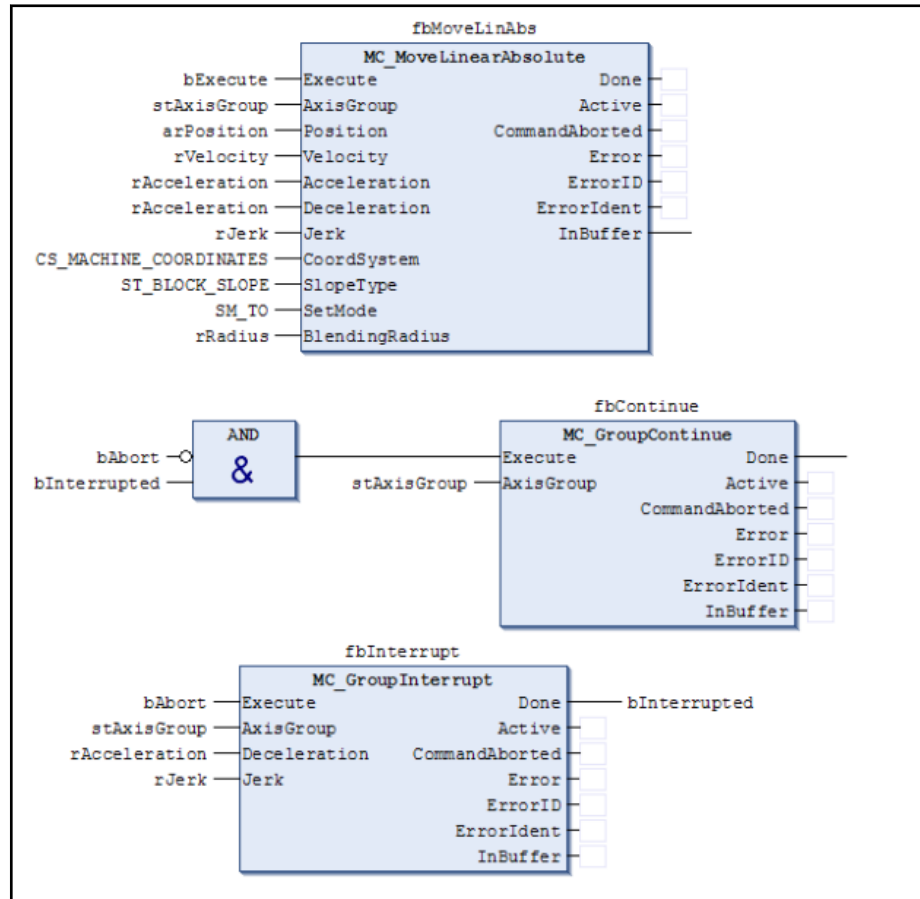


Fig. 10-3: PLCoen program example only for Robot Control V2

After starting the RCL program, the system checks if the condition of the motion command (`bAbort = FALSE`) is met. If yes, the motion sequence begins. During the motion the condition is also monitored and if it is no longer met, the motion is stopped.

If the PLCopen function blocks are used, the function blocks `MC_GroupInterrupt` and `MC_GroupContinue` have to be used in addition to the motion block. As can be seen from the example, the motion sequence is stopped or not started while the variable (condition) "`bAbort`" is set. If this variable is reset and the motion sequence is started by a rising edge at the "Execute" input of the motion function block, the motion is started or continued.

The coordinate system used for traveling is defined indirectly by the point in the RCL program. In the example, it is of type `POINT`, therefore the machine coordinate system is used. Using PLCopen function blocks, the coordinate system has to be specified at input "CoordSystem" and corresponds to "`CS_MACHINE_COORDINATES`".

Belt synchronization and wait command

RCL program example

```
BEGIN
BLOCK_SLOPE
V = rVelocity
A = rAcceleration
J = rJerk
R = rRadius

SPC_FCT_21_belt_kind (rBeltNumber, BeltKind)
SPC_FCT_53_belt_area (rBeltNumber, arBase, rFullLength, rBeginLength)

SYNCHRON
MOVE_LINEAR TO arSyncPosition
WAIT 5
SYNCHRON_END
MOVE_LINEAR TO arAsyncPosition
PROGRAM_END
```

