

IndraMotion MLC 14VRS

Robot Control V2

Reference Book
R911341588

Edition 07



Title IndraMotion
MLC 14VRS
Robot Control V2

Type of Documentation Reference Book

Document Typecode DOK-MLC***-ROCO****V14-RE07-EN-P

Internal File Reference RS-527e93f498bd3aa00a6846a501f57a0f-7-en-US-5

Change Record

Edition	Release date	Note
Edition 01	2015-01	First edition for IndraMotion MLC 14VRS
Edition 02	2015-11	Update, KinInterface/ML_KinTech/Visu, ML_Robot/ ML_RobotMovePoint, transformer 22/23/39
Edition 03	2016-06	New: KinInterface HMI Template New: RoCo_03_073_4-axis Scara with crank, update: RoCo_03_018_Contact_kinematics ML_Robot: ML_RobotGetSafeZones
Edition 04	2016-08	Robot Control: Functional description new: MoveID, Jump ML_Robot: ML_RobotMoveJumpAbsolute, ML_RobotGetMoveID, ML_RobotSetMoveID
Edition 05	2017-02	Update of interface functions, functional description of HalfJump/Jump, Transformation 81: 3-axis Canfield
Edition 06	2017-09	Integration Hexpod - Transformation, New chapter: "Connecting third-party robotic systems!", Description uniVAL drive
Edition 07	2018-03	New chapter "Point", "Axis types", supplement "Arm behavior selection for PTP motions"

Copyright © Bosch Rexroth AG 2018

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

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

Editorial Department System IndraLogic & Basis Motion Logic, MaEI (TaDo/MePe)

Table of Contents

	Page
1 About this documentation.....	13
1.1 Validity of the documentation.....	13
1.2 Documentation structure.....	13
1.3 Using safety instructions.....	14
1.3.1 Structure of the safety instructions.....	14
1.3.2 Explaining signal words and safety alert symbol.....	14
1.3.3 Symbols used.....	15
1.3.4 Explaining the signal alert symbol on the device.....	15
1.4 Required and supplementing documentation XLC/MLC.....	15
1.5 Names and abbreviations.....	21
1.6 Customer feedback.....	21
 2 Functional description.....	 23
2.1 System overview.....	23
2.2 Coordinate systems and transformations.....	23
2.2.1 Coordinate systems.....	23
Coordinate systems, overview.....	23
Types of coordinate systems.....	24
Axis coordinate system (ACS).....	25
Base coordinate system (BCS).....	26
Machine coordinate system (MCS).....	26
Product coordinate system (PCS).....	27
2.2.2 Transformations.....	27
Transformations, overview.....	27
ACS/BCS transformation.....	28
BCS/MCS transformation.....	29
MCS/PCS transformation.....	29
2.3 Axis types.....	30
2.3.1 Axis types, overview.....	30
2.3.2 Positioning axes.....	30
2.3.3 Orientation axes.....	30
2.3.4 Auxiliary axes.....	31
2.3.5 Belt axes.....	31
2.4 Point.....	31
2.4.1 Point, overview.....	31
2.4.2 Configuring the point representation.....	32
2.5 State diagram of a kinematics.....	33
2.6 Robot Control, commands and functions.....	34
2.6.1 Robot Control, commands and functions, overview.....	34
2.6.2 Command state.....	36
2.6.3 List of all commands and functions.....	36
2.7 Motion types.....	39
2.7.1 Motion types, overview.....	39
2.7.2 "Linear" motion type.....	39

Table of Contents

	Page
2.7.3	“Circular” motion type..... 40
2.7.4	Point-to-point motion type..... 41
2.7.5	Motion type Jump..... 42
2.7.6	Motion type HalfJump..... 44
2.7.7	Absolute positioning..... 47
2.7.8	Relative positioning..... 47
2.7.9	Commanding modulo-scaled axes..... 47
2.8	Manual mode..... 47
2.8.1	Manual mode, general information..... 47
2.8.2	Single-step operation..... 48
2.8.3	Continuous operation..... 49
2.9	Teach-In..... 49
2.9.1	Teach-In, general information..... 49
2.9.2	Structure of the point file..... 50
2.9.3	Write point..... 50
2.9.4	Read point..... 51
2.9.5	Read next point..... 51
2.9.6	Delete points..... 51
2.9.7	Save point file..... 51
2.9.8	Load point file..... 52
2.9.9	Delete point file..... 52
2.10	Blending..... 52
2.10.1	Blending, general information..... 52
2.10.2	Blending radius..... 53
2.10.3	Slope types..... 53
2.10.4	Block_Slope..... 54
2.10.5	Cont_Slope..... 56
2.10.6	Blending possibilities with various motion types..... 59
2.11	Robot Control, Move ID..... 59
2.11.1	Robot Control, Move ID, setting..... 60
2.11.2	Robot Control, Move ID, blending..... 60
2.12	Stopping, Interrupting, Continuing Motions..... 62
2.12.1	Stopping Motions..... 62
2.12.2	Interrupting Motions..... 63
2.12.3	Continuing Motions..... 64
2.12.4	Waiting..... 64
2.13	Belt-synchronous motion..... 65
2.13.1	Belt-synchronous motion, general information..... 65
2.13.2	Belt configuration..... 66
2.13.3	Belt values and points on the belt..... 67
2.13.4	Belt operation..... 69
2.13.5	Adaptive motion..... 72
2.13.6	Blending and belt synchronization..... 73
2.13.7	Summary..... 73
2.14	Switching on the belt..... 73
2.15	Traversing range limits..... 76

Table of Contents

	Page
2.16	Errors and error reactions..... 78
2.16.1	Errors and error reactions, general information..... 78
2.16.2	Error reaction of the commanded motion..... 79
2.16.3	Belt-specific error reaction..... 79
2.17	Cyclic command value specification..... 80
2.17.1	Cyclic command value specification, general information..... 80
2.17.2	Start and operation of the cyclic command value specification..... 80
2.17.3	Cyclic command value specification..... 81
2.17.4	Termination of cyclic command value specification..... 81
2.18	Performance measurement..... 82
3	Interface functions..... 85
3.1	Interface function, general information..... 85
3.2	Robot folder..... 85
3.2.1	Robot folder, general information..... 85
3.2.2	Adding a new kinematics..... 85
3.2.3	Kinematic commissioning..... 87
	Overview..... 87
	Safety functions..... 89
	Grouping a kinematics..... 90
	Jogging..... 90
	Positioning..... 91
	Belt configuration..... 92
	Product coordinates..... 94
	PLC configuration..... 94
3.2.4	PathViewer..... 95
	Overview..... 95
	Navigation and operation..... 96
	PathViewer context menu..... 97
3.3	Kinematic folder..... 98
3.3.1	Kinematic folder, general information..... 98
3.3.2	Kinematics – Context menu..... 98
	Kinematics – Context menu or context menu item "Kinematics", general information..... 98
	Initial commissioning..... 99
	Diagnostics..... 102
3.3.3	Kinematics – Dialogs..... 104
	Overview on dialogs..... 104
	Scaling/units..... 104
	Axis lengths and couplings..... 105
	Motion limit values..... 105
	Coordinate offset..... 106
4	Transformation schemes..... 107
4.1	Transformation schemes, overview..... 107
4.2	Transformation schemes..... 108

Table of Contents

	Page
4.2.1 Cartesian.....	108
4.2.2 2-axis delta with 1 optional orientation axis.....	109
4.2.3 4-axis Scara with linear lifting axis.....	111
4.2.4 3-axis Scara with 1 optional orientation axis.....	115
4.2.5 4-axis Scara with 1 optional orientation axis.....	119
4.2.6 2-axis parallelogram.....	123
4.2.7 Inverted 2-axis parallelogram.....	125
4.2.8 2-axis press.....	126
4.2.9 Contact kinematics.....	128
4.2.10 2-axis parallelogram V1.....	129
4.2.11 3-axis delta with 1 optional orientation axis.....	131
4.2.12 2-axis parallelogram V2.....	133
4.2.13 3-axis delta linear with 1 optional orientation axis.....	134
4.2.14 5-axis delta.....	137
4.2.15 Adabot.....	141
4.2.16 2-axis delta with asymmetric TCP.....	144
4.2.17 3-Axis Twisting Crane.....	145
4.2.18 3-axis flex stamper.....	147
4.2.19 3-axis delta with linear axes with 1 optional orientation axis.....	149
4.2.20 2-axis delta slide.....	152
4.2.21 3-Axis Parallelogram V3.....	153
4.2.22 6-axis Cartesian.....	154
4.2.23 3-axis delta with 90 degree angles.....	156
4.2.24 4-axis speedpicker.....	159
4.2.25 Stewart platform.....	162
4.2.26 4-axis Scara with crank.....	166
4.2.27 3-axis Canfield	170
4.3 User-defined transformation.....	174
4.4 Extended user-defined transformation.....	176
4.5 Mechanical information.....	180
5 ML_Robot.library.....	183
5.1 Introduction and overview.....	183
5.2 Global variables.....	184
5.2.1 Introduction and overview.....	184
5.2.2 GVL_RobotConstant, global constants.....	185
5.2.3 KinData[] - Kinematic data in the shared memory	185
5.2.4 DV_Kin[] - Kinematic data via function variable access	186
5.3 Function blocks.....	188
5.3.1 Function blocks, overview.....	188
5.3.2 Standard interfaces at function blocks.....	190
Motivation.....	190
Function block types.....	191
Block types and their interfaces.....	191
Inputs and outputs of state-controlled function blocks, final processing.....	193
Inputs and outputs of edge-controlled function blocks, final processing.....	196

Table of Contents

	Page
Inputs and outputs of state-controlled function clocks, permanent processing.....	199
Inputs and outputs of edge-triggered function blocks, permanent processing.....	202
5.3.3 InBuffer output at ML_Robot function blocks.....	204
5.3.4 Kinematics/Config.....	205
MC_GroupReset.....	205
MC_SetCartesianTransform.....	206
MC_SetKinTransform.....	208
MC_SetKinTransformExt.....	210
MC_UngroupAllAxes.....	211
ML_AddMechanicData.....	212
ML_GroupAllAxes.....	213
ML_GroupSetSafeZone.....	214
ML_SetBeltDesyncRelConfiguration.....	216
ML_SetBeltConfiguration.....	217
ML_SetBeltErrorReaction.....	218
ML_SetCylindricTransform.....	220
ML_SetRotaryTableConfiguration.....	222
ML_RobotSetMoveID.....	223
5.3.5 Kinematic/Info.....	224
MC_GroupReadActualPosition.....	224
ML_RobotGetBeltConfiguration.....	226
ML_RobotGetBeltDesyncRelConfiguration.....	227
ML_RobotGetBeltErrorReaction.....	228
ML_RobotGetCmdInfo.....	229
ML_RobotGetMechanic.....	231
ML_RobotGetMoveID.....	232
ML_RobotGetPerformance.....	233
ML_RobotGetSafeZones.....	234
ML_RobotGetTrafoValues.....	236
ML_RobotMovePoint.....	237
ML_RobotReadCyclicChannel.....	238
ML_RobotSetPerformanceEnable.....	240
ML_RobotTransformPoint.....	241
ML_SwitchOnPath.....	242
5.3.6 Kinematics/Movement.....	244
MC_GroupContinue.....	244
MC_GroupInterrupt.....	245
MC_GroupStop.....	246
MC_MoveCircularAbsolute.....	248
MC_MoveCircularRelative.....	250
MC_MoveDirectAbsolute.....	252
MC_MoveDirectRelative.....	255
MC_MoveLinearAbsolute.....	257
MC_MoveLinearRelative.....	260
ML_BeltDeSynchronisationWithLimits.....	262
ML_BeltSynchronisationWithLimits.....	263

Table of Contents

	Page
ML_GroupJogCont.....	265
ML_GroupJogStep.....	267
ML_GroupWait.....	269
ML_RobotOpenCyclicChannel.....	270
ML_RobotOpenDirectCyclicChannel.....	272
ML_RobotWriteCyclicChannel.....	273
ML_RobotMoveJumpAbsolute.....	274
5.3.7 Kinematics/Parameter.....	277
Accessing kinematic data.....	277
MB_GroupReadListParameter.....	278
MB_GroupReadParameter.....	280
MB_GroupReadRealParameter.....	281
MB_GroupReadSercosAttribute.....	282
MB_GroupReadSercosDataStatus.....	283
MB_GroupReadStringParameter.....	284
MB_GroupWriteListParameter.....	285
MB_GroupWriteParameter.....	287
MB_GroupWriteRealParameter.....	288
MB_GroupWriteStringParameter.....	289
5.3.8 Kinematic/TeachIn.....	290
ML_GroupDeletePointFile.....	290
ML_GroupDeletePoints.....	291
ML_GroupLoadPointFile.....	292
ML_GroupReadNextPoint.....	293
ML_GroupReadPoint.....	295
ML_GroupSavePointFile.....	296
ML_GroupWritePoint.....	297
5.4 Data types.....	298
5.4.1 Data types, overview.....	298
5.4.2 Data types/Config.....	300
MC_KIN_REF.....	300
MC_KIN_REF, overview.....	300
MC_KIN_REF, initialization.....	300
MC_KIN_REF, forward.....	301
MC_KIN_REF, inverse.....	301
Error handling with interface functions.....	301
MC_KIN_REF_EXT.....	302
MC_KIN_REF_EXT, overview.....	302
MC_KIN_REF_EXT, DetermineMemory.....	302
MC_KIN_REF_EXT, forward.....	303
MC_KIN_REF_EXT, GetMechanic.....	303
MC_KIN_REF_EXT, GetUsedAxes.....	303
MC_KIN_REF_EXT, initialization.....	303
MC_KIN_REF_EXT, inverse.....	304
Error handling with interface functions.....	304
ML_BELT_ERROR_REACTION_MODE.....	305

Table of Contents

	Page
ML_KIN_AXES_CONFIG_DATA.....	305
ML_KIN_AXIS_CONFIG.....	305
ML_KIN_SAFE_ZONE_TYPE.....	306
ML_KIN_USED_AXES.....	306
5.4.3 Data types/Info.....	306
E_KIN_MECH_CMD.....	306
ML_KINEMATICS_DATA_SM.....	307
ML_KIN_CMD_INFO.....	308
ML_KIN_CS_TYPE.....	308
ML_KIN_INFO_TYPE.....	308
ML_KIN_MECHANIC_DATA.....	309
ML_KIN_PERF_DATA.....	309
ML_KIN_SAFE_ZONE_DATA.....	310
ML_POINTKIND.....	310
5.4.4 Data types/TeachIn.....	310
E_KIN_TEACH_IN_POINT_FILE_OPTIONS.....	310
5.4.5 Data types/Movement.....	311
ML_ARR_COORDS.....	311
ML_CIRCMODES.....	311
ML_COORDSYS.....	312
ML_CYCLIC_MODE.....	313
ML_SET_MODE.....	313
ML_SLOPE_TYPE.....	313
6 Kinematic interface.....	315
6.1 Introduction.....	315
6.2 Functional description.....	315
6.2.1 Overview.....	315
6.2.2 Kinematic interface program.....	317
6.2.3 Kinematic interface visualizations.....	317
Overview.....	317
System overview visualization.....	318
Kinematic overview visualization.....	321
Homing overview visualization.....	321
Positioning overview (position) visualization.....	322
Belt configuration (ConveyorBelt) visualization.....	324
Jog mode/Setup visualization.....	324
TeachIn visualization.....	325
PCS configuration (ProductCoordSystem) visualization.....	327
6.2.4 Operation modes.....	328
Overview.....	328
Kinematic ready, drives ready.....	329
Kinematic ready, drives halt.....	329
Kinematic homing.....	330
Absolute PTP positioning.....	330
Relative PTP positioning.....	331

Table of Contents

	Page
Absolute linear positioning.....	332
Relative linear positioning.....	333
Absolute circular positioning.....	334
Relative circular positioning.....	335
Absolute Jump positioning.....	336
Kinematic stop.....	337
Operating mode "Interrupted".....	338
Operating mode "Continue".....	338
Operating mode "external function block".....	339
Cyclic position command value specification.....	339
6.2.5 Dialog-based kinematics commanding.....	340
6.2.6 Kinematic interface user extensions.....	343
Notes regarding the implementation of user extensions.....	343
MB_KinInterfaceType01 extension.....	344
Extending the kinematic interface structure MB_KinInterfaceType01.....	344
Extending the TE_KinematicsInterface function block.....	344
MB_KinInterfaceType11 extension.....	344
Extending the existing interface.....	344
Adding a new interface.....	345
6.2.7 Upgrading to an event-controlled kinematic interface (type 11).....	345
6.3 KinematicInterface HMI template.....	346
6.3.1 Introduction.....	346
6.3.2 KinInterface screens.....	347
Kinematic status.....	347
Kinematic jogging.....	348
Teaching points.....	349
Position kinematics.....	351
7 ML_KinTech.library.....	353
7.1 Introduction and overview.....	353
7.2 KinInterface/Const.....	353
7.2.1 Global_MB_KinInterfaceVars.....	353
7.3 KinInterface/FB.....	355
7.3.1 ML_KinematicsInitType01.....	355
7.3.2 ML_KinematicsInterfaceType01.....	358
7.3.3 ML_KinematicsInterfaceType11.....	361
7.3.4 KinInterface/FB/DataTypes.....	363
Kin_Axis_Config_Elements.....	363
Kin_Axis_Type.....	364
7.4 KinInterface/Type.....	364
7.4.1 Kinematics interface structures, overview.....	364
7.4.2 Kinematic interface operating modes, overview.....	365
7.4.3 MB_KINEMATICS_MODE_UNION.....	365
7.4.4 MB_KINEMATICS_MODE_BITS.....	366
7.4.5 MB_KINEMATICS_MODE.....	367
7.4.6 KinInterface/Type/Control.....	368

Table of Contents

	Page
Folder Kin_Interface/Type/Control, overview.....	368
MB_KINEMATICS_CONTROL_TYPE01.....	369
MB_KINEMATICS_ADMINISTRATION.....	369
MB_KINEMATICS_CONVEYOR_BELT.....	372
MB_KINEMATICS_TEACH_IN.....	373
MB_KINEMATICS_TEACH_IN_MODE.....	374
MB_KINEMATICS_ADMIN_CHANGE_OPMODE.....	375
MB_KINEMATICS_PCS_CONFIG.....	375
MB_KINEMATICS_SETUP.....	376
MB_KINEMATICS_COORDINATED.....	377
ITF_KINEMATICS structures.....	379
7.4.7 KinInterface/Type/Status.....	380
Folder Kin_Interface/Type/Status, overview.....	380
MB_KINEMATICS_STATUS_TYPE01.....	380
MB_KINEMATICS_ADMIN_STATUS.....	381
MB_KINEMATICS_CONV_BELT_STATUS.....	383
MB_KINEMATICS_TEACH_IN_STATUS.....	384
MB_KINEMATICS_PCS_STATUS.....	384
MB_KINEMATICS_DIAGNOSIS.....	385
MB_KINEMATICS_SETUP_STATUS.....	386
7.5 KinJog.....	386
7.5.1 KinJog/FB.....	386
ML_KinJog.....	386
7.5.2 KinJog/Type.....	389
ML_JOGMODE.....	389
8 Application examples.....	391
8.1 Application examples, overview.....	391
8.2 Using coordinate systems.....	391
8.3 Belt synchronization.....	392
8.4 User-defined transformation.....	393
8.5 Extended user-defined transformation.....	396
8.6 Extending a point.....	402
8.7 Belt synchronization with "Adaptive Motion".....	402
8.8 Determining the command buffer size.....	403
8.9 Kinematic synchronization with cyclic command value specification.....	405
8.10 Sequential programming with the kinematic interface.....	408
8.11 Registration of a user-defined transformation using the KinematicInterface.....	410
9 Parameter.....	413
9.1 Parameters, general information.....	413
9.2 Axis-related parameters.....	413
9.2.1 Axis-related parameters, general information.....	413
9.2.2 A-0-0028, Travel range limit switch.....	413
9.2.3 A-0-0029, Position polarities.....	413

Table of Contents

	Page
9.2.4	A-0-0030, Position limit value, positive..... 414
9.2.5	A-0-0031, Position limit value, negative..... 415
9.2.6	A-0-0032, Velocity limit value, positive..... 415
9.2.7	A-0-0033, Velocity limit value, negative..... 415
9.2.8	A-0-0034, Acceleration limit value, bipolar..... 415
9.2.9	A-0-0035, Velocity limit value, bipolar..... 415
9.2.10	A-0-0036, Jerk limit value, bipolar..... 416
9.2.11	A-0-0038, Torque/force limit value, bipolar..... 416
9.2.12	A-0-0039, Torque/force limit value, positive..... 416
9.2.13	A-0-0040, Torque/force limit value, negative..... 417
9.2.14	A-0-0048, Velocity polarities..... 417
9.2.15	A-0-0049, Torque-/force polarities..... 418
9.2.16	A-0-0224, Rapid stop deceleration..... 418
9.2.17	A-0-0225, Rapid stop jerk..... 418
9.2.18	A-0-0240, Feedrate override..... 419
9.2.19	A-0-0241, Feedrate override control word..... 419
9.3	Control-related parameters..... 419
9.3.1	Control-related parameters, general information..... 419
9.3.2	C-0-0123, Maximum number of kinematics..... 419
9.3.3	C-0-2480, Configured kinematic number..... 419
9.3.4	C-0-2482, List of non-linked axes..... 419
9.3.5	C-0-2483, List of configured kinematics..... 419
9.3.6	C-0-2484, Kinematic configuration list..... 420
9.4	Kinematic-related parameters..... 420
9.4.1	Kinematic-related parameters, general information..... 420
9.4.2	K-0-0001, Logic kinematic number..... 420
9.4.3	K-0-0002, Kinematic name..... 420
9.4.4	K-0-0003, Maximum path velocity..... 421
9.4.5	K-0-0005, Software limit switch, positive, MCS..... 421
9.4.6	K-0-0006, Software limit switch, negative, MCS..... 421
9.4.7	K-0-0007, Axis number of kinematics..... 421
9.4.8	K-0-0008, Axis configuration list..... 421
9.4.9	K-0-0009, Osci: list of all K-parameters that can be oscilloscoped..... 423
9.4.10	K-0-0010, List of all K-parameters..... 423
9.4.11	K-0-0011, List of all invalid K-parameters..... 424
9.4.12	K-0-0012, Checksum of all K-parameters..... 424
9.4.13	K-0-0013, IDN list of checksum K-parameters..... 424
9.4.14	K-0-0016, Project identification number..... 424
9.4.15	K-0-0019, Kinematic type..... 424
9.4.16	K-0-0020, Kinematic diagnostic message..... 425
9.4.17	K-0-0021, Kinematic status..... 425
9.4.18	K-0-0022, Extended kinematic status..... 426
9.4.19	K-0-0023, Kinematic diagnostic number..... 427
9.4.20	K-0-0028, Velocity limit auxiliary axes..... 427
9.4.21	K-0-0029, Acceleration limit auxiliary axes..... 427
9.4.22	K-0-0030, Jerk limit auxiliary axes..... 428

Table of Contents

	Page
9.4.23 K-0-0031, Transformation scheme.....	428
9.4.24 K-0-0034, Maximum path acceleration (bipolar).....	428
9.4.25 K-0-0035, Maximum path jerk (bipolar).....	429
9.4.26 K-0-0037, Jerk for move commands during deceleration phase.....	429
9.4.27 K-0-0040, Transformation, lengths of the mechanics.....	429
9.4.28 K-0-0041, Transformation, factors.....	429
9.4.29 K-0-0042, Offset of BCS system.....	429
9.4.30 K-0-0050, Distance units MCS.....	430
9.4.31 K-0-0051, Velocity units MCS.....	430
9.4.32 K-0-0052, Acceleration units MCS.....	431
9.4.33 K-0-0053, Jerk units MCS.....	432
9.4.34 K-0-0054, Measuring unit for kinematic data.....	432
9.4.35 K-0-0070, Belt number of kinematics.....	433
9.4.36 K-0-0090, Taskpriority preparation.....	433
9.4.37 K-0-0095, ID of movement.....	433
9.4.38 K-0-0096, Peak value of command buffer size.....	433
9.4.39 K-0-0097, Maximum command buffer size.....	433
9.4.40 K-0-0098, Current command buffer size.....	433
9.4.41 K-0-0099, Kinematic configuration.....	434
9.4.42 K-0-0100, Current position, Cartesian Machine Coordinate System.....	435
9.4.43 K-0-0102, Current path velocity.....	435
9.4.44 K-0-0103, Current path acceleration.....	435
9.4.45 K-0-0104, Current path jerk.....	435
9.4.46 K-0-0150, Interpolated command position.....	435
9.4.47 K-0-0152, Interpolated path velocity.....	436
9.4.48 K-0-0153, Interpolated path acceleration.....	436
9.4.49 K-0-0154, Interpolated path jerk.....	436
9.4.50 K-0-0223, Selection of error reaction.....	436
9.4.51 K-0-0224, Emergency halt deceleration.....	436
9.4.52 K-0-0225, Emergency halt jerk.....	437
9.4.53 K-0-0240, Feedrate override.....	437
9.4.54 K-0-0241, Feedrate override control word.....	437
9.4.55 K-0-0242, Acceleration override.....	437
9.4.56 K-0-0243, Deceleration override.....	437
9.4.57 K-0-0250, Continuous slope lead time.....	438
9.4.58 K-0-0815, Odometer.....	438
9.4.59 K-0-1000, Command: Load basic kinematic parameters.....	438
9.4.60 K-0-2100, List of all backup operation K-parameters.....	438
10 Diagnostics.....	441
11 Connecting third-party robotic systems.....	443
11.1 General information.....	443
11.2 uniVAL drive.....	443
11.2.1 Overview.....	443

Table of Contents

	Page
11.2.2 Creating axes.....	443
11.2.3 Configuring axes.....	444
11.2.4 Configuring the kinematic transformation.....	444
12 Upgrading from Robot Control V1 to Robot Control V2.....	445
12.1 Overview.....	445
12.2 Adjusting the IndraWorks project.....	445
12.3 Converting RCL commands.....	446
12.4 Converting RCL programs.....	451
12.5 Converting IEC commands.....	455
13 Service and support.....	459
Index.....	461

1 About this documentation

1.1 Validity of the documentation

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

Required qualification: Individual who is able to assess the tasks assigned and to 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:

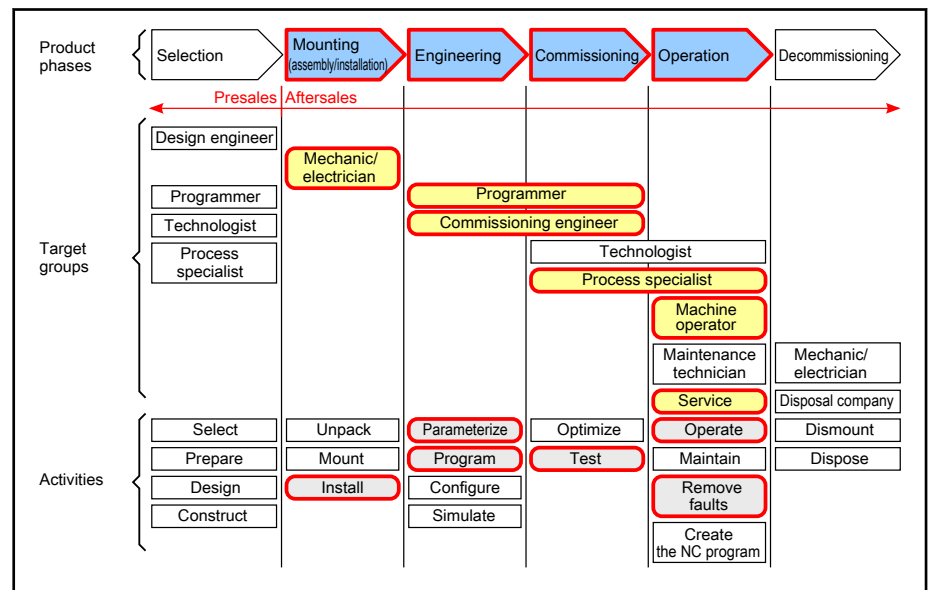


Fig. 1-1: Application phases

1.2 Documentation structure

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

Included are

- [Functional description, page 23](#), functional scope of Robot Control V2
- [Interface functions, page 85](#), description of the IndraWorks interface of the "Robot" component
- [Transformation schemes, page 107](#), description of the transformations
- [Application examples on page 391](#)
- [Kinematic interface on page 315](#) with library [ML_KinTech.library on page 353](#)
- Programming references for further information

For **both Robot Control types**, the description of the mutual library is contained in [ML_Robot.library, page 183](#).

About this documentation

1.3 Using safety instructions

1.3.1 Structure of the safety instructions

The safety instructions are structured as follows:

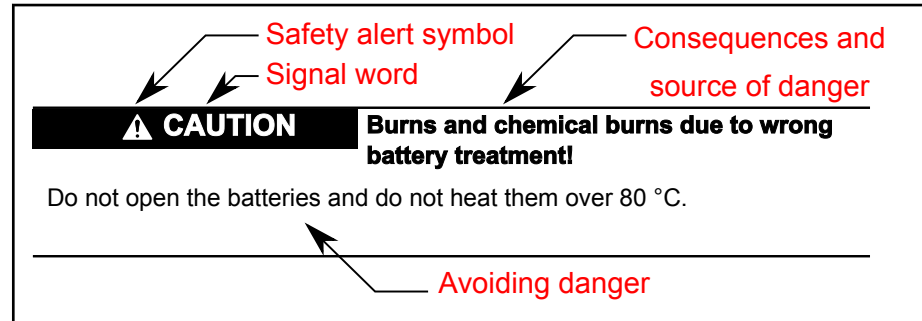


Fig. 1-2: Structure of the safety instructions

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 draws attention to the safety instruction and indicates the risk potential.

The safety alert symbol (triangular safety reflector with exclamation marks), preceding the signal words Danger, Warning, Caution indicates hazards for persons.

⚠ 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 can occur.

NOTICE

In case of non-compliance with this safety instruction, material damage can occur.

1.3.3 Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.3.4 Explaining the signal alert symbol on the device



If this symbol is on your device, you have to observe the documentation on the device. The respective documentation informs on the type of hazard as well as the steps required to avoid this hazard.

1.4 Required and supplementing documentation XLC/MLC

Documentation titles with type codes and part numbers

System overview
IndraLogic XLC IndraMotion MLC 14VRS System Overview DOK-XLCMLC-SYSTEM**V14-PRRS-EN-P, R911341504 This documentation provides an overview on the possible hardware/software components of the automation systems IndraLogic XLC and IndraMotion MLC of the mentioned version. The documentation facilitates assembling a system.
First steps
IndraMotion MLC 14VRS First Steps DOK-MLC***-F*STEP**V14-CORS-EN-P, R911341517 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.
IndraLogic XLC 14VRS First Steps DOK-XLC***-F*STEP**V14-CORS-EN-P, R911341489 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.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Project Conversion DOK-XLCMLC-PROCONV*V14-APRS-EN-P, R911341494 This documentation describes the project conversion of the IndraMotion MLC.

Tab. 1-2: XCL/MLC documentation overview - First steps

About this documentation

Engineering
IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 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.
IndraLogic XLC IndraMotion MLC 14VRS Function Description DOK-XLCMLC-FUNC****V14-APRS-DE-P, R911341700 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

Tab. 1-3: XCL/MLC documentation overview - Engineering

Diagnostics and service
IndraLogic XLC IndraMotion MLC 14VRS Diagnostics DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481 This documentation includes all control diagnostics implemented in the control systems IndraLogic XLC and IndraMotion MLC.
IndraLogic XLC IndraMotion MLC 14VRS Parameters DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479 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.
IndraLogic XLC IndraMotion MLC 14VRS Commissioning DOK-XLCMLC-STARTUP*V14-CORS-EN-P, R911341502 This documentation describes the steps to commission and service the IndraMotion MLC and IndraLogic XLC systems. It includes checklists for frequent tasks and a detailed description of the steps.
WebAssistant DOK-GENERL-WEB*ASSIST*-APRS-EN-P, R911381469 This documentation describes the WebAssistant. The WebAssistant is a webbased diagnostic tool used to access a control system via an Ethernet highspeed connection. The WebAssistant allows OEMs, end users and service engineers to access and to remotely diagnose a system. A PC with Internet Explorer version 11 or higher, Firefox version 45 or above or Google Chrome version 54 or above is required. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM2 • IndraLogic XLC L25/L45/L65/L75/VEP
IndraWorks IndraMotion Service Tool DOK-IWORKS-IMST*****-APRS-DE-P, R911341383 This documentation describes the IndraMotion Service Tool (IMST). IMST is a web-based diagnostic tool used to access a control system via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. The PC has to use at least Internet Explorer 8, Firefox 3.5 or a higher version. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM12/XM2/VPx • IndraLogic XLC L25/L45/L65/L75

Tab. 1-4: XLC/MLC documentation overview - Diagnostics and service

Firmware
Rexroth IndraDrive MPx-20, Functions ¹⁾ DOK-INDRV*-MP*-20VRS**-AP01-EN-P, R911345608 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-20, MPM-20, MPC-20 and MPE-20.
Rexroth IndraDrive MPx-16 to MPx-20 und PSB, Parameters DOK-INDRV*-GEN1-PARA**-RERS-EN-P, R911328651 This documentation contains the descriptions of all parameters implemented in the firmware for drive controllers of the IndraDrive range. It supports the drive controller parameterization.
Rexroth IndraDrive MPx-16 to MPx-20 and PSB, Diagnostic Messages DOK-INDRV*-GEN1-DIAG**-RERS-EN-P, R911326738 This documentation contains the descriptions of all diagnostic messages implemented in the following firmware products: • Drive controller firmware variants MPx-16 to MPx-20 • PSB firmware variants for "HMu05" and "KMV03" type supply units • Firmware variants for "HMu01" type supply units It assists machine operators and installation programmers with troubleshooting.
Hardware
Rexroth IndraDrive Control Sections CSB02, CSE02, CSH02, CDB02 DOK-INDRV*-Cxx02*****-PRRS-EN-P, R911338962
Rexroth IndraDrive Cs Drive Systems with HCS01 DOK-INDRV*-HCS01*****-PRRS-EN-P, R911322210
Rexroth IndraDrive Mi Drive Systems with KCU02, KSM02, KMS02/03, KMV03 DOK-INDRV*-KCU02+KSM02-PRxx-EN-P, R911335703
Rexroth IndraDrive ML Drive Systems with HMu05 DOK-INDRV*-Hxx05*****-PRxx-EN-P, R911344279

1) In preparation
Tab. 1-5: Overview of documentations for drive controllers

PLC
IndraWorks 14VRS PLC Programming System IndraLogic 2G DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic use, first steps, visualization, menu items and editors.
IndraWorks 14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V14-LIRS-EN-P, R911341491 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCOpen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.

Tab. 1-6: XCL/MLC documentation overview - PLC

About this documentation

Field buses
IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. The focus of this documentation is on the configuration, parameterization, commissioning and the diagnostics of the different periphery connections. This documentation is the basis for the online help.
IndraWorks 14VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V14-LIRS-EN-P, R911343575 This documentation describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDPslave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
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.
Sercos System Manual for I/O Devices DOK-CONTRL-ILS3*****-APxx-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a Sercos interface.

xx

Corresponding edition

Tab. 1-7:

XCL/MLC documentation overview - Field buses

HMI
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" facilitates working with the "Rexroth WinStudio" software for optimal results. This document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI Connection DOK-XLCMLC-HMI*****V14-APRS-EN-P, R911341497 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.

Tab. 1-8:

XCL/MLC documentation overview - HMI

Technology
IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487 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.
IndraMotion MLC 14VRS Technology Libraries DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511 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.
IndraWorks 14VRS Energy Efficiency Management DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229 This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
IndraMotion MLC 14VRS RegisterControl (Application Description) DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507 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.
Rexroth IndraMotion MLC 14VRS Winder Function Application DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513 This application-related system documentation describes the application of the winder technology functions.
Rexroth IndraWorks CamBuilder 13VRS DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.
IndraMotion MLC 14VRS Robot Control V2 DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588 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 documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

Tab. 1-9: XCL/MLC documentation overview - Technology

Open Core Engineering - OCE
IndraLogic XLC IndraMotion MLC 14VRS First Steps MLPI (online help) DOK-XLCMLC-MLPI****V14-CORS-EN-E, R911341033 This document describes the installation and the commissioning of the MLPI interface.
IndraLogic XLC IndraMotion MLC 14VRS Automation Interface (online help) DOK-XLCMLC-AUT*INT*V14-APRS-EN-E, R911341499 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.

Tab. 1-10: XCL/MLC documentation overview - Open Core Engineering - OCE

About this documentation

SafeLogic
IndraWorks SafeLogic 14VRS First Steps DOK-IWORKS-SL*STEP*V14-CORS-EN-P, R911341520 This documentation describes the initial commissioning of the Safety function module and its external Safety periphery. Taking the example of a project, all required steps to commission a simple Safety application are executed.
IndraLogic XLC IndraMotion MLC 14VRS SafeLogic System Overview DOK-XLCMLC-SL**SYS*V14-PRRS-EN-P, R911341696 This documentation describes the project planning, configuration, creation and commissioning of safety-related devices.
IndraWorks 14VRS SafeLogic - Project Configuration, Version 1.2.0.0. DOK-IWORKS-SL**PRJ*V14-APRS-EN-P, R911341694 This documentation describes the configuration of Safety applications.
IndraControl CFL01.1-F1 Safety Function Module DOK-CONTRL-SL**FM*****-ITRS-DE-P, R911336576 This documentation is provided with the hardware and describes the commissioning, the installation, the decommissioning and disposal of the hardware.

Tab. 1-11: XCL/MLC documentation overview - SafeLogic

Hardware
Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRxx-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.
Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-EN-P, R911328474 This documentation describes the IndraControl L25 controls.
Rexroth IndraControl Lxx 14VRS Function Modules DOK-CONTRL-FM*LXX**V14-APRS-EN-P, R911341583 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.
IndraControl XM21, XM22 Controls DOK-CONTRL-IC*XM2*****-ITRS-EN-P, R911340667 This documentation describes the IndraControl XM21/XM22 controls.
IndraControl XFE 01.1 Extension Modules Profibus, RT-Ethernet, Sercos, CAN DOK-CONTRL-XFE**EXTMOD-ITRS-EN-P, R911345570 This documentation describes all extension modules of the XM controls.

xx Corresponding edition

Tab. 1-12: XCL/MLC documentation overview - Hardware

Hydraulics
Rexroth IndraMotion MLC 14VRS Hydraulic Functions DOK-MLC***-TF*HMOT*V14-LIRS-EN-P, R911341907 This documentation describes the function blocks, functions and data types of the MH_TechHydrBase and MH_TechHydrMotion libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Sequential Programming DOK-XLCMLC-SEQPROG*V14-LIRS-EN-P, R911341909 This documentation describes the sequential programming options, also when using GAT, and the MH_HydrControlProg library.

Tab. 1-13: XCL/MLC documentation overview - Hydraulics

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. 1-14: Names and abbreviations used

1.6 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

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
- IndraMotion MLC L75: 16 kinematics

One kinematics may comprise up to 16 axes.



There are two Robot Control versions:

- Robot Control V1
- Robot Control V2

Up to version 13VRS of the MLC firmware, Robot Control V1 is supported.

From version 13VRS of the MLC firmware, Robot Control V2 is supported.

In case of new projects, only use Robot Control V2. Robot Control V1 is available in version 13VRS only due to compatibility reasons.

Switching from Robot Control V1 to Robot Control V2, refer to [Upgrading from Robot Control V1 to Robot Control V2 on page 445](#).

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.

Functional description

Coordinate system	Description
Axis coordinate system (ACS) on page 25	Axis reference
Base coordinate system (BCS) on page 26	Kinematics reference
Machine coordinate system (MCS) on page 26	Reference to the machine system
Product coordinate system (PCS) on page 27	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
- **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

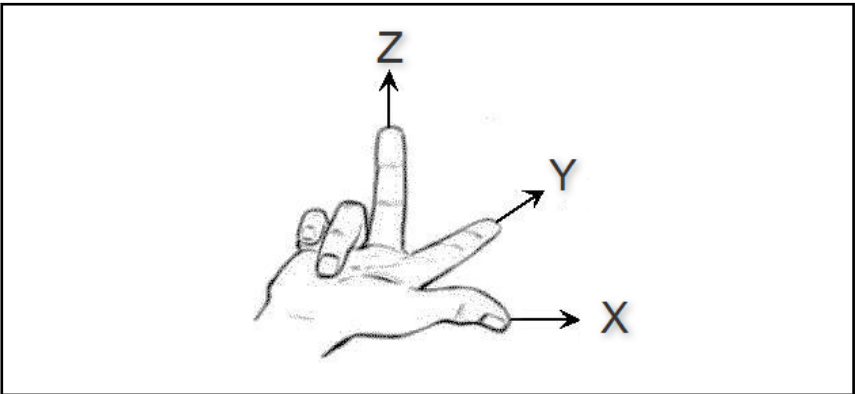


Fig. 2-1: Right-handed coordinate system

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

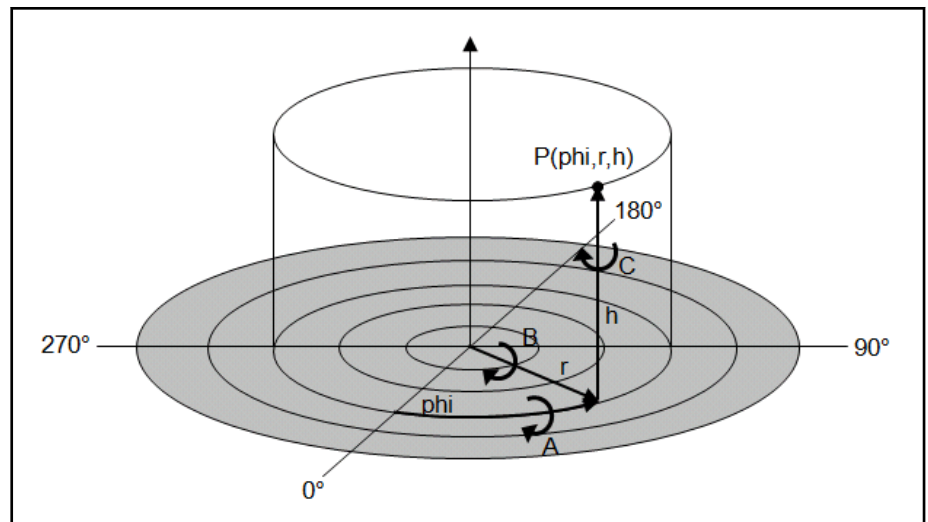
A indicates the rotation around the x-axis. B indicates the rotation of the y-axis (resulting from the rotation around the x-axis) and C indicates the rotation of the z-axis (resulting from the rotation around the x-axis and y-axis).

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).

Functional description



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

Therefore, the position of an object is specified by the 6 tuple (phi, 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).

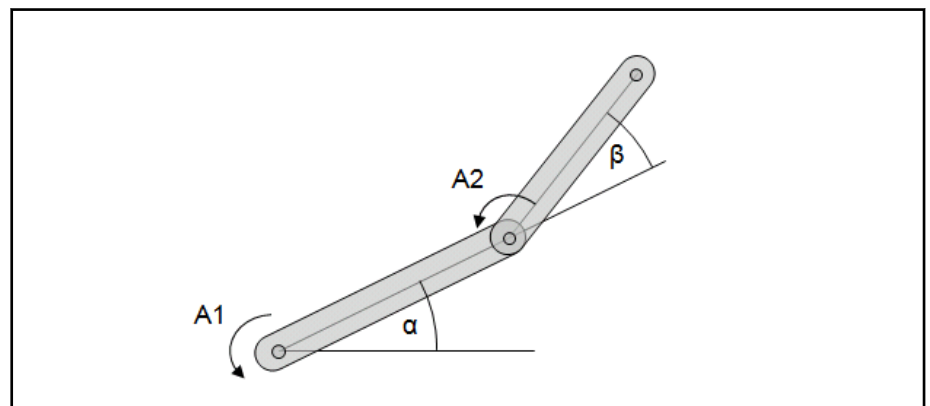


Fig. 2-3: Axis-related coordinate system of a 2-axis kinematics

Axis coordinate system (ACS)

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

Functional description

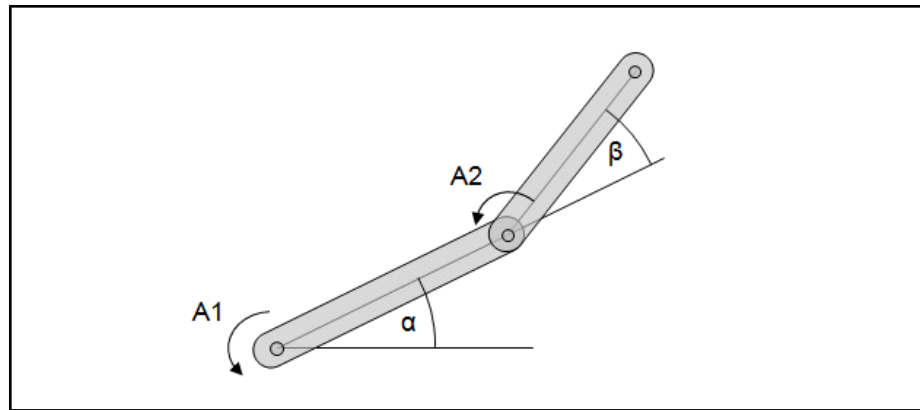


Fig. 2-4: ACS

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.

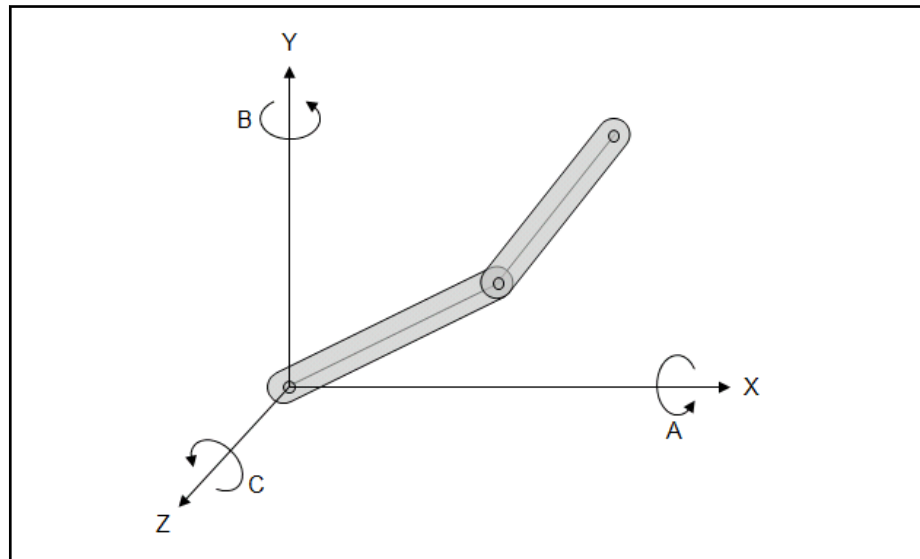


Fig. 2-5: BCS

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.

Functional description

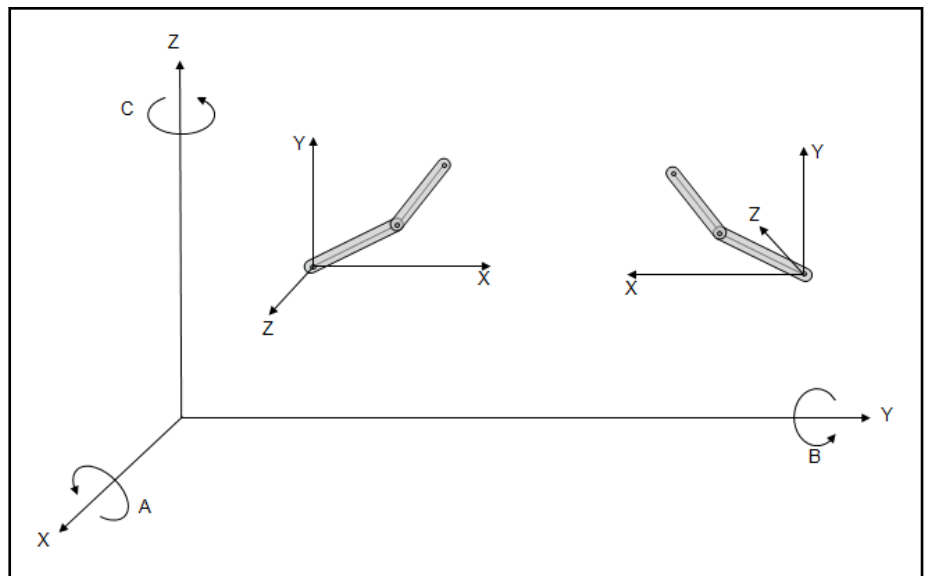


Fig. 2-6: MCS

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 65](#)).

16 PCS may be specified per kinematics.

The PCS is either a Cartesian or cylindrical coordinate system.

The use of product coordinate systems in connection with belt synchronization is described in [Belt synchronization on page 392](#).

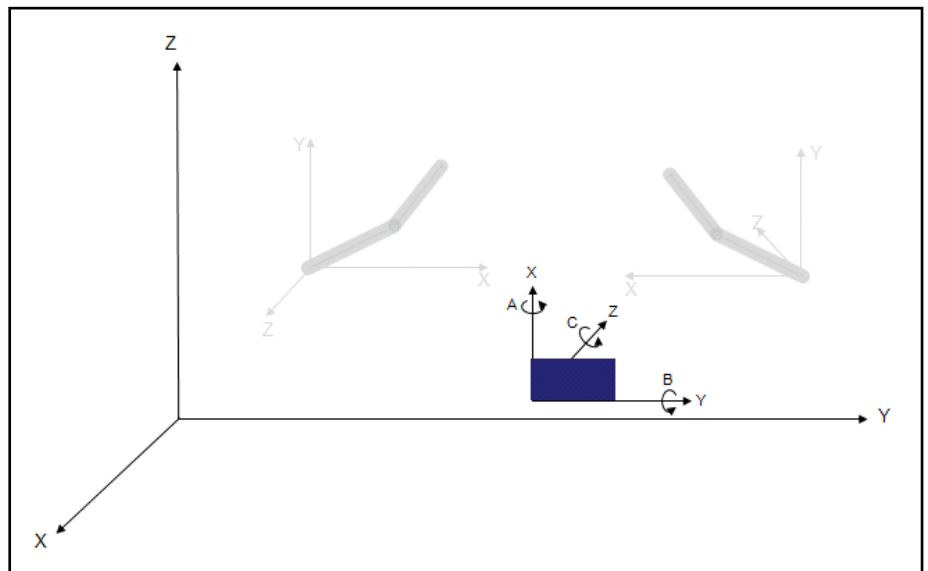


Fig. 2-7: PCS (Cartesian)

2.2.2 Transformations

Transformations, overview

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

Functional description

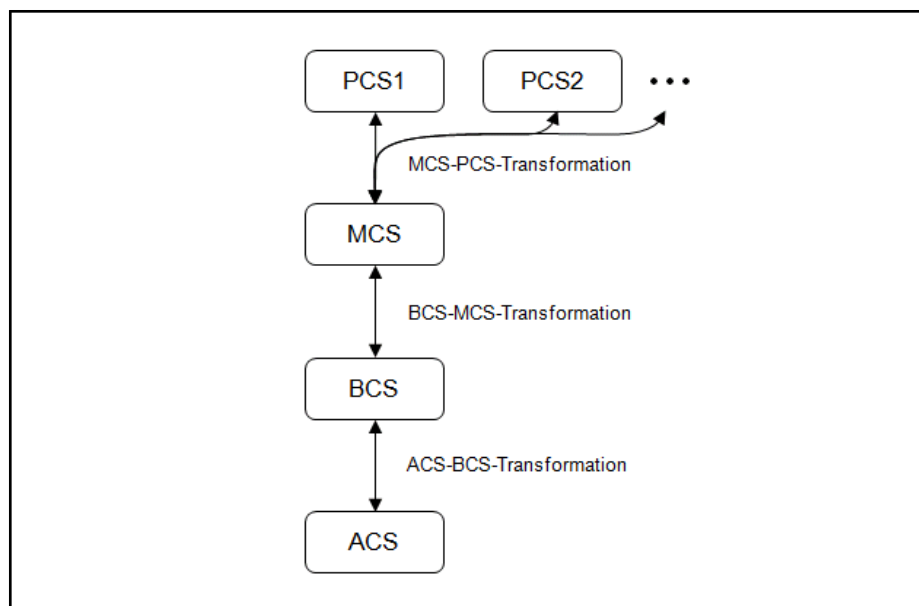


Fig. 2-8: Overview of transformations

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).

For a list of available transformation schemes, see: [Transformation schemes on page 107](#).

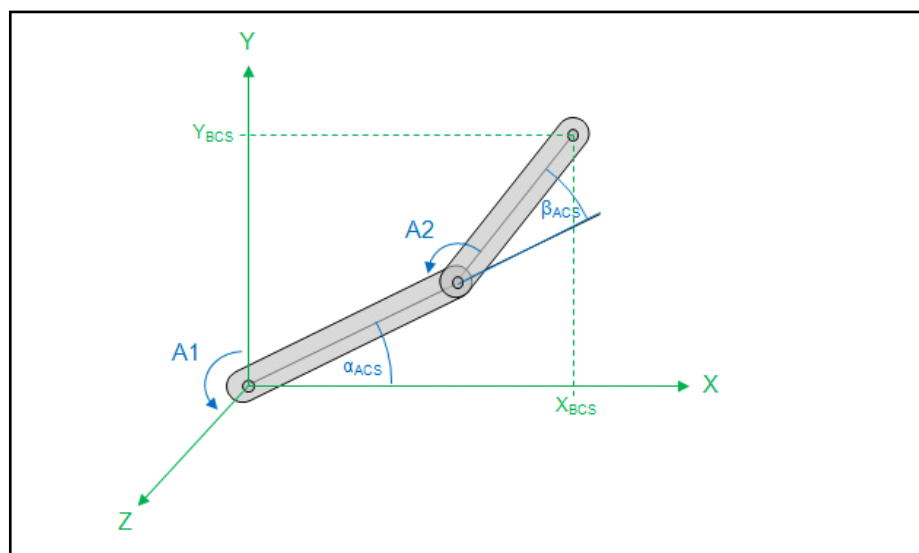


Fig. 2-9: ACS/BCS

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 BCS system).

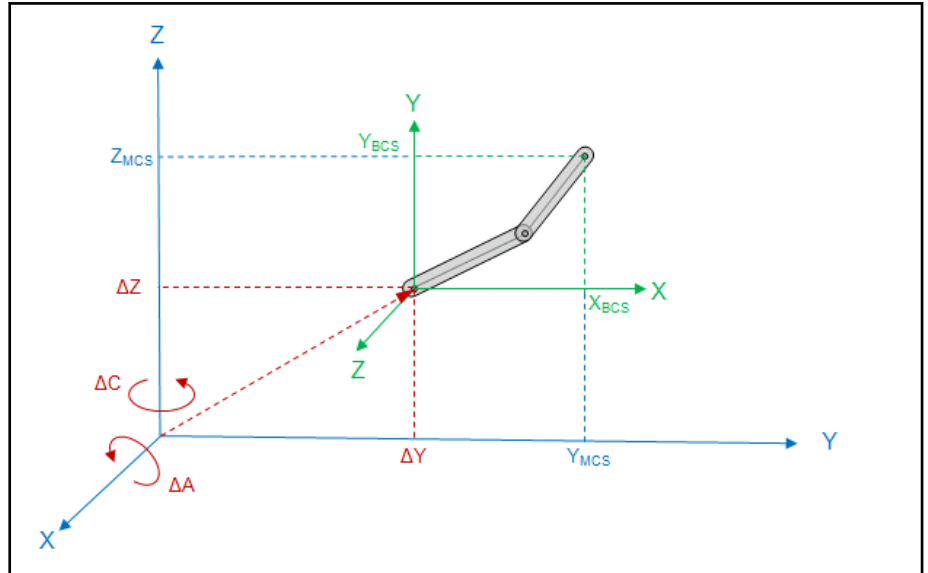


Fig. 2-10: BCS/MCS

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 using the command [Get Trafo Values on page 38](#).

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 37](#), with a Cartesian PCS
- [Set cylindric transform on page 37](#), with a cylindrical PCS

Functional description

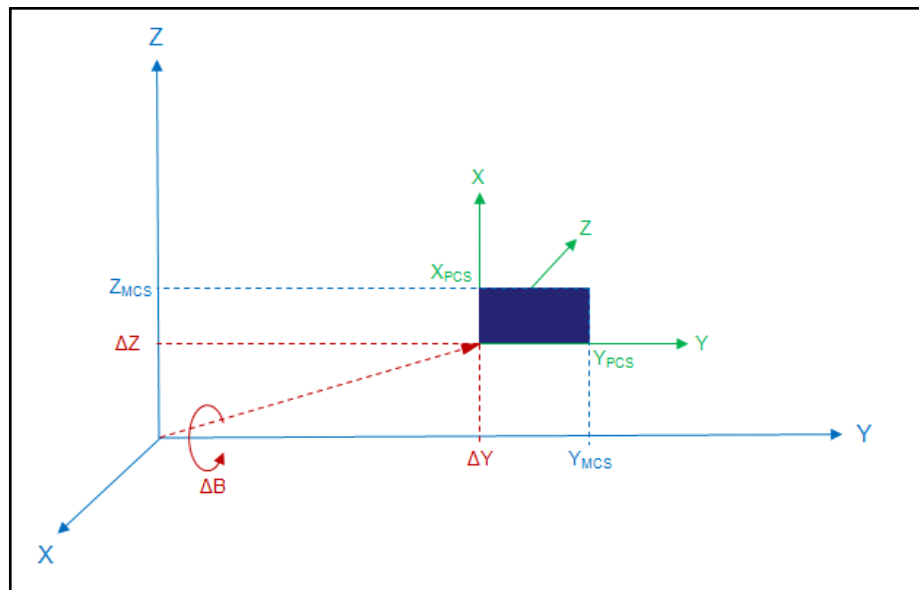


Fig. 2-11: MCS/PCS

Example:

Set Cartesian Transform = $(\Delta X, \Delta Y, \Delta Z, \Delta A, \Delta B, \Delta C) = (0, 111, 499, 0^\circ, -90^\circ, 0^\circ)$

MCS: $(X_{MCS}, Y_{MCS}, Z_{MCS}, A_{MCS}, B_{MCS}, C_{MCS}) = (0, 211, 699, 0^\circ, 0^\circ, 0^\circ)$

PCS: $(X_{PCS}, Y_{PCS}, Z_{PCS}, A_{PCS}, B_{PCS}, C_{PCS}) = (200, 100, 0, 0^\circ, 90^\circ, 0^\circ)$



Rotation and displacement between MCS and PCS can be read using the command [Get Trafo Values](#) on page 38.

2.3 Axis types

2.3.1 Axis types, overview

There are the following axis types in the MLC Robot Control:

- Positioning axes
- Orientation axes
- Auxiliary axes
- Belt axes

2.3.2 Positioning axes

Positioning axes determine the position in space. They determine the coordinates X, Y and Z.

They significantly determine the path interpolation in space.

Up to three positioning axes can be created per kinematics.

2.3.3 Orientation axes

Orientation axes determine the orientation in space. They determine the twists A, B and C.

They significantly determine the orientation interpolation in space.

Up to three orientation axes can be created per kinematics.

2.3.4 Auxiliary axes

There are two types of auxiliary axes:

- being part of the transformation. A change in position of this axis can affect the position and/or the orientation in space.
- not being part of the transformation. A change in position of this axis does not affect the position and/or the orientation in space.

Both types are moved fully synchronously to the spatial motion (also refer to [Motion types on page 39](#)).

Up to ten auxiliary can be created per kinematics.



Please refer to the [Transformation scheme on page 107](#) for whether an auxiliary axis is used.

2.3.5 Belt axes

A belt of a kinematics can be assigned via a belt axis.

The function "[Belt-synchronous motion on page 65](#)" can be used to synchronize a kinematics to a belt.

Up to 16 belt axes can be created per kinematics.

2.4 Point

2.4.1 Point, overview

One point provides all information on a position of a kinematics. The coordinates of a point have the sequence: Positioning axes, orientation axes, auxiliary axes and belt axes. Thus, the point has the maximum structure (X, Y, Z, A, B, C, H1, ..., H10, B1, ..., B16).

There are the following variants to represent a point:

- Representing a point
The point representation depends on the kinematics used. Only the axes used in a kinematics are represented in the point. Coordinates of unused axes are not represented.
- Connecting all positioning axes
All positions (X, Y, Z) are always represented irrespective of the kinematics used.
- Connecting all orientation axes
All orientations (A, B, C) are represented irrespective of the kinematics used.



Connecting der positioning axes und orientation axes does not affect the representation of the points in the ACS.

Unused auxiliary and belt axes are not represented in the point.

Example:

Representing a point for a Scara kinematics

A Scara kinematics consists of one belt B1, the two position axes X and Y and one auxiliary axis H1. A point for this kinematics consists of four coordinates: (X, Y, H1, B1). This composition of a point applies to all coordinate systems. The Z-coordinate and the orientations A, B, C which are

Functional description

not included in the transformation are not represented and do not have to be programmed.

2.4.2 Configuring the point representation

The following options can be used for the point representation:

- [K-0-0099](#) Bit 7: Connecting all positioning axes
- [K-0-0099](#) Bit 8: Connecting all orientation axes

Due to these options, the coordinates move in the point.



DANGER

Using these options, the coordinates move in the point

It is required to check and adjust all points in the user program.

Example:

Coordinate position for different point representations

	Point	Connecting Positioning axes	Connecting Orientation axes	Connecting Positioning axes and Orientation axes
Point representation	(X, Z, B, H1, B1)	(X, Y, Z, B, H1, B1)	(X, Z, A, B, C, H1, B1)	(X, Y, Z, A, B, C, H1, B1)
Coordinate position for H1	4	5	6	7

Dependent parameters

The configuration of the point representation affects the following parameters:

- [K-0-0100](#), [K-0-0150](#) (position data):
The kinematic positions are extended by the additional coordinates.
- [K-0-0050](#), [K-0-0051](#), [K-0-0052](#), [K-0-0053](#) (units):
Units are added for the newly added coordinates in the point. The units are supplemented as per the additional axis type.
- [K-0-0007](#) (Number of axes):
The number of axes increases by the additional coordinates in the point.

For a use case, refer to [Extend a point on page 402](#).

Example:

Illustration in the MCS

Transformation scheme	Point	Connecting Positioning axes	Connecting Orientation axes	Connecting Positioning axes and Orientation axes
FlexStamper (39) with one belt	(X, Y, H1, B1)	(X, Y, Z, H1, B1)	(X, Y, A, B, C, H1, B1)	(X, Y, Z, A, B, C, H1, B1)

4-axis Scara (10) with one auxiliary axis	(X, Y, Z, H1)	(X, Y, Z, H1)	(X, Y, Z, A, B, C, H1)	(X, Y, Z, A, B, C, H1)
3-axis Canfield (81)	(Z, B, C, H1)	(X, Y, Z, B, C, H1)	(Z, A, B, C, H1)	(X, Y, Z, A, B, C, H1)

2.5 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.

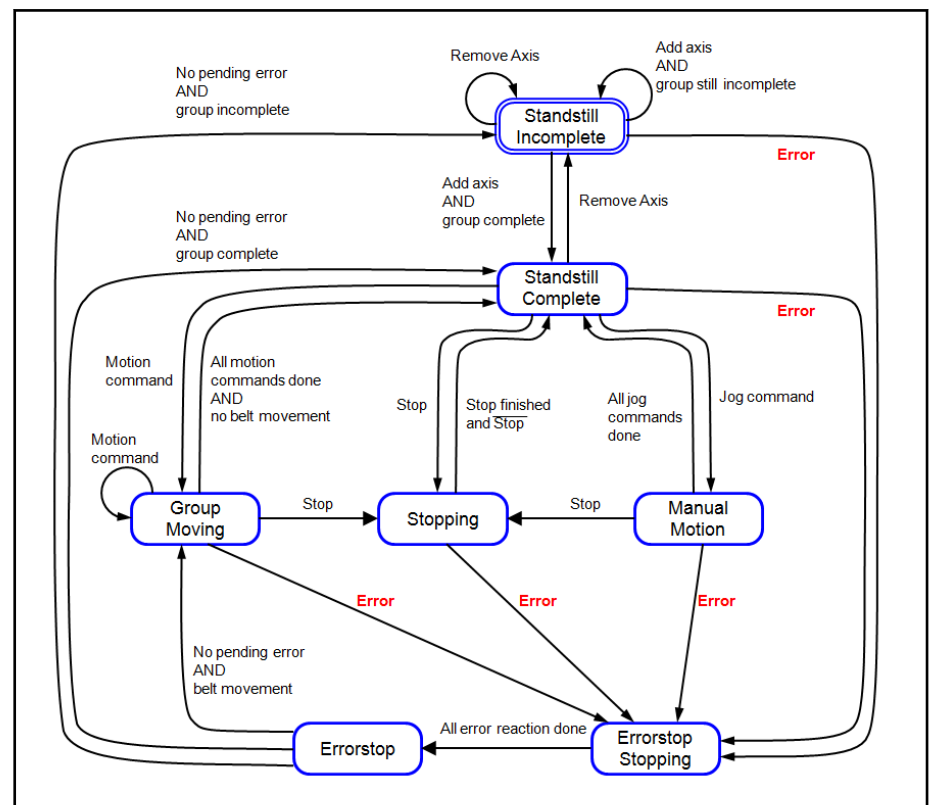


Fig. 2-12: State diagram of a kinematics

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 37](#) and [Group All Axes on page 37](#) to add axes to the kinematics. If all configured axes are added to the kinematics, the state switches to "Standstill Complete" active.

- Standstill Complete**

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

Functional description

Use the commands [Remove Axis on page 37](#) and [Remove All Axes on page 37](#) to remove axes from the kinematics. If an axis is removed from the kinematics, the state switches to "Standstill Incomplete" active.

- **Group Moving**

In state "Group Moving", at least one motion command (see [List of all motion commands and functions on page 36](#)) or belt-synchronous motion (see [Belt-synchronous motion on page 65](#)) is active.

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 36](#)).

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

- **Manual Motion**

In state "Manual Motion" at least one manual mode command (jog command) ("[Jog Step on page 36](#)" or "[Jog Cont on page 36](#)") is active.

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

- **Errorstop Stopping**

In "Errorstop Stopping" state, the error reactions of the kinematics (see [Error reaction of the commanded motion on page 79](#)) and the belts are executed (see [Belt-specific error reaction on page 79](#)).

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

When all error reactions are completed, the state switches to "Errorstop" active.

- **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" active depending on the conditions met.

2.6 Robot Control, commands and functions

2.6.1 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. A command is prepared first. This preparation can take several motion cycles depending on the system load. If a command is prepared, it can be executed.

Functional description

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 were ever in the command buffer.



For an application example for the effects of different command buffer settings, refer to [Determining the command buffer size, page 403](#).

In order to blend motions, the commands have to be buffered (see [Blending on page 52](#)).

The command state indicates the processing step of the command (see [Command state, page 36](#)).

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 36	Motion of a kinematics in space
List of all configuration commands on page 37	Configuration of a kinematics
List of all information functions on page 38	Information on a kinematics
List of all teach-in commands and functions on page 38	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 the "Tool Center Point" (TCP) to move in the space.

The TCP is a designated kinematics point 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 on page 23](#)

The TCP position can be determined using the function [Read Pos on page 36](#). It is differentiated between two positions.

- **Prepare:** TCP position of the last prepared motion command
- **Execute:** Last position commanded by the robot

The prepare position jumps to the next target position after a motion command has been prepared. In contrast, the execution position is continuous.

A motion command consists of the following:

- **End point and/or intermediate points:** Points on the motion path

Functional description

- **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 on page 39](#)
- **Blending and/or buffer type:** Type to start and/or blend the block
Refer to [Blending on page 52](#)

2.6.2 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

2.6.3 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.

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

List of all motion commands and functions

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Move Linear Absolute	MC_MoveLinearAbsolute	mlpiRobotMoveLinearAbs	MoveLinearAbsolute
Move Linear Relative	MC_MoveLinearRelative	mlpiRobotMoveLinearRel	MoveLinearRelative
Move Circular Absolute	MC_MoveCircularAbsolute	mlpiRobotMoveCircularAbs	MoveCircularAbsolute
Move Circular Relative	MC_MoveCircularRelative	mlpiRobotMoveCircularRel	MoveCircularRelative
Move Direct Absolute	MC_MoveDirectAbsolute	mlpiRobotMoveDirectAbs	MoveDirectAbsolute
Move Direct Relative	MC_MoveDirectRelative	mlpiRobotMoveDirectRel	MoveDirectRelative

Functional description

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Move Jump Absolute	ML_RobotMoveJumpAbsolute	mlpiRobotMoveJumpAbs	MoveJumpAbsolute
Synchronization with Limits	ML_BeltSynchronizationWithLimits	mlpiRobotSyncOnWithLimits	SyncOnWithLimits
Desynchronization with Limits	ML_BeltDeSynchronizationWithLimits	mlpiRobotSyncOffWithLimits	SyncOffWithLimits
Jog Cont	ML_GroupJogCont	mlpiRobotJogCont	JogCont
Jog Step	ML_GroupJogStep	mlpiRobotJogStep	JogStep
Open Cyclic Position	ML_RobotOpenCyclicChannel	mlpiRobotOpenCyclicChannel	
Open Direct Cyclic Channel	ML_RobotOpenDirectCyclicChannel		
Write Cyclic Position (function)	ML_RobotWriteCyclicChannel	mlpiRobotWriteCyclicChannel	
Stop	MC_GroupStop	mlpiRobotStop	Stop
Interrupt	MC_GroupInterrupt	mlpiRobotInterrupt	Interrupt
Continue	MC_GroupContinue	mlpiRobotContinue	Continue
Wait	ML_GroupWait	mlpiGroupWait	Wait

Tab. 2-4: List of all motion commands and functions

List of all configuration commands

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Group Reset	MC_GroupReset	mlpiRobotReset	Reset
Group Axis	MC_AddAxisToGroup	mlpiMotionAddAxisToGroup	
Remove Axis	MC_RemoveAxisFromGroup	mlpiMotionRemAxisFromGroup	
Group All Axes	ML_GroupAllAxes	mlpiRobotAddAllAxisToGroup	AddAllAxisToGroup
Ungroup All Axes	MC_UngroupAllAxes	mlpiRobotRemAllAxisFromGroup	RemAllAxisFromGroup
Set Kinematic Transformation	MC_SetKinTransform		
Set Kinematic Transform Extended	MC_SetKinTransformExt		
Set Cartesian Transformation	MC_SetCartesianTransform	mlpiRobotSetCartesianTransform	SetCartesianTransform
Set Cylindric Transformation	ML_SetCylindricTransform	mlpiRobotSetCylindricTransform	SetCylindricTransform
Set Belt Configuration	ML_SetBeltConfiguration	mlpiRobotSetBeltConfiguration	SetBeltConfiguration
Set Belt Desync Relative Configuration	ML_SetBeltDesyncRelConfiguration	mlpiRobotSetBeltDesyncRelConfiguration	SetBeltDesyncRelConfiguration
Set Rotary Table Configuration	ML_SetRotaryTableConfiguration	mlpiRobotSetRotaryTableConfiguration	SetRotaryTableConfiguration

Functional description

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Set Belt Error Reaction	ML_SetBeltErrorReaction	mlpiRobotSetBeltErrorReaction	SetBeltErrorReaction
Set Safe Zone	ML_GroupSetSafeZone	mlpiRobotSetSafeZone	SetSafeZone
Set Move ID	ML_RobotSetMoveID	mlpiRobotSetMoveId	SetMoveId

Tab. 2-5: List of all configuration commands

List of all information functions

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Read Pos	MC_GroupReadActualPosition	mlpiReadPos	ReadPos
Switch On Path	ML_SwitchOnPath		
Read Cyclic Position	ML_RobotReadCyclicChannel	mlpiRobotReadCyclicChannel	
Move Point	ML_RobotMovePoint	mlpiRobotMovePoint	MovePoint
Transform Point	ML_RobotTransformPoint	mlpiTransformPoint	TransformPoint
Set Performance Enable	ML_RobotSetPerformanceEnable	mlpiRobotSetPerformanceEnable	SetPerformanceEnable
Get Performance	ML_RobotGetPerformance	mlpiRobotGetPerformanceResult	GetPerformanceResult
Get Mechanic	ML_RobotGetMechanic	mlpiRobotGetMechanicalData	GetMechanicData
Get Trafo Values	ML_RobotGetTrafoValues	mlpiRobotGetTrafoValues	GetTrafoValues
Get Belt Configuration	ML_RobotGetBeltConfiguration	mlpiRobotGetBeltConfiguration	GetBeltConfiguration
Get Belt Desync Relative Configuration	ML_RobotGetBeltDesyncRelConfiguration	mlpiRobotGetBeltDesyncRelConfiguration	GetBeltDesyncRelConfiguration
Get Belt Error Reaction	ML_RobotGetBeltErrorreaction	mlpiRobotGetBeltErrorReaction	GetBeltErrorReaction
Get Safe Zone	ML_RobotGetSafeZones	mlpiRobotGetSafeZones	GetSafeZones
Get Move ID	ML_RobotGetMoveID	mlpiRobotGetMoveId	GetMoveId

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
Write point	ML_GroupWritePoint	mlpiRobotTeachInWritePoint	
Read Point	ML_GroupReadPoint	mlpiRobotTeachInReadPoint	
Read Next Point	ML_GroupReadNextPoint	mlpiRobotTeachInReadNextPoint	
Delete Points	ML_GroupDeletePoints	mlpiRobotTeachInDeletePoints	

Functional description

Reference description	PLC	MLPI core	MLPI COM MLPI Java
Save Point File	ML_GroupSavePointFile	mlpiRobotTeachInSavePointFile	
Load Point File	ML_GroupLoadPointFile	mlpiRobotTeachInLoadPointFile	
Delete Point File	ML_GroupDeletePointFile	mlpiRobotTeachInDeletePointFile	

Tab. 2-7: List of all teach-in commands and functions

2.7 Motion types

2.7.1 Motion types, overview

The MLC Robot Control supports a kinematic method in four motion types:

- Linear (straight line in the space), [“Linear” motion type on page 39](#)
- Circular (circle segment in the space), [“Circular” motion type on page 40](#)
- Point to point (PTP), [Point-to-point motion type on page 41](#)
- Jump, [Motion type Jump on page 42](#)
- Jump, [Motion type HalfJump on page 44](#)



With interpolation, orientation motions are controlled fully synchronously.

Furthermore, it is distinguished between [“Absolute positioning on page 47”](#) and [“Relative positioning on page 47”](#).

Motion types with absolute and relative positioning

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

Tab. 2-8: Overview on motion types

All motion types are defined by motion limit values. These include:

- Maximum velocity - V
- Maximum acceleration/deceleration -A
- Maximum jerk- J

2.7.2 “Linear” motion type

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

The straight line is geometrically defined by two points. Since the current kinematic position P1 is known to the control, it is sufficient to specify a target point P2.

Functional description

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

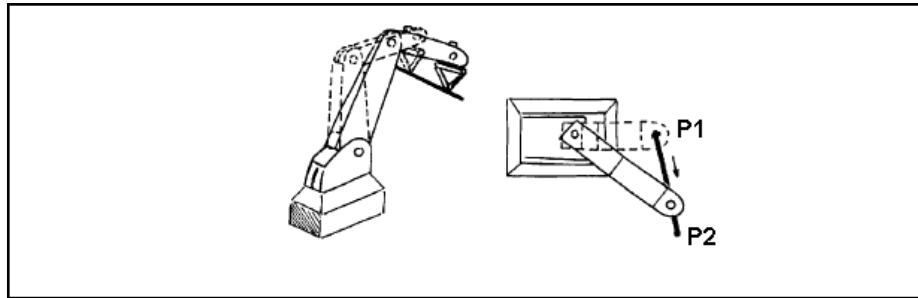


Fig. 2-13: "Linear" interpolation method

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

- [Move Linear Absolute on page 36](#)
- [Move Linear Relative on page 36](#)

Auxiliary axes

Auxiliary axes are moved fully synchronously to the spatial motion. This means, that after x % of the distance in space, x% of the auxiliary axis motion is also reached. Therefore, acceleration, traveling with a constant velocity and deceleration on the spatial path are time-synchronous to the auxiliary axis motion.

Furthermore, the motion limit values are limited to the path and thus ensure that the limit values of the auxiliary axes are not exceeded. Thus, it is possible that the programmed motion limit values of the motion cannot be reached.

The limit values of the auxiliary axes are determined from the minimum of the single axis limit values and the limits of the auxiliary axes:

- $\text{LimitValue\_V} = \text{Minimum}(\text{SingleAxisLimitValue\_V (A-0-0032, A-0-0033, A-0-0035)}, \text{Limit\_V (K-0-0028)})$
- $\text{LimitValue\_A} = \text{Minimum}(\text{SingleAxisLimitValue\_A (A-0-0034)}, \text{Limit\_A (K-0-0029)})$
- $\text{LimitValue\_J} = \text{Minimum}(\text{SingleAxisLimitValue\_J (A-0-0036)}, \text{Limit\_J (K-0-0030)})$

If only one auxiliary motion is commanded without spatial motion, this motion is performed using the limit values of the auxiliary axis.

2.7.3 "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 approached. Therefore, it is sufficient to specify the intermediate point PZ and the target point P2.

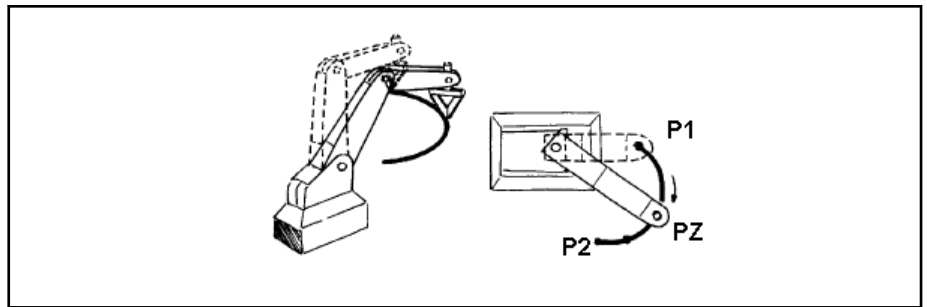


Fig. 2-14: "Circular" interpolation type

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

- [Move Circular Absolute on page 36](#)
- [Move Circular Relative on page 36](#)

Auxiliary axes

Auxiliary axes are moved fully synchronously to the spatial motion. This means, that after x % of the distance in space, x% of the auxiliary axis motion is also reached. Therefore, acceleration, traveling with a constant velocity and deceleration on the spatial path are time-synchronous to the auxiliary axis motion.

Furthermore, the motion limit values are limited to the path and thus ensure that the limit values of the auxiliary axes are not exceeded. Thus, it is possible that the programmed motion limit values of the motion cannot be reached.

The limit values of the auxiliary axes are determined from the minimum of the single axis limit values and the limits of the auxiliary axes:

- $\text{LimitValue\_V} = \text{Minimum}(\text{SingleAxisLimitValue\_V (A-0-0032, A-0-0033, A-0-0035)}, \text{Limit\_V (K-0-0028)})$
- $\text{LimitValue\_A} = \text{Minimum}(\text{SingleAxisLimitValue\_A (A-0-0034)}, \text{Limit\_A (K-0-0029)})$
- $\text{LimitValue\_J} = \text{Minimum}(\text{SingleAxisLimitValue\_J (A-0-0036)}, \text{Limit\_J (K-0-0030)})$

If only an auxiliary motion is commanded without spatial motion, this motion is performed using the limit values of the auxiliary axis.

Degenerate circles

Degenerated circles are created if:

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

In these cases, the kinematics travels on a straight line between 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 point are located on a straight line and the intermediate point is behind the end point.

In this case, the intermediate point is not reached.

2.7.4 Point-to-point motion type

With this motion type, the commanded position is approached the fastest possible.

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

Functional description

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 space and the path velocity are irrelevant. It is guaranteed that the motions of all single axes end 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 called fully synchronous PTP.

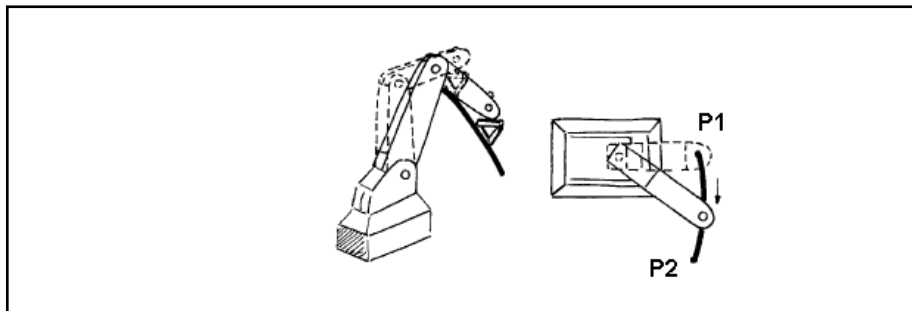


Fig. 2-15: "Direct" interpolation type

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

- [Move Direct Absolute on page 36](#)
- [Move Direct Relative on page 36](#)

2.7.5 Motion type Jump

The motion type approaches the target point by a vertical motion in z-direction, an arched motion via the end point P2 and a vertical motion in z-direction. The motion is executed with an optimized velocity profile with multiple acceleration or deceleration ramps.

The motion is geometrically defined by the starting point, target point, starting height, end height and maximum height. The control knows point P1, as it is the last commanded position. Therefore, it is sufficient to specify the end point P2 as well as the starting height, end height and maximum height.

The starting height defines the minimum height before the start of the horizontal motion. The end height defines the minimum height at the end of the horizontal motion. The maximum height defines the maximum height that can be reached during the entire motion sequence.

Starting and end height are specified relatively to the starting or end point in z-direction. The maximum height has to be specified relatively to the lower z-value of both points (P1 and P2) in z-direction.



The motion type can traverse into positive and into negative z-direction. Thus, the "MoveJump" function can also be commanded in a PCS twisted to the MCS. For the negative case, the maximum height has to be specified relatively to the higher z-value of both points (P1 and P2) in z-direction.

To avoid collisions with packaging sleeves or similar material, a vertical motion is executed before reaching the starting height or the end height (after a horizontal motion). For belt-synchronous motion sequences, the synchronous operation is not changed during a vertical motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion.

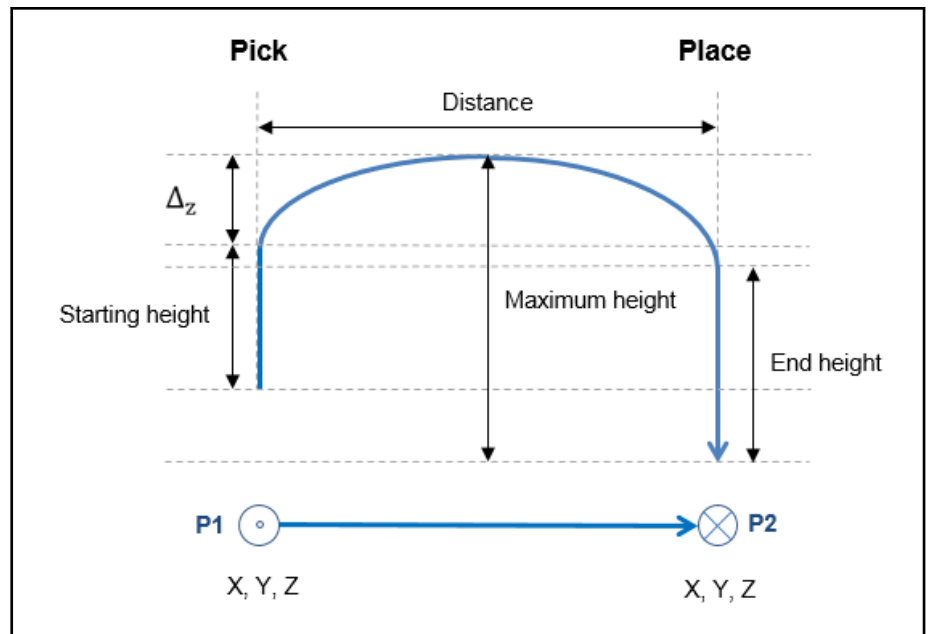


Fig. 2-16: Motion sequence and required parameters

Restrictions:

- For “Jump” in positive z-direction: Starting, maximum and end height must not assume any negative values
- For “Jump” in negative z-direction: Starting, maximum and end height must not assume any positive values
- The maximum height has to be absolutely higher than the starting and end height, as Δ_z has to be greater than zero
- The ratio of Δ_z to the distance must not exceed 1:50, as otherwise the curve in space cannot be calculated
- Starting point and end point must not be the same point



The vertical motion at the start and end of the motion is always executed in z-direction. Thus, starting, maximum and end height are also always interpreted in z-direction and have to be specified relatively to the starting or end point.

“MoveJump” over obstacle

Static obstacles between the pick-and-place belt can be automatically overcome using the “MoveJump” function. A linear horizontal segment is inserted at the highest position of the motion. This segment does not have to be positioned in the center of the motion. The length is identical to the width of the obstacle. An obstacle nearly as high as the maximum height can be jumped over without collision.



If no narrow safe zone is within the center of the motion, the linear segment is widened up to the center of the motion.

Obstacles are set up with safe zones and defined with the command Set Safe Zone. No entries at the interface are required for the “MoveJump” function. As soon as the safe zones touch an imaginary line between the end of the starting height and the end height, an additional segment is inserted into “MoveJump”.

Functional description

The user has to ensure that the maximum height of the motion is above the safe zone. If the path violates the safe zone, an error message is output. Refer to the chapter [Traversing range limits on page 76](#).

Safe zones are programmed as static obstacles in space. For pick-and-place applications, the distances to the obstacles change dynamically. The "MoveJump" function responds automatically to these changed distances and adjusts the length and the position of the linear segment.

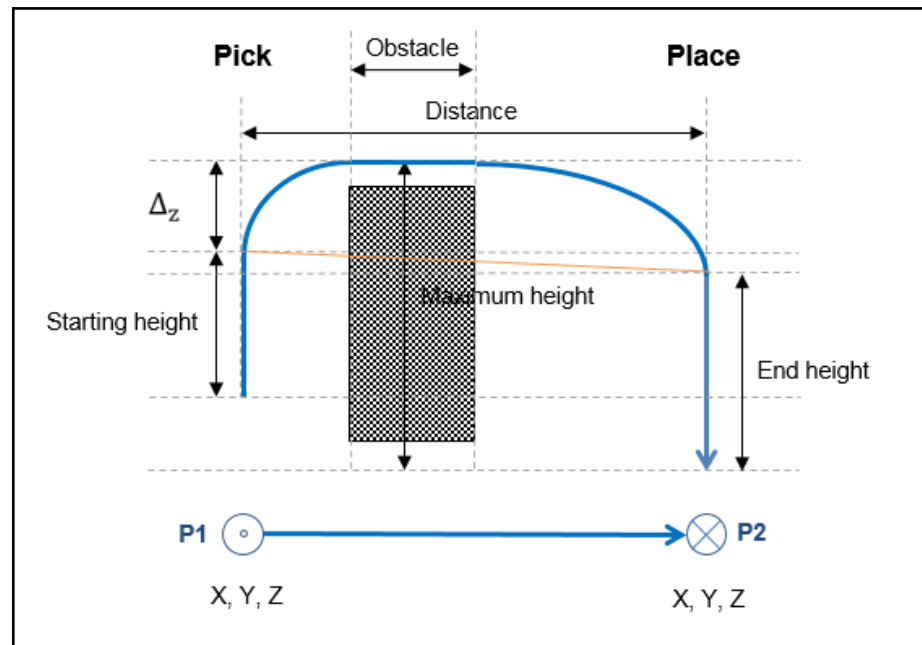


Fig. 2-17: Motion sequence over an obstacle

2.7.6 Motion type HalfJump

HalfJumpUp The motion type approaches the target point P2 by a vertical motion in z-direction and an arched motion.

The motion is geometrically defined by the starting point, target point and the starting height. The control knows point P1, as it is the last commanded position. Therefore, it is sufficient to specify the end point P2 and starting height.



As the same interface as for the MoveJump motion type is used, 0 has to be specified as maximum height for differentiation purposes. In this case, the end height is ignored.

The motion type can - like the "Jump" - traverse into positive and into negative z-direction.

The starting height defines the minimum height before the start of the horizontal motion.

The starting height is specified relatively to the starting point in z-direction.

To avoid collisions with packaging sleeves or similar material, a vertical motion is executed before reaching the starting height. For belt-synchronous motion sequences, the synchronous operation is not changed during a vertical motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion.

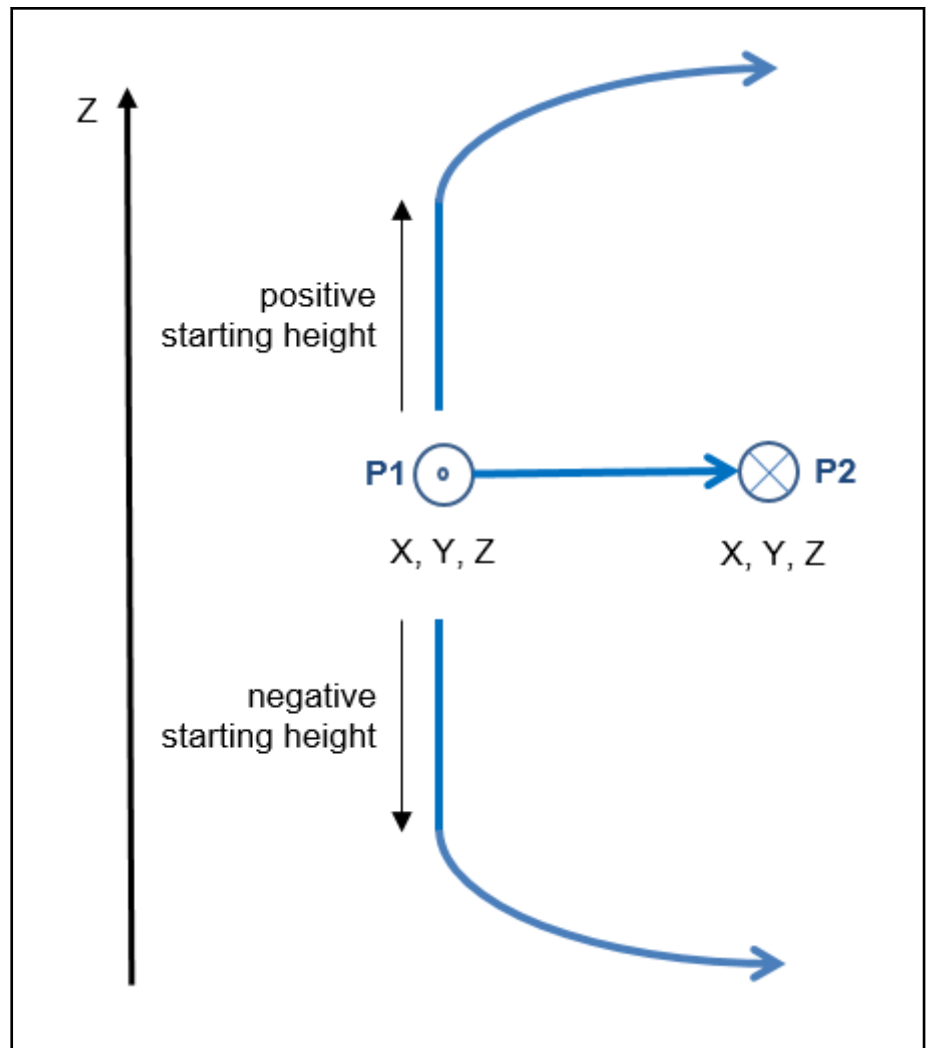


Fig. 2-18: Motion sequence and required parameters

Restrictions:

- For the positive HalfJumpUp: Starting height must not assume any negative values
- For the negative HalfJumpUp: Starting height must not assume any positive values
- Starting point and end point must not be the same point



The vertical motion at the start of the motion is always executed in z-direction. Thus, the starting height is always interpreted in z-direction and has to be specified relatively to the starting point.

HalfJumpDown

The motion type approaches the target point P2 by an arched motion and a vertical motion in z-direction.

The motion is geometrically defined by the starting point, target point and the end height. The control knows point P1, as it is the last commanded position. Therefore, it is sufficient to specify the end point P2 and the end height.



As the same interface as for the MoveJump motion type is used, 0 has to be specified as maximum height for differentiation purposes. In this case, the starting height is ignored.

Functional description

The motion type can - like the "Jump" - traverse into positive and into negative z-direction.

The end height defines the minimum height before the start of the horizontal motion.

The end height is specified relatively to the end point in z-direction.

To avoid collisions with packaging sleeves or similar material, a vertical motion is executed before reaching the end height. For belt-synchronous motion sequences, the synchronous operation is not changed during a vertical motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion. The resynchronization, synchronization and desynchronization takes place exclusively during the horizontal motion.

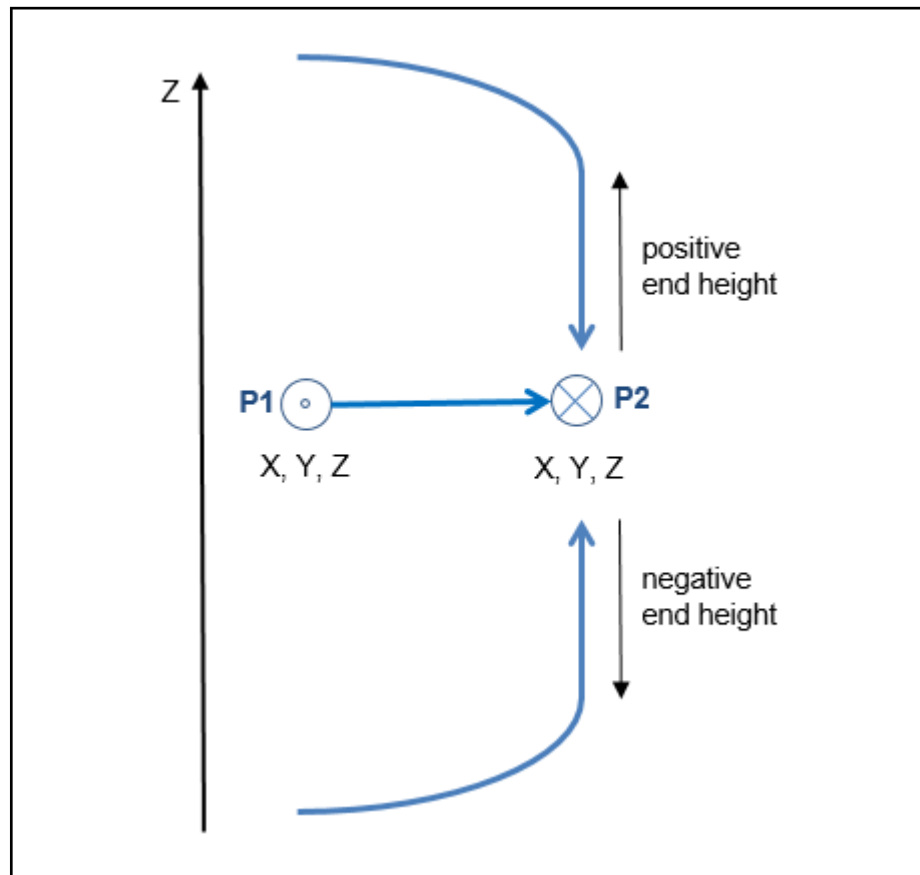


Fig. 2-19: Motion sequence and required parameters

Restrictions:

- For the positive HalfJumpDown: End height must not assume any negative values
- For the negative HalfJumpDown: End height must not assume any positive values
- Starting point and end point must not be the same point



The vertical motion at the start of the motion is always executed in z-direction. Thus, the end height is always interpreted in z-direction and has to be specified relatively to the end point.

2.7.7 Absolute positioning

Absolute positioning means specifying a position. The specified position is absolute and approached in space.

2.7.8 Relative positioning

Relative positioning means specifying a distance. The specified distance is traversed relatively to the last commanded position, i.e. the last commanded position added to the distance to be traversed results in the new absolute end position.

2.7.9 Commanding modulo-scaled axes

For modulo-scaled axes, the number of revolutions and the direction can be specified by commanding the target position for this axis. The relative distance for the motion is always the commanded value minus the current axis position with the modulo value. The following results:

- Commanding between 0 and the axis modulo value:
The axis approaches the target position, as if it were scaled as absolute
- Commanding smaller than 0:
The axis travels in a negative direction
- Commanding of larger module axis values:
The axis travels in a positive direction

Example:

Commanding of a modulo-scaled axis with a modulo value of 360°

Current modulo position	Commanded value	Relative axis motion	Modulo position after motion
100°	50°	-50°	50°
50°	350°	+300°	350°
350°	-20°	-370°	340°
340°	-370°	-710°	350°
350°	370°	+20°	10°
10°	370°	+360°	10°
10°	750°	+740°	30°

Tab. 2-9: Commanding of a modulo-scaled axis

2.8 Manual mode

2.8.1 Manual mode, general information

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

- [Single-step operation on page 48](#)
- [Continuous operation on page 49](#)

Manual mode supports the following motion types (see [Motion types on page 39](#)) using the specified coordinate system:

- PTP (ACS):

Functional description

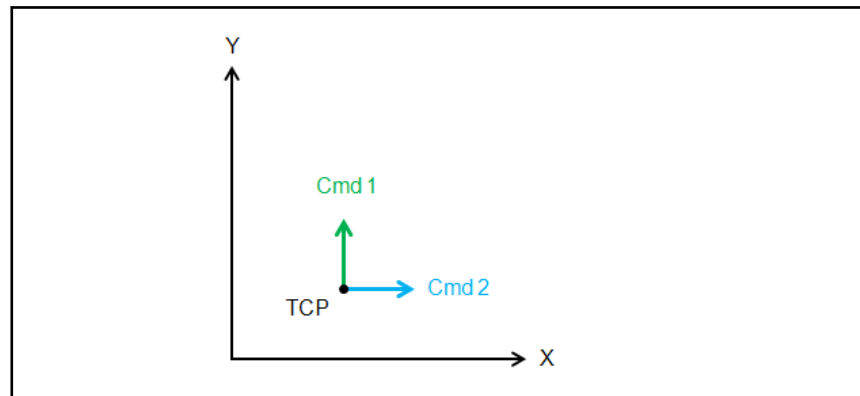
With the ACS coordinate system use the PTP motion type. Specified values in percent for velocity, acceleration, deceleration and jerk always refer to the axis with the largest "distance" or "direction".

- **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

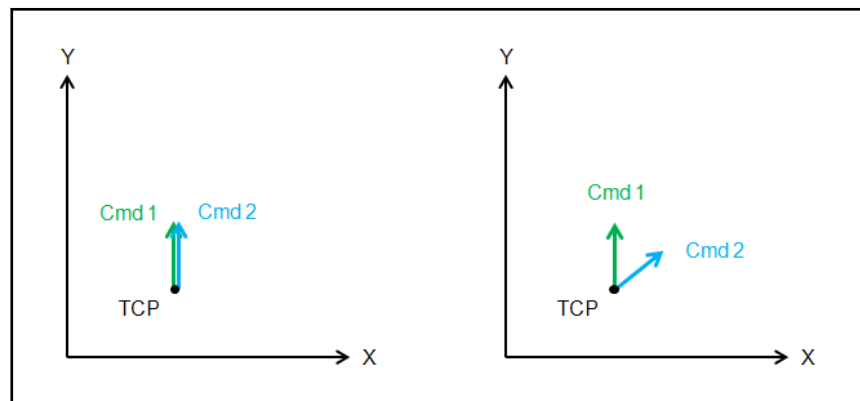


Cmd 1 Manual mode command in Y-direction

Cmd 2 Manual mode command in X-direction

Fig. 2-20: 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. 2-21: Several manual mode commands in the same or partially same direction

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

2.8.2 Single-step operation

To move a kinematics by a certain step width in manual mode, use the command [Jog Step on page 36](#).

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

2.8.3 Continuous operation

To move a kinematics continuously into a certain direction in manual mode, use the command [Jog Cont on page 36](#).

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.

2.9 Teach-In

2.9.1 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 (in the RAM):

- [Write point on page 50](#)
- [Read point on page 51](#)
- [Read next point on page 51](#)
- [Delete points on page 51](#)

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 points from a point file in a kinematics, use the following commands:

- [Save point file on page 51](#)
- [Load point file on page 52](#)

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

Use the following command to delete a point file:

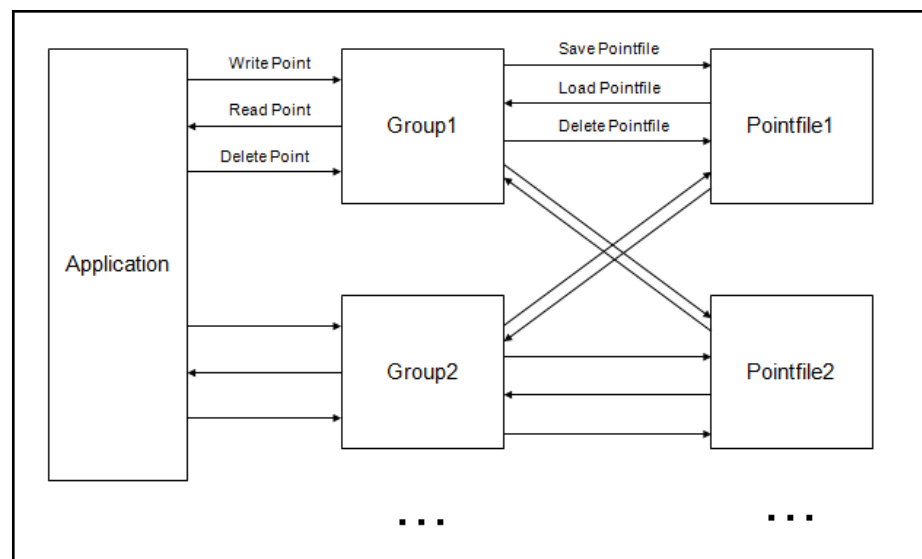
- [Delete point file on page 52](#)



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
Point file	Point file
Write point	Function "Write point"
Read point	Function "Read point" or "Read next point"
Delete point	Function "Delete points"
Save point file	Command "Save point file"
Load point file	Command "Load point file"

Fig. 2-22: Overview of the "Teach points" functionality

2.9.2 Structure of the point file

The point file is in ".txt" format.

One data set is saved per line.

One 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:

One data set from the point file Dummy.txt

```

Point1;100.000000000,20.000000000,0.000000000,0.000000000,0.00000000
00,0.000000000,0.000000000,0.000000000,0.000000000,0.000000000,0.00
000000,0.000000000,0.000000000,0.000000000,0.000000000,0.00000000
0,;0.000000000,0.000000000,0.000000000,0.000000000,0.000000000,0.000
00000,0.000000000,0.000000000,0.000000000,0.000000000,0.000000000,
0.000000000,0.000000000,0.000000000,0.000000000,0.000000000,;PCS1;
  
```

2.9.3 Write point

The function [Write Point on page 38](#) writes 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 in the kinematics is overwritten
- **Inactive:** The point 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"

2.9.4 Read point

The function [Read Point on page 38](#) reads a point from (the RAM of) a kinematics.

2.9.5 Read next point

The function [Read Point on page 38](#) reads the next point from (the RAM of) a kinematics.

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

If an empty string is specified as point name, the first point is read.

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 alphabetically. It is irrelevant in what order the points are written 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.

2.9.6 Delete points

The function [Delete Points on page 38](#) 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.

2.9.7 Save point file

Execute the command [Save Point File on page 38](#) to save points from a kinematics in a point file.

Functional description

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, i.e. the existing points maintain their values.
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.

2.9.8 Load point file

Execute the command [Load Point File on page 38](#) 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, i.e. the existing points maintain their values.
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.

2.9.9 Delete point file

The function [Delete Point File on page 38](#) deletes a point file of a kinematics.

2.10 Blending

2.10.1 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.

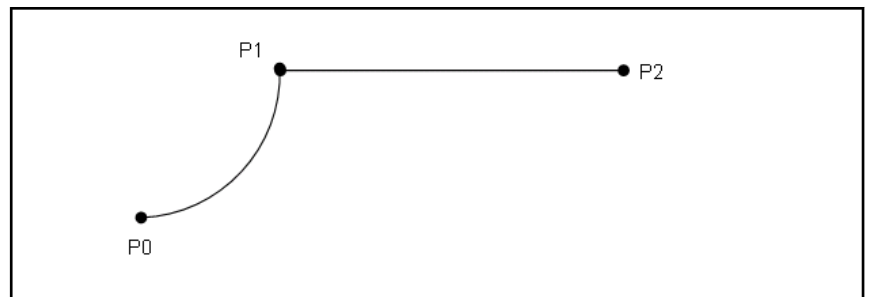


Fig. 2-23: TO

- **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.

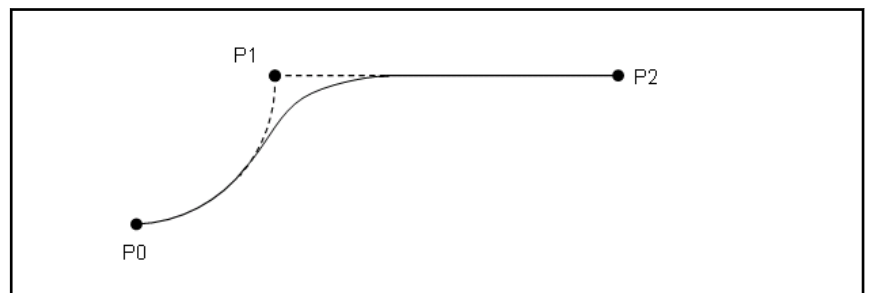


Fig. 2-24: VIA

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.

2.10.2 Blending radius

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

The programmed radius R_{Prog}

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.

2.10.3 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 depends 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. 2-10: Overview of slope types

2.10.4 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.

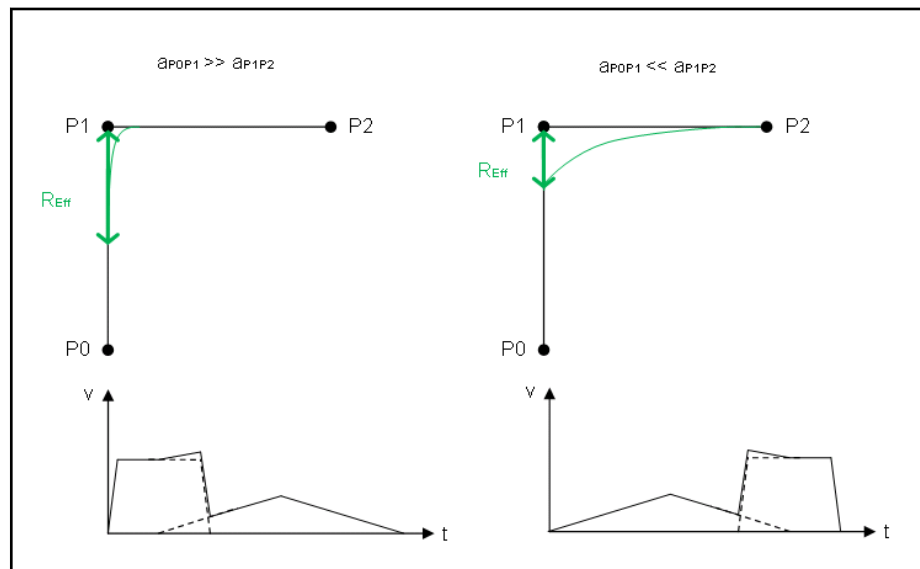


Fig. 2-25: Path depending on the maximum values of the velocity profile

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

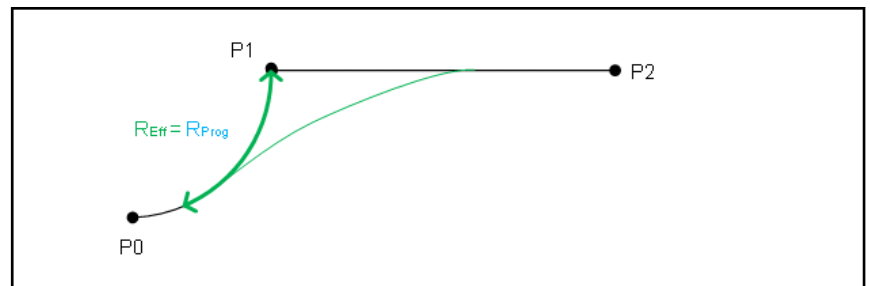


Fig. 2-26: Blending without limitation

• 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.

$$R_{Eff} = \min \left(R_{Prog}, \frac{1}{2} * \overline{P_0 P_1}, \frac{1}{2} * \overline{P_1 P_2} \right)$$

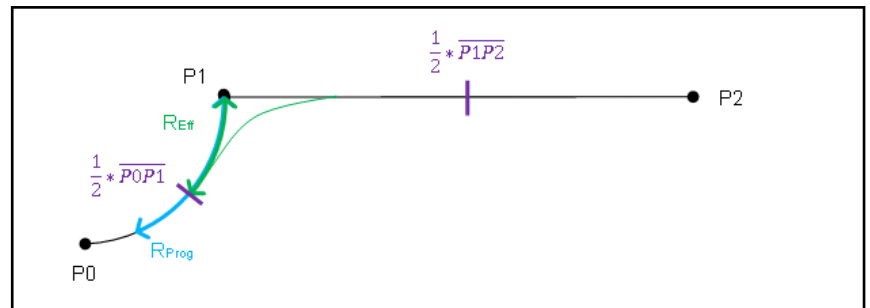
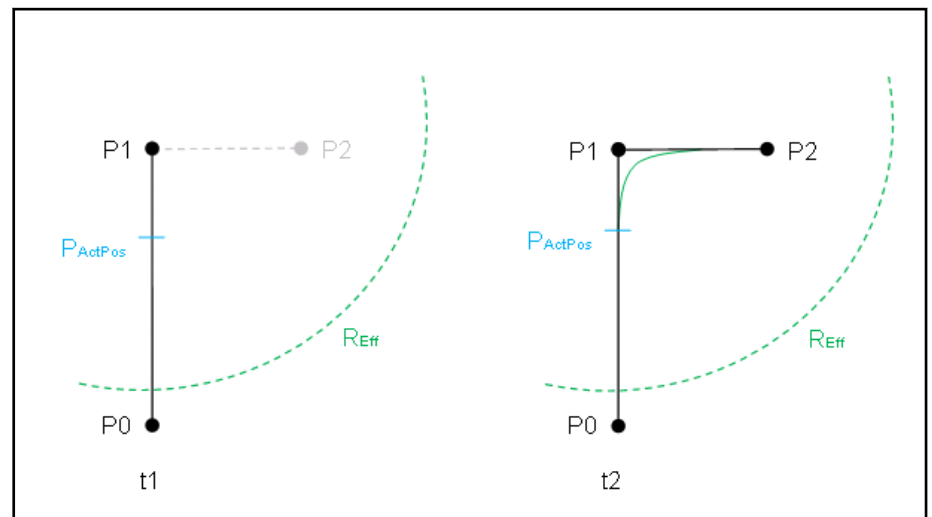


Fig. 2-27: Blending with limitation to half the path distance

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.

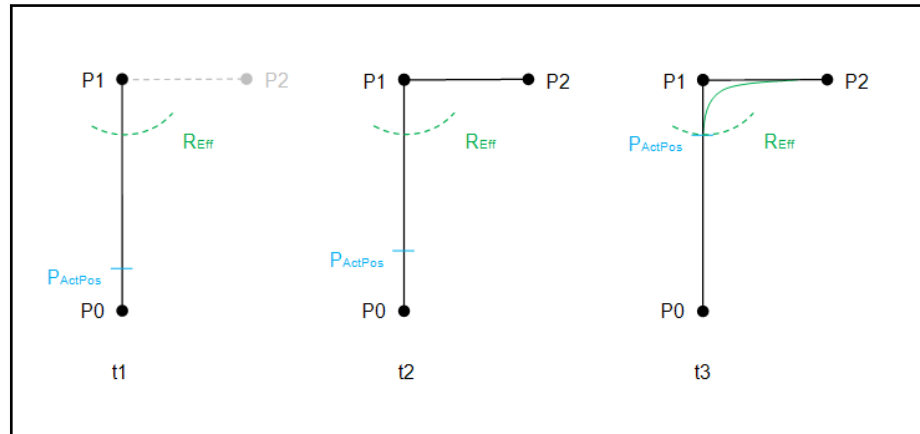


t1 At time t1, the motion command to P2 is issued
 t2 At time t2, the motion command to P2 is prepared

Fig. 2-28: 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. 2-29:

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

2.10.5 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.

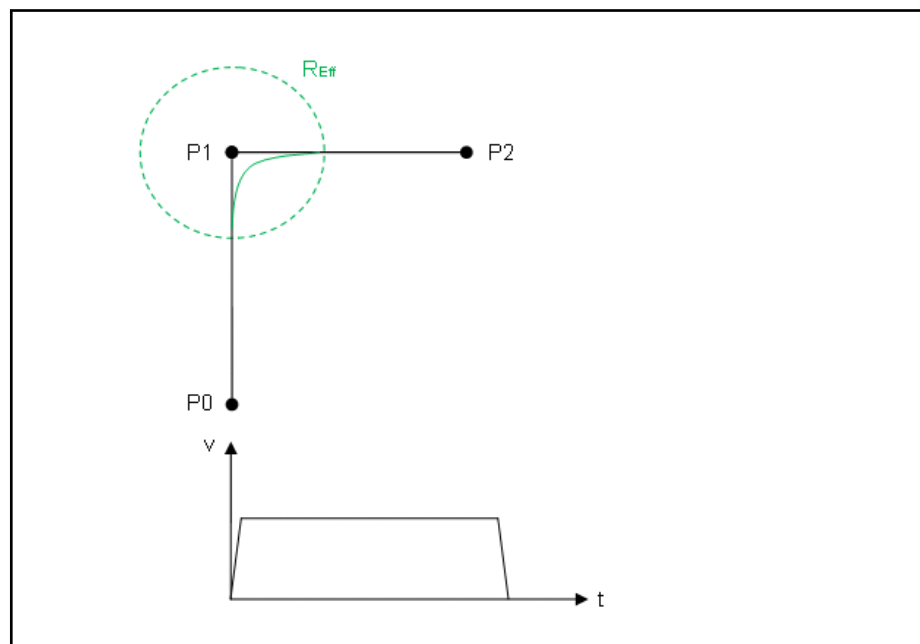


Fig. 2-30: Blending with Cont_Slope

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.

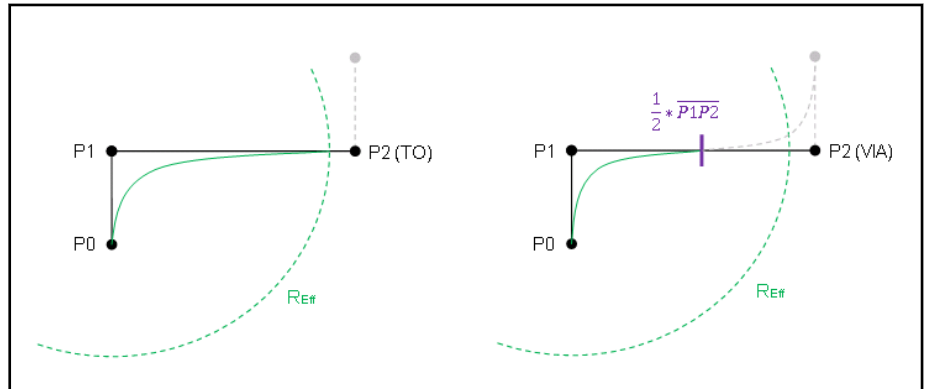
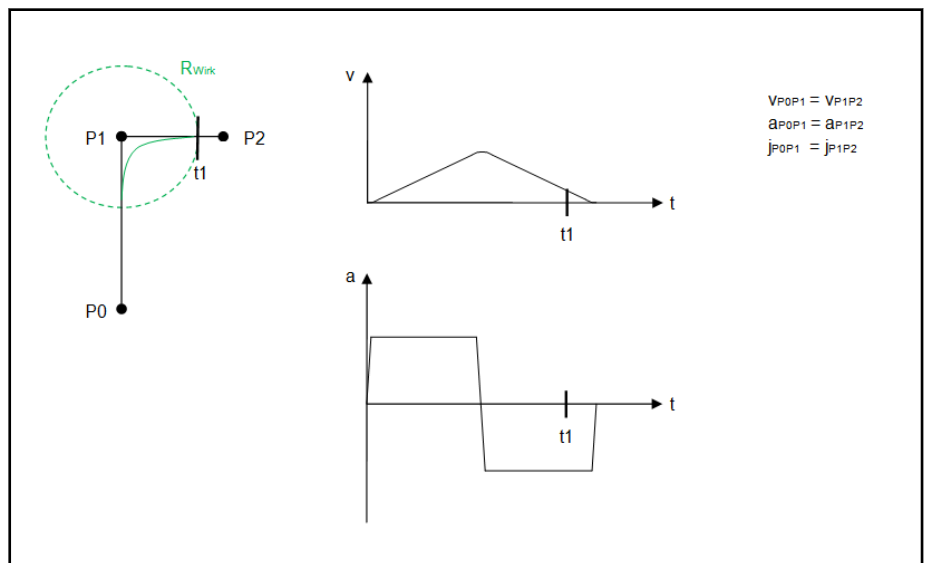


Fig. 2-31: R_{Eff} limitation with TO, VIA

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.

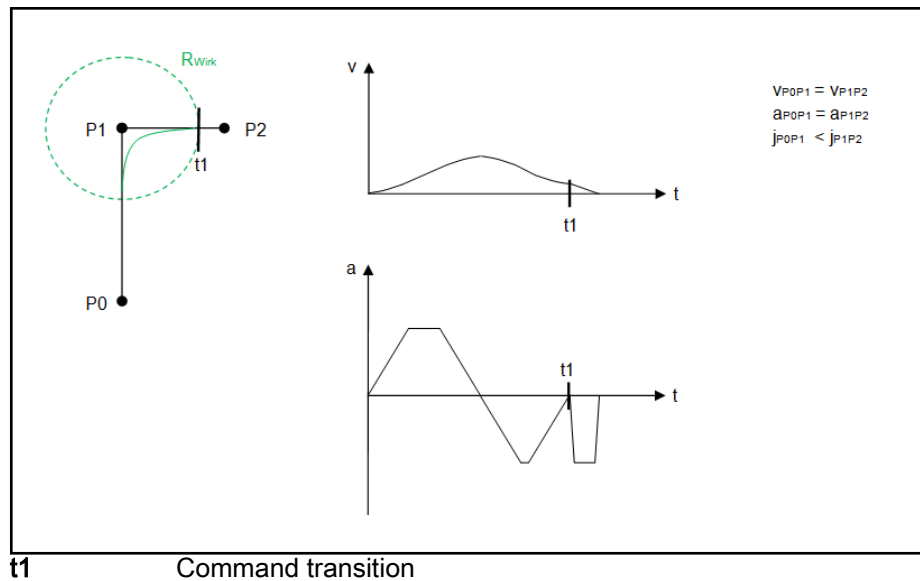


t1 Command transition

Fig. 2-32: 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. 2-33: 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.

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

At time t_1 , the motion command to P2 is issued.

At time t_2 , the motion command to P2 is prepared.

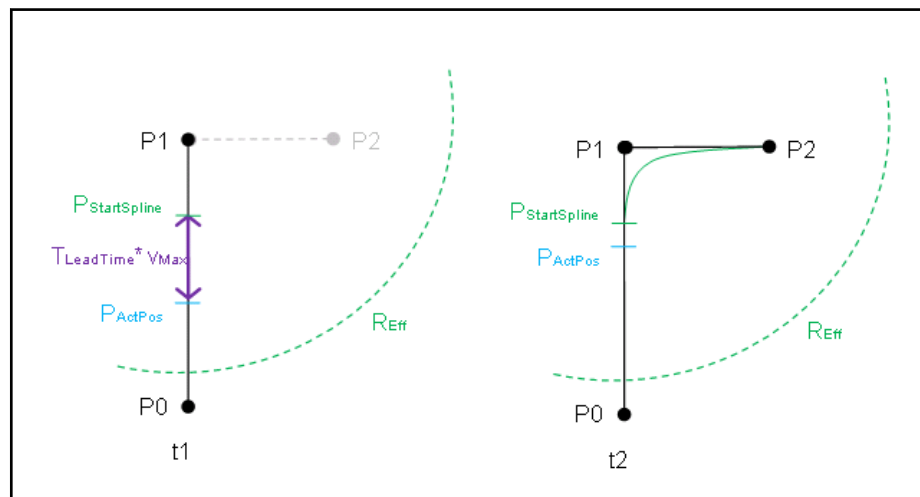


Fig. 2-34: Late blending with Cont_Slope

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).

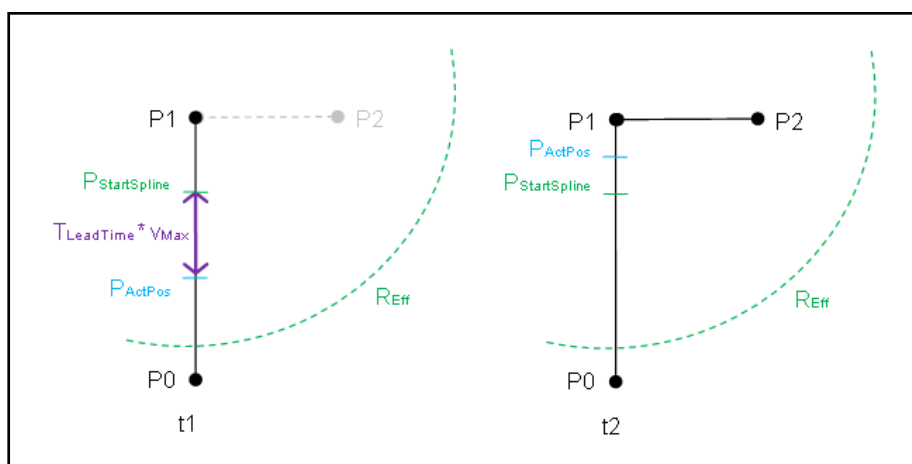


Fig. 2-35: Late blending with *Cont_Slope* failed

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 whether a spline is being calculated. Another bit indicates if the spline is valid.

2.10.6 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. 2-11: Blending possibilities with *Block_Slope*

Cont_Slope

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

Tab. 2-12: Blending possibilities with *Cont_Slope*

2.11 Robot Control, Move ID

The Move ID is a numerical value, specifying the current robot position on the web.

The integer part references is a number of the motion instruction. This number is automatically increased upon each new motion instruction.

Functional description

The decimal part shows the motion progress in percent.

Example:

Move ID of a motion

- Get Move ID: If no motion command has been sent yet, the Move ID is 0
 - Move Linear Absolute: A motion command is send
 - Get Move ID: After a command has been sent, the function Get Move ID returns, e.g., the value 0.12. This means that the motion is completed by 12%
 - Get Move ID: At the end of a motion, the function Get Move ID returns the value 1.00. This means that the motion is completed and that no other command is active
-

2.11.1 Robot Control, Move ID, setting

The Move ID can be set via the function [Set Move ID, page 38](#). The new Move ID applies to all commands that are commanded after setting the Move ID. Block already in the block buffer are not affected.

Example:

Setting the Move ID

- Move Linear Absolute: A motion command is send
 - Get Move ID: After a command has been sent, the function Get Move ID returns, e.g., the value 0.12. This means that the motion is completed by 12%
 - Set Move ID (15): For new commands, Move Id 15 is set
 - Move Linear Absolute: A second motion command is send
 - Get Move ID: The function Get Move ID returns, e.g., the value 0.75. This means that the first motion is completed by 75%
 - Get Move ID: The function Get Move ID returns, e.g., the value 15.25. This means that the second motion is completed by 25%
-

2.11.2 Robot Control, Move ID, blending

Block Slope The Block Slope mode blends by premature starting of the following block. The Move ID results from the addition of the process of the parallel running individual blocks.

Functional description

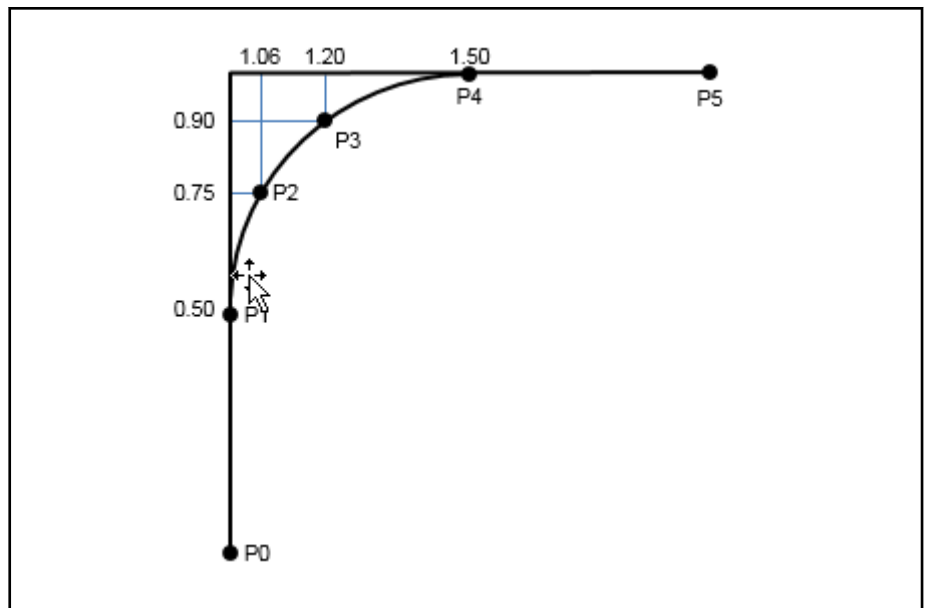


Fig. 2-36: Move ID Block Slope

- P0: The first block starts at this point. The Move ID is 0.0
- P1: The second block starts at this point. The first block is processed by 50%. The Move ID is 0.5
- P2: At this point, the first block is processed by 75% and the second block is processed by 6%. The Move ID is thus 0.81
- P3: At this point, the first block is processed by 90% and the second block is processed by 20%. The Move ID is thus 1.1
- P4: At this point, the first block is complete and the second block is processed by 50%. The Move ID is thus 1.5
- P5: At this point, both blocks are completed processed. The Move ID is thus 2.0

Cont Slope The Cont Slope mode blends by shortening the original blocks in the blending area and by replacing them with a spline. The Move ID in the blending area results from the processing progress of the spline.

Functional description

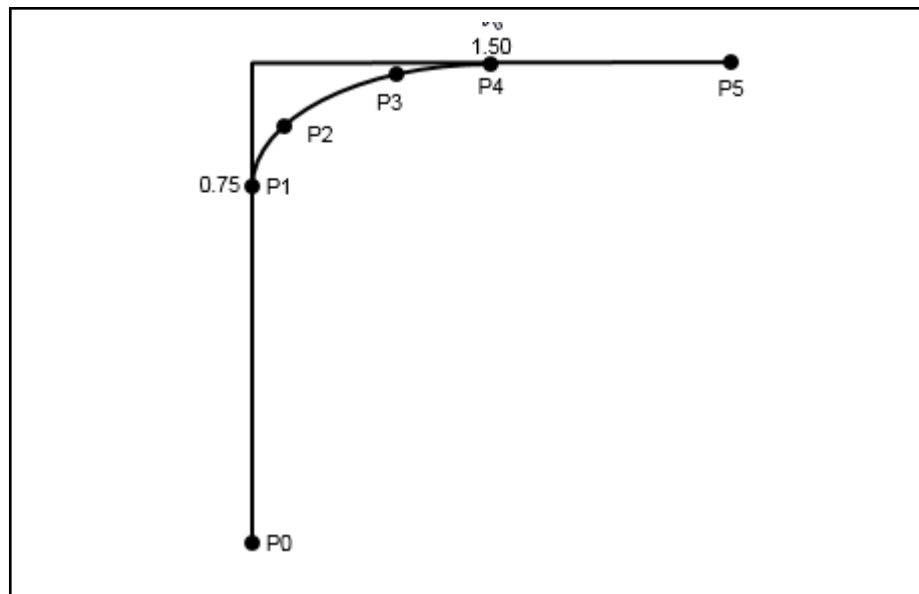


Fig. 2-37: Move ID Cont Slope

- P0: The first block starts at this point. The Move ID is 0.0
- P1: The blending area starts at this point. The first block is processed by 75%. The Move ID is 0.75
- P2: At this point, the spline is processed by 20%. As the spline is processed between 0.75 and 1.5, a Move ID of 0.9 ($0.75 + 20\%(1.5 - 0.75)$) is the result
- P3: At this point, the spline is processed by 80%. As the spline is processed between 0.75 and 1.5, a Move ID of 1.35 ($0.75 + 80\%(1.5 - 0.75)$) is the result
- P4: At this point, the spline is completed processed. The Move ID is thus 1.5
- P5: At this point, the second block is completed processed. The Move ID is thus 2.0

2.12 Stopping, Interrupting, Continuing Motions

2.12.1 Stopping Motions

The command [Stop on page 36](#) stops all motions of a kinematics.

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 67](#)).

- The contents of the block buffer are emptied

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

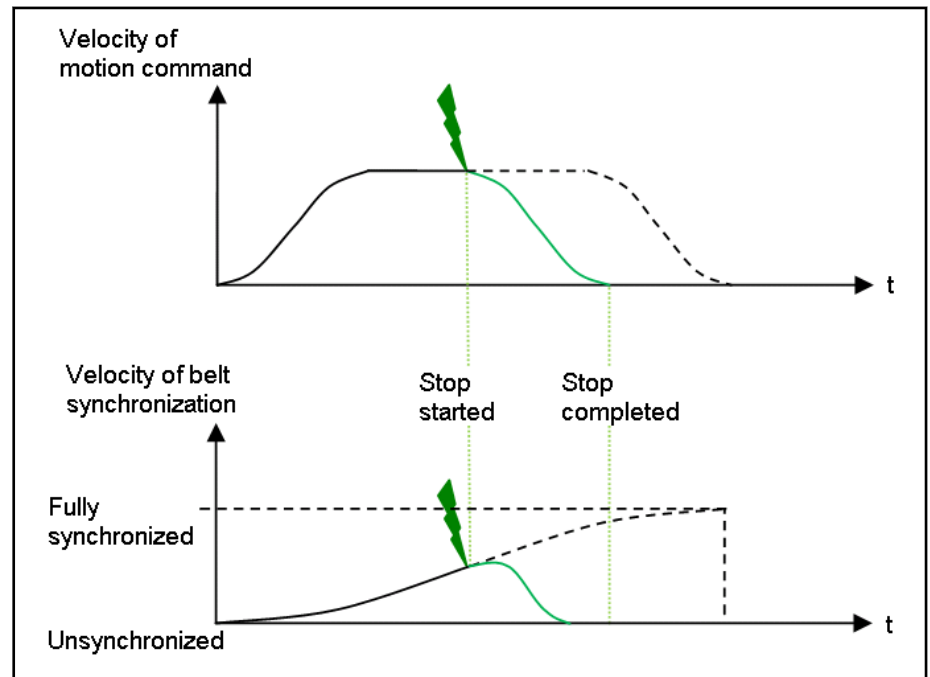


Fig. 2-38: Velocity curve in case of stop

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

2.12.2 Interrupting Motions

The command [Interrupt on page 36](#) 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

Functional description

- The contents of the block buffer remain unchanged

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

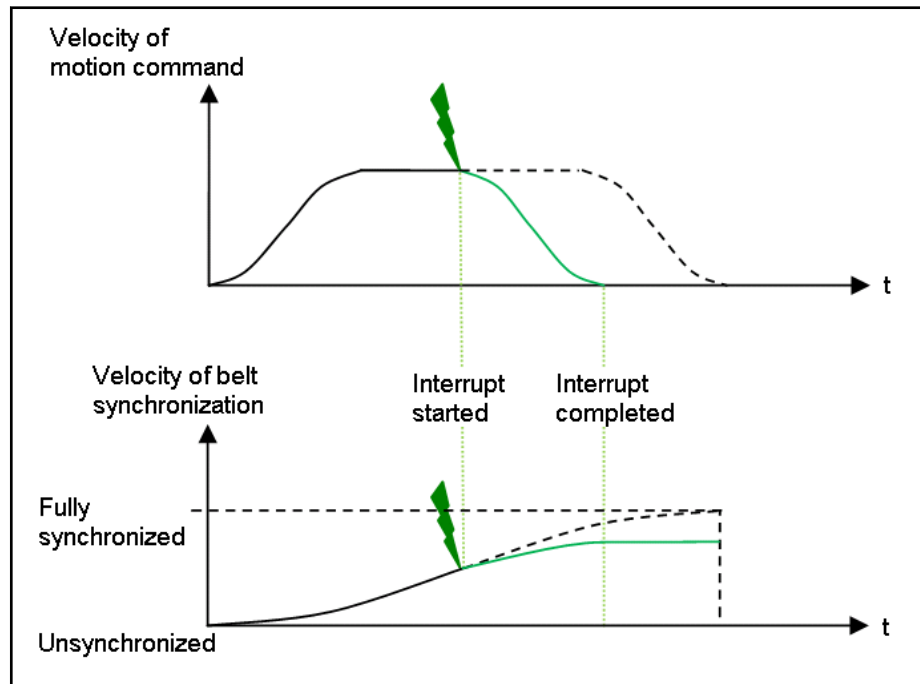


Fig. 2-39: Velocity curve in case of interrupt

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 36](#)
2. The motion is stopped using the command [Stop on page 36](#)

2.12.3 Continuing Motions

To restart a kinematics motion interrupted by the command [Interrupt on page 36](#), execute the command [Continue on page 36](#).

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

2.12.4 Waiting

Use the command [Wait on page 36](#) 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

Functional description

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.

2.13 Belt-synchronous motion

2.13.1 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. Depending on the configuration, the belt has to be straight or circular and may be positioned anywhere in the space. The maximum number of available belts per kinematics is equal to 16 minus the kinematic axes (positioning/orientation/auxiliary axes).

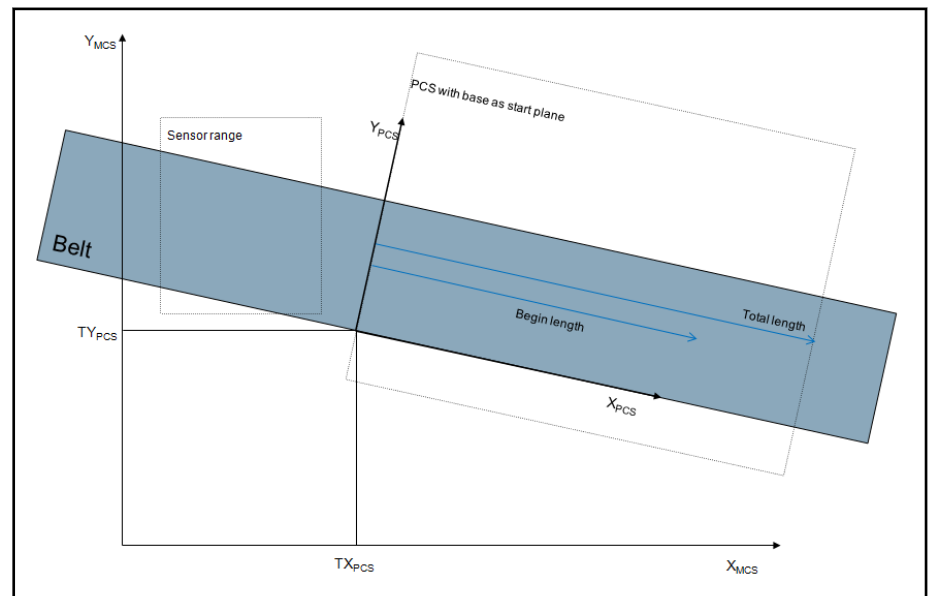


Fig. 2-40: Overview belt-synchronous - straight belt

The straight belt illustrated in the image is also represented by a product coordinate system (Product coordinate system (PCS) on page 27). " TX_{PCS} " and " TY_{PCS} " indicate the offset of the start plane in the machine coordinate system. "Begin length" and "Total length" define the distance between the start plane and the point of later synchronization or the belt lengths and are both applied in the direction of the X coordinate of the PCS.

Additionally, the sensor range before the start plane can also be realized by means of an individual product coordinate system to support commanding of kinematics.

Functional description

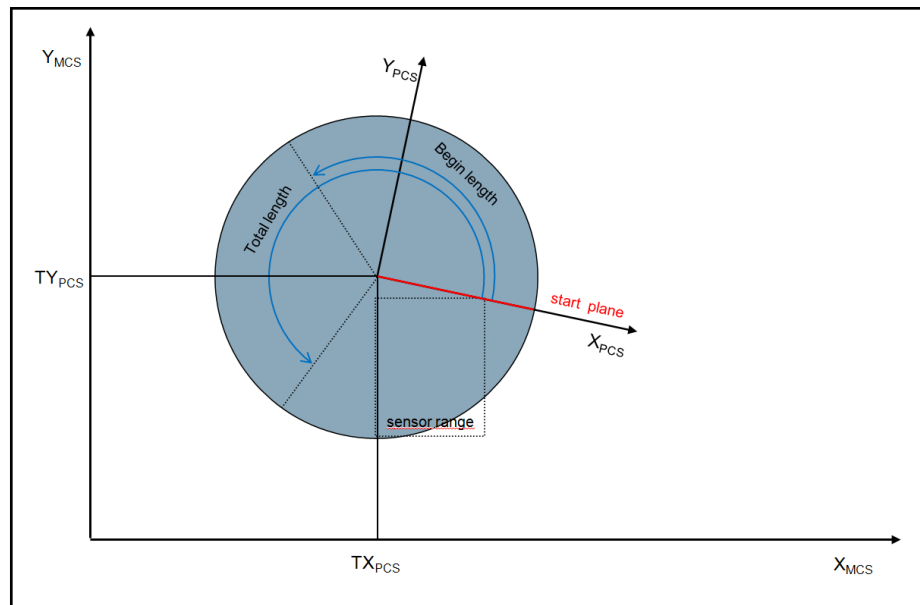


Fig. 2-41: Overview belt-synchronous - Turntable

The turntable illustrated in the image is also represented by a product coordinate system (Product coordinate system (PCS) on page 27) equivalent to the straight belt. Here too, " TX_{PCS} " and " TY_{PCS} " indicate the offset of the start plane in the machine coordinate system. "Begin length" and "Total length", however, define the angular rotation of the belt to the X-axis of the PCS.

Independent of the applied belt type, the motion of the belt is automatically and internally registered in the kinematics. Current belt values can be read via parameter [K-0-0100](#) (Current position, Cartesian Machine Coordinate System) or the KinData interface.

2.13.2 Belt configuration

The belt configuration consists of the following:

- [Creating a belt in K-0-0008 on page 66](#)
- [Basic configuration on page 66](#)
- [Belt-specific error reaction on page 67](#)

Creating a belt in [K-0-0008](#)

A belt has to be created in parameter [K-0-0008](#) (Axis configuration list). This may either be realized when adding a kinematics (Adding a new kinematics, step 2: Axis assignment, belt axis type) or by direct writing of the parameter. Only then can the belt be used.

Basic configuration

Belts have to be configured once before belt-synchronous traveling of a kinematics. Depending on the belt type (linear belt or turntable), this is realized using the command [Set Belt Configuration on page 37](#) or [Set Rotary Table Configuration on page 37](#). This way, a product coordinate system starting at PCS1 is automatically generated and numbered continuously (e.g.: belt 4 corresponds to PCS 4). Previously configured PCS are automatically overwritten by this command. Therefore, PCS that do not represent belts have to be configured later.

Command [Get Belt Configuration on page 38](#) can be used to determine the current configuration of a belt. Additionally, command [Get Trafo Values on page 38](#) can be used to read out the product coordinate system behind the belt.

The basic configuration consists of the following parts:

Functional description

- **Spatial position (Origin):** The position of the Y/Z-plane of the product coordinate system defines the start plane. The position of the X-axis defines the belt direction.
- **Begin 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.

"Begin 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.

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 37](#) to configure the error reaction.

Read out the configured error reaction using ["Get Belt Error Reaction on page 38"](#).



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 standard values for deceleration (1000 mm/s²) and jerk (10000 mm/s³).

2.13.3 Belt values and points on the belt

Points can be coupled with the motion of belts:

If the belt is moved, the point moves in the same way. Points are coupled to belts using the so-called belt value. The belt value is a part of the coordinates of each point. Each belt has an independent belt value. The position of the belt value within the coordinates defines the axis configuration of the kinematics:

Example:

3-axis delta with an orientation axis and 2 belts

A 3-axis delta with an orientation axis and 2 belts was configured. Therefore, every 5th coordinate of each point is the belt value for the first belt and every 6th coordinate the belt value for the second belt.

Points that are not coupled with a belt feature a belt value of 0 in the respective coordinate. Points that are coupled with a belt feature a belt value that is not 0 in the respective coordinate.

Besides the belt value of a point, there is also a current belt value of a belt representing the current position of the belt. The current belt value is continuously adjusted with the movement of the belt. The current belt values can be read out from the current position of the kinematics ([K-0-0100](#)).



The current belt value of a belt is not equal to the position value of the axis assigned to the belt. Although the two values are adjusted similarly, there may be an offset or a different modulation.

Functional description

For the current belt value too, the following applies:

The position of the belt value within parameter [K-0-0100](#) defines the axis configuration of the kinematics:

Example:

3-axis delta with an orientation axis and 2 belts

A 3-axis delta with an orientation axis and 2 belts was configured. Therefore, every 5th coordinate in parameter [K-0-0100](#) is the belt value for the first belt and every 6th coordinate in parameter [K-0-0100](#) the belt value of the second belt.

Let $K100$ be $P_{MCS} = (X, Y, Z, \varphi, b1, b2)$

- $b1$: Belt value for the first belt
- $b2$: Belt value for the second belt

If a point is to be coupled with the motion of a belt, the current belt value of the respective belt has to be read at the time of point recording and the recorded point to be coupled has to be assigned. For this purpose, the current belt value of the belt has to be read from the respective coordinate of parameter [K-0-0100](#) and copied into the respective coordinate of the point. The point may be in any coordinate system. The belt value is equal in each coordinate system.

Example:

3-axis delta with an orientation axis and 2 belts

For this camera, a PCS3 product coordinate system was defined and rotated and positioned towards the MCS in such a way that the recording plane of the camera is at the height of the belt. A product was detected by the camera. The camera provides the P_{PCS3} product coordinates = (5, 8, 0, 45, 0, 0). The point is located at $X = 5$, $Y = 8$ and rotated by 45 degrees. At the time the product was detected, the current position of the robot ([K-0-0100](#)) was recorded. Let it be $P_{MCS} = (10, 10, -700, 0, 20, 30)$. If the product is located on the first belt, the point P_{PCS3} coupled with the belt = (5, 8, 0, 45, 20, 0).



It is ensured that the current belt value in parameter [K-0-0100](#) is always a value not equal to 0. This ensures that the point is always coupled to a belt after copying the current belt value of a belt from parameter [K-0-0100](#) to a point.

This information can be applied by the system to determine the current position of a point coupled to the belt at any time. For this purpose, the internally recorded belt value is applied to the current belt value (see application example "Belt synchronization" [Belt synchronization on page 392](#)). For the displacement of a point to a point behind the start plane, the following general mapping rule is applied:

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

$$P'_{n\_PCS(k)} = (P'_{x(n)}, P_y, \dots, B_{ref})$$

k	Belt number
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 of point P at the time of point recording
B_{P(n)_PCS(k)}	Belt value of point P at the time n in product coordinate system k
P'_{n_PCS(k)}	Point behind the start plane at time n in the product coordinate system k

Fig. 2-42: Mapping of point P at time n-1 to a point behind the start plane at time n

Coupled 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 38](#), e.g. for display purposes. This function moves a coupled point to the current belt value and converts the spatial position to the new belt value. 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. Transformation to different coordinate systems lead to changes in other coordinates due to belt rotations or the use of rotary tables.

Example:

Tracking of points

A 2-axis Cartesian with one belt (PCS1) is configured. The belt (PCS1) is not shifted towards the MCS and rotated by 90 degrees. The belt moves in X direction in PCS1 or in Y direction in MCS. A point $P_{MCS} = (3, 5)$ was coupled with this belt. At this time, the belt had a belt value of 10. Therefore, the coupled point has the coordinates $P_{MCS} = (3, 5, 10)$, i.e. $P_{PCS1} = (5, -3, 10)$. If the belt is further moved by 100, the new belt value is 110. If the Move Point function is called with $P_{MCS} = (3, 5, 10)$, the function returns $P'_{MCS} = (3, 105, 110)$. Transformation of this point into the PCS1 coordinate system leads to a point $P'_{PCS1} = (105, -3, 110)$.

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

In parameter [K-0-0099](#) (Kinematic configuration) the belt modulo value may optionally be set to equal the axis modulo value.



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.

2.13.4 Belt operation

Running operation is divided into the following phases:

- [Synchronization on page 70](#)
- [Traveling fully synchronously on page 71](#)
- [Desynchronization on page 71](#)

Functional description

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. 2-13: Overview of the command possibilities for belt-synchronous operation

Synchronization For synchronization using [Move Linear Absolute on page 36](#), [Move Direct Absolute on page 36](#) or [Move Circular Absolute on page 36](#), only one point coupled with a belt has to be commanded.

If this point is in front of the start plane, the kinematics waits until the end point has passed the start plane and then moves to the end point in synchronization with the belt movement. An alternative is described in [Adaptive motion on page 72](#).

When the commanded point is reached, the kinematics is fully synchronous with the belt, i.e. the belt velocity is determined by the kinematics using the block distance. The greater the block distance the smaller the exceedance 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".

For this reason, synchronization with maximum values for acceleration and jerk [Synchronization with Limits on page 36](#) 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.

Functional description

Traveling fully synchronously

Fully synchronous traveling using [Move Linear Absolute on page 36](#), [Move Direct Absolute on page 36](#) or [Move Circular Absolute on page 36](#) works in the same way as described above in "Synchronization". Only one coupled point has to be commanded.

In addition, it is possible to travel by a relative distance on the belt using [Move Linear Relative on page 36](#), [Move Direct Relative on page 36](#) or [Move Circular Relative on page 36](#). 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 distance as belt value in addition to the commanded distance.

Example:

Two-axis Cartesian with one belt

A distance of 50 mm is to be moved in belt direction. The relative point P = (0, 0, 50) is commanded using the Move Linear Relative command.



In case of a linear motion, the path is straight relative to the belt. This is also the case if the belt is accelerated or decelerated during this motion.

In case of a direct motion (PTP), the path on the belt is undefined. In this case it is only guaranteed that the end point of the motion is completely synchronous to the belt (end point synchronous motion).

Desynchronization

For desynchronization using [Move Linear Absolute on page 36](#), [Move Direct Absolute on page 36](#) or [Move Circular Absolute on page 36](#), an uncoupled point has to be commanded (belt value of "0"). When the commanded end point is reached, the kinematics is asynchronous, i.e. the belt velocity is reduced by the kinematics using the block distance. The greater the block distance the smaller the exceedance of the block limit values during the desynchronization command. In reverse, the kinematics decreases the belt velocity in jumps with very small distances, e.g. belt distance close to "0".

Enter the belt value "1000000" at the position of the belt axis and issue the command [Move Linear Relative on page 36](#), [Move Direct Relative on page 36](#) or [Move Circular Relative on page 36](#) for relative desynchronization. This means, that the kinematics travels from its current position by the distance specified in the point definition and is desynchronized afterwards. Relative desynchronization is divided into three parts:

- Fully synchronous:
Relative motion fully synchronous to the belt
- Desynchronization:
Relative motion with simultaneous desynchronization
- Asynchronous:
Relative asynchronous motion

The length of the individual sections of the relative desynchronization motion can be configured in percent with the command [Set Belt Desync Relative Configuration on page 36](#). In the default setting, the fully synchronous section is 0%, the desynchronization section is 50% and the asynchronous section is 50%. The absolute spatial end point is only known after the desynchronization section.

Functional description

Furthermore, desynchronization may be executed using the command [Desynchronization with Limits on page 36](#). 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.

2.13.5 Adaptive motion

As a rule, belt-synchronous motions 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 until the start plane is reached. If this waiting or standstill time is not desired, the option "adaptive motion" can be enabled via parameter [K-0-0099](#) bit 6.

Activation leads to a start of the motion as soon as the coupled point is commanded even if the point is before the start plane. The target point of the motion is the shifted coupled point in the start plane. The progress of motion depends directly on the progress of the belt. The motion is completed as soon as the product (the coupled point) has reached the start plane.

This offers the following advantages:

- Time savings: The coupled point is already reached when this point has reached the start plane. Without "adaptive motion", the first motion would be started at this point in time.
- Preservation of mechanical parts: The waiting time is applied for motion. This way, slower motions are generally applied to preserve mechanical parts.
- Energy savings: Energy is saved thanks to slower motion with reduced acceleration and deceleration.

Characteristics of "adaptive motion":

- If an error occurs, the motion is stopped on the path with the known error reactions.
- As customary, the stop functionality is available.
- The override mechanism is not available in case of adaptive interpolation as the motion of the target point is decisive for the interpolation and thus, the product reference would be lost. The override equal to "0", however, is possible at any time.

If you want to stop the kinematics, the belt axis has to be stopped as this belt axis is the reference for interpolation.

The following applies: Stops the belt and the commanded target point as well as the kinematics.

- Interrupt and Continue as well as Override are not available.
- The limit values on the path are also complied with with this interpolation type. Under some circumstances, this can mean the target point is only reached after the start plane.
- With "adaptive motion", the path can also reach points before the start plane. However, it is ensured that the path is always in the mechanical working area of kinematics.

The following generally applies: The function can dynamically be activated at any time. If the adaptive interpolation is enabled by a belt-synchronous motion, adaptive interpolation is possible from the next belt-synchronous motion. The control automatically decides when an adaptive interpolation of belt-synchronous motion takes place.

Functional description

Furthermore, the functionality can be disabled at any time during operation.

The entire functionality does not effect the commanding of the kinematics, i.e. available programs do not have to be changed to be able to use this functionality.



If the coupled point is not only before the belt area but also before the working area of the kinematics, the kinematics are also moved directly to the section of coupled point and start plane. However, synchronization is started as soon as the coupled point is in the working area of the kinematics.

For more information about this topic, refer to [Belt synchronization with Adaptive Motion, page 402](#).

2.13.6 Blending and belt synchronization

Blending of the individual motion blocks is guaranteed in all cases as described in [Blending on page 52](#). Belt-synchronous traveling is not linked to a certain type of blending but independent of the motion criteria.

Blending is limited in case of the following commands:

- [Synchronization with Limits, page 36](#)

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. Blending is not possible.

- Relative desynchronization

It is only possible to blend the asynchronous section of the relative desynchronization.

2.13.7 Summary

A belt is represented by a **PCS** ([Product coordinate system \(PCS\) on page 27](#)) which may be offset and rotated in relation to the **MCS** ([Machine coordinate system \(MCS\) on page 26](#)). It may be straight or circular.

For an application example of a straight belt, refer to [Belt synchronization on page 392](#).

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 36](#). 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 79](#).

2.14 Switching on the belt

Functional description

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

Functional description

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

Functional principle

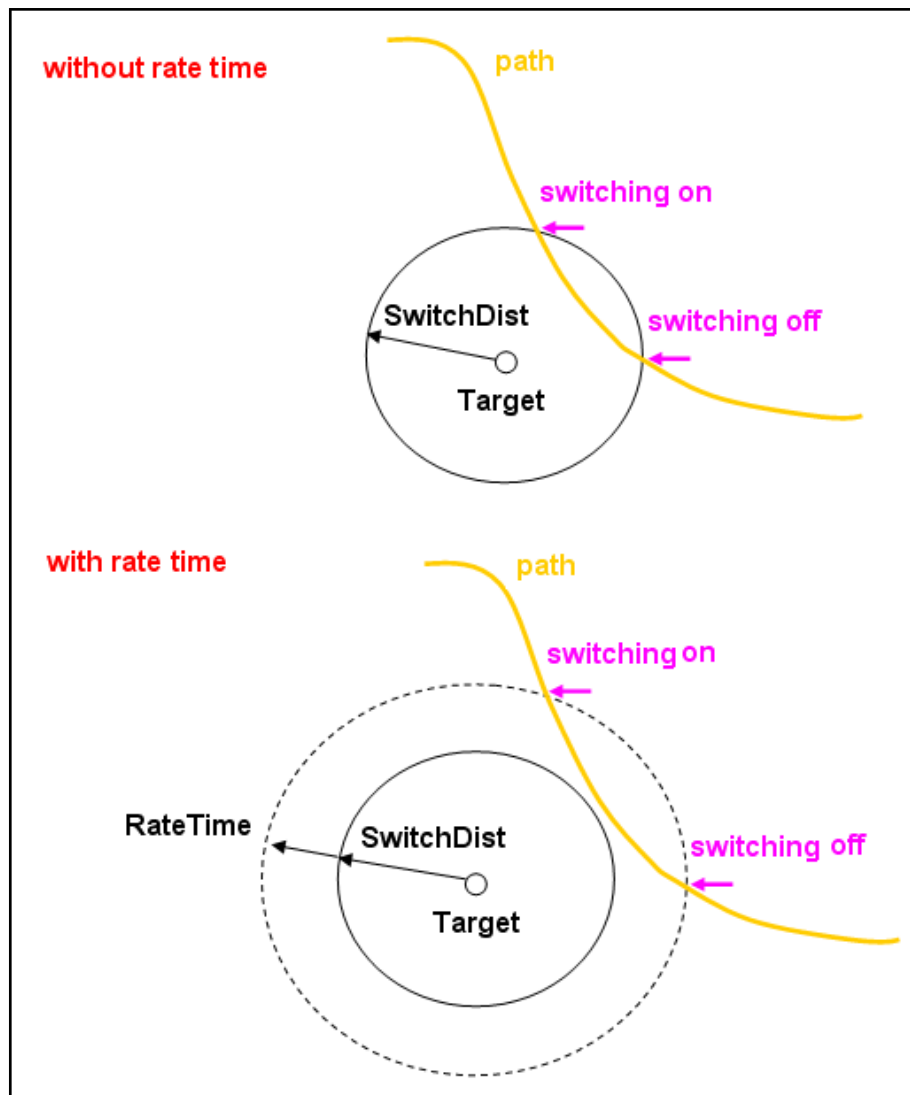


Fig. 2-43: Functional principle

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

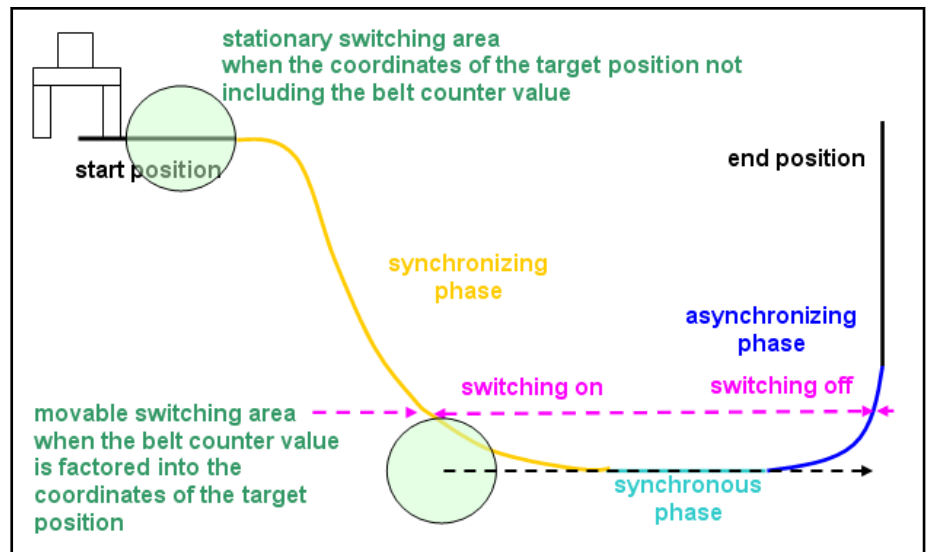


Fig. 2-44: Example

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

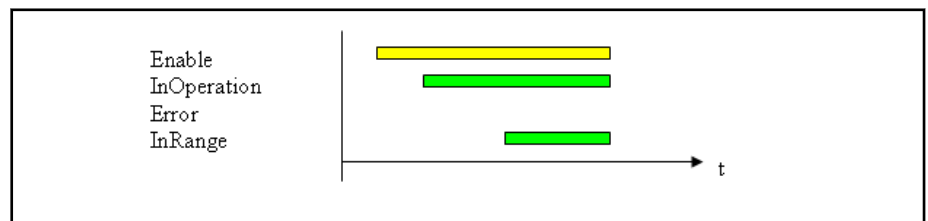


Fig. 2-45: Time response

The function block ML_SwitchOnPath, ML_Robot, 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.

2.15 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.

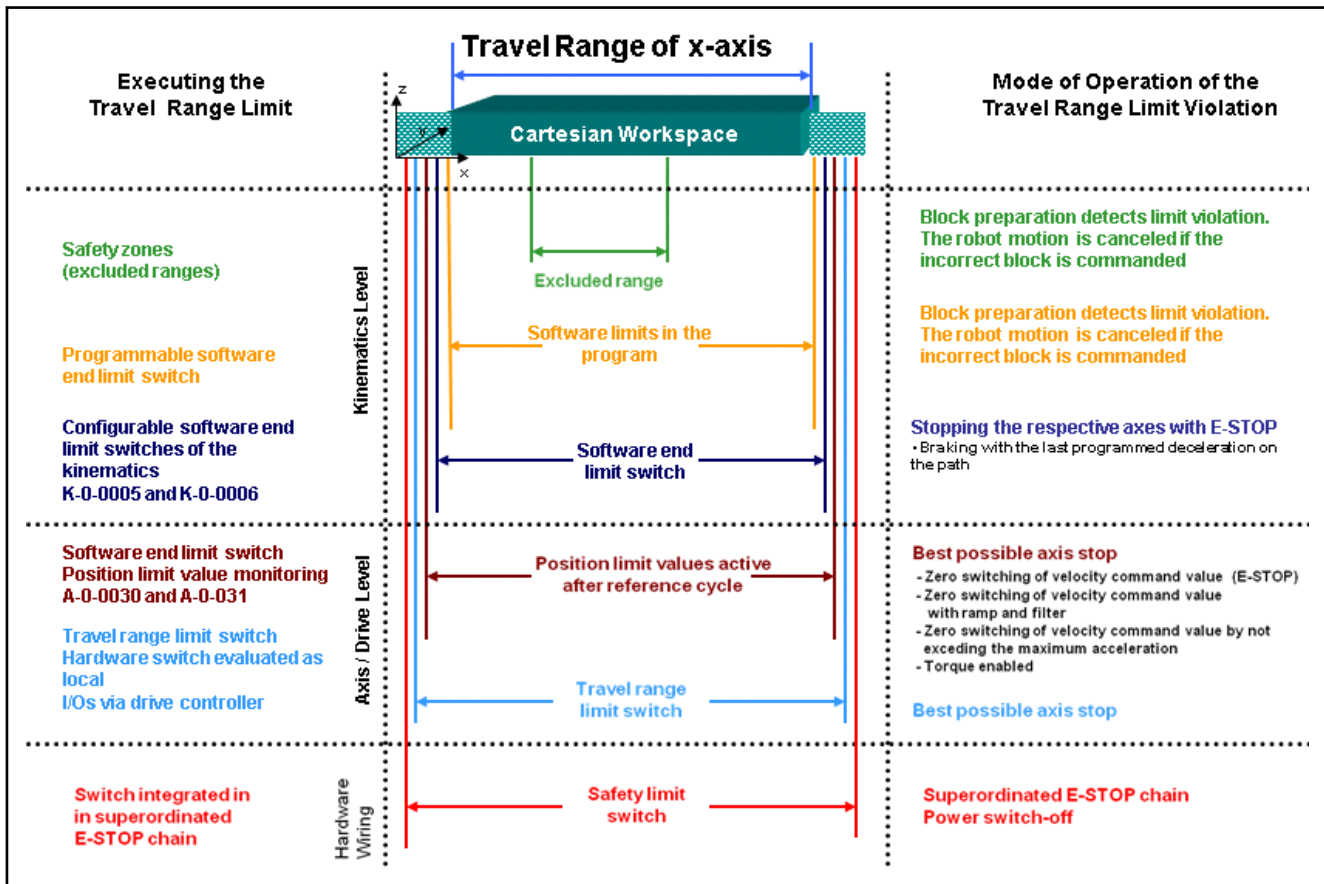


Fig. 2-46: Overview of traversing range limits

Programmable safe zones

Programmable safe zones are defined by the command [Set Safe Zone](#) on [page 37](#).

32 safe zones are available.

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 a safe zone, such as the minimum point S1 (0,0,0) are also part of the safe zone.

Functional description

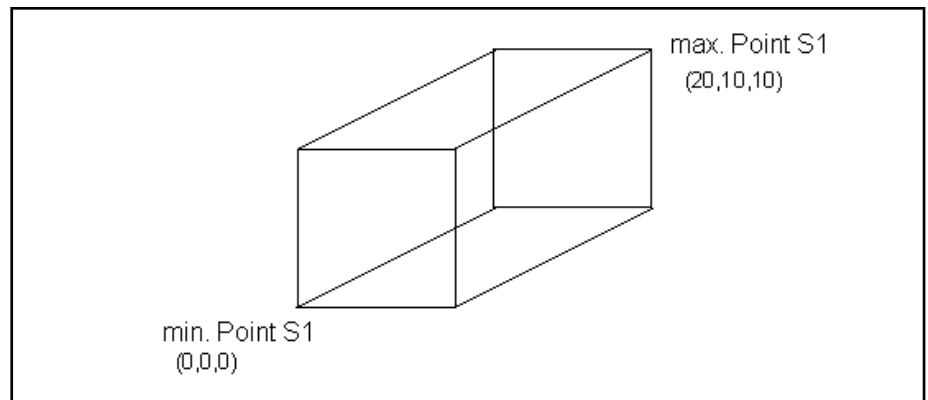


Fig. 2-47: Maximum and minimum point of a safe zone

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 (Robot Control V1) 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. If, e.g. the TCP position is set to the safe zone limit, the TCP kinematics is within the defined safe zone. The kinematics can thus be commanded to point outside the safe zone without violating a safe zone.

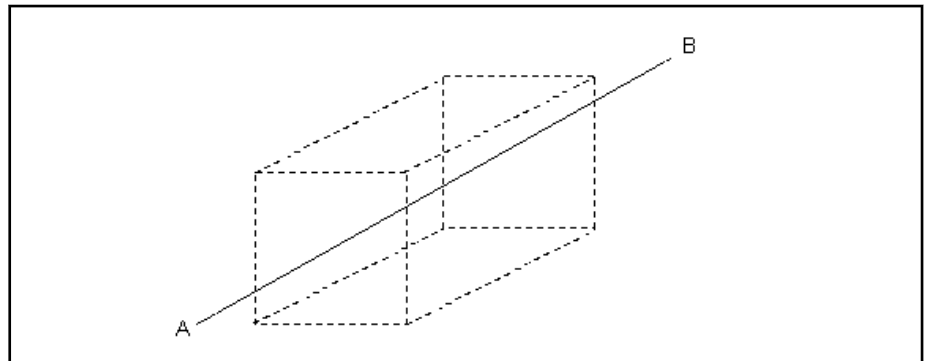


Fig. 2-48: Safe zone with motion from A to B

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

- Each kinematics considers only 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 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

Functional description

- The maximum point must be larger than the minimum point, since the safe zone check would otherwise become 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 37](#).

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.

Software limit switchesPosition limit value monitoring [A-0-0030](#) and [A-0-0031](#)

The position limit values of the axes are monitored by the kinematics.

It is not possible to command points whose target or intermediate point (for CIRCULAR interpolation) exceeds the position limit values in the axis coordinates. This verification takes place in the block preparation.

Moreover, the commanded kinematic position is checked cyclically for the position limit values of the axes. In contrast to Robot Control V1, changes in the parameters [A-0-0030](#), Position limit value, positive and [A-0-0031](#), Position limit value, negative are directly applied.

2.16 Errors and error reactions

2.16.1 Errors and error reactions, general information

Several messages, warnings and errors may be returned.

The following is supported:

Error abbreviation	Meaning	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. 2-14: 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 79](#)
- [Belt-specific error reaction on page 79](#)

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 33](#)). 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 35](#)).

2.16.2 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, page 35](#))
- **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.

2.16.3 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 37](#). 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 of an error, desynchronization is carried out over 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. 2-15: 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.

Functional description

2.17 Cyclic command value specification

2.17.1 Cyclic 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.

2.17.2 Start and operation of the cyclic command value specification

The cyclic command value specification is started using [Open Cyclic Channel](#) and [Open Cyclic Direct Channel](#), page 36. Adding the command values in MCS is performed in the Open Cyclic Channel command. The command values in the ACS (PTP motion) are added in the Open Direct Cyclic Channel command. Thus, the arms can be changed during the cyclic command value specification.

Blending the cyclic command value range is possible (see [fig. 2-50 "Schematic illustration of the cyclical command value specification with blending" on page 81](#)) 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. Blending between PTP and linear motions is not possible.

Furthermore, [fig. 2-49 "Schematic illustration of the cyclical command value specification without blending" on page 80](#) explains cyclical command value specification without blending.

Once per motion cycle, a new kinematics command value **has** to be set using the function [Write Cyclic Channel on page 36](#).



The function [Write Cyclic Channel on page 36](#) **must** be called in the motion cycle as otherwise, unexpected error messages or jerking axes can sporadically occur!

The command value may be set in any coordinate system.

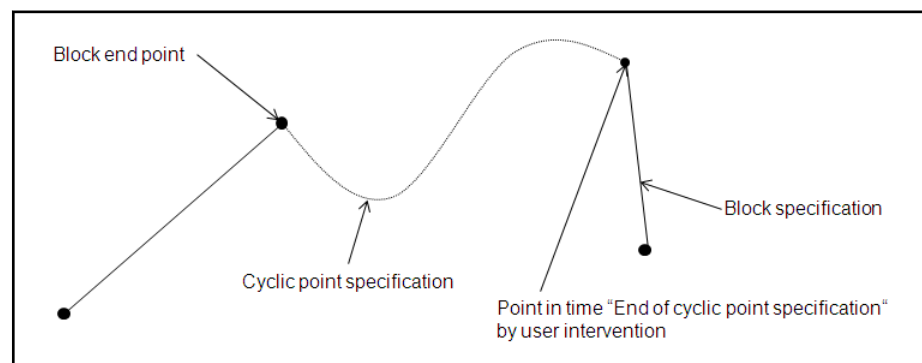


Fig. 2-49: Schematic illustration of the cyclical command value specification without blending

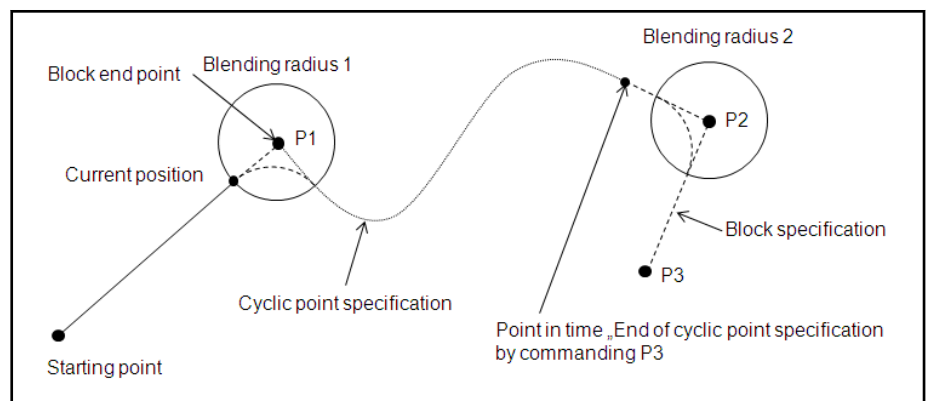


Fig. 2-50: Schematic illustration of the cyclical command value specification with blending

2.17.3 Cyclic command value specification

In the cyclic 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 Pos on page 38](#).

Based on this, the command values can be specified cyclically (see [List of all motion commands and functions on page 36](#)).

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.

2.17.4 Termination of cyclic command value specification

A cyclic command value specification is completed if:

- An error reaction is started
- A Stop or Interrupt is commanded
- The velocity override is set to 0
- A new traversing command is commanded

If the cyclic command value specification is completed using a traversing command, blending on this traversing command is possible, see [fig. 2-50 "Schematic illustration of the cyclical command value specification with blending" on page 81](#). 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 is determined that is the starting point for the next travel command, see point P2 in [fig. 2-50 "Schematic illustration of the cyclical command value specification with blending" on page 81](#).

Functional description

[blending" on page 81](#). 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.

2.18 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 38](#).

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

Read the results using function [Get Performance on page 38](#), 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. 2-16: Performance measurement entries

The following measurements are available (excerpt):

Name of the entry	Description
Shared memory	In the PLC there is a Shared Memory, ML_Robot, for each kinematics 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 34)
GetCmd	Reading status information of commands (see Command state on page 36)

Functional description

Name of the entry	Description
prepare	Block preparation
MoveExecute	Cyclical execution of a move command

Tab. 2-17: Available measurements (excerpt)

3 Interface functions

3.1 Interface function, general information

IndraWorks Engineering supports work with kinematics in the "Robot" folder.

This folder is available in a

- IndraMotion MLC, hardware basis IndraControl CML 25/45/65/75 or
- IndraMotion MLC, hardware basis IndraControl XM21/XM22 with "Robot Control" function package.

In the "Robot" folder, the Global Variable List "Global Data" can be added to kinematics - if not available yet - (context menu of the "Robot" folder or main menu item "Robot").

For each kinematics created in the "Robot" folder, an individual folder with the kinematic name is created.

3.2 Robot folder

3.2.1 Robot folder, general information

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

The menu item contains the following submenu items:

- **Add:** Add a kinematics
Also refer to [Adding a new kinematics on page85](#)
- **Kinematic commissioning:** The commissioning dialog of the kinematics can be opened
Also refer to [Kinematic commissioning on page87](#)
- **PathViewer:** Opens the PathViewer
Also refer to [PathViewer on page95](#)
- **Parameter:** Opens the parameter export and import for all kinematics
- **Export:** Exports the kinematic configuration
- **Import:** Imports an exported kinematic configuration
- **Online comparison:** Compares online and offline device data
- **Compare:** Compares two projects

The same submenu items can also be found in the context menu of the "Robot" node

Additionally, the following submenu items are available:

- **Find element:** Searches within the kinematic configuration
- **New view:** The kinematic configuration is displayed in another Project Explorer window

3.2.2 Adding a new kinematics

To add a new kinematics, click on **Add** below the "Robot" node.

A wizard guides the creation of a new kinematics:

Step 1: Entering device names, comment and author

Interface functions

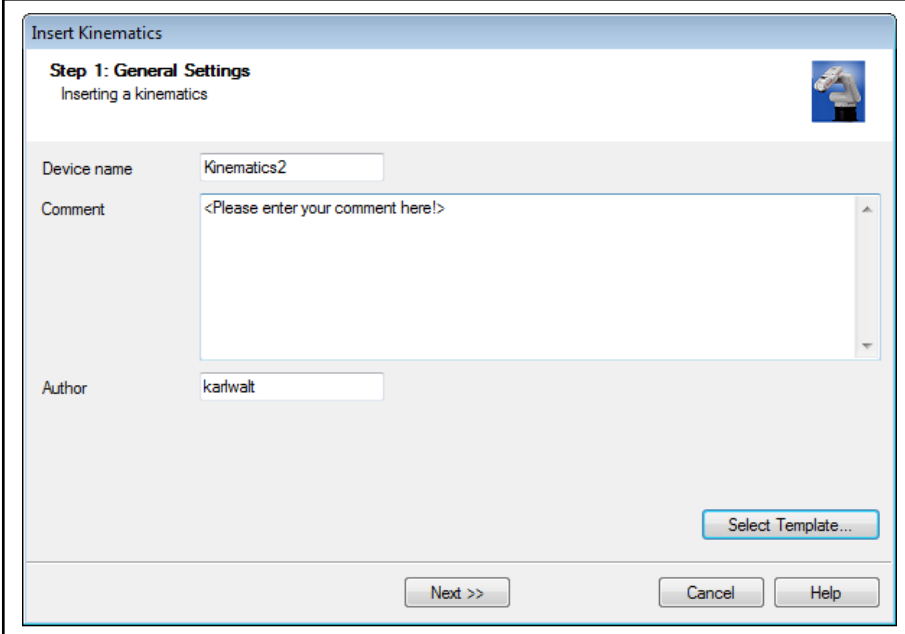


Fig. 3-1: Creating kinematics, step 1

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

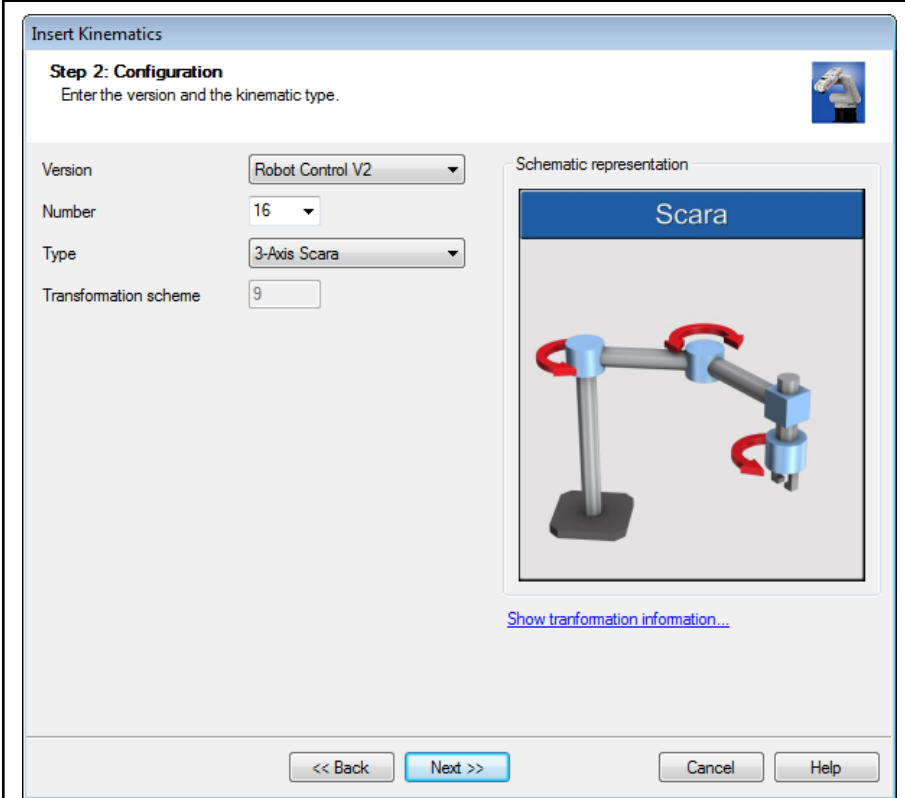


Fig. 3-2: Creating kinematics, step 2

Step 3: Assign axes

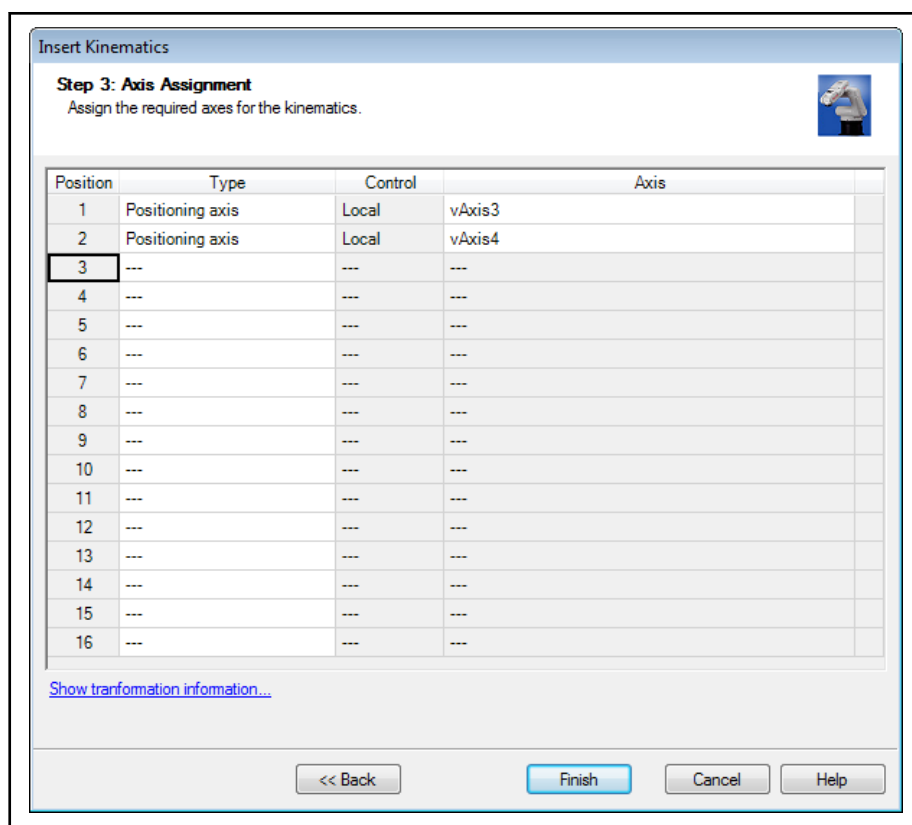


Fig. 3-3: Creating kinematics, step 3

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

After the kinematics has been created, a new menu item (kinematic name) is displayed below the "Robot" node.

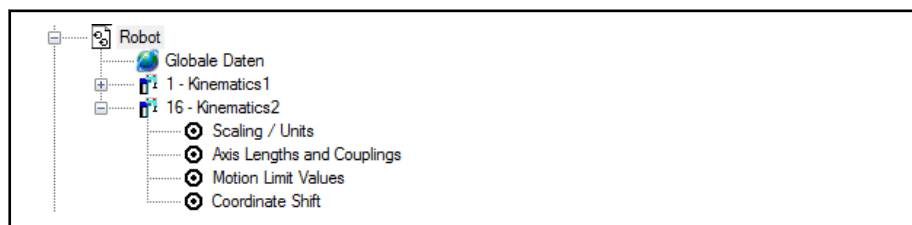


Fig. 3-4: New entry "Kinematics2"

3.2.3 Kinematic commissioning

Overview

Use the "Kinematic Commissioning" dialog in the context menu of the Robot node to move kinematics without writing a PLC program. This is especially useful during commissioning if the PLC program is not complete or if a kinematics is to be moved differently as intended in the PLC program for test purposes.



This dialog can only be selected in online mode. Furthermore, kinematics can only be operated in communication phase P4 (BB).

The dialog is divided into an overview and a details section.

Interface functions

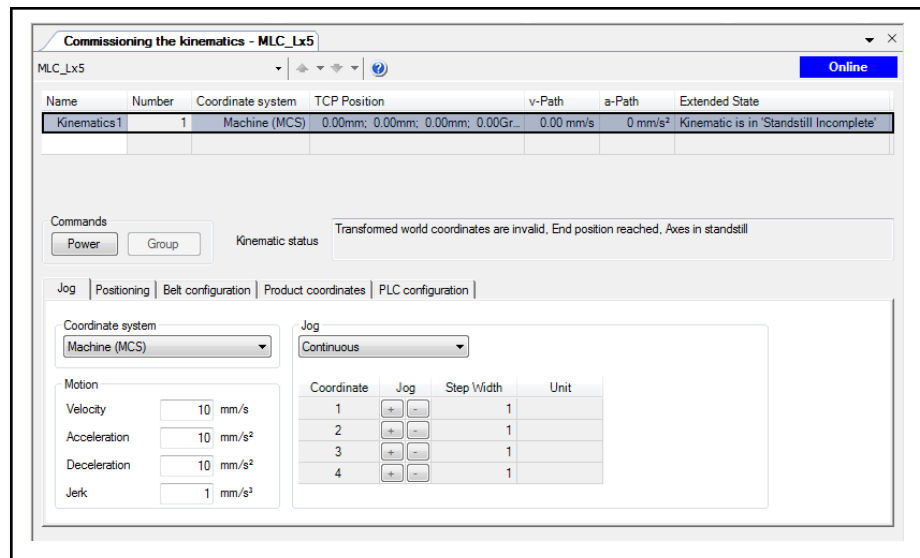


Fig. 3-5: Kinematic commissioning dialog

Overview section

The most important kinematic data can be displayed in the overview section. By selecting a configured kinematics in the "Name" column, the other columns are filled with data. Additionally, the coordinate system in which the TCP position is displayed can be selected.

Apart from the kinematic name, the following is displayed:

- Number: Kinematic number
- TCP position: Current TCP position in the selected coordinate system
- v-path: Current path velocity
- a-path: Current path acceleration
- Extended status: Extended kinematic status (corresponds to the decoded content of parameter [K-0-0022](#))

To delete a kinematic, either use the blank entry in the "Name" column or use the <Del> key (after having select the relevant line).

Details section

In the details section, motion commands can be triggered, belt and production coordinate systems can be configured and PLC configuration blocks can be generated. These motion commands are oriented at the state diagram of the PLCopen (refer to "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries", chapter "State diagram of a kinematics") and identical to the commands generated by an equivalent function blocks.

All commands and values refer to the kinematics selected in the overview section. Via the tabs, the relevant commands can be selected and command values can be specified. The fields in a tab are mere entry field, i.e., no status is specified. If an invalid value is entered, the input field and its tab are colored in red and an error message is displayed in the tooltip of the entry field. No motion command can be sent to the axis, i.e. corresponding buttons are disabled.

Furthermore, the power of all axes assigned to the kinematics in the overview section can be activated and the kinematics can be grouped. Via the "Kinematic status" field, the current status of the kinematics is represented, corresponding to the decoded content of parameter [K-0-0021](#).

Safety functions

Before displaying the actual commissioning dialog, it has to be confirmed that all measures ensuring safe kinematic operation have been taken.

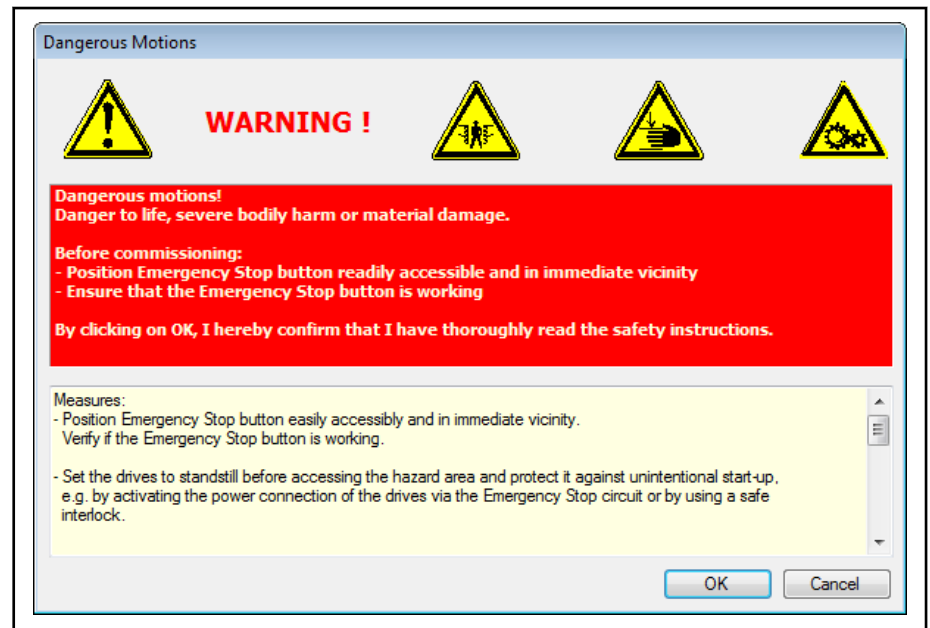


Fig. 3-6: Warning when opening the kinematic commissioning dialog

While the commissioning dialog is opened, all kinematic axes in the overview section can be stopped immediately using the "Kinematics OFF" window.

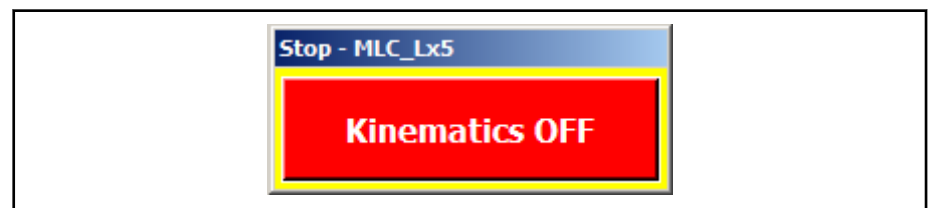


Fig. 3-7: Stop all kinematic axes

Kinematics may only be moved when they are displayed in the dialog. Thus, all axes are stopped when the dialog is closed. Furthermore, a running kinematics cannot be deleted or changed in the overview section.

To avoid conflicts with the PLC program, kinematics can only be moved if one of the following conditions apply:

- All PLC user applications are stopped
- All running PLC user application contain the variable "AllowKinematicsCommissioning". Its value is TRUE in the "UserVarGlobal" or "UserVarAppGlobal" if a device application is used. To move axes at a running PLC, create a Boolean variable called "AllowKinematicsCommissioning" in the node "UserVarGlobal" or "UserVarAppGlobal". Apply this variable to the symbol configuration of the PLC and set its value to TRUE. Thus, the PLC program can control whether and when moving axes via the dialog is permitted

If kinematics are moved via the commissioning dialog and if the previously mentioned conditions are not fulfilled anymore, all kinematics are immediately stopped and further motion commands are inhibited.

Interface functions



When closing the dialog, a stop command is sent to all running kinematics and it is waited for their standstill. If a kinematics requires more than two seconds up to the standstill, the dialog is still closed and a warning is issued in the IndraWorks status bar. In this case, the kinematics is still stopped, but remains in the "Stopping" state. To finally stop a kinematics ("Standstill" state), open the dialog at any time and issue the "Stop" command again to the kinematics. Optionally, stop the kinematics via the PLC.



Response when connection is interrupted between the IndraWorks interface and the control.

If the connection between PC and control is interrupted (e.g. network cable defective or removed), the control outputs the error "F536000E MLPI: Watchdog". This stops all commanding kinematics automatically.

Grouping a kinematics

In order to move a kinematics, activate the power of all corresponding axes and assign this axis to the kinematics. The enabling functionality corresponds to the one of the "MC_Power" function block (refer to the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Library"). In case of a successfully completed power up of all axes assigned to the kinematics, the button **Power** is highlighted in green. In case of a successfully completed kinematic axis grouping, the button **Group** is highlighted in green.

By pressing the relevant button again, the axes can be ungrouped and the power can be switched off. Both cases (power up and grouping or ungrouping and switching power off) can only be executed in the correct order. Consequently, a kinematics cannot be grouped if the axes were not powered up. If the power of the axes cannot be switched off if the kinematic is grouped.

To power up the axes and the kinematic can be grouped, the following prerequisites have to be met:

- Sercos bus is in P4 (operating mode)
- All corresponding axes are not in P2 (parameterization mode)
- Control and power section are ready for operation (i.e. bit 14 is set in the parameter A-0-0021)
- No error is pending at the kinematics or the corresponding axes

If one of the mentioned prerequisites is not met, power up is not possible and the missing prerequisites are shown as tooltip next to the buttons.

In case of virtual axes, power up is not explicitly necessary. These axes are enabled automatically and can be grouped if all other requirements are met.

Jogging

A grouped kinematics can be jogged continuously or incrementally in a coordinate system in the "Jog" tab. Under "Coordinate system" select the desired coordinate system and enter the velocity, acceleration, deceleration and jerk values under "Motion". Under "Jog", select continuous or incremental jogging. When selecting incremental jogging, the step width can still be specified. Using the buttons + and -, the command for the coordinate can be executed.

Interface functions

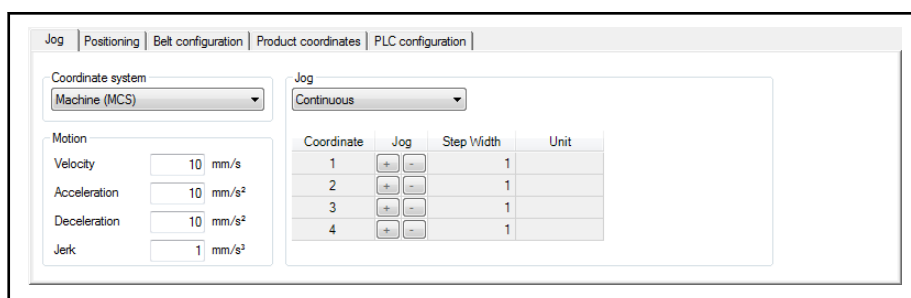


Fig. 3-8: Kinematic commissioning; "Jogging" tab

The functionality corresponds to the functionality of the function blocks "ML_GroupJogStep" and "ML_GroupJogCont". By selecting the jogging type, the relevant function block is implicitly enabled. The input fields are analogously significant to the input parameters of the function block:

Incremental jogging

Input field in the dialog	Input parameters of the function block "ML_GroupJogStep"
"Coordinate system" section	CoordSystem
Motion/velocity	Velocity
Motion/acceleration	Acceleration
Motion/deceleration	Deceleration
Motion/jerk	Jerk
Jogging/step width	Distance

Tab. 3-1: Assigning the input fields in the dialog to the inputs of the "ML_GroupJogStep" function block

Continuous jogging

Input field in the dialog	Input parameters of the function block "ML_GroupJogCont"
"Coordinate system" section	CoordSystem
Motion/velocity	Velocity
Motion/acceleration	Acceleration
Motion/deceleration	Deceleration
Motion/jerk	Jerk

Tab. 3-2: Assigning the input fields in the dialog to the inputs of the "ML_GroupJogCont" function block

Positioning

A grouped kinematics can be positioned in any coordinate system in the "Position" tab. Select the coordinate system in the "Coordinate system" section as well as the target position and enter the intermediate position, if the positioning type "Circular" is selected. Under "Motion", select the positioning type and enter the values for velocity, acceleration, deceleration and jerk. Using the **Start** button, execute the command and stop the command with **Stop**.

Interface functions

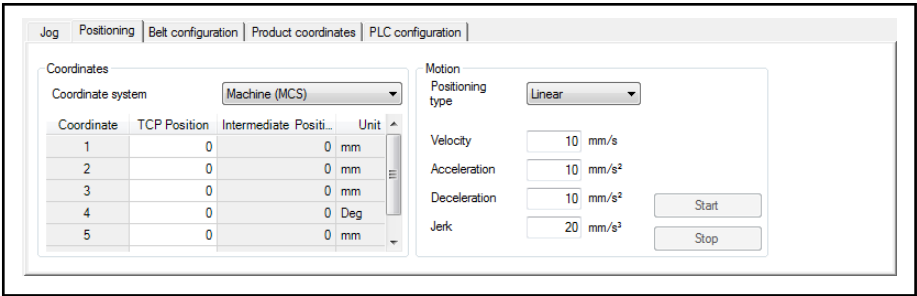


Fig. 3-9: Kinematic commissioning; "Positioning" tab

The functionality corresponds to the functionality of the function blocks "MC_MoveLinearAbsolute", "MC_MoveDirectAbsolute" and "MC_MoveCircularAbsolute". By selecting the positioning type, the relevant function block is selected. The input fields are analogously significant to the input parameters of the function block:

Input field in the dialog	Input parameters of the "MC_MoveLinearAbsolute" and "MC_MoveDirectAbsolute" function blocks
Coordinates/coordinate systems	CoordSystem
Coordinates/position TCP	Position
Motion/velocity	Velocity
Motion/acceleration	Acceleration
Motion/deceleration	Deceleration
Motion/jerk	Jerk

Tab. 3-3: Assigning the input fields in the dialog to the inputs of the"MC_MoveLinearAbsolute" and "MC_MoveDirectAbsolute" function blocks

Input field in the dialog	Input parameters of the "MC_MoveCircularAbsolute" function block
Coordinates/coordinate systems	CoordSystem
Coordinates/position TCP	EndPoint
Coordinates/intermediate position	AuxPoint
Motion/velocity	Velocity
Motion/acceleration	Acceleration
Motion/deceleration	Deceleration
Motion/jerk	Jerk

Tab. 3-4: Assigning the input fields in the dialog to the inputs of the "MC_MoveCircularAbsolute" function block

Belt configuration

In the "Belt configuration" tab, belts can be configured that are part of the selected kinematics. If the kinematic has several belt axes, the belt axis to be configured can be selected under "Axis". In the sections "Belt" and "Coordinate offset", the belt type, the start and total length as well as the offset and the rotation against the machine coordinate system has to be specified. In the "Error reaction" section, specify the belt reaction in case of

Interface functions

an error. Depending on the selected reaction time, specify the reaction type, the desynchronization time or the deceleration and the jerk. Using the **Write configuration** button, the belt is configured in the control.

Fig. 3-10: Kinematic commissioning, "Belt configuration" tab

The functionality corresponds to the functionality of the function blocks "ML_SetBeltConfiguration", "ML_SetRotaryTableConfiguration" and "ML_SetBeltErrorReaction". By selecting the belt type, the relevant function block is implicitly selected. The input fields are analogously significant to the input parameters of the function block:

Input field in the dialog	Input parameters of the function block "ML_SetBeltConfiguration"
"Axis" section	ConveyorBelt
Belt/starting length	BeginLength
Belt/total length	TotalLength
"Coordinate offset" area	ConveyorBeltOrigin

Tab. 3-5: Assigning the input fields in the dialog to the inputs of the "ML_SetBeltConfiguration" function block

Input field in the dialog	Input parameters of the "ML_SetRotaryTableConfiguration" function block
"Axis" section	ConveyorBelt
Belt/starting length	BeginLength
Belt/total length	TotalLength
"Coordinate offset" area	RotaryTableOrigin

Tab. 3-6: Assigning the input fields in the dialog to the inputs of the "ML_SetRotaryTableConfiguration" function block

Input field in the dialog	Input parameters of the "ML_SetBeltErrorReaction" function block
Error reaction/reaction type	ErrorReactionType
Error reaction/desynchronization time	DeSyncTime
Error reaction/deceleration	Deceleration
Error reaction/jerk	Jerk

Tab. 3-7: Assigning the input fields in the dialog to the inputs of the "ML_SetBeltErrorReaction" function block

Interface functions

Product coordinates

Product coordinate systems can be defined in the "Product coordinates" tab. In the "Coordinate" section, the coordinate system to be defined as well as the coordinate type (Cartesian, cylindrical) can be selected. Specify the offset and the rotation of the workpiece coordinate system compared to the world coordinate system in the "Coordinate offset" section. Using the **Write configuration** button, the product coordinate system is configured in the control.

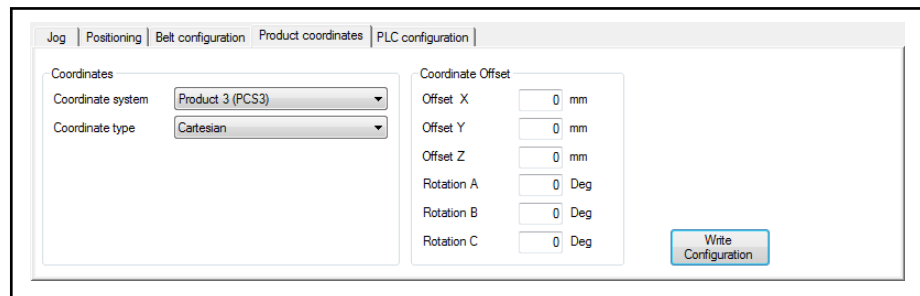


Fig. 3-11: Kinematic commissioning, "Product coordinates" tab

The functionality corresponds to the functionality of the function blocks "MC_SetCartesianTransform" and "ML_SetCylindricTransform". By selecting the coordinate type, the relevant function block is implicitly selected. The input fields are analogously significant to the input parameters of the function block:

Input field in the dialog	Input parameters of the "MC_SetCartesianTransform" and "ML_SetCylindricTransform" function blocks
Coordinates/coordinate systems	ConveyorBelt
"Coordinate offset" area	ConveyorBeltOrigin
Coordinate offset/offset X	TransX
Coordinate offset/offset Y	TransY
Coordinate offset/offset Z	TransZ
Coordinate offset/rotation X	RotAngle1
Coordinate offset/rotation Y	RotAngle2
Coordinate offset/rotation Z	RotAngle3

Tab. 3-8: Assigning the input fields in the dialog to the inputs of the "MC_SetCartesianTransform" and "ML_SetCylindricTransform" function blocks

PLC configuration

After a control restart, the configuration of belt coordinate systems and product coordinate systems is lost. To avoid reconfiguration, use the "PLC configuration" tab to continuously save the configuration in a function block. Thus, by calling the function block during the initialization phase of a application, the configuration can be restored quickly and easily.

There is the option to backup all or only the currently configured belts and product coordinate systems. In the "Product coordinate systems", select one of the two options. Using the **Generate PLC block** button, the function block is generated.

Interface functions

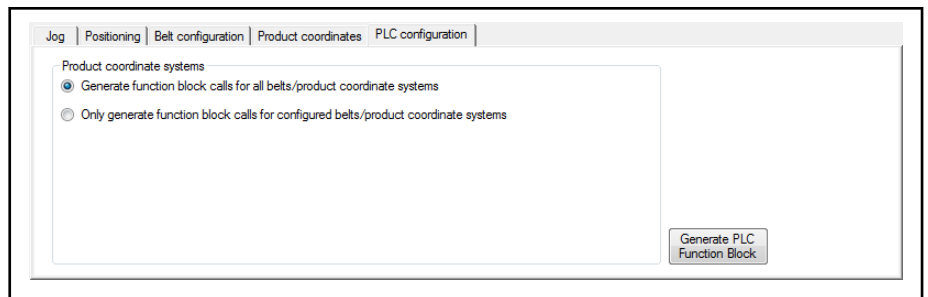


Fig. 3-12: Kinematic commissioning, "PLC configuration" tab

Press **Generate PLC function block** to create a new folder called "KinematicsInitialConfig" below the application node. The function block is created in this folder. The function block name is composed of the kinematic name and the suffix "...InitialConfig". For each configured kinematics, a corresponding function block can be generated which configures the belt and product coordinate systems. If a function block already exists for the selected kinematics, it is replaced.

3.2.4 PathViewer Overview

The "PathViewer" dialog in the context menu of the robot node provides project planning support for kinematics applications. This way, all configured kinematics can be illustrated as 3D rods and spheres with any perspective. This is particularly useful in the project planning phases of the PLC program if a real system is not available and the kinematics are to be moved for testing.



This dialog can only be selected in online mode. Furthermore, kinematics can only be displayed in communication phase P4 (BB).

Transmission of data between control and PathViewer is realized via Ethernet connection. For this reason, real-time cannot be ensured. Accuracy of the TCP track cannot be guaranteed and may not comply precisely with the path curve applied. In case of respective utilization of the connection or excessive belt velocities, this deviation may be extreme.

The PathViewer can be accessed via the context menu of the robot node or the "Robot" folder in the IndraWorks main menu. After opening, all kinematics configured in the control and their coordinate systems are displayed spatially.



User-defined kinematics (transformer number 100 or 111) are only displayed if the mechanical information in method "GetMechanic" (refer to [Mechanical information on page 180](#)) is available.

The following figure illustrates a representation example of delta, Scara and parallelogram.