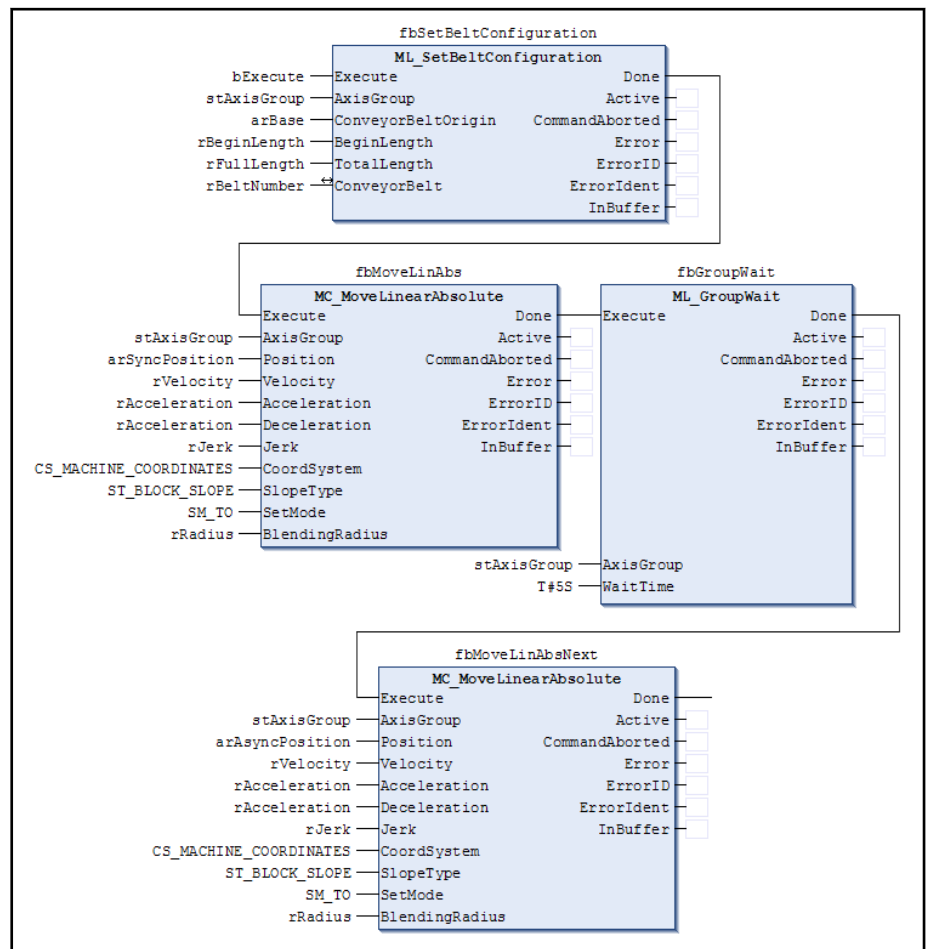


Fig.10-4: PLCOpen program example only for Robot Control V2

Following the start of the RCL program the belt is configured by the special functions SPC_FCT 21 and 53. Then, the synchronization process starts automatically. If the PLCOpen function blocks are used, belt configuration is carried out by the function block ML_SetBeltConfiguration (Robot Control V2) or ML_SetBeltParameter (Robot Control V1). When the configuration is completed, the synchronization motion starts. When the motion is completed, i.e. when the kinematics reaches the commanded point and is fully synchronous with the specified belt, the function block ML_GroupWait (Robot Control V2) or TON (Robot Control V1) is started and the specified time begins to elapse. Then, the desynchronization process is started, which desynchronizes the belt and begins traveling to the new target point.

Appendix

For more information on synchronization and desynchronization, see [Belt Operation on page 52](#).

In this example, all conditions for the next step are defined by the "Done" output of the function blocks. The output "InBuffer" can also be used instead. This causes the commands to be written into the block buffer and to be processed one after the other.

For other use cases on "Belt-synchronization", see

- [Application example 1: Using Coordinate Systems on page 315](#)
- [Application example 2: Belt Synchronization on page 316](#)

10.3.3 Converting IEC Commands

Existing IEC programs for Robot Control V1 can be converted to IEC programs for Robot Control V2 in a few steps. Obviously, this conversion process greatly depends on the respective program.

In the following sections, the conversion steps are explained briefly.



We recommend backing up parameters prior to the conversion from Robot Control V1 to Robot Control V2!

Step 1: Changing the Robot Control version

Change the Robot Control version from "Robot Control V1" to "Robot Control V2" in the kinematics configuration.



All existing RCL programs are deleted!