Interface functions

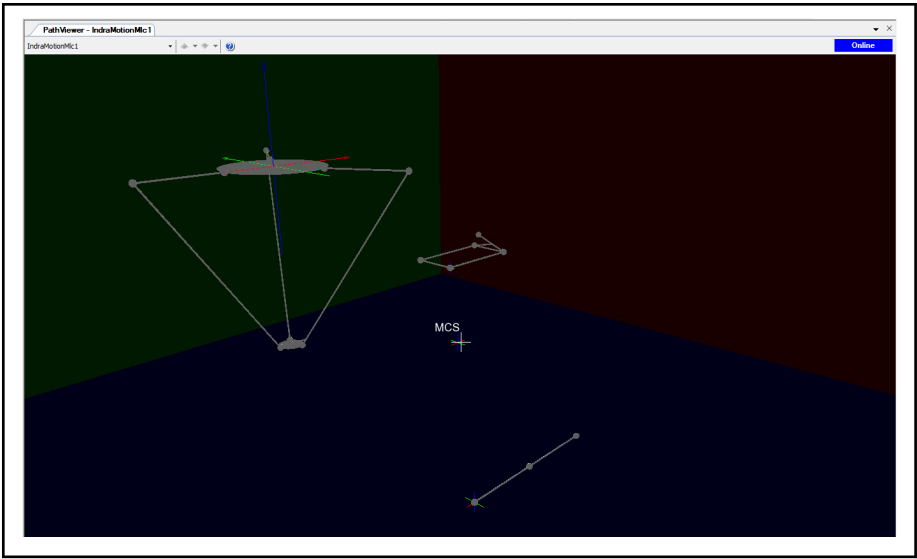
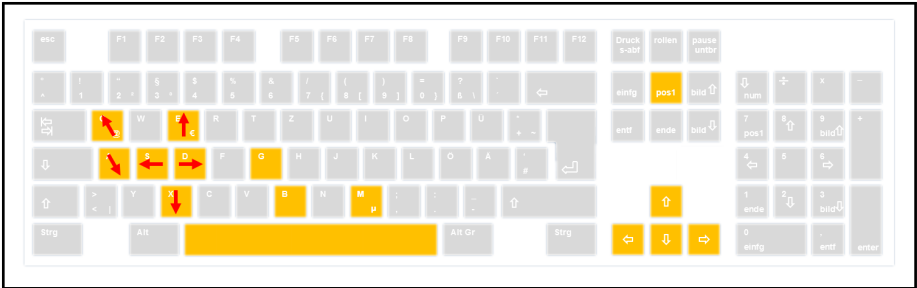


Fig. 3-13: PathViewer dialog

Navigation and operation

In PathViewer, motions are applied by interaction between mouse and keyboard while operation is also possible with only one of the two input devices.

The following overview is to illustrate spatial navigation in PathViewer using a keyboard.



Q	Move forward
A	Move backward
S	Move left
D	Move right
E	Move up
X	Move down
G	Move to kinematics displayed in the cross
B	Show the limits of individual elements
pos1	Move to MCS
Arrow left	Turn left
Arrow right	Turn right
Arrow up	Turn down
Arrow down	Turn up
M	Increase view motion velocity
Space bar	Decrease view motion velocity

Fig. 3-14: PathViewer keyboard layout

Besides keyboard operation, the mouse can also be used to realize spatial motions.

The following table illustrates the reaction on mouse movements in connection with different actions.

Reaction to mouse actions

Action	Reaction	Keyboard
Holding left mouse button and drawing	Turning the view (3D)	Arrow keys
Holding right mouse button and drawing	Moving the view (2D)	S, D, E, X
Turning mouse wheel forward	Move forward	Q
Turning mouse wheel backward	Move backward	A
Left click on kinematics	Move to kinematics BCS	G
Moving the mouse cursor onto kinematics	The kinematics name is displayed in the tooltip	
Right click	Opening the PathViewer context menu	

Tab. 3-9: PathViewer operation by mouse

The two keyboard keys "Shift" and "Ctrl" also have an effect on the scrolling speed of the mouse wheel.

PathViewer context menu

By right-clicking with the mouse onto the PathViewer interface, the context menu is opened. It contains the following menu items:

- **Position:** Here, the view can be saved, restored or centered on an existing kinematics
- **Recording:** Here, kinematics motions can be recorded
- **Delete TCP tracks:** All TCP tracks are deleted
- **Settings...:** Opens the PathViewer options

Recording

Under the "Recordings" menu item, the motions of the existing kinematics can be recorded similarly to a video for later viewing. Recording is started via the "Start recording" submenu item. If this point is active, the motions of the kinematics are stored in temporary memory. The maximum duration for recording is limited to 10 minutes. If this limit is reached, the running recording is automatically stopped.

By repeated operation of the "Start recording" submenu item, it is deactivated and recording stopped. The data is available in temporary memory until recording is started again or PathViewer is closed.

Recorded motions can be replayed via the respective "View recording" submenu item. Operation of this submenu item opens a new window to control the replay. The recording may be started, stopped and manually controlled using a slider as often as required. Operation of this menu item during recording automatically stops the recording process.

Settings

The "Settings" menu item offers several options for PathViewer.

The following table describes the individual options in the "General" tab.

Options in "General" tab

Function	Meaning
TCP track	

Interface functions

	Display TCP track	By enabling/disabling this checkbox, display of the TCP track is enabled/disabled.
	Length of the TCP track	Here, the maximum length of the displayed TCP track can be controlled. The time specified is the display time of a point of the TCP track. Older points are also deleted while the kinematics are at a standstill!
	Display TCP track as points	If the checkbox is disabled, the TCP track is displayed as linear interpolated line between the individual points. As the points are transmitted via Ethernet connections and not in real time, the TCP track displayed as line may not completely comply with the moved path curve. By enabling this checkbox, the TCP track is displayed as a point cloud without interpolation. Point sizes can be individually adjusted by specification. This display type offers the advantage that transferred points exactly represent the position of the TCP. This way, the path curve can be precisely analyzed in case of quick periodic motion of the TCP.
Keyboard control		
	Mirror Y-axis	By enabling/disabling this checkbox, the functions of the keys "Arrow up" and "Arrow down" are switched.
Space size		
	Enter space size manually	If the checkbox is disabled, the space size is automatically calculated from the distances between the kinematics. If only one kinematic is available, a pre-defined space size is generated. By enabling this checkbox, the space size can be specified manually. The extension of the coordinate systems is also adjusted automatically.

Tab. 3-10: "Settings" context menu in PathViewer

In the second tab "Colors", the colors of the TCP track, the robot, the walls and the cross can be set.

3.3 Kinematic folder

3.3.1 Kinematic folder, general information

Each kinematics created in the "Robot" folder has a separate folder with identical name in the Project Explorer with

- Context menu, see [Kinematics – Context menu, page 98](#) and
- Dialogs [Kinematics – Dialogs, page 104](#)

3.3.2 Kinematics – Context menu

Kinematics – Context menu or context menu item "Kinematics", general information

There are the following menu items in the context menu of a kinematics and in the main menu item **Kinematics**:

- **Initial commissioning** - Commissioning cycle of the kinematics, see [Initial commissioning, page 99](#)
- **Parameter** - Editing parameters

Interface functions

- **Diagnostics** - Diagnostic functions of the kinematics, see [Diagnostics, page 102](#)
- **Export...** - Saves Robot elements to a file
- **Import...** - Imports Robot elements from a file
- **Compare...** - Compares and merges objects

The following menu items are **only** available 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...** - Provides a printout of the kinematic data
- **New view** - Displays the kinematics in an individual window
- **Properties...** - Provides information on the kinematics (such as name or type)

Initial commissioning

The initial commissioning of a kinematics is supported by a wizard in the following steps:

1. Load basic parameters

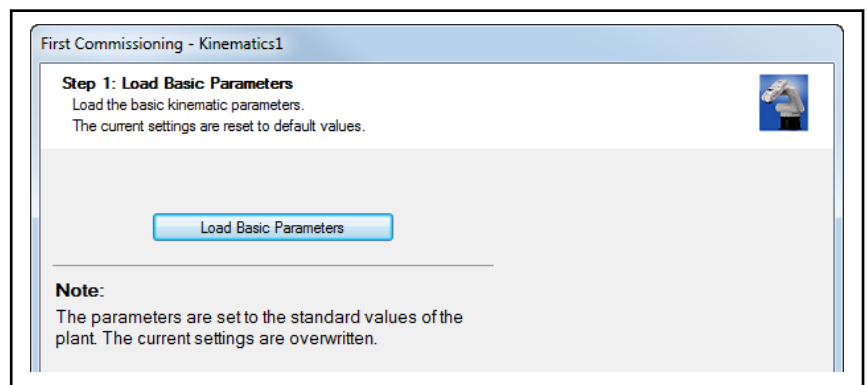


Fig. 3-15: Load basic parameters of the kinematics dialog

The parameters are set to factory default values. The current values are overwritten.

On this basis, the settings can be selected in the following steps.

Interface functions

2. Scaling/units

First Commissioning - Delta_virt

Step 2: Scaling / Units
Specify the scalings and measuring units of the kinematics.

Measuring unit for kinematic data

☒ Scale kinematics in millimeter [mm]
☐ Scale kinematics in inch [in]

Scaling of the assigned axes

Axis Name	Scaling of Position Data	Format of Position Data
vAxis1	Rotary	Absolute
vAxis2	Rotary	Absolute
vAxis3	Rotary	Absolute
vAxis4	Rotary	Modulo

Adjust Axis Scalings

Fig. 3-16: Scaling/units dialog of kinematics

The following parameters are configured:

- K-0-0054, Measuring unit for kinematic data
- A-0-0053, Scaling type for acceleration data
- A-0-0056, Scaling type for velocity data
- A-0-0059, Scaling type for position data

By means of these parameters, the unit for kinematics data is defined and the scaling of all axes assigned to kinematics adjusted.

Also refer to [Scaling/units on page 104](#).

3. Axis length and coupling

First Commissioning - Kinematics1

Step 3: Axis length and coupling
Specify the axis lengths and the axis couplings.

Achslängen			Achskopplungen	
Name	Wert	Einheit	Name	Wert
L1	0	mm	K1	0
L2	0	mm	K2	0
L3	0	mm	K3	0
L4	0	mm	K4	0
L5	0	mm	K5	0
L6	0	mm	K6	0
L7	0	mm	K7	0
L8	0	mm	K8	0
L9	0	mm	K9	0
L10	0	mm	K10	0
L11	0	mm	K11	0
L12	0	mm	K12	0
L13	0	mm	K13	0
L14	0	mm	K14	0
L15	0	mm	K15	0
L16	0	mm	K16	0

Kinematics type

Diagram showing a delta robot configuration with axes A1, A2, A3 and TCP. The diagram includes labels for lengths L1, L2, L3, L4, L5 and angles +A1, +A2, +A3.

Fig. 3-17: Axis length and kinematic coupling dialog

The following parameters are configured:

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

These parameters are required to transform axis positions to Cartesian coordinates.

Also refer to [Axis lengths and couplings](#) on page 105.

4. Motion limit values

First Commissioning - Kinematics1

Step 4: Motion Limit Values
Specify the values for motion limits.

Maximum path velocity: mm/s

Maximum path acceleration: mm/s²

Maximum path jerk: 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. 3-18: Motion limit values of the kinematics dialog

The following parameters are configured:

- 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

These parameters are required to define the motion limit values of the kinematics.

See also [Motion limit values](#), page 105.

5. Coordinate offset

First Commissioning - Kinematics1

Step 5: Coordinate Offset
Specify the values for offset or twisting of the coordinate system.

Linear Offset

X: mm

Y: mm

Z: mm

Rotation

A: Deg

B: Deg

C: Deg

Fig. 3-19: Coordinate offset of the kinematics dialog

The following parameter is configured:

- K-0-0042, Offset of BCS system

Interface functions

Use this parameter to rotate and offset the basic coordinate system (BCS) of a kinematics with regard to the machine coordinate system (MCS).

Also refer to [Coordinate offset on page 106](#).

Diagnostics

This dialog graphically displays the status information of the respective kinematics.

The upper part of the window displays general information on the kinematics. The center part displays the kinematic-specific information on all axes of a kinematics in a table.

The information contains

- The offset and rotation of the selected coordinate system to the machine coordinate system
- Current position and programmed end position in the selected coordinate system

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

The screenshot shows the 'Kinematics2' diagnostics window. At the top, it displays 'Kinematics number: 2', 'Kinematics name: Kinematics2', and 'Kinematics type: Cartesian'. Below this, the 'State' is 'Kinematik ist in "StandStill Incomplete"'. The 'Coordinate systems' section shows 'Machine (MCS)' selected. The 'Coordinate Offset to MCS' table shows offsets for X, Y, Z, A, B, and C axes, all set to 0. The main table lists axes 7 through 12 with their coordinate names, positions, end positions, and units. The 'Motion' section shows a yellow warning box: 'Kinematics is in state "Standstill Incomplete" Not all axes are assigned to the kinematics'. At the bottom, there are three panels: 'Axes' (with radio buttons for 'All axes ready for operation', 'Target position reached', 'All kinematic axes at standstill', and 'Homing required'), 'Errors/Warnings' (with radio buttons for 'Warning exists', 'Error exists', and 'Error reaction is enabled'), and 'Messages' (with radio buttons for 'Blending', 'Belt-synchronous', 'Calculating spline', and 'Spline is valid').

Kinematics number	Kinematics name	Kinematics type
2	Kinematics2	Cartesian

State: Kinematik ist in "StandStill Incomplete"

Coordinate systems: Machine (MCS)

Coordinate Offset to MCS						
X	Y	Z	A	B	C	Type
0	0	0	0	0	0	Cartesian

Axis No.	Coordinate name	Position	End position	Unit
7	Coordinate 1	0	0	mm
8	Coordinate 2	0	0	mm
9	Coordinate 3	0	0	mm
10	Coordinate 4	0	0	Grad
11	Coordinate 5	0	0	Grad
12	Coordinate 6	0	0	Grad

Motion: Kinematics is in state "Standstill Incomplete" Not all axes are assigned to the kinematics

Axes

- ☐ All axes ready for operation ("AH" or "AF")
- ☒ Target position reached
- ☒ All kinematic axes at standstill
- ☐ Homing required

Errors/Warnings

- ☐ Warning exists
- ☐ Error exists
- ☐ Error reaction is enabled

Messages

- ☐ Blending
- ☐ Belt-synchronous
- ☐ Calculating spline
- ☐ Spline is valid

Fig. 3-20: Diagnostics of a kinematics

Coordinate systems

The setting under "Coordinate systems" specifies the coordinate system in which the position values are to be displayed.

The following systems can be selected: Machine coordinate system (MCS), Axis coordinate system (ACS), Basic coordinate system (BCS) as well as any configured product coordinate system (PCS):

Interface functions

- **Machine coordinates (MCS)** = Displays the coordinates in space related to the machine zero point

Coordinate systems		Coordinate Offset to MCS						
Machine (MCS)		X	Y	Z	A	B	C	Type
		0	0	0	0	0	0	Cartesian
No.	Coordinate name	Position		End position		Unit		
1	Coordinate 1	0				0 mm		
2	Coordinate 2	0				0 mm		
3	Coordinate 3	0				0 mm		
4	Coordinate 4	0				0 Grad		
5	Coordinate 5	0				0 Grad		
6	Coordinate 6	0				0 Grad		

Fig. 3-21: Display of the machine coordinates (MCS)

- **Axis coordinates (ACS)**= Displays the positions of the individual axes of the kinematics

Coordinate systems		Coordinate Offset to MCS						
Axis (ACS)		X	Y	Z	A	B	C	Type
		--	--	--	--	--	--	--
Axis No.	Axis name	Position		End position		Unit		
7	Kart_X	0				0 mm		
8	Kart_Y	0				0 mm		
9	Kart_Z	0				0 mm		
10	Kart_C	0				0 Grad		
11	Kart_Bi	0				0 Grad		
12	Kart_Bo	0				0 Grad		

Fig. 3-22: Display of the axis coordinates (ACS)

- **Basic coordinate system (BCS)**= Displays the position in space in relation to the basis of the kinematics

Coordinate systems		Coordinate Offset to MCS						
Basic (BCS)		X	Y	Z	A	B	C	Type
		0	0	0	0	0	0	Cartesian
No.	Coordinate name	Position		End position		Unit		
1	Coordinate 1	0				0 mm		
2	Coordinate 2	0				0 mm		
3	Coordinate 3	0				0 mm		
4	Coordinate 4	0				0 Grad		
5	Coordinate 5	0				0 Grad		
6	Coordinate 6	0				0 Grad		

Fig. 3-23: Display of the basic coordinates (ACS)

- **Product coordinate system (BCS)**= Displays the position in space in relation to the zero point of the belt or product

Interface functions

Coordinate systems

Product 1 (PCS1)

Coordinate Offset to MCS

X	Y	Z	A	B	C	Type
15	10	0	0	0	0	Cartesian

No.	Coordinate name	Position	End position	Unit
1	Coordinate 1	-15	-15	mm
2	Coordinate 2	-10	-10	mm
3	Coordinate 3	0	0	mm
4	Coordinate 4	0	0	Grad
5	Coordinate 5	0	0	mm
6	Coordinate 6	0	0	Grad

Fig. 3-24: Display of the product coordinates (PCS)

List of axis coordinates

The list of the coordinates shows:

- Number of axis/coordinate
- Name of axis/coordinate
- Current position of the axis/coordinate
- Programmed end position of the axis/coordinate

3.3.3 Kinematics – Dialogs

Overview on dialogs

There are the following individual dialogs when opening the "Kinematics" folder in the Project Explorer.

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

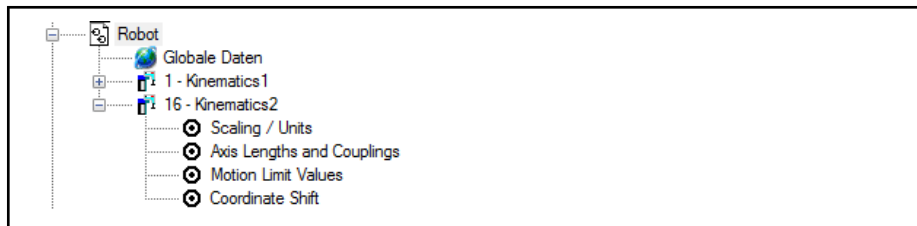


Fig. 3-25: Kinematic dialogs (Robot Control V2)

Overview on dialogs

- [chapter "Scaling/units" on page 104](#)
- [chapter "Axis lengths and couplings" on page 105](#)
- [chapter "Motion limit values" on page 105](#)
- [chapter "Coordinate offset" on page 106](#)

Scaling/units

Use this dialog to toggle between the units millimeter [mm] and inch [in]. The axis scalings of all axes assigned to the kinematics can be adjusted in one step. The adjustments depend on the transformation used.



Fig. 3-26: Toggling measuring units and axis scalings

The following parameters are evaluated:

- [K-0-0054](#), Measuring unit for kinematic data
- A-0-0053, Scaling type for acceleration data
- A-0-0056, Scaling type for velocity data
- A-0-0059, Scaling type for position data

Axis lengths and couplings

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

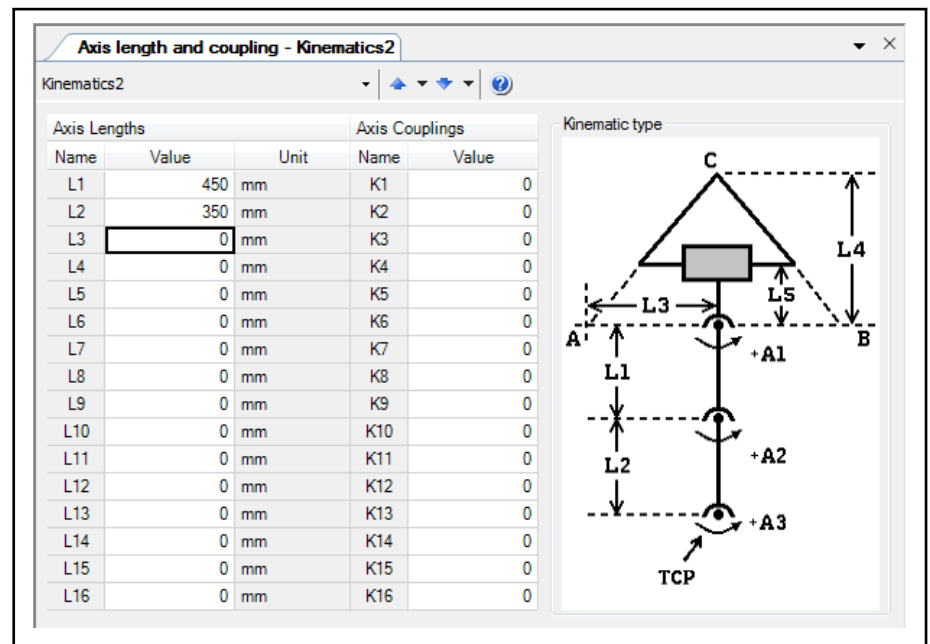


Fig. 3-27: Defining axis lengths and axis couplings

The following parameters are evaluated:

- [K-0-0040](#), Transformation, lengths of the mechanics
- [K-0-0041](#), Transformation, factors

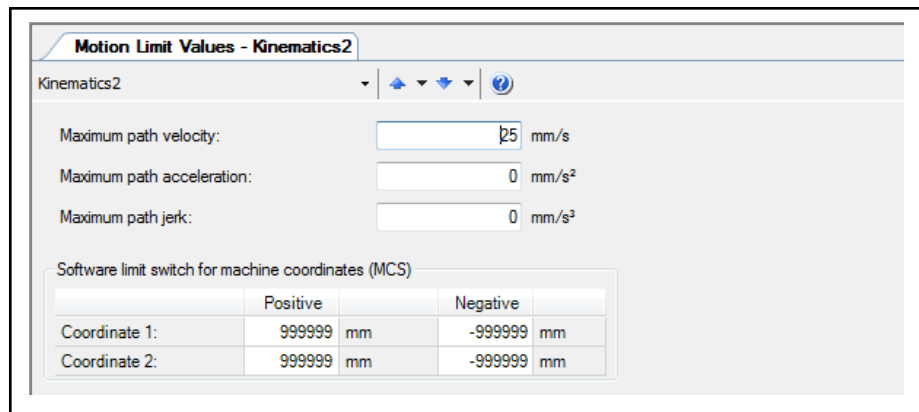
Motion limit values

The following limit values can be entered:

- Maximum path velocity
- Maximum path acceleration (bipolar)
- Maximum path jerk (bipolar)
- Software limit switch, positive, MCS
- Software limit switch, negative, MCS

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

Interface functions



Motion Limit Values - Kinematics2

Kinematics2

Maximum path velocity: mm/s

Maximum path acceleration: mm/s²

Maximum path jerk: mm/s³

Software limit switch for machine coordinates (MCS)

	Positive	Negative
Coordinate 1:	999999 mm	-999999 mm
Coordinate 2:	999999 mm	-999999 mm

Fig. 3-28: Motion Limit Values dialog

The following parameters are evaluated:

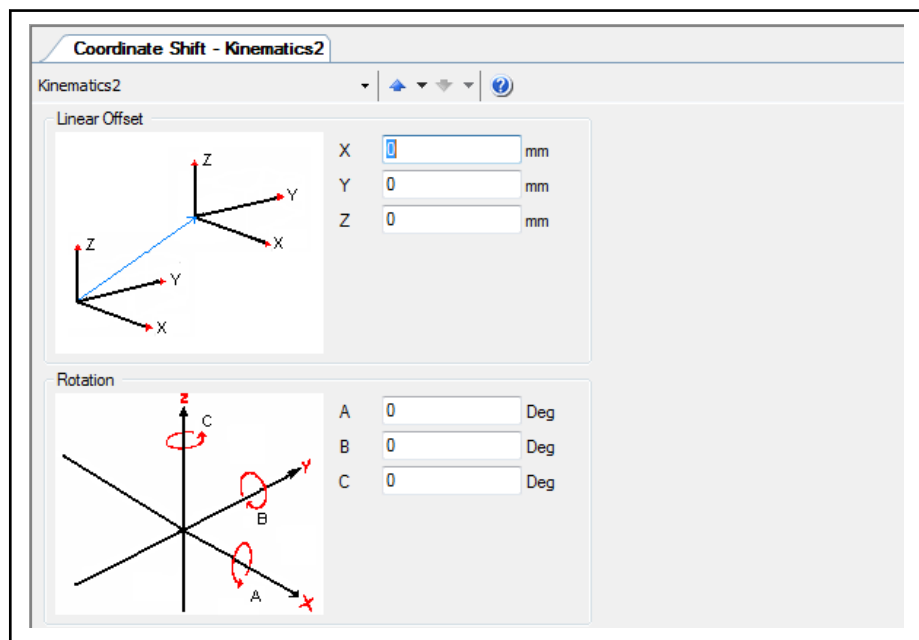
- [K-0-0003](#), Maximum path velocity
- [K-0-0034](#), Maximum path acceleration (bipolar)
- [K-0-0035](#), Maximum path jerk (bipolar)
- [K-0-0005](#), Software limit switch, positive, MCS
- [K-0-0006](#), Software limit switch, negative, MCS

Coordinate offset

Use this dialog to rotate and shift the basic coordinate system (BCS) of a kinematics with regard to the machine coordinate system (MCS).

The values specified are thus position and rotation of a kinematics in space.

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



Coordinate Shift - Kinematics2

Kinematics2

Linear Offset

X: mm

Y: mm

Z: mm

Rotation

A: Deg

B: Deg

C: Deg

Fig. 3-29: Overview Coordinate Shift

The following parameter is evaluated:

- [K-0-0042](#), Offset of BCS 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 28](#)).

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 108
7	2-axis delta with 1 optional orientation axis on page 109
8	4-axis Scara with linear lifting axis on page 111
9	3-axis Scara with 1 optional orientation axis on page 115
10	4-axis Scara with 1 optional orientation axis on page 119
12	2-axis parallelogram on page 123
13	Inverted 2-axis parallelogram on page 125
14	2-axis press on page 126
18	Contact kinematics on page 128
19	2-axis parallelogram V1 on page 129
20	3-axis delta with 1 optional orientation axis on page 131
21	2-axis parallelogram V2 on page 133
22	3-axis delta linear with 1 optional orientation axis on page 134
23	5-axis delta on page 137
30	Adabot on page 141
32	2-axis delta with asymmetric TCP on page 144
38	3-Axis Twisting Crane on page 145

Transformation schemes

Transformation scheme	Name
39	3-axis flex stamper on page 147
40	3-axis delta with linear axes with 1 optional orientation axis on page 149
45	2-axis delta slide on page 152
46	3-Axis Parallelogram V3 on page 153
47	6-axis Cartesian on page 154
56	3-axis delta with 90 degree angles on page 156
61	4-axis speedpicker on page 159
66	Stewart platform on page 162
73	4-axis Scara with crank on page 166
81	3-axis Canfield on page 170
100	User-defined transformation on page 174
111	Extended user-defined transformation on page 176

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

4.2 Transformation schemes

4.2.1 Cartesian

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

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 36](#) are identical to [Move Linear motions on page 36](#) in relation to the path motion.

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

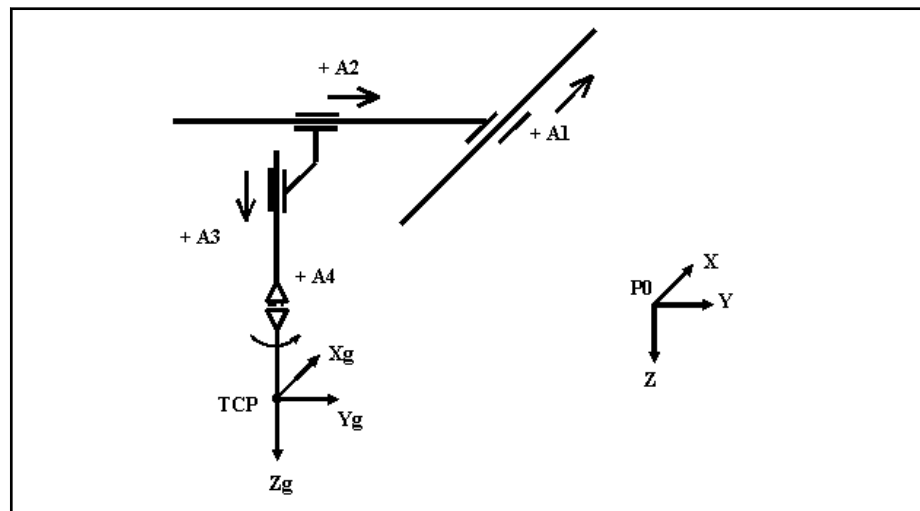


Fig. 4-1: Schematic view of the Cartesian

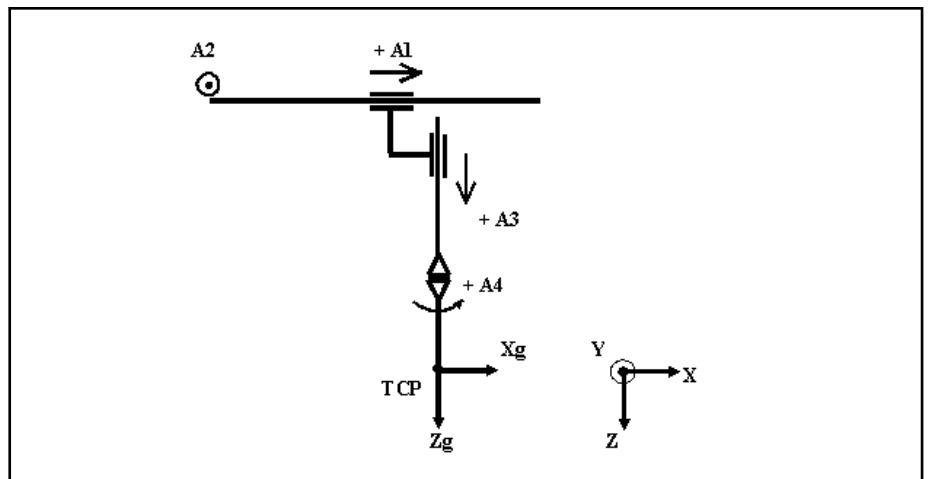


Fig. 4-2: Page view: A typical X-Y-Z-C Cartesian system

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.

Coordinates of a point

Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

4.2.2 2-axis delta with 1 optional orientation axis

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

Transformation schemes

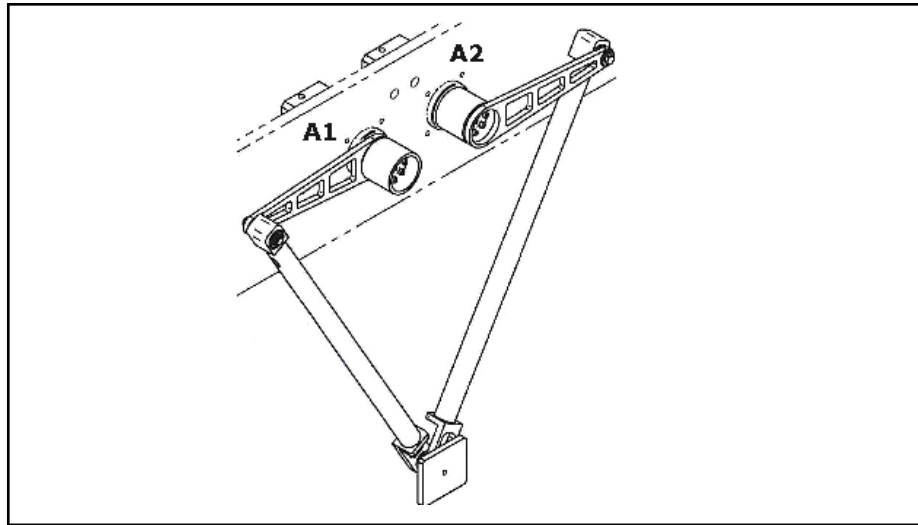


Fig. 4-3: Robot type 07, 2-axis delta

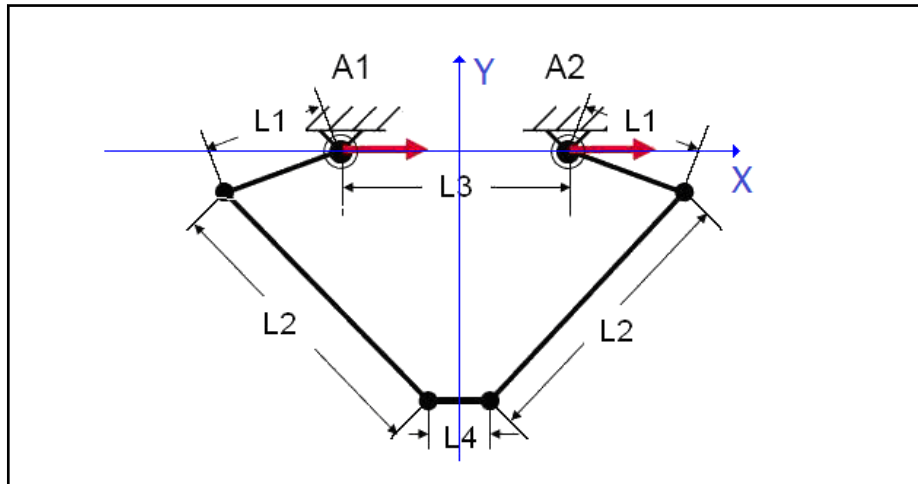


Fig. 4-4: Schematic view of 2-axis delta (X-Y view)

Axis directions of rotation and zero positions

fig. 4-3 "Robot type 07, 2-axis delta" on page 110 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-Y view)" on page 110).

- The third axis rotates around the Z-axis. The zero point for the third axis (orientation axis) is in the positive direction of the X-axis
- The arm length L_1 is measured from the rotation center point to the center point of the rod holders
- 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 die plate measured at the rod mountings
- Positive counterclockwise rotation of the positioning axes A1 and A2
- The Cartesian zero point of the bipod robot is in the center of both arm rotation points A1 and A2

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Transformation schemes

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+ 90 .. 360 degrees
A2:	Positioning axis	Rotary absolute	+ 90 .. -180 degrees
A3:	Orientation axis	Rotary abs./modulo	optional

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of the rod to the die 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

4.2.3 4-axis Scara with linear lifting axis

To use this transformation, enter transformation number 8 in parameter "K-0-0031, Transformation scheme".

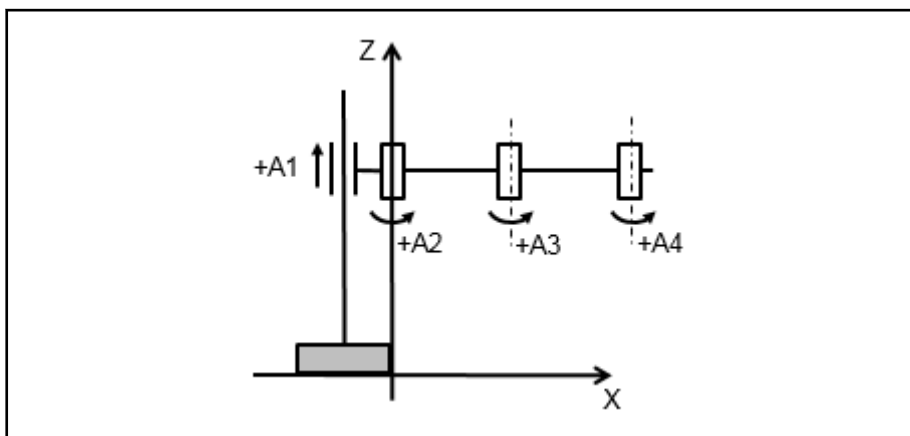


Fig. 4-5: Side view 4-axis Scara with linear distance

Transformation schemes

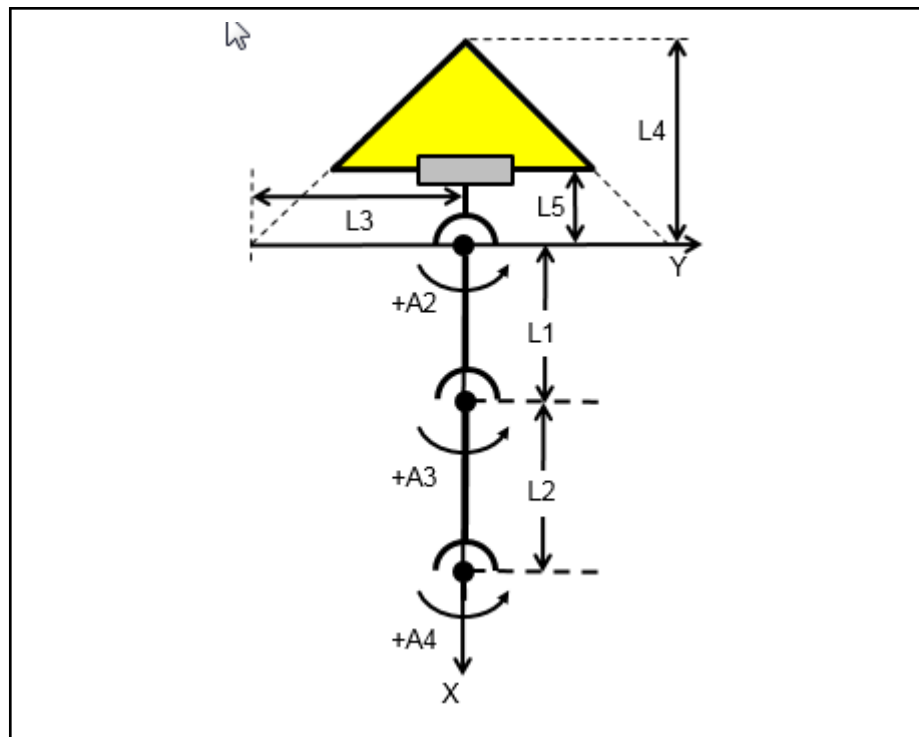


Fig. 4-6: Top view 4-axis Scara with linear distance

Axis directions of rotation and zero positions

- If the delta robot assumes the position displayed in the figures, the machine coordinates (MCS) for all of the axes have to be zero. This has to be ensured by an appropriate setting of the actual values of the reference point
- In the zero position, axes 2, 3 and 4 are extended
- The zero point of the first axis can be freely selected
- The figures also show arm lengths L_1 , L_2 and the positive rotary directions of the axes
- Using the lengths L_3 , L_4 , L_5 , a prohibited area can be defined in which the Tool Center Point (TCP) cannot enter.
- This is done to avoid collisions if the traversing ranges of the axes 2, 3 are very large so that the robot could potentially collide with its own support.
This forbidden area is highlighted in yellow in the top view
- If the length $L_3=0$ is active, no area monitoring is active
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive values, whereas L_1 , L_2 have to be unequal to zero
- K_1 specifies the mechanic coupling of the axis A_2 with the axis A_3
Such a coupling exists if the axis A_3 moves along when traversing the axis A_2 without the measuring system of axis A_3 detecting this motion. This is generally the case if the motor for the axis A_3 is not directly on the axis, but instead on its support. The coupling factor K_1 is inversely defined to the coupling factor K_2 .

Example:

Coupling factor K_1

$K_1 = 0$: A motion of axis A_2 affects the position of the second arm in space: The angle between the first arm and the second arm does not change if the axis A_2 is moved. The position of the second arm changes analogously to the motion of axis 1 with regard to the x-axis.

$K_1 = 1$: A motion of axis A_2 does not affect the position of the second arm in space: The angle between the first arm and the second arm changes analogously to the motion of axis A_2 . The position of the second arm remains constant with regard to the x-axis.

- K_2 specifies the mechanic coupling of the axes A_2 and A_3 with the axis A_4 .

Such a coupling exists if the axis A_4 moves along when traversing the axes A_2 and A_3 without the measuring system of axis A_4 detecting this motion. This is generally the case if the motor for the axis A_4 is not directly on the axis, but instead on its support.

Example:

Coupling factor K_2

$K_2 = 0$: A motion of axis A_2 or axis A_3 affects the orientation in space. Axis A_4 has to compensate the motion.

$K_2 = 1$: A motion of axis A_2 or axis A_3 does not affect the orientation in space. Axis A_4 does not have to compensate the motion.



The maximum traversing range of the axes A_2 and A_3 may not exceed or fall below +/-179 degrees. The traversing range of the axes A_1 and A_4 is not restricted.

Arm behavior selection for PTP motions

If a motion [Point-to-point on page 41](#) chapter 2.7.4 "Point-to-point motion type" on page 41 is executed and the point to be approached is not specified in the ACS, there are two methods to approach this point.

Transformation schemes

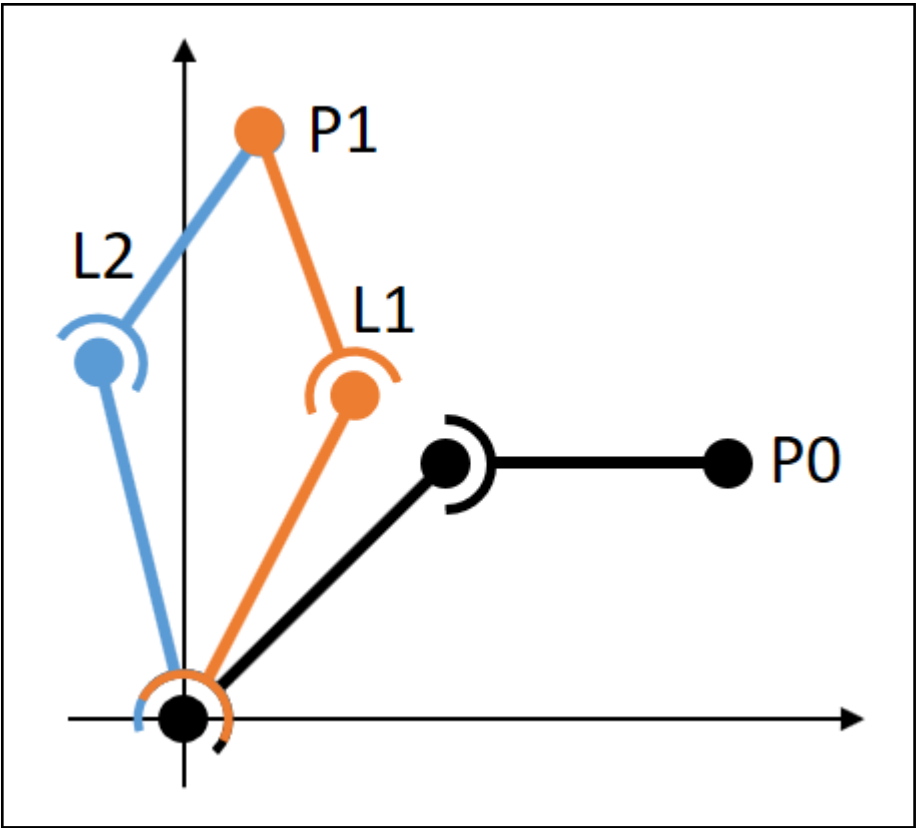


Fig. 4-7: Different arm behaviors of a Scara robot

With a point-to-point motion from point P0 to point P1 that was not specified in the ACS can be approached in two ways (L1 and L2).

The arm selection behavior can be configured using **K₄**:

- 0: When the target point is approached, the arm behavior is not changed at the starting point in time
- 1: When the target point is approached, the motion of the first arm is minimized from the starting to the target point. The arm behavior selected ensures that axis 1 does not exceed the limits of $\pm 179^\circ$

Example:

Configuration₄

- ₄ = 0: Point P1 is approached with L2. The arm behavior does not change during the motion
- ₄ = 1: Point P1 is approached with L2. Thus, the consequence is that L2 has a lower motion of arm 1

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Translatory absolute	
A2:	Positioning axis	Rotary absolute	+/- 179 degrees

Transformation schemes

A3:	Positioning axis	Rotary absolute	+/- 179 degrees
A4:	Orientation axis	Rotary	Optional

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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 zero point to the point of intersection of the triangle with the y-axis	mm or inch (see K-0-0054)	Optional, see (1) ≥0
L4:	Distance from zero point to the tip of the triangle	mm or inch (see K-0-0054)	Optional, see (1) ≥0
L5:	Distance from zero point to the support	mm or inch (see K-0-0054)	Optional, see (1) ≥0

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

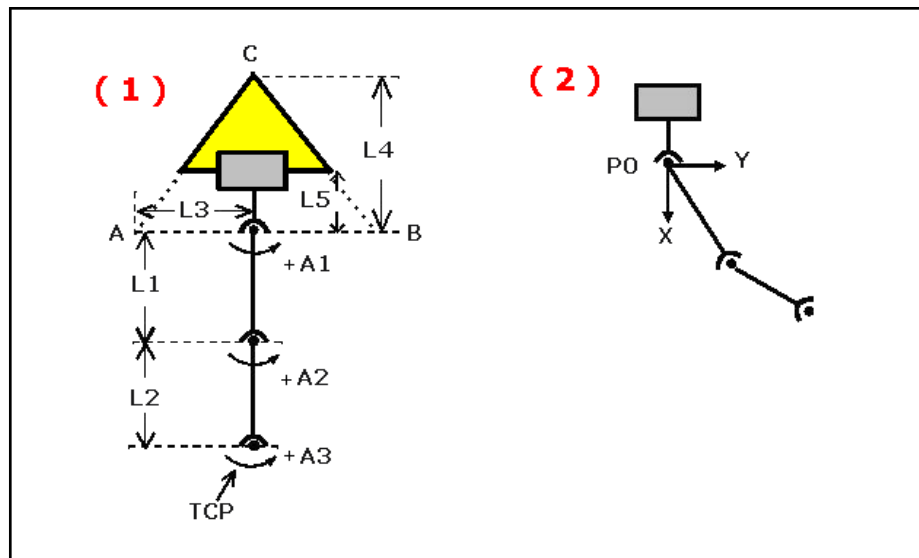
Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Mechanical coupling between axis 2 and 3	Factor	Optional
K2:	Mechanical coupling between the axes 2, 3 and 4	Factor	Optional
K4:	Arm behavior selection for PTP motions and point specification in non-ACS • 0: Arm behavior remains • 1: The arm behavior selected minimizes the motion of the first arm	Factor	Optional

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

4.2.4 3-axis Scara with 1 optional orientation axis

To use this transformation, enter transformation number 9 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes



(1) Top view with zero positions for all axes; yellow: prohibited area

(2) Zero point reference of all axes

Fig. 4-8: Schematic view of 3-axis Scara with 1 optional orientation axis

Axis directions of rotation and zero positions

fig. 4-8 "Schematic view of 3-axis Scara with 1 optional orientation axis" on page 116: (1) displays the zero positions of all axes in a top view.

- If the delta robot assumes the position displayed in the figures, the machine coordinates (MCS) for all of the axes have to be zero. This has to be ensured by an appropriate setting of the actual values of the reference point
- The figures also show arm lengths L_1 , L_2 and the positive rotary directions of the axes
- Using the lengths L_3 , L_4 , L_5 , a prohibited area can be defined in which the Tool Center Point (TCP) cannot enter. This is done to avoid collisions if the traversing ranges of the axes 1, 2 are very large so that the robot could potentially collide with its own support. This forbidden area is highlighted in yellow in (1)
- If the length $L_3=0$ is active, no area monitoring is active
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive values, whereas L_1 , L_2 have to be unequal to zero
- K_1 specifies the mechanic coupling of the axes A_1 and A_2 with the axis A_3 .

Such a coupling exists if the axis A_3 moves along when traversing the axes A_1 and A_2 without the measuring system of axis A_3 detecting this motion. This is generally the case if the motor for the axis A_3 is not directly on the axis, but instead on its support

Example:

Coupling factor K_1

$K_1 = 0$: A motion of axis A_1 or axis A_2 affects the orientation in space. Axis A_3 has to compensate the motion.

Transformation schemes

$K_1 = 1$: A motion of axis A_1 or axis A_2 does not affect the orientation in space. Axis A_3 does not have to compensate the motion.

- K_2 specifies the mechanic coupling of the axis A_1 with the axis A_2 .
 Such a coupling exists if the axis A_2 moves along when traversing the axis A_1 without the measuring system of axis A_2 detecting this motion. This is generally the case if the motor for the axis A_2 is not directly on the axis, but instead on its support. The coupling factor K_2 is inversely defined to the coupling factor K_1 .

Example:

Coupling factor K_2

$K_2 = 0$: A motion of axis A_1 affects the position of the second arm in space: The angle between the first arm and the second arm does not change if the axis A_1 is moved. The position of the second arm changes analogously to the motion of axis A_1 with regard to the x-axis.

$K_2 = 1$: A motion of axis A_1 does not affect the position of the second arm in space: The angle between the first arm and the second arm changes analogously to the motion of axis A_1 . The position of the second arm remains constant with regard to the x-axis.



The maximum traversing range of the axes A_1 und A_2 may not exceed or fall below ± 179 degrees. The travel range of the third axis is not limited.

The respective base coordinate system (BCS) is identified in [fig. 4-8 "Schematic view of 3-axis Scara with 1 optional orientation axis" on page 116](#) by the two main axes (x, y).

The zero point of the basic coordinates can be found in the base point P_0 of the robot.

Arm behavior selection for PTP motions

If a motion [Point-to-point on page 41](#) [chapter 2.7.4 "Point-to-point motion type" on page 41](#) is executed and the point to be approached is not specified in the ACS, there are two methods to approach this point.

Transformation schemes

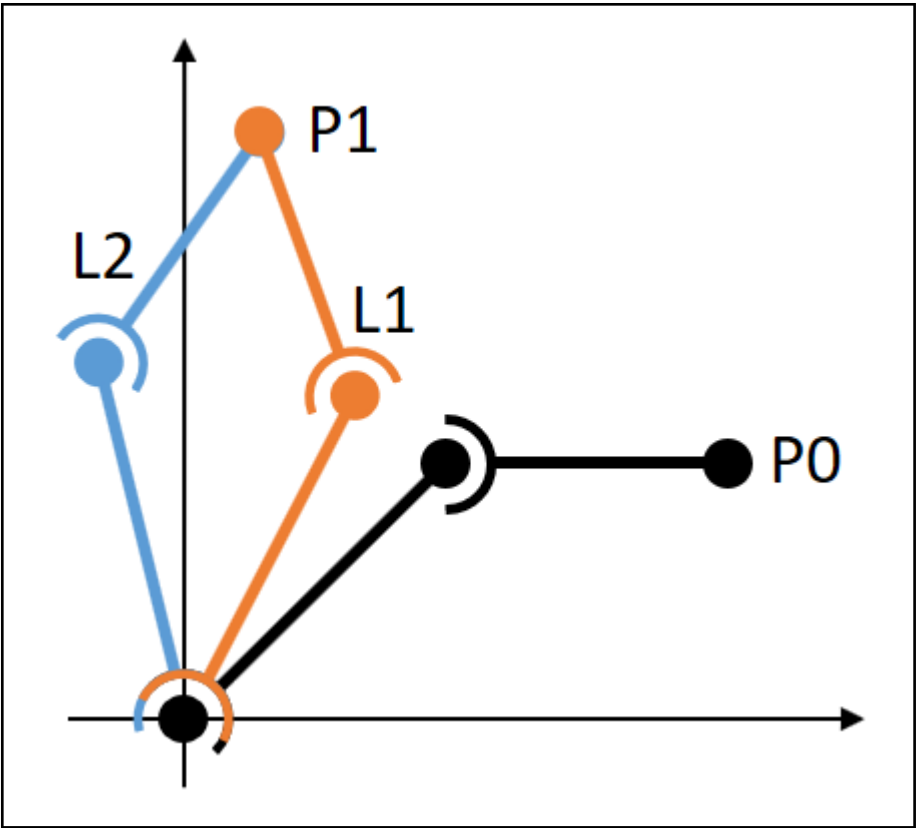


Fig. 4-9: Different arm behaviors of a Scara robot

With a point-to-point motion from point P0 to point P1 that was not specified in the ACS can be approached in two ways (L1 and L2).
The arm selection behavior can be configured using **K₄**:

- 0: When the target point is approached, the arm behavior is not changed at the starting point in time
- 1: When the target point is approached, the motion of the first arm is minimized from the starting to the target point. The arm behavior selected ensures that axis 1 does not exceed the limits of $\pm 179^\circ$

Example:

Configuration₄

- **₄** = 0: Point P1 is approached with L2. The arm behavior does not change during the motion
- **₄** = 1: Point P1 is approached with L2. Thus, the consequence is that L2 has a lower motion of arm 1

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+/- 179 degrees

Transformation schemes

A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Orientation axis	Rotary	Optional

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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 zero point to the point of intersection of the straight line AC	mm or inch (see K-0-0054)	>=0
L4:	Distance from zero point to the tip C of the triangle	mm or inch (see K-0-0054)	>=0
L5:	Distance from zero point to straight line AB	mm or inch (see K-0-0054)	>=0

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

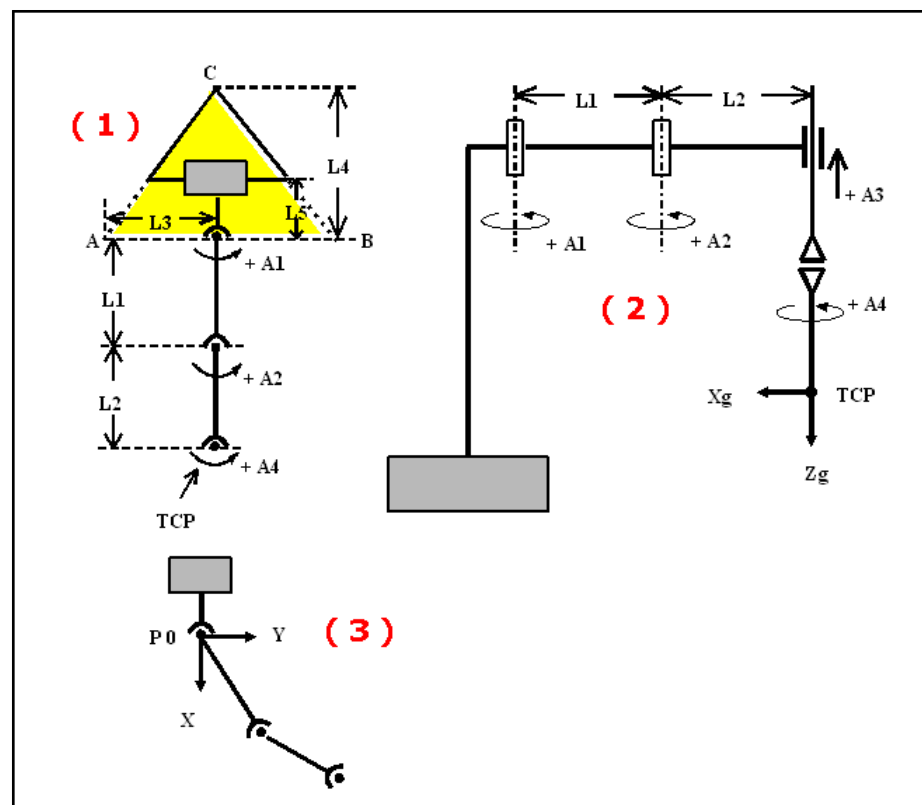
Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Mechanical coupling between the axes 1, 2 and 3	Factor	Optional
K2:	Mechanical coupling between axis 1 and 2	Factor	Optional
K4:	Arm behavior selection for PTP motions and point specification in non-ACS • 0: Arm behavior remains • 1: The arm behavior selected minimizes the motion of the first arm	Factor	Optional

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

4.2.5 4-axis Scara with 1 optional orientation axis

To use this transformation, enter transformation number 10 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes



- (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. 4-10: Schematic view of 4-axis Scara with 1 optional orientation axis

Axis directions of rotation and zero positions

fig. 4-10 "Schematic view of 4-axis Scara with 1 optional orientation axis" on page 120: (1) displays the zero positions of all axes in top view and (2) in side view.

- If the delta robot assumes the position displayed in the figures, the machine coordinates (MCS) for all of the axes have to be zero. This has to be ensured by an appropriate setting of the actual values of the reference point.
 - In the zero position, axes 1, 2 and 4 are extended to the front.
 - The zero point for the third axis can be selected freely.
 - 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 is done to avoid collisions if the traversing ranges of the axes 1, 2 are very large so that the robot could potentially collide with 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 the axes 1, 2 moves along when traversing the axis 4 without the measuring system of the fourth axis detecting 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 can consider a mechanical coupling between the fourth axis and the third axis.

Such a coupling exists if the third coordinate (Z-axis) changes when axis 4 traverses without the measuring system of the third axis detecting this motion.

This is for example the case if the axes 3 and 4 are a lifting spindle and a rotating spindle.

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 of the basic coordinates can be found in the base point P_0 of the robot.

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

Arm behavior selection for PTP motions

If a motion [Point-to-point on page 41](#)chapter 2.7.4 "Point-to-point motion type" on page 41 is executed and the point to be approached is not specified in the ACS, there are two methods to approach this point.

Transformation schemes

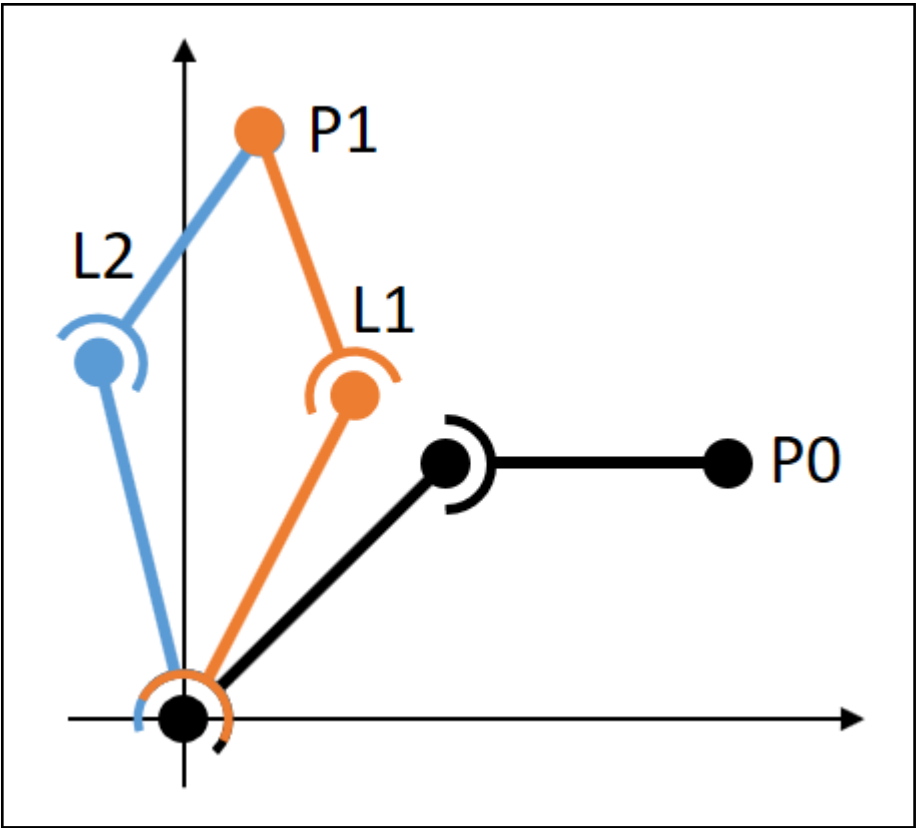


Fig. 4-11: Different arm behaviors of a Scara robot

With a point-to-point motion from point P0 to point P1 that was not specified in the ACS can be approached in two ways (L1 and L2).

The arm selection behavior can be configured using K_4 :

- 0: When the target point is approached, the arm behavior is not changed at the starting point in time
- 1: When the target point is approached, the motion of the first arm is minimized from the starting to the target point. The arm behavior selected ensures that axis 1 does not exceed the limits of $\pm 179^\circ$

Example:

Configuration₄

- $_4 = 0$: Point P1 is approached with L2. The arm behavior does not change during the motion
- $_4 = 1$: Point P1 is approached with L2. Thus, the consequence is that L2 has a lower motion of arm 1

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees

Transformation schemes

A3:	Positioning axis	Translatory absolute	optional
A4:	Orientation axis	rotary	optional

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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 zero point to the point of intersection of the straight line AC	mm or inch (see K-0-0054)	Optional, see (1) ≥0
L4:	Distance from zero point to the tip C of the triangle	mm or inch (see K-0-0054)	Optional, see (1) ≥0
L5:	Distance from zero point to straight line AB	mm or inch (see K-0-0054)	Optional, see (1) ≥0

Tab. 4-12: 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 4	Factor	Optional
K2:	Mechanical coupling between motors 4 and 3 (distance/lathe spindle)	mm/degree	optional
K4:	Arm behavior selection for PTP motions and point specification in non-ACS • 0: Arm behavior remains • 1: The arm behavior selected minimizes the motion of the first arm	Factor	Optional

Tab. 4-13: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.6 2-axis parallelogram

To use this transformation, enter transformation number 12 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes

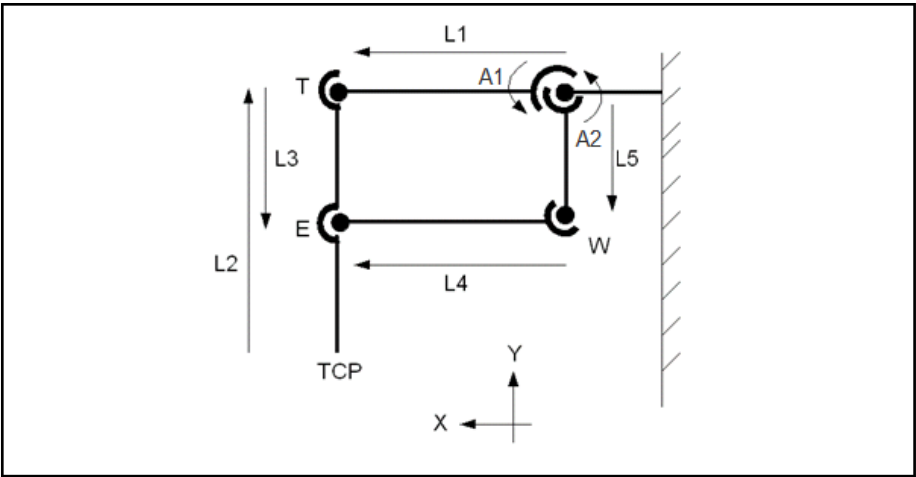


Fig. 4-12: Schematic view of 2-axis parallelogram

Axis directions of rotation and zero positions

fig. 4-12 "Schematic view of 2-axis parallelogram" on page 124 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

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	

Tab. 4-14: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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. 4-15: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.7 Inverted 2-axis parallelogram

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

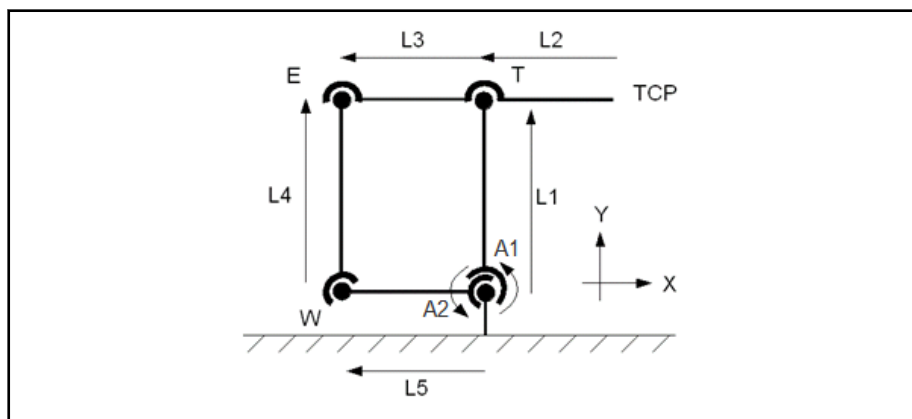


Fig. 4-13: Schematic view of 2-axis parallelogram

Axis directions of rotation and zero positions

fig. 4-13 "Schematic view of 2-axis parallelogram" on page 125 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 .

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes

Transformation schemes

- Belt axes

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

4.2.8 2-axis press

To use this transformation, enter transformation number 14 in parameter "K-0-0031, Transformation scheme".

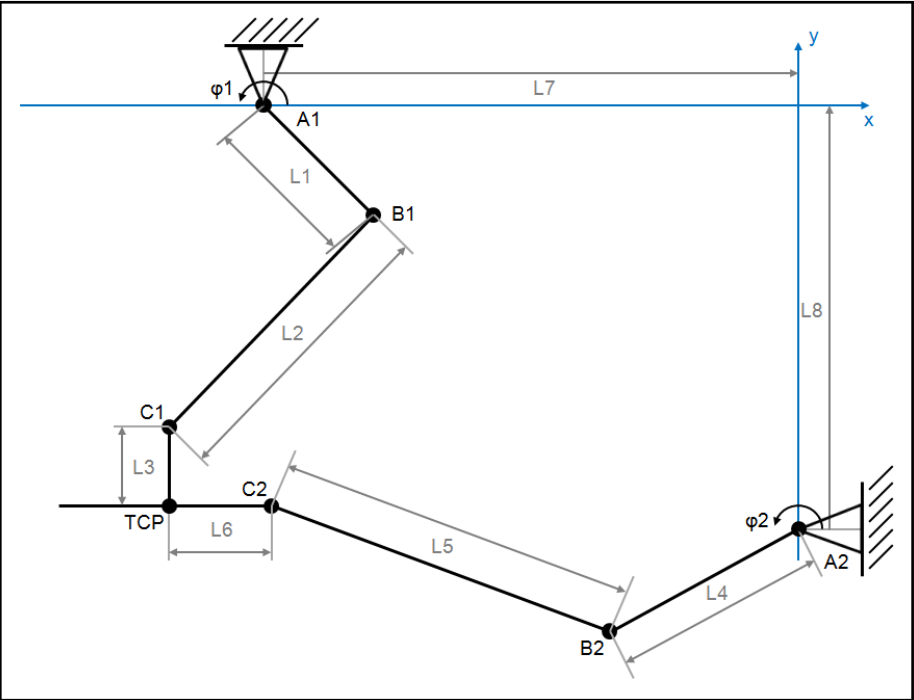


Fig. 4-14: Robot type 14: 2-axis press with dimensions

Transformation schemes

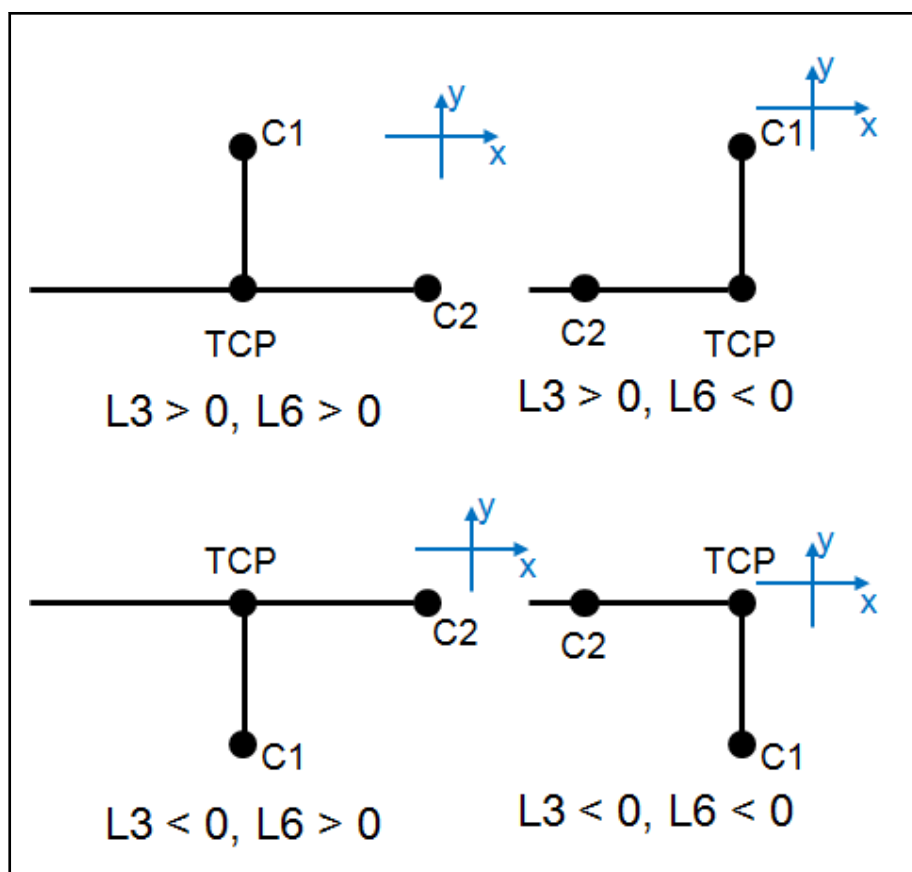


Fig. 4-15: Position of TCP depending on the lengths $L3$ and $L6$

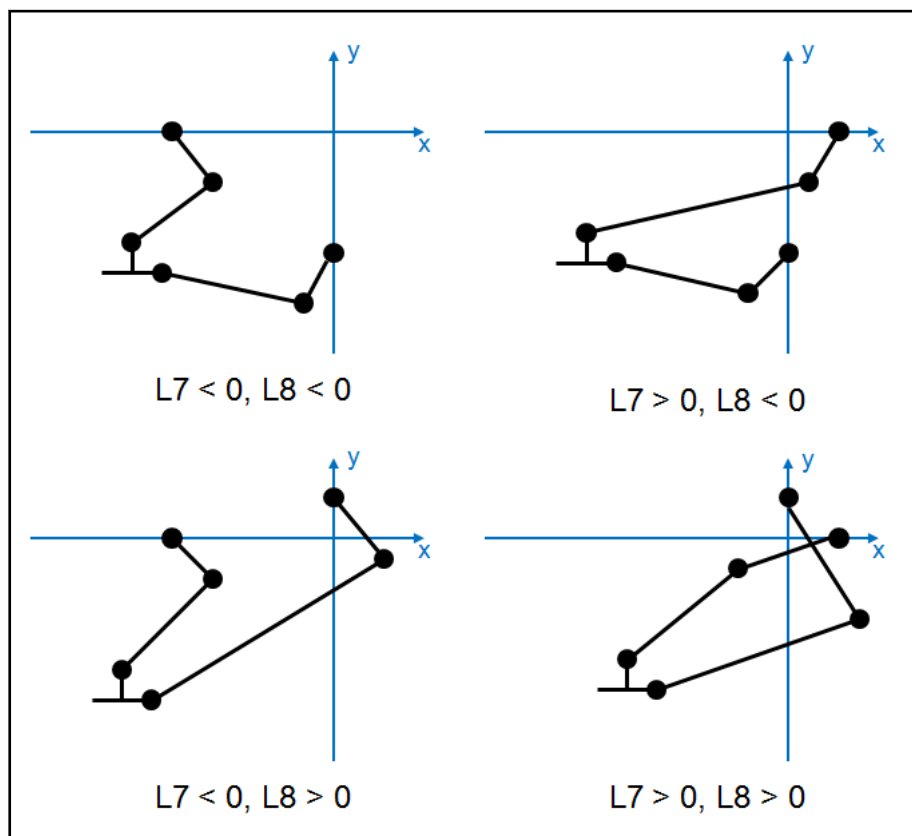


Fig. 4-16: Position of axes depending on $L7$ and $L8$

Transformation schemes

Axis directions of rotation and zero positions

- The zero position of the axes A1 and A2 is in positive X-direction. The direction of rotation corresponds to the mathematically positive direction
- The rod lengths L1, L2, L4 L5 are measured from the reference points on both sides. Lengths L3, L6, L7 L8 may also be negative and are to be understood as distance in negative direction in this case
- Axis A1 is located on the X-axis, axis A2 is located on the Y-axis

Coordinates of a point

Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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

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

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the first rod of the first arm	mm or inch (see K-0-0054)	>0
L2:	Length of the second rod of the first arm	mm or inch (see K-0-0054)	>0
L3:	Distance between the end of the first arm and TCP in y	mm or inch (see K-0-0054)	
L4:	Length of the first rod of the second arm	mm or inch (see K-0-0054)	>0
L5:	Length of the second rod of the second arm	mm or inch (see K-0-0054)	>0
L6:	Distance between the end of the second arm and TCP in x	mm or inch (see K-0-0054)	
L7:	Distance between A1 and the coordinate origin on the X-axis	mm or inch (see K-0-0054)	
L8:	Distance between A2 and the coordinate origin on the Y-axis	mm or inch (see K-0-0054)	

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

4.2.9 Contact kinematics

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

Specify diameter (d) of the axes A1 and A2 as length L1 of the transformation lengths ([K-0-0040](#)) whose measuring unit corresponds to mm or inch (see [K-0-0054](#)). The default value of the diameter is 2.

Transformation schemes

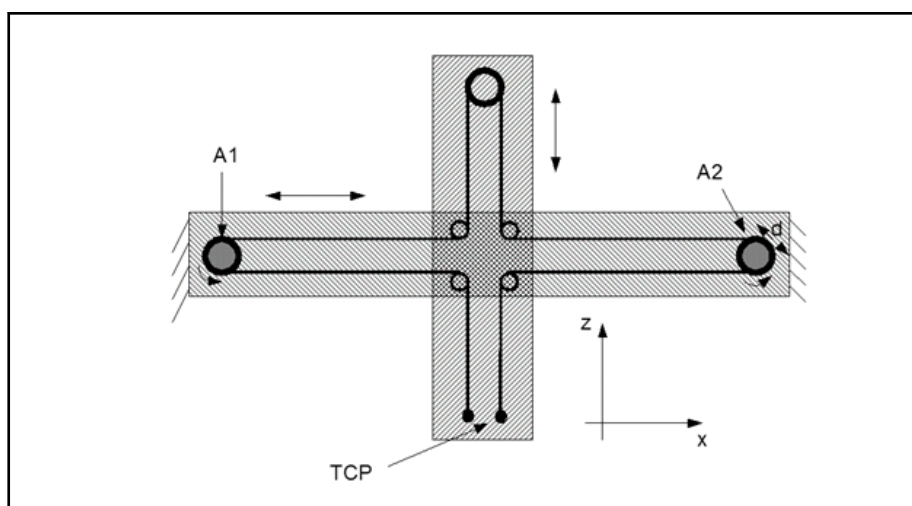


Fig. 4-17: Schematic representation of the contact kinematics

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 travelled 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 traversing range limits of the axes may not exceed the mechanical travel ranges
- Enter the diameter (d) in L₁ (axis lengths) (default value: 2)
- The respective base coordinate system is identified by the two main axes (X, Z)

Coordinates of a point

Point (X, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	

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

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit
L1:	Diameter (d) of the axes A1 and A2	mm or inch (see K-0-0054)

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

4.2.10 2-axis parallelogram V1

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

Transformation schemes

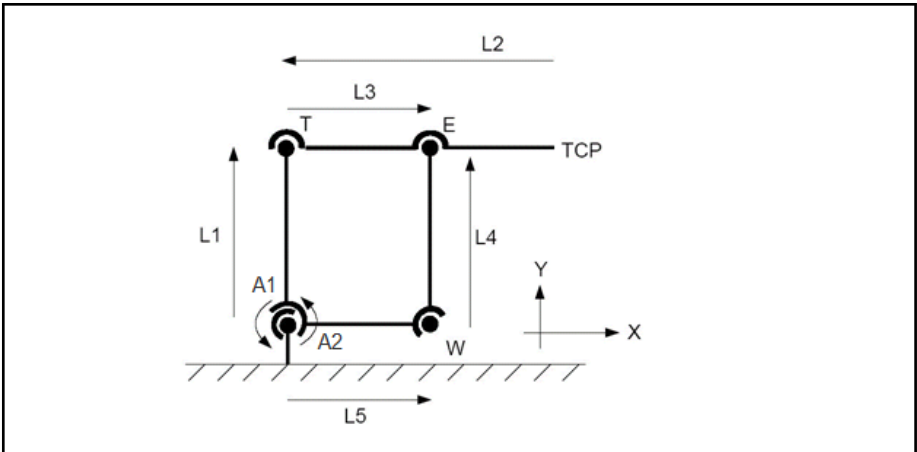


Fig. 4-18: Schematic view of 2-axis parallelogram V1

Axis directions of rotation and zero positions

- fig. 4-18 "Schematic view of 2-axis parallelogram V1" on page 130 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 .

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	

Tab. 4-22: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformati on 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

Transformation schemes

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. 4-23: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.11 3-axis delta with 1 optional orientation axis

To use this transformation, enter transformation number 20 in parameter "[K-0-0031](#), Transformation scheme".

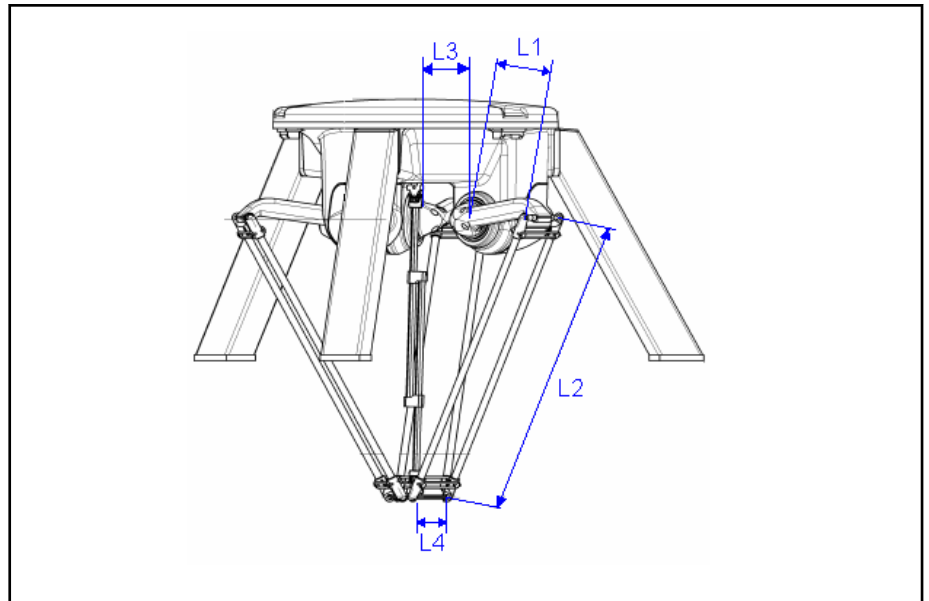


Fig. 4-19: Schematic view of 3-axis delta with 1 optional orientation axis

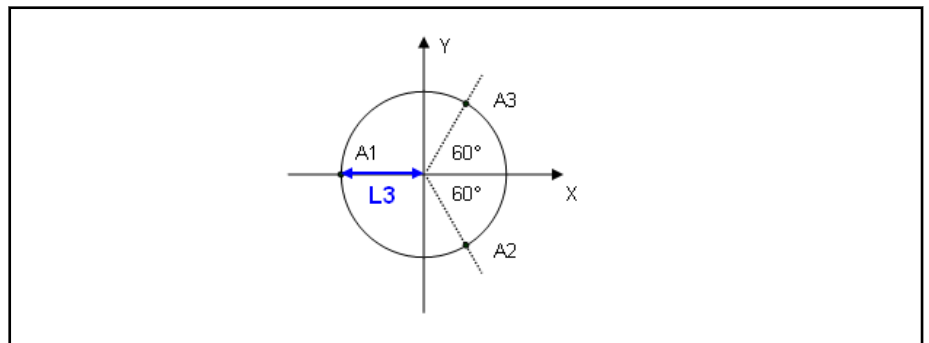


Fig. 4-20: Radius L3 of the motor base and the coordinate system (X-Y view)

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

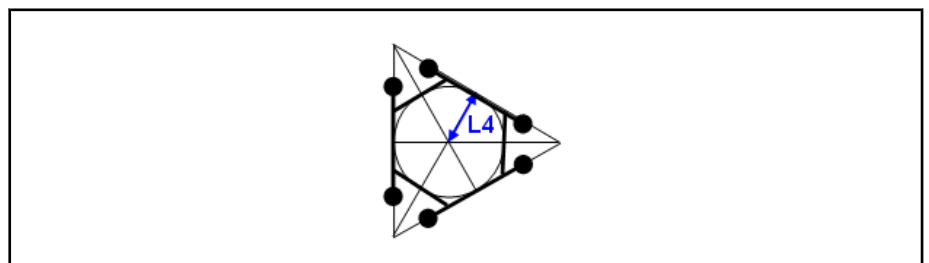
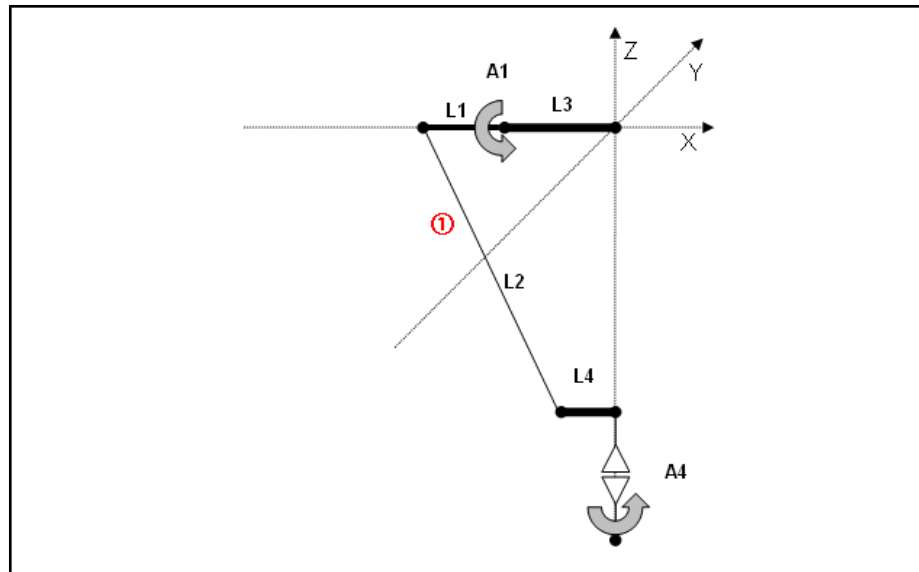


Fig. 4-21: Radius L4 of the die plate

Transformation schemes



① Connection rod

Fig. 4-22: Representation of zero position of the first axis

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

Axis directions of rotation and zero positions

- fig. 4-19 "Schematic view of 3-axis delta with 1 optional orientation axis" on page 131 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 has to be ensured by an appropriate setting of the actual values of the reference point.

- 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. 4-20 "Radius L_3 of the motor base and the coordinate system (X-Y view)" on page 131).
- L_4 is the radius of the inner circle of the equilateral triangle on the die plate. The equilateral triangle is defined by the reference points of the rods. (see fig. 4-21 "Radius L_4 of the die plate" on page 131).
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down in 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 motor base and the coordinate system (X-Y view)" on page 131).

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	
A3:	Positioning axis	Rotary absolute	
A4:	Orientation axis	Rotary abs./modulo	optional

Tab. 4-24: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of the rod to the die 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. 4-25: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.12 2-axis parallelogram V2

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

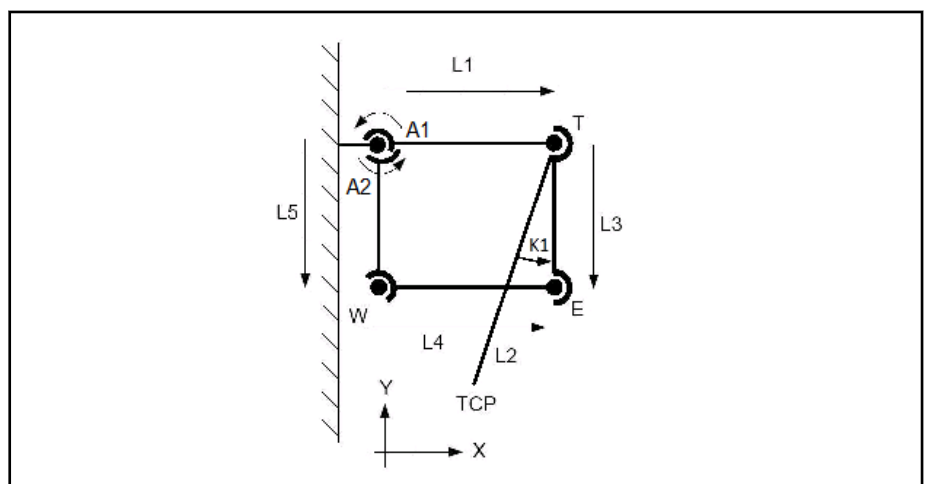


Fig. 4-23: Schematic view of 2-axis parallelogram V2

Axis directions of rotation and zero positions

fig. 4-23 "Schematic view of 2-axis parallelogram V2" on page 133 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

Transformation schemes

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

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	

Tab. 4-26: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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. 4-27: 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. 4-28: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.13 3-axis delta linear with 1 optional orientation axis

To use this transformation, enter transformation number 22 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes

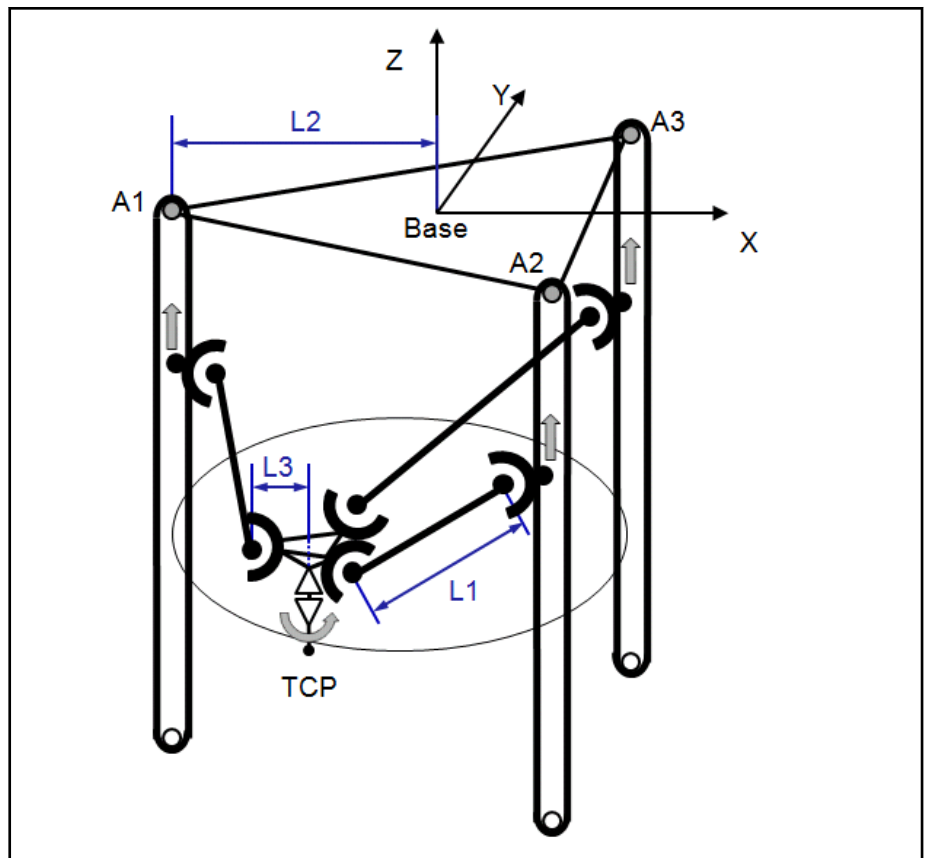


Fig. 4-24: Schematic view of 3-axis delta linear with 1 optional orientation axis
 The three axes A1 to A3 are all displaced by 120° from each other.

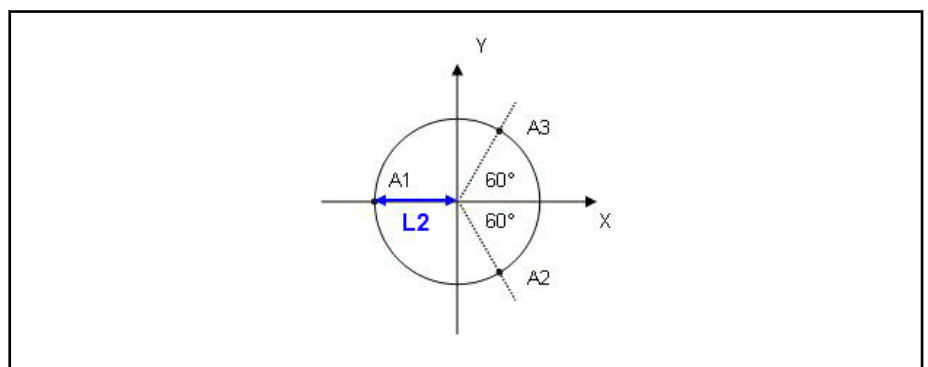


Fig. 4-25: Radius L2 of the motor base and the coordinate system (X-Y view)

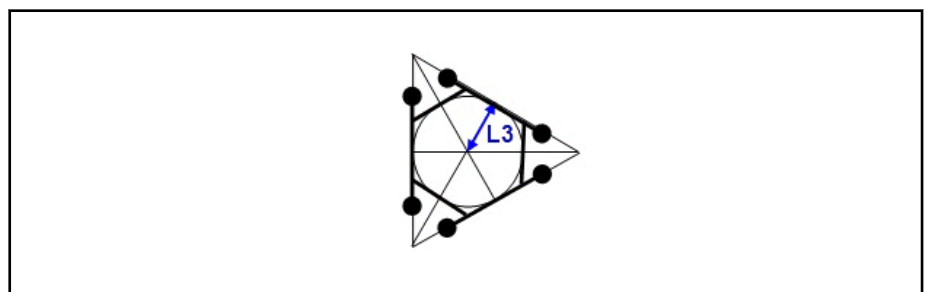


Fig. 4-26: Radius L3 of the die plate

Transformation schemes

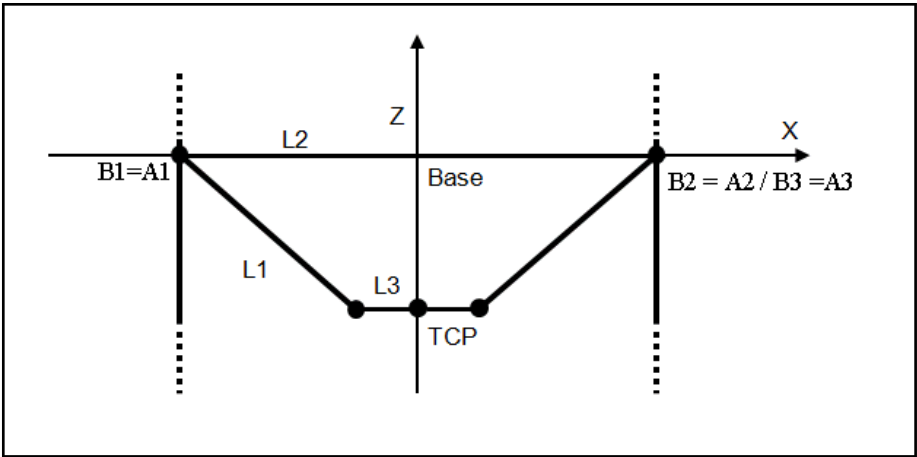


Fig. 4-27: Zero position of the axes

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

Axis directions of rotation and zero positions

- fig. 4-27 "Zero position of the axes" on page 136 shows a delta linear 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 has to be ensured by an appropriate setting of the actual values of the reference point.

- The zero point for the 4th axis (orientation axis) is in the positive direction of the X-axis
- The rod length L_1 is measured from the reference points on both sides
- L_2 is the radius of the perimeter formed by the three guide rails, measured at the rotary center points of the arms
- L_3 is the radius of the inner circle of the equilateral triangle on the plate. The equilateral triangle is defined by the reference points of the rods. The radius has to be at least L_2 (see fig. 4-26 "Radius L_4 of the die plate" on page 135)
- The retract feed direction of the positioning axes A1 to A3 is positive if the arms move down in Z-direction. The auxiliary axis A4 (rotation along the Z-axis) rotates positively counterclockwise.
- The Cartesian zero point for the delta linear robot is at the center point of the perimeter defined by the three guide rails in zero position (see also fig. 4-25 "Radius L_3 of the motor base and the coordinate system (X-Y view)" on page 135)

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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

Transformation schemes

A3:	Positioning axis	Translatory absolute	
A4:	Orientation axis	Rotary absolute/modulo	Optional

Tab. 4-29: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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

Tab. 4-30: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.14 5-axis delta

To use this transformation, enter transformation number 23 in parameter "K-0-0031, Transformation scheme".

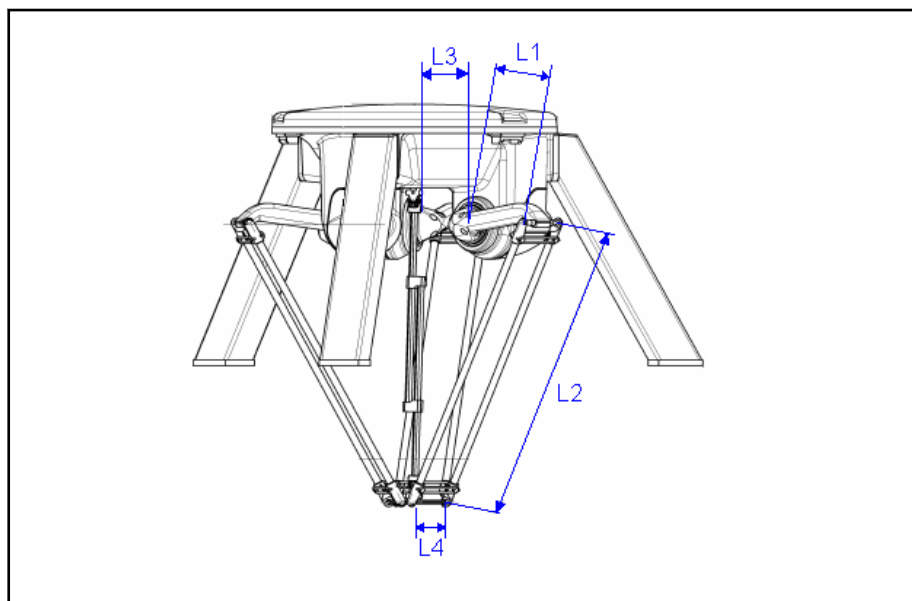


Fig. 4-28: Schematic view of 5-axis delta without TCP extension

Transformation schemes

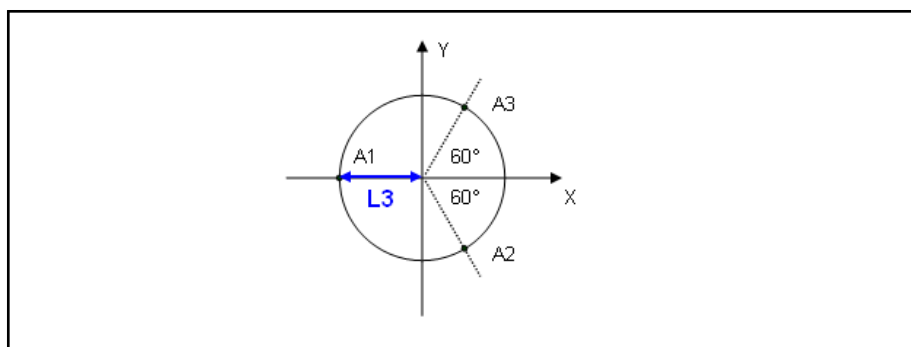


Fig. 4-29: Radius $L3$ of the joint base and the coordinate system (X-Y view)
The three joints A1 to A3 are all displaced by 120° from each other.

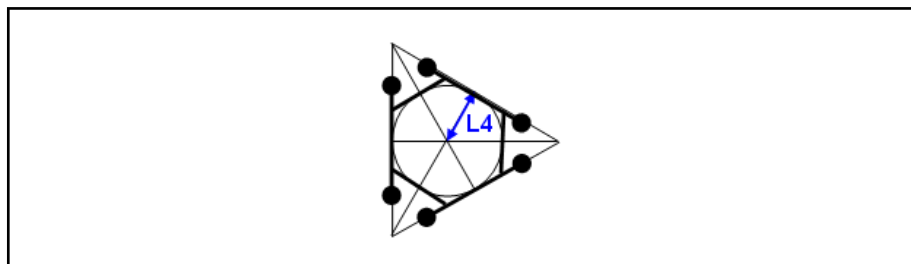
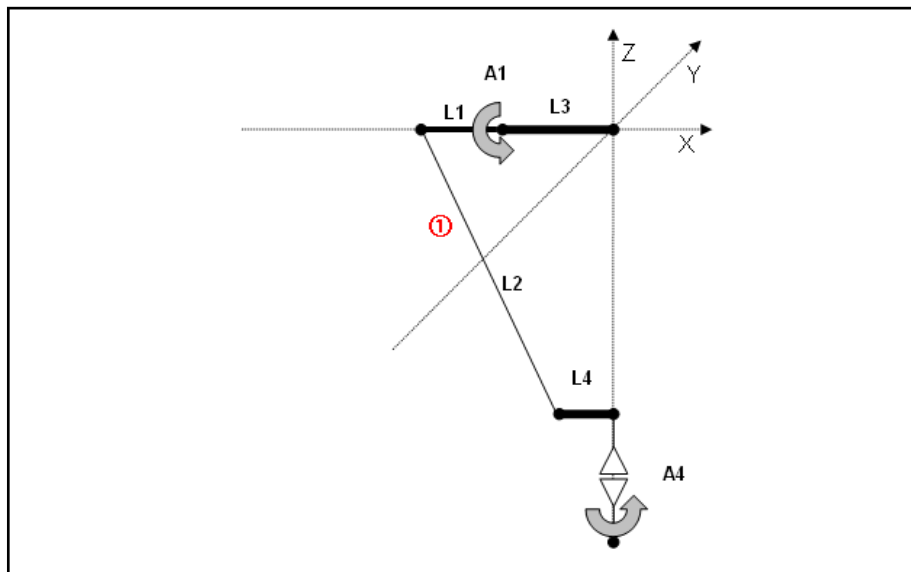


Fig. 4-30: Radius $L4$ of the die plate



① Connection rod

Fig. 4-31: Representation of zero position of the first axis

Transformation schemes

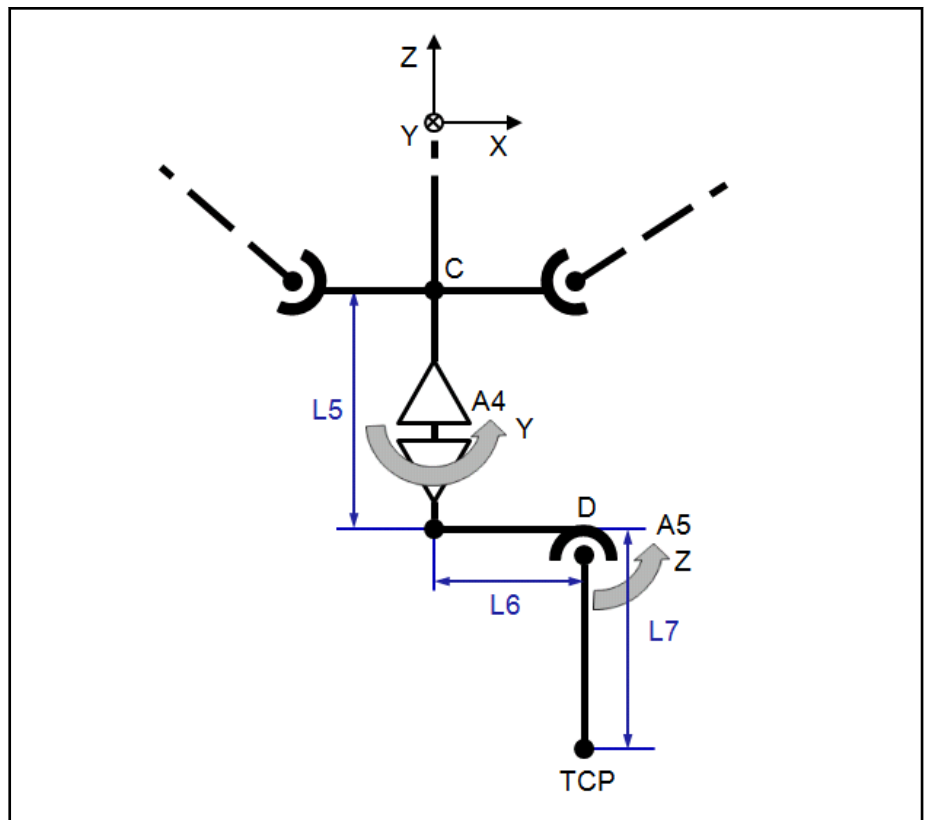


Fig. 4-32: Arrangement of the extension and fifth axis zero position

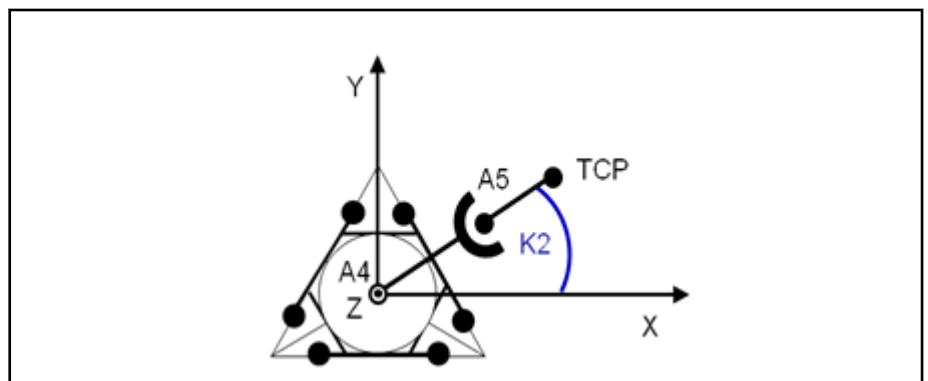


Fig. 4-33: Fourth axis zero position

Axis directions of rotation and zero positions

- fig. 4-28 "Schematic view of 5-axis delta without TCP extension" on page 137, fig. 4-32 "Arrangement of the extension and fifth axis zero position" on page 139 and fig. 4-33 "Fourth axis zero position" on page 139 show 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 has to be ensured by an appropriate setting of the actual values of the reference point.