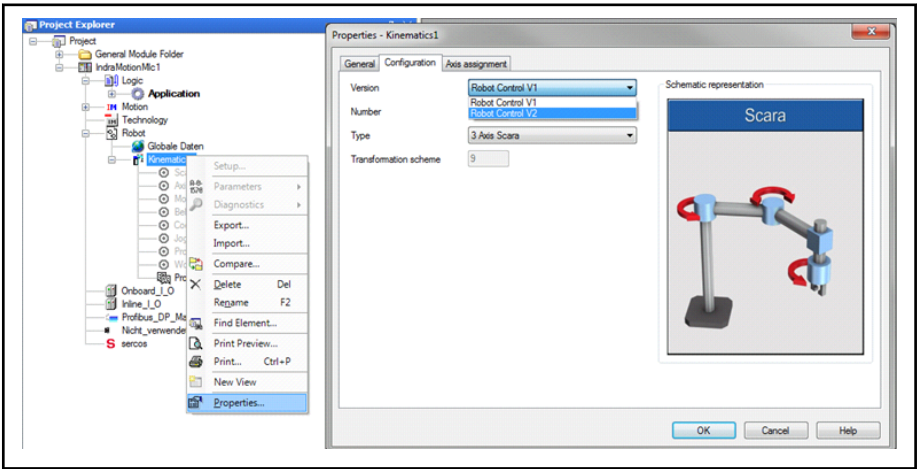


Fig. 10-5: Robot Control V2 kinematic configuration

Step 2: Replacing ML_SetBelt-Parameter by ML_SetBeltConfiguration



This step is only required with belt synchronization!

Replace the function block ML_SetBeltParameter by the equivalent function block ML_SetBeltConfiguration.

The variables for the inputs "Execute", "Full/TotalLength", "BeginLength", "AxisGroup" and "ConveyorBelt" can be maintained.

The configuration of input "ConveyorBeltOrigin" is described in the next step.

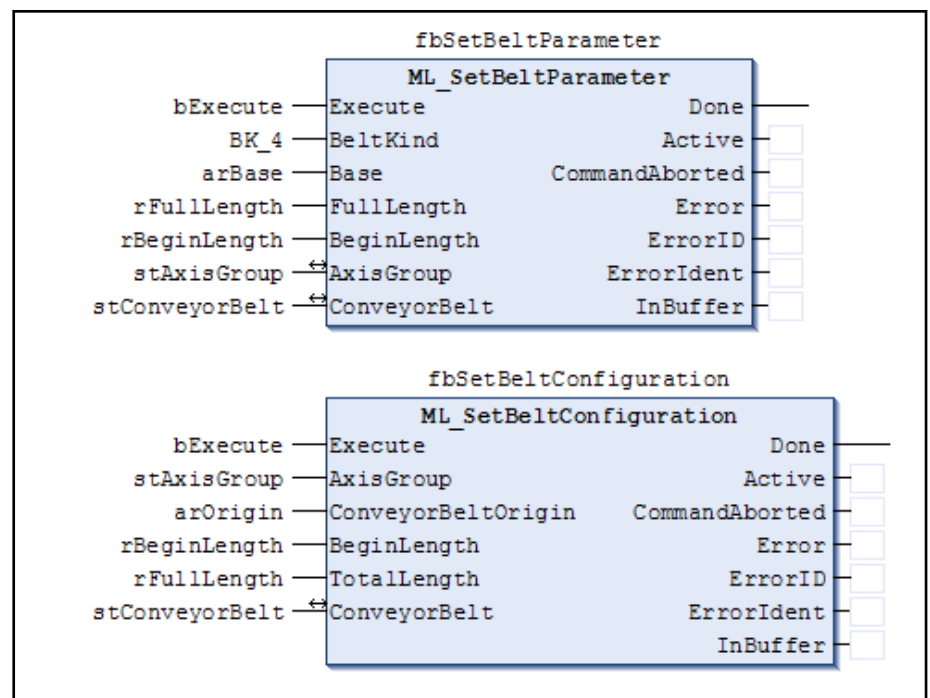


Fig. 10-6: Function blocks for belt configuration

Step 3: Definition of the position and rotation of the belt (ConveyorBeltOrigin)



This step is only required with belt synchronization!

With Robot Control V1 the belt position is defined by the input "Base" of the function block ML_SetBeltParameter and its rotation by the belt coupling factors.

With Robot Control V2 these two configurations are combined in the input "ConveyorBeltOrigin" of the function block ML_SetBeltConfiguration.

The first three values determine the belt displacement and the last three values determine the belt rotation in relation to the machine coordinate system (MCS). Therefore, a new variable with the type ARRAY [1..6] OF REAL has to be created for this input.

Step 4: Changes in the application

Remove all function blocks that are only valid for Robot Control V1.

If this is not done and a function block is called that is not valid for Robot Control V2, this function block returns the respective error message and the program may be interrupted.

Step 5: Restoring parameters

Due to the change from Robot Control V1 to V2, the internal kinematics is deleted and a new kinematics is created. Therefore, the kinematics parameters backed up earlier have to be restored.

When these steps are completed, the change from Robot Control V1 to Robot Control V2 was successful and the project can be reloaded to the control.

11 Service and Support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany

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

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

Phone: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

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

Service worldwide

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

Preparing information

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

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

Index

A

A-0-0028, Travel range limit switch.....	329
A-0-0030, Position limit value, positive.....	329
A-0-0031, Position limit value, negative.....	330
A-0-0032, Velocity limit value, positive.....	330
A-0-0033, Velocity limit value, negative.....	330
A-0-0034, Acceleration limit value, bipolar.....	330
A-0-0036, Jerk limit value, bipolar.....	331
A-0-0038, Torque/force limit value, bipolar.....	331
A-0-0224, Rapid stop deceleration.....	331
A-0-0225, Rapid stop jerk.....	332
A-0-0240, Feedrate override.....	332
A-0-0241, Feedrate override control word.....	332
Abbreviations.....	18
About this documentation.....	11
Validity of the documentation.....	11
Absolute positioning.....	34
Access kinematic data.....	189
ACS.....	21
Active.....	116
Adding a kinematics to the system overview....	303
Application phases.....	11
Axis Coordinate System.....	21
Axis coordinate system (ACS).....	21

B

Base Coordinate System.....	21
Base coordinate system (BCS).....	21
BCS.....	21
Belt configuration.....	50
Belt operation.....	52
Belt values.....	51
Belt-specific error reaction.....	60
Belt-synchronous motion.....	49
Blending and belt synchronization.....	53
Blending radius.....	40
Block_Slope.....	41

C

C-0-0123, Maximum number of kinematics.....	332
C-0-2480, Configured kinematic number.....	332
C-0-2482, List of non-linked axes.....	333
C-0-2483, List of configured kinematics.....	333
C-0-2484, Kinematic configuration list.....	333
Circular.....	33
CommandAborted.....	116
Cont_Slope.....	43
Continuous operation.....	36
Coordinate systems.....	19
Coordinates.....	19
Cyclical command value specification	
Robot Control V2.....	61

D

Degenerate circles.....	33

Diagnostics of MB_ReadxxxParameter /	
MB_WritexxxParameter / direct access.....	357
Diagnostics on the kinematic state.....	358
Documentation structure.....	11
Done.....	116
DV_Kin[].....	110

E

E_KIN_MECH_CMD.....	237
E_KIN_TEACH_IN_POINT_FILE_OPTIONS....	242
Enable.....	116
Error codes	
MC_KIN_REF.....	233
MC_KIN_REF_EXT.....	236
MC_MoveCircularAbsolute.....	168
MC_MoveCircularRelative.....	171
MC_MoveDirectAbsolute.....	173
MC_MoveDirectRelative.....	175
MC_MoveLinearAbsolute.....	177
MC_MoveLinearRelative.....	179
MC_SetCartesianTransform.....	131
MC_SetKinTransform.....	132, 133
ML_GroupAllAxes.....	137
ML_GroupJogCont.....	183
ML_GroupJogStep.....	185
ML_RobotGetBeltConfiguration.....	151
ML_RobotGetBeltErrorReaction.....	153
ML_RobotGetMechanic.....	155
ML_RobotGetPerformance.....	156
ML_RobotGetTrafoValues.....	157
ML_RobotMovePoint.....	158
ML_RobotReadCyclicPos.....	159
ML_RobotSetPerformanceEnable.....	160
ML_RobotTransformPoint.....	162
ML_SetBeltConfiguration.....	142
ML_SetBeltErrorReaction.....	143
ML_SetBeltParameter.....	145
ML_SetCylindricTransform.....	147
ML_SetRotaryTableConfiguration.....	148
ML_SwitchOnPath.....	163
ML_UngroupAllAxes.....	135
Error reaction	
Belt-specific.....	60
Commanded motion.....	60
Error tables	
MLC_TABLE.....	351
Error_Tables	
MLC_TABLE.....	351
SERCOS_TABLE.....	360
Execute.....	116
ExecuteLock.....	116