- In the zero position, the axes 1, 2 and 3 (positioning axes) are horizontal. In zero position, the fifth axis (orientation axis) points vertically in negative Z-direction

Transformation schemes

- The zero point of the fourth axis (orientation axis) is defined by the K2 factor in degrees with a mathematically positive direction of rotation regarding the XY-plane
- The arm length L_1 is measured from the rotation center point to the center point of the rod holders
- 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. 4-29 "Radius L3 of the joint base and the coordinate system \(X-Y view\)" on page 138](#))
- L_4 is the radius of the inner circle of the equilateral triangle on die plate. The equilateral triangle is defined by the reference points of the rods. (see [fig. 4-30 "Radius L4 of the die plate" on page 138](#))
- L_5 is the vertical length in Z-direction from the center of the inner circle of the equilateral triangle on the tool plate to the bottom reference point of axis 5. L_6 is equivalent to the horizontal length from the center to the reference point [fig. 4-32 "Arrangement of the extension and fifth axis zero position" on page 139](#)
- L_7 is the arm length from the rotation center of the fifth axis to the end point (see [fig. 4-32 "Arrangement of the extension and fifth axis zero position" on page 139](#))
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down in Z-direction. The auxiliary axis A4 (rotation along the Z-axis) rotates positively counterclockwise. The direction of rotation of the auxiliary axis A5 is positive if the arm moves down in Z-direction
- 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. 4-29 "Radius L3 of the joint base and the coordinate system \(X-Y view\)" on page 138](#))
- The transformation considers a mechanical coupling of the axes 4 and 5.

The transmission ratio of the coupling is specified in K2. The driving output element is axis 5 and the driving input element is axis 4. K2 is defined as output by input

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+/- 90 degrees
A2:	Positioning axis	Rotary absolute	+/- 90 degrees
A3:	Positioning axis	Rotary absolute	+/- 90 degrees
A4:	Orientation axis	Rotary absolute	
A5:	Orientation axis	Rotary absolute	

Tab. 4-31: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

Transformation schemes

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of the rod to the die 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:	Vertical length of the center of the base plate to the bottom reference point	mm or inch (see K-0-0054)	>0
L6:	Horizontal length of the center of the base plate to the bottom reference point	mm or inch (see K-0-0054)	>0
L7:	Length of the arm of the fifth axis	mm or inch (see K-0-0054)	>0

Tab. 4-32: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation length	Axis configuration	Measuring unit	Note
K1:	Mechanical coupling of the fourth and fifth axis	Factor	>0
K2:	Offset of the zero position of the fourth axis	Degrees	optional

Tab. 4-33: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.15 Adabot

To use this transformation, enter transformation number 30 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes

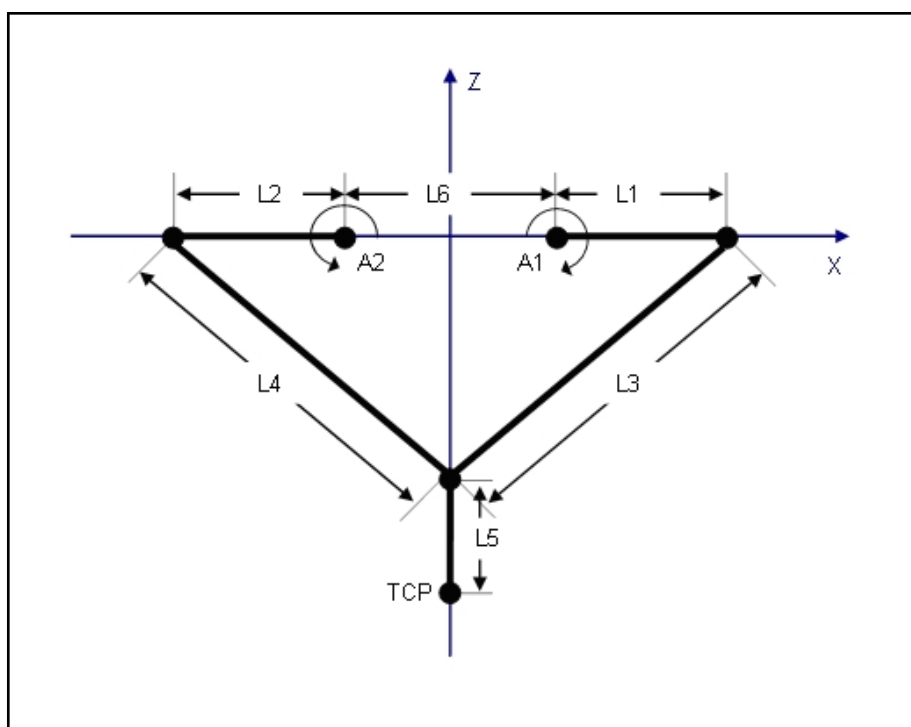


Fig. 4-34: Adabot (X-Z view, axis zero positioning)

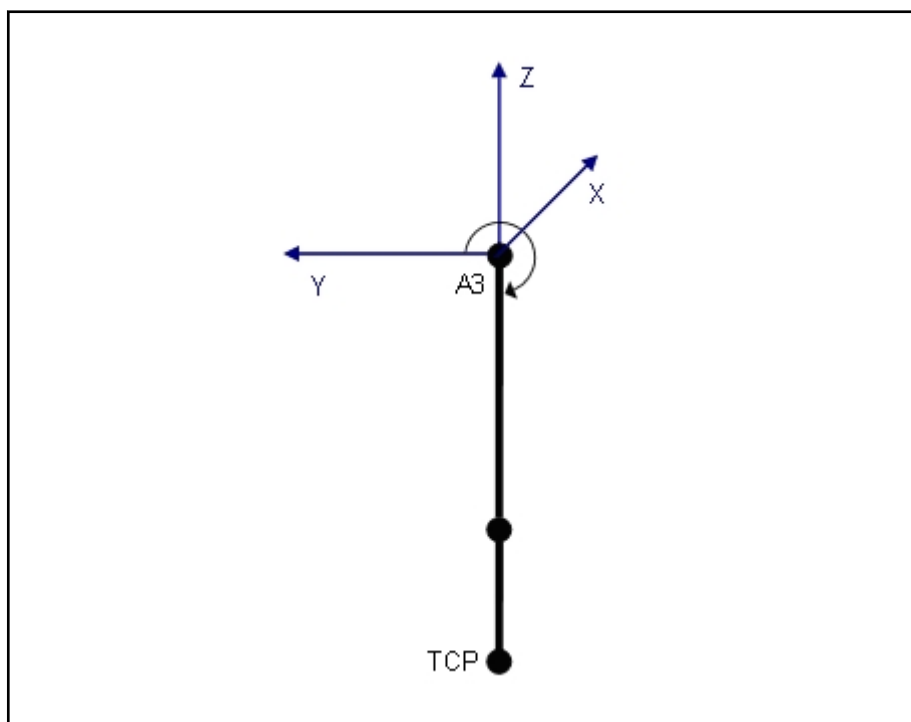


Fig. 4-35: Adabot (X-Z view, axis zero positioning)

Transformation schemes

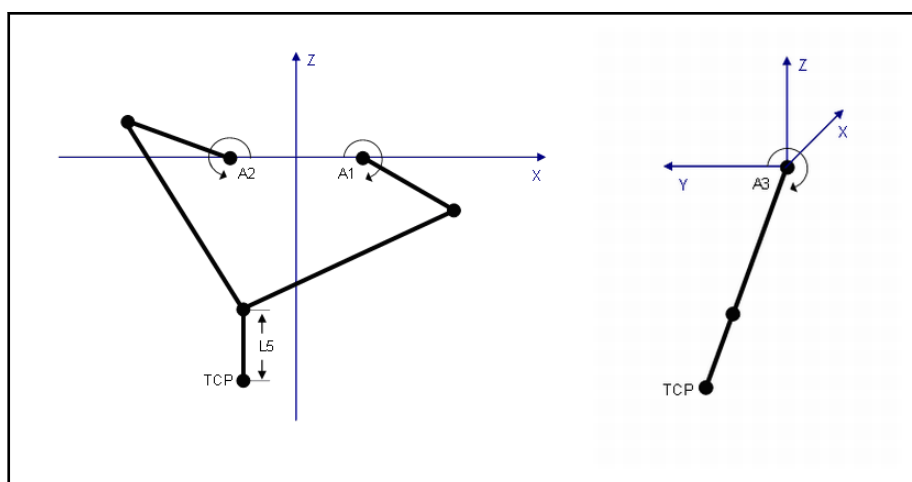


Fig. 4-36: Adabot (behavior of rod L5)

Axis directions of rotation and zero positions

fig. 4-34 "Adabot (X-Z view, axis zero positioning)" on page 142 and fig. 4-35 "Adabot (X-Z view, axis zero positioning)" on page 142 show the schematic kinematic representation of the Adabot kinematics for zero positioning of all axes.

- The figures also show the arm lengths L_1 , L_2 , L_3 , L_4 , L_5 , L_6 .
- The positive directions of rotation of the axes A1, A2, A3 are also shown.
- The lengths L_1 , L_2 , L_3 , L_4 , L_6 always have to be entered as positive number and unequal to zero. L_5 always has to be specified as positive number or zero.

The rod to TCP (L_5) is always held vertically to the x-axis by the mechanics (not represented in the figure). The rod to TCP (L_5) is always in the plane spanned by the rods L_1 , L_2 , L_3 , L_4 (see fig. 4-36 "Adabot (behavior of rod L5)" on page 143).

The Cartesian zero point of the Adabot kinematics is in the center of both arm rotation points A1 and A2 (see fig. 4-34 "Adabot (X-Z view, axis zero positioning)" on page 142).

Coordinates of a point

Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+/- 90 degrees
A2:	Positioning axis	Rotary absolute	+/- 90 degrees
A3:	Positioning axis	Rotary absolute	+/- 90 degrees
A4:	Orientation axis	rotary	optional

Tab. 4-34: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

Transformation schemes

- Orientation axes interpreted as auxiliary axes
- Belt axes



Changing the arm behavior of the L1,L3 and L2,L4 arms (e.g. axis A1 and axis A2 are at -86 degrees in [fig. 4-34 "Adabot \(X-Z view, axis zero positioning\)"](#) on page 142) is not supported by the kinematics and is interpreted as an error.

TCP coordinates are interpreted as error by the kinematics outside the reachable space of the entered arm lengths in the MCS.

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of arm at motor (right)	mm or in (see K-0-0054)	>0
L2:	Length of arm at motor (left)	mm or in (see K-0-0054)	>0
L3:	Length of rod at arm L1	mm or in (see K-0-0054)	>0
L4:	Length of rod at arm L2	mm or in (see K-0-0054)	>0
L5:	Rod to TCP	mm or in (see K-0-0054)	>=0
L6:	Distance of the two rotation points of the arms	mm or in (see K-0-0054)	>0

Tab. 4-35: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.16 2-axis delta with asymmetric TCP

To use this transformation, enter transformation number 32 in parameter "[K-0-0031](#), Transformation scheme".

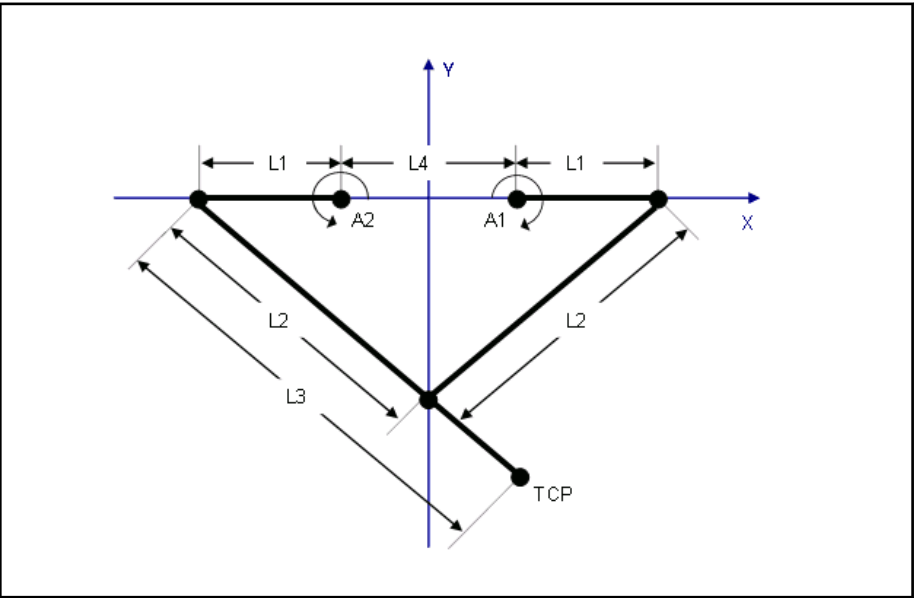


Fig. 4-37: 2-axis delta with asymmetric TCP (X-Y view, axis zero positioning)

Coordinates of a point

Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Transformation schemes

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	+/- 90 degrees
A2:	Positioning axis	Rotary absolute	+/- 90 degrees
A3:	Orientation axis	rotary abs./modulo	optional

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

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or in (see K-0-0054)	>0
L2:	Length of short rod	mm or in (see K-0-0054)	>0
L3:	Length of the long rod to the TCP	mm or in (see K-0-0054)	>L2
L4:	Distance of the two rotation points of the arms	mm or in (see K-0-0054)	>0

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

Axis directions of rotation and zero positions

fig. 4-37 "2-axis delta with asymmetric TCP (X-Y view, axis zero positioning)" on page 144 shows the schematic representation of the 2-axis delta kinematics with asymmetric TCP when all axes are in zero position.

- The figures also show the lengths L_1 , L_2 , L_3 , L_4
- The positive directions of rotation of the axes A1, A2 are also shown
- The lengths L_1 , L_2 , L_3 , L_4 always have to be entered as positive number or unequal to zero. The value of L_3 always has to be greater than of L_2
- Length $(L_4/2 + L_1)$ has to be smaller than length L_2
- The Cartesian zero point is in the center of the two arm rotation points A1 and A2

4.2.17 3-Axis Twisting Crane

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

Transformation schemes

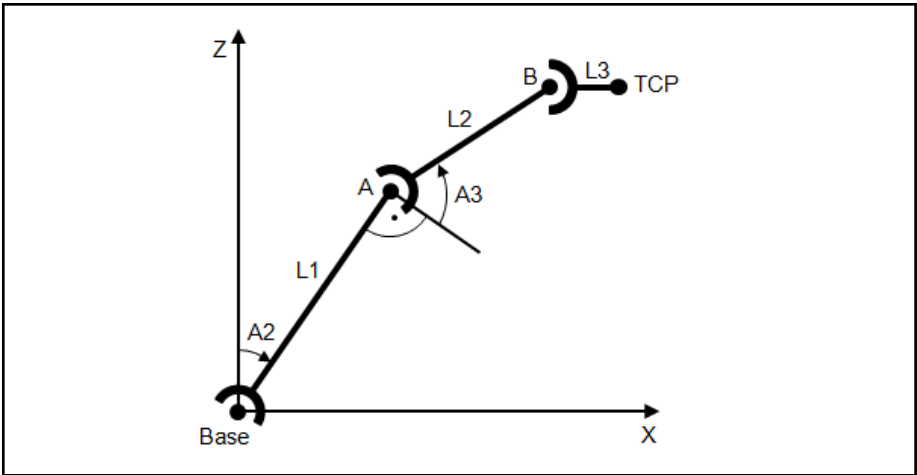


Fig. 4-38: Schematic side view of 3-axis twisting crane

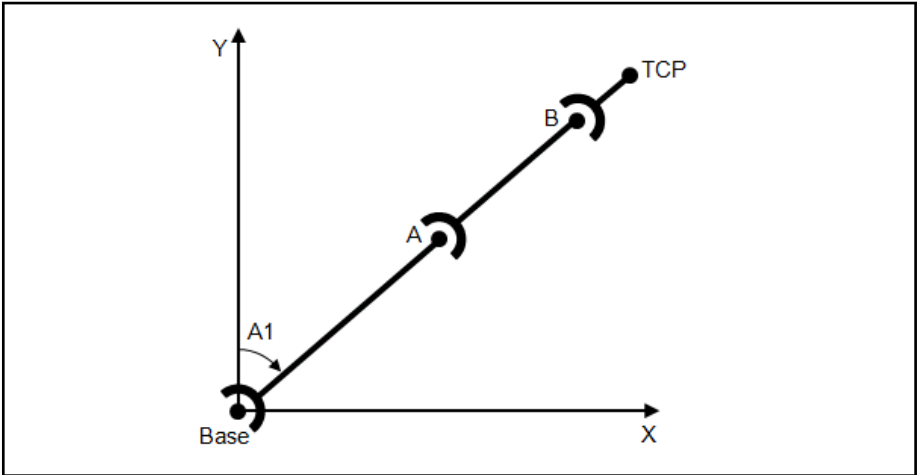


Fig. 4-39: Schematic top view of 3-axis twisting crane

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

Coordinates of a point

Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	

Transformation schemes

A2:	Positioning axis	Rotary absolute	$-180^\circ < x < 180^\circ$
A3:	Positioning axis	Rotary absolute	$-90^\circ < x < 90^\circ$

Tab. 4-38: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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. 4-39: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.18 3-axis flex stamper

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

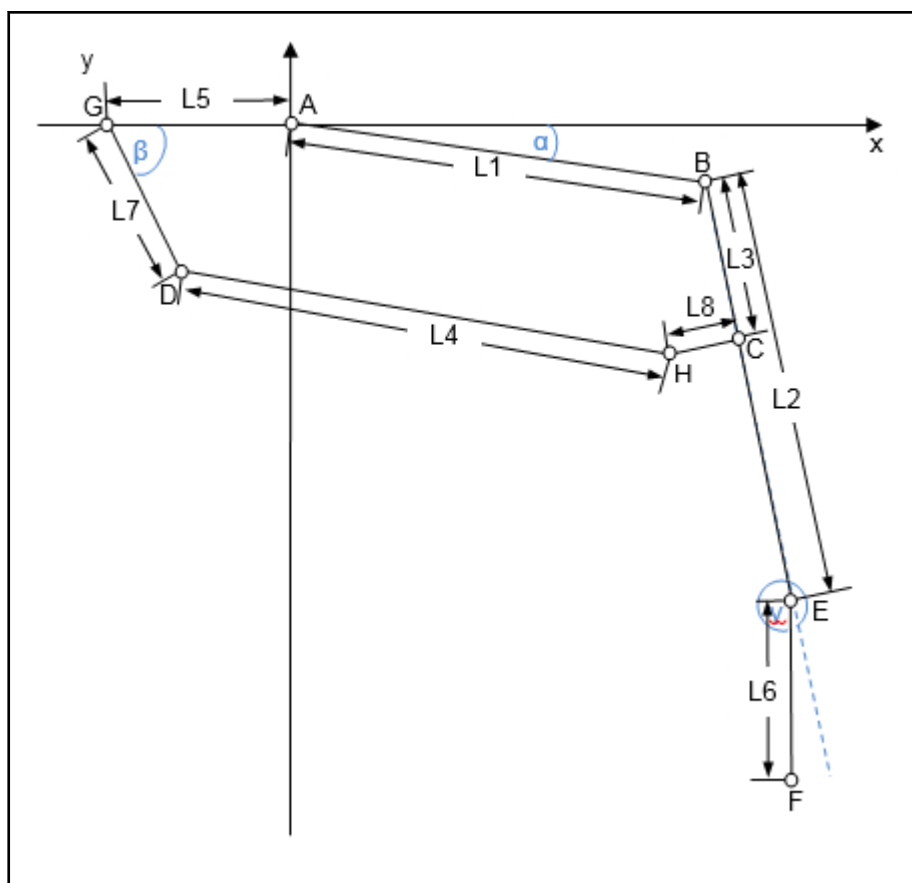


Fig. 4-40: Schematic view of 3-axis flex stamper

Axis directions of rotation and zero positions

fig. 4-40 "Schematic view of 3-axis flex stamper" on page 147 shows

- The schematic kinematics representation in position axis 1 = α , axis 2 = β , axis 3 = γ

Transformation schemes

- The zero position of the axes A1 and A2 is in positive X-direction
- The zero position of axis A3 is the lengthening of L2
- Direction of rotation of the three axes is in counterclockwise direction
- The arm lengths $L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8$
- 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

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	$-270^\circ \leq x < 90^\circ$
A2:	Positioning axis	Rotary absolute	$-270^\circ \leq x < 90^\circ$
A3:	Orientation axis	Rotary absolute	$0^\circ < x < 360^\circ$

Tab. 4-40: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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, H	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
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
L8:	Arm length C, H	mm or inch (see K-0-0054)	≥ 0

Tab. 4-41: 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 parallel to the Y-axis. If L6 = 0, A3 is used as optional orientation axis 1: The angle of axis 3 can be specified as orientation around the Z-axis by the user

Tab. 4-42: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.19 3-axis delta with linear axes with 1 optional orientation axis

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

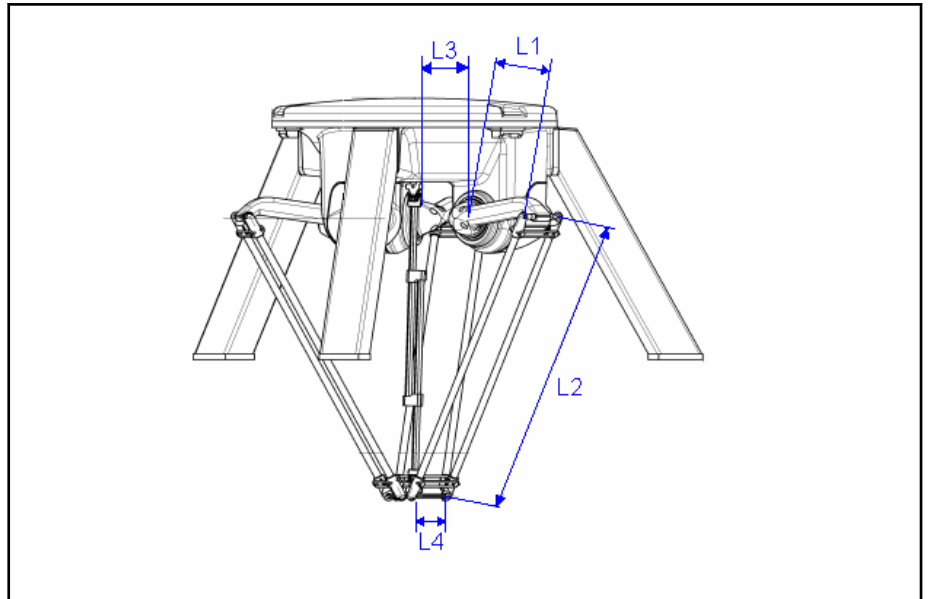


Fig. 4-41: Schematic view of 3-axis delta with 1 optional orientation axis

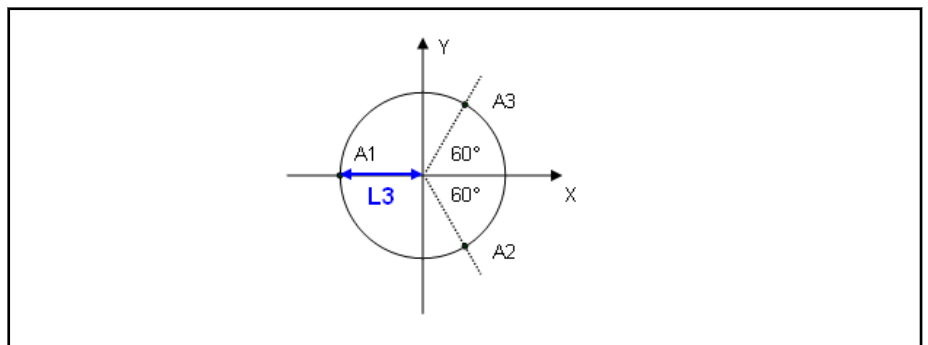


Fig. 4-42: Radius L3 of the joint base and the coordinate system (X-Y view)

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

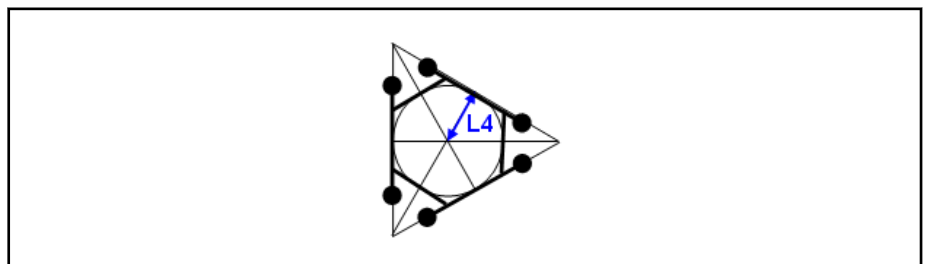
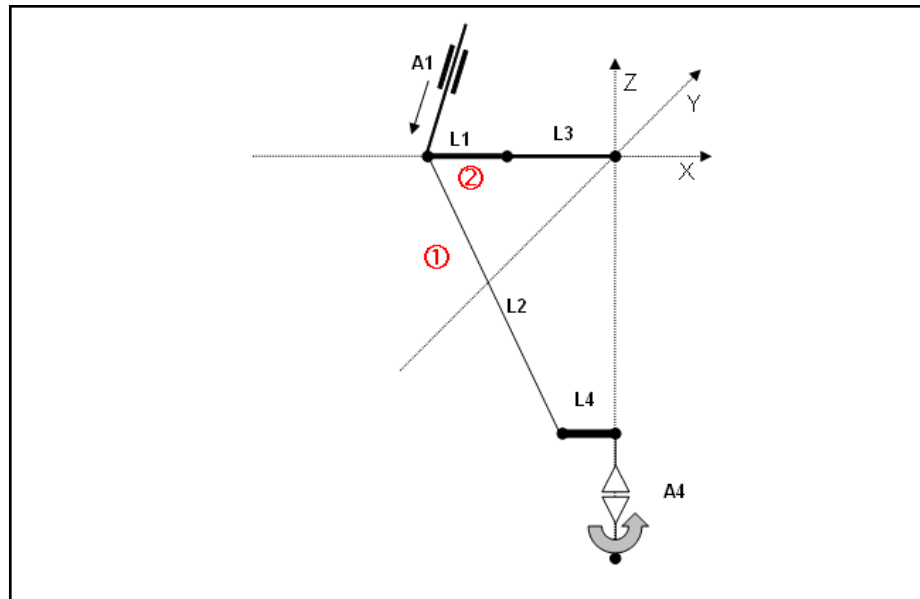


Fig. 4-43: Radius L4 of the die plate

Transformation schemes



① Connection rod

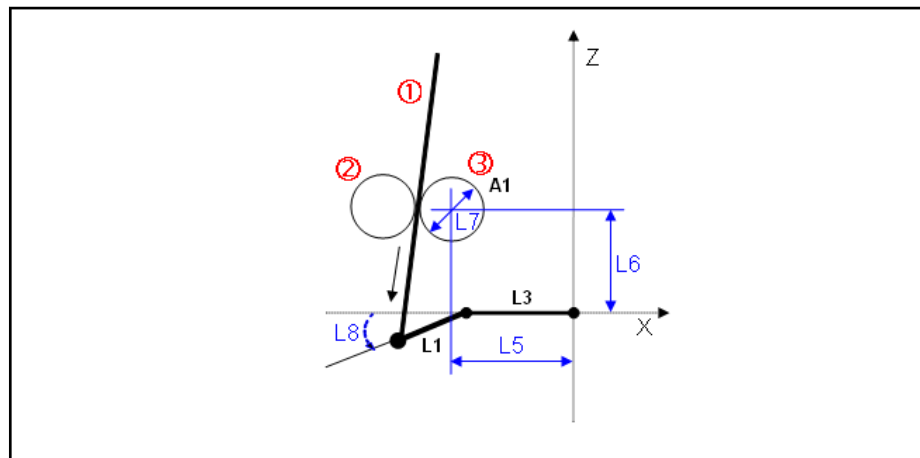
② Arm

Fig. 4-44: 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. 4-45: Advanced settings

Axis directions of rotation and zero positions

fig. 4-41 "Schematic view of 3-axis delta with 1 optional orientation axis" on page 149 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 has to be ensured by an appropriate setting of the actual values of the reference point.

- In zero position, the arms are horizontal

Transformation schemes

- 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 rotation center point to the center point of the rod holders
- 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. 4-42 "Radius L3 of the joint base and the coordinate system \(X-Y view\)" on page 149](#))
- L_4 is the radius of the inner circle of the equilateral triangle on the plate. The equilateral triangle is defined by the reference points of the rods. (see [fig. 4-43 "Radius L4 of the die plate" on page 149](#))
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down in Z-direction. The auxiliary 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. 4-42 "Radius L3 of the joint base and the coordinate system \(X-Y view\)" on page 149](#)).

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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

Tab. 4-43: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of the rod to the die 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. 4-44: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.20 2-axis delta slide

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

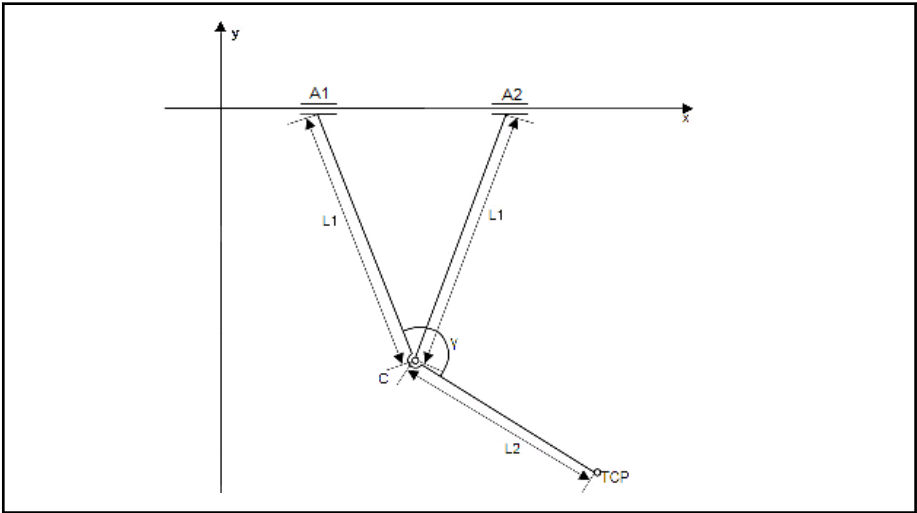


Fig. 4-46: Schematic view of 2-axis delta slide

Axis directions of rotation and zero positions

- fig. 4-46 "Schematic view of 2-axis delta slide" on page 152 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)

Coordinates of a point Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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

Tab. 4-45: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation schemes

Transformation on 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. 4-46: K-0-0040, Transformation lengths, axis configuration, axis lengths

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

Tab. 4-47: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.21 3-Axis Parallelogram V3

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

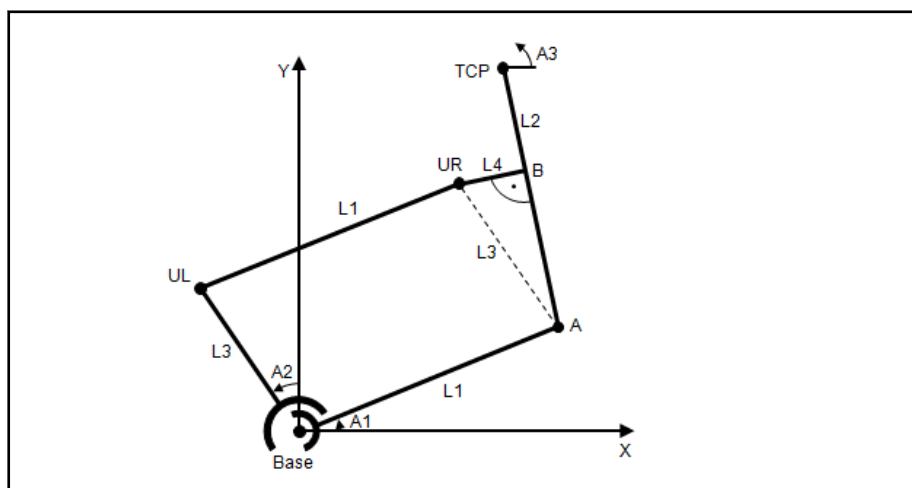


Fig. 4-47: Schematic view of 3-axis parallelogram V3

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

Coordinates of a point

Point (X, Y, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Transformation schemes

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

Tab. 4-48: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

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. 4-49: K-0-0040, Transformation lengths, axis configuration, axis lengths

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

Tab. 4-50: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.22 6-axis Cartesian

To use this transformation, enter transformation number 47 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes

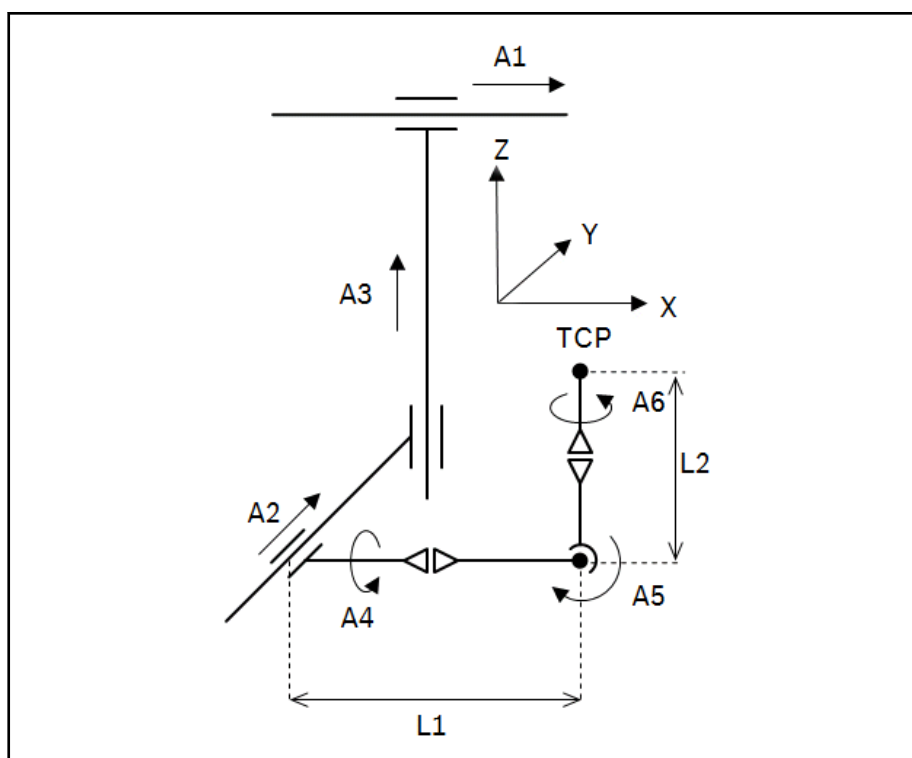


Fig. 4-48: 6-axis Cartesian (axis zero positioning)

Axis directions of rotation and zero positions

fig. 4-48 "6-axis Cartesian (axis zero positioning)" on page 155 shows the schematic kinematic representation for 6-axes Cartesian in zero position of all axes A4, A5, A6. Zero positions of axes A1, A2, and A3 correspond to the origin of the basic coordinate system.

The arm lengths L1 and L2 as well as the positive directions of rotations are also represented.

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Translatory absolute	
A2:	Positioning axis	Translatory absolute	
A3:	Positioning axis	Translatory absolute	
A4:	Orientation axis	Rotary absolute	
A5:	Orientation axis	Rotary absolute	
A6:	Orientation axis	Rotary absolute	

Tab. 4-51: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation schemes

Transformati on length	Axis configuration	Measuring unit	Axis length
L1:	Arm length between A2 and A5	mm or in (see K-0-0054)	>=0
L2:	Length of arm between A5 and TCP	mm or in (see K-0-0054)	>=0

Tab. 4-52: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.23 3-axis delta with 90 degree angles

To use this transformation, enter transformation number 56 in parameter "[K-0-0031](#), Transformation scheme".

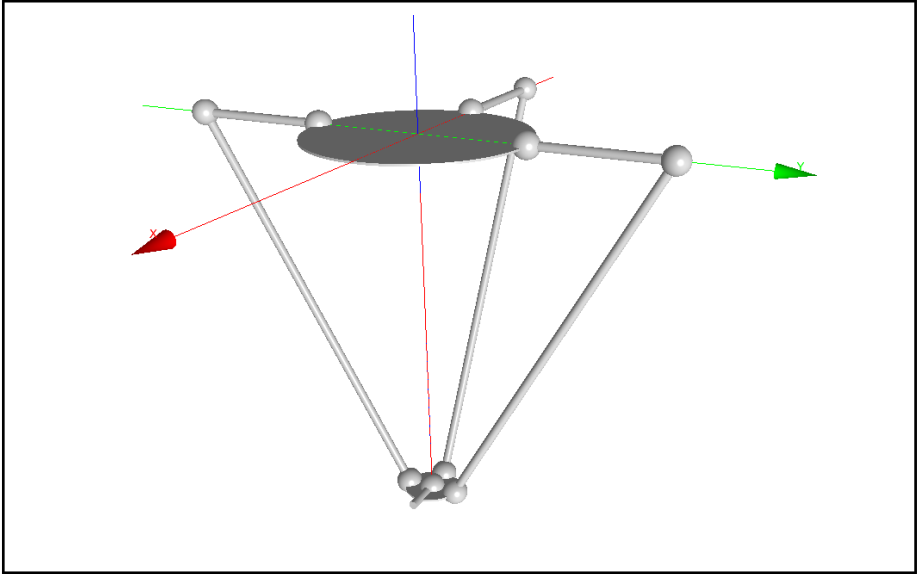


Fig. 4-49: Schematic view of 3-axis delta with 90 degree angles

Transformation schemes

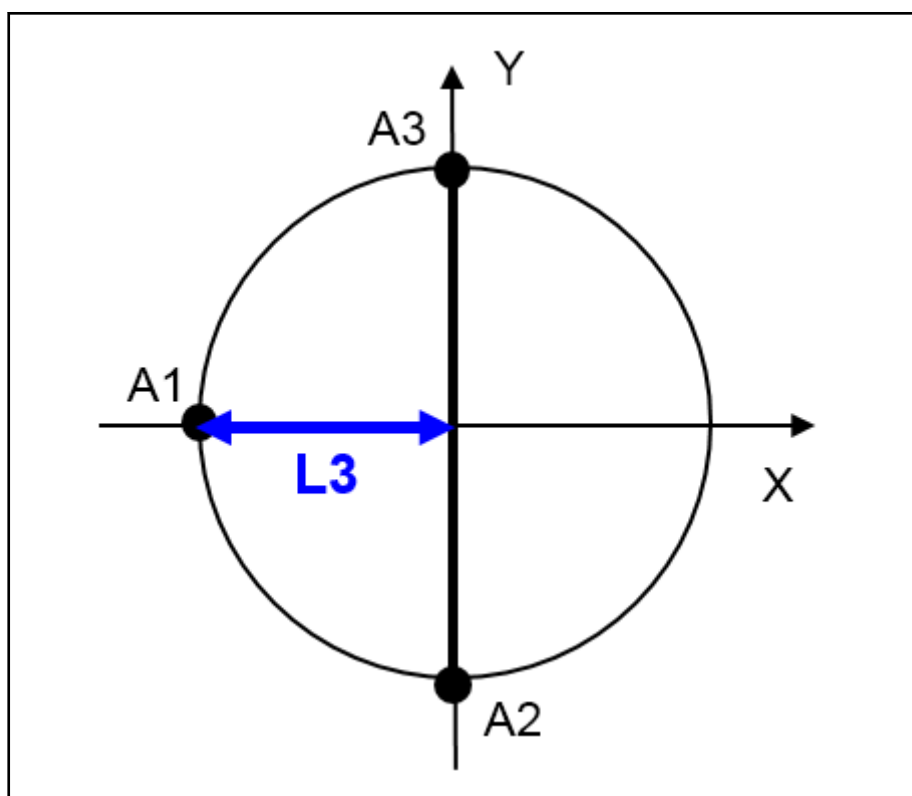


Fig. 4-50: Radius L3 of the motor mount and the coordinate system (x-,y-view)

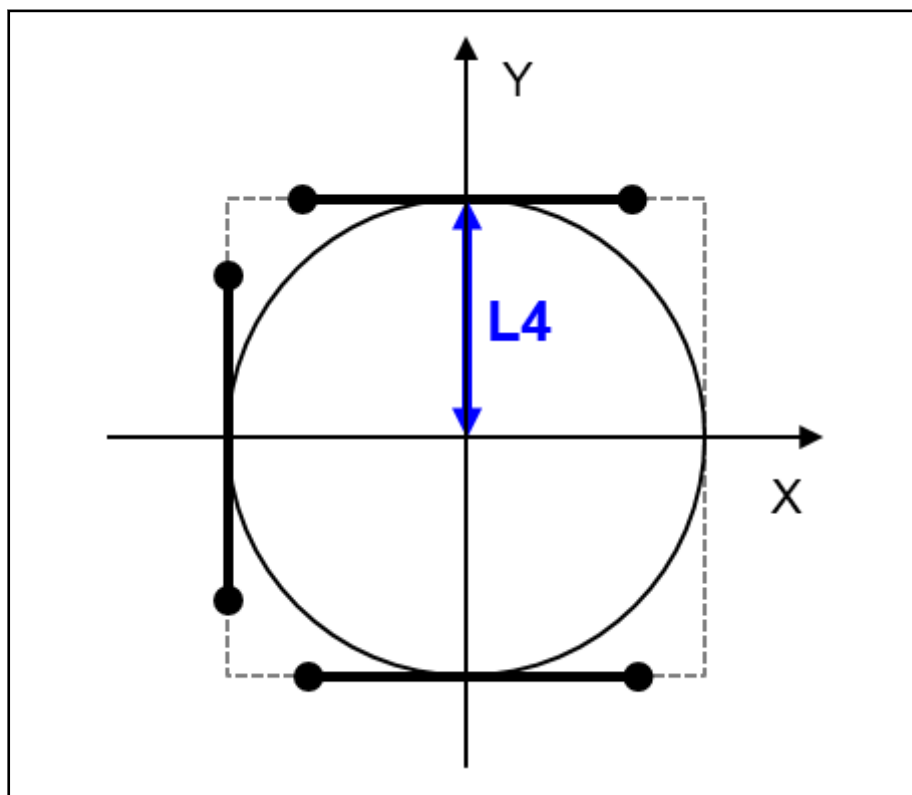
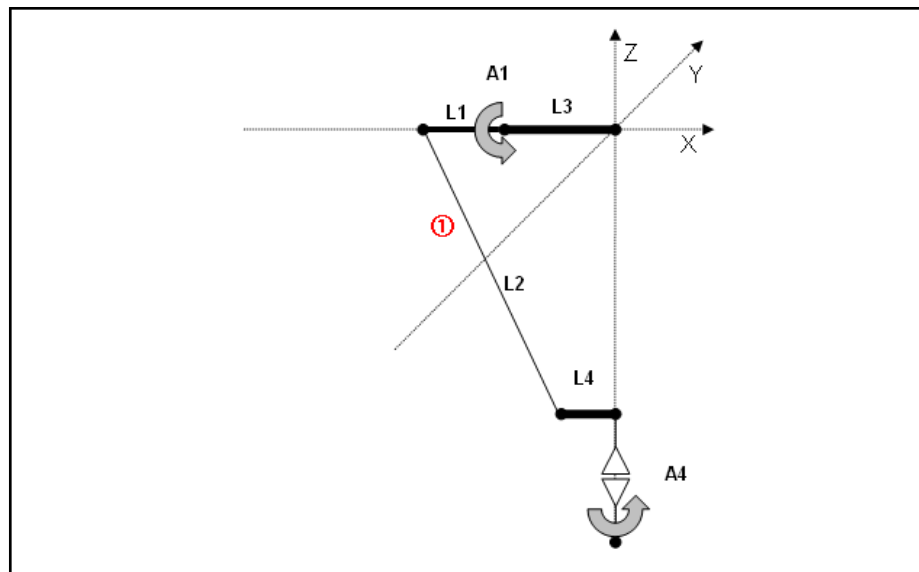


Fig. 4-51: Radius L4 of the die plate

Transformation schemes



① Connection rod

Fig. 4-52: Representation of zero position of the first axis

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

Axis directions of rotation and zero positions

- [fig. 4-49 "Schematic view of 3-axis delta with 90 degree angles" on page 156](#) 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 has to be ensured by an appropriate setting of the actual values of the reference point.

- In the zero position, axes 1, 2 and 3 (positioning axes) are horizontal
- Axis 1 point towards the zero position to the direction of the positive y-axis.
- Axis 2 point towards the zero position to the direction of the negative y-axis
- Axis 3 point towards the zero position to the direction of the negative x-axis
- 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 rotation center point to the center point of the rod holders
- 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. 4-50 "Radius L3 of the motor mount and the coordinate system \(x-,y-view\)" on page 157](#))
- L_4 is the radius of the internal circle of the square on the die plate. The square is defined by the reference points of the rods (see [fig. 4-51 "Radius L4 of the die plate" on page 157](#))
- The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down in Z-direction. The auxiliary axis A4 (rotation along the Z-axis) rotates positively counterclockwise.

Transformation schemes

- 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. 4-52](#) "Representation of zero position of the first axis" on page 158)

Coordinates of a point Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	
A3:	Positioning axis	Rotary absolute	
A4:	Orientation axis	Rotary absolute/modulo	Optional

Tab. 4-53: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm (leg) at the motor	mm or inch (see K-0-0054)	>0
L2:	Length of the rod to the die 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. 4-54: K-0-0040, Transformation lengths, axis configuration, axis lengths

4.2.24 4-axis speedpicker

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

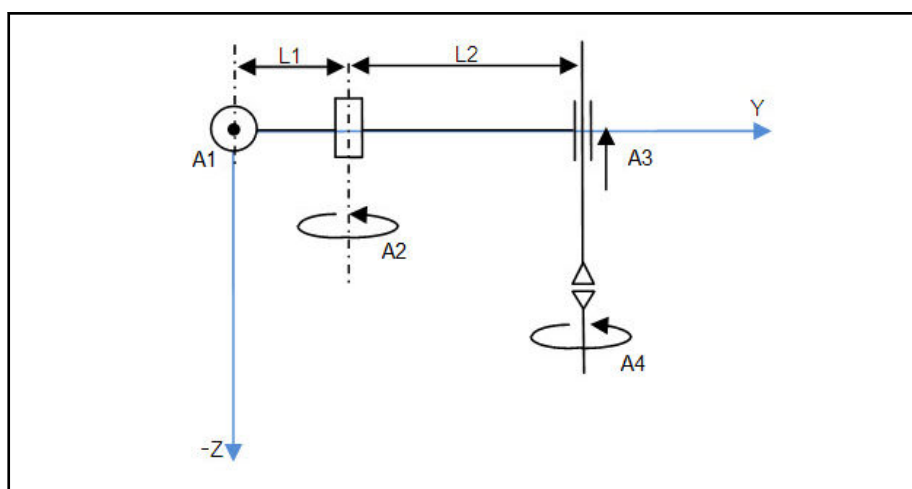


Fig. 4-53: Schematic view of 4-axis speedpicker

Transformation schemes

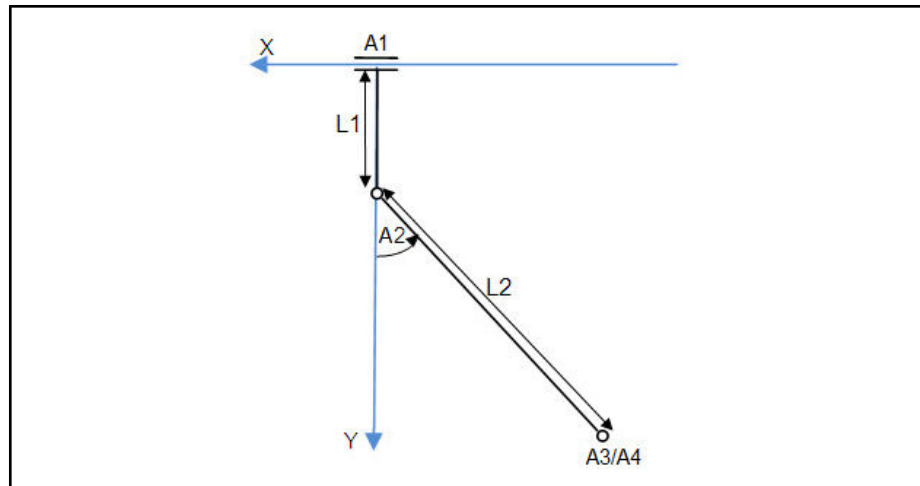


Fig. 4-54: Schematic view of 4-axis speedpicker (from above)

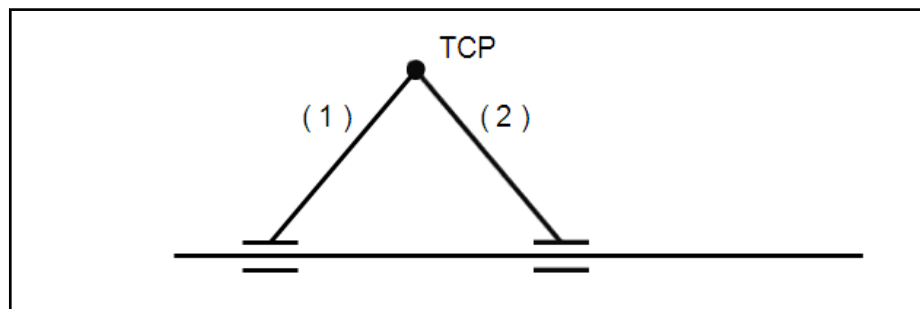
Axis directions of rotation and zero positions

fig. 4-53 "Schematic view of 4-axis speedpicker" on page 159 and fig. 4-54 "Schematic view of 4-axis speedpicker (from above)" on page 160 show

- 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. 4-55: 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.

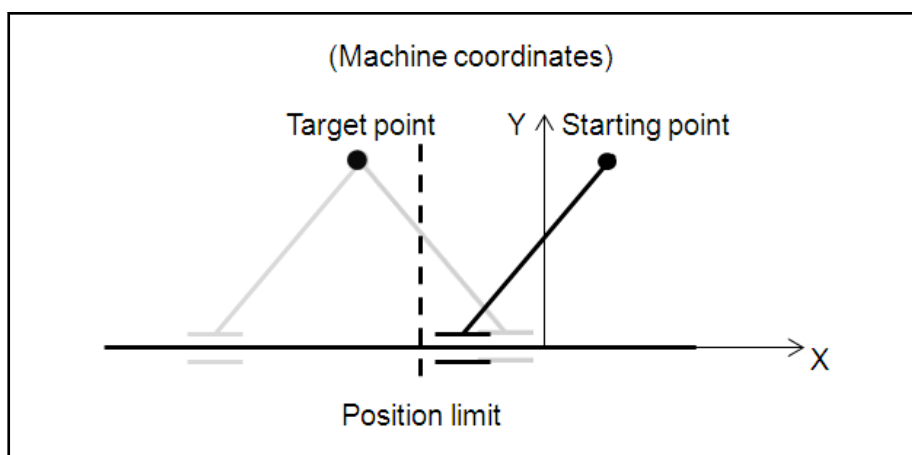


Fig. 4-56: Target point

The value in parameter K_3 controls the behavior:

- 0: Arm behavior is maintained. In this case, the arm travels to the left to reach 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 the arm travels to the right to reach 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 the arm travels to the left to reach 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, the arm travels to the right to reach the target point

Coordinates of a point

Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

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. 4-55: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation schemes

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. 4-56: 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, the arm travels to the left to reach 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, the arm travels to the right to reach 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. 4-57: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.25 Stewart platform

To use this transformation, enter transformation number 66 in parameter "[K-0-0031](#), Transformation scheme".

Transformation schemes

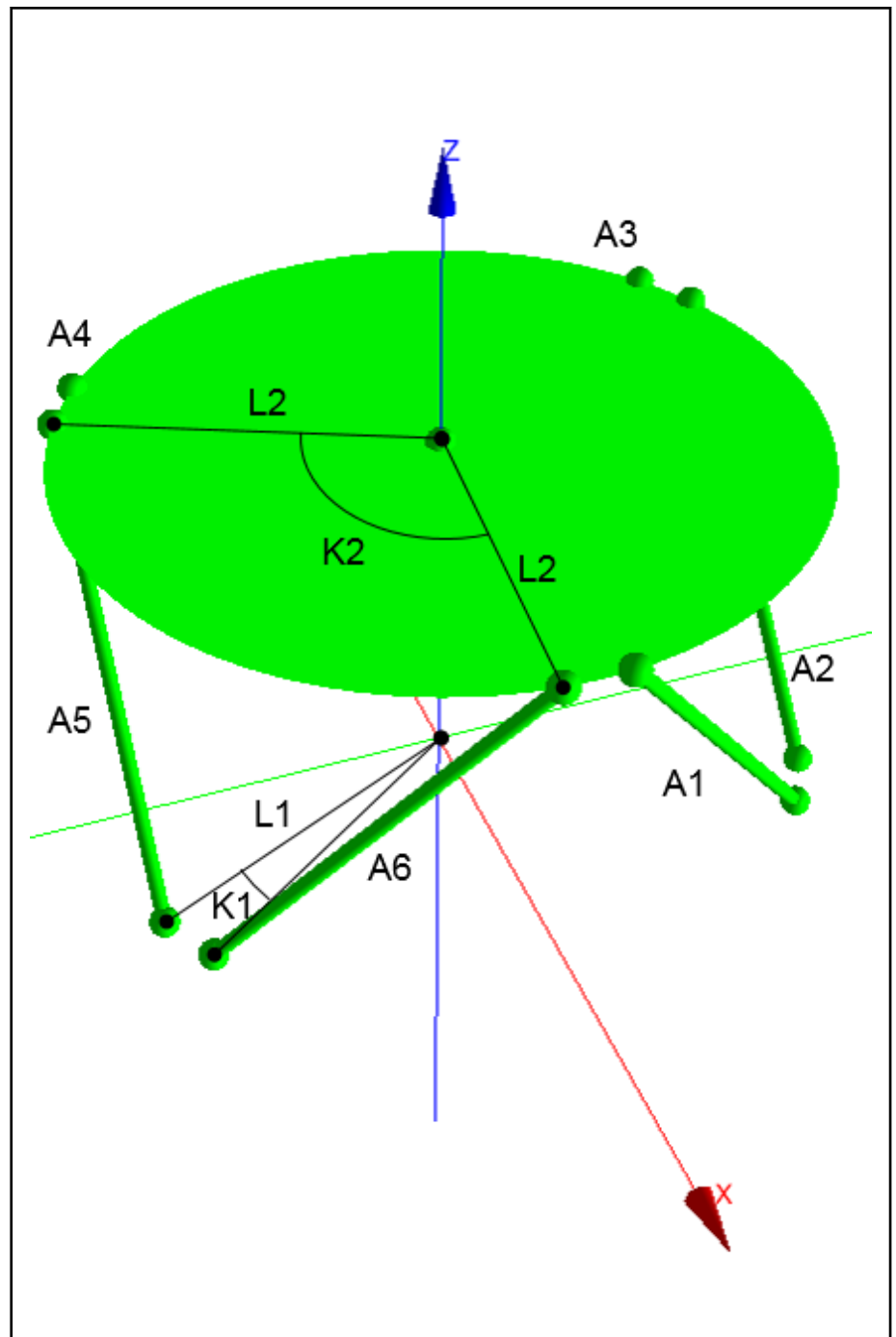


Fig. 4-57: View on the Stewart platform

Axis lengths and traversing direction

- The length of the axis is the distance of each axis from the mounting position of the basis to the mounting position of the platform. It is recommended to completely retract the axes A1 to A6 and to reference the axes on a length between the joints at the basis and the platform.
- The length of the axis increases in positive traversing direction.
- If $L1 = 0$ is specified, the points for the mounting positions of the basis can be directly specified in the lengths from L3 to L14. L1 and K1 are ignored in this case. The mounting positions of the basis are defined as follows:

Transformation schemes

- Mounting position A1: X = L3, Y= L4
- Mounting position A2: X = L5, Y= L6
- Mounting position A3: X = L7, Y= L8
- Mounting position A4: X = L9, Y= L10
- Mounting position A5: X = L11, Y= L12
- Mounting position A6: X = L13, Y= L14

Coordinates of a point Point (X, Y, Z, A, B, C, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling
A1	Positioning axis	Translatory absolute
A2	Positioning axis	Translatory absolute
A3	Positioning axis	Translatory absolute
A4	Orientation axis	Translatory absolute
A5	Orientation axis	Translatory absolute
A6	Orientation axis	Translatory absolute

Tab. 4-58: K-0-0008, Axis configuration list, axis assignment

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit
L1:	Radius of the basis	mm or inch (see K-0-0054)
L2:	Radius of the platform	mm or inch (see K-0-0054)
L3:	X-value of the mounting positions of the basis of axis 1	mm or inch (see K-0-0054)
L4:	Y-value of the mounting positions of the basis of axis 1	mm or inch (see K-0-0054)
L5:	X-value of the mounting positions of the basis of axis 2	mm or inch (see K-0-0054)
L6:	Y-value of the mounting positions of the basis of axis 2	mm or inch (see K-0-0054)
L7:	X-value of the mounting positions of the basis of axis 3	mm or inch (see K-0-0054)
L8:	Y-value of the mounting positions of the basis of axis 3	mm or inch (see K-0-0054)
L9:	X-value of the mounting positions of the basis of axis 4	mm or inch (see K-0-0054)
L10:	Y-value of the mounting positions of the basis of axis 4	mm or inch (see K-0-0054)

Transformation schemes

L11:	X-value of the mounting positions of the basis of axis 5	mm or inch (see K-0-0054)
L12:	Y-value of the mounting positions of the basis of axis 5	mm or inch (see K-0-0054)
L13:	X-value of the mounting positions of the basis of axis 6	mm or inch (see K-0-0054)
L14:	Y-value of the mounting positions of the basis of axis 6	mm or inch (see K-0-0054)

Tab. 4-59: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit
K1:	Angle between A1 and A2, A3 and A4, A5 and A6 on the basis	Degrees
K2:	Angle between A1 and A2, A3 and A4, A5 and A6 on the platform	Degrees
K3:	Accuracy of the approximation for the forward transformation If the value is 0.0, 0.01 is used as default value	mm or inch (see K-0-0054)
K4:	Number of iterations for the forward transformation after the forward transformation has been aborted with errors. If the value is 0, 32 is used as default value	Number

Tab. 4-60: K-0-0041, Transformation factors, axis configuration, axis coupling

Notes on the forward transformation

An approximation method of Newton-Raphson is used for the forward transformation. This forward transformation requires an abort conditions (how exact the position is to be computed in space). To exactly compute the position for the forward transformation, the required computation time increases.

The maximum number of iterations can still be selected. If the maximum number is reached, it is assumed that the axis angles cannot be transformed to a position in space and that the error F22B0021 is generated.

The transformation from axis to world coordinates (forward transformation) of the Stewart platform is ambiguous. This means that multiple world (space) coordinate positions exist for one set of axis angles. It is recommended to limit the traversing range in space via the parameters [K-0-0005](#) and [K-0-0006](#). Generally, data is provided with the mechanics.

As the forward transformation is an iterative procedure, this forward transformation requires more time than the other transformations. These increased computation times occur for the following use cases:

Transformation schemes

Increased computation times

Point in time	Context
- Control in phase 4 - Kinematics not grouped	Motion kernel actual value detection
- Kinematics grouped - Traversing instruction active with PTP	Motion kernel command value generation
- Kinematics grouped - One of the following functions is used: "Transform Point" function "Move Point" function "Get Mechanic" function	Calling task

4.2.26 4-axis Scara with crank

To use this transformation, enter transformation number 73 in parameter "K-0-0031, Transformation scheme".

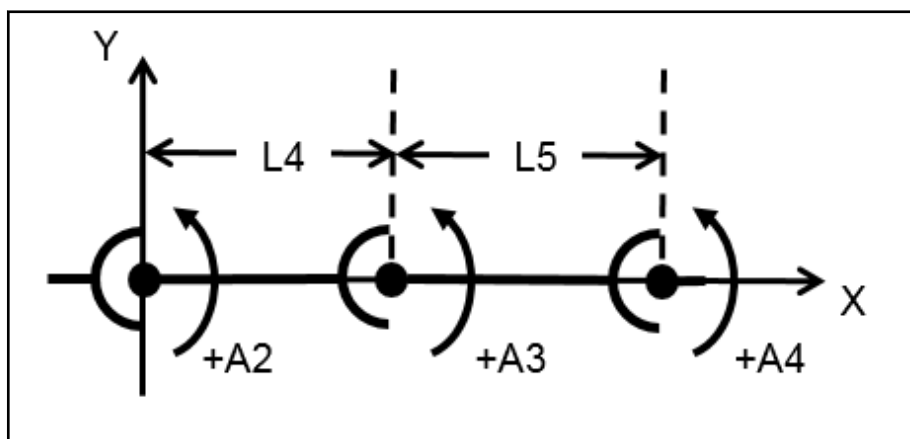


Fig. 4-58: Schematic view of 4-axis Scara with crank (x-y-plane)

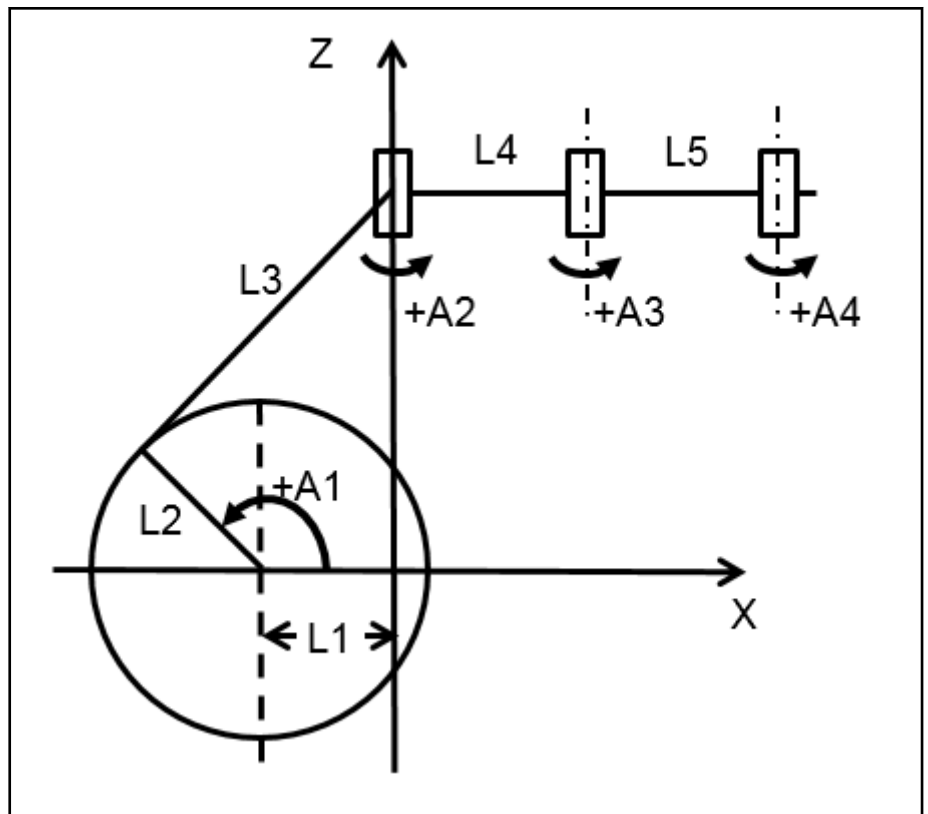


Fig. 4-59: Schematic view of 4-axis Scara with crank (x-z-plane)

Axis directions of rotation and zero positions

fig. 4-58 "Schematic view of 4-axis Scara with crank (x-y-plane)" on page 166: Displays the zero position of the axes A_2 , A_3 and A_4 in top view.

fig. 4-59 "Schematic view of 4-axis Scara with crank (x-z-plane)" on page 167: Displays the positions of all axes in side view.

- The axes A_1 , A_2 , A_3 and A_4 point towards the zero position to the direction of the positive x-axis. The positive direction of rotation is opposite to the rotation in clockwise rotation
- The figures also show arm lengths L_4 , L_5 and the positive rotary directions of the axes
- The lengths L_2 , L_3 , L_4 , L_5 always have to be entered as positive values
- L_4 , L_5 have to be unequal to zero
- The length L_1 defines the distance between the crank and the z-axis. Without loss of generality, it is assumed that the origin of the crank is located on the x-axis. Positive values of L_1 cause a displacement of the crank origin on the negative x-axis.
- The length L_2 specifies the length of the crank
- The length L_3 specifies the length of the pushrod
- The translatory motion caused by the crank is along the z-axis
- Comply with the relations between $DIST(L_1) < L_2 + L_3$.
- K_1 specifies the mechanic coupling of the axis A_2 with the axis A_3 .

Such a coupling exists if the axis A_3 moves along when traversing the axis A_2 without the measuring system of axis A_3 detecting this motion. This is generally the case if the motor for the axis A_3 is not directly on

Transformation schemes

the axis, but instead on its support. The coupling factor K_1 is inversely defined to the coupling factor K_2 .

*Example:*Coupling factor K_1

$K_1 = 0$: A motion of axis A_2 affects the position of the second arm in space: The angle between the first arm and the second arm does not change if the axis A_2 is moved. The position of the second arm changes analogously to the motion of axis 1 with regard to the x-axis.

$K_1 = 1$: A motion of axis A_2 does not affect the position of the second arm in space: The angle between the first arm and the second arm changes analogously to the motion of axis A_2 . The position of the second arm remains constant with regard to the x-axis.

- K_2 specifies the mechanic coupling of the axes A_2 and A_3 with the axis A_4 .

Such a coupling exists if the axis A_4 moves along when traversing the axes A_2 and A_3 without the measuring system of axis A_4 detecting this motion. This is generally the case if the motor for the axis A_4 is not directly on the axis, but instead on its support.

*Example:*Coupling factor K_2

$K_2 = 0$: A motion of axis A_2 or axis A_3 affects the orientation in space. Axis A_4 has to compensate the motion.

$K_2 = 1$: A motion of axis A_2 or axis A_3 does not affect the orientation in space. Axis A_4 does not have to compensate the motion.



The maximum traversing range of the axes A_2 and A_3 may not exceed or fall below ± 179 degrees. The traversing range of the axes A_1 and A_4 is not restricted.

The respective base coordinate system (BCS) is identified in [fig. 4-58 "Schematic view of 4-axis Scara with crank \(x-y-plane\)" on page 166](#) by the two main axes (x, y).

Arm behavior selection for PTP motions

If a motion [Point-to-point on page 41](#) [chapter 2.7.4 "Point-to-point motion type" on page 41](#) is executed and the point to be approached is not specified in the ACS, there are two methods to approach this point.

Transformation schemes

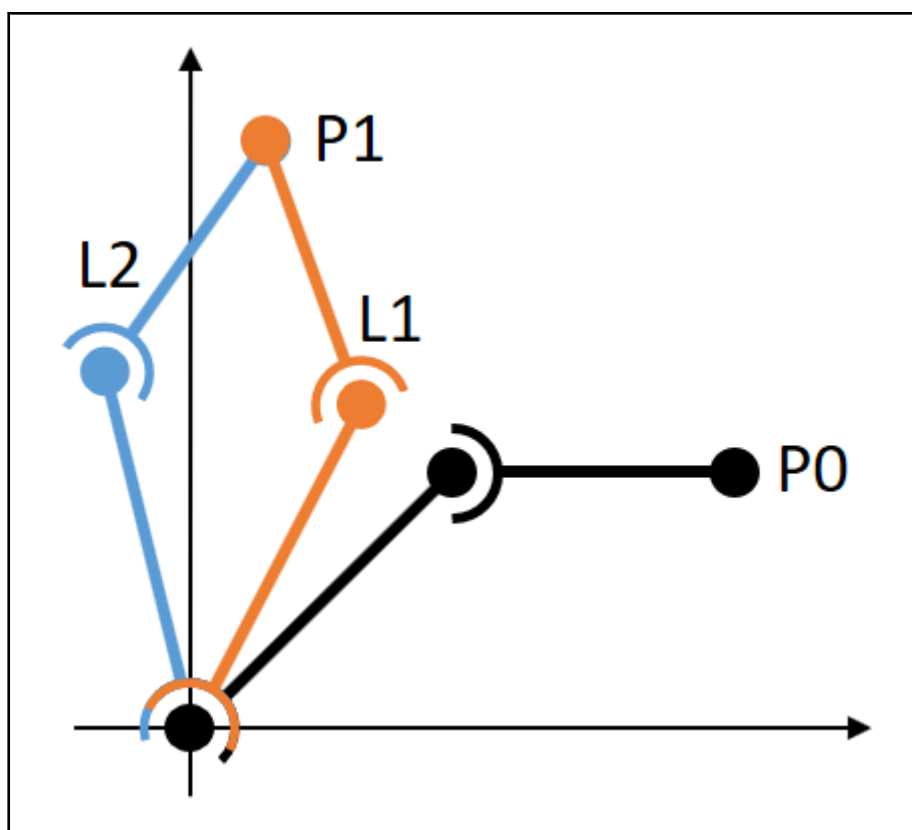


Fig. 4-60: Different arm behaviors of a Scara robot

With a point-to-point motion from point P0 to point P1 that was not specified in the ACS can be approached in two ways (L1 and L2).

The arm selection behavior can be configured using K_4 :

- 0: When the target point is approached, the arm behavior is not changed at the starting point in time
- 1: When the target point is approached, the motion of the first arm is minimized from the starting to the target point. The arm behavior selected ensures that axis 1 does not exceed the limits of $\pm 179^\circ$

Example:

Configuration₄

- $_4 = 0$: Point P1 is approached with L2. The arm behavior does not change during the motion
- $_4 = 1$: Point P1 is approached with L2. Thus, the consequence is that L2 has a lower motion of arm 1

Coordinates of a point

Point (X, Y, Z, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1:	Positioning axis	Rotary absolute	
A2:	Positioning axis	Rotary absolute	+/- 179 degrees

Transformation schemes

A3:	Positioning axis	Rotary absolute	+/- 179 degrees
A4:	Orientation axis	Rotary	Optional

Tab. 4-61: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Distance of the crank to the x-axis	mm or inch (see K-0-0054)	$\text{DIST}(L_1) < L_2 + L_3$
L2:	Length of crank	mm or inch (see K-0-0054)	≥ 0
L3:	Length of the pushrod	mm or inch (see K-0-0054)	≥ 0
L4:	Length of the first arm	mm or inch (see K-0-0054)	> 0
L5:	Length of the second arm	mm or inch (see K-0-0054)	> 0

Tab. 4-62: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit	Axis coupling
K1:	Mechanical coupling between axis 2 and 3	Factor	Optional
K2:	Mechanical coupling between the axes 2, 3 and 4	Factor	Optional
K4:	Arm behavior selection for PTP motions and point specification in non-ACS • 0: Arm behavior remains • 1: The arm behavior selected minimizes the motion of the first arm	Factor	Optional

Tab. 4-63: K-0-0041, Transformation factors, axis configuration, axis coupling

4.2.27 3-axis Canfield

To use this transformation, enter transformation number 81 in parameter "K-0-0031, Transformation scheme".

Transformation schemes

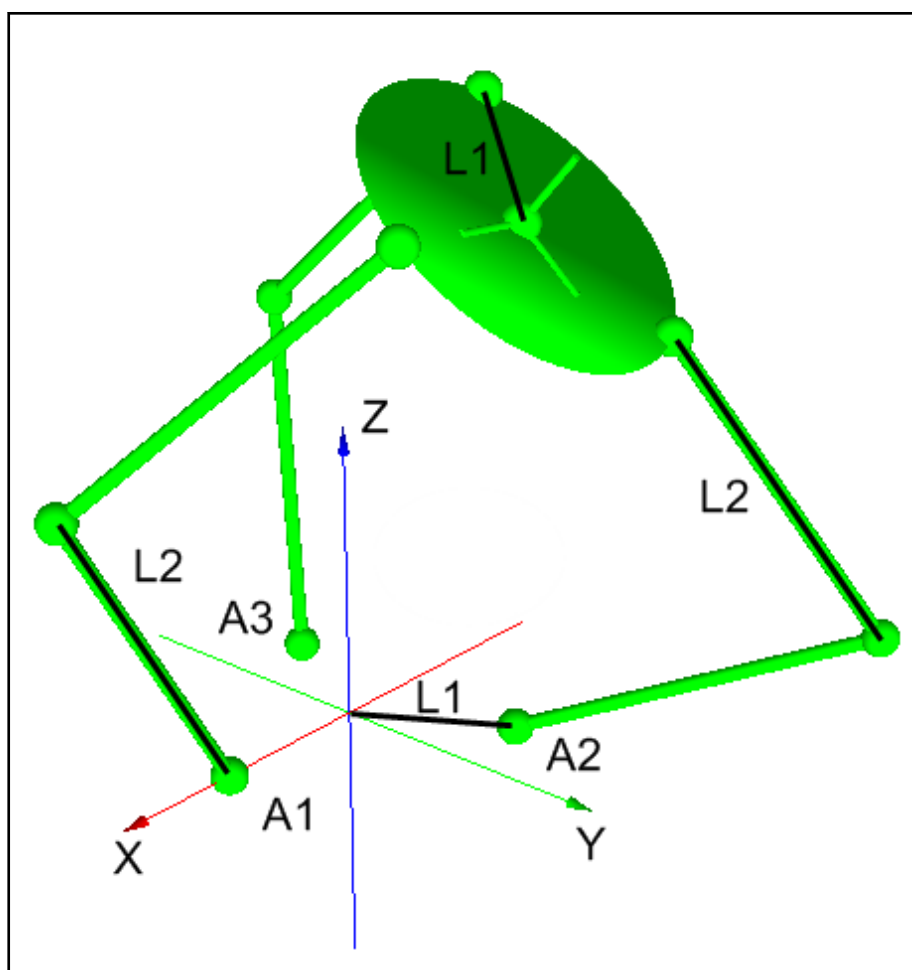


Fig. 4-61: View of a 3-axis Canfield

Axis directions of rotation and zero positions

- All axes are on 0 if they are on the X-Y-plane
- Mathematically, the positive direction of rotation of the axis is the negative direction of rotation. In the previous figure, all axes have negative position values
- A fourth axes can be mounted in the direction of the normal vector of the disk

Coordinates of a point

Point (Z, B, C, H1, ..., H10, B1, ..., B16)

Auxiliary axes that are not configured, are not shown as belt axes.

Parameter settings

Axis name	Axis type	Scaling	Notes
A1	Positioning axis	Rotary absolute	
A2	Orientation axis	Rotary absolute	
A3	Orientation axis	Rotary absolute	
A4	Orientation axis	Rotary	optional

Tab. 4-64: K-0-0008, Axis configuration list, axis assignment

All orientation axes of this transformation scheme are interpreted as auxiliary axes.

Transformation schemes

All other axes can be assigned with the following axes up to its maximum number of axes:

- Orientation axes interpreted as auxiliary axes
- Belt axes

Transformation length	Axis configuration	Measuring unit
L1:	Radius of the disk, distance of the axes to the center of the basic coordinate system	mm or inch (see K-0-0054)
L2:	Length of all arms	mm or inch (see K-0-0054)

Tab. 4-65: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Axis configuration	Measuring unit
K1:	Minimum plunge depth of the first safe zone	mm or inch (see K-0-0054)
K2:	Maximum plunge depth of the first safe zone	mm or inch (see K-0-0054)
K3:	Inclination of the first safe zone	Degrees
K4:	Yaw angle of the first safe zone	Degrees
K5:	Angle of the fourth axis of the first safe zone	Degrees
K6:	Minimum plunge depth of the second safe zone	mm or inch (see K-0-0054)
K7:	Maximum plunge depth of the second safe zone	mm or inch (see K-0-0054)
K8:	Inclination of the second safe zone	Degrees
K9:	Yaw angle of the second safe zone	Degrees
K10:	Angle of the fourth axis of the second safe zone	Degrees
K11:	Minimum plunge depth of the third safe zone	mm or inch (see K-0-0054)
K12:	Maximum plunge depth of the third safe zone	mm or inch (see K-0-0054)
K13:	Inclination of the third safe zone	Degrees
K14:	Yaw angle of the third safe zone	Degrees
K15:	Angle of the fourth axis of the third safe zone	Degrees

Tab. 4-66: K-0-0041, Transformation factors, axis configuration, axis coupling

Coordinates in the MCS

Coordinates in the MCS have shape (PD, inclination, yaw angle).

- PD: Plunge depth in mm or inch (see [K-0-0054](#)), distance of the TCP to the z-axis of the basic coordinate system (refer to the figure below)
- Inclination: Rotation by the y-axis of the basic coordinate system in degrees
- Yaw angle: Rotation by the z-axis of the basic coordinate system in degrees

Transformation schemes

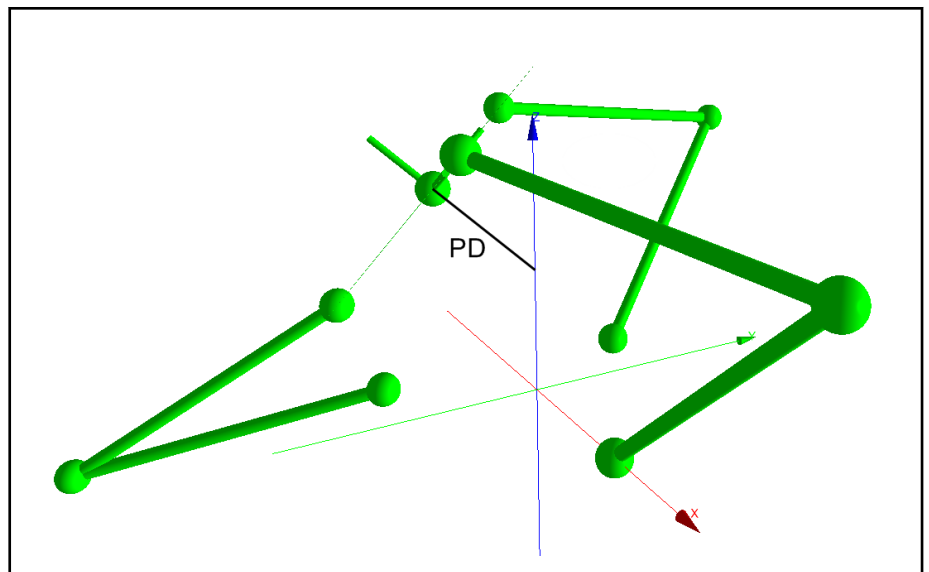


Fig. 4-62: Plunge depth

Recommendation

It is recommended to command this kinematic type with Move Direct Absolute and the end points in the MCS.

If this kinematic type is commanded with Move Linear Absolute, inclination and yaw angle are interpolated using Euler angles. This results in curve-shaped paths.

Example:

Motion from the MCS (85.0, 20.0, -179.0) to MCS (85.0, 70.0, 179.0) with Move Linear Absolute and Move Direct Absolute

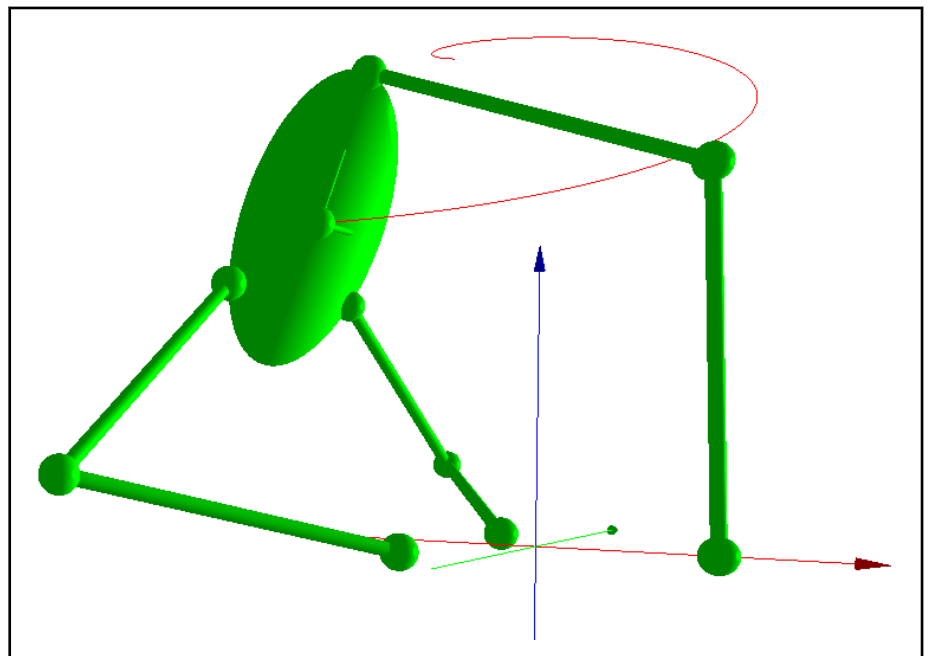


Fig. 4-63: Path with Move Linear Absolute

Transformation schemes

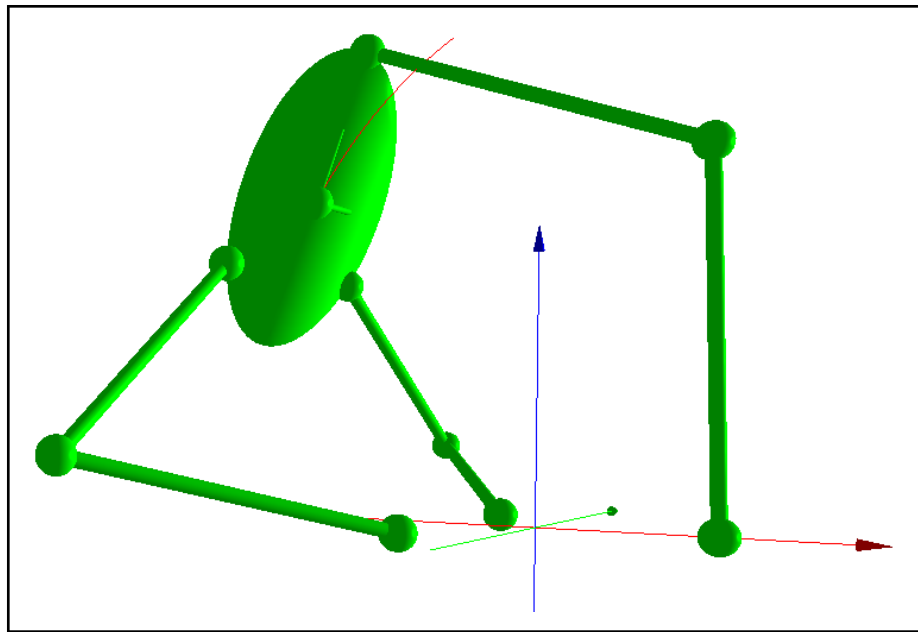


Fig. 4-64: Path with Move Direct Absolute

Safe zones of the transformation

It is possible to configure the transformation-specific safe zones. At least one point has to be in one safe zone to be valid. If not at least one point is in one safe zone, the error F22B0023 is generated.

Safe zone 1: Safe zone 1 is defined by the coupling factors 1 to 5.
If $K1 = 0$, safe zone 1 is not used.
The minimum position in the MCS is $(K1, -K3, -K4, -K5)$.
The maximum position in the MCS is $(K2, K3, K4, K5)$.

Safe zone 2: Safe zone 2 is defined by the coupling factors 6 to 10.
If $K6 = 0$, safe zone 2 is not used.
The minimum position in the MCS is $(K6, -K8, -K9, -K10)$.
The maximum position in the MCS is $(K7, K8, K9, K10)$.

Safe zone 3: Safe zone 3 is defined by the coupling factors 6 to 15.
If $K11 = 0$, safe zone 3 is not used.
Safe zone 3 exists three times: Once as configured, once twisted by 120° around the z-axis, once twisted by -120° around the z-axis.
The Minimum position in the MCS is $(K11, -K13, -K14, -K15)$.
The maximum position in the MCS is $(K12, K13, K14, K15)$.

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, ML_Robot, 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 be executed successfully.
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, page 27](#)).

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, page 27](#)).

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 at 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 the interface MC_KIN_REF, ML_Robot), this function block can be registered in the system with function block MC_SetKinTransform, ML_Robot,.

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 [User-defined transformation, page 393](#).

4.4 Extended 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 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 to define a transformation.

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

- Increased calculation accuracy by using the data type LREAL
- Handling transformation ambiguities
- Representing the mechanic view of the kinematics by using mechanical information (see [Mechanical information, page 180](#))
- Specifies the degree of freedom of a kinematics



Due to the previously mentioned 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, H1 ... H10 in exactly this order.

Missing degrees of freedom are not omitted in all interface methods of an extended, user-defined transformation, as is the case in a simple, user-defined transformation but are always available. These coordinates can be ignored in the methods.

The transformation reports the supported degrees of freedom to the system via the interface method `GetUsedAxes`, `MC_KIN_REF_EXT`, in `ML_Robot`.

At the specified degrees of freedom, points have the coordinate values of the point to be transformed. This applies to ACS as well as to BCS points. In the ACS, the coordinate values are the axis positions of the axis. In the BCS, the coordinate values refer to the following:

- X, Y, Z: Cartesian coordinates of the point
- A, B, C: Rotation of the point in space
 - A: Rotation around the X-axis Definition range: -180° - $+180^{\circ}$
 - B: Rotation around the Y-axis Definition range: -90° - $+90^{\circ}$
 - C: Rotation around the Z-axis Definition range: -180° - $+180^{\circ}$
- H1 ... H10: Auxiliary axes

In case of transformation that only have **one** degree of freedom in the orientation, it is recommended to not specify this degree of freedom as the actual degree of freedom but to consider the orientation axis as auxiliary axis. Disadvantages when specifying **one** degree of freedom as real orientation axis:

- The values A, B, C only have a limited definition range. If the orientation axis has a greater definition range or is scaled in modulo format, it is time-consuming to convert the transformation.
- In case of a spatial rotation of the kinematics, the spatial orientation is also transformed. Depending on the rotation, coordinate values of the orientation are sporadically interchanged or are not available anymore at all. As unavailable degrees of freedom of a transformation can be omitted for the user, the user cannot command the orientation anymore

If the orientation axis is defined as auxiliary axis, there are the following properties:

- There is not limited definition range for the axes. Axes scaled in module format or in absolute format can be used without any restrictions in the definition range
- The user always has to command the orientation axis with axis angles. The orientation is not transformed in case of a rotated kinematics. This causes less confusion among the users when commanding the orientation

Auxiliary axes are automatically copied between ACS and BSC by the system and do not have to be copied to the forward inverse position kinematics and the inverse position kinematics. However, the auxiliary axes can be calculated if they influence the TCP position.

Example:

Scara with two positioning axes and one orientation axis

A Scara robot has 2 positioning axes influencing X and Y and one orientation axis around Z (C-coordinate).

Transformation schemes

Due to the above mentioned reasons, it is not recommended to implement the interface method `GetUsedAxes` so that it returns `ML_KIN_USED_AXES.X_AXIS` OR `ML_KIN_USED_AXES.Y_AXIS` OR `ML_KIN_USED_AXES.Z_AXIS` (coordinate 1, 2 and 6).

It is better to only return `ML_KIN_USED_AXES.X_AXIS` OR `ML_KIN_USED_AXES.Y_AXIS` (auxiliary axes are always automatically defined).

Thus, the orientation axis values are not to be transmitted to coordinate 6 (`ML_KIN_USED_AXES.Z_AXIS`) but to coordinate 7 to benefit from the above-mentioned properties.

Consequently, ACS and BCS points contain:

- Coordinate X or Motor 1 at coordinate 1
- Coordinate Y or Motor 2 at coordinate 2
- Coordinate H1 or Motor 3 at coordinate 7

The transformation has to process these coordinates. All other coordinates can be ignored.

Methods to be implemented

To define an extended transformation, the interface `MC_KIN_REF_EXT`, `ML_Robot`, 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** Method to define the degrees of freedom of the transformation.

If a transformation has a degree of freedom around an axis, the corresponding value of data type `ML_KIN_USED_AXES`, `ML_Robot`, has to be returned.

In case of several degrees of freedom, several of these values require 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 be executed successfully.

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 the initialization cannot be completed successfully, an MLC error code method has to be returned:

- **DetermineMemory:** Using this method, transformation ambiguities can be resolved. A reference to an ACS point is transferred to the method. This point has the same ambiguous solution that is to be used for the next inverse position kinematics. Furthermore, information is provided in

Transformation schemes

the method whether the transformation can independently chose one of the ambiguous solutions in the next inversion position kinematics.

- **Forward:** method that converts coordinates from the axis coordinate system into coordinates of the base coordinate system ([Transformations, page 27](#)).

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, page 27](#)).

For this purpose, the method receives two references to the coordinates of the basic 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:** Method to receive mechanical information for a point in the ACS by the transformation.

This method contains

- A reference to a point in the ACS coordinate system
- A reference to a structure that is to be saved in the mechanical information
- The number of free elements in this structure
- A reference to the value that specifies how many elements of this structure were used
- As well as information if only changed values are to be returned

To complete the structure, the auxiliary function ML_AddMechanicData, ML_Robot is available.

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 it is, 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

Unlike for simple user-defined transformation, an MLC error code may be returned as return value of the functions in case of extended user-defined

Transformation schemes

transformation. This allows easier diagnostics for the user. An MLC error code has the form 16#Fxxxxxx. If an interface function was executed without errors, a code in form of 16#0xxxxxx has to be 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 the interface MC_KIN_REF_EXT, ML_Robot), this function block can be registered in the system with function block MC_SetKinTransformExt, ML_Robot,.

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 [Extended user-defined transformation, page 396](#).

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 38](#).

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

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

Transformation schemes

- **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 38](#) 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 38](#) 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. 4-67: Example of mechanical data

This data can be visualized as follows:

Transformation schemes

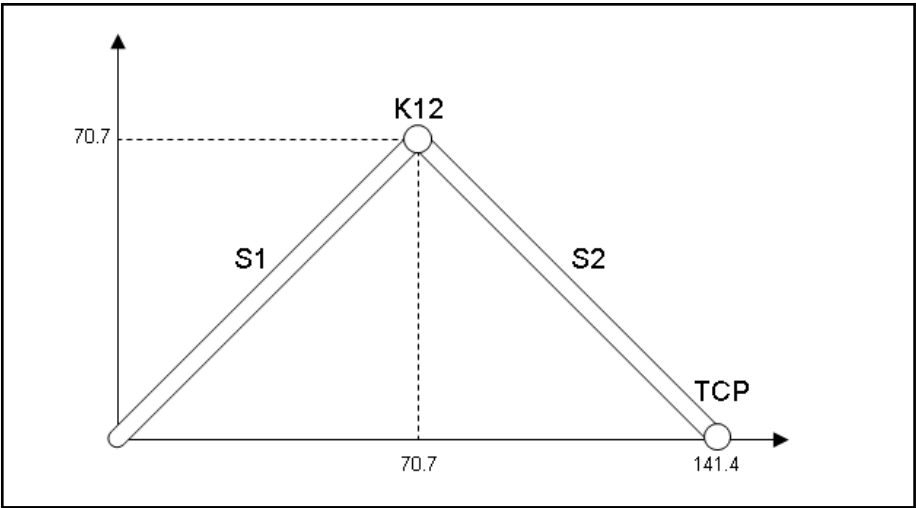


Fig. 4-65: Visualized data of GetMechanic

If the function [Get Mechanic on page 38](#) 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. 4-68: Example of mechanical data

5 ML_Robot.library

5.1 Introduction and overview



The library requires the ROB (robotics) function package for the control.

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. IndraLogic 2G, version 14 detects the hardware and provides the correct library.



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 23](#).