F

First steps, kinematics interface.....	302
Function Variable Access	

Index

DV_Kin[].....	110
function variable access status.....	111

G

Global variables.....	301
Global_MB_KinInterfaceVars.....	301
GVL_K_Param_ID.....	189
GVL_RobotConstant.....	109

I

Information representation	
Names and abbreviations.....	18
InGear.....	116
InOperation.....	116
InSync.....	116
IntelliSense	
Kinematic data.....	109
Kinematic Data v.....	110
Intermediate point.....	33
InTorque.....	116
InVelocity.....	116

K

K-0-0001, Logic kinematic number.....	334
K-0-0002, Kinematic name.....	334
K-0-0003, Maximum path velocity.....	334
K-0-0005, Software limit switch, positive, MCS	335
K-0-0006, Software limit switch, negative, MCS.....	335
K-0-0007, Axis number of kinematics.....	335
K-0-0008, Axis configuration list.....	335
K-0-0009, Osci: list of all K-parameters that can be oscilloscoped.....	337
K-0-0010, List of all K-parameters.....	337
K-0-0011, List of all invalid K-parameters.....	337
K-0-0012, Checksum of all K-parameters.....	338
K-0-0013, IDN list of checksum K-parameters..	338
K-0-0016, Project identification number.....	338
K-0-0019, Kinematic type.....	338
K-0-0020, Kinematic diagnostic message.....	338
K-0-0021, Kinematic status.....	339
K-0-0022, Extended kinematic status.....	340
K-0-0023, Kinematic diagnostic number.....	340
K-0-0031, Transformation scheme.....	340
K-0-0034, Maximum path acceleration (bipolar).....	340
K-0-0035, Maximum path jerk (bipolar).....	341
K-0-0040, Transformation, lengths of the mechanics.....	341
K-0-0041, Transformation, factors.....	341
K-0-0042, Offset of MCS system.....	341
K-0-0050, Distance units MCS.....	342
K-0-0051, Velocity units MCS.....	342
K-0-0052, Acceleration units MCS.....	343
K-0-0053, Jerk units MCS.....	343
K-0-0054, Measuring unit for kinematic data....	344
K-0-0070, Belt number of kinematics.....	344

K-0-0096, Peak value of command buffer size.	345
K-0-0097, Maximum command buffer size.....	345
K-0-0098, Current command buffer size.....	345
K-0-0099, Kinematic configuration.....	345
K-0-0100, Current position, Cartesian Machine Coordinate System.....	345
K-0-0102, Current path velocity.....	346
K-0-0103, Current path acceleration.....	346
K-0-0104, Current path jerk.....	346
K-0-0150, Interpolated command position.....	346
K-0-0152, Interpolated path velocity.....	346
K-0-0153, Interpolated path acceleration.....	346
K-0-0154, Interpolated path jerk.....	346
K-0-0223, Selection of error reaction.....	347
K-0-0224, Emergency halt deceleration.....	347
K-0-0225, Emergency halt jerk.....	347
K-0-0240, Feedrate override.....	347
K-0-0241, Feedrate override control word.....	347
K-0-0242, Acceleration override.....	348
K-0-0243, Deceleration override.....	348
K-0-0250, Continuous slope lead time.....	348
K-0-0815, Odometer.....	348
K-0-1000, Command: Load basic kinematic parameters.....	349
K-0-2100, List of all K-parameters to be backed up.....	349
KinData[] - Kinematic data in shared memory. .	109
Kinematic Data via Function Variable Access. .	110
Kinematics interface function blocks.....	280
Kinematics interface operation modes	
Overview.....	290
Kinematics interface structures.....	255
MB_KINEMATICS_ADMIN_STATUS.....	273
MB_KINEMATICS_ADMINISTRATION.....	257
MB_KINEMATICS_CONTROL_TYPE01.....	256
MB_KINEMATICS_CONTROL_TYPE01 structure.....	256
MB_KINEMATICS_CONVEYOR_BELT.....	261
MB_KINEMATICS_CONVEYOR_BELT_V1	263
MB_KINEMATICS_COORDINATED.....	270
MB_KINEMATICS_DIAGNOSIS.....	278
MB_KINEMATICS_MODE_UNION.....	260
MB_KINEMATICS_RCL.....	272
MB_KINEMATICS_RCL_POINT.....	264
MB_KINEMATICS_RCL_PROG.....	266
MB_KINEMATICS_RCL_STATUS.....	276
MB_KINEMATICS_SETUP.....	268
MB_KINEMATICS_SETUP_STATUS.....	280
MB_KINEMATICS_STATUS_TYPE01.....	273
MB_KINEMATICS_STATUS_TYPE01 structure.....	272
Kinematics interface visualizations.....	302
KinSystem_Overview_All.....	303

L

Library	
ML_KinTech.library.....	254

ML_Robot.library.....	107
Linear.....	32

M

Machine Coordinate System.....	22
Machine coordinate system (MCS).....	22
Manual mode.....	34, 39
MB_GroupReadListParameter.....	190
MB_GroupReadParameter.....	191
MB_GroupReadRealParameter.....	192
MB_GroupReadSercosAttribute.....	193
MB_GroupReadSercosDataStatus.....	194
MB_GroupReadStringParameter.....	195
MB_GroupWriteListParameter.....	196
MB_GroupWriteParameter.....	198
MB_GroupWriteRealParameter.....	199
MB_GroupWriteStringParameter.....	199
MB_IRDProgSelect.....	201
MB_IRDProgStart.....	202
MB_IRDProgStop.....	202
MB_KINEMATICS_ADMIN_STATUS.....	273
MB_KINEMATICS_ADMINISTRATION.....	257
MB_KINEMATICS_CONVEYOR_BELT.....	261
MB_KINEMATICS_CONVEYOR_BELT_V1.....	263
MB_KINEMATICS_COORDINATED.....	270
MB_KINEMATICS_DIAGNOSIS.....	278
MB_KINEMATICS_MODE_UNION.....	260
MB_KINEMATICS_RCL.....	272
MB_KINEMATICS_RCL_POINT.....	264
MB_KINEMATICS_RCL_PROG.....	266
MB_KINEMATICS_RCL_STATUS.....	276
MB_KINEMATICS_SETUP.....	268
MB_KINEMATICS_SETUP_STATUS.....	280
MB_KINEMATICS_STATUS_TYPE01.....	273
MC_GroupContinue.....	163
MC_GroupInterrupt.....	164
MC_GroupReadActualPosition.....	148
MC_GroupReset.....	128
MC_GroupStop.....	165
MC_KIN_REF.....	232
MC_KIN_REF_EXT.....	234
MC_MoveCircularAbsolute.....	166
MC_MoveCircularRelative.....	169
MC_MoveDirectAbsolute.....	171
MC_MoveDirectRelative.....	173
MC_MoveLinearAbsolute.....	175
MC_MoveLinearRelative.....	177
MC_SetCartesianTransform.....	129
MC_SetKinTransform.....	131
MC_SetKinTransformExt.....	132
MC_UngroupAllAxes.....	133
MCS.....	22
Mechanical information.....	103
ML_AddMechanicData.....	135
ML_ARR_COORDS.....	243
ML_ARR_COORDS, application.....	248
ML_ARR_PLACEMENT.....	244