Library structure:

- Data types of the ML_Robot library, Data Types ([Data types, overview on page 298](#)):

Data types

- **Config:** Function block data types for configuration of a kinematics
- **Note:** Function block data types for information acquisition of a kinematics
- **Movement:** Function block data types for kinematics movement
- **Teach:** Data types for function block for teaching

- Definitions of global variables ([Global variables on page 184](#)):

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 188](#)):

Kinematics

- **Config:** Function blocks for configuration of a kinematics
- **Note:** Function blocks for kinematics information
- **Movement:** Function blocks for movements of a kinematics
- **Parameters:** Function blocks or parameter access to a kinematics
- **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 using this library

ML_Robot.library

5.2 Global variables

5.2.1 Introduction and overview



The library ML_Robot requires the ROB (robotics) function package for the control

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 [GVL_RobotConstant](#), [global constants on page 185](#))

it is possible to access indexed kinematic data via

- **Function variable access** (see [DV_Kin\[\] - Kinematic data via function variable access on page 186](#))
- **Shared memory access** (see [KinData\[\] - Kinematic data in the shared memory on page 185](#))



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 irrespectively 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 GVL_RobotConstant, global constants

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

Variable name	Type	Description	MLC			
			L25	L45/ L65/ L75	XM2	VPx
MB_MIN_KIN_NUMBER	INT	Smallest kinematic number	1			
MB_MAX_KIN_NUMBER	INT	Greatest kinematic number	4	16	16	16
MB_NUMBER_OF_KIN	INT	MB_MAX_KIN_NUMBER - MB_MIN_KIN_NUMBER + 1	4	16	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		255			
KIN_OBJ_2_OBJECT	Array [0..16]] of OBJECTS		[0,1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15,16]			

Tab. 5-1: Global constants of the ML_Robot library

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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_KINEMATICS_DATA_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_KINEMATICS_DATA_SM;
...
END_VAR
```



Please find the addresses "%MBaaa" of KinData in the library manager of ML_Robot, GVL_Robot documentation.

Also refer to

- [ML_KINEMATICS_DATA_SM on page 307](#)

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

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.

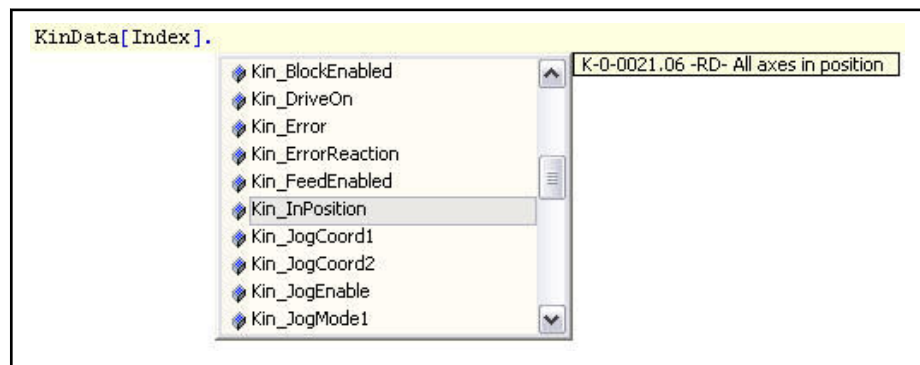


Fig. 5-1: IntelliSense of a variable in the shared memory



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 for example 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.

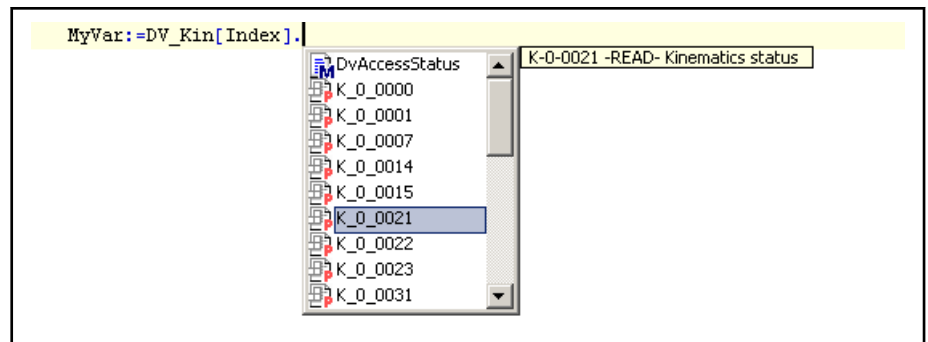


Fig. 5-2: Function variable access using the IntelliSense functionality



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 277](#)

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



The library requires the ROB (robotics) function package of the control

ML_Robot function blocks:

Area of application (folder in library)	
Function block	Description
Function blocks to configure a kinematics (Kinematic/Config)	
MC_GroupReset, page 205	Resets all kinematic errors
MC_SetCartesianTransform, page 206	Defines a Cartesian product coordinate system
MC_SetKinTransform, page 208	Registers a user-defined transformation for a kinematics
MC_SetKinTransformExt, page 210	Registers an extended user-defined transformation for a kinematics
MC_UngroupAllAxes, page 211	Reads K-0-0008 and removes all kinematic axes
ML_AddMechanicData, page 212	Inserts an entry in the ML_KIN_MECHANIC_DATA structure
ML_GroupAllAxes, page 213	Reads K-0-0008 of the selected kinematics and adds all kinematic axes
ML_GroupSetSafeZone, page 214	Configures a safe zone for the kinematics
ML_RobotSetMoveID, page 223	Changes the current motion identification number
ML_SetBeltConfiguration, page 217	Configures a belt
ML_SetBeltDesyncRelConfiguration, page 216	Configures relative desynchronization of a belt

ML_SetBeltErrorReaction, page 218	Configures the error reaction of a belt
ML_SetCylindricTransform, page 220	Defines a cylindrical product coordinate system
ML_SetRotaryTableConfiguration, page 222	Configures a rotary table
Function blocks to collect information about a kinematics (kinematic/Info)	
MC_GroupReadActualPosition, page 224	Reads the current position of the kinematics
ML_RobotGetBeltConfiguration, page 226	Returns the current belt configuration
ML_RobotGetBeltDesyncRelConfiguration, page 227	Reads the configuration for relative desynchronization of a belt
ML_RobotGetBeltErrorReaction, page 228	Returns the configured belt error reaction of the robot
ML_RobotGetCmdInfo, page 229	Returns information on all commands in the block buffer
ML_RobotGetMechanic, page 231	Returns the mechanical kinematic data
ML_RobotGetMoveID, page 232	Returns the current motion identification number
ML_RobotGetPerformance, page 233	Returns the performance data of the kinematics
ML_RobotGetSafeZones, page 234	Returns the safe zone of a kinematics
ML_RobotGetTrafoValues, page 236	Returns the current configuration of a product coordinate system or of the basic coordinate system
ML_RobotMovePoint, page 237	Returns the current position of a point in space
ML_RobotReadCyclicChannel, page 238	Reads the current cyclically commanded position
ML_RobotSetPerformanceEnable, page 240	Changes the activation state of the performance measurement
ML_RobotTransformPoint, page 241	Transforms a point from one coordinate system to another coordinate system
ML_SwitchOnPath, page 242	Allows switching on a path
Function blocks to move a kinematics (kinematic/movement)	
MC_GroupContinue, page 244	Restarts a kinematic motion interrupted by the MC_GroupInterrupt function block
MC_GroupInterrupt, page 245	Interrupts the kinematic motion
MC_GroupStop, page 246	Controlled stopping of a kinematics
MC_MoveCircularAbsolute, page 248	Circularly moves a kinematics from the last commanded TCP position via an intermediate point to a specified absolute target position in the specified coordinate system
MC_MoveCircularRelative, page 250	Circularly moves a kinematics from the last commanded TCP position via an intermediate point to a specified absolute target position in the specified coordinate system
MC_MoveDirectAbsolute, page 252	Moves a kinematics point-to-point from the last commanded position of the TCP to a specified absolute target position in the coordinate system
MC_MoveDirectRelative, page 255	Moves a kinematics point-to-point from the last commanded position of the TCP to a specified relative target position in the coordinate system
MC_MoveLinearAbsolute, page 257	Moves a kinematics linearly from the last commanded position of the TCP to a specified absolute target position in the coordinate system

ML_Robot.library

MC_MoveLinearRelative, page 260	Moves a kinematics linearly from the last commanded position of the TCP to a specified relative target position in the coordinate system
ML_BeltDeSynchronisationWithLimits, page 262	Desynchronizes from a belt with motion limit values
ML_BeltSynchronisationWithLimits, page 263	Synchronizes to a belt with motion limit values
ML_GroupJogCont, page 265	Continuously moves a kinematics in one direction
ML_GroupJogStep, page 267	Moves a kinematics one stop in a certain direction
ML_GroupWait, page 269	Kinematics waits for the specified time on the last commanded position
ML_RobotMoveJumpAbsolute, page 274	Moves a kinematics from the last commanded position of the TCP to a specified absolute target position in the coordinate system using a jump motion via starting, maximum and end height
ML_RobotOpenCyclicChannel, page 270	Opens a cyclic command value channel
ML_RobotOpenDirectCyclicChannel, page 272	Opens a cyclic command value channel for PTP motions
ML_RobotWriteCyclicChannel, page 273	Writes a cyclic command value
Function blocks for parameter access to a kinematics (kinematic/parameter)	
MB_GroupReadListParameter, page 278	Returns the value of a list parameter specified by the user
MB_GroupReadParameter, page 280	Returns the value of a DINT parameter specified by the user
MB_GroupReadRealParameter, page 281	Returns the value of a REAL parameter specified by the user
MB_GroupReadSercosAttribute, page 282	Allows reading the Sercos attributes of a K-parameter
MB_GroupReadSercosDataStatus, page 283	Returns the status of a K-parameter specified by the user
MB_GroupReadStringParameter, page 284	Returns the value of a STRING parameter specified by the user
MB_GroupWriteListParameter, page 285	Writes memory content of a defined length (NoOfBytes) provided from the initial address (ValueAdr) to a user-defined parameter
MB_GroupWriteParameter, page 287	Writes a DINT value in a parameter specified by the user
MB_GroupWriteRealParameter, page 288	Writes a REAL value in a parameter specified by the user
MB_GroupWriteStringParameter, page 289	Writes a STRING value in a parameter specified by the user
Function blocks to save and reuse kinematic positions as points (kinematic/TeachIn)	
ML_GroupDeletePointFile, page 290	Deletes a kinematic point file
ML_GroupDeletePoints, page 291	Deletes teach-in kinematic points
ML_GroupLoadPointFile, page 292	Loads a kinematic point file
ML_GroupReadNextPoint, page 293	Searches for the next teach-in point and returns the data of the next kinematic point
ML_GroupReadPoint, page 295	Returns teach-in data of a kinematic point
ML_GroupSavePointFile, page 296	Saves a kinematic point file
ML_GroupWritePoint, page 297	Teaches a kinematic point

Tab. 5-2: ML_Robot library 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

Using function blocks, it is possible to encapsulate complex tasks so they can be used again 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 final and tasks which require permanent intervention once they are started.

State-controlled	If a function block always repeats its task after switch-on after it has reached a defined state, this function block processes "state-controlled".
Edge-controlled	If a function block fulfills its task exactly once after being switched on, this function block processes "edge-controlled".
Final processing	If a block can be finally process its order and afterwards stop to work, this is called "final processing".
Permanent processing	If a block can never conclude its order definitively but continuous intervention is necessary, this is defined as "permanent processing".

In combination, there are four function block types possible:

Control	Processing	Example
State-controlled (Enable)	Final (Done)	Continuous reading of a parameter If reading is completed, the function block has completed its task. The function block 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 disabled the control
Edge-controlled (Execute)	Final (Done)	One-time writing of a parameter If writing is completed, the function block has completed its task. A new edge is required at the control input (Execute) to start writing again, 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 disabled via the interfaces of this function block, e.g. MC_MoveVelocity

Tab. 5-3: Overview on function block types

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 To mark whether a function block processes finally or permanently, different variable names are used for the outputs indicating the processing state.



- Done = Final processing
- In..... = Permanent processing (e.g., InOperation, InSync, InVelocity, InGear, InTorque...)

The optional "Active" output can be used with final and permanent processing function blocks. However, the behavior differs!

All outputs for the error identification (Error, ErrorID, ErrorIdent) can be found with final and permanent processing function blocks. Both show the same behavior.

Standardized identifiers

Identifier	Description
Inputs	
Enable	Enable input of state-controlled function blocks
Execute	Enable input of edge-controlled function blocks. Other function blocks can interrupt this function block
ExecuteLock	Enable input of edge-controlled function blocks. As long as this input is TRUE, the function block cannot be interrupted by others
Outputs	
Active	Output marking the processing time
Done	Processing completed successfully and the data outputs are valid
In.....	The output signals that the function block is processing its task and that the data outputs are valid
Shutdown (optional)	The output signals that the function block is currently reaching a defined final state (e.g., it stops axes, releases resources...). The function block has to be called until "Shutdown" becomes FALSE
CommandAborted	Output indicates that the function block was interrupted (e.g. by another function block)
Error	Process completed with error
ErrorID	Output for rough error classification
ErrorIdent	Output for detailed error classification

Tab. 5-4: Overview on standardized identifiers

Inputs and outputs of state-controlled function blocks, final processing

Control/ processing	I/O	Variable name	Description
State- controlled/ final	I	Enable	The input variables are registered with a positive edge of "Enable". New input values are only applied at another positive edge at "Enable". "Enable" has to be TRUE during the complete function block processing! If "Enable" is disabled, processing is interrupted and the "Done", "Active", "CommandAborted" and "Error" outputs are set to FALSE. If required, "Shutdown" signals that further signals are necessary to make the function block to reach its final state defined
	O	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, afterwards, "Enable" is still TRUE, the function block starts its processing again
	O	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediate state. Possible preprocessing are not marked with this output! "Done", "Error" and "CommandAborted" are FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to disable "Enable" to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	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 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 function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was 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 function block has to be called until "Shutdown" becomes FALSE

Tab. 5-5: I/O of state-controlled function blocks, final processing



If it is required that certain inputs are not only applied at the edge at "Enable", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!



For function blocks with "Shutdown", define the behavior of the function block when "Enable" is set to TRUE while "Shutdown" remains set (retrigger).

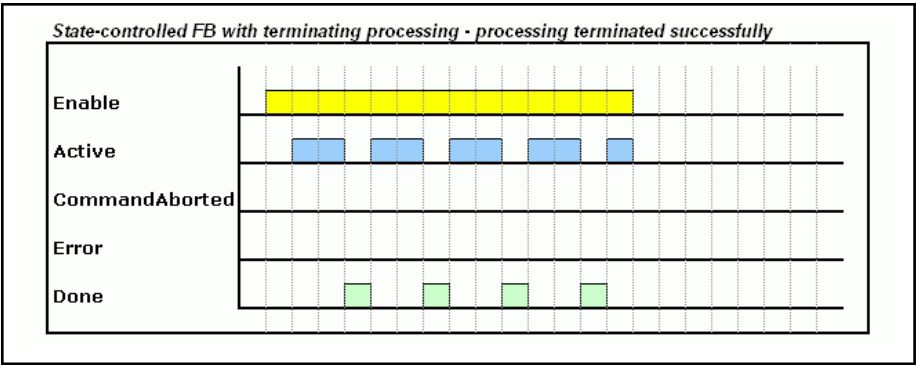


Fig. 5-3: I/O of state-controlled function blocks, final processing without "Shutdown". Processing successfully completed

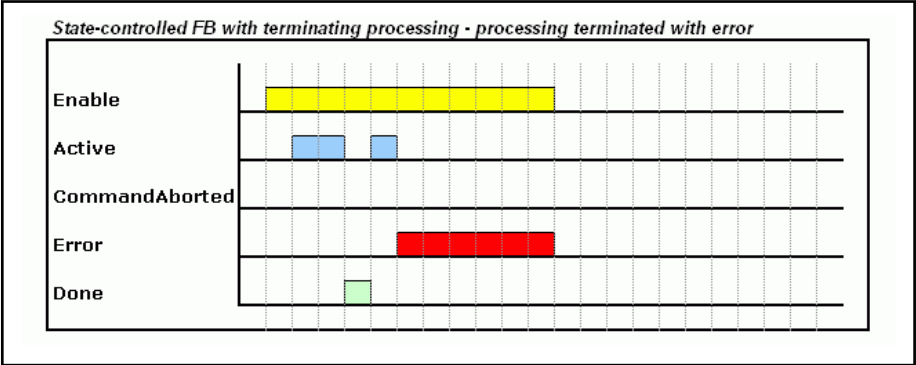


Fig. 5-4: I/O of state-controlled function blocks, final processing without "Shutdown". Processing completed with error

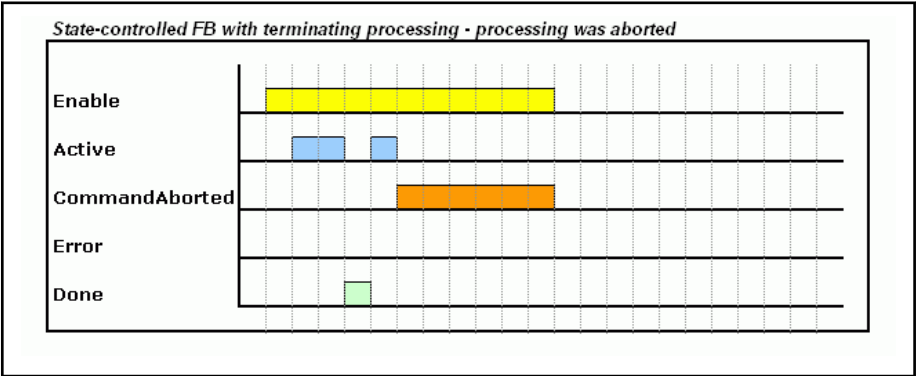


Fig. 5-5: I/O of state-controlled function blocks, final processing without "Shutdown". Processing interrupted

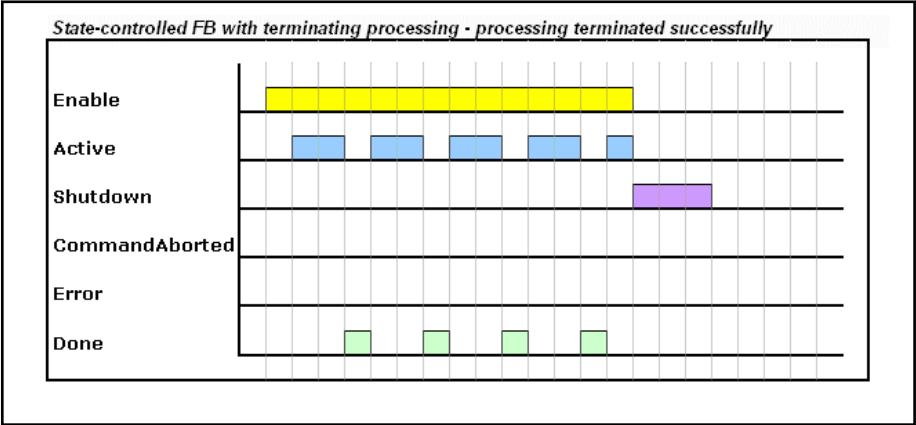


Fig. 5-6: I/O of state-controlled function blocks, final processing with "Shutdown". Processing successfully completed

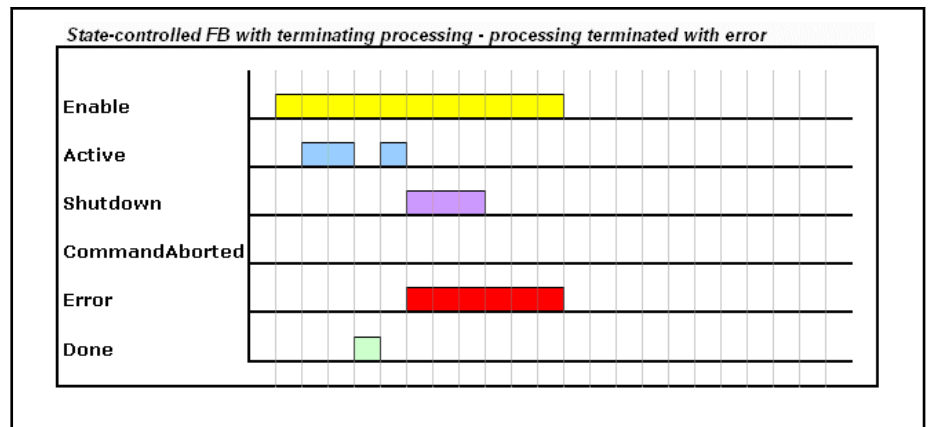


Fig. 5-7: I/O of state-controlled function blocks, final processing with "Shutdown". Processing completed with error

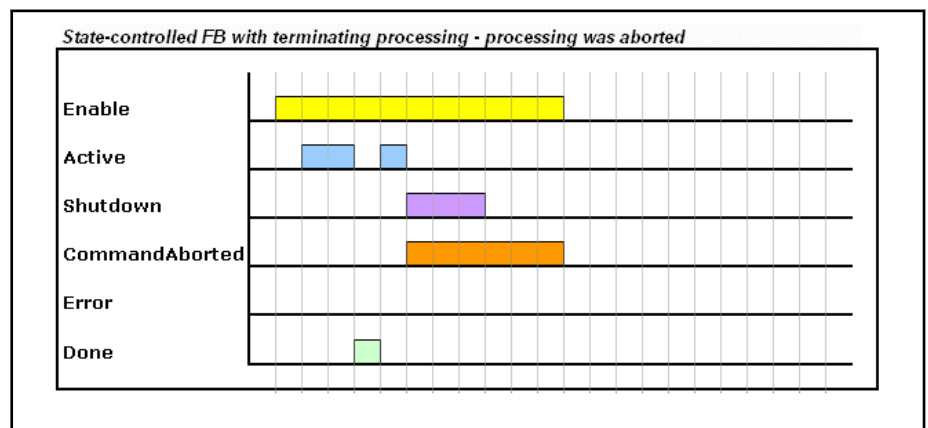


Fig. 5-8: I/O of state-controlled function blocks, final processing with "Shutdown". Processing was interrupted

ML_Robot.library

Inputs and outputs of edge-controlled function blocks, final processing

Control/ processing	I/O	Name	Description
Edge- controlled/ final	I	Execute	The input variables are registered with a positive edge of "Execute". New input values are only applied at another positive edge at "Execute". The change of edges at "Execute" suffices to start the block. For further processing, the state of "Execute" is not important. With a new edge change during the processing, the previous task is discarded, the inputs are applied 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 function block documentation provides clarity on this issue
	O	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 the 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 function block has to be called until "Shutdown" becomes FALSE
	O	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediate state. Possible preprocessing are not marked with this output! "Done", "Error", "Shutdown" and "CommandAborted" are FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time after the completion of the task (signaled via "Done") to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	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 the control input "Execute" is FALSE at the time when the order is interrupted, "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 function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was interrupted due to an error and is now in a final state. If the 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 function block has to be called until "Shutdown" becomes FALSE

Tab. 5-6: I/O of state-controlled function blocks, final processing



If certain inputs are not only applied with the 0/1 edge at "Execute", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!



For function blocks with "Shutdown", define the behavior of the function block when a new edge is detected at "Execute" while "Shutdown" remains set (retrigger).

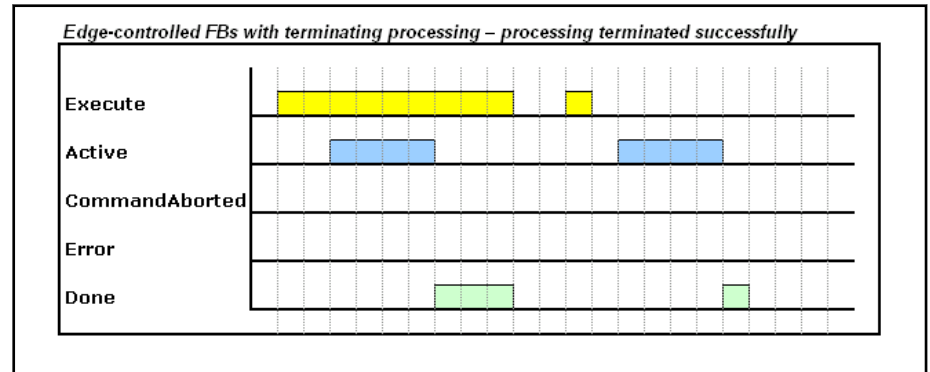


Fig. 5-9: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing successfully completed

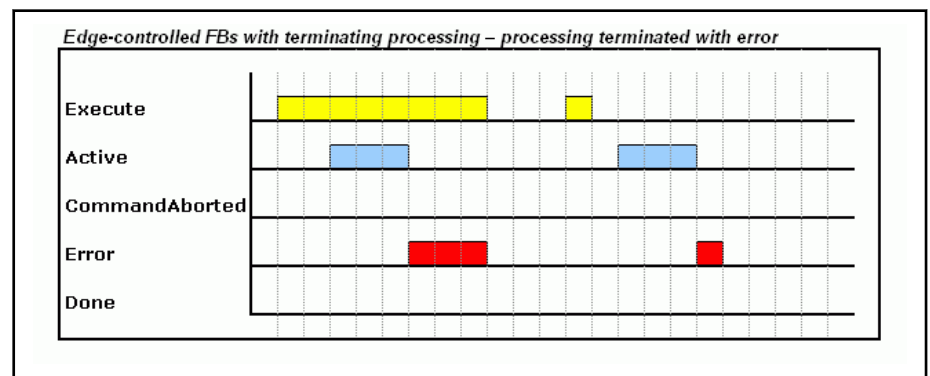


Fig. 5-10: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing completed with error

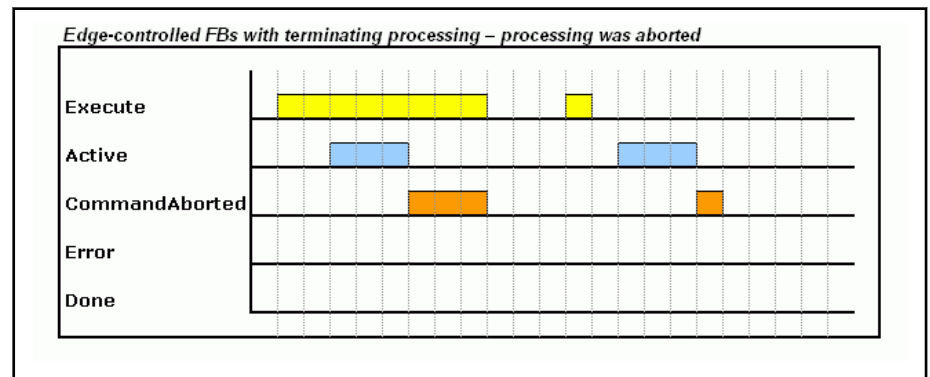


Fig. 5-11: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing interrupted

ML_Robot.library

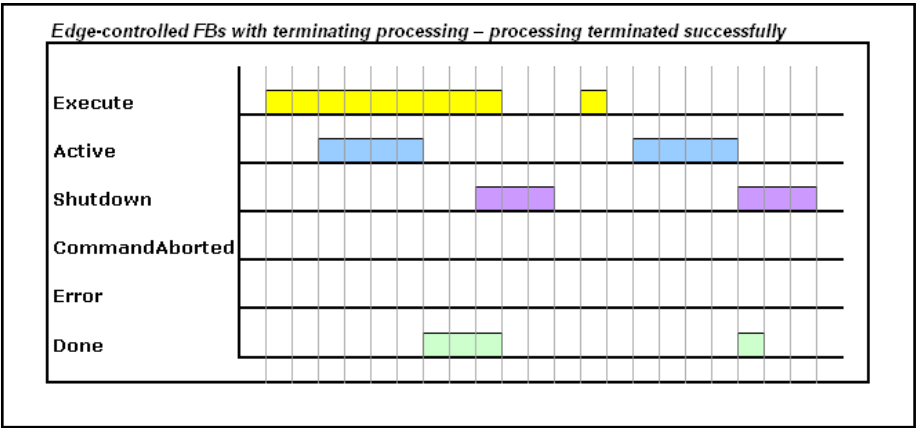


Fig. 5-12: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing successfully completed

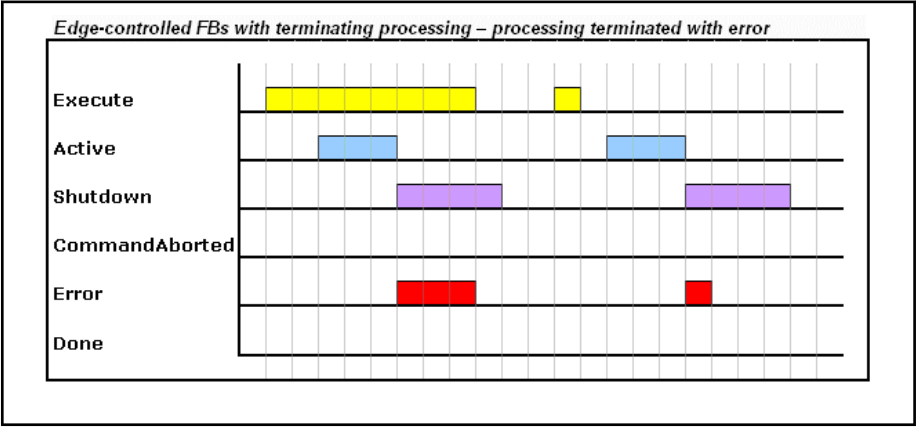


Fig. 5-13: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing completed with error

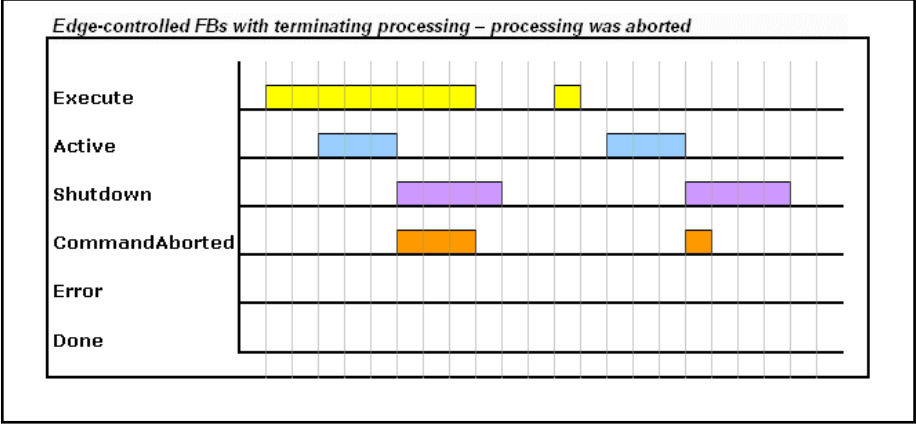


Fig. 5-14: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing interrupted

Inputs and outputs of state-controlled function clocks, permanent processing

Control/ processing	I/O	Variable name	Description
State- controlled/ permanent	I	Enable	The input variables are registered with a positive edge of "Enable". New input values are only applied at another positive edge at "Enable". "Enable" has to be TRUE during the complete function block processing! If "Enable" is disabled, processing is interrupted and the "In....", "Active", "CommandAborted" and "Error" outputs are set to FALSE. If required, "Shutdown" signals that further signals are necessary to make the function block to reach its final state defined
	O	In....	If "In...." Is TRUE, the function block reached its target, but it is still in intervention to "keep" the achieved. Thus, it is thus in a permanent final state. Data outputs are valid now. Since the function block remains in intervention, "Active" also has to remain TRUE. "Error" and "CommandAborted" are FALSE! As long as "Enable" is TRUE, "In...." also remains TRUE
	O	Active	If Active is TRUE, the function block processes its actual task. Possible preprocessing are not marked with this output! Since the function block remains in intervention permanently, "Active" remains TRUE until the function block is switched off via the control input "Enable" or until it is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" have to be FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to disable "Enable" to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	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 function block has to be called until "Shutdown" becomes FALSE
	O	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 function block has to be called until "Shutdown" becomes FALSE

Tab. 5-7: I/O of state-controlled function blocks, final processing



If it is required that certain inputs are not only applied at the edge at "Enable", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!



For function blocks with "Shutdown", define the behavior of the function block when "Enable" is set again to TRUE while "Shutdown" remains set (retrigger).

ML_Robot.library

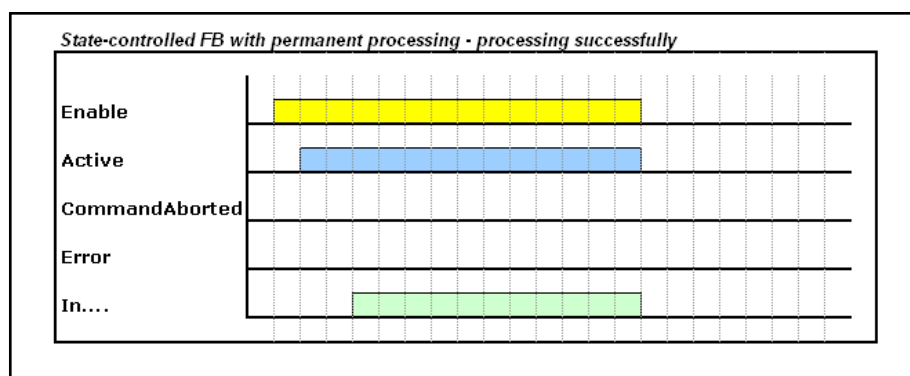


Fig. 5-15: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing successful

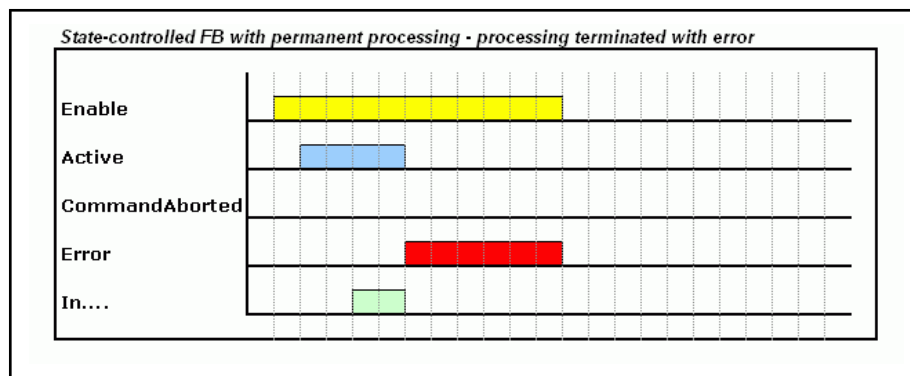


Fig. 5-16: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

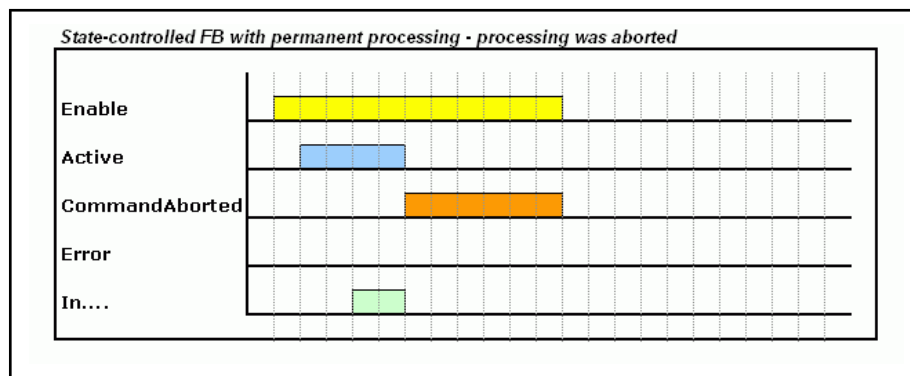


Fig. 5-17: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

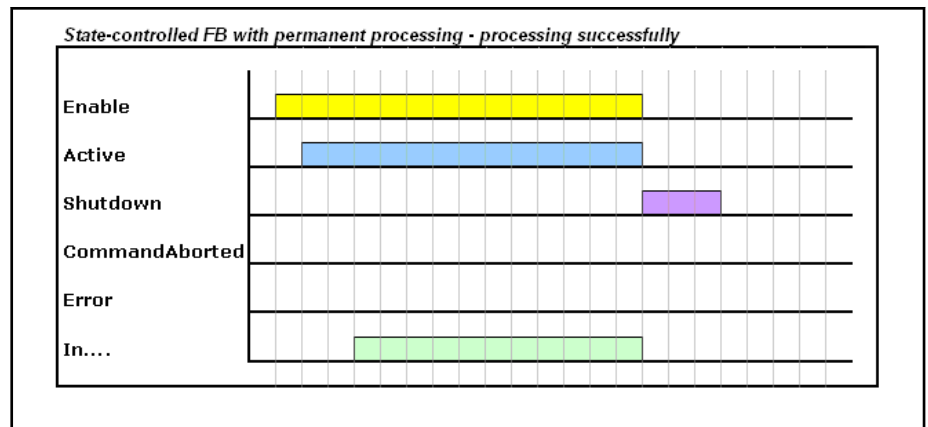


Fig. 5-18: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing successful

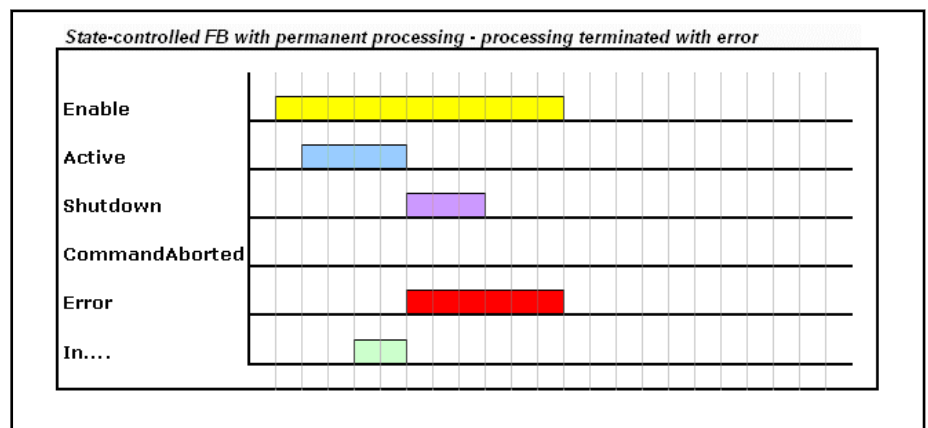


Fig. 5-19: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

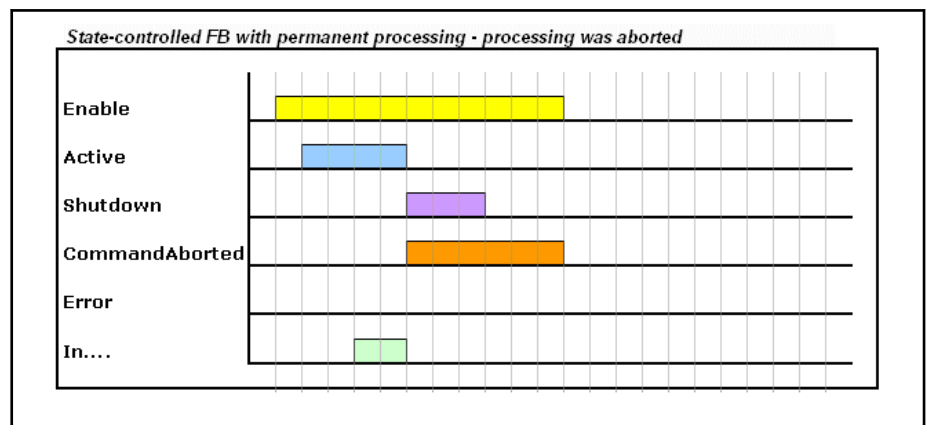


Fig. 5-20: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

ML_Robot.library

Inputs and outputs of edge-triggered function blocks, permanent processing

Control/ processing	I/O	Variable name	Description
Edge- controlled/ permanent	i	Execute	The input variables are registered with a positive edge of "Execute". New input values are only applied at another positive edge at "Execute". The change of edges at "Execute" suffices to start the block. For further processing, the state of "Execute" is not important. By changing the edge again 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 function block provided with a "Shutdown" output, retriggering could not be allowed
	O	In....	If "In...." is TRUE, the function block has reached its goal but is still in intervention in order to "keep" what it has achieved and is thus in a permanent final state. Data outputs are valid now. Since the function block remains in intervention, "Active" also has to remain TRUE. This even applies when "In...." decreases again. "Error" and "CommandAborted" are FALSE! If the 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
	O	Active	If "Active" is TRUE, the function block processes its actual task. Possible preprocessing are not marked with this output! Since the function block remains in intervention permanently, "Active" remains TRUE until the block is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" as well as "Shutdown" has to be FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to reach a defined final state and to release the resources used after the function block has canceled its task ("CommandAborted" or "Error" = TRUE). The function block has to be called until "Shutdown" becomes FALSE!
	O	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 the control input "Execute" is FALSE at the time when the order is interrupted, "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 function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was interrupted due to an error and is now in a final state. If the 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 function block has to be called until "Shutdown" becomes FALSE

Tab. 5-8: I/O of edge-controlled function blocks, permanent processing



If certain inputs are not only applied with the 0/1 edge at "Execute", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!



For function blocks with "Shutdown", define the behavior of the function block when a new edge is detected at "Execute" while "Shutdown" remains set (retrigger).

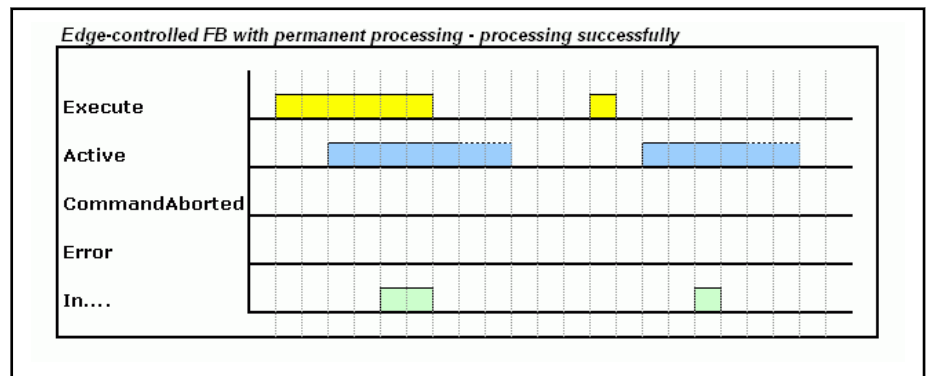


Fig. 5-21: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing successful

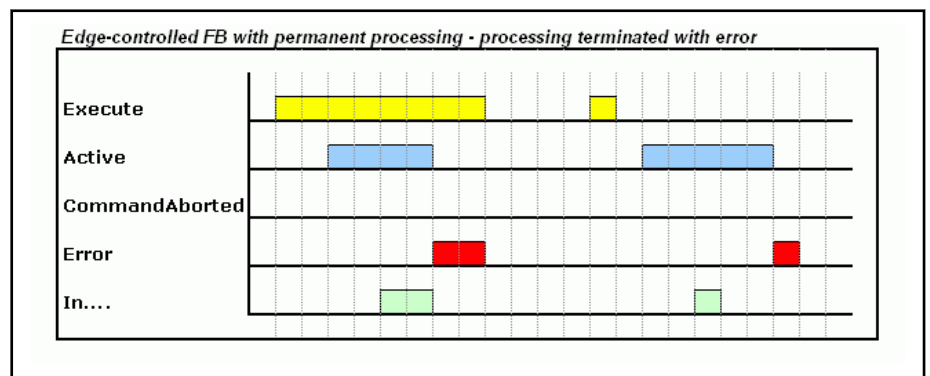


Fig. 5-22: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

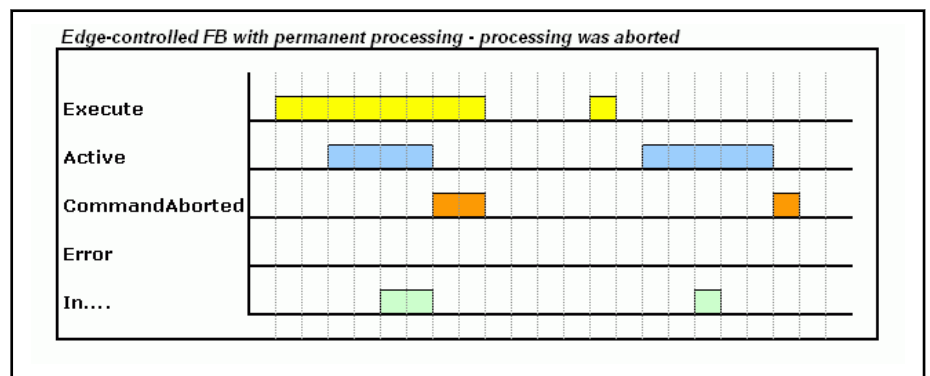


Fig. 5-23: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

ML_Robot.library

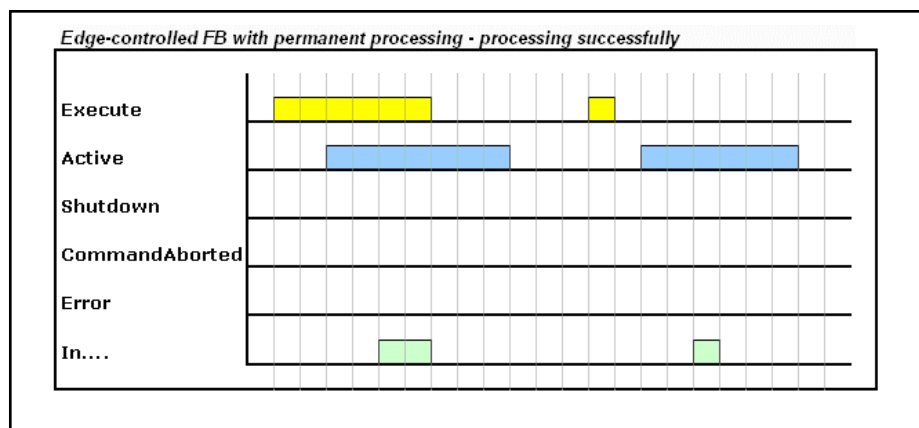


Fig. 5-24: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing successful

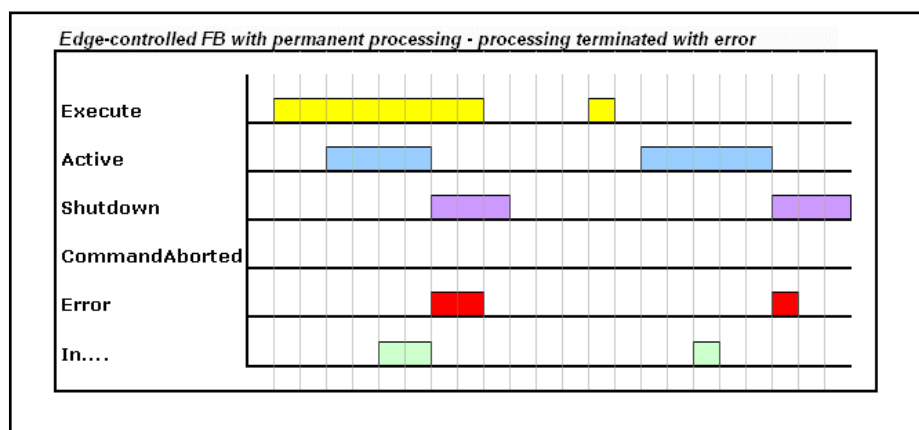


Fig. 5-25: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

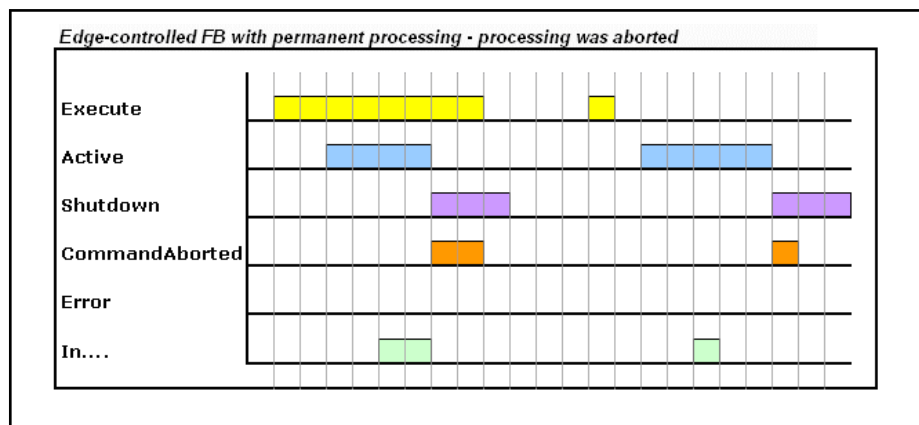


Fig. 5-26: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

5.3.3 InBuffer output at ML_Robot function blocks

ML_Robot function blocks representing the commands and which are saved in the block buffer have an InBuffer output.

(Refer to [Command state](#) on page 36).

This output indicates that the command is in the block buffer and that it is set to TRUE for at least one cycle.

The following diagrams display the output behavior relating to the other inputs and outputs.

Behavior of the InBuffer output - Processing successful

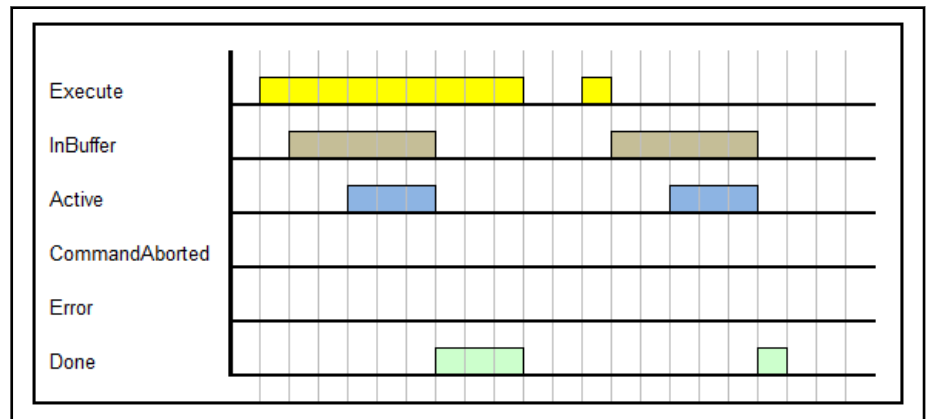


Fig. 5-27: I/O edge-controlled ML_Robot command function block. Processing successfully completed

Behavior of the InBuffer output - Errors during processing

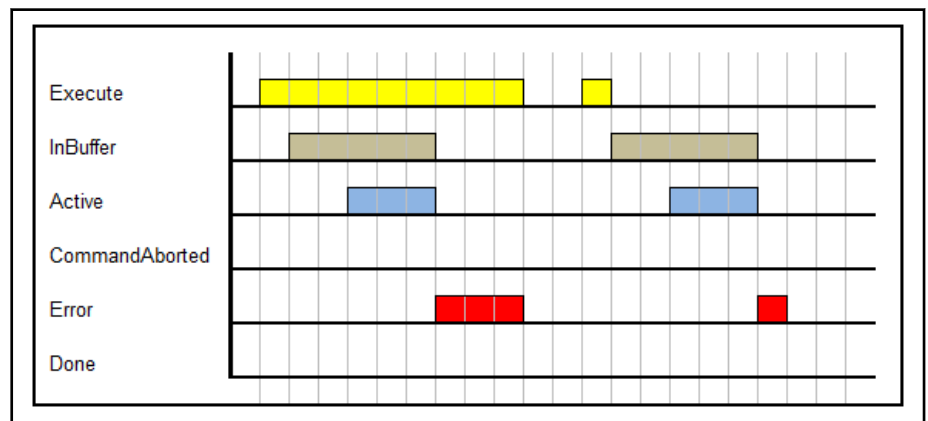


Fig. 5-28: I/O edge-controlled ML_Robot command function block. Errors during processing

Behavior of the InBuffer output - Processing canceled

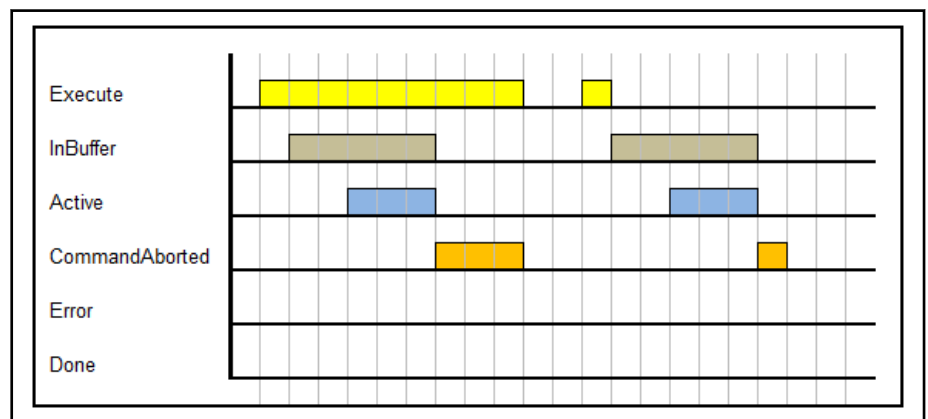


Fig. 5-29: I/O edge-controlled ML_Robot command function block. Processing canceled

5.3.4 Kinematics/Config

MC_GroupReset

Brief description The function block "MC_GroupReset" resets all reported errors of a kinematics.

ML_Robot.library

The function block influences the state machine of the kinematics.
(Refer to [State diagram of a kinematics](#), page 33).

 If there is no error in the kinematics of this function block, MC_GroupReset is not enabled.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

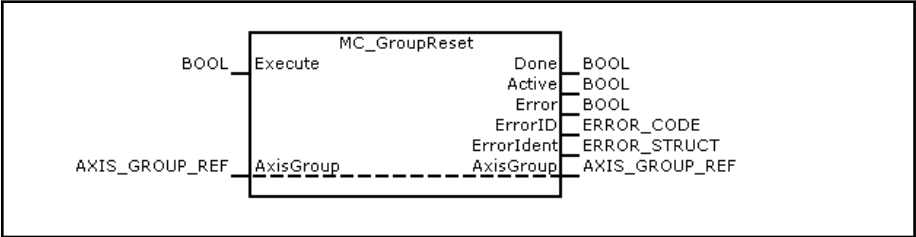


Fig. 5-30: FB MC_GroupReset

	Name	Type	Comment
VAR_IN_OUT CONSTANT	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. 5-9: Interface of the MC_GroupReset function block

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks](#), final processing on page 196.

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

MC_SetCartesianTransform

Brief description The function block "MC_SetCartesianTransform" defines a Cartesian product coordinate system.
(Refer to [Product coordinate system \(PCS\)](#) on page 27).

 This function block can only be used for Robot Control V2.
This function block can only be used for product coordinate systems that are no belt.

Interface description

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

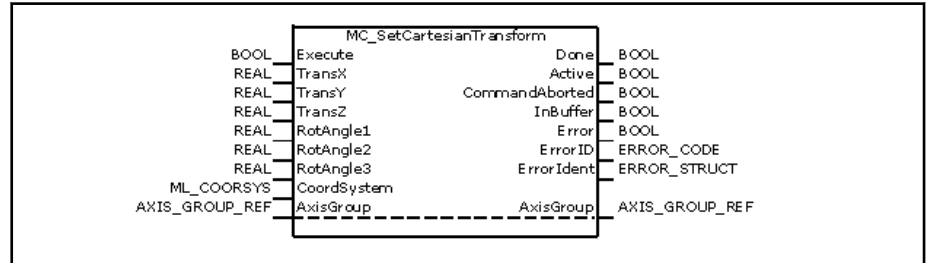


Fig. 5-31: Function block MC_SetCartesianTransform

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing 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 along the x-axis in MCS
	RotAngle2	REAL	Rotation of the PCS along the y-axis in MCS
	RotAngle3	REAL	Rotation of the PCS along the z-axis in MCS
	CoordSystem	ML_COORDSYS	Product coordinate system (PCS) that the values are to be set for (Refer to ML_COORDSYS on page 312).
VAR_OUTPUT	Done	BOOL	Definition was completed
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-10: 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"

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
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_COORDI NATES_1	CS_PRODU CT_COORDI NATES_16	CS_PRODUCT_COOR DINATES_1	Rising edge at "Execute"

Tab. 5-11: Input behavior of MC_SetCartesianTransform

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-12: 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 174](#)).



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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

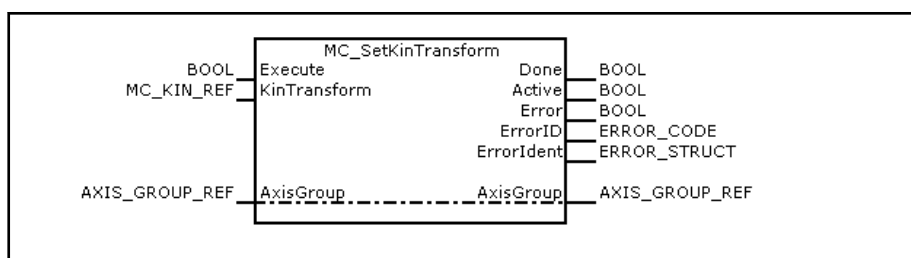


Fig. 5-32: MC_SetKinTransform function block

	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 300).
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-13: MC_SetKinTransform function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
KinTransform	MC_KIN_REF on page 300				Rising edge at "Execute"

Tab. 5-14: Input behavior of the MC_SetKinTransform

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 number of the function block MC_SetKinTransform

ML_Robot.library

MC_SetKinTransformExt

Brief description

The function block "MC_SetKinTransformExt" registers an extended user-defined transformation for a kinematics.
(Refer to [Extended user-defined transformation on page 176](#)).



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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

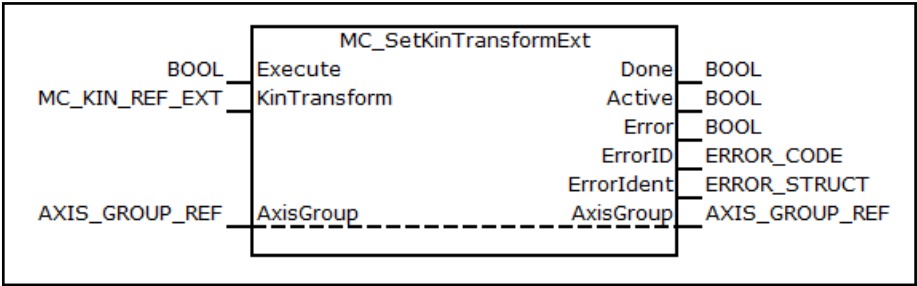


Fig. 5-33: MC_SetKinTransformExt function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing 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 on page 302).
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-16: MC_SetKinTransformExt function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
KinTransform	MC_KIN_REF_EXT on page 302				Rising edge at "Execute"

Tab. 5-17: Input behavior of the MC_SetKinTransformExt

Signal-time diagram

Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-18: 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.

Therefore, this function block directly influences the state machine of the kinematics.

(Refer to [State diagram of a kinematics](#), page 33).

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

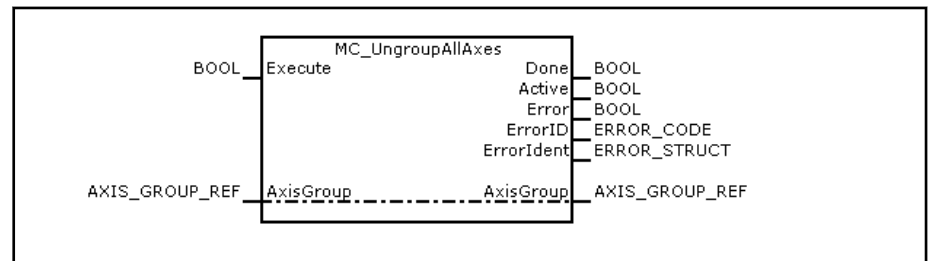


Fig. 5-34: MC_UngroupAllAxes function block

	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 212.
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-19: MC_UngroupAllAxes function block

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 5-20: Input behavior of the MC_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. 5-21: 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, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR 16#06	16#F0290402	16#00	Invalid kinematic number

Tab. 5-22: Generated error numbers of the MC_UngroupAllAxes function block for MLC_TABLE

ML_AddMechanicData

Brief description The "ML_AddMechanicData" function inserts an entry in the ML_KIN_MECHANIC_DATA structure.

(Refer to [Mechanical information](#), page 180).

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

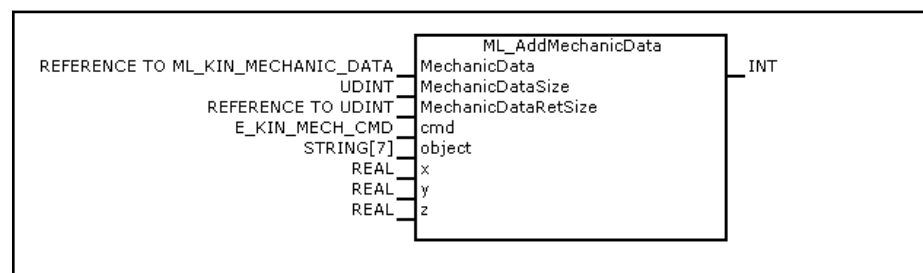


Fig. 5-35: ML_AddMechanicData function

	Name	Type	Comment
VAR_INPUT	MechanicData	REFERENCE TO ML_KIN_MECHANIC_DATA	Structure to be completed ML_KIN_MECHANIC_DATA on page 309
	MechanicDataSize	UDINT	Number of elements in MechanicData

	Name	Type	Comment
	MechanicDataRetSize	REFERENCE TO UDINT	Number of used elements in MechanicData
	cmd	E_KIN_MECH_CMD	Command for object name E_KIN_MECH_CMD on page 306
	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. 5-23: 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, page 33](#)).

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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

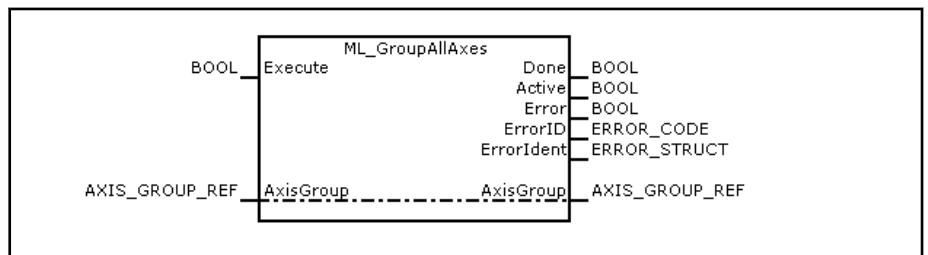


Fig. 5-36: ML_GroupAllAxes function block

	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

ML_Robot.library

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	For the error structure with further error subdivision, see F_Related_Table error codes , on page 214.
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-24: ML_GroupAllAxes function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 5-25: Input behavior of the ML_GroupAllAxes function block

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks](#), final processing on page 196.

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. 5-26: 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, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR 16#06	16#F0290402	16#00	Invalid kinematic number

Tab. 5-27: 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](#), page 76).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

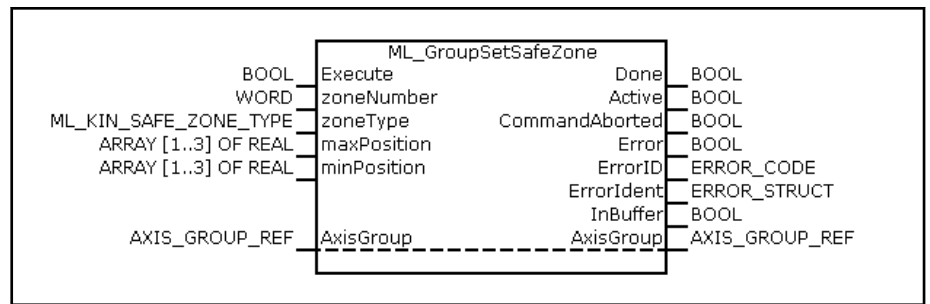


Fig. 5-37: Function block ML_GroupSetSafeZone

	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 306)
	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 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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-28: Function block ML_GroupSetSafeZone

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. 5-29: Input behavior of ML_GroupSetSafeZone

Signal-time diagram

- Inputs and outputs of edge-controlled function blocks, final processing on page 196
- InBuffer output at ML_Robot function blocks on page 204

ML_Robot.library

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_SetBeltDesyncRelConfiguration

Brief description The ML_SetBeltDesyncRelConfiguration function block configures relative desynchronization of a belt.
(Refer to [Belt-synchronous motion](#), page 65).

 This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

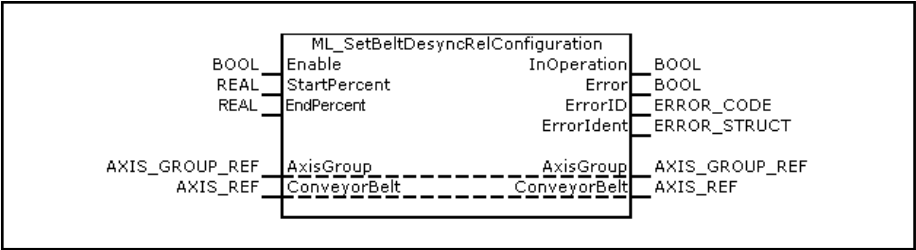


Fig. 5-38: Function block ML_SetBeltDesyncRelConfiguration

	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-30: Function block ML_SetBeltDesyncRelConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
StartPercent	REAL	0.0	100.0	0.0	Continuous
EndPercent	REAL	0.0	100.0	50.0	Continuous

Tab. 5-31: Input behavior of ML_SetBeltDesyncRelConfiguration

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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, page 65](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

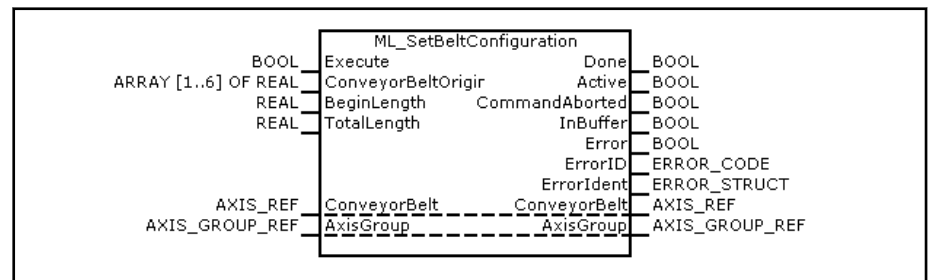


Fig. 5-39: Function block ML_SetBeltConfiguration

	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 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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-32: 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. 5-33: Input behavior of ML_SetBeltConfiguration

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-34: 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 65).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

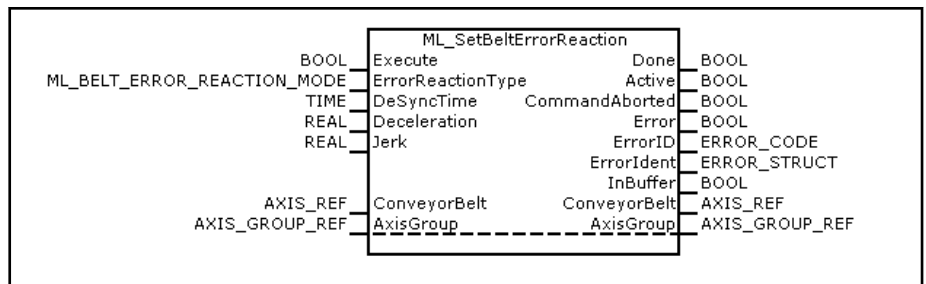


Fig. 5-40: Function block ML_SetBeltErrorReaction

	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 305).
	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 was aborted by another function block
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
VAR_IN_OUT CONSTANT	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-35: 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. 5-36: Input behavior of ML_SetBeltErrorReaction

Signal-time diagram • Inputs and outputs of edge-controlled function blocks, final processing on page 196

ML_Robot.library

- [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-37: Error numbers of the function block ML_SetBeltErrorReaction

ML_SetCylindricTransform

Brief description The function block "MC_SetCylindricTransform" defines a cylindrical product coordinate system.

(Refer to [Product coordinate system \(PCS\)](#), page 27).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

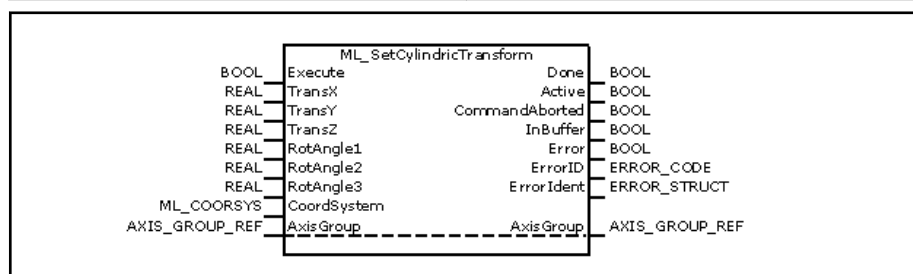


Fig. 5-41: Function block ML_SetCylindricTransform

	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
	RotAngle1	REAL	Rotation of the PCS along the x-axis in MCS
	RotAngle2	REAL	Rotation of the PCS along the y-axis in MCS
	RotAngle3	REAL	Rotation of the PCS along the z-axis in MCS

	Name	Type	Comment
	CoordSystem	ML_COORDSYS	Product coordinate system that the values are to be defined for See ML_COORDSYS , page 312
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-38: 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_COORDI NATES_1	CS_PRODU CT_COORDI NATES_16	CS_PRODUCT_COOR DINATES_1	Rising edge at "Execute"

Tab. 5-39: Input behavior of *ML_SetCylindricTransform*

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at *ML_Robot* function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_Robot.library

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. 5-40: 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](#), page 65).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

Interface description

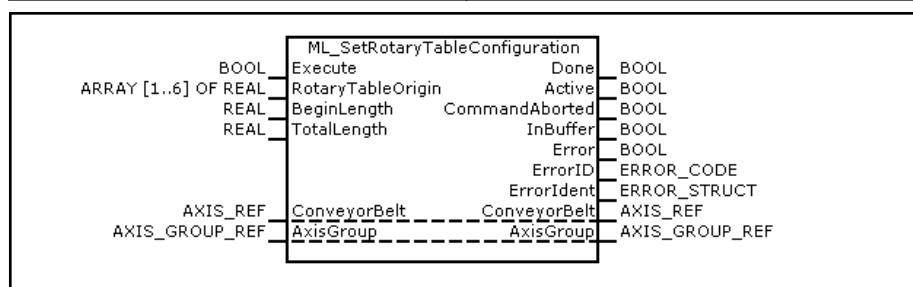


Fig. 5-42: Function block ML_SetRotaryTableConfiguration

	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

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-41: Function block ML_SetRotaryTableConfiguration

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. 5-42: Input behavior of ML_SetRotaryTableConfiguration

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-43: Error numbers of the function block ML_SetRotaryTableConfiguration

ML_RobotSetMoveID

Brief description The "ML_RobotSetMoveID" function block changes the current motion identification number.



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Config

ML_Robot.library

Interface description

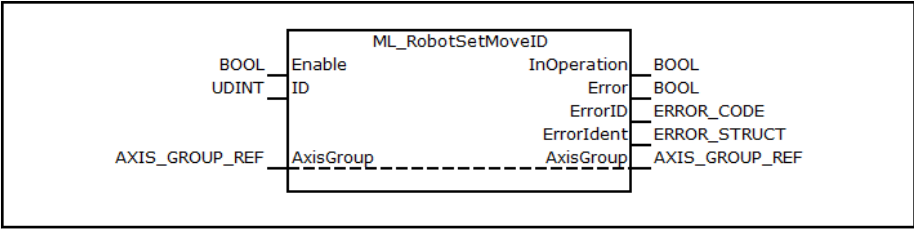


Fig. 5-43: ML_RobotSetMoveID function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
	ID	UDINT	New motion identification number
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-44: FB ML_RobotSetMoveID interface

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
ID	UDINT			0	Continuous if "Enable" = TRUE

Tab. 5-45: Input behavior of RobotSetMoveID

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-46: Error numbers of the function block ML_RobotRobotSetMoveID

5.3.5 Kinematic/Info

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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

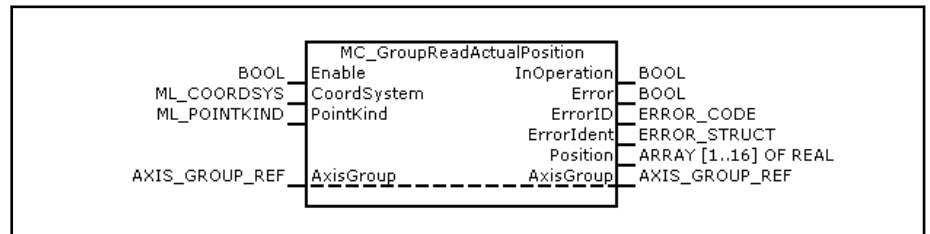


Fig. 5-44: Function block MC_GroupReadActualPosition

	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 , page 312
	PointKind	ML_POINTKIND	Point type regarding chronological sequence: Block preparation/block processing See ML_POINTKIND , page 310
VAR_OUTPUT	InOperation	BOOL	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-47: 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. 5-48: Input behavior of GroupReadActualPosition

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

ML_Robot.library

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotGetBeltConfiguration

Brief description The function block "ML_RobotGetBeltConfiguration" returns the current configuration of a belt.
(Refer to [Belt-synchronous motion](#), page 65).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

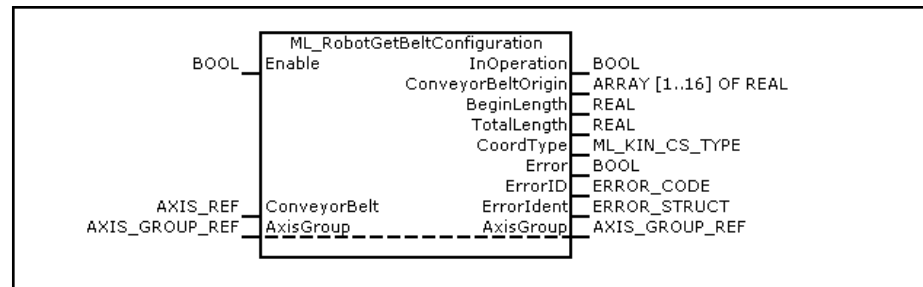


Fig. 5-45: Function block ML_RobotGetBeltConfiguration

	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 308)
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt

Tab. 5-49: Function block ML_RobotGetBeltConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-50: Input behavior of ML_RobotGetBeltConfiguration

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-51: Error numbers of the function block ML_RobotGetBeltConfiguration

ML_RobotGetBeltDesyncRelConfiguration

Brief description The "ML_RobotGetBeltDesyncRelConfiguration" function block reads the configuration for relative desynchronization of a belt.
(Refer to [Belt-synchronous motion, page 65](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

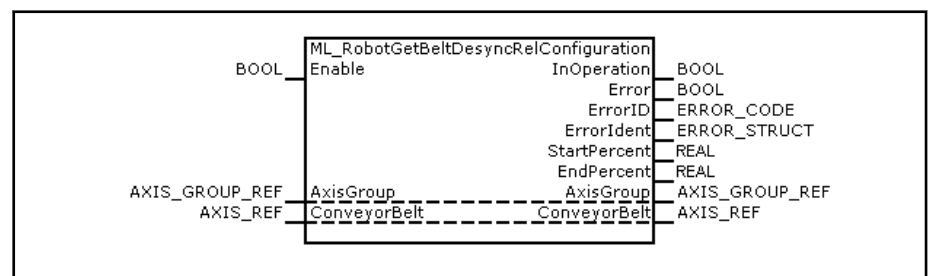


Fig. 5-46: Function block ML_RobotGetBeltDesyncRelConfiguration

	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

ML_Robot.library

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-52: Function block ML_RobotGetBeltDesyncRelConfiguration

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-53: Input behavior of ML_RobotGetBeltDesyncRelConfiguration

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotGetBeltErrorReaction

Brief description The function block "ML_RobotGetBeltErrorReaction" returns the configured belt error reaction of the robot.

(Refer to [Mechanical information, page 180](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

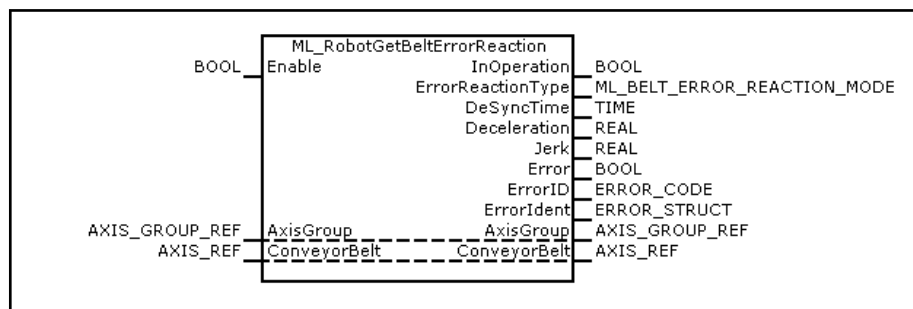


Fig. 5-47: Function block ML_RobotGetBeltErrorReaction

	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

	Name	Type	Comment
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Reference to the belt

Tab. 5-54: Function block ML_RobotGetBeltErrorReaction

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-55: Input behavior of ML_RobotGetBeltErrorReaction

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-56: Error numbers of the function block ML_RobotGetBeltErrorReaction

ML_RobotGetCmdInfo

Brief description The function block "ML_RobotGetCmdInfo" returns information on all commands in the block buffer.



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

ML_Robot.library

Interface description

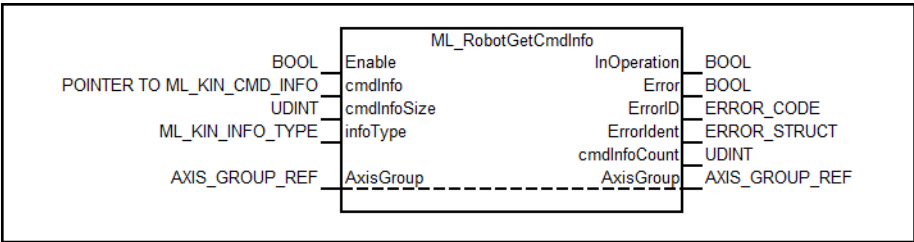


Fig. 5-48: Function block ML_RobotGetCmdInfo

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
	cmdInfo	POINTER TO ML_KIN_CMD_INFO	Pointer to the array of information to be filled on the commands in the block buffer
	cmdInfoSize	UDINT	Number of available elements in the array of information
	infoType	ML_KIN_INFO_TYPE	Type of information to be output
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
	cmdInfoCount	UDINT	Number of data read in the array of information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-57: Function block ML_RobotGetCmdInfo

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
cmdInfo	POINTER TO ML_KIN_MOVE_INFO			NULL	Continuous if "Enable" = TRUE
cmdInfoSize	UDINT			0	Continuous if "Enable" = TRUE
infoType	ML_KIN_INFO_TYPE			KIN_INFO_TYPE_PROGRESS	Continuous if "Enable" = TRUE

Tab. 5-58: Input behavior of the ML_RobotGetCmdInfo

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-59: Error numbers of the ML_RobotGetCmdInfo function block

ML_RobotGetMechanic

Brief description The function block "ML_RobotGetMechanic" returns mechanical kinematic data.

(Refer to [Mechanical information](#), page 180).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

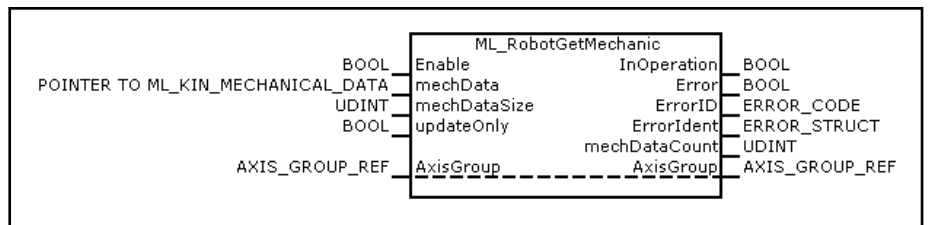


Fig. 5-49: Function block ML_RobotGetMechanic

	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-60: Function block ML_RobotGetMechanic

ML_Robot.library

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. 5-61: Input behavior of ML_RobotGetMechanic

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-62: Error numbers of the function block ML_RobotGetMechanic

ML_RobotGetMoveID

Brief description The "ML_RobotGetMoveID" function block returns the current motion identification number.



This function block can only be used for Robot Control V2.

Interface description

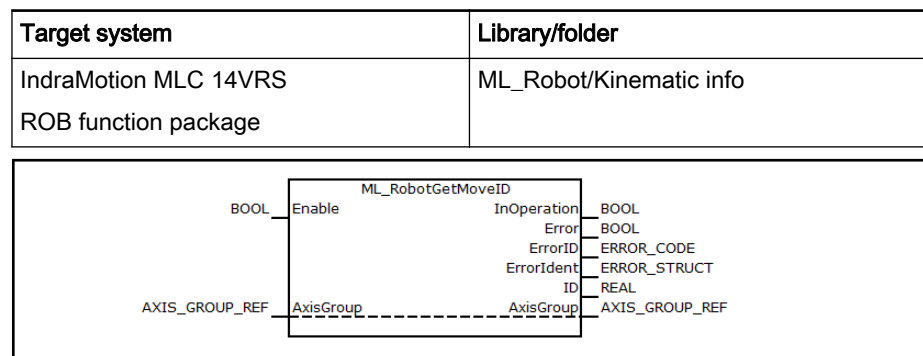


Fig. 5-50: ML_RobotGetMoveID function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed

	Name	Type	Comment
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	ID	REAL	Current motion identification number
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-63: FB ML_RobotGetMoveID interface

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-64: ML_RobotGetMoveID input behavior

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-65: Error numbers of the function block ML_RobotRobotGetMoveID

ML_RobotGetPerformance

Brief description The function block "ML_RobotGetPerformance" returns performance data of the kinematics.
 (Refer to [Performance measurement](#), page 82).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

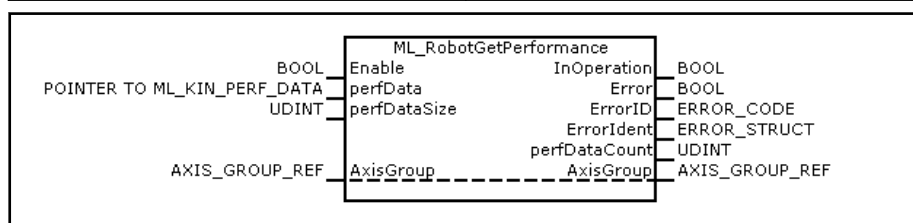


Fig. 5-51: Function block ML_RobotGetPerformance

ML_Robot.library

	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
	perfDataCount	UDINT	Number of data read in the array of performance data
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-66: Function block ML_RobotGetPerformance

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. 5-67: Input behavior of ML_RobotGetPerformance

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-68: Error numbers of the function block ML_RobotGetPerformance

ML_RobotGetSafeZones

Brief description

The function block "ML_RobotGetSafeZones" returns the safe zone of a kinematics.



This function block can only be used for Robot Control V2.

Interface description

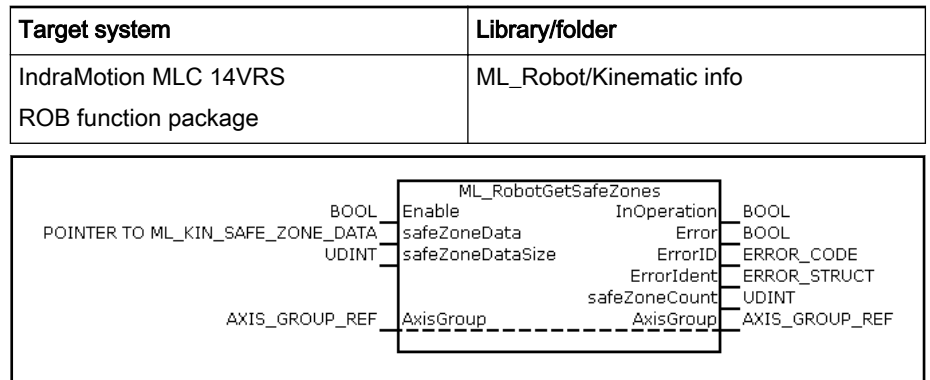


Fig. 5-52: ML_RobotGetSafeZones function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
	safeZoneData	POINTER TO ML_KIN_SAFE_ZONE_DATA	Pointer to the array of safe zone data to be filled Refer to ML_KIN_SAFE_ZONE_DATA on page 310
	safeZoneDataSize	UDINT	Number of available elements in the array of safe zone 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
	safeZoneCount	UDINT	Number of data read in the array of safe zone data
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-69: Interface of the ML_RobotGetSafeZones

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
safeZoneData	POINTER TO ML_KIN_SAFE_ZONE_DATA			NULL	Continuous if "Enable" = TRUE
safeZoneDataSize	UDINT			0	Continuous if "Enable" = TRUE

Tab. 5-70: Input behavior of the ML_RobotGetSafeZones

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-71: Error numbers of the ML_RobotGetSafeZones function block

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](#), page 23).



This function block can only be used for Robot Control V2.

Interface description

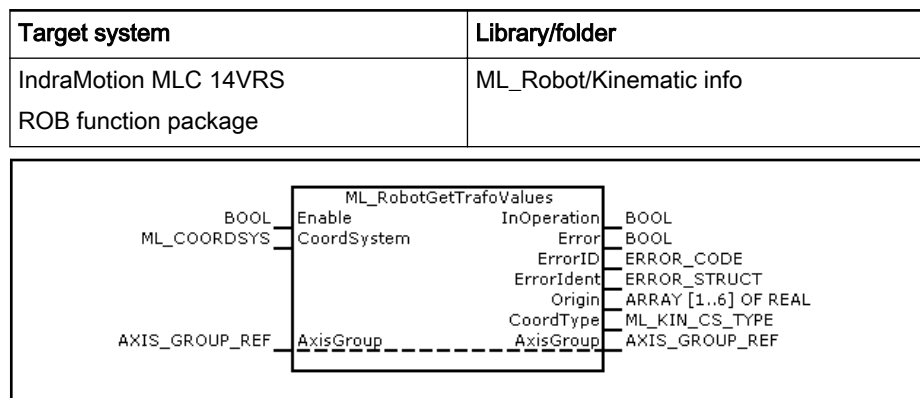


Fig. 5-53: Function block ML_RobotGetTrafoValues

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	CoordSystem	ML_COORDSYS	Coordinate system of the point (only product or basic coordinate system valid!)
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
	Origin	ARRAY [1..6] OF REAL	Position and orientation of the coordinate system
	CoordType	ML_KIN_CS_TYPE	Type of coordinate system
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-72: 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. 5-73: Input behavior of ML_RobotGetTrafoValues

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-74: Error numbers of the function block ML_RobotGetTrafoValues

ML_RobotMovePoint

Brief description The function block "ML_RobotMovePoint" moves a point to the current position in space. Usually, the function is called for all points on a belt. If the point contains one or several belt values, the point is moved to the current belt value by this function. When calling the point, it has the actual current position in space with the current belt value in the specified coordinate system.

If the point does not have a belt value, the point is not modified by the function.

(Refer to [Belt-synchronous motion on page 65](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

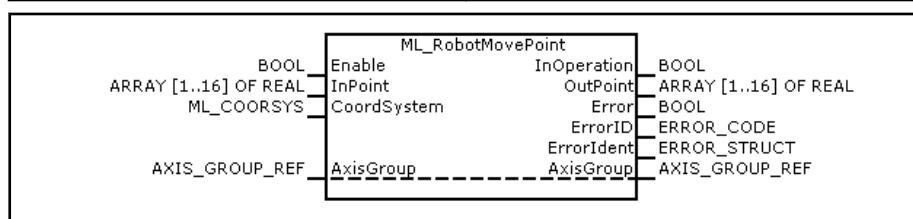


Fig. 5-54: Function block ML_RobotMovePoint

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing 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 point position in the specified coordinate system
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-75: 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. 5-76: Input behavior of ML_RobotMovePoint

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-77: Error numbers of the function block ML_RobotMovePoint

ML_RobotReadCyclicChannel**Brief description**

The function block "ML_RobotReadCyclicChannel" reads the current cyclic position.

(Refer to [Cyclic command value specification, page 80](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

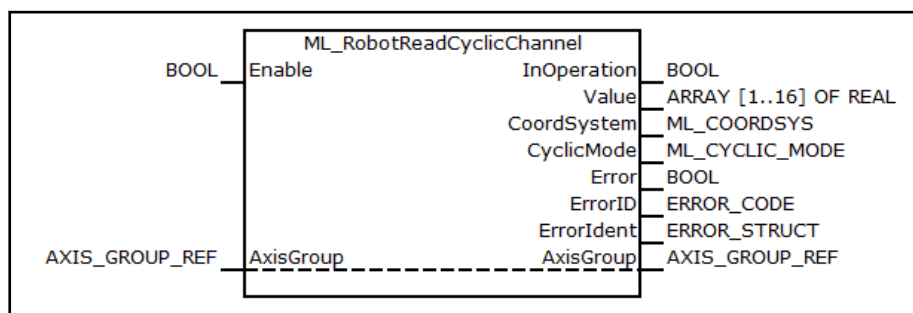


Fig. 5-55: Function block ML_RobotReadCyclicChannel

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing of function block enabled
VAR_OUTPUT	InOperation	BOOL	Definition was completed
	Value	ARRAY [1..16] OF REAL	Last cyclic commanded command values depending on the type of cyclic command value specification (CyclicMode)
	CoordSystem	ML_COORDSYS	Coordinate system of the last cyclically commanded position ML_COORDSYS on page 312
	CyclicMode	ML_CYCLIC_MODE	Type of cyclical command value specification ML_CYCLIC_MODE on page 313
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-78: Function block ML_RobotReadCyclicChannel

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 5-79: Input behavior of ML_RobotReadCyclicChannel

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-80: Error numbers of the function block ML_RobotReadCyclicChannel

ML_RobotSetPerformanceEnable

Brief description The function block "ML_RobotSetPerformanceEnable" changes the activation state of the performance measurement.

(Refer to [Performance measurement](#), page 82).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

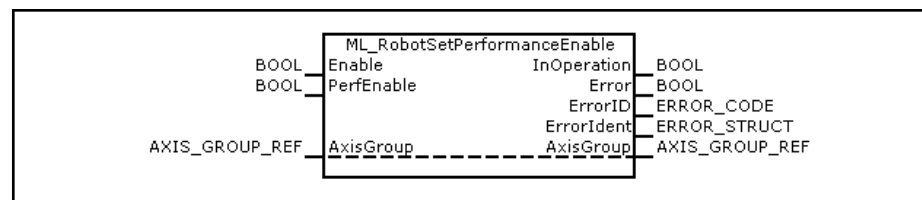


Fig. 5-56: Function block ML_RobotSetPerformanceEnable

	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-81: 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. 5-82: Input behavior of ML_RobotSetPerformanceEnable

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks](#), permanent processing on page. 199

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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_RobotSetPerformanceEnable

ML_RobotTransformPoint

Brief description

The function block "ML_RobotTransformPoint" transforms a point from a coordinate system to another coordinate system.

(Refer to [Coordinate systems](#), page 23).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

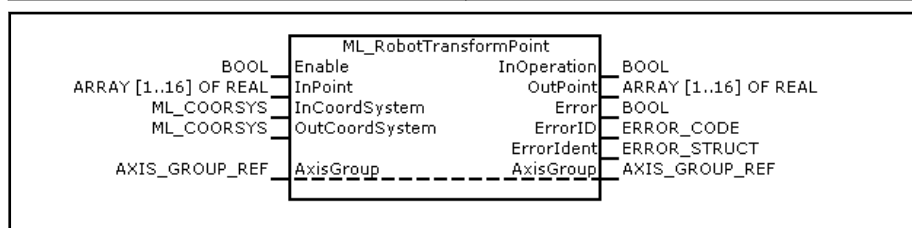


Fig. 5-57: Function block ML_RobotTransformPoint

	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 ML_COORDSYS on page 312
	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

ML_Robot.library

	Name	Type	Comment
	OutPoint	ARRAY [1..16] OF REAL	Current spatial position of the point
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-84: 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. 5-85: Input behavior of ML_RobotTransformPoint

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-86: 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, page 65](#)).

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic info

Interface description

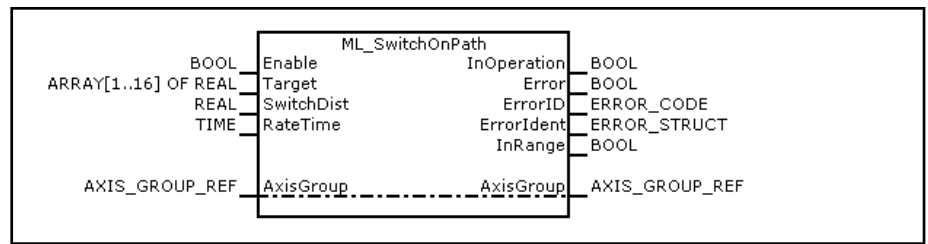


Fig. 5-58: ML_SwitchOnPath function block

	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-87: ML_SwitchOnPath function block

Name	Type	Default value	Effective
Enable	BOOL	FALSE	Continuous
Target	ARRAY [1..16] OF REAL	16(0.0)	Continuous
SwitchDist	REAL	16(0.0)	Continuous
RateTime	TIME	0.0	Continuous

Tab. 5-88: Input behavior of the ML_SwitchOnPath

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-89: Error numbers of the function block ML_SwitchOnPath

5.3.6 Kinematics/Movement

MC_GroupContinue

Brief description The function block "MC_GroupContinue" restarts a kinematics motion that was interrupted by the function block [MC_GroupInterrupt](#) on page 245.
(See [Continuing Motions](#) on page 64)



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

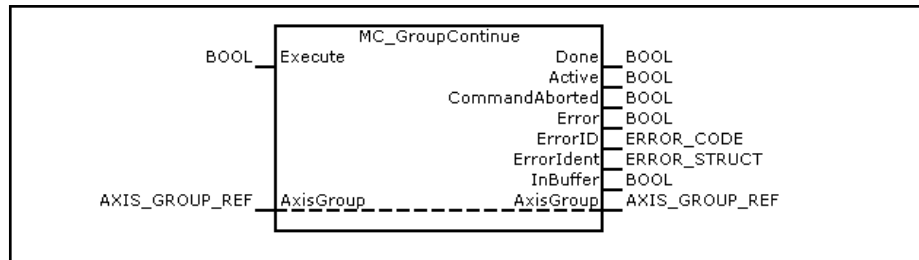


Fig. 5-59: Function block MC_GroupContinue

	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
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-90: Function block MC_GroupContinue

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 5-91: Input behavior of MC_GroupContinue

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

MC_GroupInterrupt

Brief description The function block "MC_GroupInterrupt" interrupts the movement of a kinematics.

(Refer to [Interrupting Motions, page 63](#)).

The motion can be resumed with the function block [MC_GroupContinue on page 244](#).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

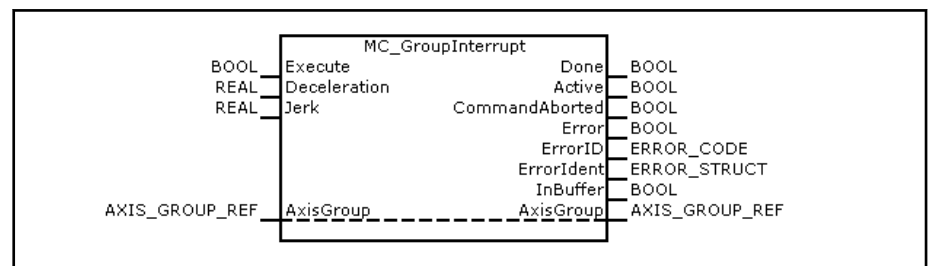


Fig. 5-60: Function block MC_GroupInterrupt

	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
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

ML_Robot.library

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-92: 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. 5-93: Input behavior of MC_GroupInterrupt

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

MC_GroupStop

Brief description The function block "MC_GroupStop" stops a kinematics in a controlled way.
(Refer to [Stopping Motions, page 62](#)).
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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

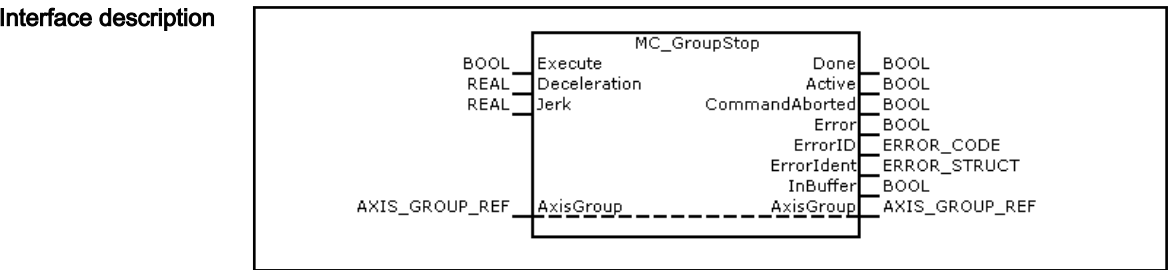


Fig. 5-61: Function block MC_GroupStop

	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
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-94: 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. 5-95: Input behavior of MC_GroupStop

Signal-time diagram

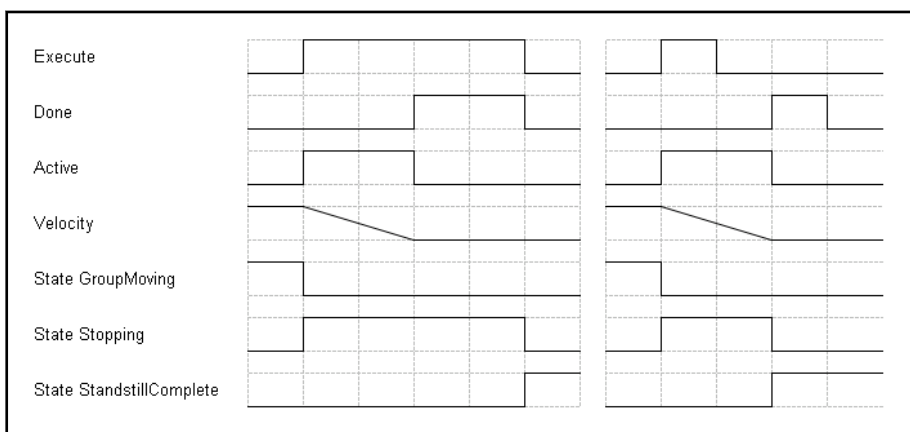


Fig. 5-62: Time diagram of MC_GroupStop

- [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling

The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_Robot.library

MC_MoveCircularAbsolute

Brief description

The function block "MC_MoveCircularAbsolute" moves a kinematics circularly from the last commanded position of the TCP position via an intermediate point to a specified absolute target position in the given coordinate system.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

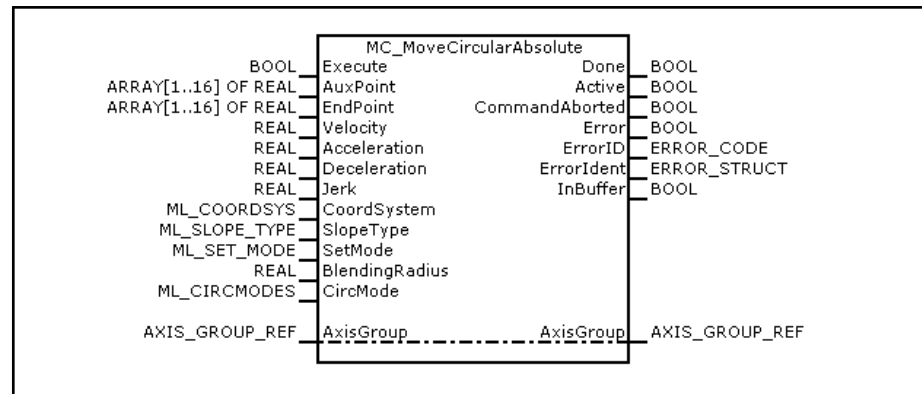


Fig. 5-63: MC_MoveCircularAbsolute function block

	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 ML_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Blending radius Robot Control V1: Blending radius in per cent of A-0-0230 the axes involved Robot Control V2: R>0 --> VIA; R=0 --> TO
	CircMode	ML_CIRCMODES	Type of circular motion ML_CIRCMODES on page 311
VAR_OUTPUT	Done	BOOL	Motion completed

	Name	Type	Comment
	Active	BOOL	Motion is executed
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-96: 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"
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. 5-97: Input behavior of the MC_MoveCircularAbsolute

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)
- Error handling** The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

ML_Robot.library

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 213

Tab. 5-98: 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 position of the TCP position via an intermediate point to a specified absolute target position in the given coordinate system. The intermediate and end points are specified in relation to the starting point.

 This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

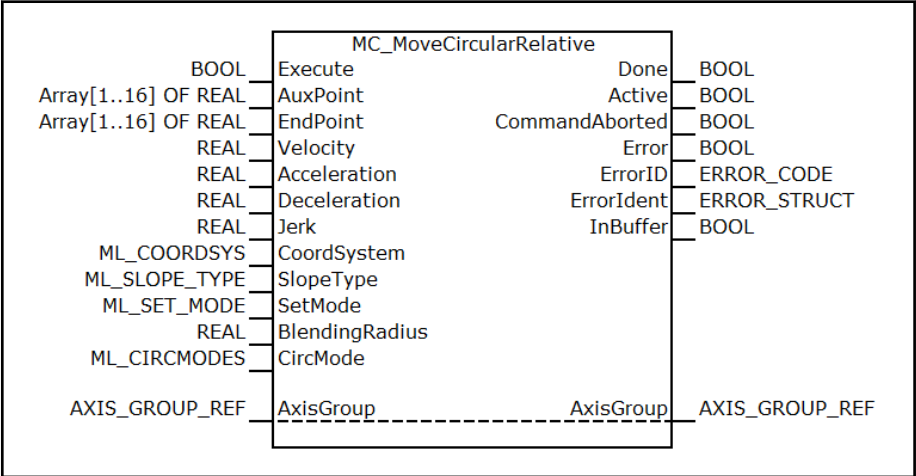


Fig. 5-64: Function block MC_MoveCircularRelative

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	AuxPoint	ARRAY [1..16] OF REAL	Intermediate position of the motion, relative specification to the starting point
	EndPoint	ARRAY [1..16] OF REAL	Target position of the motion, relative specification 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 ML_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Blending radius
	CircMode	ML_CIRCMODES	Type of circular motion ML_CIRCMODES on page 311
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-99: 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"

ML_Robot.library

Name	Type	Min. value	Max. value	Default value	Effective
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. 5-100: Input behavior of MC_MoveCircularRelative

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 213

Tab. 5-101: Error numbers of the function block MC_MoveCircularRelative

MC_MoveDirectAbsolute

Brief description The function block "MC_MoveDirectAbsolute" moves a kinematics point-to-point from the last commanded TCP position to a specified absolute target position in the given coordinate system.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

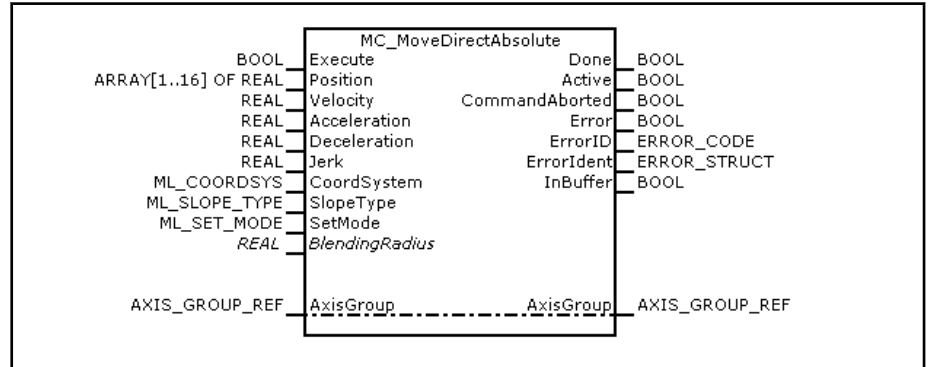


Fig. 5-65: MC_MoveDirectAbsolute function block

	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)
	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 ML_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Robot Control V1: Blending radius in per cent of A-0-0230 the axes involved Robot Control V2: R>0 --> VIA; R=0 --> TO
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	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

ML_Robot.library

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-102: 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_COOR DINATES	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. 5-103: Input behavior of the MC_MoveDirectAbsolute

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-104: Error numbers of the function block MC_MoveDirectAbsolute

MC_MoveDirectRelative

Brief description

The function block "MC_MoveDirectRelative" moves a kinematics point-to-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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

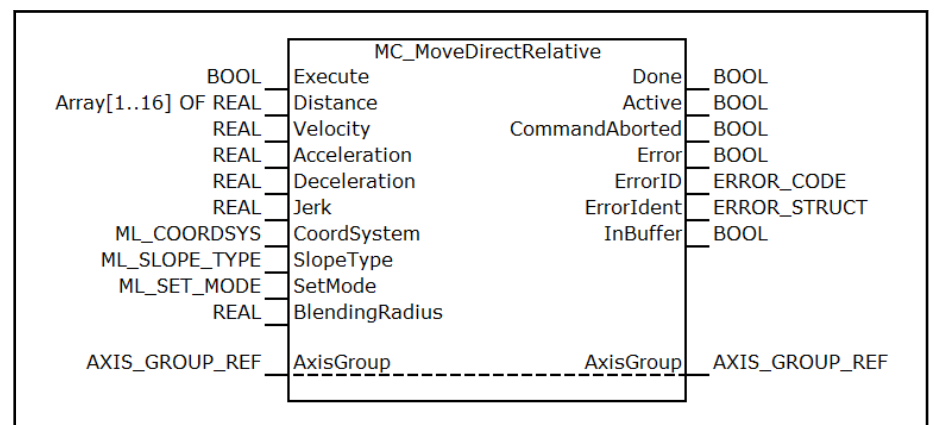


Fig. 5-66: Function block MC_MoveDirectRelative

	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)
	Acceleration	REAL	Percentage of the maximum axis acceleration (axis parameter A-0-0034)

ML_Robot.library

	Name	Type	Comment
	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 ML_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Robot Control V2: R>0 --> VIA; R=0 --> TO
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-105: 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_COOR DINATES	Rising edge at "Execute"
SlopeType	ML_SLOPE_TYPE			ST_BLOCK_SLOPE	Rising edge at "Execute"

Name	Type	Min. value	Max. value	Default value	Effective
SetMode	ML_SET_MODE			SM_TO	Rising edge at "Execute"
BlendingRadius	REAL	0.0	1000.0	0.0	Rising edge at "Execute"

Tab. 5-106: Input behavior of MC_MoveDirectRelative

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG***V14-RERS-EN-P, R911341481

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. 5-107: 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 specified coordinate system.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

ML_Robot.library

Interface description

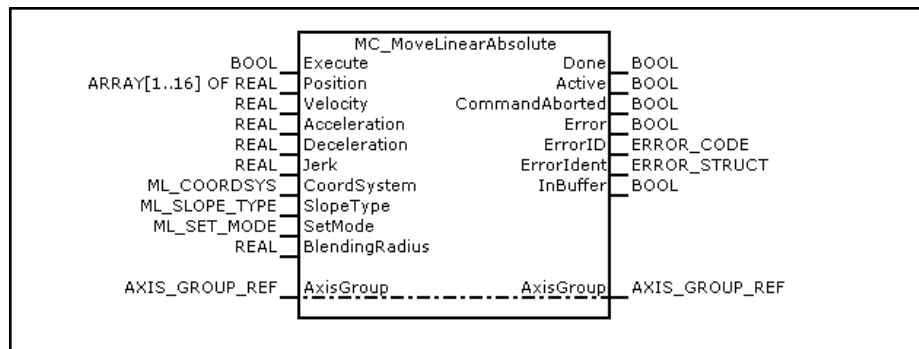


Fig. 5-67: MC_MoveLinearAbsolute function block

	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
	Deceleration	REAL	Maximum path deceleration
	Jerk	REAL	Maximum path jerk
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion ML_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Blending radius R>0 --> VIA; R=0 --> TO
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-108: 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"

Name	Type	Min. value	Max. value	Default value	Effective
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. 5-109: Input behavior of the MC_MoveLinearAbsolute

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG***V14-RERS-EN-P, R911341481

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. 5-110: Error numbers of the function block MC_MoveLinearAbsolute

ML_Robot.library

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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

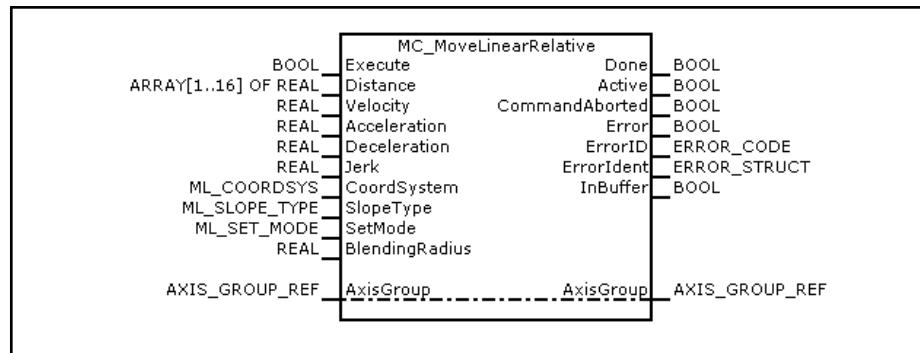


Fig. 5-68: MC_MoveLinearRelative function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Distance	ARRAY [1..16] OF REAL	Relative distance 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_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Blending radius Robot Control V1: Blending radius in per cent of A-0-0230 the axes involved Robot Control V2: R>0 --> VIA; R=0 --> TO
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

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-111: 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. 5-112: Input behavior of the MC_MoveLinearRelative

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_Robot.library

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. 5-113: Error numbers of the function block MC_MoveLinearRelative

ML_BeltDeSynchronisationWithLimits

Brief description The function block "ML_BeltDeSynchronisationWithLimits" carries out desynchronization from a belt using motion limit values.
(Refer to [Belt-synchronous motion](#), page 65).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

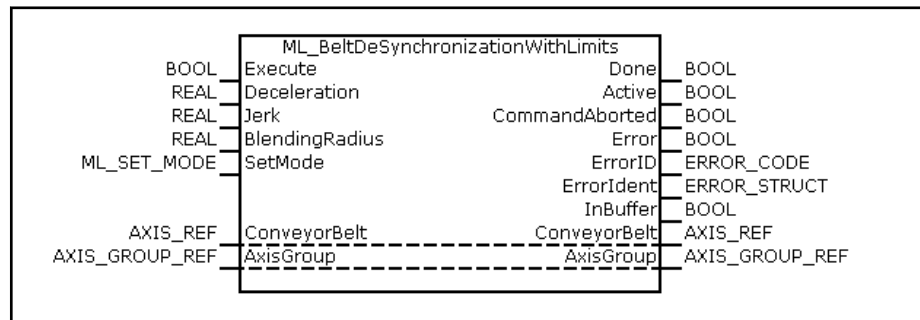


Fig. 5-69: Function block ML_BeltDeSynchronisationWithLimits

	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) ML_SET_MODE , page 313
VAR_OUTPUT	Done	BOOL	Registering of the transformation completed
	Active	BOOL	Registering of the transformation active

	Name	Type	Comment
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-114: 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. 5-115: Input behavior of *ML_BeltDeSynchronisationWithLimits*

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 65](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

ML_Robot.library

Interface description

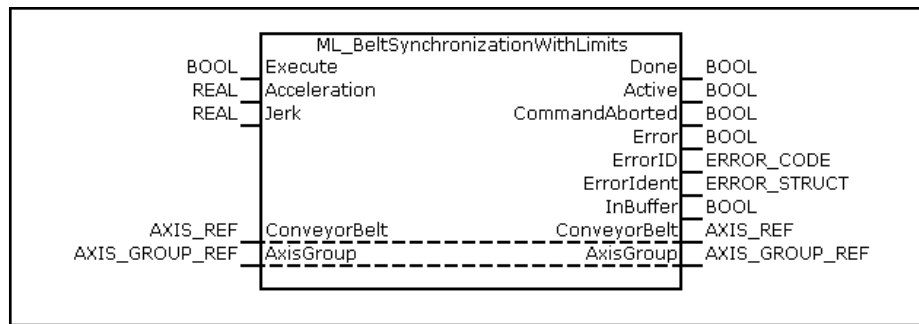


Fig. 5-70: Function block ML_BeltSynchronisationWithLimits

	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
	InBuffer	BOOL	Command successfully put in block buffer.
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics
	ConveyorBelt	AXIS_REF	Axis reference of the belt axis

Tab. 5-116: 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. 5-117: Input behavior of ML_BeltSynchronisationWithLimits

Signal-time diagram

- Inputs and outputs of edge-controlled function blocks, final processing on page 196
- InBuffer output at ML_Robot function blocks on page 204

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 47](#)



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

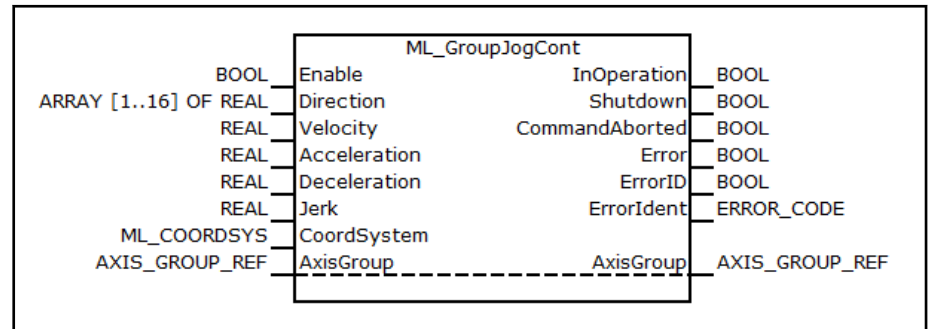


Fig. 5-71: Function block ML_GroupJogCont

	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

ML_Robot.library

	Name	Type	Comment
	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 ML_COORDSYS , page 312
VAR_OUTPUT	InOperation	BOOL	Motion is executed
	Shutdown	BOOL	Motion is stopped
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-118: 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_COOR DINATES	Rising edge at "Enable"

Tab. 5-119: Input behavior of MC_GroupJogCont

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-120: 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 47](#)



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

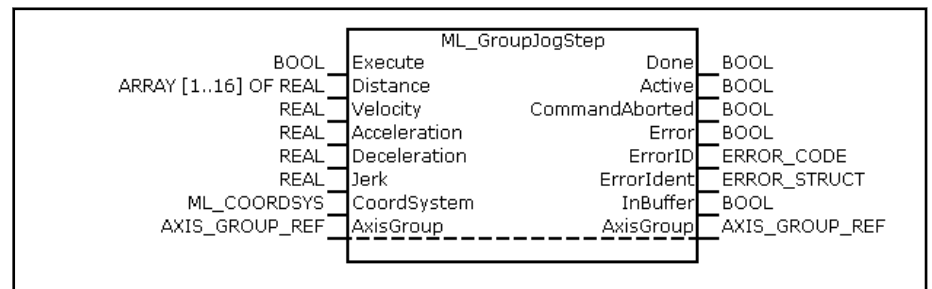


Fig. 5-72: Function block ML_GroupJogStep

	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

ML_Robot.library

	Name	Type	Comment
	Acceleration	REAL	Maximum path acceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s ² or in/s ²
	Deceleration	REAL	Maximum path deceleration CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s ² or in/s ²
	Jerk	REAL	Maximum path jerk CoordSystem = CS_AXES_COORDINATES: indicated in percent CoordSystem ≠ CS_AXES_COORDINATES: indicated in mm/s ³ or in/s ³
	CoordSystem	ML_COORDSYS	Coordinate system of the target position of the motion ML_COORDSYS , page 312
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-121: 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_COOR DINATES	Rising edge at "Execute"

Tab. 5-122: Input behavior of ML_GroupJogStep

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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. 5-123: Error numbers of the function block ML_GroupJogStep

ML_GroupWait

Brief description Due to the Coordinated Motion function block "ML_GroupWait" the kinematics waits for the specified time in the last commanded position.
 (Refer to [Waiting](#), page 64).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

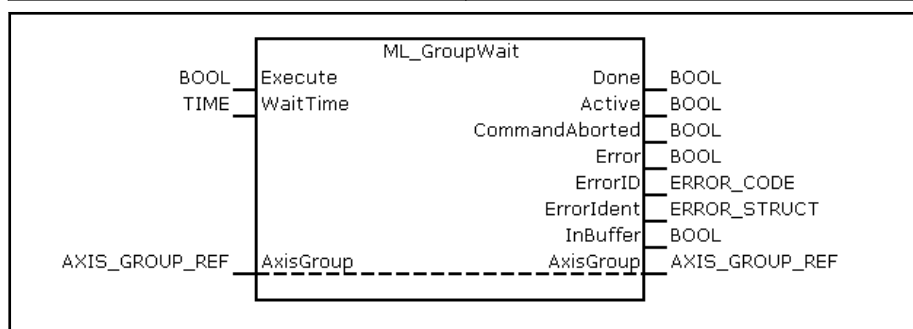


Fig. 5-73: Function block ML_GroupWait

ML_Robot.library

	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 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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-124: 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. 5-125: Input behavior of ML_GroupWait

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotOpenCyclicChannel

Brief description The function block "ML_RobotOpenCyclicChannel" opens a cyclic command value channel.
(Refer to [Cyclic command value specification, page 80](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

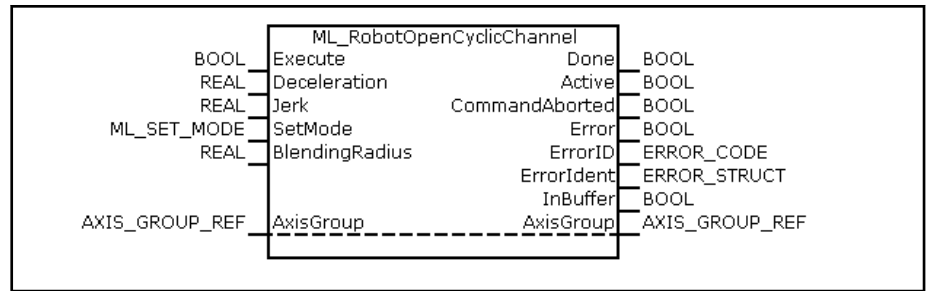


Fig. 5-74: Function block ML_RobotOpenCyclicChannel

	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) ML_SET_MODE, page 313
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
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-126: Function block ML_RobotOpenCyclicChannel

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. 5-127: Input behavior of ML_RobotOpenCyclicChannel

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:

ML_Robot.library

MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotOpenDirectCyclicChannel

Brief description The function block "ML_RobotOpenDirectCyclicChannel" opens a cyclic command value channel for PTP motions.
(Refer to [Cyclic command value specification, page 80](#)).

 This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

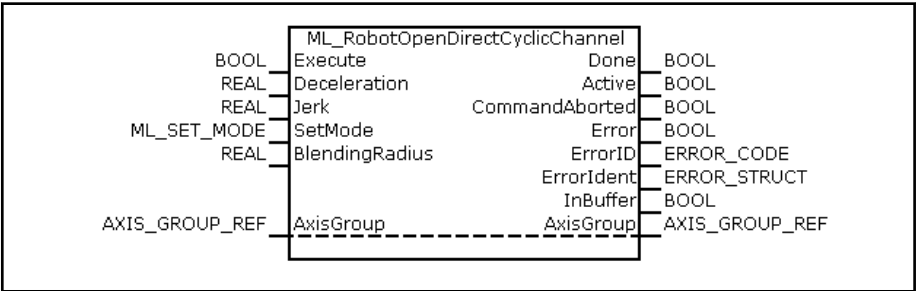


Fig. 5-75: Function block ML_RobotOpenDirectCyclicChannel

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Deceleration	REAL	Maximum path deceleration in percent
	Jerk	REAL	Maximum path jerk in percent
	BlendingRadius	REAL	Blending radius
	SetMode	ML_SET_MODE	Blending type (VIA, TO) ML_SET_MODE, page 313
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-128: Function block ML_RobotOpenDirectCyclicChannel

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. 5-129: Input behavior of ML_RobotOpenDirectCyclicChannel

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotWriteCyclicChannel

Brief description The function block "ML_RobotWriteCyclicChannel" writes a cyclic command value.

(Refer to [Cyclic command value specification, page 80](#)).



This function block can only be used for Robot Control V2.



The function block **must be** called in the motion cycle as otherwise, unexpected error messages or jerking axes can occur!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

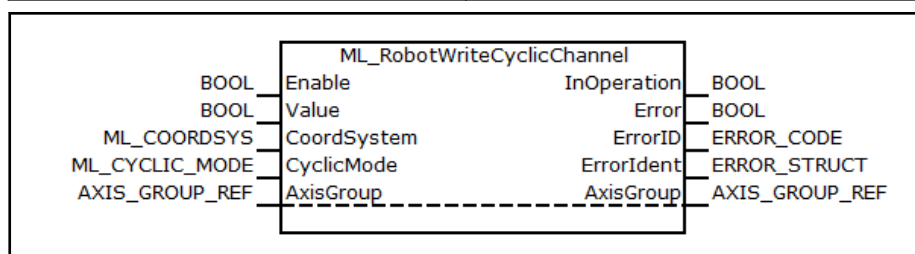


Fig. 5-76: Function block ML_RobotWriteCyclicChannel

ML_Robot.library

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing of function block enabled
	Value	ARRAY [1..16] OF REAL	Cyclic command value depending on the type of cyclic command value specification (CyclicMode)
	CoordSystem	ML_COORDSYS	Coordinate system of the cyclic command position ML_COORDSYS , page 312
	CyclicMode	ML_CYCLIC_MODE	Type of cyclical command value specification ML_CYCLIC_MODE , page 313
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-130: Function block ML_RobotWriteCyclicChannel

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Value	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			CYCLIC_MODE_POS_ABS	Rising edge at "Execute"

Tab. 5-131: Input behavior of ML_RobotWriteCyclicChannel

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks](#), permanent processing on page. 199

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_RobotMoveJumpAbsolute

Brief description The "ML_RobotMoveJumpAbsolute" function block moves a kinematics with a Jump- motion from the last commanded position of the TCP to a specified absolute target position in the coordinate system via starting, maximum and end height.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic Movement

Interface description

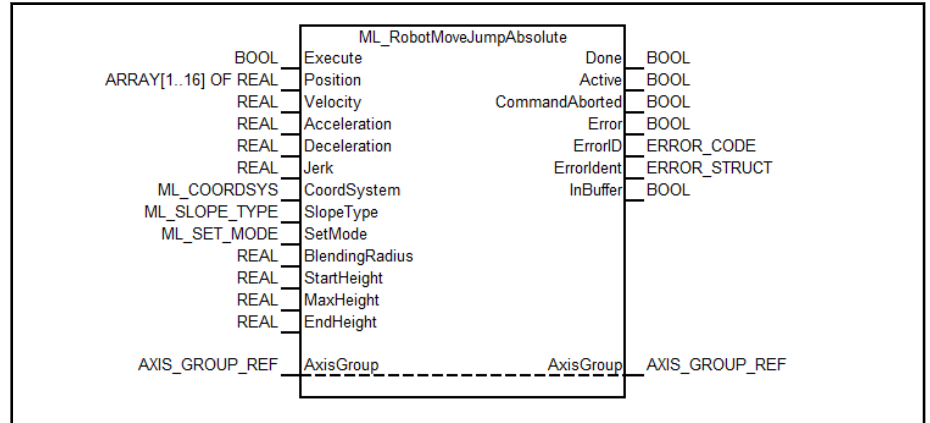


Fig. 5-77: ML_RobotMoveJumpAbsolute interface

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	Position	ARRAY[1..16] OF REAL	Absolute 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_COORDSYS on page 312
	SlopeType	ML_SLOPE_TYPE	Slope type of the motion ML_SLOPE_TYPE on page 313
	SetMode	ML_SET_MODE	Type of blending (VIA, TO) ML_SET_MODE on page 313
	BlendingRadius	REAL	Blending radius R>0 --> VIA; R=0 --> TO
	StartHeight	REAL	Minimum height at the start of the horizontal motion
	MaxHeight	REAL	Maximum height of the jump motion
	EndHeight	REAL	Minimum height at the end of the horizontal motion
VAR_OUTPUT	Done	BOOL	Motion completed
	Active	BOOL	Motion is executed
	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

ML_Robot.library

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-132: ML_RobotMoveJumpAbsolute interface

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_COOR DINATES	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"
StartHeight	REAL			0.0	Rising edge at "Execute"
MaxHeight	REAL			0.0	Rising edge at "Execute"
EndHeight	REAL			0.0	Rising edge at "Execute"

Tab. 5-133: ML_RobotMoveJumpAbsolute interface

- Signal-time diagram**
- [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#)
 - [InBuffer output at ML_Robot function blocks on page 204](#)

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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' rejected Cause: Axes have previously not been grouped to the kinematics using the ML_GroupAllAxes function block
OTHER_ERROR 16#FE	16#F12C0F70	16#00	Spline parameter length invalid Cause: The parameter length of the spline could not be determined
OTHER_ERROR 16#FE	16#F12C0F72	16#00	Spline definition lengths invalid Cause: A length of the spline definition is close to 0
CALCULATION_ERROR 16#07	16#F2299582	16#00	Error during spline calculation Cause: The velocity window for the calculation was exceeded. Typically, this error occurs if a very low height - in proportion to the distance between starting point and target point - was commanded using the command MoveJump
CALCULATION_ERROR 16#07	16#F2299583	16#00	Velocity error exceedance during spline calculation Cause: The velocity window for the calculation was exceeded. Typically, this error occurs if a very low height - in proportion to the distance between starting point and target point - was commanded using the command MoveJump. Refer to Motion type Jump on page 42
INPUT_RANGE_ERROR 16#06	16#F2299584	16#00	Invalid heights at Move Jump Cause: One or several invalid heights were specified. Refer to Motion type Jump on page 42

Tab. 5-134: Error numbers of the function block ML_RobotMoveJumpAbsolute

5.3.7 Kinematics/Parameter

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

ML_Robot.library

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
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab. 5-135: 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:=(CntlNo:=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 up to 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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

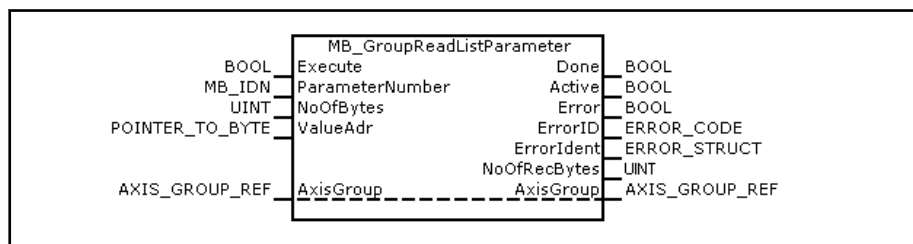


Fig. 5-78: FB MB_GroupReadListParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific 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 277)
	NoOfBytes	UINT	Maximum number of data to be read in bytes
	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. 5-136: Interface of FB MB_GroupReadListParameter

Signal-time diagram

Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling

The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

ML_Robot.library

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

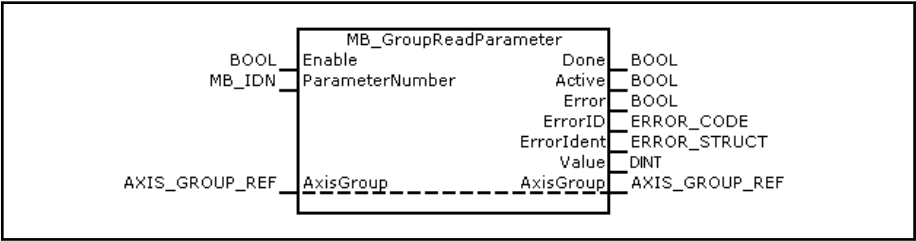


Fig. 5-79: FB MB_GroupReadParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	ParameterNum ber	MB_IDN	Number of the desired K-parameter (see Accessing kinematic data on page 277)
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. 5-137: Interface of FB MB_GroupReadParameter

Signal-time diagram Refer to [Inputs and outputs of state-controlled function blocks, final processing on page 193](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

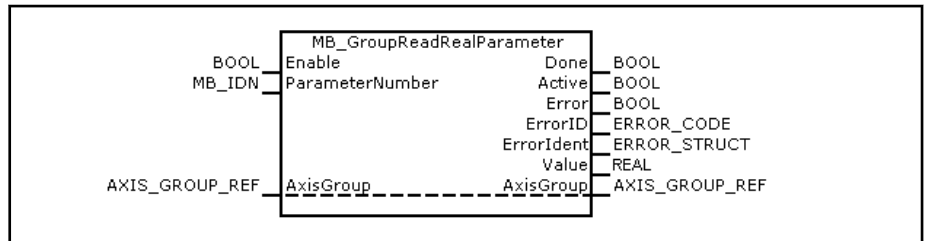


Fig. 5-80: FB MB_GroupReadRealParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	ParameterNumber	MB_IDN	Number of the desired K-parameter (see Accessing kinematic data on page 277)
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. 5-138: Interface of the MB_GroupReadRealParameter FB

ML_Robot.library

Signal-time diagram Refer to [Inputs and outputs of state-controlled function blocks, final processing on page 193](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

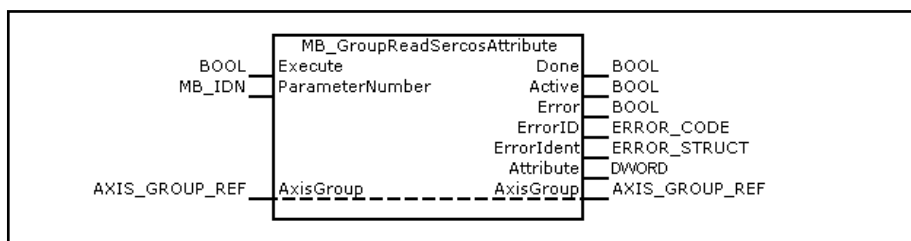


Fig. 5-81: FB MB_GroupReadSercosAttribute

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific 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 277)
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

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	DWORD	4 byte value, contains Sercos attributes of the K-parameter

Tab. 5-139: Interface of the MB_GroupReadSercosAttribute function block

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

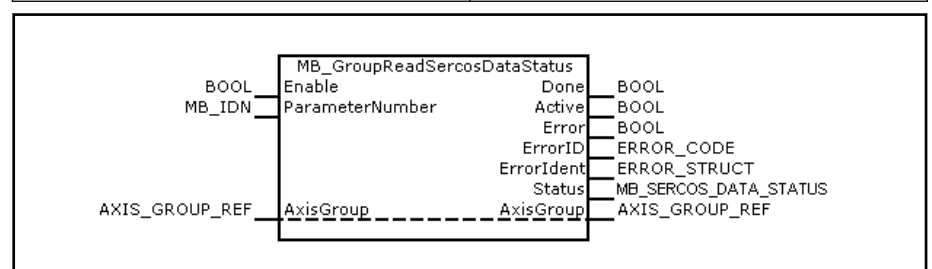


Fig. 5-82: FB MB_GroupReadSercosDataStatus

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific kinematics
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.

ML_Robot.library

	Name	Type	Comment
	ParameterNumber	MB_IDN	Number of the desired K-parameter (see Accessing kinematic data on page 277)
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 (ML_Base)

Tab. 5-140: Interface of FB MB_GroupReadSercosDataStatus

Brief description Using this function block and a [MB_GroupWriteParameter, page 287](#), drive commands can be triggered and their processing tracked:

- 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, final processing on page 193](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

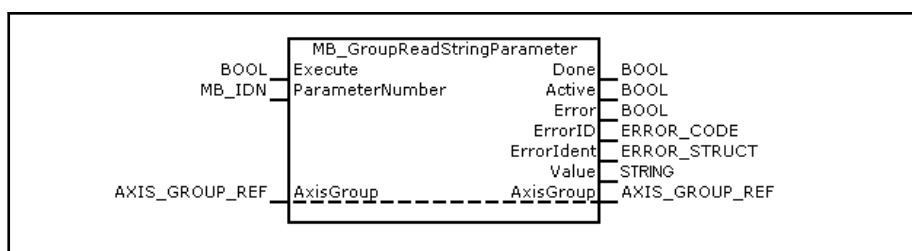


Fig. 5-83: FB MB_GroupReadStringParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific 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 277)
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 up to 255 characters

Tab. 5-141: Interface of the MB_GroupReadStringParameter function block

Signal-time diagram

Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling

The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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 "MB_GroupWriteListParameter" function block writes memory content of a defined length (NoOfBytes) provided from the initial address (ValueAdr) to a user-defined parameter.

ML_Robot.library



The specified number of bytes must exactly match the data volume to be transferred!

If the number is too high, garbage files are transmitted. If the number is too low, data are 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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

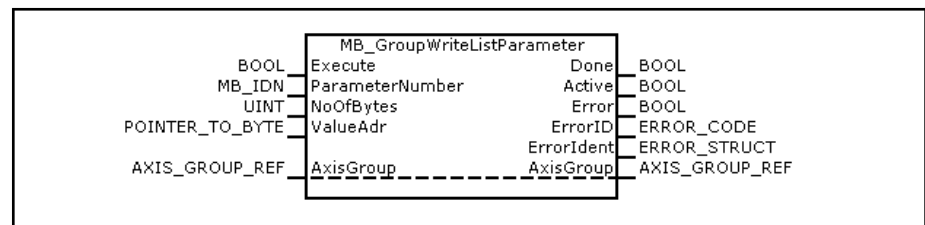


Fig. 5-84: FB MB_GroupWriteListParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific 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 277)
	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. 5-142: Interface of FB MB_GroupWriteListParameter

Signal-time diagram

Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling

The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

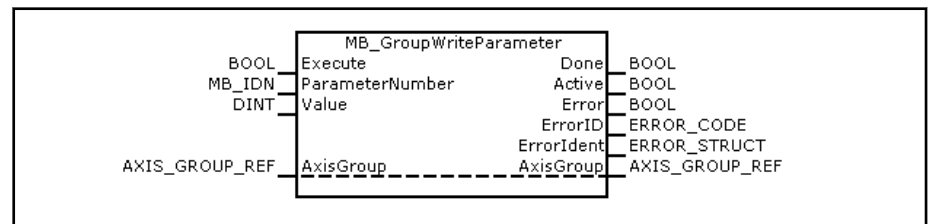


Fig. 5-85: FB MB_GroupWriteParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	ParameterNumber	MB_IDN	Number of the desired K-parameter (see Accessing kinematic data on page 277)
	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. 5-143: Interface of the MB_GroupWriteParameter FB

Signal-time diagram

Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

ML_Robot.library

- Error handling** The function block generates the following types of error messages:
- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
 - MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

MB_GroupWriteRealParameter

Brief description The function block "MB_GroupWriteRealParameter" writes a value to a parameter specified by the user.



Only for K-parameters!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

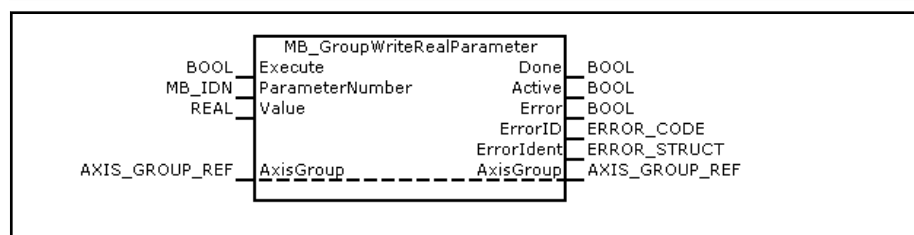


Fig. 5-86: FB MB_GroupWriteRealParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific kinematics
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter.
	ParameterNumber	MB_IDN	Number of the desired K-parameter (see Accessing kinematic data on page 277)
	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. 5-144: Interface of the function block MB_GroupWriteRealParameter

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)
- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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!

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic parameter

Interface description

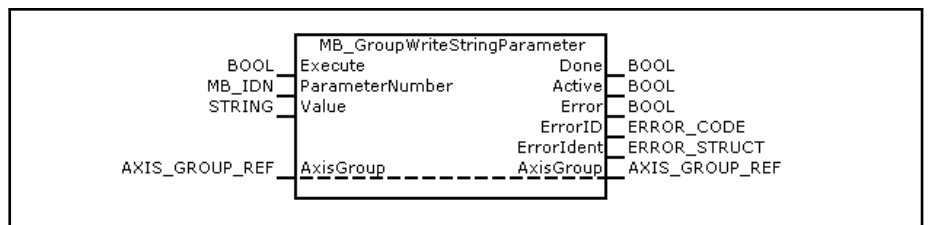


Fig. 5-87: FB MB_GroupWriteStringParameter

	Name	Type	Comment
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Contains information on the specific 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 277)
	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. 5-145: Interface of FB MB_GroupWriteStringParameter

Signal-time diagram Refer to [Inputs and outputs of edge-controlled function blocks, final processing on page 196](#).

Error handling The function block generates the following types of error messages:

- SERCOS_TABLE, 16#10 - error when accessing the parameter (e.g. parameter is not supported)

ML_Robot.library

- MLC_TABLE, 16#30 - error when accessing the kinematics/control

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

In addition, the following can be displayed:

- F00D1222, Too many data to transmit, data larger than 255 bytes
- F00D1224, Error, no memory available
- F00D1225, Error, cannot handle a single parameter
- F00D1227, Error while writing date

5.3.8 Kinematic/TeachIn

ML_GroupDeletePointFile

Brief description

The function block "ML_GroupDeletePointFile" deletes a point file of a kinematics.

(Refer to [Teach-In on page 49](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

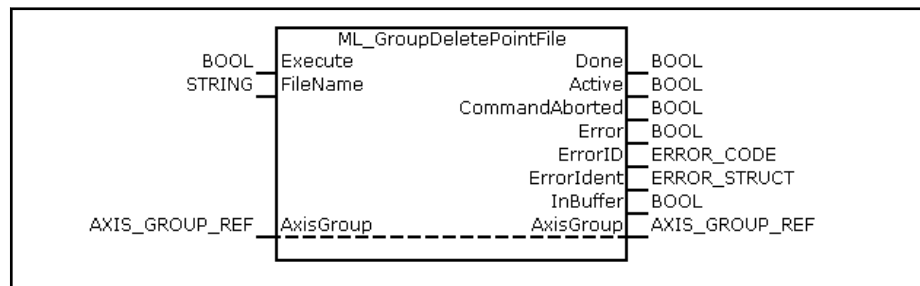


Fig. 5-88: Function block ML_GroupDeletePointFile

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	FileName	STRING(20)	Name of the file to be deleted
VAR_OUTPUT	Done	BOOL	Delete file has been completed
	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

	Name	Type	Comment
	InBuffer	BOOL	Command successfully put in block buffer
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-146: Function block ML_GroupDeletePointFile

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	STRING(20)			"	Rising edge at "Execute"

Tab. 5-147: Input behavior of ML_GroupDeletePointFile

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_GroupDeletePoints

Brief description The function block "ML_GroupDeletePoints" deletes teach-in points of a kinematics.
(Refer to [Teach-In on page 49](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

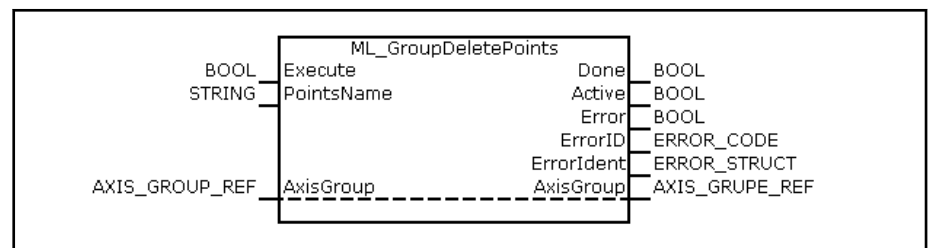


Fig. 5-89: Function block ML_GroupDeletePoints

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	PointsName	STRING(20)	Point(s) that are to be deleted Name may contain the wildcard "***"
VAR_OUTPUT	Done	BOOL	Deleting has been completed

ML_Robot.library

	Name	Type	Comment
	Active	BOOL	Deleting is active
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-148: Function block ML_GroupDeletePoints

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
PointsName	STRING(20)			"	Rising edge at "Execute"

Tab. 5-149: Input behavior of ML_GroupDeletePoints

- Signal-time diagram
- Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)
- Error handling
- The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_GroupLoadPointFile

- Brief description
- The function block "ML_GroupLoadPointFile" loads a point file of a kinematics.
(Refer to [Teach-In on page 49](#)).

 This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

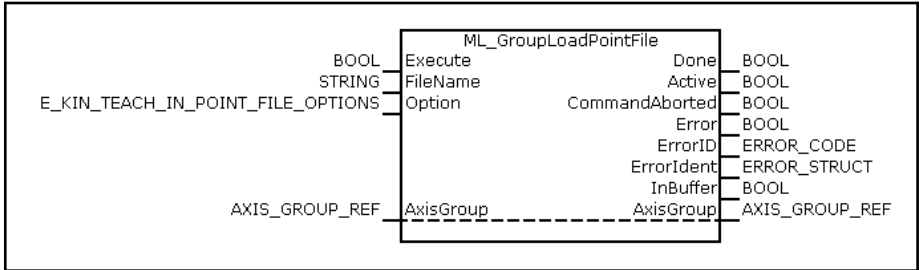


Fig. 5-90: Function block ML_GroupLoadPointFile

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	FileName	STRING(20)	Name of the file to be loaded
	Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS	Options for loading the file Refer to E_KIN_TEACH_IN_POINT_FILE_OPTIONS on page 310
VAR_OUTPUT	Done	BOOL	Definition was completed
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-150: Function block ML_GroupLoadPointFile

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	String(20)			"	Rising edge at "Execute"
Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS			E_GROUP_TEACH_IN_NEW	Rising edge at "Execute"

Tab. 5-151: Input behavior of ML_GroupLoadPointFile

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

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](#) on page 49).



This function block can only be used for Robot Control V2.

ML_Robot.library

Interface description

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

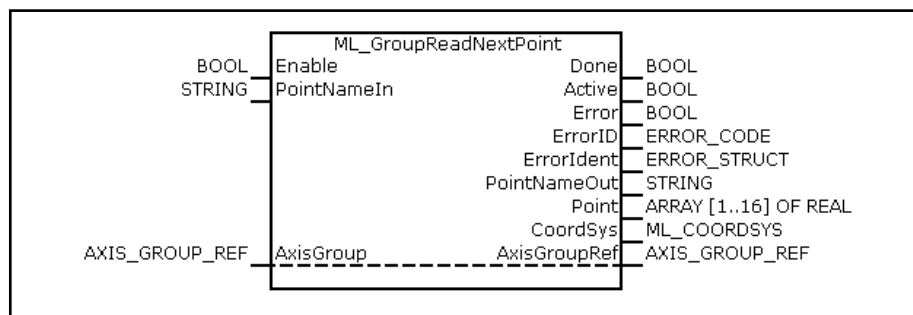


Fig. 5-91: Function block ML_GroupReadNextPoint

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
	PointNameIn	STRING(20)	Name of the point
VAR_OUTPUT	Done	BOOL	Reading of the point has been completed
	Active	BOOL	Reading of the point is active
	PointNameOut	STRING(19)	Name of the next point
	CoordSys	ML_COORDSYS	Coordinate system of the next point (Refer to ML_COORDSYS on page 312).
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-152: Function block ML_GroupReadNextPoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PointNameIn	STRING(20)			"	Continuous

Tab. 5-153: Input behavior of ML_GroupReadNextPoint

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_GroupReadPoint

Brief description The function block "ML_GroupReadPoint" returns the teach-in data of a kinematics point.
(Refer to [Teach-In](#) on page 49).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

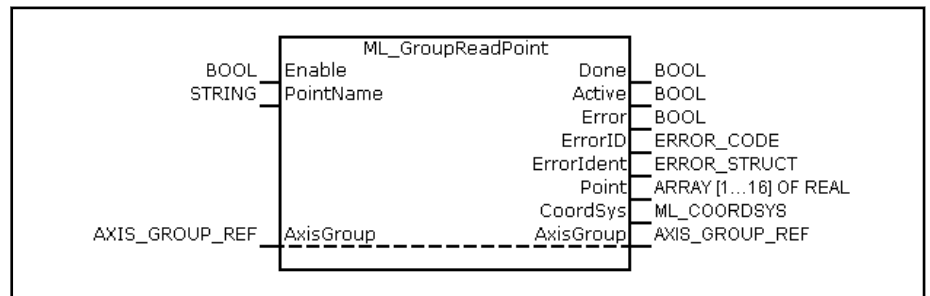


Fig. 5-92: Function block ML_GroupReadPoint

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Function block processing enabled
	PointName	STRING(20)	Name of the point
VAR_OUTPUT	Done	BOOL	Reading of the point has been completed
	Active	BOOL	Reading of the point is active
	CoordSys	ML_COORDSYS	Coordinate system of the point (Refer to ML_COORDSYS on page 312)
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-154: Function block ML_GroupReadPoint

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PointName	STRING(20)			"	Continuous

Tab. 5-155: Input behavior of ML_GroupReadPoint

Signal-time diagram Refer to [Inputs and outputs of state-controlled function clocks, permanent processing](#) on page. 199

Error handling The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

ML_Robot.library

The detailed descriptions are contained in:
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_GroupSavePointFile

Brief description The function block "ML_GroupSavePointFile" saves a point file of a kinematics.
(Refer to [Teach-In on page 49](#)).

 This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

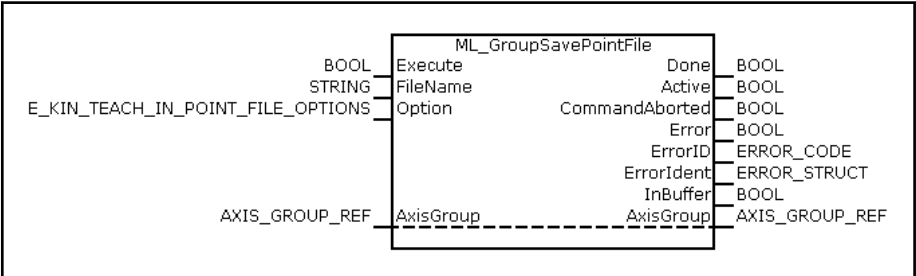


Fig. 5-93: Function block ML_GroupSavePointFile

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	FileName	STRING(20)	Name of the file to be saved
	Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS	Options for saving the file Refer to E_KIN_TEACH_IN_POINT_FILE_OPTIONS on page 310
VAR_OUTPUT	Done	BOOL	Save file has been completed
	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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-156: Function block ML_GroupSavePointFile

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FileName	STRING(20)			"	Rising edge at "Execute"
Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS			E_GROUP_TEACH_IN_NEW	Rising edge at "Execute"

Tab. 5-157: Input behavior of ML_GroupSavePointFile

- Signal-time diagram**
- Inputs and outputs of edge-controlled function blocks, final processing on page 196
 - InBuffer output at ML_Robot function blocks on page 204

Error handling The function block generates the following types of error messages:
 MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics
 DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

ML_GroupWritePoint

Brief description The function block "ML_GroupWritePoint" teaches a point of a kinematics.
 (Refer to [Teach-In on page 49](#)).



This function block can only be used for Robot Control V2.

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_Robot/Kinematic TeachIn

Interface description

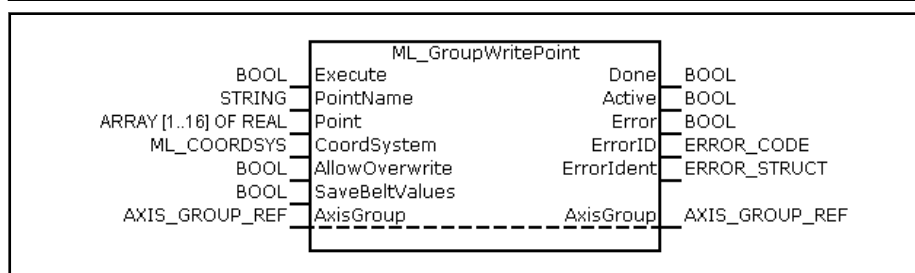


Fig. 5-94: Function block ML_GroupWritePoint

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Function block processing enabled
	PointName	STRING(20)	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 on page 312).
	AllowOverwrite	BOOL	Permits overwriting an existing point

ML_Robot.library

	Name	Type	Comment
	SaveBeltValues	BOOL	Saves the belt values of the point
VAR_OUTPUT	Done	BOOL	Definition was completed
	Active	BOOL	Definition is executed
	Error	BOOL	Indicates that an error occurred
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Kinematic reference of the commanded kinematics

Tab. 5-158: Function block ML_GroupWritePoint

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
PointName	STRING(20)			"	Rising edge at "Execute"
Point	ARRAY [1..16] OF REAL			16(0.0)	Rising edge at "Execute"
CoordSys	ML_COORDSYS			CS_MACHINE_COOR DINATES	Rising edge at "Execute" Continuous
AllowOverwrite	BOOL			FALSE	Rising edge at "Execute"
SaveBeltValues	BOOL			FALSE	Rising edge at "Execute"

Tab. 5-159: Input behavior of ML_GroupWritePoint

Signal-time diagram

Refer to [Inputs and outputs of state-controlled function clocks, permanent processing on page. 199](#)

Error handling

The function block generates the following types of error messages:
MLC_TABLE, 16#30 - error when accessing the axis/control.

The detailed descriptions are contained in:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

5.4 Data types

5.4.1 Data types, overview

The following data types support the function blocks of the "ML_Robot" library.



The library ML_Robot requires the ROB (robotics) function package for the control

ML_Robot data types

Area of application (folder in library)	
Data types	Description
Function block data types for configuration of a kinematics (Data Types/Config)	
MC_KIN_REF on page 300	Defines the method of a transformation MC_KIN_REF, initialization on page 300 MC_KIN_REF, forward on page 301 MC_KIN_REF, inverse on page 301 Error handling with interface functions on page 301
MC_KIN_REF_EXT on page 302	Defines the method of an extended transformation MC_KIN_REF_EXT, GetUsedAxes on page 303 MC_KIN_REF_EXT, DetermineMemory on page 302 MC_KIN_REF_EXT, initialization on page 303 MC_KIN_REF_EXT, forward on page 303 MC_KIN_REF_EXT, inverse on page 304 MC_KIN_REF_EXT, GetMechanic on page 303 Error handling with interface functions on page 304
ML_BELT_ERROR_REACTION_MODE on page 305	Defines which error reaction are triggered in case of belt synchronous operation
ML_KIN_AXES_CONFIG_DATA on page 305	The axis configuration of a kinematics is stored in the structure
ML_KIN_AXIS_CONFIG on page 305	Configuring an axis in a kinematics
ML_KIN_SAFE_ZONE_TYPE on page 306	Defines how a safe zone is configured
ML_KIN_USED_AXES on page 306	Defines the degree of freedom of a kinematics
Function block data types for collecting information about a kinematics (Data Types/Info)	
E_KIN_MECH_CMD on page 306	Defines how a mechanical partial object of a kinematics is to be generated and/or modified
ML_KINEMATICS_DATA_SM on page 307	Contains the information of the corresponding kinematics which can be accessed via Shared Memory
ML_KIN_CMD_INFO on page 308	Defines the structure of an information of a command in the block buffer
ML_KIN_CS_TYPE on page 308	Defines the transformation type
ML_KIN_INFO_TYPE on page 308	Defines the type of information of a command to be provided in the block buffer
ML_KIN_MECHANIC_DATA on page 309	Defines the mechanical robot structure
ML_KIN_PERF_DATA on page 309	Defines the performance measurement data
ML_POINTKIND on page 310	Defines the point type regarding the chronological order
Function block data types to move a kinematics (Data Types/Movement)	
ML_ARR_COORDS on page 311	Coordinates of a kinematic position in machine, world or workpiece coordinates
ML_CIRCMODES on page 311	Defines a circle segment to be traveled (input interpretation "AuxPoint")

ML_Robot.library

ML_COORDSYS on page 312	Defines in which coordinate system points are entered
ML_CYCLIC_MODE on page 313	Defines how to specify cyclic command values
ML_SET_MODE on page 313	Defines the blending method, i.e. whether the motion ends at the end point or if blending takes place
ML_SLOPE_TYPE on page 313	Defines the slope type of the motion
Data types for function blocks to save and reuse kinematic positions as points (Data Types/TeachIn)	
E_KIN_TEACH_IN_POINT_FILE_OPTIONS on page 310 on	Contains options to load and save point files

Tab. 5-160: ML_Robot library data types

5.4.2 Data types/Config

MC_KIN_REF

MC_KIN_REF, overview

The MC_KIN_REF interface defines transformation methods.

A function block has to implement this interface to register a transformation with the [MC_SetKinTransform function block on page 208](#)

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. 5-161: MC_KIN_REF interface

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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, page 305
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. 5-162: 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 301](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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. 5-163: 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 301](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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. 5-164: 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 301](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

Error handling with interface functions

If a floating point error occurs at 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

ML_Robot.library

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

MC_KIN_REF_EXT

MC_KIN_REF_EXT, overview

The MC_KIN_REF_EXT interface defines extended transformation methods.

A function block has to implement this interface to be able to register an extended transformation with function block:

[MC_SetKinTransformExt on page 210](#)

Method	Return value	Arguments	Description
DetermineMemory	UDINT	ACS TrafoChoise	Method to resolve ambiguities
Forward	UDINT	BCS ACS	Forward position kinematics
GetMechanic	UDINT	ACS MechanicData MechanicDataSize MechanicDataRetSize UpdateOnly	Method for mechanical information about a transformation
GetUsedAxes	UINT		Degrees of freedom of a transformation
Initialization	UDINT	AxesConfigData TransformationLengths TransformationFactors	Initialization functions of a transformation
Inverse	UDINT	BCS ACS	Inverse position kinematics

Tab. 5-165: MC_KIN_REF interface

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

MC_KIN_REF_EXT, DetermineMemory

This method is called by the system to enable the transformation to resolve ambiguities occurring during the next inverse position kinematics.

Argument	Type	Description
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in axis coordinate system
TrafoChoise	BOOL	Transformation can autonomously decide which ambiguous solutions are to be used during the next inverse position kinematics.

Tab. 5-166: Arguments of the interface function DetermineMemory

If the method cannot be completed successfully, the MLC error code has to be returned:

(see [Error handling with interface functions, page 304](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 basic coordinate system
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in axis coordinate system

Tab. 5-167: Arguments of the interface function "Forward"

If a point cannot be transformed, the MLC error code has to be returned:

(see [Error handling with interface functions, page 304](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

MC_KIN_REF_EXT, GetMechanic

This method is called by the system if the mechanical information of a point in the axis coordinate system (ACS) is to be queried. To complete the mechanical information, refer to the auxiliary function [ML_AddMechanicData on page 212](#).

Argument	Type	Description
ACS	REFERENCE TO ARRAY [1..16] OF LREAL	Point in axis coordinate system
MechanicData	REFERENCE TO ML_KIN_MECHANIC_DATA	ML_KIN_MECHANIC_DATA, page 309 , Required mechanical data
MechanicDataSize	UDINT	Number of elements in MechanicData
MechanicDataRetSize	REFERENCE TO UDINT	Number of completed elements in MechanicData
UpdateOnly	BOOL	Only return changeable entires

Tab. 5-168: Arguments of the interface function "Inverse"

If the method cannot be completed successfully, the MLC error code has to be returned:

(see [Error handling with interface functions, page 304](#)).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

MC_KIN_REF_EXT, GetUsedAxes

This method is called by the system to query possible degrees of freedom of the transformation. OR-linked values of type ML_KIN_USED_AXES have to be returned if the transformation supports one of the degrees of freedom.

(see [ML_KIN_USED_AXES, page 306](#))

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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.

ML_Robot.library

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 , page 305
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. 5-169: Arguments of the interface function "Initialization"

If the initialization cannot be completed successfully, the MLC error code has to be returned:

(see [Error handling with interface functions](#), page 304).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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.

Argument	Type	Description
OriginalCoordinates	REFERENCE TO ARRAY [1..16] OF LREAL	Point in basic coordinate system
JointCoordinates	REFERENCE TO ARRAY [1..16] OF LREAL	Point in axis coordinate system

Tab. 5-170: Arguments of the interface function "Inverse"

If a point cannot be transformed, the MLC error code has to be returned:

(see [Error handling with interface functions](#), page 304).

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 it is, 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

Unlike for simple user-defined transformation, an MLC error code may be returned as return value of the functions in case of extended user-defined transformation. This allows easier diagnostics for the user. An MLC error code has the form 16#Fxxxxxxx. If an interface function was executed without errors, a code in form of 16#0xxxxxxx has to be 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

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 218](#)

Element	Value	Description
ML_BELT_ERROR_NO_SPECIFIC_REACTION	0	No error reaction, the robot continues to move in synchronization with the belt
ML_BELT_ERROR_DESYNC_OVER_TIME	1	Desynchronization over time
ML_BELT_ERROR_DESYNC_WITH_LIMITS	2	Desynchronization using limit values

Tab. 5-171: Elements of the enumeration type ML_BELT_ERROR_REACTION_MODE

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_AXES_CONFIG_DATA

The axis configuration of a kinematics is stored in the structure "ML_KIN_AXES_CONFIG_DATA".

The following interface uses this structure:

- [MC_KIN_REF on page 300](#)
- [MC_KIN_REF_EXT on page 302](#)

Structure element	Type	Description
Configuration	ML_KIN_AXIS_CONFIG	Axis configuration Refer to ML_KIN_AXIS_CONFIG, page 305
ControlNumber	WORD	Control number of the axis
AxisNumber	WORD	Axis number of the axis

Tab. 5-172: ML_KIN_AXES_CONFIG_DATA data structure

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 305](#)

ML_Robot.library

Element	Value	Description
ML_KIN_AXIS_TYPE_NONE	16#00	Axis is not used in the kinematics
ML_KIN_AXIS_TYPE_POSITION	16#08	Axis is used as positioning axis of the kinematics
ML_KIN_AXIS_TYPE_ORIENTATION	16#10	Axis is used as orientation axis of the kinematics
ML_KIN_AXIS_TYPE_BELT	16#20	Axis is used as belt axis of the kinematics

Tab. 5-173: Definition of the elements of the enumeration type *ML_KIN_AXIS_CONFIG*

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_SAFE_ZONE_TYPE

Type definitions The enumeration type "ML_KIN_SAFE_ZONE_TYPE" is used by the function blocks [ML_GroupSetSafeZone on page 214](#) and [ML_RobotGetSafeZones on page 234](#).

It defines the configuration of a safe zone:

Element	Value	Description
ML_KIN_SAFE_ZONE_DEACTIVATE	0	Deactivated
ML_KIN_SAFE_ZONE_INSIDE_OK	1	Software limit switch
ML_KIN_SAFE_ZONE_OUTSIDE_OK	2	Safe zone

Tab. 5-174: Elements of the *ML_RobotGetSafeZones* enumeration type

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_USED_AXES

Type definitions The enumeration type "ML_KIN_USED_AXES" defines the degrees of freedom of a kinematics.

The enumeration type is used to define a kinematics:

[Extended user-defined transformation on page 176](#)

Element	Value	Description
X_AXIS	16#01	Degrees of freedom x-axis
Y_AXIS	16#02	Degrees of freedom y-axis
Z_AXIS	16#04	Degrees of freedom z-axis
A_AXIS	16#08	Degrees of freedom rotation around the x-axis
B_AXIS	16#10	Degrees of freedom rotation around the y-axis
C_AXIS	16#20	Degrees of freedom rotation around the z-axis

Tab. 5-175: Definition of the elements of the enumeration type *ML_KIN_USED_AXES*

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

5.4.3 Data types/Info

E_KIN_MECH_CMD

Brief description The enumeration type "E_KIN_MECH_CMD" defines how a mechanical partial object 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 309](#)

Element	Value	Description
KIN_MECH_CMD_CREATE_ROD	0	- 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	1	- Create a sphere - The sphere is at the zero point - The sphere has a radius of 1
KIN_MECH_CMD_POSITION	2	Change the position of an object to (x,y,z)
KIN_MECH_CMD_DIRECTION	3	Change the direction of an object to (x,y,z)
KIN_MECH_CMD_SIZE	4	Change the size of an object to (x,y,z)

Tab. 5-176: Elements of enumeration type E_KIN_MECH_CMD

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KINEMATICSDATA_SM

Brief description An instance of the structure "ML_KINEMATICSDATA_SM" contains information on the corresponding kinematics that can be accessed via **shared memory**.

The respective kinematic types support the following structural elements in ML_KINEMATICSDATA_SM:

Structure element	Description	Access
uiCoordSum_i	K-0-0007 , Axis number of kinematics	read
uiUpdateCount_i	Update counter (odd: update starts, even: update completed)	read
arPositionJointCoord_i	Generated command values for the assigned axes	read
arPositionOriginalCoord_i	K-0-0100 , Current position, Cartesian Machine Coordinate System	read
arPositionWorkpieceCoord_i	K-0-01050 , Current positions, Cartesian workpiece coordinates (Robot Control V1)	read
K-0-0021 Kinematic status		
Kin_StandStill	K-0-0021.05 , all axes of the kinematics are stopped	read
Kin_InPosition	K-0-0021.06 , the set end position is reached	read
Kin_BeltSynchronous	K-0-0021.07 , belt-synchronous	read
Kin_Warning	K-0-0021.08 , warning	read
Kin_Error	K-0-0021.09 , error	read
Kin_NotHomed	K-0-0021.10 , homing required; not all axes are referenced	read
Kin_Blending	K-0-0021.11 , blending enabled	read
Kin_DrivesOn	K-0-0021.14 All axes are ready for operation (AF)	read
Kin_ErrorReaction	K-0-0021.17 , error reaction enabled	read
Kin_FullyBeltSynchronous	K-0-0021.20 , belt-synchronous	read
Kin_WaitBeltStartCondition	K-0-0021.22 , waiting for belt-synchronous start condition	read

ML_Robot.library

Structure element	Description	Access
Kis_SplineCalcActive	K-0-0021.28 , calculating spline for blending motion	read
Kis_SplineValid	K-0-0021.29 , spline is valid	read
K-0-0022 Extended kinematic status		
Kis_Errorstop	K-0-0022.00 , kinematics is in "Errorstop"	read
Kis_Stopping	K-0-0022.01 , kinematics is in "Stopping"	read
Kis_Incomplete	K-0-0022.03 , not all axes are assigned to the kinematics (incomplete)	read
Kis_StandStill	K-0-0022.07 , kinematics is in "StandStill"	read
Kis_ManualMotion	K-0-0022.10 , kinematics is in jog state KinData[].stJogCom_q.enJogCmdVel ≠ KIN_JOG_STOP	read
Kis_GroupMoving	K-0-0022.12 , kinematics is in "GroupMoving"	read

Tab. 5-177: Structural elements in ML_KINEMATICSDATA_SM

Also refer to:

- [KinData\[\]](#) - Kinematic data in the shared memory on page 185

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_CMD_INFO

Brief description The structure "ML_KIN_CMD_INFO" defines the structure of information of a command in the block buffer.

The structure is used by:

- Function block [ML_RobotGetCmdInfo](#) on page 229

Structural element	Type	Description
infol	UDINT	Command ID
infoType	ML_KIN_INFO_TYPE	Information type of the command Refer to ML_KIN_INFO_TYPE on page 308
infoValue	REAL	Information value of the command

Tab. 5-178: ML_KIN_CMD_INFO data structure

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_CS_TYPE

Brief description The enumeration type "ML_KIN_CS_TYPE" defines the transformation type.

Element	Value	Description
CS_TYPE_CARTESIAN	0	Cartesian transformation
CS_TYPE_CYLINDRIC	1	Cylindrical transformation

Tab. 5-179: Elements of enumeration type ML_KIN_CS_TYPE

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_INFO_TYPE

Brief description The enumeration type "ML_KIN_INFO_TYPE" defines the type of information of a command to be provided in the block buffer.

The enumeration type is used by the following structure:

[ML_KIN_CMD_INFO](#) on page 308

Element	Value	Description
KIN_INFO_TYPE_ALL	0	Reads all information on all commands currently available in the block buffer
KIN_INFO_TYPE_PROGRESS	1	Displays the progress of the commands
KIN_INFO_TYPE_TOTAL_DISTANCE	2	Displays the total distance of the commands
KIN_INFO_TYPE_UID	3	Displays the unique ID of the commands
KIN_INFO_TYPE_STATE	4	Displays the state of the commands

Tab. 5-180: Elements of the ML_KIN_INFO_TYPE enumeration type

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_KIN_MECHANIC_DATA

Brief description Structure "ML_KIN_MECHANIC_DATA" defines the mechanical structure of a robot.

The structure is used by:

- [ML_RobotGetMechanic FB instance, page 231](#)
- FN [ML_AddMechanicData, page 212](#)

Structure element	Type	Description
cmd	E_KIN_MECH_CMD	Command to a mechanical partial object Refer to E_KIN_MECH_CMD, page 306
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. 5-181: Data structure ML_KIN_MECHANIC_DATA

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 233](#)

Structure element	Type	Description
name	STRING[15]	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. 5-182: Data structure ML_KIN_PERF_DATA

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_Robot.library

ML_KIN_SAFE_ZONE_DATA

Brief description The structure "ML_KIN_SAFE_ZONE_DATA" defines the data of the safe zone measurement.

The following function block uses this structure:

[ML_RobotGetSafeZones on page 234.](#)

Structure element	Type	Description
zoneNumber	UINT	Number of safe zones
zoneType	ML_KIN_SAFE_ZONE_TYPE	Type of the safe zone (Refer to ML_KIN_SAFE_ZONE_TYPE on page 306)
maxPos	ARRAY [1..3] OF REAL	Maximum position
minPos	ARRAY [1..3] OF REAL	Minimum position

Tab. 5-183: ML_KIN_SAFE_ZONE_DATA data structure

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

ML_POINTKIND

Brief description (Refer to [Robot Control, commands and functions, page 34](#)).

Element	Value	Description
POINTKIND_PREPARE	0	Point refers to block preparation
POINTKIND_EXECUTE	1	Point refers to block processing

Tab. 5-184: Definition of the elements of the enumeration type ML_POINTKIND

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

5.4.4 Data types/TeachIn

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 292](#)
- [ML_GroupSavePointFile on page 296](#)

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.

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

5.4.5 Data types/Movement

ML_ARR_COORDS

Brief description The array saves the coordinates of a kinematic position in joint, world or workpiece coordinates.

Type definition