ML_BELT_ERROR_REACTION_MODE.....	230
ML_BELT_KIND.....	230
ML_BeltDeSynchronisationWithLimits.....	179
ML_BeltSynchronisationWithLimits.....	180
ML_CIRCUMODES.....	244
ML_COORD_BITS.....	241
ML_COORD_BITS, application.....	242
ML_COORDSYS.....	244
ML_CYCLIC_MODE.....	245
ML_GroupAllAxes.....	135
ML_GroupDeletePointFile.....	221
ML_GroupDeletePoints.....	222
ML_GroupJogCont.....	181
ML_GroupJogStep.....	183
ML_GroupLoadPointFile.....	223
ML_GroupReadNextPoint.....	224
ML_GroupReadPoint.....	225
ML_GroupSavePointFile.....	226
ML_GroupSetSafeZone.....	137
ML_GroupWait.....	186
ML_GroupWritePoint.....	228
ML_KIN_AXES_CONFIG_DATA.....	230
ML_KIN_AXIS_CONFIG.....	231
ML_KIN_CS_TYPE.....	239
ML_KIN_E_COORD_SYS.....	241
ML_KIN_E_COORD_SYS, application.....	242
ML_KIN_E_JOG_VEL.....	241
ML_KIN_E_JOG_VEL, application.....	242
ML_KIN_JOG.....	242
ML_KIN_MECHANIC_DATA.....	240
ML_KIN_PERF_DATA.....	240
ML_KIN_SAFE_ZONE_TYPE.....	231
ML_KIN_USED_AXES.....	231
ML_KINEMATICS_DATA_SM.....	237
ML_KinematicsInitType01.....	280, 281
ML_KinematicsInterfaceType01.....	283, 284
ML_KinGetDistanceBlendOption.....	149
ML_KinGetUserBit.....	215
ML_KinGetUserBit_DWORD.....	215
ML_KinGetUserDInt.....	216
ML_KinGetUserReal.....	217
ML_KinJog.....	287, 289
ML_KinJogV1.....	285, 286
ML_KinSetDistanceBlendOption.....	138
ML_KinSetUserBit.....	218
ML_KinSetUserDInt.....	219
ML_KinSetUserPoint.....	220
ML_KinSetUserReal.....	220
ML_Kintech.library.....	254
ML_POINT_TYPE.....	246
ML_POINT_TYPE, application.....	248
ML_POINTKIND.....	240
ML_Robot	
Config function blocks.....	128
Data types.....	229
Function blocks.....	112
Info function blocks.....	148
Info functions.....	148

Index

Movement function blocks.....	163
Parameter function blocks.....	189
RCL function blocks.....	201, 214
Read/write parameters.....	108
TeachIn function blocks.....	221
ML_Robot.library.....	107
ML_RobotGetBeltConfiguration.....	150
ML_RobotGetBeltDesyncRelConfiguration.....	151
ML_RobotGetBeltErrorReaction.....	152
ML_RobotGetMechanic.....	153
ML_RobotGetPerformance.....	155
ML_RobotGetTrafoValues.....	156
ML_RobotMovePoint.....	157
ML_RobotOpenCyclicPos.....	187
ML_RobotReadCyclicPos.....	158
ML_RobotSetPerformanceEnable.....	159
ML_RobotTransformPoint.....	160
ML_RobotWriteCyclicPos.....	188
ML_ROCO_FILE_DIR.....	247
ML_ROCO_NAME.....	247
ML_ROCO_NAME, application.....	247
ML_ROCO_POINT.....	248
ML_ROCO_POINT_NAME.....	248
ML_ROCO_POINT_NAME, application.....	248
ML_ROCO_WORLD_COORD, application.....	248
ML_RoCoDir.....	203
ML_RoCoGetPoint.....	206
ML_RoCoGetPointList.....	209
ML_RoCoSetPoint.....	211
ML_SET_MODE.....	246
ML_SetBeltConfiguration.....	140
ML_SetBeltDesyncRelConfiguration.....	139
ML_SetBeltErrorReaction.....	142
ML_SetBeltParameter.....	143
ML_SetCylindricTransform.....	145
ML_SetRotaryTableConfiguration.....	147
ML_SLOPE_TYPE.....	246
ML_SwitchOnPath.....	162
MLC_TABLE.....	351
MLPI COM	
Robot Control.....	29
MLPI core	
Robot Control.....	29
ModeCoordAB.....	291
ModeCoordAH.....	291
ModeCoordContinue.....	299
ModeCoordExternalFB.....	300
ModeCoordHoming.....	292
ModeCoordInterrupt.....	299
ModeCoordPosCircAbs.....	296
ModeCoordPosCircRel.....	297
ModeCoordPosDirAbs.....	293
ModeCoordPosDirRel.....	293
ModeCoordPosLinAbs.....	294
ModeCoordPosLinRel.....	295
ModeCoordRCL.....	300
ModeCoordStopping.....	298
Motion type	
Circular.....	33
Linear.....	32
PTP.....	34
Motions	
Continue.....	48
Interrupt.....	48
Stop.....	46
Wait.....	49
N	
Navigation system overview.....	304
O	
Operation modes.....	290
Absolute circular positioning.....	296
Absolute linear positioning.....	294
Absolute PTP positioning.....	293
Continue.....	299
External function block.....	300
Interrupt.....	299
Kinematics homing.....	292
Kinematics ready.....	291
Kinematics stopping.....	298
RCL.....	300
Relative circular positioning.....	297
Relative linear positioning.....	295
Relative PTP positioning.....	293
Orientation motions	
Circular.....	33
Linear.....	32
P	
Parameter access of the PLC program.....	189
PCS.....	22
Performance measurement.....	63
Positioning	
Absolute.....	34
Relative.....	34
Product coordinate system (PCS).....	22
Program for the kinematics interface.....	302
PTP.....	34
R	
RCL User Interface.....	214
REFERENCE_TO	
Kinematic data.....	109
Kinematic Data.....	110
Relative positioning.....	34
Robot Control	
Axis-relevant parameters.....	329
Command states.....	28
Commands.....	29
Control-related parameters.....	332
Diagnostics.....	351
Functions.....	29
Kinematic-related parameters.....	334
Parameter.....	329