```
TYPE (* 1..max. number of the kinematic coordinates *)
  ML_ARR_COORDS:ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL;
END_TYPE
```

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 248](#)

Element	Description
CM_BOARDER	The "AuxPoint" input defines a point to be traveled through on the circle segment

Tab. 5-185: Elements of the enumeration type ML_CIRCMODES

ML_Robot.library

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 224](#)
- [MC_MoveCircularAbsolute on page 248](#)
- [MC_MoveCircularRelative on page 250](#)
- [MC_MoveDirectAbsolute on page 252](#)
- [MC_MoveDirectRelative on page 255](#)
- [MC_MoveLinearAbsolute on page 257](#)
- [MC_MoveLinearRelative on page 260](#)
- [MC_SetCartesianTransform on page 206](#)
- [ML_GroupJogCont on page 265](#)
- [ML_GroupJogStep on page 267](#)
- [ML_GroupReadNextPoint on page 293](#)
- [ML_GroupReadPoint on page 295](#)
- [ML_GroupWritePoint on page 297](#)
- [ML_RobotGetTrafoValues on page 236](#)
- [ML_RobotMovePoint on page 237](#)
- [ML_RobotReadCyclicChannel on page 238](#)
- [ML_RobotTransformPoint on page 241](#)
- [ML_RobotWriteCyclicChannel on page 273](#)

Element	Value	Description
CS_MACHINE_COORDINATES	0	Machine coordinate system
CS_AXES_COORDINATES	1	Axis coordinate system
CS_BASE_COORDINATES	2	Base coordinate system
CS_GRIPPER_COORDINATES	4	Gripper coordinate system
CS_PRODUCT_COORDINATES_1	16#0003	Product coordinate system 1
CS_PRODUCT_COORDINATES_2	16#0103	Product coordinate system 2
CS_PRODUCT_COORDINATES_3	16#0203	Product coordinate system 3
CS_PRODUCT_COORDINATES_4	16#0303	Product coordinate system 4
CS_PRODUCT_COORDINATES_5	16#0403	Product coordinate system 5
CS_PRODUCT_COORDINATES_6	16#0503	Product coordinate system 6
CS_PRODUCT_COORDINATES_7	16#0603	Product coordinate system 7
CS_PRODUCT_COORDINATES_8	16#0703	Product coordinate system 8
CS_PRODUCT_COORDINATES_9	16#0803	Product coordinate system 9
CS_PRODUCT_COORDINATES_10	16#0903	Product coordinate system 10
CS_PRODUCT_COORDINATES_11	16#0A03	Product coordinate system 11

Element	Value	Description
CS_PRODUCT_COORDINATES_12	16#0B03	Product coordinate system 12
CS_PRODUCT_COORDINATES_13	16#0C03	Product coordinate system 13
CS_PRODUCT_COORDINATES_14	16#0D03	Product coordinate system 14
CS_PRODUCT_COORDINATES_15	16#0E03	Product coordinate system 15
CS_PRODUCT_COORDINATES_16	16#0F03	Product coordinate system 16

Tab. 5-186: Elements of the enumeration type *ML_COORDSYS*

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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_RobotWriteCyclicChannel on page 273](#)

Element	Value	Description
CYCLIC_MODE_POS_ABS	0	The cyclic command value specification is an absolute position
CYCLIC_MODE_POS_REL	1	The cyclic command value specification is a relative delta for the last commanded position in each cycle
CYCLIC_MODE_VEL	2	The cyclic command value specification is a velocity

Tab. 5-187: Definition of the elements of the enumeration type *ML_CYCLIC_MODE*

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 the following function blocks:

- [MC_MoveLinearAbsolute on page 257](#)
- [MC_MoveLinearRelative on page 260](#)
- [MC_MoveDirectAbsolute on page 252](#)
- [MC_MoveCircularAbsolute on page 248](#)

Element	Value	Description
SM_TO	0	The motion ends at the end point with a velocity of 0
SM_VIA	1	The motion can be blended

Tab. 5-188: Definition of the elements of the enumeration type *ML_SET_MODE*

Library ML_Robot for IndraMotion MLC 14VRS, function package ROB

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 257](#)
- [MC_MoveLinearRelative on page 260](#)

ML_Robot.library

- [MC_MoveDirectAbsolute on page 252](#)
- [MC_MoveDirectRelative on page 255](#)
- [MC_MoveCircularAbsolute on page 248](#)
- [MC_MoveCircularRelative on page 250](#)

Element	Value	Description
ST_BLOCK_SLOPE	0	The spatial deviation in the blending range depends on the velocity
ST_PROGRAM_SLOPE	1	The spatial deviation in the blending range depends on the velocity
ST_CONTINUOUS_SLOPE	2	The spatial deviation in the blending range does not depend on the velocity and path acceleration remains continuous

*Tab. 5-189: Elements of the enumeration type ML_SLOPE_TYPE***Library** ML_Robot for IndraMotion MLC 14VRS, function package ROB

6 Kinematic interface

6.1 Introduction

The **KinematicInterface** pools and extends Robot Control functionalities. It represents the counterpart for the **AxisInterface** and facilitates kinematics control, whilst also making it more convenient in doing so. This is why well-structured application programs for Robot Control applications can be developed in a very short period of time.

System requirements

The following hardware and software are necessary for the KinematicInterface:

- Software: IndraMotion MLC 14VRS
- Firmware: IndraMotion ML* 14VRS
- *CML control:*
 - IndraControl L25 or
 - IndraControl L45 or
 - IndraControl L65 or
 - IndraControl L75
- *XM control:*
 - IndraControl XM2
- Template: AxisInterface example project or GAT

Integration of KinematicInterface

The integration of the KinematicInterface depends on whether an existing or a new project is used as the basis.

In existing projects, the AxisInterface example project or a GAT version is a requirement.

The following procedure may be adopted:

1. Start GAT Wizard using the symbol .
2. Select the "Plug-in" node
3. Select the "Add plug-in" function
4. Select the "KinInterface" plug-in and confirm the import using **Apply**

In new projects, the KinematicInterface can be directly selected when a control is being created.

The following steps have to be performed:

1. Select "GAT Wizard" as the PLC application in the second step, "Configuration", when creating a control.
2. Select the plug-in "KinematicInterface" in the subsequent step, "PLC Application".
3. Finish creating the control. The KinematicInterface is automatically added together with the selected template.

6.2 Functional description

6.2.1 Overview

The Kinematic Interface is provided as a plug-in for the IndraMotion AxisInterface programming template. This is why the structural design and mode of operation is similar to that of the AxisInterface. What in the

Kinematic interface

AxisInterface constitutes the axis are the kinematics in the Kinematic interface.

The following figure shows the data structure of the Kinematic interface:

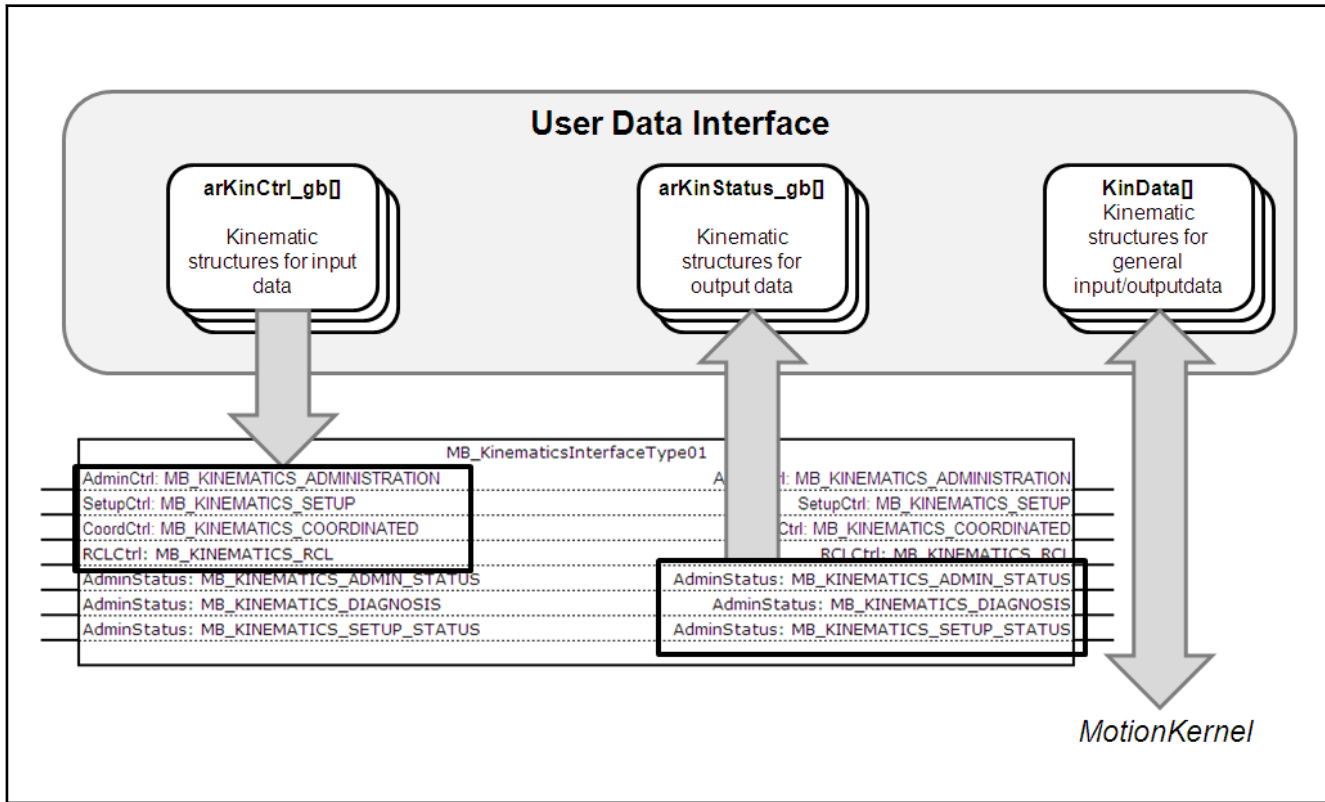


Fig. 6-1: Kinematic interface - Data structure of the interface

The following table shows the user interface for the Kinematic interface data structure:

User interface	Type	Description
arKinCtrl[]_gb	MB_KINEMATICS_CONTROL_TYPE01 (user extension: TE_KINEMATICS_CONTROL_TYPE01) MB_KINEMATICS_CONTROL_TYPE11 (user extension: TE_KINEMATICS_CONTROL_TYPE11)	Control structure, including command values and variables to activate the operation modes
arKinStatus[]_gb	MB_KINEMATICS_STATUS_TYPE01 (user extension: TE_KINEMATICS_CONTROL_TYPE01)	Status structure, including diagnostic information and acknowledgements for the operation modes
KinData[]		Status structure and control structure, including actual values and status bits (this is processed by MotionControl in the firmware)

Tab. 6-1: Kinematic interface - Data structure of the user interface

As for the programming template "AxisInterface example project", the kinematic interface provides an equivalent example project that is automatically generated when integrating the plug-in. This example project can be found under the tab **Logic ► Application ► AxisInterface ► Plugins ► KinInterface**.

The offset of 10 refers to the performance-optimized, event-controlled operating principle. Therefore, TYPE11 in the functional scope is equivalent to TYPE01, but operates in an event-controlled way.

Changes of MB_KinInterface-Type11 to MB_KinInterfaceType01

Apart from the points mentioned below, the event-controlled kinematic interface MB_KinInterfaceType11 has the same functions as the cyclic kinematic interface MB_KinInterfaceType01:

1. Subelements of the arKinCtrl[]_gb control structure contain interfaces and properties instead of structures. Using the interfaces and properties saves runtime, since a direct code is executed internally for the assignment to the control structure. This allows direct actions in the assignment and the data do not need to be cyclically scanned.
2. When specifying the operation modes using the bits (arKinCtrl_gb[].Admin._ OpMode.b), all the other bits for the operation modes are automatically deleted.
3. The CmdDone (arKinStatus_gb[].Admin.CmdDone) output is immediately reset when the operation mode and command values are changed. This means that the CmdDone output can be directly queried in the next code line after specification of the operation mode or command value.
4. The runtime for the cyclical call of the kinematic interface function block is considerably less.

6.2.2 Kinematic interface program

*The program **TE_KinInterfaceType01Prog** has the following tasks:*

- **Initializing the kinematic interface:**

When reaching the parameterization mode (phase 2), the kinematic interface is initialized for all configured kinematics using the function block **TE_KinematicsInitAllKinematicsType01** or **TE_KinematicsInitAllKinematicsType11**. When reaching the operating mode (phase 4), the initialization is started again to be able to detect changes in the parameterization

- **Cyclic call of the kinematic interface:**

After a successful initialization (signalized by "TE_KinInterfaceType01Prog.InitDone"), the kinematic interface is called cyclically. The **TE_KinematicsInterface** function block is called which is derived from the "MB_KinInterfaceType01" or "MB_KinInterfaceType11" function block and thus provides user extensions.

- **User extensions of the kinematic interface:**

The "TE_KinematicsInterface" function block provides the option to add more elements to the kinematic interface structure to optimally extend the user application. For more information, see [Kinematic interface user extensions, page 343](#).

6.2.3 Kinematic interface visualizations

Overview

Visualization masks are provided to the kinematic interface to provide a simple template interface to set and control the kinematics.

Kinematic interface

The following visualizations are available:

Visualization	Description
SystemOverview	
• KinSystem_Overview_All	Overall view on all defined kinematics, including simple diagnostics, status and coordinates
KinematicsModes	
• Kinematics_Overview	Shows current operating mode and current status of the kinematic interface together with the navigation buttons for all other visualizations
• KinHoming_mode	Commands and monitors the reference operating mode
• KinPosition_mode	Commands and monitors the positioning operating modes
Utilities	
• ConveyorBelt	Configures one or several belts
• Setup	Commands and monitors the jog mode
• TeachIn	Writing, reading and deleting a point or reading of a next point Saving, loading and deleting of point files
• ProductCoordSystem	Configures one or several product coordinate system

Tab. 6-2: Kinematic interface visualizations

The following global variables are used to control and for the access to system information within the visualizations:

- arKinCtrl_gb[]
- arKinStatus_gb[]
- KinData[]
- VisuKinematicsNo

System overview visualization

The **KinSystem_Overview_All** visualization allows the user to quickly connect all kinematics configured in the project. The standard kinematics displayed do not have a reference. How this reference has to be set is described below. Additional kinematics can be added for display in the offline mode.

The visualization delivers an overall system status and allows faults to be deleted. Individual kinematic operating elements provide the kinematic name and the kinematic number. Individual kinematic operating elements provide both diagnostics as well as the current position and world coordinates, together with the operation mode status displays.

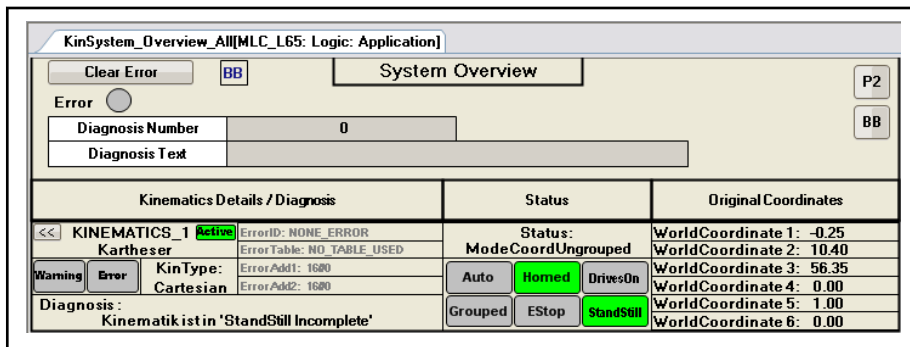


Fig. 6-2: KinSystem_Overview_All display

Clear Error This button is used to trigger "Clear Error" via the IMC interface.



Neither kinematic nor axis errors are deleted. These have to be deleted separately in the respective kinematic or axis overviews.

P2 This button allows the user to switch over into parameterization mode P2.

BB This button allows the user to switch over into the BB operating mode.

Adding a kinematics to the system overview

A kinematics is added to the system overview display in the same way as when adding a new visualization: by selecting the relevant element and specifying the kinematics reference. Listed below are two procedures for how a new kinematics can be added to the system overview:

Option 1: Creating a kinematics visualization using Frame

1. Open the relevant IndraWorks project and select the tab **Logic** ► **Application** ► **AxisInterface** ► **Plug-in** ► **KinInterface** ► **Visualizations** ► **SystemOverview**.
2. With IndraLogic in offline mode, double-click on the "KinSystem_Overview_All" visualization.
3. In the **Tools** window, select the button **Frame**  and generate a plan with corresponding size below the last kinematics.

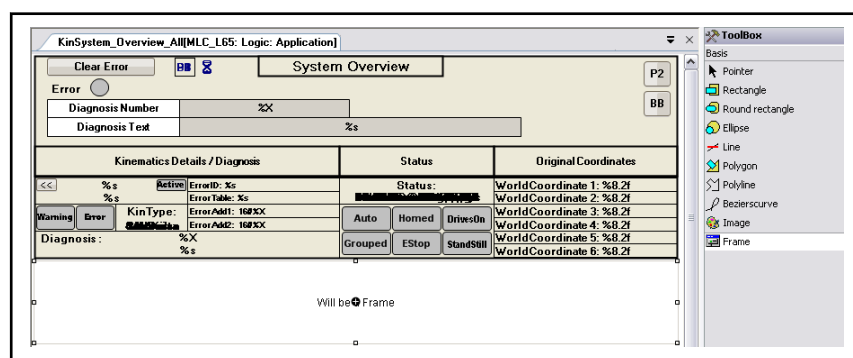


Fig. 6-3: Adding a kinematics to the system overview

4. Right-click on the frame and activate "Frame selection". Select the **KinSystem_Overview** element from the visualization selection window. A completely new system overview is displayed for the kinematics.
5. Open the Properties window for the recently inserted Frame and enter the name of the kinematics under **Referenced Visualizations** ► **KinSystem_Overview** ► **m_Input_KinRef**. The reference for the first kinematics can be adjusted in exactly the same way.



The name of the kinematics can, among other things, be found in the "MlcVarGlobal" global variable list.

The system overview is now available for the created kinematics.

Option 2: Copying an existing kinematics visualization

1. Open the relevant IndraWorks project and select the tab **Logic** ► **Application** ► **AxisInterface** ► **Plug-in** ► **KinInterface** ► **Visualizations** ► **SystemOverview**.

Kinematic interface

2. With IndraLogic in offline mode, double-click on the "KinSystem_Overview_All" visualization.
3. Copy the already present kinematics visualization and insert it as often as required.
4. Open the Properties window for the recently inserted kinematics visualizations and enter the name of the kinematics under **Referenced Visualizations** **Referenced Visualizations** **KinSystem_Overview** **m_Input_KinRef**. The reference for the first kinematics can be adjusted in exactly the same way.



The name of the kinematics can, among other things, be found in the "MlcVarGlobal" global variable list.

The system overview is now available for the specified kinematics.



One of the options mentioned above needs to be repeated for all kinematics added to the project.

Navigation system overview

From the kinematic system overview, access the visualization of the single kinematics "Kinematics_Overview" via . This button is located at the left of the "Kinematics Details / Diagnosis" column. All other visualizations can be accessed from the individual kinematics display.

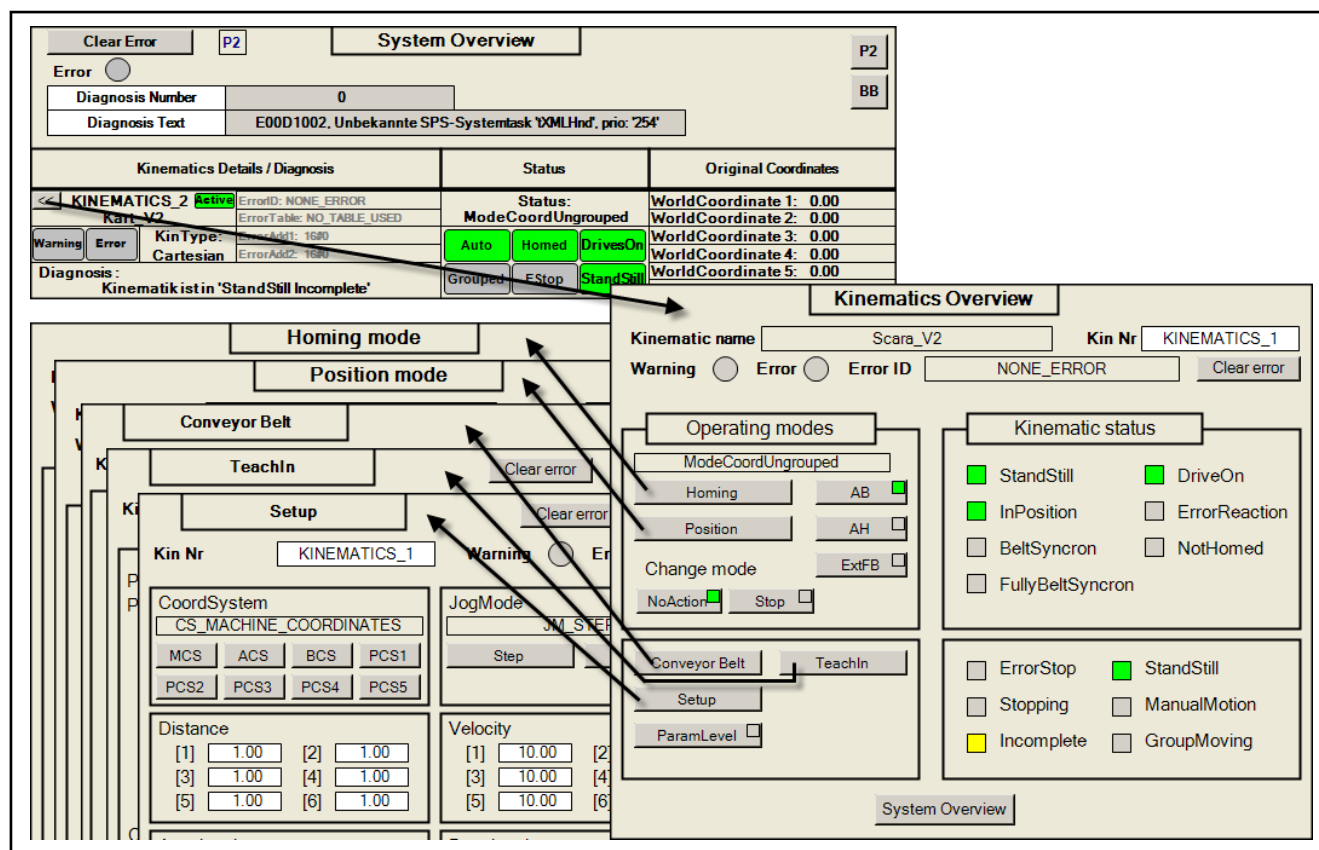


Fig. 6-4: Navigation system overview



Each visualization provides the means to switch back and forth between various kinematics by entering a kinematic number in the Kin Nr field.

Kinematic overview visualization

The **Kinematics_Overview** visualization displays clearly the most important functions used in the KinematicInterface and diagnostics used in the KinData interface.

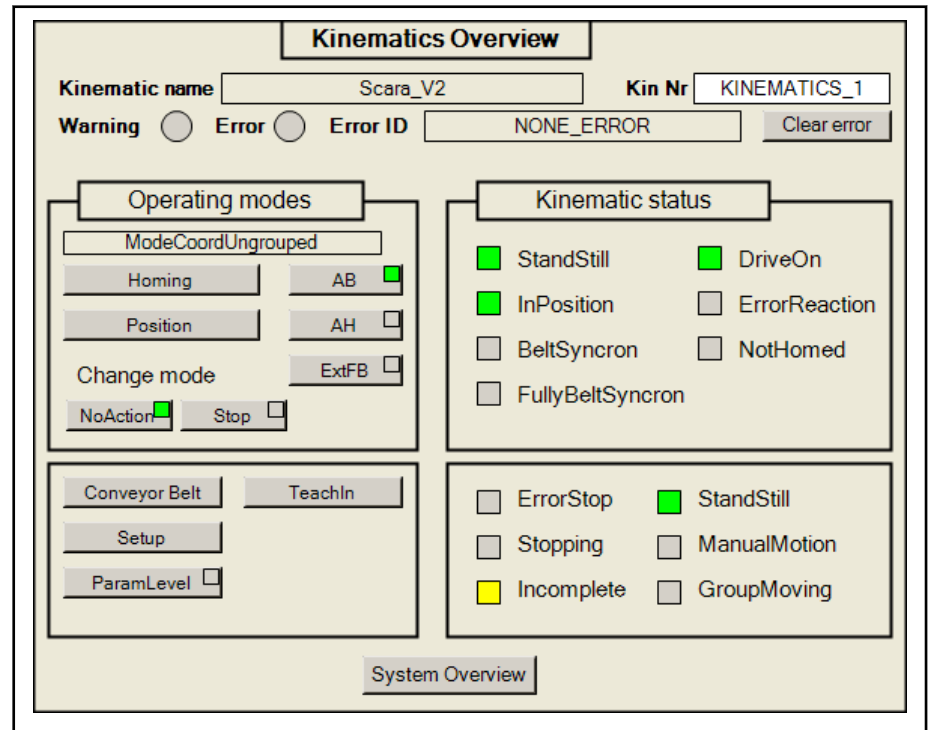


Fig. 6-5: Kinematic overview of visualization

The **Homing** and **Position** buttons can be used to switch over the visualization for the relevant operation mode where special values can be entered for the respective operation mode. The **AB**, **AH**, and **ExtFB** buttons allow the user to directly switch over into the relevant operation mode. The **NoAction** and **Stop** buttons help the user to define the reaction after the operation mode has been changed.

The visualizations saved under "Utilities" to configure a belt, jog a kinematics and teach points are opened via **Conveyor Belt**, **Setup** and **TeachIn**. If Jog mode is active, then this is signaled via the "SetupMode ACTIVE" message.

The right part of the visualization shows the diagnostic data of the KinData interface and displays the current PLCOpen status for the kinematics.

Homing overview visualization

The **KinHoming_mode** visualization is used for kinematic axis homing.

Kinematic interface

Homing mode

Kinematic name: Kin Nr:

Warning ☐ Error ☐ Error ID:

Homing sequence		Actual values	
1. Axis No	1	☐	1. Axis homed
2. Axis No	2	☐	2. Axis homed
3. Axis No	3	☐	3. Axis homed
4. Axis No	4	☐	4. Axis homed
5. Axis No	5	☐	5. Axis homed
6. Axis No	6	☐	6. Axis homed
7. Axis No	0	☐	7. Axis homed
8. Axis No	0	☐	8. Axis homed
9. Axis No	0	☐	9. Axis homed
10. Axis No	0	☐	10. Axis homed
11. Axis No	0	☐	11. Axis homed
12. Axis No	0	☐	12. Axis homed
13. Axis No	0	☐	13. Axis homed
14. Axis No	0	☐	14. Axis homed
15. Axis No	0	☐	15. Axis homed
16. Axis No	0	☐	16. Axis homed
		☐	Kinematics homed

Fig. 6-6: Homing overview visualization

The order of the axes to be homed can be defined by the user. By default, all axes assigned to the kinematics are entered corresponding to the configuration order. The axis number 0 is the cancel condition for the homing sequence.

In the right part of the visualization, the reference status of each specified axis as well as the kinematics is displayed. If the axis or the kinematics is in reference, the display element color changes to green.



The display element for "Kinematics homed" only turns green if the homing sequence has been processed successfully even if all axes are already referenced.

Positioning overview (position) visualization

The visualization **KinPosition_mode** serves the positioning of the kinematics.

Kinematic interface

Position mode

Kinematic name Scara_V2

Kin Nr KINEMATICS_1

Warning ☐ **Error** ☐

Error ID NONE_ERROR

Command values

Point

[1]	0.00	[2]	0.00	[3]	0.00
[4]	0.00	[5]	0.00	[6]	0.00
[7]	0.00	[8]	0.00	[9]	0.00
[10]	0.00	[11]	0.00	[12]	0.00
[13]	0.00	[14]	0.00	[15]	0.00
[16]	0.00				

Velocity 10.00

Acceleration 10.00

Deceleration 10.00

Jerk 0.00

BlendRadius 0.00

AuxPoint

[1]	0.00	[2]	0.00	[3]	0.00
[4]	0.00	[5]	0.00	[6]	0.00
[7]	0.00	[8]	0.00	[9]	0.00
[10]	0.00	[11]	0.00	[12]	0.00
[13]	0.00	[14]	0.00	[15]	0.00
[16]	0.00				

Current values

OrigCoord 1 0.00

OrigCoord 2 0.00

OrigCoord 3 0.00

OrigCoord 4 0.00

OrigCoord 5 0.00

OrigCoord 6 0.00

OrigCoord 7 0.00

OrigCoord 8 0.00

OrigCoord 9 0.00

OrigCoord 10 0.00

OrigCoord 11 0.00

OrigCoord 12 0.00

OrigCoord 13 0.00

OrigCoord 14 0.00

OrigCoord 15 0.00

OrigCoord 16 0.00

Velocity 0.00

Acceleration 0.00

Jerk 0.00

CoordSystem MCS ACS BCS PCS1

PCS2 PCS3 PCS4 PCS5 PCS6 PCS7

SlopeType BS CS

SetMode TO VIA

☐ Direct Abs ☐ Linear Abs ☐ Circular Abs

☐ Direct Rel ☐ Linear Rel ☐ Circular Rel

☐ Stop ☐ Interrupt ☐ Continue

☒ EnableCyclicScanning

Current coordination system CS_MACHINE_COORDINATES

Current slope type ST_BLOCK_SLOPE

Current set mode SM_TO

☐ InBuffer ☐ Stopping

☐ CmdDone ☐ ErrorStop

☒ InPosition ☐ GroupMoving

☐ BeltSynchron ☐ FullyBeltSynchron

Current operating mode ModeCoordUngrouped

Fig. 6-7: Positioning overview visualization

The following can be entered or selected:

- Position
- Velocity
- Acceleration
- Deceleration
- Jerk
- Blending radius
- Intermediate position
- Coordinate system
- Slope type of the motion
- Blending type

Via the interfaces **DirectAbs**, **LinearAbs**, etc. the relevant positioning operating mode can be selected. For more details, refer to the operating mode description.

Kinematic interface

At the right part of the visualization, the current position in world coordinates, the parameterized boundary conditions, important status elements of the kinematic interface and the KinData interface as well as the current operating mode.

Belt configuration (ConveyorBelt) visualization

The visualization **ConveyorBelt** is used to configure all belts of the kinematics.

Fig. 6-8: Belt configuration visualization

By specifying the belt configuration and the belt error reaction, the belt can be parameterized. Multiply the belt configuration overview according to the number of belts. The procedure is the same as for adding a kinematics to the system overview.

Refer to [Adding a kinematics to the system overview on page 319](#).

Enter a continuous belt number as referenced visualization. By means of this belt number, internal access to the `arKinCtrl_gb[].Admin.ConveyorBelt` and `arKinStatus_gb[].Admin.ConveyorBeltKinematicInterface` structures is possible.



The belt is internally configured with the `ML_SetBeltConfiguration` and `ML_SetBeltErrorReaction` function blocks. The current values are read out with the `ML_RobotGetBeltConfiguration` and `ML_RobotGetBeltErrorReaction` function blocks. Please refer to the relevant documents for further information.

Jog mode/Setup visualization

The **Setup** visualization provides the option of jogging the kinematics.

Setup Clear error

Kin Nr KINEMATICS_1 **Warning** ☐ **Error** ☐

CoordSystem
 CS_MACHINE_COORDINATES
 MCS ACS BCS PCS1
 PCS2 PCS3 PCS4 PCS5

JogMode
 JM_STEP
 Step Cont

Distance
 [1] 1.00 [2] 1.00
 [3] 1.00 [4] 1.00
 [5] 1.00 [6] 1.00

Velocity
 [1] 10.00 [2] 10.00
 [3] 10.00 [4] 10.00
 [5] 10.00 [6] 10.00

Acceleration
 [1] 10.00 [2] 10.00
 [3] 10.00 [4] 10.00
 [5] 10.00 [6] 10.00

Deceleration
 [1] 10.00 [2] 10.00
 [3] 10.00 [4] 10.00
 [5] 10.00 [6] 10.00

Jerk
 [1] 10.00 [2] 10.00
 [3] 10.00 [4] 10.00
 [5] 10.00 [6] 10.00

1+ 2+ 3+ 4+ 5+ 6+
 1- 2- 3- 4- 5- 6-
 Enable ☐ **EnableAck** ☐

Kin Overview System Overview TeachIn

Fig. 6-9: Jog mode/Setup visualization

The coordinates system in which jogging is supposed to take place is defined in the "CoordSystem" section. In the same way, whether a jog command triggers an incremental ("Step") or continuous ("Cont") travel motion is defined under "JogMode".

In the other sections, the distance, velocity, acceleration, deceleration, and the jerk can be defined for the first six axes in the kinematics. The entered values are interpreted as absolute (MCS, BCS, PCS) or percentage (ACS) values depending on the coordinate system selected. In the case of a percentage interpretation, the values relate to the corresponding limit values for the axis.

The 1+ to 6- buttons can be used to jog the first six axes in the kinematics in a positive or negative direction according to the settings.



Belt axes do not belong to the axis group and thus cannot be jogged.

TeachIn visualization

The **TeachIn** visualization provides the option of teaching points.

Kinematic interface

TeachIn

Kin Nr: KINEMATICS_%d

Warning ☐ Error ☐

Command values

PointName: %s

Point

[1]	% 2f	[2]	% 2f
[3]	% 2f	[4]	% 2f
[5]	% 2f	[6]	% 2f
[7]	% 2f	[8]	% 2f
[9]	% 2f	[10]	% 2f
[11]	% 2f	[12]	% 2f
[13]	% 2f	[14]	% 2f
[15]	% 2f	[16]	% 2f

CoordSystem: %s

MCS ACS BCS PCS1
PCS2 PCS3 PCS4 PCS5

AllowOverwrite: TRUE FALSE
SaveBeltValues: TRUE FALSE

FileName: %s

Option: %s

New Append Append Overwrite

Actual values

PointNameOut: %s

Point

[1]	%8.2f	[2]	%8.2f
[3]	%8.2f	[4]	%8.2f
[5]	%8.2f	[6]	%8.2f
[7]	%8.2f	[8]	%8.2f
[9]	%8.2f	[10]	%8.2f
[11]	%8.2f	[12]	%8.2f
[13]	%8.2f	[14]	%8.2f
[15]	%8.2f	[16]	%8.2f

CoordSystem: %s

Control

Point

Write ☐ Read ☐
Delete ☐ ReadNext ☐

PointFile

Save ☐ Execute ☐
Delete ☐
Load ☐

Kin Overview System Overview Setup

Fig. 6-10: TeachIn visualization

Required entries for writing, reading, and deleting points are defined in the first part of the "Command values" section:

- PointName - Name of the point
- Point - 16 coordinates for the point
- CoordSystem - Coordinate system where the point is valid
- AllowOverwrite - Allow or disallow the overwriting of points
- SaveBeltValues - Save or do not save belt values in the point

Required entries for saving, loading, and deleting point files are defined in the second part of the "Command values" section:

- FileName - Name of the point file
- Option - Options for saving and loading the point file (for more detailed information, see E_KIN_TEACH_IN_POINT_FILE_OPTIONS)

After a point or the next point has been read, the relevant outputs are displayed in the "Actual values" section. The name, the coordinates, and the coordinate system of the point or next point are shown.

In the "Control" section, the buttons **Write**, **Read**, **ReadNext**, **Delete** (for points), and **Save**, **Delete**, **Load** (for point files) can be used to select the required command, and the **Execute** button to trigger it.

PCS configuration (ProductCoordSystem) visualization

The visualization **ProductCoordSystem** is used to configure a product coordinate system.

Fig. 6-11: PCS configuration visualization

By specifying the origin and the coordinate system type (CoordType), the corresponding product coordinate system can be configured. The coordinate system number is selected via the "PCS no.". By means of this coordinate system number, internal access to the `arKinCtrl_gb[].Admin.PCSConfig` und `arKinStatus_gb[].Admin.PCSStatus` KinematicInterface structures is possible.



The product coordinate system is internally configured with the `MC_SetCartesianTransform` or `ML_SetCylindricTransform` function blocks. The current configuration is read out via the `ML_RobotGetTrafoValues` function block.

6.2.4 Operation modes

Overview

There are two methods to activate an operation mode:

1. Selection of Enum values

Assigning a value of the MB_KINEMATICS_MODE TYPE:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordAH;
```

2. Using the bit access

Setting a bit using the bit access functionality of TYPE MB_KINEMATICS_MODE_BITS..

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AH:= TRUE;
```



Only 1 bit can be set in arKinCtrl_gb[].Admin._OpMode. Setting several bits causes a error in the kinematic 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 attributes (e.g. Point, Velocity) that are used or whose default values were changed have to be defined before the current operation mode switch is executed.

The following section describes the operating modes supported by the kinematic interface and the attributes that can be assigned.



The kinematic interface internally uses the AxisInterface to execute switchings (e.g. grouping/ungrouping of all axes, referencing of axes).

The following operating modes are supported:

- ModeCoordAB, [Kinematic ready, drives ready](#), page 329
- ModeCoordAH, [Kinematic ready, drives halt](#), page 329
- ModeCoordHoming, [Kinematic homing](#), page 330
- ModeCoordPosDirAbs, [Absolute PTP positioning](#), page 330
- ModeCoordPosDirRel, [Relative PTP positioning](#), page 331
- ModeCoordPosLinAbs, [Absolute linear positioning](#), page 332
- ModeCoordPosLinRel, [Relative linear positioning](#), page 333
- ModeCoordPosCircAbs, [Absolute circular positioning](#), page 334
- ModeCoordPosCircRel, [Relative circular positioning](#), page 335
- ModeCoordPosJumpAbs, [Absolute Jump positioning](#), page 336
- ModeCoordStopping, [Kinematic stop](#), page 337
- ModeCoordInterrupted, [Operating mode Interrupted](#), page 338
- ModeCoordContinue, [Operating mode Continue](#), page 338
- ModeCoordExternalFB, [Operating mode external function block](#), page 339
- ModeCoordCyclic, [Cyclic position command value specification](#), page 339

Kinematic ready, drives ready

Activating this operating mode ungroups the kinematics. All axes of the kinematics are switched to AB (drive ready). In this operating mode, the kinematic interface is deactivated and the kinematics is in the PLCOpen state "Standstill incomplete".

The following command activates the operating 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. 6-3: Attribute Kinematic ready, drives ready



When selecting the operating mode "ModeCoordAB", the kinematic interface returns the operating mode "ModeCoordUngrouped" in the status structure!

Kinematic ready, drives halt

Activating this operating mode ungroups the kinematics. All kinematic axes are switched to AH (drive halt).

In this operating mode, the kinematic interface is deactivated and the kinematics is in the PLCOpen state "Standstill incomplete". The axes are transferred to the PLCOpen state "Standstill", i.e. they can be commanded using the AxisInterface or separate PLCOpen function blocks.

The following command activates the operating 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. 6-4: Attributes kinematic ready, drives halt

Kinematic interface



When selecting the operating mode "ModeCoordAH", the kinematic interface returns the operating mode "ModeCoordUngrouped" in the status structure!

Kinematic homing

If the operating mode is activated, the kinematic axes are homed. Homing is executed via the SetupMode of the AxisInterface.

The following command activates the operating 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. 6-5: Attributes kinematic homing



If the kinematic is already grouped when selecting the operating mode "ModeCoordHoming", homing is not possible anymore and the kinematic interface returns an error message.

Absolute PTP positioning

Activating this operating mode causes an absolute PTP motion to a previously specified target position.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating 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. 6-6: "Absolute PTP Positioning" attributes

Relative PTP positioning

Activating this operating mode causes an relative PTP motion to a previously specified position position.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosDirRel;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_DIR_REL := TRUE;
```

Kinematic 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. 6-7: "Relative PTP Positioning" attributes

Absolute linear positioning

Activating this operating mode causes an absolute linear motion to a previously specified target position.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosLinAbs;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_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. 6-8: "Absolute Linear Positioning" attributes

Relative linear positioning

Activating this operating mode causes an relative linear motion to a previously specified position position.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosLinRel;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_LIN_REL := TRUE;
```

Kinematic 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. 6-9: "Relative Linear Positioning" attributes

Absolute circular positioning

When activating this operating mode, an absolute circular motion is executed to a specified target position and around a specified intermediate point.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosCircAbs;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_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. 6-10: "Absolute Circular Positioning" attributes

Relative circular positioning

When activating this operating mode, a relative circular motion is executed around a specified position difference and around a specified intermediate point.

The kinematic is grouped and changes to the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosCircRel;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_CIRC_REL := TRUE;
```

Kinematic 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. 6-11: "Relative Circular Positioning" attributes

Absolute Jump positioning

By activating this operating mode, the absolute jump motion is executed on a previously specified target position.

The kinematic is grouped and changes to the "GroupMoving" PLCOpen state.

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordPosJumpAbs;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_POS_JUMP_ABS := TRUE;
```

The following table contains the attributes supported by this operating mode:

Element	Name	Type	Default	Cyclic 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
	CoordMode.StartHeight	REAL	0.0	No
	CoordMode.MaxHeight	REAL	0.0	No
	CoordMode.EndHeight	REAL	0.0	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_JUMP_ABS	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab. 6-12: Attribute position Absolute Jump

Kinematic stop

Activating this operating mode stops the kinematics by maintaining the grouping.

The kinematic changes to PLCOpen state "Stopping".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordStopping;
```

or

```
arKinCtrl_gb[ ].Admin._OpMode.b.MODE_COORD_STOPPING := TRUE;
```

Kinematic 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
	AdminCtrl.StopDeceleration	REAL	999.0	Yes
	AdminCtrl.StopJerk	REAL	9999.0	Yes
arKinStatus_gb[]	Admin.MODE_COORD_STOPPING	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab. 6-13: Kinematic stop attribute

Operating mode "Interrupted"

Activating this operating mode interrupts the kinematic motion by maintaining the grouping.

The kinematic remains in the PLCOpen state "GroupMoving".

The following command activates the operating 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. 6-14: Operating mode "Interrupted" attribute

Operating mode "Continue"

Activating this operating mode starts a motion that has previously been interrupted.

The kinematic is in the PLCOpen state "GroupMoving".

The following command activates the operating mode:

```
arKinCtrl_gb[ ].Admin._OpMode.en:= ModeCoordContinue;
```

Kinematic 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. 6-15: Operating mode "Continue" attribute

Operating mode "external function block"

Activating this operating mode groups the kinematic and waits for an external motion command.

The kinematic changes to PLCOpen state "StandStill Complete".

The following command activates the operating 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. 6-16: Attributes, operation mode "external function block"

Cyclic position command value specification

By activating this operating mode, the cyclic position command value specification is activated.

The kinematic is grouped and changes to the "GroupMoving" PLCOpen state.

The following command activates the operating mode:

```
arKinCtrl_gb[].Admin._OpMode.en:= ModeCoordCyclic;
```

or

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_CYCLIC := TRUE;
```

Kinematic interface

The following table contains the attributes supported by this operating mode:

Element	Name	Type	Default	Cyclic scanned
arKinCtrl_gb[]	Admin.ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Yes
	CoordMode.Point	ARRAY [1..16] OF REAL	16(0.0)	Yes
	CoordMode.Deceleration	REAL	10.0	No
	CoordMode.Jerk	REAL	100.0	No
	CoordMode.CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	No
	CoordMode.SetMode	ML_SET_MODE	SM_TO	No
	CoordMode.BlendingRadius	REAL	0.0	No
	CoordMode.CyclicMode	ML_CYCLIC_MODE	CYCLIC_MODE_POS_ABS	No
	CoordMode.CyclicChannelType	ML_CYCLIC_CHANNEL_MODE_SUBTYPE	ML_CYCLIC_CHANNEL_MODE_LINEAR	No
	CoordMode.CyclicVelocity	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(0.0)]	No
arKinStatus_gb[]	Admin.MODE_COORD_POS_CYCLIC	BOOL		Not applicable
	Admin.CmdDone	BOOL		Not applicable
	Admin.InBuffer	BOOL		Not applicable

Tab. 6-17: Attributes of the cyclic position command value specification

6.2.5 Dialog-based kinematics commanding

For comfortable commanding of kinematics, **dialog-based kinematics commanding** is available for all GAT programming templates of Bosch Rexroth. This commanding enables input of all required command values of the selected kinematics in a dialog and generation of the respective PLC code. This enables successful and time-saving commanding of kinematics without knowing the precise syntax.

Requirements Before the initial use of the dialog-based kinematics commanding, the following requirements have to be met:

- The "KinematicInterface" plug-in has to be added
- The kinematics container has to be added (not for AxisInterface example project)
- All kinematics of the unit/module have to be assigned via the GAT wizard (not for AxisInterface example project)

- All axes of kinematics of the unit/module have to be assigned to the respective unit/module (not for AxisInterface example project)

Adding the kinematics container

The kinematics container is added via the GAT wizard. Depending on the selected GAT variant, "Add kinematics container" can be added via the context menu of the unit or the module. However, this is only possible if the "KinematicInterface" plug-in has already been added.

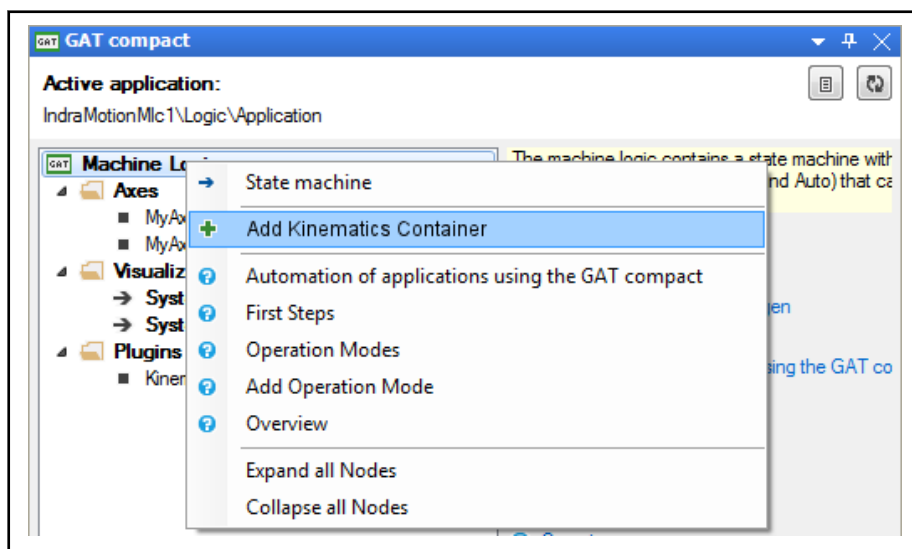


Fig. 6-12: Adding the kinematics container

Calling the ST code templates

The context menu "ST code templates ..." provides access to dialog-based commanding. Differently from axis commanding, this dialog is available in the entire ST program code (ST - "Structured Text").

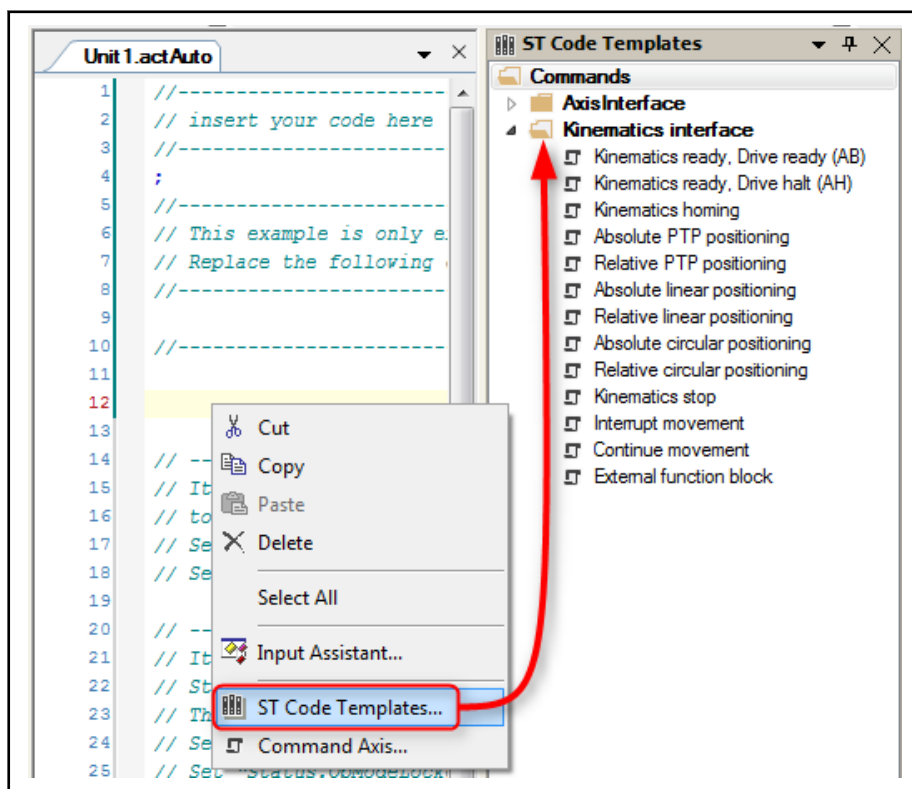
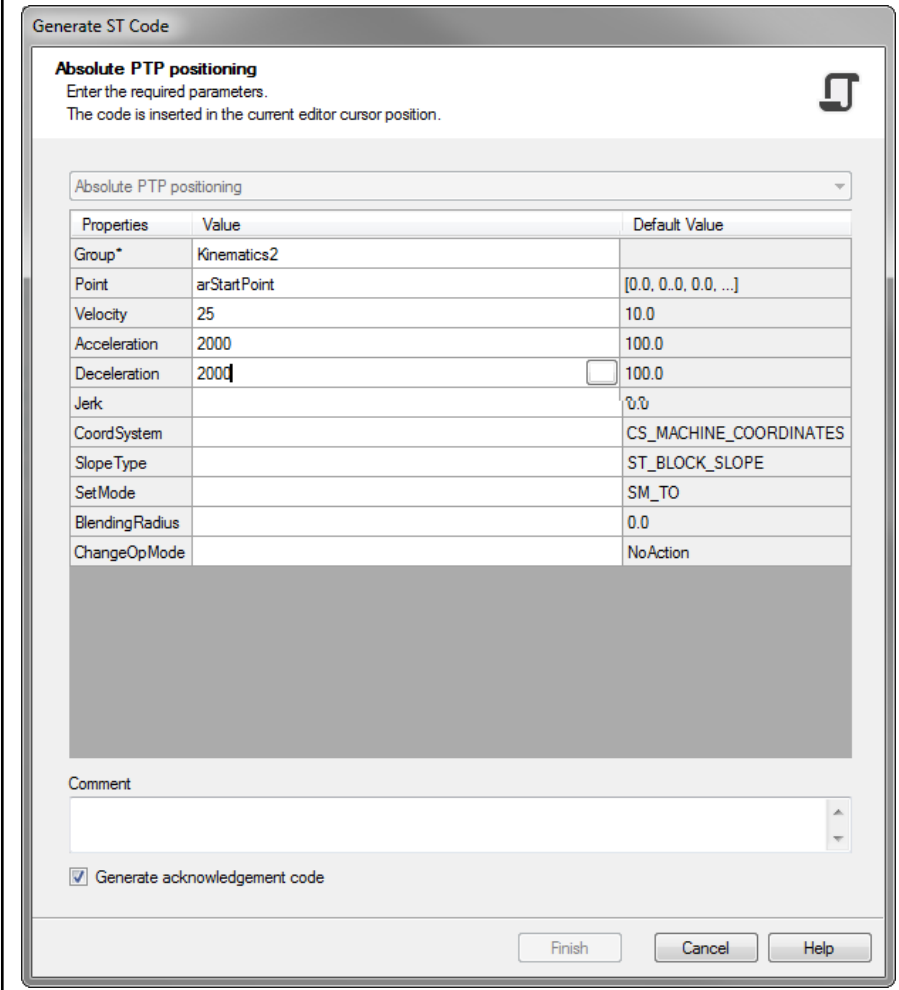


Fig. 6-13: Calling the dialog-based kinematics commanding

Kinematic interface

With a double click on one of the templates, an additional window is displayed to specify all values required for the operation mode.



Generate ST Code

Absolute PTP positioning
Enter the required parameters.
The code is inserted in the current editor cursor position.

Absolute PTP positioning

Properties	Value	Default Value
Group*	Kinematics2	
Point	arStartPoint	[0.0, 0.0, 0.0, ...]
Velocity	25	10.0
Acceleration	2000	100.0
Deceleration	2000	☐ 100.0
Jerk		0.0
CoordSystem		CS_MACHINE_COORDINATES
SlopeType		ST_BLOCK_SLOPE
SetMode		SM_TO
BlendingRadius		0.0
ChangeOpMode		NoAction

Comment

☒ Generate acknowledgement code

Finish Cancel Help

Fig. 6-14: Absolute linear positioning via dialog-based kinematics commanding

For properties marked with a *, values have to be assigned. All additional properties do not necessarily require any assigned values. In this case, the standard values apply and no code is generated. Values can be specified by direct input or by means of PLC variables. The input assistance (F2) can also be used. Optional adding of acknowledgment codes can be enabled or disabled with the "Generate acknowledgement code" checkbox.

Depending on where the generated PLC code is to be inserted and the GAT variant used, the global structures of the cinematics interface or the kinematics container (kinematics) are addressed. This is automatically recognized by dialog-based kinematics commanding.

After required values have been input and the **Complete** button pressed, the PLC code is inserted at the current cursor position in the program code (in the following, illustrated at the "actAuto" unit action of a GAT decentral project).


```
12 Kinematics.Kinematics2.Ctrl.CoordMode.Point := arStartPoint;  
13 Kinematics.Kinematics2.Ctrl.CoordMode.Velocity := 25;  
14 Kinematics.Kinematics2.Ctrl.CoordMode.Acceleration := 2000;  
15 Kinematics.Kinematics2.Ctrl.CoordMode.Deceleration := 2000;  
16 Kinematics.Kinematics2.Ctrl.Admin._OpMode.en := ModeCoordPosLinAbs;  
17 (///Code example to check if the commanded operation mode is acknowledged  
18 // Note: it can be necessary to wait one PLC cycle before checking the acknowledge  
19 (e.g. when the previous OpMode was the same)  
20 IF (Kinematics.Kinematics2.Ctrl.Admin._OpModeAck = ModeCoordPosLinAbs AND  
21     Kinematics.Kinematics2.Ctrl.Admin.CmdDone (* or Admin.InBuffer *)) THEN  
22     // commanded operation mode is acknowledged  
23     ;  
24 END_IF  
25 *)
```

Fig. 6-15: Automatically generated program code of the "Absolute linear positioning" template

This program code corresponds to kinematics commanding in the GAT area in connection with the kinematics container. The acknowledgment code is commented out and inserted and can be individually adjusted by the user.

User extension The dialog-based kinematics commanding function also supports the KinematicInterface extensions programmed by the user. These extensions can also be implemented into dialog-based kinematics commanding, equivalent to dialog-based axis commanding. For further information, please refer to the GAT documentation (see **Rexroth IndraLogic XLC IndraMotion MLC 14VRS, Generic Application Template**, DOK-XLCMLC-TF*GAT**V14-APRS-DE-P, R911341486).

6.2.6 Kinematic interface user extensions

The `arKinCtrl_gb[]` and `arKinStatus_gb[]` structures can be extended by the user to adapt the kinematic interface to special applications.

This section shows how the kinematic interface is extended by adding a new functionality. In the following it is assumed that the extension structures prepared in the kinematic interface will be used. The following describes the required changes in the POU's and structures.

Notes regarding the implementation of user extensions

The user extensions were implemented using the object-oriented extensions from IndraLogic 2G.

Please note the following particularities:

- The **TE_KinematicsInterface** function block is derived from the basic function block "MB_KinInterfaceType01". The keyword "SUPER" can be used to call the basic function block or its methods/actions. For example, often the method `mSetError` of `MB_KinInterfaceType01` is called using `SUPER^.mSetError(...)`; to enter errors in the diagnostic system
- Within the **TE_KinematicsInterface** function block, 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. are part of the basic function block and should not be used as no user extensions are available.

- The inputs of the **TE_KinematicsInterface** function block are defined as references. In the method "mInitExtension", the reference are initialized

Kinematic interface

once and do not have to be transferred again during a cyclic call of the function block

MB_KinInterfaceType01 extension

Extending the kinematic interface structure MB_KinInterfaceType01

If the structures `TE_KINEMATICS_CONTROL_TYPE01` and `TE_KINEMATICS_STATUS_TYPE01` structures are to be extended with additional substructures, the following additional steps must be added to the procedure listed below:

1. Enter new elements in `TE_KINEMATICS_CONTROL_TYPE01` or `TE_KINEMATICS_STATUS_TYPE01`
2. Add the additional inputs at the `TE_KinematicsInterface` function block as REFERENCE TO
3. Initialize the references in the method "mInitExtension" of the FB `TE_KinematicsInterface`



If the substructures `TE_KINEMATICS_ADMINISTRATION`, `TE_KINEMATICS_COORDINATED` etc. are to be extended, only step 1 of the previously described work flow is required for the respective structure. Insert the extension after (* add your extensions below here | Erweiterungen hier einfüegen*) in this structure. Subsequently, the newly created input or output is available via the corresponding *Ext structure element.

Extending the TE_KinematicsInterface function block

The last step in the user extension sequence is to extend the `TE_KinematicsInterface` function block so that the program code of the user extension can be inserted.

1. In the project tree, right-click to select **TE_KinematicsInterface** and **Add ► Action**
2. Assign a name matching the extension to the new action.
3. Insert the program code in the new action.
For access to the `arKinCtrl_gb[]` and `arKinStatus_gb[]` structures, use the inputs with the suffix *Ext (e.g. `AdminCtrlExt`, `DiagStatusExt`, etc.).
4. In the `TE_KinematicsInterface` function block, call the new action.
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 kinematic interface structure and can be viewed using the `UserVarGlobal` variables.

MB_KinInterfaceType11 extension

Extending the existing interface

The following steps show how the kinematic interface can be extended by the "Homing Sequence" function. The extension affects an already existing interface and is contained in the kinematic interface example project and in all GAT projects operating on an open code basis. To understand the following steps, the code for the example project should be compared in parallel. The following steps can also be used analogously for other extensions.

- Extend the ITF_KINEMATICS_ADMINISTRATION_EXT interface by the "HomingSequence" property. The "Get" and "Set" parts remain empty
- Create a new "HomeMode" folder below the TE_KinematicsInterface function block
- Below the new folder "HomeMode", add the above mentioned property "HomingSequence" and implement the program code
- The Set Property to be implemented checks the change in the input value and uses the value on a local variable
- The Get Property to be implemented copies the value of the local variable in the direction of the application
- Add the required additional local variables in the declaration for the TE_KinematicsInterface function block
- Implement the additional "actModeHome" action
- Call the additional action "actModeHome" in the TE_KinematicsInterface function block

Adding a new interface

The following steps show how the kinematic interface is extended by one new interface.

- Add the new interface below the "Control" folder
- Add the new properties below the new interface
- Also add these properties below the TE_KinematicsInterface function block
- In the declaration for the TE_KinematicsInterface function block, enter the implementation of the new interface using the keyword IMPLEMENTS <InterfaceName>
- Implement the Get and Set property below the TE_KinematicsInterface function block using the appropriate program code
- Add another action below the TE_KinematicsInterface function block
- Add the new interface in the TE_KINEMATICS_CONTROL_TYPE11 data structure
- Initialize the new interface in the mInitExtension method of the TE_KinematicsInterface function block
- Add the required additional local variables in the declaration for the TE_KinematicsInterface function block
- Optional: Extend the status structure. Use the same procedure as in the extensions for the Type01 function block

6.2.7 Upgrading to an event-controlled kinematic interface (type 11)

By default, the kinematic interface plugin contains the kinematic interface with cyclic operation (MB_KinInterfaceType01).

The event-controlled kinematic interface (Type11) is available for reducing the runtime and to improve performance.

Existing PLC applications with the kinematic interface plugin (Type01) can be converted to the event-controlled kinematic interface (Type11) using the following steps.

Kinematic interface



The event-controlled kinematic interface can only be used in combination with the event-controlled AxisInterface.

To upgrade to the event-controlled AxisInterface, see Upgrading to an event-controlled AxisInterface (documentation IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template).

1. Create IndraWorks project archive as a backup copy
2. Deleting plugin "Kinematic interface" in the GAT Wizard (if not already deleted by changing to an event-controlled AxisInterface)
3. Adding plugin "Kinematic interface (event-controlled)" using the GAT Wizard
4. Save, compile, and load the project
5. Check for correct functioning

6.3 KinematicInterface HMI template

6.3.1 Introduction

In order to make use of the KinematicInterface in connection with the VH2110 visualization device, the GAT-HMI template was extended by kinematics functions. The template contains basic functions like status of the kinematics, jogging and positioning of kinematics and their axes as well as teaching of points. It is already prepared and ready for activation for all GAT variants (central, decentral and compact) and can be individually extended and modified as necessary.

The KinematicInterface HMI template is based on the GAT-HMI template and extends it. For this reason, only kinematics-specific visualization parts and their function are described. For further information on the general GAT-HMI template, refer to the GAT-HMI Template documentation.



The Kinematic interface HMI template is **only** available for the VH2110 visualization device with a resolution of 1280 x 800!

To edit and execute the WinStudio visualization, at least the following development environment license (or higher) is required:

WinStudio development license, max. 1,500 variables (part no. R911323596).

Prerequisites

By adding the VH2110 visualization device and the GAT HMI template, the parts of the KinematicInterface HMI template are automatically added. For their use, the following requirements have to be met:

- The project has to be based on one of the GAT variants central, decentral or compact
- The added "KinematicInterface" plug-in requires least version 14.8.0.0
- The "HMI_KinInterface" function block has to be cyclically called (for example, in a cyclic action like "actCyclic...")
- When calling the "HMI_KinInterface" function block, the IP address of the visualization device has to be specified
- UDP communication has to be activated on the HMI device and the IP address of the control has to be entered (for reading out of the potentiometer)

Communication between HMI and PLC

When calling the "HMI_KinInterface" function block, it has to be observed that initialization of the KinematicInterface is completed (`TE_KinInterfaceType01Prog.InitDone = TRUE`) before the function block is activated.

Communication between the HMI and the control is realized using the variables of the global "Global_HMI_KinInterface" variable list.

Variable	Description
stKinHMICtrl_gb	Structure for control of the KinematicInterface
stKinHMIStatus_gb	Structure for status output of the KinematicInterface
uiKinHMINumber_gb	Currently selected kinematic number
arKinHMIActive_gb	Array of all active kinematics
iKinHMIMaxNumber	highest kinematic number available for selection

Tab. 6-18: Structure of "Global_HMI_KinInterface"

The input and output elements of the HMI template are not directly connected to the structures of the KinematicInterface. This function can be realized with the two structures "stKinHMICtrl_gb" and "stKinHMIStatus_gb". These are connected with the KinematicInterface structures by referencing ("arKinCtrl_gb" and "arKinStatus_gb"). If different kinematics are selected via the HMI, the HMI structures are newly referenced to the respective KinematicInterface structures. This enables control of kinematics simultaneously via the PLC as well as the HMI.

The kinematics to be controlled are defined by the "uiKinHMINumber_gb" variable. This variable illustrates the index of the KinematicInterface structures for referencing the HMI structures. For example, for selection of the second kinematics (`uiKinHMINumber_gb = 2`), the second elements of "arKinCtrl_gb" and "arKinStatus_gb" are referenced with "arKinHMICtrl_gb" and "arKinHMIStatus_gb". The values of the variables and the kinematics currently selected are defined directly by switchover via the HMI. However, only active kinematics or numbers of active kinematics can be selected.

Active kinematics are identified via the "arKinHMIActive_gb" array. If a kinematic (number) exists, the elements of this array are assigned the value `True`, otherwise `False`. The maximum size of the array corresponds to the value of the "iKunHMIMaxNumer" variable that corresponds to the maximum kinematic number available for selection.

6.3.2 KinInterface screens

Kinematic status

The "Kinematic status" submenu in the "Production" operating area illustrates the status of the currently selected kinematic. The following is displayed

- the kinematic name
- the position of the TCP in the selected coordinate system
- the velocity of the TCP
- the offset of the selected coordinate system to the machine coordinate system (MCS)
- the type of the selected coordinate system
- the state of the kinematics
- the current override.

Kinematic interface

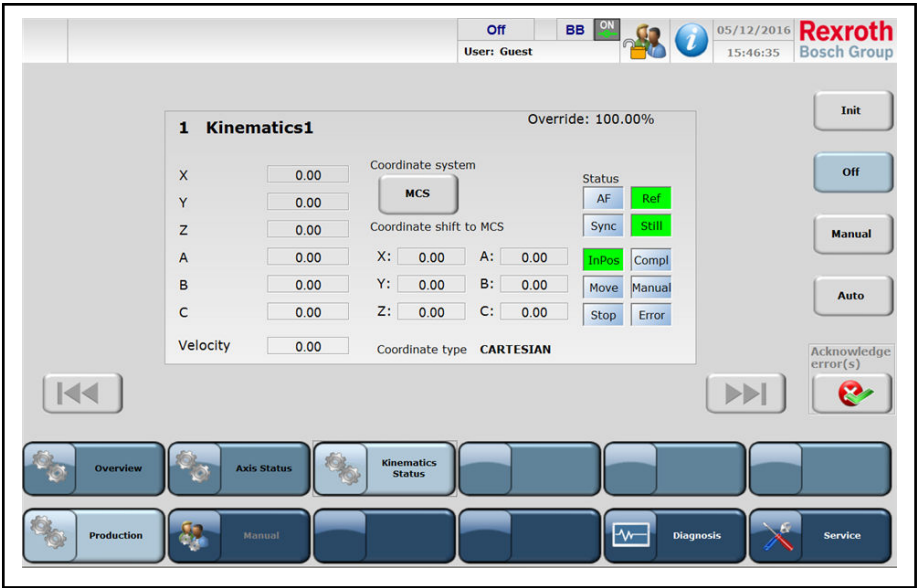


Fig. 6-16: KinematicInterface HMI template - Kinematics status

With the "Coordinate system" button displaying the currently selected coordinate system, the selected coordinate system can be changed. If a Cartesian coordinate system is selected (MCS, BCS, PCSx), the position of the TCP is indicated with "X Y Z A B C". However, if the axis coordinate system (ACS) is selected, the position of the TCP is labeled according to the axis names and the number of the illustrated position values is reduced to the number of axes.

If additional kinematics are configured in the project, switching buttons for further selection are displayed. These buttons can be used to switch to the next kinematics. Kinematic numbers that are not available are skipped. This enables jumping from kinematic number 3 to 5 if two kinematics with the numbers 3 and 5 are configured. If the first or last kinematic is reached, the respective switching button is hidden.

 On switching of kinematics, the currently set value of the override is also applied for following kinematics!

Screen assignment	Name in the template
Header	Header
Main screen	Production_KinStatus
Sidebar	Production_SideBar
Navigation, upper	Production_Navigation
Navigation, lower	Main_Navigation

Tab. 6-19: Template screens used

Kinematic jogging

The "Setup Kinematics" submenu in the "Manual" operating area contains all control elements for manual jogging of kinematics and their axes. This menu can only be accessed in "Manual" operation mode.

The main screen of this submenu is divided into two panes: The top pane is 100% identical to the "Kinematics Status" screen from the "Production" operating area. Via the bottom pane it is possible to change the jog configuration, activate jog mode and reference the kinematic axes.

Kinematic interface

Additionally, the two sidebars are located at the left and right of the main screen with buttons labeled according to the selected coordinate system with "X Y Z A B C" or the axis names of the kinematics axes. This enables jogging of the kinematics if the jog mode is activated and the kinematic referenced.

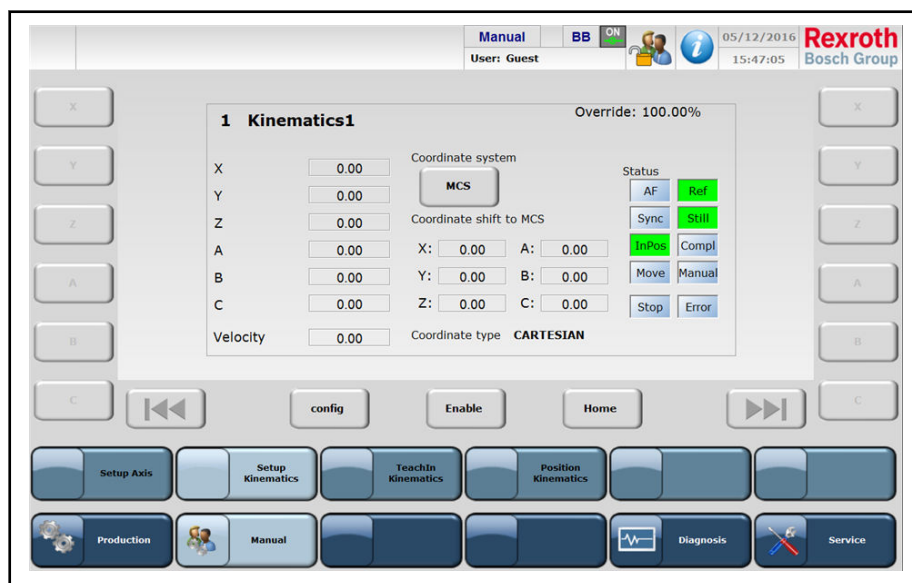


Fig. 6-17: KinematicInterface HMI template - Setup kinematics

Inputs by means of the **config** and **enable** buttons have a direct effect on the `arKinCtrl_gb[index].SetupMode` KinematicInterface structure of the respectively selected kinematic. Additionally, the **Home** button activates the "ModeCoordHoming" operation mode and all kinematic axes are referenced subsequently. The latter is only possible if the jog mode is disabled!



On switching of kinematics, the currently set value of the override is also applied for following kinematics!

Screen assignment	Name in the template
Header	Header
Main screen	Manual_Kinematics_Main Manual_Kin_Jog_SideBar_Bottom
Sidebar	Manual_Kin_Jog_SideBar_Left Manual_Kin_Jog_SideBar_Right
Navigation, upper	Manual_Navigation
Navigation, lower	Main_Navigation

Tab. 6-20: Template screens used

Teaching points

The "TeachIn Kinematics" submenu in the "Manual" operating area contains all control elements for writing, reading and deleting points as well as saving, loading and deleting point files. This menu can only be accessed in "Manual" operation mode.

The main screen of this submenu is divided into two panes: The top pane is 100% identical to the "Kinematics Status" screen from the "Production" operating area. Via the bottom pane it is possible to change the TeachIn configuration and move to teach-in points.

Kinematic interface

Additionally, the two sidebars are located at the left and right of the main screen for triggering of the respective action. The left sidebar contains writing, reading and deleting of points. In the right sidebar, point files can be saved, loaded or deleted.



Fig. 6-18: KinematicInterface HMI template - TeachIn kinematics

With the **config** button, the TeachIn configuration with two functions according to the selected action can be accessed. If a point is to be written/deleted or a point file saved/loaded/deleted, the point, point name, file name, coordinate system and the point and file options can be adjusted here. The point can be entered either manually or using the **Get current position** button.

However, if a (next) point is to be read, this point is illustrated in this menu. By entering the point name or with the **Read** or **ReadNext** button, the point, point name and the coordinate system is updated with the read-in values.

Additionally, the "Move to Point" parameters can be adjusted in TeachIn configuration. With the **Move to Point** button, these are - together with the point - written to the `arKinCtrl_gb[index].CoordMode` KinematicInterface structure and the kinematic is set to the "CoordModePosDirAbs" operation mode. The point is thus moved to by PTP command.



On switching of kinematics, the currently set value of the override is also applied for following kinematics!

Screen assignment	Name in the template
Header	Header
Main screen	Manual_Kinematics_Main Manual_Kin_TeachIn_SideBar_Bottom
Sidebar	Manual_Kin_TeachIn_SideBar_Left Manual_Kin_TeachIn_SideBar_Right
Navigation, upper	Manual_Navigation
Navigation, lower	Main_Navigation

Tab. 6-21: Template screens used

Position kinematics

The "Position Kinematics" submenu in the "Manual" operating area contains all control elements for repositioning of kinematics and their axes. This menu can only be accessed in "Manual" operation mode.

The main screen of this submenu is divided into two panes: The top pane is 100% identical to the "Kinematics Status" screen from the "Production" operating area. Via the bottom pane it is possible to change the positioning configuration.

Additionally, the two sidebars are located at the left and right of the main screen for triggering of the respective action. The left sidebar contains starting and stopping of motion processes as well as deactivation of the kinematic. In the right sidebar, motion processes can be interrupted or continued.



Fig. 6-19: KinematicInterface HMI template - Position kinematics

Inputs by means of the **config** button have a direct effect on the `arKinCtrl_gb[index].CoordMode` KinematicInterface structure of the respectively selected kinematic. With the **Start** button, a motion with these values is executed. Repeated operation with the same interpolation type enables retriggering of the KinematicInterface.

Already started motions may be stopped, interrupted or continued with the **Stop**, **Interrupt** and **Continue** buttons. The kinematic is set to the respective **ModeCoordStopping**, **ModeCoordInterrupted** or **ModeCoordContinue** operation mode.



On switching of kinematics, the currently set value of the override is also applied for following kinematics!

Screen assignment	Name in the template
Header	Header
Main screen	Manual_Kinematics_Main Manual_Kin_Jog_SideBar_Bottom
Sidebar	Manual_Kin_Jog_SideBar_Left Manual_Kin_Jog_SideBar_Right

Kinematic interface

Navigation, upper	Manual_Navigation
Navigation, lower	Main_Navigation

Tab. 6-22: Template screens used

7 ML_KinTech.library

7.1 Introduction and overview



The library requires the ROB (robotics) function package of the control.

Program organization units of ML_KinTech

Area of application (folder in library)	
POUs	Description
Global constants (KinInterface/Const)	
Global_MB_KinInterfaceVars, page 353	Contains information about the axes assigned to the kinematics Contains information about the Robot Control version of the kinematics Facilitates simultaneous access to the AxisInterface from the kinematic interface as well as from a user application
KinInterface/function block	
ML_KinematicsInitType01, page 355	Is used to initialize the KinematicInterface for each kinematics. The function block must be called only once when the program is started or each time the phase is switched from parameterization mode to the operating mode
ML_KinematicsInterfaceType01, page 358	Is used to configure the kinematic interface for each kinematics. The function block has to be called cyclically (in the motion cycle or slower than the motion cycle) as long as the user is in the operating mode
ML_KinematicsInterfaceType11, page 361	Is used to configure the kinematic interface for each kinematics. The functional scope is equivalent to ML_KinematicsInterfaceType01 , whereby this function block works event-controlled and thus performance-optimized. The function block has to be called cyclically (in the motion cycle or slower than the motion cycle) as long as the user is in the operating mode
	Kinematic interface operating modes, overview, page 365 Folder Kin_Interface/Type/Status, overview, page 380 Folder Kin_Interface/Type/Control, overview, page 368
KinJog/FB	
ML_KinJog, page 386	Is used to jog individual axes or axis groups of a kinematics with a Robot Control V2 Function block uses ML_JOGMODE, page 389

Tab. 7-1: ML_KinTech library

7.2 KinInterface/Const

7.2.1 Global_MB_KinInterfaceVars

The ML_KinTech library contains the global variable list **Global_MB_KinInterfaceVars**.

ML_KinTech.library

VAR_GLOBAL Global_MB_KinInterfaceVars

Name	Data type	Comment
arK8ConfigList_gb	ARRAY [1..MB_MAX_KIN_NUMBER] OF Kin_Axis_Config_Elements	Axis configuration list, parameter K-0-0008
arpAdminAxisCtrl_gb	ARRAY [1..MB_MAX_AXIS_NUMBER] OF POINTER TO MB_AXIS_ADMINISTRATION	Pointer to the AxisInterface structure arAxisCtrl_gb[].Admin
arpCoordModeAxisCtrl_gb	ARRAY [1..MB_MAX_AXIS_NUMBER] OF POINTER TO MB_AXIS_COORDINATED	Pointer to the AxisInterface structure arAxisCtrl_gb[].CoordMode
arpAdminAxisStatus_gb	ARRAY [1..MB_MAX_AXIS_NUMBER] OF POINTER TO MB_AXIS_ADMIN_STATUS	Pointer to the AxisInterface structure arAxisStatus_gb[].Admin
arpDiagAxisStatus_gb	ARRAY [1..MB_MAX_AXIS_NUMBER] OF POINTER TO MB_AXIS_DIAGNOSIS	Pointer to the AxisInterface structure arAxisStatus_gb[].Diag

The library defines the following constants:

1. **arK8ConfigList_gb** ([Kin_Axis_Config_Elements](#), page 363):

This array constant contains information about the axes assigned to the kinematics (K-0-0008):

- **uiNumberOfKinAxis:** Total number of axes
- **uiNumberOfMoveAxis:** Number of positioning and orientation axes
- **uiNumberOfBeltAxis:** Number of belt axes
- **arAxesConfig:** Axis configuration (axis type, control number, axis number)

2. **arpAxisCtrlAdmin_gb:**

This pointer constant points to the AdminCtrl structure of the AxisInterface (MB_AXIS_ADMINISTRATION). It facilitates parallel access to the AxisInterface from the KinematicInterface as well as from a user application.

3. **arpAxisCtrlCoordMode_gb:**

This pointer constant points to the CoordModeCtrl structure of the AxisInterface (MB_AXIS_COORDINATED). It facilitates parallel access to the AxisInterface from the KinematicInterface as well as from a user application.

4. **arpAxisStatusAdmin_gb:**

This pointer constant points to the AdminStatus structure of the AxisInterface (MB_AXIS_ADMIN_STATUS). It facilitates parallel access to the AxisInterface from the KinematicInterface as well as from a user application.

5. **arpAxisStatusDiag_gb:**

This pointer constant points to the DiagStatus structure of the AxisInterface (MB_AXIS_DIAGNOSIS). It facilitates parallel access to the AxisInterface from the KinematicInterface as well as from a user application.

7.3 KinInterface/FB

7.3.1 ML_KinematicsInitType01

Brief description The **ML_KinematicsInitType01** function block is used to initialize the KinematicInterface for each kinematics.

The function block must be called only once when the program is started or each time the phase is switched from parameterization mode to the operating mode.

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/FB

Tab. 7-2: ML_KinematicsInitType01 reference table

Interface description

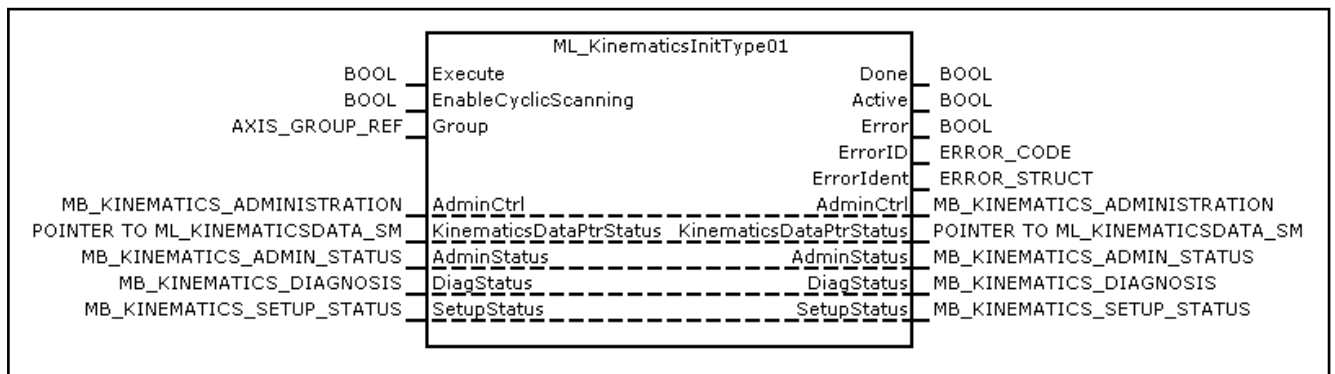


Fig. 7-1: ML_KinematicsInitType01 function block

I/O type	Name	Data type	Comment
VAR_INPUT	Execute	BOOL	Start of initialization using a positive edge
	EnableCyclicScanning	BOOL	Cyclic check and transmission of changed entries for the current operation mode
	Group	AXIS_GROUP_REF	Kinematics to be initialized in the KinematicInterface
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 error information Troubleshooting, page 358

ML_KinTech.library

I/O type	Name	Data type	Comment
VAR_IN_OUT	AdminCtrl	MB_KINEMATICS_ADMINISTRATION	Axis management arKinCtrl_gb[].Admin MB_KINEMATICS_ADMINISTRATION, page 369
	KinematicsDataPtrStatus	POINTER TO ML_KINEMATICS_DATA_SM	Pointer to KinData [] of this kinematics
	AdminStatus	MB_KINEMATICS_ADMIN_STATUS	Status management of the kinematics arKinStatus_gb[].Admin MB_KINEMATICS_ADMIN_STATUS, page 381
	DiagStatus	MB_KINEMATICS_DIAGNOSIS	Diagnostic information on the kinematics arKinStatus_gb[].Diag MB_KINEMATICS_DIAGNOSIS, page 385
	SetupStatus	MB_KINEMATICS_SETUP_STATUS	Status of jog/setup function of the kinematics arKinStatus_gb[].SetupMode MB_KINEMATICS_SETUP_STATUS, page 386

Tab. 7-3: Interface variables of ML_KinematicsInitType01



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 special extensions could be added to the existing code in the user application.

This is why the required elements of [MB_KINEMATICS_CONTROL_TYPE01, page 369](#), and [MB_KINEMATICS_STATUS_TYPE01, page 380](#), are transferred as separate inputs.

For more details on this, refer to "KinematicInterface User Extension".

Rexroth IndraLogic XLC IndraMotion MLC 14VRS, Generic Application Template, DOK-XLCMLC-TF*GATV14-APRS-EN-P, R911341487**

or

Rexroth IndraMotion MLC 14VRS Robot Control V2, DOK-MLC*-ROCO****V14-RERS-EN-P, R911341588**

Functional Description

The ML_KinematicsInitType01 function block initializes the following structure elements with default values:

Administration control	
AdminCtrl._OpMode.en	ModeCoordAB
AdminCtrl.ClearError	FALSE
AdminCtrl.EnableCyclicScanning	EnableCyclicScanning
Administration status	

AdminStatus._OpModeAck	ModeCoordUngrouped
AdminStatus.Active	FALSE
AdminStatus.Name	''
AdminStatus.TrafoNb	999
AdminStatus.CmdDone	FALSE
AdminStatus.InBuffer	FALSE
AdminStatus.ParamLevelAck	FALSE
AdminStatus.ConveyorBelt[].BeginLength	0.0
AdminStatus.ConveyorBelt[].TotalLength	0.0
AdminStatus.ConveyorBelt[].ConveyorBeltOrigin[]	0.0
AdminStatus.ConveyorBelt[].CoordType	CS_TYPE_CARTESIAN
AdminStatus.ConveyorBelt[].ErrorReactionType	ML_BELT_ERROR_DESYNC_WITH_LIMITS
AdminStatus.ConveyorBelt[].ErrorReactionDesyncTime	T#300MS
AdminStatus.ConveyorBelt[].ErrorReactionDeceleration	1000.0
AdminStatus.ConveyorBelt[].ErrorReactionJerk	0.0
AdminStatus.ConveyorBelt[].Done	FALSE
AdminStatus.PCSStatus[].Origin[]	0.0
AdminStatus.PCSStatus[].CoordType	CS_TYPE_CARTESIAN
AdminStatus.PCSStatus[].Done	FALSE
AdminStatus.KinTransformPLCDone	FALSE
Diagnostics	
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'
Setup status	
SetupStatus.EnableAck	FALSE
KinematicsDataPtrStatus	POINTER TO KinData[] of the kinematics

Tab. 7-4: Structure elements initialized by ML_KinematicsInitType01

The **ML_KinematicsInitType01** function block reads the following parameters and writes the values to the following structure elements:

K-parameter (name)	Structural element
K-0-0031 (Transformation scheme)	AdminStatus.TrafoNb
K-0-0002 (Kinematic name)	AdminStatus.Name
K-0-0008 (Axis configuration list)	only required for internal purposes

Tab. 7-5: Parameters read out by *ML_KinematicsInitType01*

EnableCyclicScanning: The "EnableCyclicScanning" input activates the cyclic check and transmission of the changed inputs for the current operation mode. This is relevant for the values in `arKinCtrl_gb[]` with the comment: "Cyclically tested and transferred in case of change.". The default value is TRUE.

Group: The "Group" input specifies for which kinematics the initialization process is going to be carried out for.

Troubleshooting 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 kinematic number "Group.GroupNo"
INPUT_INVALID_ERROR, (16#0001)	16#2600	16#0001	Invalid control number "Group.CntrlNo". This has to be set to LOCAL_CNTRL.

Tab. 7-6: Error codes *ML_KinematicsInitType01*

7.3.2 **ML_KinematicsInterfaceType01**

Brief description The **ML_KinematicsInterfaceType01** function block is used to configure the kinematic interface for each kinematics.

This function block has to be called cyclically (in the motion cycle or slower than the motion cycle) as long as the user is in the operating mode.

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/FB

Tab. 7-7: *ML_KinematicsInitType01* reference table

Interface description

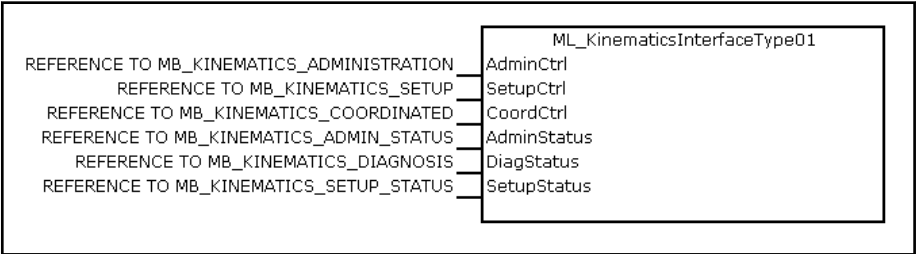


Fig. 7-2: *ML_KinematicsInterfaceType01* function block

I/O type	Name	Data type	Comment
VAR_INPUT	AdminCtrl	REFERENCE TO MB_KINEMATICS_ADMINISTRATION	Axis management arKinCtrl_gb[].Admin MB_KINEMATICS_ADMINISTRATION, page 369
	SetupCtrl	REFERENCE TO MB_KINEMATICS_SETUP	Setup mode arKinCtrl_gb[].SetupMode MB_KINEMATICS_SETUP, page 376
	CoordCtrl	REFERENCE TO MB_KINEMATICS_COORDINATED	Operating mode coordinated motion arKinCtrl_gb[].CoordMode MB_KINEMATICS_COORDINATED, page 377
	AdminStatus	REFERENCE TO MB_KINEMATICS_ADMIN_STATUS	Status management of the axis arKinStatus_gb[].Admin MB_KINEMATICS_ADMIN_STATUS, page 381
	DiagStatus	REFERENCE TO MB_KINEMATICS_DIAGNOSIS	Status diagnostics of the axis arKinStatus_gb[].Diag MB_KINEMATICS_DIAGNOSIS, page 385
	SetupStatus	REFERENCE TO MB_KINEMATICS_SETUP_STATUS	Status jog/set-up mode arKinStatus_gb[].SetupMode MB_KINEMATICS_SETUP_STATUS, page 386

Tab. 7-8: *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 special extensions could be added to the existing code in the user application.

This is why the required elements of [MB_KINEMATICS_CONTROL_TYPE01, page 369](#), and [MB_KINEMATICS_STATUS_TYPE01, page 380](#), are transferred as separate inputs.

For more details on this, refer to "KinematicInterface User Extension".

Rexroth IndraLogic XLC IndraMotion MLC 14VRS, Generic Application Template, DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487

or

Rexroth IndraMotion MLC 14VRS Robot Control V2, DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588

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.

ML_KinTech.library

Troubleshooting 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
RESOURCE_ERROR, (16#0003)	16#2600	16#0001	Invalid control number "Group.CntrlNo". Has to be set to LOCAL_CNTRL
DEVICE_ERROR, (16#0008)	16#2601	16#0000	Kinematics reports an error in KinData structure
INPUT_INVALID_ERROR, (16#0001)	16#2602	Number of OpMode	Invalid operating mode selected
INPUT_INVALID_ERROR, (16#0001)	16#2603	Number of OpMode	Invalid, unsupported operating mode activated
INPUT_RANGE_ERROR, (16#0006)	16#2604	16#0000	At least one input reference (AdminCtrl, AdminStatus, etc.) is not set
ACCESS_ERROR, (16#0004)	16#2605	16#0000	Referencing of the axis/axes not possible and not required
ACCESS_ERROR, (16#0004)	16#2605	Number of the axis to be referenced	Referencing of axis/axes not possible
RESOURCE_ERROR, (16#0003)	16#2606	16#0000	Operating mode has not been changed during a SetupModes
OTHER_ERROR, (16#7FFE)	16#2607	16#0000	Kinematics could not be grouped

Tab. 7-9: Error codes ML_KinematicsInterfaceType01

Additionally, the errors of the following internally called function blocks can be transferred:

MB_GroupReadParameter	ML_GroupReadNextPoint
MB_GroupReadStringParameter	ML_GroupReadPoint
MC_GroupContinue	ML_GroupSavePointFile
MC_GroupInterrupt	ML_GroupWritePoint
MC_GroupReset	ML_KinJog
MC_GroupStop	ML_SetBeltConfiguration
MC_MoveLinearAbsolute	ML_SetBeltErrorReaction
MC_MoveLinearRelative	ML_SetCylindricTransform
MC_MoveDirectAbsolute	ML_RobotGetBeltConfiguration
MC_MoveDirectRelative	ML_RobotGetBeltErrorReaction
MC_MoveCircularAbsolute	ML_RobotGetTrafoValues
MC_MoveCircularRelative	ML_RobotMoveJumpAbsolute

MC_SetCartesianTransform	ML_RobotOpenCyclicChannel
MC_SetKinTransform	ML_RobotOpenDirectCyclicChannel
ML_GroupDeletePointFile	ML_RobotWriteCyclicChannel
ML_GroupDeletePoints	TP
ML_GroupLoadPointFile	TON

Tab. 7-10: Sources of transferred error codes ML_KinematicsInterfaceType01

7.3.3 ML_KinematicsInterfaceType11

Brief description The **ML_KinematicsInterfaceType11** function block is used to configure the kinematic interface for each kinematics. The functional scope is equivalent to ML_KinematicsInterfaceType01, whereby this function block works event-controlled and thus performance-optimized.

This function block has to be called cyclically (in the motion cycle or slower than the motion cycle) as long as the user is in the operating mode.

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/FB

Tab. 7-11: ML_KinematicsInitType11 reference table

Interface description

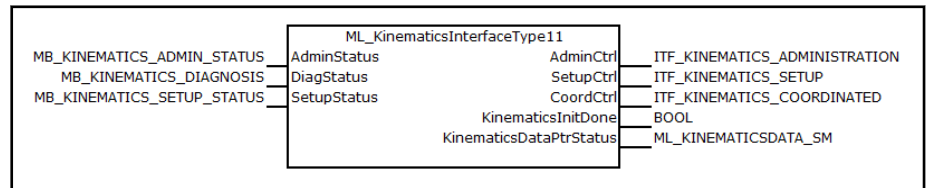


Fig. 7-3: ML_KinematicsInterfaceType11 function block

I/O type	Name	Data type	Comment
VAR_INPUT	AdminStatus	REFERENCE TO MB_KINEMATICS_ADMIN_STATUS	Status management of the axis arKinStatus_gb[].Admin MB_KINEMATICS_ADMIN_STATUS, page 381
	DiagStatus	REFERENCE TO MB_KINEMATICS_DIAGNOSIS	Status diagnostics of the axis arKinStatus_gb[].Diag MB_KINEMATICS_DIAGNOSIS, page 385
	SetupStatus	REFERENCE TO MB_KINEMATICS_SETUP_STATUS	Status jog/set-up mode arKinStatus_gb[].SetupMode MB_KINEMATICS_SETUP_STATUS, page 386

ML_KinTech.library

I/O type	Name	Data type	Comment
VAR_OUT UT	AdminCtrl	ITF_KINEMATICS_ADMINISTRATION	Axis management arKinCtrl_gb[].Admin Corresponds to type MB_KINEMATICS_ADMINISTRATION , page 369 with regard to content
	SetupCtrl	ITF_KINEMATICS_SETUP	Setup mode arKinCtrl_gb[].SetupMode Corresponds to type MB_KINEMATICS_SETUP , page 376 with regard to content
	CoordCtrl	ITF_KINEMATICS_COORDINATED	Operating mode coordinated motion arKinCtrl_gb[].CoordMode Corresponds to type MB_KINEMATICS_COORDINATED , page 377 with regard to content
	KinematicsInitDone	BOOL	Initialization of the kinematics is completed
	KinematicsDataPtrStatus	POINTER TO ML_KINEMATICS_DATA_SM	Pointer to KinData interface of kinematics (ML_Robot.library) - Input .KinematicsDataPtrStatus^. for drop-down menu An instance of the structure ML_KINEMATICS_DATA_SM, ML_Base contains information on the corresponding kinematics that can be accessed via shared memory

Tab. 7-12: ML_KinematicsInterfaceType11 interface variables



The function block uses so-called interfaces for specifying the operating modes and command values.

To optimize performance, the status structures are connected to REFERENCE TO instead of as VAR_IN_OUT as VAR_INPUT. Thus, a one-time initialization of the inputs is sufficient and the structures do not have to be transferred upon the cyclic function block call.

Troubleshooting

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
RESOURCE_ERROR, (16#0003)	16#2600	16#0001	Invalid control number "Group.CntrlNo". Has to be set to LOCAL_CNTRL
DEVICE_ERROR, (16#0008)	16#2601	16#0000	Kinematics reports an error in KinData structure
INPUT_INVALID_ERROR, (16#0001)	16#2602	Number of OpMode	Invalid operating mode selected

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR, (16#0001)	16#2603	Number of OpMode	Invalid, unsupported operating mode activated
INPUT_RANGE_ERROR, (16#0006)	16#2604	16#0000	At least one input reference (AdminCtrl, AdminStatus, etc.) is not set
ACCESS_ERROR, (16#0004)	16#2605	16#0000	Referencing of the axis/axes not possible and not required
ACCESS_ERROR, (16#0004)	16#2605	Number of the axis to be referenced	Referencing of axis/axes not possible
RESOURCE_ERROR, (16#0003)	16#2606	16#0000	Operating mode has not been changed during a SetupModes
OTHER_ERROR, (16#7FFE)	16#2607	16#0000	Kinematics could not be grouped

Tab. 7-13: Error codes ML_KinematicsInterfaceType11

Additionally, the errors of the following internally called function blocks can be transferred:

MB_GroupReadParameter	ML_GroupReadNextPoint
MB_GroupReadStringParameter	ML_GroupReadPoint
MC_GroupContinue	ML_GroupSavePointFile
MC_GroupInterrupt	ML_GroupWritePoint
MC_GroupReset	ML_KinJog
MC_GroupStop	ML_SetBeltConfiguration
MC_MoveLinearAbsolute	ML_SetBeltErrorReaction
MC_MoveLinearRelative	ML_SetCylindricTransform
MC_MoveDirectAbsolute	ML_RobotGetBeltConfiguration
MC_MoveDirectRelative	ML_RobotGetBeltErrorReaction
MC_MoveCircularAbsolute	ML_RobotGetTrafoValues
MC_MoveCircularRelative	ML_RobotMoveJumpAbsolute
MC_SetCartesianTransform	ML_RobotOpenCyclicChannel
MC_SetKinTransform	ML_RobotOpenDirectCyclicChannel
ML_GroupDeletePointFile	ML_RobotWriteCyclicChannel
ML_GroupDeletePoints	TP
ML_GroupLoadPointFile	TON

Tab. 7-14: Sources of transferred error codes ML_KinematicsInterfaceType11

7.3.4 KinInterface/FB/DataTypes

Kin_Axis_Config_Elements

This structure constant contains information about the axes assigned to the kinematics (K-0-0008):

ML_KinTech.library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/FB/DataTypes

Tab. 7-15: Reference table Kin_Axis_Config_Elements

STRUCT Kin_Axis_Config_Elements

Name	Data type	Initial value	Comment
uiNumberOfKinAxes	UINT	0	Number of axes assigned to the kinematics
uiNumberOfMoveAxes	UINT	0	Number of positioning and orientation axes assigned to the kinematics
uiNumberOfBeltAxes	UINT	0	Number of belt axes assigned to the kinematics
arAxesConfig	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF ML_KIN_AXES_CONFIG_DATA		Axis configuration (axis type, control number, axis number)

This structure uses

- ML_Robot.library: ML_KIN_AXES_CONFIG_DATA

Kin_Axis_Type

The enumeration type names the possible axis types: (No axis, positioning axis, orientation axis, belt axis)

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/FB/DataTypes

Tab. 7-16: Kin_Axis_Type reference table

ENUM Kin_Axis_Type_

Name	Data type	Initial value	Comment
NoAxisCfg	UINT	0	No axis
PositionAxisCfg	UINT	8	Positioning axis
OrientationAxisCfg	UINT	16	Orientation axis
BeltAxisCfg	UINT	32	Belt axis

7.4 KinInterface/Type**7.4.1 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.**

7.4.2 Kinematic interface operating modes, overview

There are two methods to activate an operating 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

Set a bit using the bit access functionality.

```
arKinCtrl_gb[].Admin._OpMode.b.MODE_COORD_AH:= TRUE;
```



Only one bit can be set in `arKinCtrl_gb[].Admin._OpMode`. Setting several bits causes an error in the KinematicInterface.

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) that are used or that have default values that were changed need to be defined before the current operation mode switch is executed.



The KinematicInterface uses the AxisInterface internally in order to carry out several switchovers (e.g. grouping / ungrouping all axes, homing of axes) .

Also refer to

- [MB_KINEMATICS_MODE_UNION, page 365](#)
- [MB_KINEMATICS_MODE_BITS, page 366](#)
- [MB_KINEMATICS_MODE, page 367](#)

7.4.3 MB_KINEMATICS_MODE_UNION

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type

Tab. 7-17: MB_KINEMATICS_MODE_UNION reference table

Brief description **MB_AXIS_Mode_UNION** is a union data type (UNION) used by the **_OpMode** element contained in the **MB_KINEMATICS_ADMINISTRATION** structure.

The control of **_OpMode** is possible using both data types as they are contained in the same storage location.

UNION MB_KINEMATICS_MODE_UNION

Name	Data type	Comment
en	MB_KINEMATICS_MODE	Operating modes as enumeration of type MB_KINEMATICS_MODE , page 367,
b	MB_KINEMATICS_MODE_BITS	Operating modes as bit structure of type MB_KINEMATICS_MODE_BITS , page 366

- en
- Access to the operating modes via an enumeration value
- b
- Access to the operating mode via a bit structure



Fig. 7-4: MB_KINEMATICS_MODE_UNION overview

7.4.4 MB_KINEMATICS_MODE_BITS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type

Tab. 7-18: MB_KINEMATICS_MODE_BITS reference table

Brief description The **MB_KINEMATICS_MODE_BITS** structure contains the bit structure of the **_OpMode** element of the **MB_KINEMATICS_ADMINISTRATION**, page 369, and for the structure **MB_KINEMATICS_ADMIN_STATUS**, page 381.

STRUCT MB_KINEMATICS_MODE_BITS

Name	Data type	Initial value	Comment
MODE_COORD_AB	BIT	FALSE	Kinematics ready, drive ready, OpMode(Ack).0
MODE_COORD_AH	BIT	FALSE	Kinematics ready, drive ready, OpMode(Ack).1

Name	Data type	Initial value	Comment
MODE_COORD_HOMING	BIT	FALSE	Referencing, AH, AxisInterface -> SetupMode, OpMode(Ack).2
MODE_COORD_POS_DIR_ABS	BIT	FALSE	Absolute PTP positioning, AF, MC_MoveDirectAbsolute, OpMode(Ack).3
MODE_COORD_POS_DIR_REL	BIT	FALSE	Relative PTP positioning, AF, MC_MoveDirectRelative, OpMode(Ack).4
MODE_COORD_POS_LIN_ABS	BIT	FALSE	Absolute linear positioning, AF, MC_MoveLinearAbsolute, OpMode(Ack).5
MODE_COORD_POS_LIN_REL	BIT	FALSE	Relative linear positioning, AF, MC_MoveLinearRelative, OpMode(Ack).6
MODE_COORD_POS_CIRC_ABS	BIT	FALSE	Absolute circular positioning, AF, MC_MoveCircularAbsolute, OpMode(Ack).7
MODE_COORD_POS_CIRC_REL	BIT	FALSE	Relative circular positioning, AF, MC_MoveCircularRelative, OpMode(Ack).8
MODE_COORD_POS_JUMP_ABS	BIT	FALSE	Absolute jump positioning, AF, ML_RobotMoveJumpAbsolute, OpMode(Ack).9
MODE_COORD_STOPPING	BIT	FALSE	Kinematic stop, AH, MC_GroupStop, OpMode(Ack).10
MODE_COORD_INTERRUPTED	BIT	FALSE	Operating mode "Interrupted", AF, MC_GroupInterrupt, OpMode(Ack).11
MODE_COORD_CONTINUE	BIT	FALSE	Operating mode "Continue", AF, MC_GroupContinue, OpMode(Ack).12
MODE_COORD_EXTERNAL_FB	BIT	FALSE	Operating mode "External function block", power on subsequently , the external function block takes over the control, OpMode(Ack).13
MODE_COORD_CYCLIC	BIT	FALSE	Cyclic position command value specification, AF, ML_RobotWriteCyclicChannel, OpMode(Ack).14

7.4.5 MB_KINEMATICS_MODE

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type

Tab. 7-19: MB_KINEMATICS_MODE reference table

Brief description

The MB_KINEMATICS_MODE (ENUM) structure contains the constants of the element **_OpMode** of the structure [MB_KINEMATICS_ADMINISTRATION](#), [page 369](#), and the element **_OpModeAck** of the structure [MB_KINEMATICS_ADMIN_STATUS](#), [page 381](#). The enumeration type is used by the [MB_KINEMATICS_MODE_UNION](#) structures, [page 365](#), and [MB_KINEMATICS_ADMIN_STATUS](#).

ML_KinTech.library

STRUCT MB_KINEMATICS_MODE

Name	Data type	Initial value	Comment
ModeCoordUngrouped	UINT	16#0	All axes in AB or AH/AF
ModeCoordAB	UINT	16#1	AB, all axes without torque
ModeCoordAH	UINT	16#2	AH, all axes with torque, final state "standstill" (AB->AH or AF->AH)
ModeCoordHoming	UINT	16#4	Referencing, AH, AxisInterface -> SetupMode
ModeCoordPosDirAbs	UINT	16#8	Absolute PTP positioning, AF, MC_MoveCircularAbsolute
ModeCoordPosDirRel	UINT	16#10	Relative PTP positioning, AF, MC_MoveDirectRelative
ModeCoordPosLinAbs	UINT	16#20	Absolute linear positioning, AF, MC_MoveCircularAbsolute
ModeCoordPosLinRel	UINT	16#40	Relative linear positioning, AF, MC_MoveLinearRelative
ModeCoordPosCircAbs	UINT	16#80	Absolute circular positioning, AF, MC_MoveCircularAbsolute
ModeCoordPosCircRel	UINT	16#100	Relative circular positioning, AF, MC_MoveCircularRelative
ModeCoordPosJumpAbs	UINT	16#200	Absolute jump positioning, AF, ML_RobotMoveJumpAbsolute
ModeCoordStopping	UINT	16#400	Kinematic stop, AH, MC_GroupStop
ModeCoordInterrupted	UINT	16#800	Operating mode "Interrupted", AF, MC_GroupInterrupt
ModeCoordContinue	UINT	16#1000	Operating mode "Continue", AF, MC_GroupContinue
ModeCoordExternalFB	UINT	16#2000	Operating mode "External function block", power on and then , the external function block takes over the control
ModeCoordCyclic	UINT	16#4000	Cyclic position command value specification, AF, ML_RobotWriteCyclicChannel

7.4.6 KinInterface/Type/Control

Folder Kin_Interface/Type/Control, overview

MB_KINEMATICS_CONTROL_TYPE01 on page 369			
	MB_KINEMATICS_ADMINISTRATION on page 369		
		MB_KINEMATICS_MODE_UNION on page 365	
		MB_KINEMATICS_CONVEYOR_BELT on page 372	
		MB_KINEMATICS_TEACH_IN on page 373	
			MB_KINEMATICS_TEACH_IN_MODE on page 374
		MB_KINEMATICS_ADMIN_CHANGE_OPMODE on page 375	
		MB_KINEMATICS_PCS_CONFIG on page 375	
	MB_KINEMATICS_SETUP on page 376		
	MB_KINEMATICS_COORDINATED on page 377		

Tab. 7-20: MB_KINEMATICS_CONTROL_TYPE01 structure

MB_KINEMATICS_CONTROL_TYPE01

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-21: MB_KINEMATICS_CONTROL_TYPE01 reference table

Brief description The MB_KINEMATICS_CONTROL_TYPE01 data structure is accessed via the arKinCtrl_gb[] instance.

STRUCT MB_KINEMATICS_CONTROL_TYPE01

Name	Data type	Comment
Admin	MB_KINEMATICS_ADMINISTRATION	Kinematic management
SetupMode	MB_KINEMATICS_SETUP	Jog/setup functions
CoordMode	MB_KINEMATICS_COORDINATED	Coordination operating mode (linear, circular, PTP; absolute, relative)

MB_KINEMATICS_ADMINISTRATION

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-22: MB_KINEMATICS_ADMINISTRATION reference table

Brief description The MB_KINEMATICS_ADMINISTRATION structure is a substructure of [MB_KINEMATICS_CONTROL_TYPE01](#), [page 369](#), and contains control variables and command values for kinematic administration.

STRUCT MB_KINEMATICS_ADMINISTRATION

Name	Data type	Initial value	Comment
_OpMode	MB_KINEMATICS_MODE_UNION	STRUCT (en := ModeCoordAb)	State-controlled Write the input data of the operating mode before selecting the operating mode - Operating mode selection (ENUMs): e.g. ..._OpMode.en := ModeCoordAB - Operating mode selection (BIT variables): e.g. ..._OpMode.b.MODE_COORD_AB := TRUE Note: Always only set 1 input!
Group	AXIS_GROUP_REF	STRUCT (CntrlNo := RIL_CommonTypes. LOCAL_CNTRL, GroupNo := NO_KINEMATICS)	Kinematic reference
KinTransformPLC	MC_KIN_REF	0	Pointer to transformation interface of the kinematics

ML_KinTech.library

Name	Data type	Initial value	Comment
DiagNbrRefreshTime	TIME	TIME#200ms	Refresh rate of arKinStatus[].Diag.Number - Switched off if zero - Reads drive parameters, if $\geq T\#0\text{ms}$
StopDeceleration	REAL	999.0	Deceleration Positive value, (unit: e.g. in $[\text{rad/s}^2]$ according to parameterization)
StopJerk	REAL	9999.0	Deceleration jerk Positive value, (unit: e.g. in $[\text{rad/s}^3]$ according to parameterization)
ClearError	BOOL	FALSE	Edge-controlled Deletes kinematic errors with MC_GroupReset()
VelocityOverride	REAL	100	Feed correction Is written to parameter K-0-0240
AccelerationOverride	REAL	100	Acceleration correction Is written to parameter K-0-0242
DecelerationOverride	REAL	100	Deceleration correction Is written to parameter K-0-0243
EnableCyclicScanning	BOOL	TRUE	State-controlled Activates the temporary scanning of input data in the current operating mode Affects variables in arKinCtrl_gb with note "Note: is permanently cyclically processed"
ParamLevel	BOOL	FALSE	Switches the axes to parameterization mode 1, <i>Parameter A-0-0024 bit 3</i> - TRUE: enables parameterization mode - FALSE: disables parameterization mode
RetriggerOpMode	BOOL	FALSE	Edge-controlled Restart of the same operating mode, e.g. ModeCoordPosLinAbs with same values
ConveyorBelt	ARRAY [1..MB_MAX_KIN_ AXIS_NUMBER] OF MB_KINEMATICS_ CONVEYOR_BELT		Configuration of a transport belt
TeachIn	MB_KINEMATICS_TEACH_IN		Teaching/saving the points MB_KINEMATICS_TEACH_IN on page 373

Name	Data type	Initial value	Comment
ChangeOpMode	MB_KINEMATICS_ADMIN_CHANGE_OPMODE	NoAction	Behavior after an operating mode change MB_KINEMATICS_ADMIN_CHANGE_OPMODE on page 375
PCSConfig	ARRAY [1..ML_MAX_KIN_PCS_NUMBER] OF MB_KINEMATICS_PCS_CONFIG		Configures of a product coordinate system MB_KINEMATICS_PCS_CONFIG on page 375

OpMode Operating mode selection

For more information, refer to [Kinematic interface operating modes, overview, page 365](#).



Setting several bits in `arKinCtrl_gb[].Admin._OpMode.b` causes an error.

KinTransformPLC

By using this element, user-defined transformations can be registered. If a transformation function block is created, it is registered in the PLC.



The registration of a user-defined transformation is only possible in parameterization mode **P2**. When trying to register a transformation in operation mode **BB**, there is no result.



If the kinematic interface is started for the first time with a user-defined transformation, the control reports the error "Transformation not registered". This is due to the fact that the IMC interface tries to switch to operation mode. After successful registration of the transformation, errors can be acknowledged.

EnableCyclicScanning

This state-controlled element controls cyclic reading of `arKinCtrl_gb[].CoordMode.Point`. By setting the element, a new motion is started upon each new position and thus a new block is loaded to the block buffer.

ParamLevel

With the edge-controlled element, all axes of the kinematics can be set to parameterization mode. By resetting the element, the axes are reset to its default state.



If the parameterization mode is active, the project tree displays "PM" for the respective axes.

RetriggerOpMode

This edge-controlled element causes a restart of the currently used operating mode. If the element is set, the kinematic interface internally switches to "ModeCoordExternalFB" and subsequently back to the original operating mode. Thus, a multiple relative traversing of the kinematics is possible without changing the position values.

ConveyorBelt

Access to attributes to configure belts ([MB_KINEMATICS_CONVEYOR_BELT, page 372](#)).



A maximum of 16 belts can be configured per kinematics.

PCSConfig

Access to attributes to configure product coordinate systems ([MB_KINEMATICS_PCS_CONFIG, page 375](#)).

ML_KinTech.library



A maximum of 16 product coordinate systems can be configured per kinematics.

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

MB_KINEMATICS_CONVEYOR_BELT

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-23: MB_KINEMATICS_CONVEYOR_BELT reference table

Brief description The MB_KINEMATICS_CONVEYOR_BELT structure is used to configure a belt.

The elements contained in this structure are equivalent to the function block inputs (ML_Robot.library)

- ML_SetBeltConfiguration
- ML_SetBeltErrorReaction

Please refer to the relevant documents for further information.

STRUCT MB_KINEMATICS_CONVEYOR_BELT

Name	Data type	Initial value	Comment
Execute	BOOL	FALSE	Edge-controlled Executing the belt configuration
ConveyorBeltOrigin	ARRAY [1..6] OF REAL	[6(0.0)]	Defines the belt position and rotation in relation to the WCS
TotalLength	REAL	0.0	Defines the length of the belt-synchronous travel ranges from the starting position
BeginLength	REAL	0.0	Defines the length of the late start of a belt-synchronous motion from the starting position
Axis	AXIS_REF		Reference of the belt axis
ErrorReactionType	ML_BELT_ERROR_REACTION_MODE	ML_BELT_ERROR_NO_SPECIFIC_REACTION	<i>Defines the error reaction type</i> • ML_BELT_ERROR_NO_SPECIFIC_REACTION • ML_BELT_ERROR_DESYNC_OVER_TIME • ML_BELT_ERROR_DESYNC_WITH_LIMITS
ErrorReactionDesyncTime	TIME	TIME#0ms	Defines the time for desynchronization
ErrorReactionDeceleration	REAL	0.0	Defines the deceleration rate for desynchronization
ErrorReactionJerk	REAL	0.0	Defines the deceleration jerk for desynchronization

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

MB_KINEMATICS_TEACH_IN

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-24: MB_KINEMATICS_TEACH_IN reference table

Brief description

The MB_KINEMATICS_TEACH_IN structure is used for

- Writing a single point (ML_GroupWritePoint, ML_Robot)
- Reading a single point (ML_GroupReadPoint, ML_Robot)
- Reading the next point (ML_GroupReadNextPoint, ML_Robot)
- Deleting one or several points (ML_GroupDeletePoints, ML_Robot)
- Saving a point file (ML_GroupSavePointFile, ML_Robot)
- Loading a point file (ML_GroupLoadPointFile, ML_Robot)
- Deleting a point file (ML_GroupDeletePointFile, ML_Robot)

Please refer to the documentation for the relevant function block for further information.

See Rexroth IndraMotion MLC 14VRS RobotControl V2, DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588.

STRUCT MB_KINEMATICS_TEACH_IN

Name	Data type	Initial value	Comment
Execute	BOOL	FALSE	Edge-controlled Execution of processing
Mode	MB_KINEMATICS_TEACH_IN_MODE	WritePoint	Type of processing of a point or a point file MB_KINEMATICS_TEACH_IN_MODE , page 374
FileName	STRING	"	<i>File name</i> • Input of the ML_GroupSavePointFile function block • Input of the ML_GroupLoadPointFile function block • Input of the ML_GroupDeletePointFile function block
PointName	STRING(19)	"	<i>Name of the point</i> • Input of the ML_GroupWritePoint function block • Input of the ML_GroupReadPoint function block • Input of the ML_GroupReadnextPoint function block • Input of the ML_GroupDeletePoints function block
Point	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(0.0)]	<i>Point array</i> • Input of the ML_GroupWritePoint function block

ML_KinTech.library

Name	Data type	Initial value	Comment
CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	<i>Coordinate system of the point (ML_Robot.library)</i> - Input of the ML_GroupWritePoint function block <i>Possible coordinate systems</i> - CS_MACHINE_COORDINATES - CS_AXES_COORDINATES - CS_BASE_COORDINATES - CS_PRODUCT_COORDINATES_n (n = 1..16)
Option	E_KIN_TEACH_IN_POINT_FILE_OPTIONS	E_GROUP_TEACH_IN_NEW	<i>Option to load and save point files</i> - Input of the ML_GroupSavePointFile function block - Input of the ML_GroupLoadPointFile function block <i>Possible options (ML_Robot.library)</i> - E_GROUP_TEACH_IN_NEW - E_GROUP_TEACH_IN_APPEND - E_GROUP_TEACH_IN_APPEND_OVERWRITE
AllowOverwrite	BOOL	FALSE	<i>Allowing overwriting of existing points</i> Input of the ML_GroupWritePoint function block
SaveBeltValues	BOOL	FALSE	<i>Saving the belt values of the point</i> Input of the ML_GroupWritePoint function block

- Folder [Kin_Interface/Type/Control](#), overview on page 368

MB_KINEMATICS_TEACH_IN_MODE

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-25: MB_KINEMATICS_TEACH_IN_MODE reference table

The enumeration type specifies the possible access types of the "TEACH_IN" operating mode.

ENUM MB_KINEMATICS_TEACH_IN_MODE

Name	Data type	Initial value	Comment
WritePoint	UINT	0	Writing a point
ReadPoint	UINT	1	Reading a point
ReadNextPoint	UINT	2	Reading a next point
DeletePoints	UINT	4	Deleting a point

Name	Data type	Initial value	Comment
SavePointFile	UINT	8	Saving a point file
LoadPointFile	UINT	10	Loading of a point file
DeletePointFile	UINT	12	Deleting a point file

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

MB_KINEMATICS_ADMIN_CHANGE_OPMODE

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-26: MB_KINEMATICS_ADMIN_CHANGE_OPMODE reference table

The enumeration type describes the behavior after an operating mode change.

ENUM MB_KINEMATICS_TEACH_IN_MODE

Name	Data type	Initial value	Comment
NoAction	UINT	16#0	Executes the current block and commands the next block
StopMovement	UINT	16#1	Stops the current block and starts the next block

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

MB_KINEMATICS_PCS_CONFIG

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-27: MB_KINEMATICS_PCS_CONFIG reference table

Brief description

The MB_KINEMATICS_PCS_CONFIG structure is a substructure of [MB_KINEMATICS_CONTROL_TYPE01, page 369](#) and contains variables to configure the product coordinate system (PCS) of a kinematics.

STRUCT MB_KINEMATICS_PCS_CONFIG

Name	Data type	Initial value	Comment
Execute	BOOL	FALSE	Start PCS configuration
Origin	ARRAY [1..6] OF REAL	[6 (0)]	Orientation of the PCS in MCS • [TransX, TransY, TransZ, RotationX, RotationY, RotationZ]
CoordType	ML_KIN_CS_TYPE	CS_TYPE_CARTESIAN	PCS type • Cartesian coordinate system • Cylindric coordinate system

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

ML_KinTech.library

MB_KINEMATICS_SETUP

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-28: MB_KINEMATICS_SETUP reference table

Brief description The MB_KINEMATICS_SETUP structure is a substructure of [MB_KINEMATICS_CONTROL_TYPE01](#), [page 369](#), and contains control variables for jogging a kinematics.

STRUCT MB_KINEMATICS_SETUP

Name	Data type	Initial value	Comment
Enable	BOOL	FALSE	Enable jog/setup mode
CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	<i>Possible coordinate systems (ML_Robot.library)</i> - CS_MACHINE_COORDINATES - CS_AXES_COORDINATES - CS_BASE_COORDINATES - CS_PRODUCT_COORDINATES_n (n = 1..16)
JogMode	ML_JOGMODE	JM_STEP	Jog type ML_JOGMODE , page 389
JogPlus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL	[MB_MAX_KIN_AXIS_NUMBER(FALSE)]	<i>Jog</i> - Incremental if JogMode = JM_CONT -> FALSE - Continuous if JogMode = JM_STEP -> TRUE
JogMinus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL	[MB_MAX_KIN_AXIS_NUMBER(FALSE)]	<i>Jog</i> - Incremental if JogMode = JM_CONT -> FALSE - Continuous if JogMode = JM_STEP -> TRUE
Distance	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(1.0)]	Jog distance
Velocity	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(10.0)]	Jog velocity
Acceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(10.0)]	Jog acceleration
Deceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(10.0)]	Jog deceleration
Jerk	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(10.0)]	Jog jerk

Distance This element contains the distance for incremental jogging in the axis and Cartesian coordinate system.



When jogging with the kinematic interface, only KIN_JOG_STEP_SMALL is used for traversing and the step width is determined using "Distance".

Velocity This element contains the low velocity for continuous jogging in the axis and Cartesian coordinate system.



When jogging with the kinematic interface, only KIN_JOG_CONT_SLOW is used for traversing and the traversing velocity is determined using "Velocity".

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

MB_KINEMATICS_COORDINATED

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Control

Tab. 7-29: MB_KINEMATICS_COORDINATED reference table

Brief description The MB_KINEMATICS_COORDINATED structure is a substructure of MB_KINEMATICS_CONTROL_TYPE01, [page 369](#), and contains control variables and command values for the positioning operating modes.

The elements contained in this structure correspond to the inputs of the function blocks:

- MC_MoveLinearAbsolute, ML_Robot
- MC_MoveLinearRelative, ML_Robot
- MC_MoveDirectAbsolute, ML_Robot
- MC_MoveDirectRelative, ML_Robot
- MC_MoveCircularAbsolute, ML_Robot
- MC_MoveCircularRelative, ML_Robot
- ML_RobotMoveJumpAbsolute, ML_Robot
- ML_RobotOpenCyclicChannel, ML_Robot
- ML_RobotOpenDirectCyclicChannel, ML_Robot
- ML_RobotWriteCyclicChannel, ML_Robot

Please refer to the relevant documents for more information: Rexroth IndraMotion MLC 14VRS, Robot Control V2, DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588.

ML_KinTech.library

STRUCT MB_KINEMATICS_COORDINATED

Name	Data type	Initial value	Comment
Point	BOOL	FALSE	Target position, absolute or relative (unit: e.g. in [degree] according to parameterization) (negative or positive) Release with positioning operating modes Note: Is permanently cyclically processed (EnableCyclicScanning = TRUE)
Velocity	REAL	10.0	Positioning velocity, (unit: e.g. in [rad/s ²] according to parameterization) (enter as positive) Release with positioning operating modes
Acceleration	REAL	10.0	Acceleration (unit: e.g. in [rad/s ²] according to parameterization) (always positive) Release with positioning operating modes
Deceleration	REAL	10.0	Deceleration (unit: e.g. in [rad/s ²] according to parameterization) (always positive) Release with positioning operating modes
Jerk	REAL	0.0	Jerk (unit: e.g. in [rad/s ³] according to parameterization) (always positive) Release with positioning operating modes
CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	<i>Possible coordinate systems (ML_Robot.library)</i> • CS_MACHINE_COORDINATES • CS_AXES_COORDINATES • CS_BASE_COORDINATES • CS_PRODUCT_COORDINATES_n (n = 1..16)
SlopeType	ML_SLOPE_TYPE	ST_BLOCK_SLOPE	Slope type of the motion (BlockSlope, Program Slope or ContinuousSlope) ML_Robot.library Release with positioning operating modes
SetMode	ML_SET_MODE	SM_TO	Defines whether the motion ends at the end point or whether is blended (to or via) ML_Robot.library Release with positioning operating modes
BlendingRadius	REAL	0.0	Blending radius (unit: [%] of parameter A-0-0230) Release with positioning operating modes
CircularAuxPoint	ARRAY [1..16] OF REAL	[16(0.0)]	Target position, circular (unit: e.g. in [rad/s ²] according to parameterization) (enter as negative or positive) Release with positioning operating modes "Circular"

Name	Data type	Initial value	Comment
CircularCircMode	ML_CIRCMODES	CM_BOARDER	Defines the interpretation of the "AuxPoint" input ML_Robot.library Release with positioning operating modes "Circular"
CyclicMode	ML_CYCLIC_MODE	CYCLIC_MODE_POS_ABS	Selection of the cyclic position command value specification - Cyclic command value specification as absolute position - Cyclic command value is specified as relative delta to the last commanded position in each cycle - Cyclic command value is specified as velocity
CyclicChannelType	ML_CYCLIC_CHANNEL_MODE_SUBTYPE	ML_CYCLIC_CHANNEL_MODE_LINEAR	Channel type - Input of the ML_RobotWriteCyclicChannel function block - ML_CYCLIC_CHANNEL_MODE_LINEAR - ML_CYCLIC_CHANNEL_MODE_DIRECT
CyclicVelocity	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(0.0)]	Cyclic velocity specification - Value interpretation depending on CyclicMode - is only used with CYCLIC_MODE_VEL
StartHeight	REAL	0.0	Starting height of MoveJump ML_Robot.library Release with positioning operating modes "Jump"
MaxHeight	REAL	0.0	Maximum height of MoveJump ML_Robot.library Release with positioning operating modes "Jump"
EndHeight	REAL	0.0	End height of MoveJump ML_Robot.library Release with positioning operating modes "Jump"

- [Folder Kin_Interface/Type/Control, overview on page 368](#)

ITF_KINEMATICS structures

The event-controlled kinematic interface (ML_KinematicsInterfaceType11) uses so-called interfaces for specifying the operating modes and command values. Advantage: Writing of values to the control structure triggers an immediate action which is contained in the Get and Set methods of the individual properties or structure elements. MB_KINEMATICS_CONTROL_TYPE11 created for the event-controlled kinematic interface uses this interface. The design of this structure

corresponds to the design of MB_KINEMATICS_INTERFACE_TYPE01, as shown in the following figure.

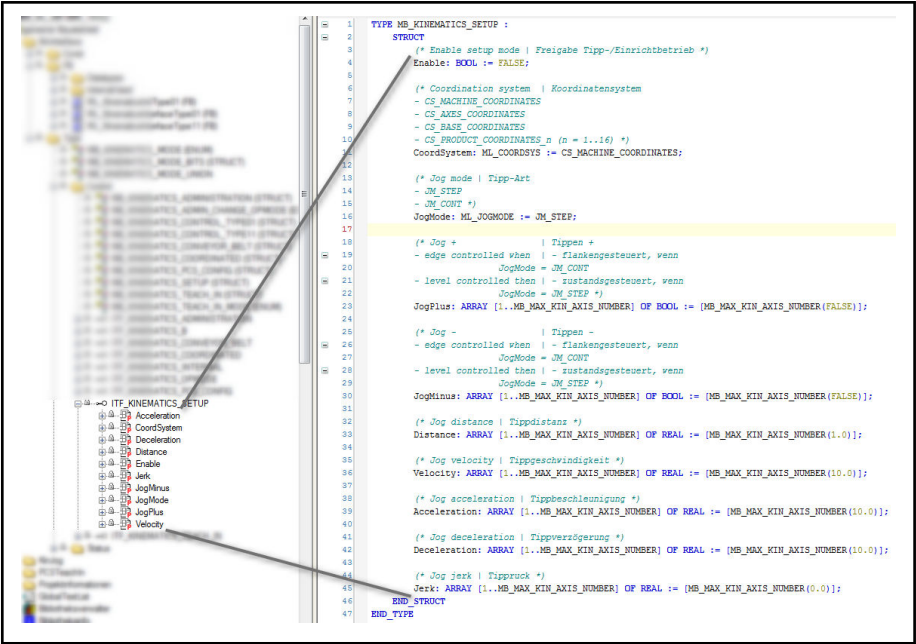


Fig. 7-5: Comparison ITF_KINEMATICS_SETUP to MB_KINEMATICS_SETUP

Due to the identical naming of the properties, the commanding of the kinematic interface does not change between Type01 and Type11. Already existing program code can be applied.

7.4.7 KinInterface/Type/Status

Folder Kin_Interface/Type/Status, overview

MB_KINEMATICS_STATUS_TYPE01 on page 380	
	MB_KINEMATICS_ADMIN_STATUS on page 381
	MB_KINEMATICS_ADMIN_STATUS on page 381
	MB_KINEMATICS_TEACH_IN_STATUS on page 384
	MB_KINEMATICS_PCS_STATUS on page 384
	MB_KINEMATICS_DIAGNOSIS on page 385
	MB_KINEMATICS_SETUP_STATUS on page 386

Tab. 7-30: MB_KINEMATICS_STATUS_TYPE01, structural design

MB_KINEMATICS_STATUS_TYPE01

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-31: MB_KINEMATICS_STATUS_TYPE01 reference table

Brief description The MB_KINEMATICS_STATUS_TYPE01 structure is accessed via the arKinStatus_gb[] instance.

STRUCT MB_KINEMATICS_STATUS_TYPE01

Name	Data type	Initial value	Comment
PtrKinData	POINTER TO ML_KINEMATICS_DATA_SM	0	Pointer to KinData interface of the kinematics (ML_Robot.library) - Input .PtrKinData^ for drop-down menu
Admin	MB_KINEMATICS_ADMIN_STATUS		Kinematic status MB_KINEMATICS_ADMIN_STATUS, page 381
Diag	MB_KINEMATICS_DIAGNOSIS		Diagnostic information of the kinematics - Error, warning, etc. MB_KINEMATICS_DIAGNOSIS, page 385
SetupMode	MB_KINEMATICS_SETUP_STATUS		Setup functions MB_KINEMATICS_SETUP_STATUS, page 386

- [Folder Kin_Interface/Type/Status, overview, page 380](#)

MB_KINEMATICS_ADMIN_STATUS

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-32: MB_KINEMATICS_ADMIN_STATUS reference table

Brief description

The MB_KINEMATICS_ADMIN_STATUS structure is a substructure of [MB_KINEMATICS_STATUS_TYPE01, page 380](#) and contains the output status for the administration of the kinematics.

STRUCT MB_KINEMATICS_ADMIN_STATUS

Name	Data type	Initial value	Comment
_OpModeAck	MB_KINEMATIC_S_MODE	ModeCoord Ungrouped	This element outputs the status of the operating mode
MODE_COORD_AB	BIT	FALSE	These elements are used solely as placeholders and are only permissible in the Control structure
MODE_COORD_AH	BIT	FALSE	These elements are used solely as placeholders and are only permissible in the Control structure
MODE_COORD_HOMING	BIT	FALSE	These elements output the status of the operation mode as a bit. If none of the bits are set, then this signals the "ModeCoordUngrouped" status AH, AxisInterface -> SetupMode, OpMode(Ack).2
MODE_COORD_POS_DIR_ABS	BIT	FALSE	Absolute PTP positioning, AF, MC_MoveDirectAbsolute, OpMode(Ack).3
MODE_COORD_POS_DIR_REL	BIT	FALSE	Relative PTP positioning, AF, MC_MoveDirectRelative, OpMode(Ack).4
MODE_COORD_POS_LIN_ABS	BIT	FALSE	Absolute linear positioning, AF, MC_MoveLinearAbsolute, OpMode(Ack).5

ML_KinTech.library

Name	Data type	Initial value	Comment
MODE_COORD_POS_LIN_REL	BIT	FALSE	Relative linear positioning, AF, MC_MoveLinearRelative, OpMode(Ack).6
MODE_COORD_POS_CIRC_ABS	BIT	FALSE	Absolute circular positioning, AF, MC_MoveCircularAbsolute, OpMode(Ack).7
MODE_COORD_POS_CIRC_REL	BIT	FALSE	Relative circular positioning, AF, MC_MoveCircularRelative, OpMode(Ack).8
MODE_COORD_POS_JUMP_ABS	BIT	FALSE	Absolute jump positioning, AF, ML_RobotMoveJumpAbsolute, OpMode(Ack).9
MODE_COORD_STOPPING	BIT	FALSE	Kinematic stop, AH, MC_GroupStop, OpMode(Ack).10
MODE_COORD_INTERRUPTED	BIT	FALSE	Operating mode "Interrupted", AF, MC_GroupInterrupt, OpMode(Ack).11
MODE_COORD_CONTINUE	BIT	FALSE	Operating mode "Continue", AF, MC_GroupContinue, OpMode(Ack).12
MODE_COORD_EXTERNAL_FB	BIT	FALSE	Operating mode "External function block", power on subsequently , the external function block takes over the control, OpMode(Ack).13
MODE_COORD_CYCLIC	BIT	FALSE	Cyclic position command value specification, AF, ML_RobotWriteCyclicChannel, OpMode(Ack).14
Active	BOOL	FALSE	TRUE: Kinematic enabled Information: Is set before initializing of the ML_KinematicsInitType01 function block
Name	STRING(40)	"	Name of the kinematics Information: Is set before initializing of the ML_KinematicsInitType01 function block
TrafoNb	DINT	999	Number of the current transformation scheme Information: Is set before initializing of the ML_KinematicsInitType01 function block
CmdDone	BOOL	FALSE	Last command is completed Valid for all positioning operating modes
InBuffer	BOOL	FALSE	Last command is in block buffer Valid for all positioning operating modes
ParamLevelAck	BOOL	FALSE	Axes are in parameterization mode 1, Parameter A-0-0024 bit 3 is active - TRUE: Parameterization mode enabled - FALSE: Parameterization mode disabled
ConveyorBelt	ARRAY [1..MB_MAX_KI N_AXIS_NUMB ER] OF MB_KINEMATIC S_CONV_BELT_ STATUS		Status of the belt configuration

Name	Data type	Initial value	Comment
KinTransformPLCDone	BOOL	FALSE	User-defined transformation has been logged in successfully
TeachIn	MB_KINEMATIC S_TEACH_IN_S TATUS		Status of teaching and saving of points MB_KINEMATICS_TEACH_IN_STATUS on page 384
PCSStatus	ARRAY [1..ML_MAX_KI N_PCS_NUMBE R] OF MB_KINEMATIC S_PCS_STATU S		Status of the configured product coordinate system

- [Folder Kin_Interface/Type/Status, overview on page 380](#)

MB_KINEMATICS_CONV_BELT_STATUS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-33: MB_KINEMATICS_CONV_BELT_STATUS reference table

Brief description

The MB_KINEMATICS_CONV_BELT_STATUS structure returns the current belt configuration.

The elements contained in this structure are equivalent to the function block inputs (ML_Robot.library):

- ML_RobotGetBeltConfiguration
- ML_RobotGetBeltErrorReaction

Please refer to the relevant documents for further information.

STRUCT MB_KINEMATICS_CONV_BELT_STATUS

Name	Data type	Initial value	Comment
ConveyorBeltOrigin	ARRAY [1..6] OF REAL	[6(0.0)]	Returns the position and rotation in relation to the MCS
TotalLength	REAL	0.0	Returns the length of the belt-synchronous travel ranges from the starting position
BeginLength	REAL	0.0	Returns the length of the late start of a belt-synchronous motion from the starting position
CoordType	ML_KIN_CS_TYPE	CS_TYPE_CARTESIA N	Returns the coordinate system type
ErrorReactionType	ML_BELT_ERROR_ REACTION_MODE	ML_BELT_ERROR_ NO_SPECIFIC_ REACTION	<i>Returns the error reaction type:</i> • ML_BELT_ERROR_NO_SPECIFIC_REACTION • ML_BELT_ERROR_DESYNC_OVER_TIME • ML_BELT_ERROR_DESYNC_WITH_LIMITS

ML_KinTech.library

Name	Data type	Initial value	Comment
ErrorReactionDesyncTime	TIME	T#300ms	Returns the time for desynchronization
ErrorReactionDeceleration	REAL	0.0	Returns the deceleration rate for desynchronization
ErrorReactionJerk	REAL	0.0	Returns the deceleration jerk for desynchronization
Done	BOOL	FALSE	Configuration of a transport belt is completed successfully

- [Folder Kin_Interface/Type/Status, overview on page 380](#)

MB_KINEMATICS_TEACH_IN_STATUS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-34: MB_KINEMATICS_SETUP_STATUS reference table

Brief description MB_KINEMATICS_TEACH_IN_STATUS represents the status structure of MB_KINEMATICS_TEACH_IN.

STRUCT MB_KINEMATICS_TEACH_IN_STATUS

Name	Data type	Initial value	Comment
TeachInModeDone	BOOL	FALSE	Status of the last teach command • TRUE: Processing completed successfully • FALSE: Processing failed or was not started
Point	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	[MB_MAX_KIN_AXIS_NUMBER(0.0)]	Last read point
CoordSystem	ML_COORDSYS	CS_MACHINE_COORDINATES	Coordinate system of the last read point
PointNameOut	STRING(19)	"	Name of the next point

- [Folder Kin_Interface/Type/Status, overview on page 380](#)

MB_KINEMATICS_PCS_STATUS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-35: MB_KINEMATICS_PCS_STATUS reference table

Brief description The MB_KINEMATICS_PCS_STATUS structure is a substructure of [MB_KINEMATICS_STATUS_TYPE01, page 380](#) and contains information about the configuration of the product coordinate system (PCS) of the kinematics.

STRUCT MB_KINEMATICS_SETUP_STATUS

Name	Data type	Initial value	Comment
InBuffer	BOOL	FALSE	PCS configuration in the block buffer
Done	BOOL	FALSE	PCS configuration completed successfully
CoordType	ML_KIN_CS_TYPE	CS_TYPE_CARTESIAN	Returns the type of the configured PCS: - Cartesian coordinate system - Cylindric coordinate system
Origin	ARRAY [1..6] OF REAL	[6 (0)]	Returns position and orientation of the PCS: [TransX, TransY, TransZ, RotationX, RotationY, RotationZ]

- Folder [Kin_Interface/Type/Status](#), overview on page 380

MB_KINEMATICS_DIAGNOSIS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-36: MB_KINEMATICS_DIAGNOSIS reference table

Brief description The MB_KINEMATICS_DIAGNOSIS structure is a substructure of [MB_KINEMATICS_STATUS_TYPE01](#), on page 380 and contains warning and error outputs.

STRUCT MB_KINEMATICS_DIAGNOSIS

Name	Data type	Initial value	Comment
Warning	BOOL	FALSE	This element shows a copy of KinData[].Kin_Warning or arAxisStatus_gb[].Diag.Warning and signals that the kinematics or their associated axes output a warning.
Error	BOOL	FALSE	This element signals that the kinematics, their associated axes, or an internal function block of the KinematicInterface report an error
ErrorID	ERROR_CODE	NONE_ERROR	This element outputs the type of error in the kinematics, in an associated axis, or in an internal function block
ErrorIdent	ERROR_STRUCT	STRUCT (Table := NO_TABLE_USED, Additional1 := 0, Additional2 := 0)	This element outputs the error ID in the kinematics, in an associated axis, or in an internal function block
Number	DWORD	16#0	This element outputs the diagnostic number in the kinematics, in an associated axis, or in an internal function block
Message	STRING(60)	"	This element outputs the diagnostic message in the kinematics, in an associated axis, or in an internal function block

arKinCtrl_gb[].Admin.DiagNbrRefreshTime is used to control the refresh rate of the "Number" element.

As soon as element "Number" changes, the text "Message" is read again.

arKinCtrl_gb[].Admin.DiagNbrRefreshTime can be used to switch off reading the diagnostic information (DiagNbrRefreshTime := T#0ms).

ML_KinTech.library

- [Folder Kin_Interface/Type/Status, overview on page 380](#)

MB_KINEMATICS_SETUP_STATUS

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinInterface/Type/Status

Tab. 7-37: MB_KINEMATICS_SETUP_STATUS reference table

Brief description The MB_KINEMATICS_SETUP_STATUS structure is a substructure of MB_KINEMATICS_STATUS_TYPE01, [page 380](#) and contains information about jogging of the kinematics.

STRUCT MB_KINEMATICS_SETUP_STATUS

Name	Data type	Initial value	Comment
EnableAck	BOOL	FALSE	Signalizes whether the setup mode is active (= TRUE) or not (= FALSE)

- [Folder Kin_Interface/Type/Status, overview on page 380](#)

7.5 KinJog

7.5.1 KinJog/FB

ML_KinJog

Brief description The ML_KinJog function block is used for jogging individual axes or axis groups in the kinematics. This function block has to be called cyclically.

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinJog/FB

Tab. 7-38: Reference table ML_KinJog

Interface description

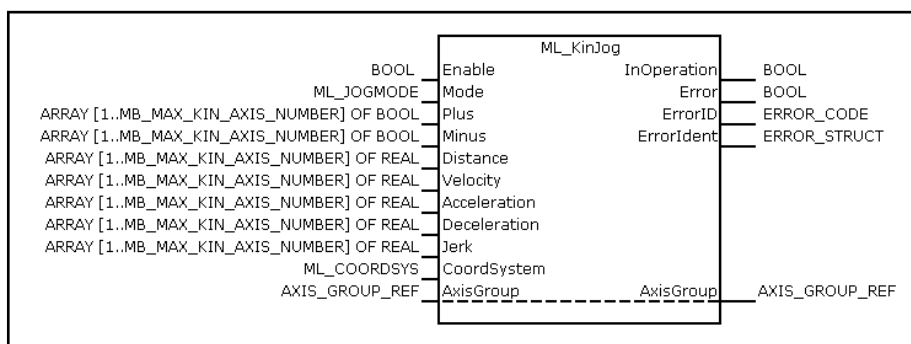


Fig. 7-6: ML_KinJog function block

I/O type	Name	Data type	Comment
VAR_INPUT	Enable	BOOL	Processing enable for function block
	Mode	ML_JOGMODE	Specification of jog type (incremental, continuous) ML_JOGMODE, page 389

I/O type	Name	Data type	Comment
	Plus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL	Contains coordinate elements that can be jogged in a positive direction
	Minus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL	Contains coordinate elements that can be jogged in a negative direction
	Distance	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specification of jog distance for each axis
	Velocity	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specification of jog velocity for each axis
	Acceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specification of acceleration for the Jog mode for each axis
	Deceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specification of deceleration for the Jog mode for each axis
	Jerk	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL	Specification of jerk for the Jog mode for each axis
	CoordSystem	ML_COORDSYS	Specification of jogging coordinate system
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 CONSTANT	AxisGroup	AXIS_GROUP_REF	Reference to the kinematics going to be jogged

Tab. 7-39: Interface variables of ML_KinJog

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Mode	ML_JOGMODE			JM_STEP	Continuous
Plus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL			16(FALSE)	Continuous
Minus	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF BOOL			16(FALSE)	Continuous
Distance	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL			16(0.0)	Rising edge at "Plus" or "Minus"

ML_KinTech.library

Name	Type	Min. value	Max. value	Default value	Effective
Velocity	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL			16(0.0)	Rising edge at "Plus" or "Minus"
Acceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL			16(0.0)	Rising edge at "Plus" or "Minus"
Deceleration	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL			16(0.0)	Rising edge at "Plus" or "Minus"
Jerk	ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL			16(0.0)	Rising edge at "Plus" or "Minus"
CoordSystem	ML_COORDSYS			CS_MACHINE_COORDINATES	Rising edge at "Plus" or "Minus"

Tab. 7-40: Input behavior of ML_KinJog

Functional description The ML_KinJog function block can be used to jog the kinematics with RobotControl V2. Here, jogging in the Cartesian coordinate area is equivalent to a linear travel motion, whereas jogging in the axis coordinate area is the same as a PTP travel motion. After the function block is successfully started, it automatically groups the axes associated with the kinematics.

Mode This input specifies the type of jogging (incremental, continuous).

The input can assume the following values:

- **JM_STEP:** Incremental jogging
- **JM_CONT:** Continuous jogging

Plus This input activates the travel motion in a positive direction for the relevant axis.

Minus This input activates the travel motion in a negative direction for the relevant axis.

Distance, Velocity, Acceleration, Deceleration, Jerk These inputs define the jogging distance, jogging velocity, jogging acceleration and the jogging jerk used for incremental and continuous jogging. The values are interpreted as absolute (MCS, BCS, PCS) or percentage values (ACS) depending on the coordinate system selected. In the case of a percentage interpretation, the values relate to the corresponding limit values for the axis.

CoordSystem This input specifies the coordinate system in which jogging is supposed to take place.

The input can assume the following values:

- **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_16:** Product coordinate system 16

Troubleshooting 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
RESOURCE_ERROR, (16#0003)	16#26FF	16#0000	Invalid RobotControl version

Tab. 7-41: Error codes of ML_KinJog

7.5.2 KinJog/Type

ML_JOGMODE

Assignment: Target system/library

Target system	Library/folder
IndraMotion MLC 14VRS ROB function package	ML_KinTech/KinJog/Type

Tab. 7-42: ML_KinematicsInitType01 reference table

Brief description The enumeration type "ML_JOGMODE" contains the available modes used for jogging the kinematics.

This enumeration type is used by the following function block:

- ML_KinJog

ENUM ML_JOGMODE

Name	Data type	Initial value	Comment
JM_STEP	INT	0	Incremental jogging
JM_CONT	INT	1	Continuous jogging

8 Application examples

8.1 Application examples, overview

In the following, the Robot Control application is explained, using different examples.

8.2 Using coordinate systems

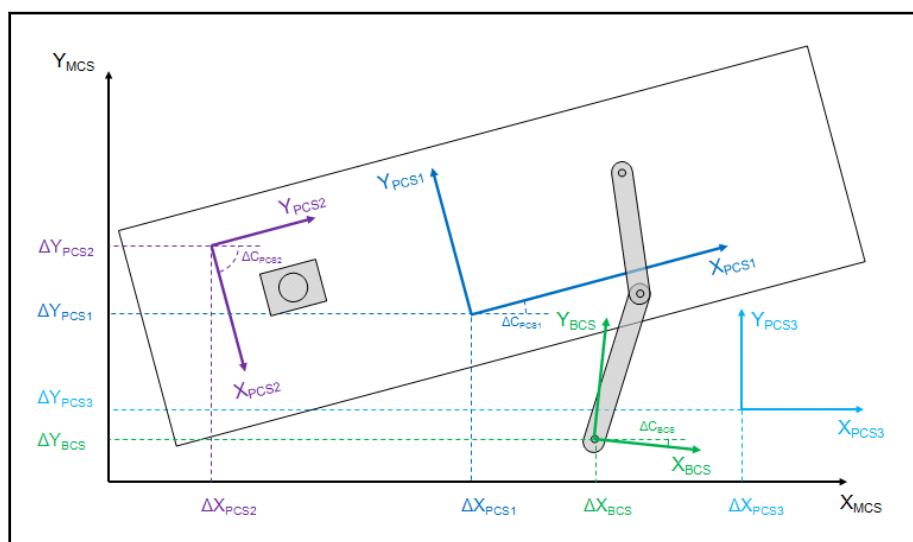


Fig. 8-1: Application example: Coordinate systems

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 37](#) with $(\Delta X_{PCS1}, \Delta Y_{PCS1}, 0, 0, 0, \Delta C_{PCS1})$.

PCS2: Camera coordinate system

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 37](#) with $(\Delta X_{PCS2}, \Delta Y_{PCS2}, 0, 0, 0, \Delta C_{PCS2})$.

PCS3: Palette coordinate system

To facilitate the positioning of products on a palette, a PCS is defined as palette coordinate system.

Moving and rotating the PCS3 to the MCS is defined via [Set Cartesian Transformation, page 37](#) with $(\Delta X_{PCS3}, \Delta Y_{PCS3}, 0, 0, 0, 0)$.

BCS: Robot coordinate system

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 palette.

The camera records the project 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 36](#)".

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 palette 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.

8.3 Belt synchronization

Initial situation

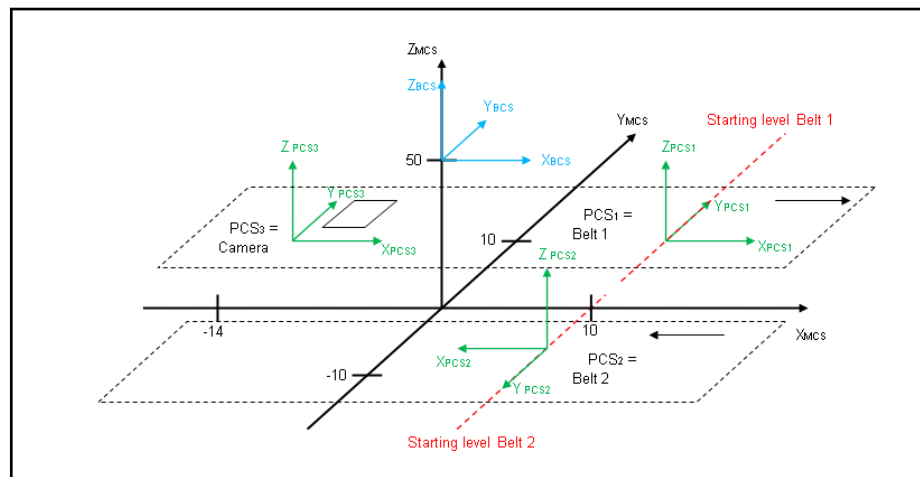


Fig. 8-2: Application example: Belt-synchronous

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 37](#) (-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 37](#) (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. To avoid waiting times, the adaptive interpolation can be enabled optionally (see [Adaptive motion, page 72](#)). The initial planes of both belts do not necessarily 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) $_{BCS}$.

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 36).

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 38.

The kinematics remains belt-synchronous until a second 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. 8-3: 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.

8.4 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 152 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
    ATCP, 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];

ATCP := SQRT((L1*L1) + (L2*L2) - (2 * L1 * L2 * COS(Gamma* GRAD_RAD_FAKTOR))); // Law of cosines
Ax := TCPx - SQRT(ATCP*ATCP - TCPy*TCPy);

Beta := ATAN2(-TCPy, (TCPx-Ax));
Delta := ACOS(-(L2*L2) - (L1*L1) - (ATCP * ATCP)) / (2 * L1 * ATCP); // 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.

Usage

```

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

```

8.5 Extended user-defined transformation

This example implements the 3-axis Scara transformation (transformation scheme 9) as 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 115](#) 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 access the interface methods.

Implementing the GetUsedAxes method

The GetUsedAxes method returns the degrees of freedom of the transformation. In this case, the transformation has its degrees of freedom in the X-Y plane. The additional orientation axis (rotation around C) is not specified as orientation axis but as auxiliary axis. As this axis is always defined, it does not have to be specified.

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 both positioning axes are scaled 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;
END_IF

IF (TransformationLengths[2] > 0.0) THEN
    dL2 := TransformationLengths[2];
ELSE
    Initialisation := 16#F12C008A; // trafo values wrong
    RETURN;
```

Application examples

```

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 arm behavior of the Scara robot is determined in the DetermineMemory method. From the axis coordinates, it is calculated whether the kinematics is right-armed or left-armed. The "TrafoChoice" is ignored. If the value is "TRUE", the arm behavior could be used for the next inverse position kinematics in which the first moves the least. Consequently, the duration of motions can be decreased when assuming that axis 2 is more dynamic than axis 1. However, this is not implemented here.

Determine Memory method

```

METHOD DetermineMemory : UDINT
VAR_INPUT
  _ACS : REFERENCE TO ARRAY [1..16] OF LREAL;
  TrafoChoice : 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.