Robot Control application example	
Extended user-defined transformation.....	321
User-defined transformation.....	317
Using coordinate systems.....	315
Robot Control application examples	
Belt synchronization.....	316
Robot Control V1 to Robot Control V2.....	367
IEC commands.....	376
RCL commands.....	367
RCL programs.....	372
Robot Control V2	
ACS/BCS transformations.....	73
Application examples.....	315
Blending.....	39
Continuous operation.....	36
Cyclical command value specification.....	61
Errors and error reactions.....	59
Functional description.....	19
Interface functions.....	65
Kinematics - context menu.....	67
Manual mode.....	34
Performance measurement.....	63
Robot folder - Dialog: Axis length and coupling.....	70
Robot folder – Dialog: Coordinate offset.....	71
Robot folder – Dialog: Motion limit values.....	70
Robot folder – Dialog: Scaling/Units.....	70
Robot folder – kinematics dialogs.....	69
Single-step operation.....	36
Slope types.....	40
Switch on path.....	54
Transformation schemes.....	73
Traversing range limits.....	57
Robot Control, commands and functions.....	27
Robot-Control V2	
System overview.....	19
RobotControl	
Kinematics – context menu: Diagnostics.....	68
Kinematics – context menu: Setup.....	68
Menu structure: Kinematics.....	65
S	
Safety alert symbol.....	12
Safety instructions.....	12
SERCOS_TABLE.....	360
Shared Memory	
KinData[].....	109
Shutdown.....	116
Signal words.....	12
Single-step operation.....	36
Standard interfaces at function blocks.....	115
State diagram	
Kinematics.....	26
StateDiagram	
Axisgroup.....	26
Support	
See service hotline.....	379

Symbols.....	12
T	
Target group.....	11
TE_KinInterfaceType01Prog.....	302
Teach-in	
Delete point file.....	39
Delete points.....	38
Load next point.....	38
Load point.....	38
Load point file.....	39
Save point.....	37
Save point file.....	38
Structure of the point file.....	37
Teach-In.....	36
Transformation	
Axis coordinates (ACS) / base coordinates (BCS).....	23
Base coordinates (BCS) / machine coordinates (MCS).....	24
Machine coordinates (MCS) / product coordinates (PCS).....	25
Transformation rule	
2-axis delta slide.....	94
2-axis delta with 1 optional orientation axis... ..	75
2-axis flex stamper.....	89
2-axis parallelogram.....	81
2-axis parallelogram V1.....	84
2-axis parallelogram V2.....	87
3-axis delta with 1 optional orientation axis... ..	85
3-axis delta with linear axes and 1 optional orientation axis.....	91
3-axis parallelogram V3.....	95
3-axis Scara with 1 optional orientation axis... ..	77
3-axis twisting crane.....	88
4-axis Scara with 1 optional orientation axis... ..	79
4-axis speedpicker.....	96
Belt kinematics.....	83
Cartesian.....	74
Extended user-defined.....	100
Inverted 2-axis parallelogram.....	82
User-defined.....	98
Transformation rules	
Overview.....	23
Transformations	
Robot Control V2.....	23
Traversing range limits.....	57
U	
Use cases	
Registering a user-defined transformation... ..	328
Sequential programming with the kinematics interface.....	326
User extension.....	313
Extend interface structures.....	314
Function block.....	314
Overview.....	313

Index

V

Visualization

Belt configuration.....	309
Jog/Setup mode.....	310
Kinematics Overview.....	305
Overview.....	302
Positioning overview.....	307
Referencing overview.....	306
Robot Control Language.....	309
Robot Control Language point.....	311
Robot Control Language program.....	312
System overview.....	303

Notes

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911341035

DOK-MLC***-ROCO****V13-RE03-EN-P