Take the fault scenarios into consideration and the fact that the auxiliary axis does not have to be copied. This is done automatically by the system!

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.

Take the fault scenarios into consideration and the fact that the auxiliary axis does not have to be copied. This is done automatically by the system!

The auxiliary function $\text{atan2}(y, x)$ is not defined in the system and has to be implemented according to the ATAN2 definition (an example is given in the subsequent section).

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 phil
  dTheta : LREAL;          //pitch angle of the distance from the TCP to the origin; leads to phil
  dTemp : LREAL;           //temporary variable
  dPHI1 : LREAL;           //angle of the motor in degrees
  dPHI2 : LREAL;           //angle of the motor in degrees
  dphil1 : LREAL;          //angle of the motor in radian; d_phi12 = d__phi1 + d__phi2
  dphi12 : 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; // to 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
```

Application examples

```

//left arm
dphi1 := dBeta + dTheta;
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;

```

**Example implementation of the
atan2 function**

In the following, an atan2 function is displayed which can be used for the backward transformation.

atan2 function

```

FUNCTION ATAN2 : REAL
VAR_INPUT
    y : REAL;
    x : REAL;

END_VAR
VAR
    lrPI: LREAL := 3.14159265359;
END_VAR

IF x > 0 THEN
    ATAN2 := ATAN(y / x);
END_IF

IF x < 0 THEN
    IF y >= 0 THEN
        ATAN2 := lrPI + ATAN(y / x);
    ELSE
        ATAN2 := -lrPI + ATAN(y / x);
    END_IF
END_IF

IF x = 0 THEN
    ATAN2 := y / ABS(y) * (lrPI / 2);
END_IF

```

GetMechanic method

The GetMechanic method calculates the corresponding mechanical data for a point in the axis coordinate system. The mechanical view for Scara should consist of 3 spheres (BASE, MID, TCP) and 2 cylinders (BASE to MID, MID to TCP). Take into consideration that the object name "TCP" is used by most visualizations to record the path. The transformation length 2 ([K-0-0040](#)) is

used for the size of rods and spheres. For simplicity's sake, the ML_AddMechanicData function of 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);
```

Application examples

```
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 PLCTrafoExt2
VAR
    myScara          : Scara;
    setKinTrafo      : MC_SetKinTransformExt;
    initTransformation : BOOL;
END_VAR

IF(initTransformation) THEN
    setKinTrafo.KinTransform := myScara;
    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
```

8.6 Extending a point

It is reasonable to connect positioning axes if a FlexStamper kinematics is to be used together with the motion type "Jump". The motion type "Jump" cannot be used, as the FlexStamper transformation is only in the X-Y-plane and a Jump motion can only be commanded in Z-direction.

By twisting the robot along the X-axis (A) by 90° in the parameter [K-0-0042](#), the robot goes into the X-Z-space. Thus, MoveJump can be used, but the Z-coordinate of the end point cannot be commanded, as only the X- and Y-coordinates are defined by the transformation. After setting bit 7 of the parameter [K-0-0099](#), the commanding of all positioning axes are enabled.

Transformation scheme	Point	Connecting axes	positioning
FlexStamper (39)	(X, Y, H1)	(X, Y, Z, H1)	

8.7 Belt synchronization with "Adaptive Motion"

In the following example, a scheme explains how a belt-synchronous point P is approached with and without "Adaptive Motion".

The coordinates of the commanded point P at the time t_0 are identical in the following drawings.

In the drawing [fig. 8-4 "Point P is approached without Adaptive Motion"](#) on [page 403](#), the point is approached without "Adaptive Motion".

- Point P is commanded by the user at the time t_0 .
- As the synchronous end point of the motion is before the start plane, the kinematics waits until the point has passed the start plane.

- If the commanded end point P reached the belt-synchronous workspace at the time t_1 , the motion starts. The kinematics starts the synchronization
- At the time t_2 , the motion is completed and the kinematics is synchronous to the end point P on the belt

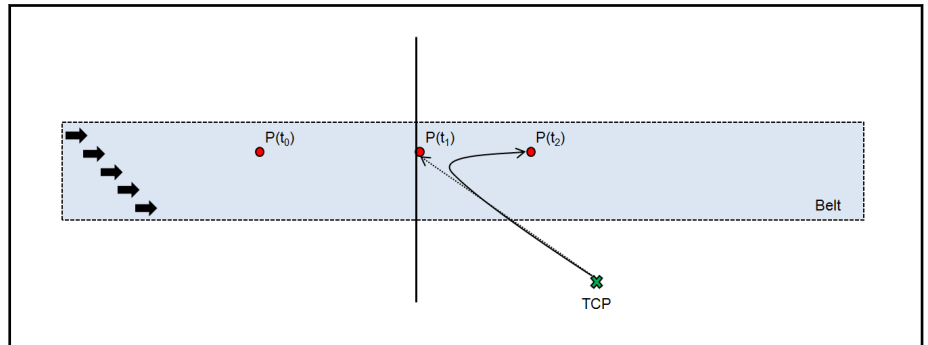


Fig. 8-4: Point P is approached without "Adaptive Motion"

In the drawing fig. 8-5 "Point P is approached with Adaptive Motion" on page 403, the point is approached with "Adaptive Motion".

- Point P is again commanded by the user at the time t_0 .
- Even though the synchronous end point of the motion is before the start plane, the kinematics starts the motion immediately at time t_0 . There is not waiting time for the kinematics
- If the commanded end point P reaches the belt-synchronous workspace at the time t_1 , the motion is completed. At this time, the kinematics is synchronous to the end point P on the belt

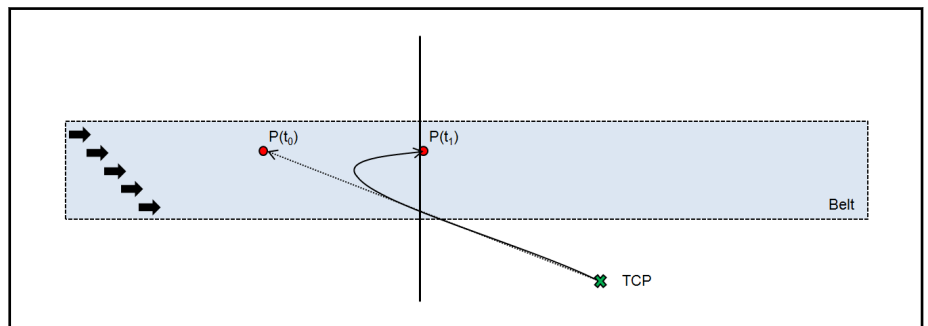


Fig. 8-5: Point P is approached with "Adaptive Motion"

Conclusion

As the kinematics with "Adaptive Motion" reaches the target point in which the kinematics without "Adaptive Motion" starts with the motion, the time between time t_1 and time t_2 is saved.

Thus, the "Adaptive Motion" functionality facilitates a time-optimized behavior which results in saving of energy apart from a motion preventing damage to the machine.

8.8 Determining the command buffer size

In the following example, 100 commands, each of the commands 1mm in the same direction are used in connection with Cont_Slope. By Cont_Slope, these commands are summarized in a motion but are logically retained as individual commands. Additionally, Cont_Slope ensures that the velocity can be reduced (standstill) at the last known velocity.

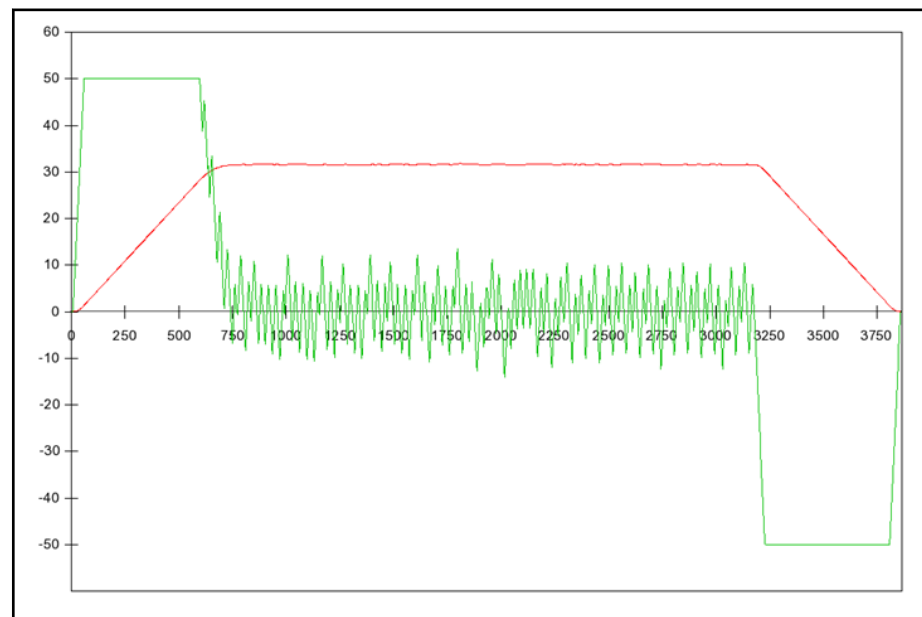
Application examples

Case 1: Block buffer of 12

12 is entered in parameter [K-0-0097](#), Maximum command buffer size. Thus, a maximum of 12 commands can be simultaneously contained in the block buffer.

More commands can only be issued once a command is completed. If commands are issued now, only the first 12 commands can be written to the command buffer directly. All other commands are queued. As long as sufficient deceleration path is available up to the last known target point, acceleration is possible.

If no new command can be issued in the command buffer due to the command buffer limit, it has to be decelerated at a certain point (or less acceleration has to be used) to stop at the last known target point. A command is completed upon deceleration. There is now space for a new command. The available deceleration path is extended. Thus, it can be accelerated again. This is continued until all blocks have been processed.



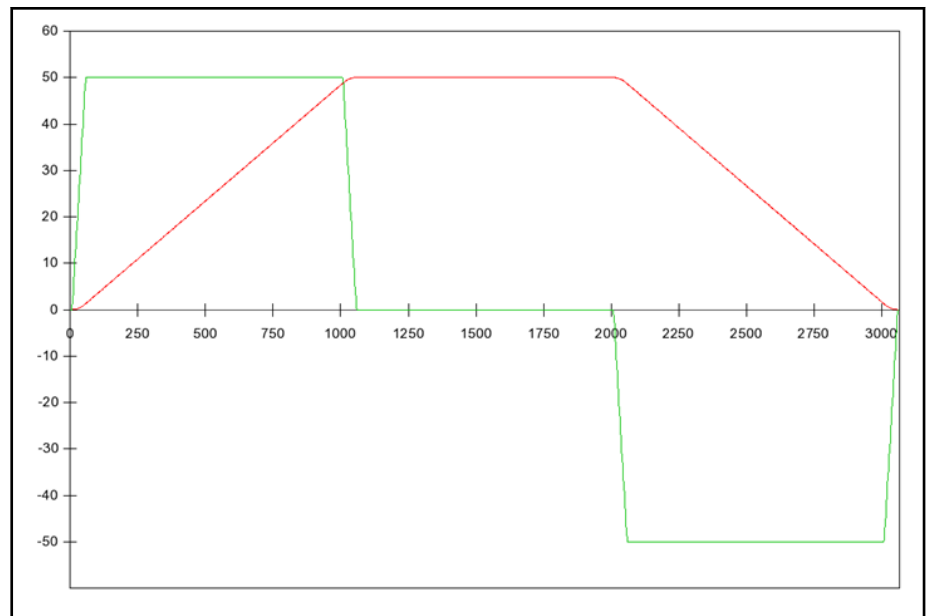
Red Velocity
Green Acceleration

Fig. 8-6: Time diagram command buffer size 12

A velocity smaller than the programmed velocity is the result. Furthermore, the resulting velocity is not constant. If there is space for a new command in the command buffer, the velocity is increased. If the deceleration path is reached, the velocity is decreased again.

Case 2: Block buffer of 100

100 is entered in parameter [K-0-0097](#), Maximum command buffer size. Thus, all 100 commands can be written in the command buffer. The complete path length is known. Acceleration on the path can be continued until the maximum path velocity is reached.



Red Velocity
 Green Acceleration

Fig. 8-7: Time diagram command buffer size 100

The result is a "smooth" velocity curve. As a higher velocity is reached, the target point is reached more quickly.

8.9 Kinematic synchronization with cyclic command value specification

Using the cyclic command value specification, the kinematic motion can be user-defined. Using cyclically generated, absolute or relative command values can be used to traverse position- or velocity-accurately. A master/slave relation between two kinematics can be realized.

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. Thus, the kinematics to be synchronized should be at the same position in the selected coordinate system!

Cyclic command values can be specified in two different ways. Adding the command values in MCS is performed in the "Open Cyclic Channel" command. If an addition is required in the axis coordinate system, the "Open Direct Cyclic Channel" command is used.

This example used the "Open Direct Cyclic Channel" command to facilitate a change of arm behavior. This means that motions of the synchronized kinematic are executed as incrementally smaller PTP motions.

Declaration

```
FUNCTION_BLOCK RobotSync
VAR_INPUT
    Enable          : BOOL := FALSE;
    Deceleration    : REAL := 1000;
    Jerk            : REAL := 10000;
    CoordSystem     : ML_COORDSYS := CS_AXES_COORDINATES;
    SetMode         : ML_SET_MODE := SM_TO;
    BlendingRadius : REAL := 0.0;
```

Application examples

```

MasterGroup   : AXIS_GROUP_REF;
SlaveGroup    : AXIS_GROUP_REF;
END_VAR
VAR_OUTPUT
  InOperation: BOOL := FALSE;
  Error       : BOOL := FALSE;
  ErrorID     : ERROR_CODE := NONE_ERROR;
  ErrorIdent  : ERROR_STRUCT := (Table:= NO_TABLE_USED, Additional1:= 16#0000, Additional2:= 16#0000);
END_VAR
VAR
  bOpen       : BOOL := FALSE;
  bWrite      : BOOL := FALSE;
  bKinStop    : BOOL := FALSE;
  bStarted    : BOOL := FALSE;
  bStopped    : BOOL := FALSE;
  uiState     : UINT := IDLE;

  fbOpenDirectCyclicChannel: ML_RobotOpenDirectCyclicChannel;
  fbWriteCyclicChannel     : ML_RobotWriteCyclicChannel;
  fbReadCyclicChannel      : ML_RobotReadCyclicChannel;
  fbGroupStop             : MC_GroupStop;
  fbRobotTransformPoint    : ML_RobotTransformPoint;
END_VAR
VAR CONSTANT
  IDLE :UINT := 0;
  INIT :UINT := 10;
  OPEN :UINT := 20;
  SYNC :UINT := 30;
  STOP :UINT := 40;
  RESET :UINT := 50;
END_VAR

```

Program part

```

(* start/stop the cyclical command value specification *)
IF Enable = TRUE AND bStarted = FALSE THEN
  uiState := INIT;
  bStarted := TRUE;
  bStopped := FALSE;
ELSIF Enable = FALSE AND bStopped = FALSE THEN
  IF KinData[SlaveGroup.GroupNo].Kis_GroupMoving THEN
    uiState := STOP;
  ELSE
    uiState := RESET;
  END_IF
  bStopped := TRUE;
  bStarted := FALSE;
  Error := FALSE;
END_IF

(* state machine *)
CASE uiState OF
  INIT: (* initialization *)
    bWrite := TRUE;
    IF fbWriteCyclicChannel.InOperation AND fbRobotTransformPoint.InOperation THEN
      uiState := OPEN;
    END_IF

  OPEN: (* open cyclical channel *)
    bOpen := TRUE;
    IF fbOpenDirectCyclicChannel.Active THEN
      InOperation := TRUE;
      uiState := SYNC;
    END_IF

  SYNC: (* write cyclical values *)
    IF fbOpenDirectCyclicChannel.Done THEN
      InOperation := FALSE;
    END_IF

  STOP: (* close cyclical channel *)
    bKinStop := TRUE;
    IF fbGroupStop.Done THEN
      uiState := RESET;
    END_IF

  RESET: (* reset internal variables *)
    bOpen := FALSE;
    bWrite := FALSE;
    bKinStop := FALSE;
    InOperation := FALSE;
    uiState := IDLE;

```


END_CASE

```
(* functions blocks for cyclical command value specification *)
fbRobotTransformPoint(
    Enable      := bWrite,
    InPoint     := KinData[MasterGroup.GroupNo].arPositionJointCoord_i,
    InCoordSystem := CS_AXES_COORDINATES,
    OutCoordSystem:= CoordSystem,
    AxisGroup   := SlaveGroup);

fbWriteCyclicChannel(
    Enable      := bWrite,
    Value       := fbRobotTransformPoint.OutPoint,
    CoordSystem:= CoordSystem,
    CyclicMode  := CYCLIC_MODE_POS_ABS,
    AxisGroup   := SlaveGroup);

fbOpenDirectCyclicChannel(
    Execute      := bOpen,
    Deceleration := Deceleration,
    Jerk         := Jerk,
    SetMode      := SetMode,
    BlendingRadius:= BlendingRadius,
    AxisGroup    := SlaveGroup);

fbReadCyclicChannel(
    Enable      := fbWriteCyclicChannel.InOperation,
    AxisGroup:= SlaveGroup);

(* function block for stopping of the cyclical command value specification *)
fbGroupStop(
    Execute      := bKinStop,
    Deceleration:= Deceleration,
    Jerk         := Jerk,
    AxisGroup    := SlaveGroup);

(* error handling *)
IF fbOpenDirectCyclicChannel.Error THEN
    Error      := TRUE;
    ErrorID    := fbOpenDirectCyclicChannel.ErrorID;
    ErrorIdent := fbOpenDirectCyclicChannel.ErrorIdent;
    uiState    := RESET;
ELSIF fbRobotTransformPoint.Error THEN
    Error      := TRUE;
    ErrorID    := fbRobotTransformPoint.ErrorID;
    ErrorIdent := fbRobotTransformPoint.ErrorIdent;
    uiState    := RESET;
ELSIF fbWriteCyclicChannel.Error THEN
    Error      := TRUE;
    ErrorID    := fbWriteCyclicChannel.ErrorID;
    ErrorIdent := fbWriteCyclicChannel.ErrorIdent;
    uiState    := RESET;
ELSIF fbReadCyclicChannel.Error THEN
    Error      := TRUE;
    ErrorID    := fbReadCyclicChannel.ErrorID;
    ErrorIdent := fbReadCyclicChannel.ErrorIdent;
    uiState    := RESET;
ELSIF fbGroupStop.Error THEN
    Error      := TRUE;
    ErrorID    := fbGroupStop.ErrorID;
    ErrorIdent := fbGroupStop.ErrorIdent;
    uiState    := RESET;
END_IF
```

Input	Description
Enable	Enabling the synchronization
Deceleration	Deceleration for desynchronization
Jerk	Jerk for desynchronization
CoordSystem	Coordinate system
SetMode	Blending for desynchronization
BlendingRadius	Blending radius for desynchronization

Application examples

MasterGroup	Master kinematics
SlaveGroup	Slave kinematics

Tab. 8-1: Input description

Output	Description
InOperation	Kinematics is synchronized
Error	Kinematics is not synchronized due to an error
ErrorID	Brief note on error cause
ErrorIdent	Detailed error information

Tab. 8-2: Output description

With a rising edge at the "Enable" input, the current absolute position of the master kinematics is written to the cyclic command value channel in the specified coordinate system. By subsequently opening the cyclic command value channel, the kinematics is set to the state "Group Moving" and approaches the command value previously written in a cycle. Thus, both kinematics should be positioned at the same position in the selected coordinate system before the synchronization so that the slave kinematics does not "jump" at this position!

If only the master kinematics is moved, the slave kinematics follows. The "InOperation" output signalizes that the kinematics are synchronized. If a synchronization is prevented due to an error, this is indicated in the "Error" output. For detailed information about the error, refer to the "ErrorID" and "ErrorIdent" outputs.

Via the "Deceleration", "Jerk", "SetMode" and "BlendingRadius" inputs, the desynchronization and thus leaving of the cyclic command value specification is determined. If a motion command (e.g. Move Direct Absolute) is issued for the slave kinematics during the synchronous phase, the slave kinematics is decelerated with the specified deceleration and jerk. Moreover, if a blending radius is specified and if the blending type is set to "SM_VIA", the new motion command is blended if a PTP motion command has been issues (see [Blending possibilities with various motion types, page 59](#)). If the desynchronization is completed, the "InOperation" output is reset.

8.10 Sequential programming with the kinematic 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.

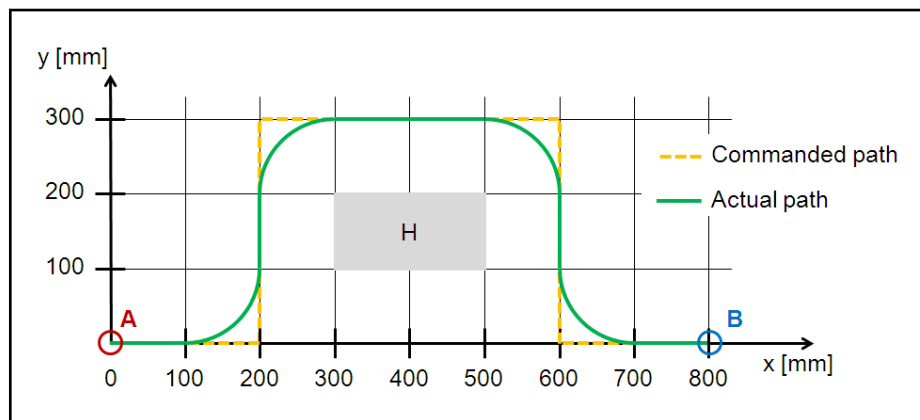


Fig. 8-8: Sequential programming with the kinematics interface - motion path

Application examples

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;
```

Application examples

```

(* 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;

(* 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.

8.11 Registration of a user-defined transformation using the KinematicInterface

If a user-defined transformation is used, i.e. the transformation number 1100, it is necessary to register the transformation in the PLC prior to phase switching to **BB**. The registration can be easily implemented via the KinematicInterface. For that purpose, the transformation function block needs

Application examples

only to be created at the "Admin.KinTransform" input of the KinematicInterface.

The following command starts the transformation registration ("fbTransformation" is equivalent to an instance in the transformation function block):

```
arKinCtrl_gb[1].Admin.KinTransformPLC:= fbTransformation;
```



The registration is only possible in parameterization mode **P2**. If a transformation is created at the input of the KinematicInterface in a different phase, then the registration is not started until Phase 2 is reached.



If the KinematicInterface is initially registered with a user-defined transformation, then the control displays the error "Transformation not registered". The cause of this can be explained in that the IMC interface is attempting to switch into the operating mode. The error can be acknowledged after the transformation has been registered successfully.

If transformation registration was successful in the PLC, then this is signaled via the status output

```
arKinStatus_gb[1].Admin.KinTransformPLCDone
```


9 Parameter

9.1 Parameters, 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 on page 419](#) are created to manage **all** kinematic data of a control:
- The following [Kinematic-related parameters on page 420](#) are created to manage data of **a** kinematics in the control. Each parameter exists once in each kinematics

9.2 Axis-related parameters

9.2.1 Axis-related parameters, general information

Limit values and axis-related data to support a Robot Control V2 kinematic are stored in A-parameters of the respective axis.

9.2.2 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

9.2.3 A-0-0029, Position polarities

Function For real axes applies:

The first four bits of this parameter can invert the polarities of the specified position data.

The polarities are switched outside the controlled system (that is at the input and output of the controlled system).

The parameter is applied to the parameter S-0-0055 of the drive device.



The position polarity has to be specified upon the first axis commissioning before creating the machine zero point reference for measuring systems, since other actual position values result when changing the polarity.

Parameter

For rotary (rotating) motors applies:

- Rotation of the motor output shaft in clockwise direction at positive position command value difference and positive polarity.

For linear motors applies:

- Motion of the primary component towards the cable connection at positive position command difference and positive polarity.

See also: [A-0-0048](#) and [A-0-0049](#)

Function The software position limit values "[A-0-0030](#), Position limit value, positive" or "[A-0-0031](#), Position limit value, negative" are activated or deactivated using bit 4 of this parameter.

Structure

Bit	Name/function
0	Position command value 0: Positive polarity 1: Negative polarity
1	Position command value, additive 0: Positive polarity 1: Negative polarity
2	Actual position value 1 0: Positive polarity 1: Negative polarity
3	Actual position value 2 0: Positive polarity 1: Negative polarity
4	Position limit value 0: Not active, 1: Active.
15..5	Reserved

9.2.4 A-0-0030, Position limit value, positive

Function A permitted traversing range of an axis can be defined by the position limit values. This parameter limits the maximum allowed positive actual position value (positive traversing direction). The first actual position value, which is not permitted anymore, is entered.

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 "A-0-2200, Command position" of the axis is set on or outside the positive position limit value, the error bit becomes TRUE in "A-0-0021, Axis status, bit 9" TRUE and generates the error "F2112053, Target position outside the traversing range".

If "A-0-0100, Actual position value" reaches or exceeds "[A-0-0030](#), Position limit value, positive" or "[A-0-0031](#), Position limit value, negative", the axis sets the error bit in "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".

9.2.5 A-0-0031, Position limit value, negative

Function A permitted traversing range of an axis can be defined by the position limit values. This parameter limits the maximum allowed negative actual position value (negative traversing direction). The first actual position value, which is not permitted anymore, is entered.

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 "A-0-2200, Command position" of the axis is set on or outside the negative position limit value, the error bit becomes TRUE in "A-0-0021, Axis status, bit 9" TRUE and generates the error "F2112053, Target position outside the traversing range".

If "A-0-0100, Actual position value" reaches or exceeds "A-0-0030, Position limit value, positive" or "A-0-0031, Position limit value, negative", the axis sets the error bit in "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".

9.2.6 A-0-0032, Velocity limit value, positive

Function The parameter describes the maximum admissible velocity in positive direction.

Use If the parameter "A-0-0102, Actual velocity value" exceeds the parameter "A-0-0032, Velocity limit value, positive" and if the parameter value is smaller than the content of "A-0-0035, Velocity limit value, bipolar", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112061, Command velocity higher than positive limit value".

9.2.7 A-0-0033, Velocity limit value, negative

Function The parameter describes the maximum admissible velocity in negative direction.

Use If the parameter "A-0-0102, Actual velocity value" falls below the parameter "A-0-0033, Velocity limit value, negative" and if the parameter value is higher than the content of "A-0-0035, Velocity limit value, bipolar", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112062, Command velocity lower than negative limit value".

9.2.8 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 "A-0-0021, Axis status, bit 8" and generates the warning "E0112039, Maximum acceleration exceeded".

9.2.9 A-0-0035, Velocity limit value, bipolar

Function The parameter symmetrically describes the maximum admissible velocity in both directions.

Use If the parameter "A-0-0102, Actual velocity value" exceeds the parameter "A-0-0035, Velocity limit value, bipolar" and if the parameter value is smaller than the content of "A-0-0032, Velocity limit value, positive", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates either the warning "E0112061, Command velocity higher than positive limit value" or "E0112062, Command velocity smaller than negative limit value".

Parameter

9.2.10 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".

9.2.11 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".

9.2.12 A-0-0039, Torque/force limit value, positive

Function The parameter describes the maximum admissible force/torque in positive direction.



As of drive firmware HDx-20 (hydraulic drive), the parameter "A-0-0039, Torque/force value, positive" is synchronized to "S-0-0550, Torque/force command value, positive 2".

- Use** If the parameter "A-0-0110, Actual torque/force value" exceeds the parameter "A-0-0039, Torque/force limit value, positive" and if the parameter value is smaller than the content of "A-0-0038, Torque/force limit value, bipolar", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112057, Torque/force limit value positively exceeded".

9.2.13 A-0-0040, Torque/force limit value, negative

- Function** The parameter describes the maximum admissible force/torque in negative direction.



As of drive firmware HDx-20 (hydraulic drive), the parameter "A-0-0040, Torque/force value, negative" is synchronized to "S-0-0551, Torque/force command value, negative 2".

- Use** If the parameter "A-0-0110, Actual torque/force value" falls below the parameter "A-0-0040, Torque/force limit value, negative" and if the parameter value is higher than the content of "A-0-0038, Torque/force limit value, bipolar", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112058, Negatively fallen below torque/force limit value".

9.2.14 A-0-0048, Velocity polarities

- Function** In this parameter, the polarity of the velocity data is switched with regard to the application.

The polarity is not switched within, but outside (at input and output) a controlled system.

Structure

Bit	Name/function
0	Velocity command value 0: Positive polarity 1: Negative polarity
1	Position command value, additive 0: Positive polarity 1: Negative polarity
2	Actual velocity value 0: Positive polarity 1: Negative polarity
15..3	Reserved



Bit 1 and bit 2 are copies of bit 0.

Changes only apply in bit 0. Different settings of the individual bits are not permitted!

For **rotary (rotating) motors** applies: Rotation of the motor output shaft in clockwise direction causes a positive actual velocity value (positive polarity).

For **linear motors** applies: Motion of the primary component towards the cable connection side causes a positive actual velocity value and (positive polarity).

Parameter

9.2.15 A-0-0049, Torque-/force polarities

Function In this parameter, the polarity of the specified torque/force data is inverted with regard to the application.

The polarity is switched outside the controlled system (that means at the input and the output of the controlled system).

Structure

Bit	Name/function
0	Torque command value 0: Positive polarity 1: Negative polarity
1	Torque command value, additive 0: Positive polarity 1: Negative polarity
2	Actual torque value 0: Positive polarity 1: Negative polarity
15..3	Reserved



Bit 1 and bit 2 are copies of bit 0.

Changes only apply in bit 0. Different settings of the individual bits are not permitted!

For **rotary (rotating) motors** applies: Rotation of the motor output shaft in clockwise direction at positive torque and positive polarity.

For **linear motors** applies: Motion of the primary component towards the cable connection side at positive force and positive polarity.

9.2.16 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.

The parameter unit is specified by "A-0-0053, Scaling type for acceleration data".

See also Functional Description: "Error Reaction"

9.2.17 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.

9.2.18 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.

9.2.19 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"

9.3 Control-related parameters

9.3.1 Control-related parameters, general information

The following C-parameters are created to manage **all** kinematic data of one control:

9.3.2 C-0-0123, Maximum number of kinematics

Function This parameter indicates the maximum number of kinematics.

9.3.3 C-0-2480, Configured kinematic number

Function This parameter indicates the number of the currently configured kinematics.

9.3.4 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.

9.3.5 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:

Parameter

Element	Value	Comment
1	1	Kinematics 1 is configured
2	2	Kinematics 2 is configured
3	3	Kinematics 3 is configured

9.3.6 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
2	RobotControl V2 kinematics configured

Use Writing on the parameter "[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

Three kinematics with the logic numbers 1, 2 and 4 are to be configured.

[C-0-2484](#) indicates:

Element	Value	Comment
1	2	Kinematics 1 is configured
2	2	Kinematics 2 is configured
3	0	No kinematics 3 configured
4	2	Kinematics 4 configured

9.4 Kinematic-related parameters

9.4.1 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.

9.4.2 K-0-0001, Logic kinematic number

Function This parameter shows the actual logical kinematics number.

9.4.3 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:

- A...Z, a...z, 0..9, ' _'
- first character must be a letter,
- "Robi_1" is identical to "ROBI_1".

9.4.4 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.

9.4.5 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.

9.4.6 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.

9.4.7 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.

9.4.8 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.

Parameter

Value	Comment
0	No axis configured
8	Positioning axis configured
16	Orientation axis configured
32	Belt axis configured

Tab. 9-1: Valid values of an axis configuration

Value	Comment
0	Individual control
1-64	Control number in C2C link

Tab. 9-2: 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.

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
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).

9.4.9 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.

9.4.10 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:

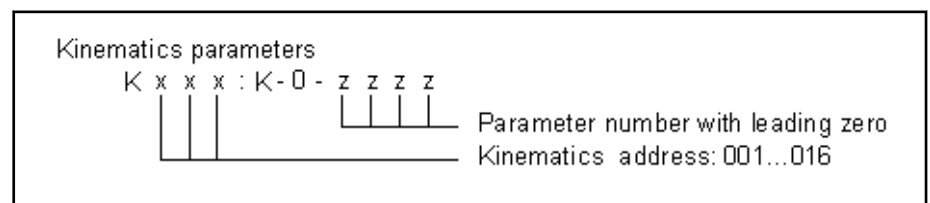


Fig. 9-1: K-parameter structure

Parameter

Example:

K001: [K-0-0002](#) - Kinematics 1; parameter [K-0-0002](#).

9.4.11 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.

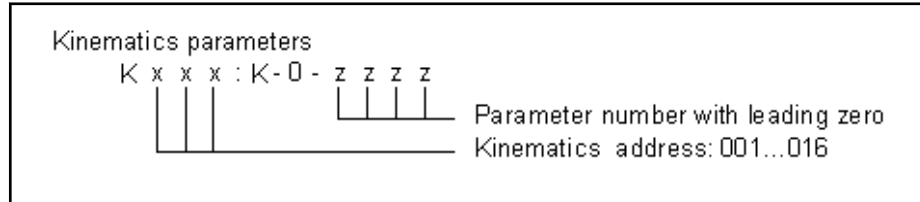


Fig. 9-2: K-parameter structure

Use K001: [K-0-0008](#) - Kinematics 1; parameter [K-0-0008](#).

9.4.12 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".

9.4.13 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](#).

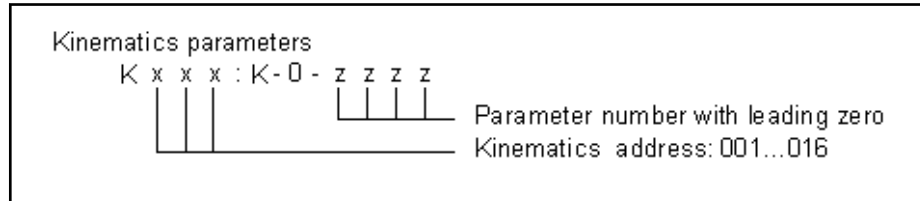


Fig. 9-3: Structure of a K-parameter

Use K001: [K-0-0008](#) - Kinematics 1; parameter [K-0-0008](#).

9.4.14 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.

9.4.15 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. 9-3: K-0-0019 Kinematic type

9.4.16 K-0-0020, Kinematic diagnostic message

Function These parameters contain the current diagnostic message for the kinematic (message, warning or error).

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.9= 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.

9.4.17 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. The information does not include the state of the Kinematic 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	Errors	Kin_Error
	10	Homing required; not all axes are referenced.	Kin_NotHomed
	11	Blending	Kin_Blending
	12	Transformed machine coordinates are invalid	Kin_TrafoValuesInvalid
	14	All axes are ready for operation (AH or AF)	Kin_DrivesOn
	17	Error reaction enabled	Kin_ErrorReaction
	20	Fully belt-synchronous	Kin_FullyBeltSynchronous
	21	BeginLength reached	Kin_BeginLengthReached
	22	Waiting for belt-synchronous start condition	Kin_WaitBeltStartCondition

Parameter

Bit	Content	Bit name
28	Spline is calculated for blending motion	
29	Spline is valid	

Tab. 9-4: K-0-0021 Kinematic status

Use Bit 5: All axes of the kinematics are stopped

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

Partially or fully-synchronous traversing to belt.

Bit 8: Warning

A warning is present.

Bit 9: Error

An error is present.

Bit 10: Homing required

At least one kinematic axis is not referenced.

Bit 11: Blending

Blending is active.

Bit 12: Transformed machine coordinates are invalid

The current machinecoordinates for the kinematics indicated in KinData are invalid. For example, this is the case if not all of the axes are assigned to the kinematics or if there is an error in the transformation.

Bit 14: All axes are ready for operation

All kinematic axes are ready for operation (AH or AF).

Bit 17: Error reaction active

Currently, an error reaction is active.

Bit 20: Fully belt-synchronous

Fully-synchronous traversing to belt.

Bit 21: BeginLength reached (belt-synchronous)

The kinematics is belt-synchronous and "Begin Length" was exceeded.

Bit 22: Waiting for start condition (belt-synchronous)

Waiting for the belt-synchronous start condition.

Bit 28: Calculating spline for blending motion

Currently, a spline for the blending motion is calculated.

Bit 29: Spline is valid

The spline calculated for the blending motion is valid. The bit remains active until the spline has been travelled.

9.4.18 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](#).

Parameter

	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

9.4.19 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.

9.4.20 K-0-0028, Velocity limit auxiliary axes

Function An additional limit value for the velocity of each auxiliary axis can be specified in this parameter.

According to the motion type, the parameter has the following effects:

- In the case of Linear and Circular, the minimum is generated from the velocity limit value in this parameter and the single-axis limit value of the auxiliary axis. This minimum is used to limit the axis motion.
- In the case of point to point, only the single-axis limit value of the auxiliary axis is used to limit the axis motion.



Please refer to the transformation scheme for whether an auxiliary axis is used.



Since this parameter is used in the block preparation, a new value only has an effect on commands that have not yet been prepared by the block preparation.

Structure

Element	Comment
1	1st auxiliary axis of the transformation scheme (if available)
2	2nd auxiliary axis of the transformation scheme (if available)
...	...

9.4.21 K-0-0029, Acceleration limit auxiliary axes

Function An additional limit value for the acceleration of each auxiliary axis can be specified in this parameter.

According to the motion type, the parameter has the following effects:

- In the case of Linear and Circular, the minimum is generated from the acceleration limit value in this parameter and the single-axis limit value of the auxiliary axis. This minimum is used to limit the axis motion.
- In the case of point to point, only the single-axis limit value of the auxiliary axis is used to limit the axis motion.

Parameter



Please refer to the transformation scheme for whether an auxiliary axis is used.



Since this parameter is used in the block preparation, a new value only has an effect on commands that have not yet been prepared by the block preparation.

Structure

Element	Comment
1	1st auxiliary axis of the transformation scheme (if available)
2	2nd auxiliary axis of the transformation scheme (if available)
...	...

9.4.22 K-0-0030, Jerk limit auxiliary axes

Function An additional limit value for the jerk of each auxiliary axis can be specified in this parameter.

According to the motion type, the parameter has the following effects:

- In the case of Linear and Circular, the minimum is generated from the jerk limit value in this parameter and the single-axis limit value of the auxiliary axis. This minimum is used to limit the axis motion.
- In the case of point to point, only the single-axis limit value of the auxiliary axis is used to limit the axis motion.



Please refer to the transformation scheme for whether an auxiliary axis is used.



Since this parameter is used in the block preparation, a new value only has an effect on commands that have not yet been prepared by the block preparation.

Structure

Element	Comment
1	1st auxiliary axis of the transformation scheme (if available)
2	2nd auxiliary axis of the transformation scheme (if available)
...	...

9.4.23 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](#).

9.4.24 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 reset if fallen below.

9.4.25 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. In Motion type PTP the jerk is not limited by this parameter.
 If the parameter value is reached, the warning E02C08CD is reported and reset if fallen below.

9.4.26 K-0-0037, Jerk for move commands during deccelearation phase

Function This parameter specifies the jerk for the deceleration phase. The percentage of the programmed jerk is used for the deceleration phase.

9.4.27 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.

9.4.28 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](#).

9.4.29 K-0-0042, Offset of BCS 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)

Parameter

Element	Description
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.

9.4.30 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).

9.4.31 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.

Parameter

- 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).

9.4.32 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](#))

Parameter

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).

9.4.33 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.

- 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).

9.4.34 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

9.4.35 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.

9.4.36 K-0-0090, Taskpriority preparation

Function This parameter adjusts the priority of block preparation.
Block preparation task priority

- 20 - Block preparation is preferably processed before communication
- 21 - The priority of block preparation is on the level of communication, and block preparation is processed equally with communication (Round Robin)
- 22 - Default: Block preparation is processed after communication



This parameter should only be adjusted by experienced users, since the system behavior has been designed for the default value.

9.4.37 K-0-0095, ID of movement

Function This parameter indicates the current motion identification number (**MoveID**).
 The integer part references to the number of the motion instruction.
 The decimal part indicates the motion progress in %.
MoveID can manually only be set to an integer value.



MoveID is not increased for **Jog**.
MoveID should not be tested with the "==" operator.

9.4.38 K-0-0096, Peak value of command buffer size

Function This parameter indicates the maximum value of the command buffer size reached. The value can be reset by writing smaller values.

9.4.39 K-0-0097, Maximum command buffer size

Function This parameter indicates the maximum number of commands that can be simultaneously stored in the block buffer.

9.4.40 K-0-0098, Current command buffer size

Function This parameter indicates the current number of commands in the block buffer.

Parameter

9.4.41 K-0-0099, Kinematic configuration

Function
Structure

This parameter configures the behavior of the kinematics.

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 be active simultaneously in BLOCK_SLOPE
5	Disables all scaling checks of the kinematic axes
6	Enables the adaptive interpolation for belt-synchronous motions
7	Enables the commanding with the complete position vector (X,Y,Z)
8	Enables the commanding with the complete orientation vector (A,B,C)
9..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 module value. Ensure that the belt operating range can be max. belt module/2.

Bit 2: Enables the complete radius to blend 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, simultaneous synchronous operation on several belts is possible. The belts have to be linearly independent, i.e. twisted in an angle of 90 degrees with respect to one another.

Bit 4: Enables the option that more than two commands can be active simultaneously in BLOCK_SLOPE

If bit 4 is set, more than two commands can be active in BLOCK_SLOPE simultaneously. Thus, more than 2 commands are blended. A third command is also started if it is in the blending radius and if the first command has not been completed yet. Please note that the resulting path is undefined due to blending of more than two commands. For example, two blocks working in opposite directions can cancel each other.

Bit 5: Disables all scaling checks of the kinematic axes

If bit 5 is set, all scaling checks of the kinematic axes are disabled. The axis scaling does not have to match the transformation. It is possible to use non-translatory scaled axes for linear belts.

⚠ CAUTION**Unexpected axis motions**

Disabling the scaling check can result in unexpected axis motions.

Bit 6: Enables the adaptive interpolation for belt-synchronous motions

If bit 6 is set, belt-synchronous motions, adaptively interpolated if required (that is waiting and standstill times of the kinematics until the fully synchronous end point of a motion of the belt-synchronous workspace is reached) can be omitted. The fully synchronous end point is reached by the kinematics at the same time as soon as it reached the belt-synchronous workspace. The interpolation of the belt-synchronous motion depends on the fully synchronous end point.

Bit 7: Enables the commanding with the complete position vector (X,Y,Z)

If bit 7 is set, points with all position coordinates can be commanded. In this mode, the vector is not restricted anymore to the coordinates required by the kinematics.

Bit 8: Enables the commanding with the complete orientation vector (A,B,C)

If bit 8 is set, points with all degrees of freedom can be commanded. This allows to twist kinematics in space beyond a transformation.

9.4.42 K-0-0100, Current position, Cartesian Machine Coordinate System

Function The parameter indicates the current position of the kinematics in the machine coordinate system (MCS).
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](#)).

9.4.43 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](#)).

9.4.44 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.

9.4.45 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.

9.4.46 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.

Parameter

9.4.47 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](#)).

9.4.48 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.

9.4.49 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.

9.4.50 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 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.

9.4.51 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.

9.4.52 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.

9.4.53 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.

9.4.54 K-0-0241, Feedrate override control word

Function Structure

This parameter controls the feed rate override.

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.

9.4.55 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.

9.4.56 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%.

Parameter



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.

9.4.57 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.

9.4.58 K-0-0815, Odometer

- Function** This parameter shows the traversed distance of the "Tool Center Points" (TCP) in kilometers.

9.4.59 K-0-1000, Command: Load basic kinematic parameters

- Function** This parameter is used as a command to load the basic kinematic parameters.
- Use** *From IndraWorks MLC:*
- Wizard "initial commissioning" in the **context menu of the kinematics** ► **Initial Commissioning** ► **Step 1: Load Basic Parameters...**
 - "Load Basic Parameters" button in the **context menu of the kinematics** ► **Parameter** ► **Load Basic Parameters...**
- From the field bus:*
- Enter the binary value "0000000000000011" (=16#0003) into the parameter ([K-0-1000](#)).
This activates the loading of the basic parameters of the kinematics.
 - After the loading , the MLC automatically resets the parameter to "16#00".

9.4.60 K-0-2100, List of all backup operation K-parameters

- 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:

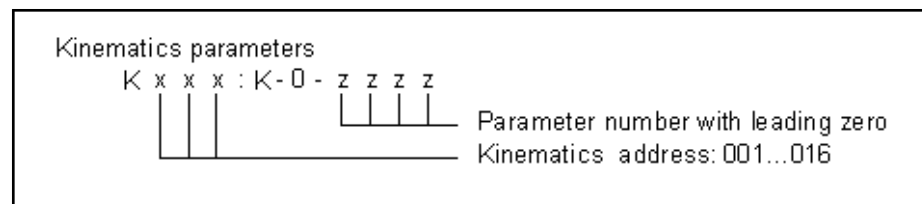


Fig. 9-4: K-parameter structure

Parameter

Example:

K001: [K-0-0002](#) - Kinematics 1; parameter [K-0-0002](#).

10 Diagnostics

The IndraMotion MLC diagnostic system supports working with kinematics on the control.

All messages, warnings and errors are described and contained in

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

11 Connecting third-party robotic systems

11.1 General information

In the field of general automation, special robot applications are frequently required. These have to be integrated completely into the automation process of an automation unit. To allow a simple engineering, Bosch Rexroth provides the Motion Logic control (MLC) combining the advantageous of specialized robot controls with the versatile Bosch Rexroth multi-axis controls. This creates a continuous "look and feel" and allows to program and parameterize the complete unit with one engineering tool (IndraWorks). Thus, the effort with regard to engineering and commissioning can be reduced significantly.

11.2 uniVAL drive

11.2.1 Overview

uniVAL by the company Stäubli provides a ready-to-connect solution to connect all Stäubli robots to blended multi-axis controls via the standardized real-time bus Sercos. This solution is already known on the automation market. Thus, Stäubli robots can be programmed with proven Bosch Rexroth control systems and parameterized via IndraWorks.

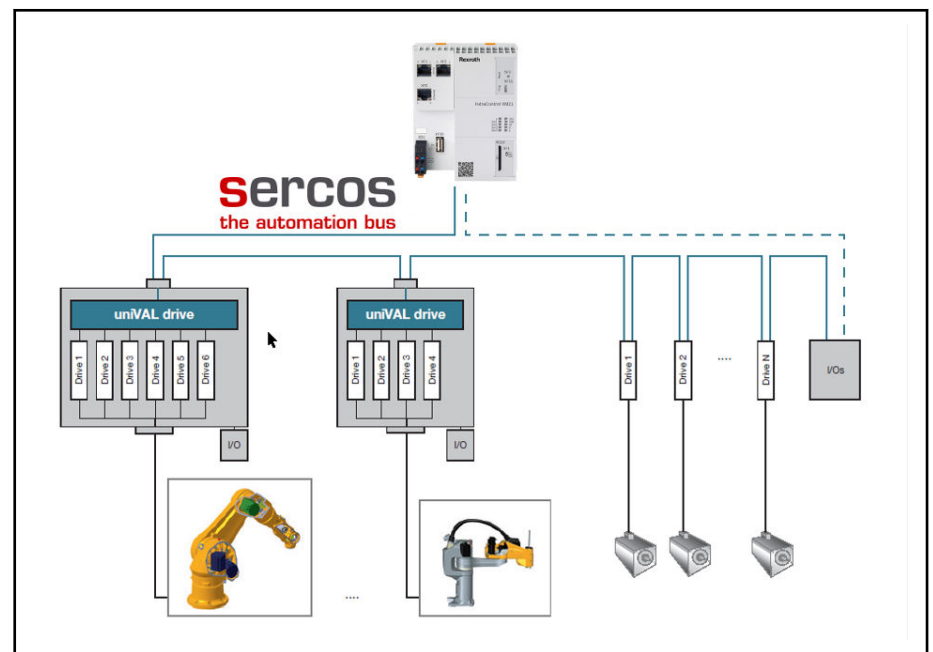


Fig. 11-1: Schematic topology structure of the Bosch Rexroth control technology with uniVAL drive

11.2.2 Creating axes

To operate Stäubli robots at an MLC, configure the respective drives of the kinematics in IndraWorks first.

Proceed as follows:

- Connect the MLC to the respective robot control of the company Stäubli using a Sercos bus, refer to [fig. 11-1 "Schematic topology structure of the Bosch Rexroth control technology with uniVAL drive" on page 443](#) (the Stäubli robot control has to be provided with the respective communication interface)

Connecting third-party robotic systems

- The drives are integrated as generic SercosDrive [also refer to chapter 6.3.1 "General information", documentation "IndraLogic XLC IndraMotion MLC 14VRS Functional Description" (DOK-XLCMLC-FUNC****V14-APRS-EN-P)]
- Creating axes in IndraWorks [also refer to "Creating a real axis", documentation "IndraLogic XLC IndraMotion MLC 14VRS Functional Description" (DOK-XLCMLC-FUNC****V14-APRS-EN-P)]

11.2.3 Configuring axes

The configuration of the individual Stäubli robots is manufacturer-specific. Thus, refer to the respective documentation by Stäubli. For information, refer to the respective internet sites of the manufacturer (www.staubli.com/en/robotics/).



- The axes have to be configured using the required A-parameters. Wherever required, they are synchronized in the respective S-parameters
 - Note the manufacturer specifications given by Stäubli when specifying scalings for the individual axes, as explicit value specifications are required for an error-free operation
-

11.2.4 Configuring the kinematic transformation

The Robot Control provides different options to configure the transformations. On how to add a kinematics or how to configure transformations, refer to the chapter [chapter 4 "Transformation schemes" on page 107](#).

12 Upgrading from Robot Control V1 to Robot Control V2

12.1 Overview

When upgrading from Robot Control V1 to Robot Control V2, the RCL or IEC program has to be converted and the IndraWorks project configuration has to be adjusted.

This conversion instructions briefly describes the first steps. However, these steps depend on whether an RCL or an IEC program is used.

- Based on an RCL program, create a function, a function block or a program in which the commands and program sequences of the RCL program are integrated in. Whether a function, a function block or a program is generated depends on the application. The task in which the function, the function block or the program is called, differs.

For more information, refer to chapter [Converting RCL commands, page 446](#) and [Converting RCL programs, page 451](#).

- If the program has been created on the basis of IEC, the upgrade effort is reduced. Interventions in the program sequence are only required if a belt synchronization is used or in case of function blocks only valid for Robot Control V1.

For more information, refer to chapter [Converting IEC commands, page 455](#).

Irrespective of whether an RCL or an IEC program is used, adjustments in an IndraWorks project have to be made.

For more information, refer to the following chapter.

12.2 Adjusting the IndraWorks project

It is recommended to backup the parameters prior to a Robot Control V2 upgrade. Thus, a parameterization of the original kinematics can be restored easily and quickly following an upgrade.



Due to safety reasons, the kinematics **must** be commissioning again after the parameters have been restored (checking the axis lengths, axis assignments, scalings, etc.)!

Adjust the version in the kinematics configuration for the Robot Control V2 upgrade. Right-click on the kinematics to open the property dialog and select the "Configuration" tab (see following figure).

Upgrading from Robot Control V1 to Robot Control V2

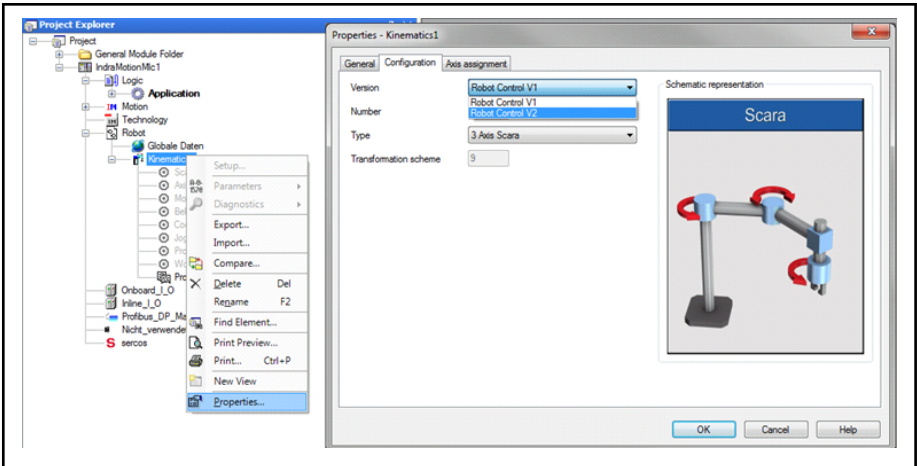


Fig. 12-1: Robot Control V2 kinematic configuration

Due to this version adjustment, the existing kinematics is deleted internally and a new kinematics is created. Therefore, the kinematics parameters backed up earlier have to be restored.



All existing RCL programs are deleted!

User-defined transformation

The kinematic type "user-defined" is not supported anymore by Robot Control V2. In the "Configuration" tab in the property dialog of the kinematics, the kinematics can be selected under "Type". Thus, select "2 axis delta (7)" for a kinematic number 1007. Transformations defined in the PLC can still be used. Use the "user-defined transformation" or the "extended user-defined transformation". With the latter kinematic type, new functionalities can be used for user-defined transformations only available under Robot Control V2.

12.3 Converting RCL commands

A RCL programming language contains all key words required for kinematics commands.

The following sections explain which PLCopen function blocks replace these key words and how the keyword functionality can be reproduced using PLCopen function blocks.

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..MB_MAX_KIN_AXIS_NUMBER] OF REAL; (using the machine coordinate system)

Upgrading from Robot Control V1 to Robot Control V2

RCL	PLCopen
JC_POINT: arPoint	arPoint: ARRAY [1..MB_MAX_KIN_AXIS_NUMBER] OF REAL; (using 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 in the RCL saves the point in a point file. When using the PLCopen function block, call the TeachIn functionality (ML_GroupWritePoint and ML_GroupSavePointFile).

Functions

RCL and PLCopen functions have largely the same syntax or may be mapped using simple function blocks. The following table compares 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
SQRT(49)	SQRT(49) -> 7
ABS(-5)	ABS(-5) -> 5
TRUNC(1.9)	TRUNC(1.9) -> 1
ORD('g')	IL_AsciiToByte('g') -> 103
CHR(103)	IL_ByteToAscii(INT_TO_BYTE(103) -> g
ROUND(7.81)	REAL_TO_INT(7.81) -> 8
13 MOD 5	13 MOD 5 -> 3
END_OF_FILE	Uses function block SysFileEOF

Upgrading from Robot Control V1 to Robot Control V2

ASSIGN	Not required
INT_ASC	See program excerpt at the end of the table
ASC_INT	See program excerpt at the end of the table
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 ML_GroupSetSafeZone
JC(...), WC(...)	Corresponds to a transformation of coordinates ML_RobotTransformPoint

No equivalent functions are available for the "INT_ASC" and "ASC_INT" functions in PLCopen. Thus, the user has to create user-defined functions.

Example of the "INT_ASC" function:

Declaration

```

VAR
  iValue: INT := 0;
  strValue: STRING(255) := '';
  strPartofValue: STRING(255) := '';
  arStrValue: ARRAY [1..10] OF STRING := [10('')];
  iPosition: INT := 1;
  iArraySize: INT := 0;
END_VAR

```

Program part

```

strValue := INT_TO_STRING(iValue);
iArraySize := SIZEOF(arStrValue)/81; //81 Bytes per String element
FOR iPosition := 1 TO iArraySize DO
  strPartofValue := MID(strValue,1,iPosition);
  arStrValue[iPosition] := strPartofValue;
END_FOR

```

Example of the "ASC_INT" function:

Declaration

```

VAR
  iValue: INT := 0;
  strValue: STRING(255) := '';
  strPartofValue: STRING(255) := '';
  arStrValue: ARRAY [1..10] OF STRING := [10('')];
  iPosition: INT := 1;
  iArraySize: INT := 0;
  iFactor: INT := 0;
END_VAR

```

Program part

```

strValue := '';
iArraySize := SIZEOF(arStrValue)/81; //81 Bytes per String element
FOR iPosition := 1 TO iArraySize DO
  strPartofValue := arStrValue[iPosition];
  IF STRING_TO_INT(strPartofValue) <> 0 THEN
    strValue := CONCAT(strValue, strPartofValue);
  ELSIF strPartofValue = '-' THEN
    iFactor := -1;
  ELSIF strPartofValue = '+' THEN
    iFactor := 1;
  END_IF
END_FOR
iValue := STRING_TO_INT(strValue) * iFactor;

```

Upgrading from Robot Control V1 to Robot Control V2

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	Can be mapped using ML_SwitchOnPath
SPC_FCT:21=belt_kind	Not required
SPC_FCT:23=clock_date	ML_GetSystemTimeUTC or IL_SysTimeDate
SPC_FCT:24=sys_count	Using parameter C-0-0423
SPC_FCT:28=belt_set	The belt value is reset to 0 automatically at 10 000 for the axis module value depending on Bit 1 of parameter K-0-0099
SPC_FCT:53=belt_area	Corresponds to the "BeginLength" and "TotalLength" inputs of the ML_SetBeltConfiguration function blocks
SPC_FCT:59=getBlendMode	Using the parameter K-0-0099 , Bit 2
SPC_FCT:60=setBlendMode	Reading the parameter K-0-0099 , Bit 2

Boundary conditions of a motion 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:

Upgrading from Robot Control V1 to Robot Control V2

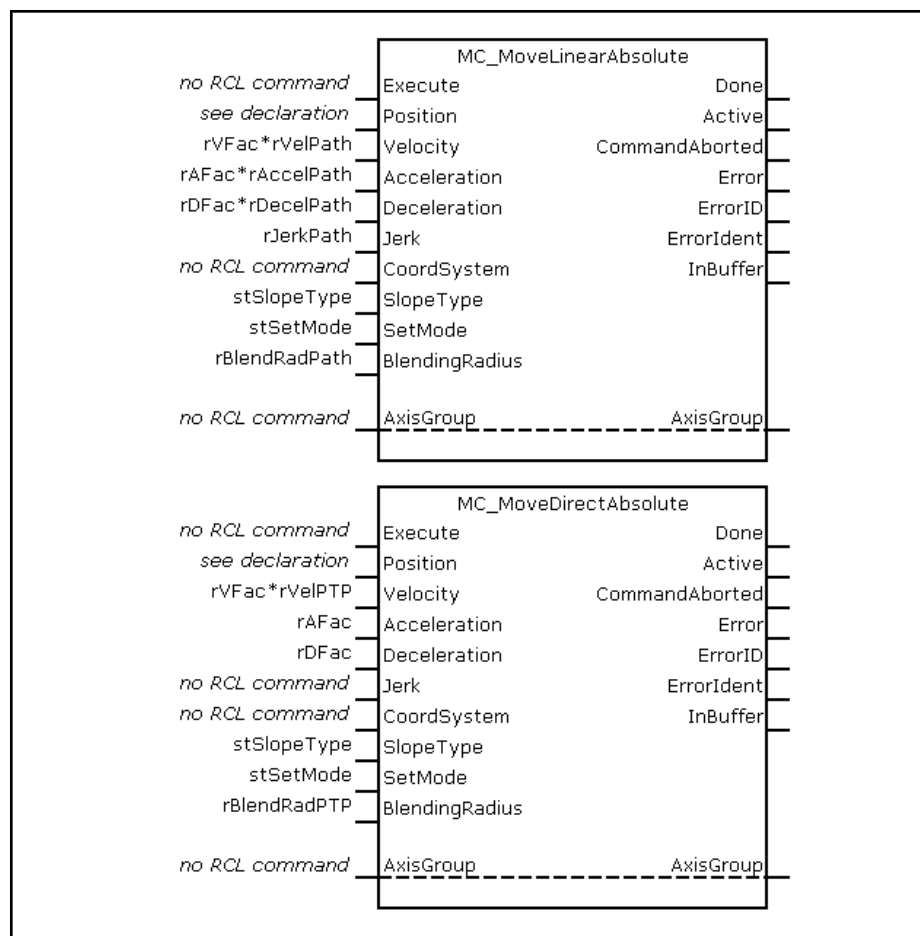


Fig. 12-2: Boundary conditions in the RCL program and inputs of the PLCopen function blocks

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 explanation at the end of the table)
UNTIL	No explicit input (see explanation at the end of the table)

Upgrading from Robot Control V1 to Robot Control V2

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) in case of relative motion commands.

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 451](#)

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 MC_MoveLinearRelative MC_MoveCircularRelative
PAUSE	MC_GroupInterrupt
HALT	MC_GroupStop
POS	MC_GroupReadActualPosition
SYNC	The belt value is reset to 0 automatically at 10 000 for the axis module value depending on Bit 1 of parameter K-0-0099
SYNCHRON	Corresponds to a belt value $\neq 0$ at the target point
SYNCHRON_END	Corresponds to a belt value $= 0$ in the target point
WAIT ...	ML_GroupWait
WAIT UNTIL ...	Can be mapped using an IF query

12.4 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.

Upgrading from Robot Control V1 to Robot Control V2

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

```

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.

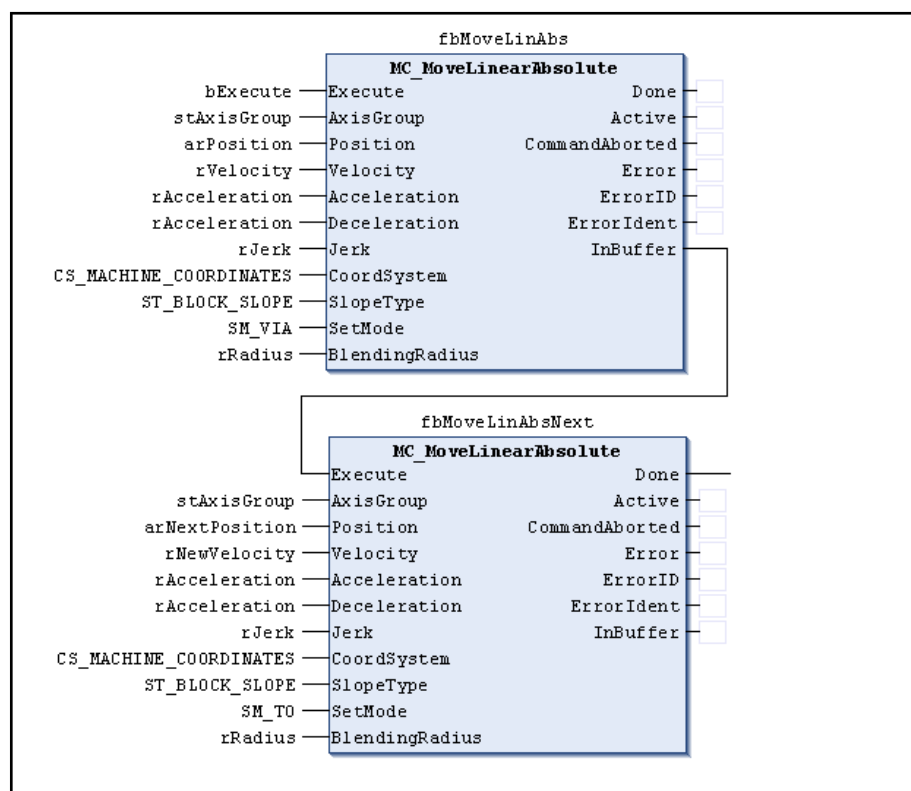


Fig. 12-3: *PLCopen program example*

If a motion is blended depends with PLCOpen function blocks on the function block output used to activate the following function block. As shown in the example (see figure above), 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.

Upgrading from Robot Control V1 to Robot Control V2

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!

Motion with an abort condition

RCL program example

```
BEGIN
BLOCK SLOPE
V = rVelocity
A = rAcceleration
J = rJerk
R = rRadius

MOVE LINEAR TO arPosition UNTIL bAbort = FALSE
PROGRAM_END
```

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.

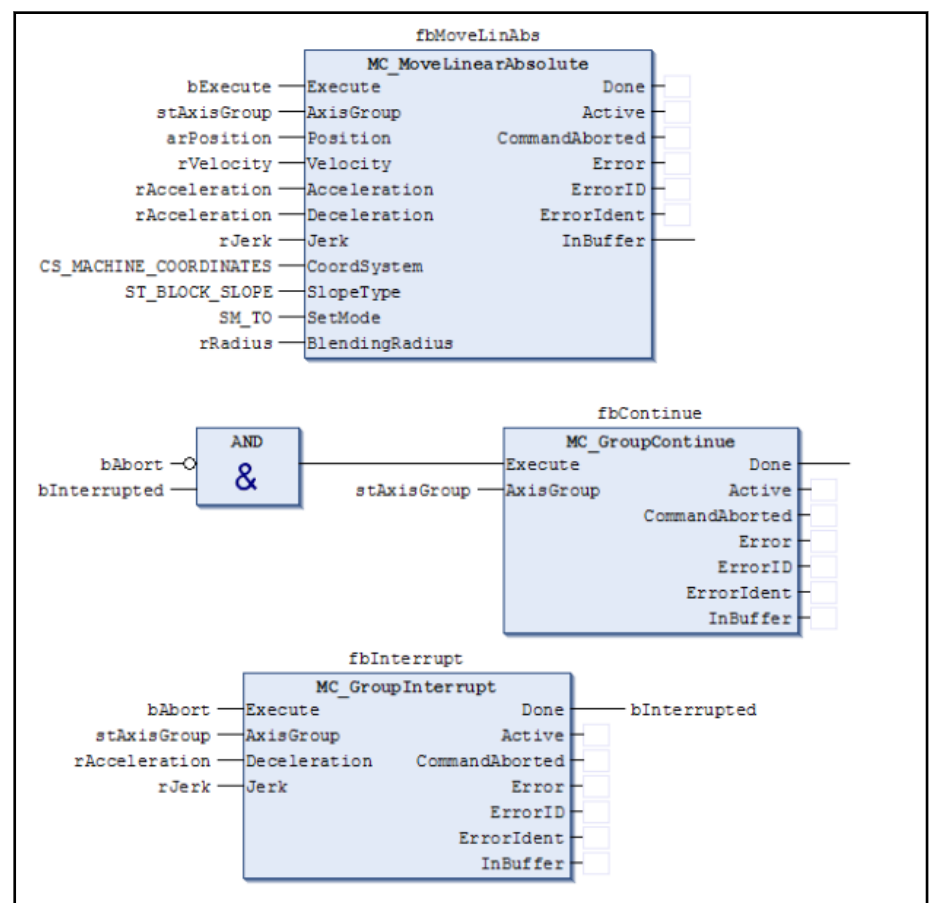


Fig. 12-4: PLCopen program example only for Robot Control V2

Upgrading from Robot Control V1 to Robot Control V2

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 (instance "fbMoveLinAbs"). As can be seen from the example, the motion sequence is aborted or not started while the variable (condition) is set to "bAbort". If this variable is reset, the motion is continued or can be started by a rising edge at the "Execute" input of the "fbMoveLinAbs" instance.

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

```

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.

Upgrading from Robot Control V1 to Robot Control V2

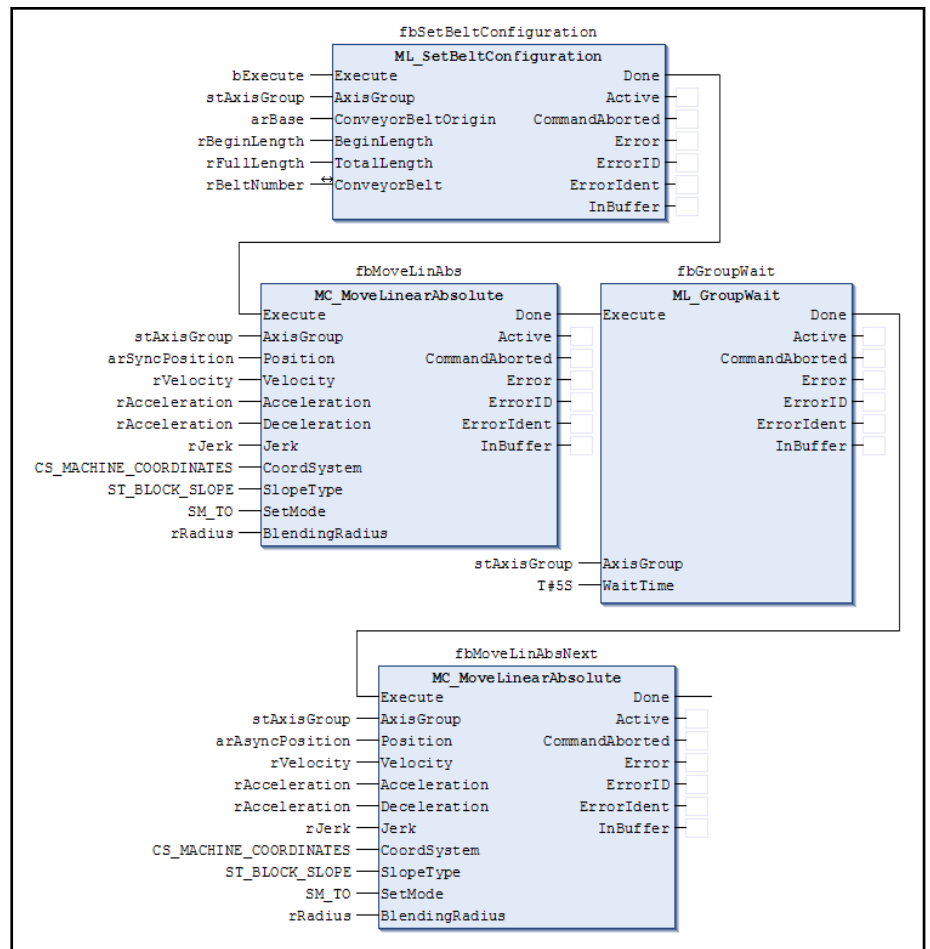


Fig. 12-5: PLCOpen program example only for Robot Control V2

When using the PLCOpen function block, the belt is configured by the `ML_SetBeltConfiguration` function block. 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` 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.

For more information on synchronization and desynchronization, see [Belt operation on page 69](#).

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

- [Using coordinate systems on page 391](#)
- [Belt synchronization on page 392](#)

12.5 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

Upgrading from Robot Control V1 to Robot Control V2

process greatly depends on the respective program. Operator intervention in the IEC program is only required if belts or function blocks only valid for Robot Control V1 are used.

Replacing ML_SetBeltParameter by ML_SetBeltConfiguration

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.

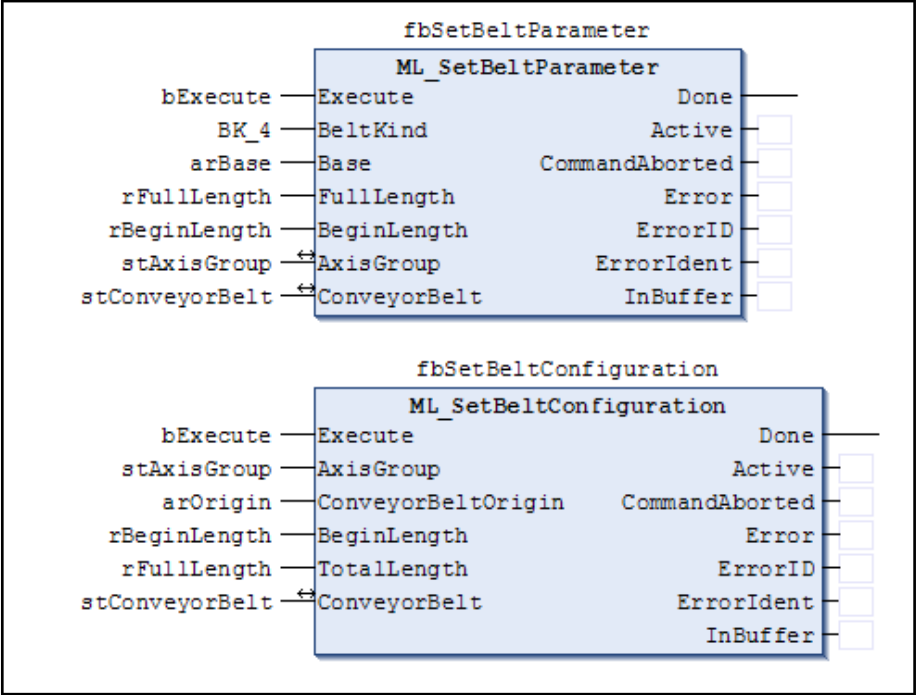


Fig. 12-6: Function blocks for belt configuration

The values for the "ConveyorBeltOrigin" input, however, cannot be applied directly from "Base" as the function and the array size have changed.

Definition of the position and rotation of the belt (ConveyorBeltOrigin)

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.

Replacing more function blocks

Remove or replace 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.

The following function blocks should be replaced accordingly:

Robot Control V1	Robot Control V2
ML_KinSetDistanceBlendOption	Using the parameter K-0-0099 , Bit 2
ML_SetBeltParameter	ML_SetBeltConfiguration

Upgrading from Robot Control V1 to Robot Control V2

Robot Control V1	Robot Control V2
ML_KinGetDistanceBlendOption	Using the parameter K-0-0099 , Bit 2
ML_IRDProgSelect	Not required anymore as RCL data is not available anymore
ML_IRDProgStart	Not required anymore as RCL data is not available anymore
ML_IRDProgStop	MC_GroupStop
ML_RoCoDir	Not required anymore as RCL data is not available anymore
ML_RoCoGetPoint	ML_GroupReadPoint
ML_RoCoGetPointList	Can be mapped using ML_GroupReadNextPoint (connect output "PointNameOut" to input "PointNameIn")
ML_RoCoSetPoint	ML_GroupWritePoint
ML_KinGetUserBit	Not required anymore as RCL data is not available anymore
ML_KinGetUserBit_DWORD	Not required anymore as RCL data is not available anymore
ML_KinGetUserDInt	Not required anymore as RCL data is not available anymore
ML_KinGetUserReal	Not required anymore as RCL data is not available anymore
ML_KinSetUserBit	Not required anymore as RCL data is not available anymore
ML_KinSetUserDInt	Not required anymore as RCL data is not available anymore
ML_KinSetUserPoint	Not required anymore as RCL data is not available anymore
ML_KinSetUserReal	Not required anymore as RCL data is not available anymore

Update the plug-in when using the kinematic interface.

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.

13 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 Hotline** and **Service Helpdesk** 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
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

A

A-0-0028, Travel range limit switch.....	413
A-0-0029, Position polarities.....	413
A-0-0030, Position limit value, positive.....	414
A-0-0031, Position limit value, negative.....	415
A-0-0032, Velocity limit value, positive.....	415
A-0-0033, Velocity limit value, negative.....	415
A-0-0034, Acceleration limit value, bipolar.....	415
A-0-0035, Velocity limit value, bipolar.....	415
A-0-0036, Jerk limit value, bipolar.....	416
A-0-0038, Torque/force limit value, bipolar.....	416
A-0-0039, Torque/force limit value, positive.....	416
A-0-0040, Torque/force limit value, negative....	417
A-0-0048, Velocity polarities.....	417
A-0-0049, Torque-/force polarities.....	418
A-0-0224, Rapid stop deceleration.....	418
A-0-0225, Rapid stop jerk.....	418
A-0-0240, Feedrate override.....	419
A-0-0241, Feedrate override control word.....	419
Abbreviations.....	21
About this documentation.....	13
Validity of the documentation.....	13
Absolute positioning.....	47
Access kinematic data.....	277
ACS.....	25
Active.....	192
Adding a kinematics to the system overview....	319
ANSI Z535.6-2006.....	14
Application phases.....	13
ASC_INT, Creating.....	447
Auxiliary axes	
Circular.....	41
Linear.....	40
Axis coordinate system.....	25
Axis coordinate system (ACS).....	25
Axis types.....	30

B

Base coordinate system.....	26
Base coordinate system (BCS).....	26
BCS.....	26
Belt configuration.....	66
Belt operation	
Adaptive motion.....	72
General.....	69
Belt values.....	67
Belt-specific error reaction.....	79
Belt-synchronous motion.....	65
Blending and belt synchronization.....	73
Blending radius.....	53
Block_Slope.....	54

C

C-0-0123, Maximum number of kinematics.....	419
C-0-2480, Configured kinematic number.....	419

C-0-2482, List of non-linked axes.....	419
C-0-2483, List of configured kinematics.....	419
C-0-2484, Kinematic configuration list.....	420
Circular.....	40
CommandAborted.....	192
Commanding	
Modulo-scaled axes.....	47
Complaints.....	21
Connecting third-party systems.....	443
uniVAL drive.....	443
Cont_Slope.....	56
Continuous operation.....	49
Coordinate systems.....	23
Coordinates.....	23
Criticism.....	21
Customer Feedback.....	21
Cyclic command value specification	
Robot Control V2.....	80

D

Degenerate circles.....	41
Dialog-based kinematics commanding.....	340
Documentation structure.....	13
Done.....	192
DV_Kin[].....	186

E

E_KIN_MECH_CMD.....	306
E_KIN_TEACH_IN_POINT_FILE_OPTIONS....	310
Enable.....	192
Error codes	
MC_KIN_REF.....	301
MC_KIN_REF_EXT.....	304
MC_MoveCircularAbsolute.....	250
MC_MoveCircularRelative.....	252
MC_MoveDirectAbsolute.....	255
MC_MoveDirectRelative.....	257
MC_MoveLinearAbsolute.....	259
MC_MoveLinearRelative.....	262
MC_SetCartesianTransform.....	208
MC_SetKinTransform.....	209, 211
ML_GroupAllAxes.....	214
ML_GroupJogCont.....	267
ML_GroupJogStep.....	269
ML_KinematicsInitType01.....	358
ML_KinematicsInterfaceType01.....	360, 361
ML_KinematicsInterfaceType11.....	363
ML_RobotGetBeltConfiguration.....	227
ML_RobotGetBeltErrorReaction.....	229
ML_RobotGetCmdInfo.....	231
ML_RobotGetMechanic.....	232
ML_RobotGetMoveID.....	233
ML_RobotGetPerformance.....	234
ML_RobotGetSafeZones.....	236
ML_RobotGetTrafoValues.....	237

Index

ML_RobotMoveJumpAbsolute.....	277	Inputs and outputs of edge-controlled function blocks, final processing.....	196
ML_RobotMovePoint.....	238	Inputs and outputs of edge-triggered function blocks, permanent processing.....	202
ML_RobotReadCyclicChannel.....	240	Inputs and outputs of state-controlled function blocks, final processing.....	193
ML_RobotSetMoveID.....	224	Inputs and outputs of state-controlled function clocks, permanent processing.....	199
ML_RobotSetPerformanceEnable.....	241	InSync.....	192
ML_RobotTransformPoint.....	242	INT_ASC, Creating.....	447
ML_SetBeltConfiguration.....	218	IntelliSense	
ML_SetBeltErrorReaction.....	220	Kinematic data.....	185, 186
ML_SetCylindricTransform.....	222	Intermediate point.....	40
ML_SetRotaryTableConfiguration.....	223	InTorque.....	192
ML_SwitchOnPath.....	244	InVelocity.....	192
ML_UngroupAllAxes.....	212	ITF_KINEMATICS structures.....	379
Error reaction			
Belt-specific.....	79		
Commanded motion.....	79		
Execute.....	192		
ExecuteLock.....	192		
Extending the interface structure (MB_KinInterfaceType11)			
Add new interface.....	345		
Extending the function block.....	344		
Extending the interface.....	344		
Notes regarding the implementation.....	343		
F			
Feedback.....	21		
Folder Kin_Interface/Type/Control, overview....	368		
Folder Kin_Interface/Type/Status, overview....	380		
Function block types.....	191		
Function block types and their interfaces.....	191		
Function variable access			
DV_Kin[].....	186		
function variable access status.....	187		
G			
Global variables.....	353		
Global_MB_KinInterfaceVars.....	353		
GVL_K_Param_ID.....	277		
GVL_RobotConstant.....	185		
H			
HalfJump.....	44		
HalfJumpDown.....	45		
HalfJumpUp.....	44		
Hazard warnings.....	14		
Helpdesk.....	459		
Hotline.....	459		
I			
InBuffer.....	204		
Information representation			
Names and abbreviations.....	21		
InGear.....	192		
Initial commissioning			
Kinematics.....	99		
InOperation.....	192		
		J	
		Jump.....	42
		K	
		K-0-0001, Logic kinematic number.....	420
		K-0-0002, Kinematic name.....	420
		K-0-0003, Maximum path velocity.....	421
		K-0-0005, Software limit switch, positive, MCS.....	421
		K-0-0006, Software limit switch, negative, MCS.....	421
		K-0-0007, Axis number of kinematics.....	421
		K-0-0008, Axis configuration list.....	421
		K-0-0009, Osci: list of all K-parameters that can be oscilloscoped.....	423
		K-0-0010, List of all K-parameters.....	423
		K-0-0011, List of all invalid K-parameters.....	424
		K-0-0012, Checksum of all K-parameters.....	424
		K-0-0013, IDN list of checksum K-parameters.....	424
		K-0-0016, Project identification number.....	424
		K-0-0019, Kinematic type.....	424
		K-0-0020, Kinematic diagnostic message.....	425
		K-0-0021, Kinematic status.....	425
		K-0-0022, Extended kinematic status.....	426
		K-0-0023, Kinematic diagnostic number.....	427
		K-0-0028, Velocity limit auxiliary axes.....	427
		K-0-0029, Acceleration limit auxiliary axes.....	427
		K-0-0030, Jerk limit auxiliary axes.....	428
		K-0-0031, Transformation scheme.....	428
		K-0-0034, Maximum path acceleration (bipolar).....	428
		K-0-0035, Maximum path jerk (bipolar).....	429
		K-0-0037, Jerk for move commands during deceleration phase.....	429
		K-0-0040, Transformation, lengths of the mechanics.....	429
		K-0-0041, Transformation, factors.....	429
		K-0-0042, Offset of BCS system.....	429
		K-0-0050, Distance units MCS.....	430

K-0-0051, Velocity units MCS.....	430
K-0-0052, Acceleration units MCS.....	431
K-0-0053, Jerk units MCS.....	432
K-0-0054, Measuring unit for kinematic data....	432
K-0-0070, Belt number of kinematics.....	433
K-0-0090, Taskpriority preparation.....	433
K-0-0095, ID of movement.....	433
K-0-0096, Peak value of command buffer size.....	433
K-0-0097, Maximum command buffer size.....	433
K-0-0098, Current command buffer size.....	433
K-0-0099, Kinematic configuration.....	434
K-0-0100, Current position, Cartesian Ma- chine Coordinate System.....	435
K-0-0102, Current path velocity.....	435
K-0-0103, Current path acceleration.....	435
K-0-0104, Current path jerk.....	435
K-0-0150, Interpolated command position.....	435
K-0-0152, Interpolated path velocity.....	436
K-0-0153, Interpolated path acceleration.....	436
K-0-0154, Interpolated path jerk.....	436
K-0-0223, Selection of error reaction.....	436
K-0-0224, Emergency halt deceleration.....	436
K-0-0225, Emergency halt jerk.....	437
K-0-0240, Feedrate override.....	437
K-0-0241, Feedrate override control word.....	437
K-0-0242, Acceleration override.....	437
K-0-0243, Deceleration override.....	437
K-0-0250, Continuous slope lead time.....	438
K-0-0815, Odometer.....	438
K-0-1000, Command: Load basic kinematic parameters.....	438
K-0-2100, List of all backup operation K-pa- rameters.....	438
Kin_Axis_Config_Elements.....	363
Kin_Axis_Type.....	364
KinData[] - Kinematic data in shared memo- ry.....	185
Kinematic commissioning.....	87
Belt configuration.....	92
Grouping a kinematics.....	90
Jogging.....	90
Overview.....	87
PLC configuration.....	94
Positioning.....	91
Product coordinates.....	94
Safety functions.....	89
Kinematic data via function variable access.....	186
Kinematic interface operating modes Overview.....	365
Kinematic interface program.....	317
Kinematic interface structures	
MB_KINEMATICS_ADMIN_STATUS.....	381
MB_KINEMATICS_ADMINISTRATION.....	369
MB_KINEMATICS_CONTROL_TYPE01.....	369
MB_KINEMATICS_CONV_BELT_STA- TUS.....	383
MB_KINEMATICS_CONVEYOR_BELT.....	372

MB_KINEMATICS_COORDINATED.....	377
MB_KINEMATICS_DIAGNOSIS.....	385
MB_KINEMATICS_MODE.....	367
MB_KINEMATICS_MODE_BITS.....	366
MB_KINEMATICS_MODE_UNION.....	365
MB_KINEMATICS_PCS_CONFIG.....	375
MB_KINEMATICS_PCS_STATUS.....	384
MB_KINEMATICS_SETUP.....	376
MB_KINEMATICS_SETUP_STATUS.....	386
MB_KINEMATICS_STATUS_TYPE01.....	380
MB_KINEMATICS_TEACH_IN.....	373
MB_KINEMATICS_TEACH_IN_STATUS....	384
Kinematic interface visualizations.....	317
KinInterface/Const.....	353
KinInterface/FB.....	355
KinInterface/FB/DataTypes.....	363
KinInterface/Type.....	364
KinInterface/Type/Control.....	368
KinInterface/Type/Status.....	380
KinJog.....	386
KinJog/FB.....	386
KinJog/Type.....	389
KinSystem_Overview_All.....	318

L

Library	
ML_KinTech.....	353
ML_Robot.....	183
Linear.....	39
Load basic parameters	
Kinematics.....	99

M

Machine coordinate system.....	26
Machine coordinate system (MCS).....	26
Manual mode.....	47, 52
MB_GroupReadListParameter.....	278
MB_GroupReadParameter.....	280
MB_GroupReadRealParameter.....	281
MB_GroupReadSercosAttribute.....	282
MB_GroupReadSercosDataStatus.....	283
MB_GroupReadStringParameter.....	284
MB_GroupWriteListParameter.....	285
MB_GroupWriteParameter.....	287
MB_GroupWriteRealParameter.....	288
MB_GroupWriteStringParameter.....	289
MB_KINEMATICS_ADMIN_CHANGE_OP- MODE.....	375
MB_KINEMATICS_ADMIN_STATUS.....	381
MB_KINEMATICS_ADMINISTRATION.....	369
MB_KINEMATICS_CONTROL_TYPE01.....	369
MB_KINEMATICS_CONV_BELT_STATUS.....	383
MB_KINEMATICS_CONVEYOR_BELT.....	372
MB_KINEMATICS_COORDINATED.....	377
MB_KINEMATICS_DIAGNOSIS.....	385
MB_KINEMATICS_MODE.....	367
MB_KINEMATICS_MODE_BITS.....	366

Index

MB_KINEMATICS_MODE_UNION.....	365	ML_KIN_PERF_DATA.....	309
MB_KINEMATICS_PCS_CONFIG.....	375	ML_KIN_SAFE_ZONE_DATA.....	310
MB_KINEMATICS_PCS_STATUS.....	384	ML_KIN_SAFE_ZONE_TYPE.....	306
MB_KINEMATICS_SETUP.....	376	ML_KIN_USED_AXES.....	306
MB_KINEMATICS_SETUP_STATUS.....	386	ML_KINEMATICSDATA_SM.....	307
MB_KINEMATICS_STATUS_TYPE01.....	380	ML_KinematicsInitType01.....	355
MB_KINEMATICS_TEACH_IN.....	373	ML_KinematicsInterfaceType01.....	358
MB_KINEMATICS_TEACH_IN_MODE.....	374	ML_KinematicsInterfaceType11.....	361
MB_KINEMATICS_TEACH_IN_STATUS.....	384	ML_KinGetDistanceBlendOption, Convert-	
MC_GroupContinue.....	244	ing.....	457
MC_GroupInterrupt.....	245	ML_KinGetUserBit_DWORD, Converting.....	457
MC_GroupReadActualPosition.....	224	ML_KinGetUserBit, Converting.....	457
MC_GroupReset.....	205	ML_KinGetUserDInt, Converting.....	457
MC_GroupStop.....	246	ML_KinGetUserReal, Converting.....	457
MC_KIN_REF.....	300	ML_KinInterface.....	353
MC_KIN_REF_EXT.....	302	ML_KinJog.....	386
MC_MoveCircularAbsolute.....	248	ML_KinSetDistanceBlendOption, Converting...	456
MC_MoveCircularRelative.....	250	ML_KinSetUserBit, Converting.....	457
MC_MoveDirectAbsolute.....	252	ML_KinSetUserDInt, Converting.....	457
MC_MoveDirectRelative.....	255	ML_KinSetUserPoint, Converting.....	457
MC_MoveLinearAbsolute.....	257	ML_KinSetUserReal, Converting.....	457
MC_MoveLinearRelative.....	260	ML_KinTech.library.....	353
MC_SetCartesianTransform.....	206	KinInterface/Const.....	353
MC_SetKinTransform.....	208	KinInterface/FB.....	355
MC_SetKinTransformExt.....	210	KinInterface/FB/DataTypes.....	363
MC_UngroupAllAxes.....	211	KinInterface/Type.....	364
MCS.....	26	KinInterface/Type/Control.....	368
Mechanical information.....	180	KinInterface/Type/Status.....	380
ML_AddMechanicData.....	212	ML_POINTKIND.....	310
ML_ARR_COORDS.....	311	ML_Robot	
ML_BELT_ERROR_REACTION_MODE.....	305	Config function blocks.....	205
ML_BeltDeSynchronisationWithLimits.....	262	Data types.....	298
ML_BeltSynchronisationWithLimits.....	263	Function blocks.....	188, 204
ML_CIRCMODES.....	311	Info function blocks.....	224
ML_COORDSYS.....	312	Info functions.....	224
ML_CYCLIC_MODE.....	313	Movement function blocks.....	244
ML_GroupAllAxes.....	213	Parameter function blocks.....	277
ML_GroupDeletePointFile.....	290	Read/write parameters.....	184
ML_GroupDeletePoints.....	291	TeachIn function blocks.....	290
ML_GroupJogCont.....	265	ML_Robot.library.....	183
ML_GroupJogStep.....	267	ML_RobotGetBeltConfiguration.....	226
ML_GroupLoadPointFile.....	292	ML_RobotGetBeltDesyncRelConfiguration.....	227
ML_GroupReadNextPoint.....	293	ML_RobotGetBeltErrorReaction.....	228
ML_GroupReadPoint.....	295	ML_RobotGetCmdInfo.....	229
ML_GroupSavePointFile.....	296	ML_RobotGetMechanic.....	231
ML_GroupSetSafeZone.....	214	ML_RobotGetMoveID.....	232
ML_GroupWait.....	269	ML_RobotGetPerformance.....	233
ML_GroupWritePoint.....	297	ML_RobotGetSafeZones.....	234
ML_IRDProgSelect, Converting.....	457	ML_RobotGetTrafoValues.....	236
ML_IRDProgStart, Converting.....	457	ML_RobotMoveJumpAbsolute.....	274
ML_IRDProgStop, Converting.....	457	ML_RobotMovePoint.....	237
ML_JOGMODE.....	389	ML_RobotOpenCyclicChannel.....	270
ML_KIN_AXES_CONFIG_DATA.....	305	ML_RobotOpenDirectCyclicChannel.....	272
ML_KIN_AXIS_CONFIG.....	305	ML_RobotReadCyclicChannel.....	238
ML_KIN_CMD_INFO.....	308	ML_RobotSetMoveID.....	223
ML_KIN_CS_TYPE.....	308	ML_RobotSetPerformanceEnable.....	240
ML_KIN_INFO_TYPE.....	308	ML_RobotTransformPoint.....	241
ML_KIN_MECHANIC_DATA.....	309	ML_RobotWriteCyclicChannel.....	273

ML_RoCoDir, Converting.....	457
ML_RoCoGetPoint, Converting.....	457
ML_RoCoGetPointList, Converting.....	457
ML_RoCoSetPoint, Converting.....	457
ML_SET_MODE.....	313
ML_SetBeltConfiguration.....	217
ML_SetBeltDesyncRelConfiguration.....	216
ML_SetBeltErrorReaction.....	218
ML_SetBeltParameter, Converting.....	456
ML_SetCylindricTransform.....	220
ML_SetRotaryTableConfiguration.....	222
ML_SLOPE_TYPE.....	313
ML_SwitchOnPath.....	242
MLPI COM	
Robot Control.....	36
MLPI core	
Robot Control.....	36
ModeCoordAB.....	329
ModeCoordAH.....	329
ModeCoordContinue.....	338
ModeCoordCyclic.....	339
ModeCoordExternalFB.....	339
ModeCoordHoming.....	330
ModeCoordInterrupt.....	338
ModeCoordPosCircAbs.....	334
ModeCoordPosCircRel.....	335
ModeCoordPosDirAbs.....	330
ModeCoordPosDirRel.....	331
ModeCoordPosJumpAbs.....	336
ModeCoordPosLinAbs.....	332
ModeCoordPosLinRel.....	333
ModeCoordStopping.....	337
Motion type	
Circular.....	40
HalfJump.....	44
HalfJumpDown.....	45
HalfJumpUp.....	44
Jump.....	42
Linear.....	39
PTP.....	41
Motions	
Continue.....	64
Interrupt.....	63
Stop.....	62
Wait.....	64
N	
Navigation system overview.....	320
O	
Operating modes	
Absolute circular positioning.....	334
Absolute Jump positioning.....	336
Absolute linear positioning.....	332
Absolute PTP positioning.....	330
Continue.....	338
External function block.....	339

Interrupt.....	338
Kinematic homing.....	330
Kinematic ready.....	329
Kinematic stop.....	337
Relative circular positioning.....	335
Relative linear positioning.....	333
Relative PTP positioning.....	331
Operating modes of the kinematic interface	
Overview.....	328
Operation modes.....	328
Cyclic position command value specification.....	339
P	
Parameter access of the PLC program.....	277
PathViewer.....	95
Context menu.....	97
Navigation and operation.....	96
Overview.....	95
PCS.....	27
Performance measurement.....	82
Point.....	31
Positioning	
Absolute.....	47
Relative.....	47
Product coordinate system (PCS).....	27
ProductCoordSystem.....	327
PTP.....	41
R	
REFERENCE_TO	
Kinematic data.....	185, 186
Relative positioning.....	47
Robot Control	
Axis-related parameters.....	413
Command states.....	36
Commands.....	36
Control-related parameters.....	419
Diagnostics.....	441
Functions.....	36
Kinematic-related parameters.....	420
Kinematics – Context menu: Diagnostics.....	102
Parameter.....	413
Robot Control application example	
Adaptive Motion.....	402
Command buffer.....	403
Cyclic command value specification.....	405
Extended user-defined transformation.....	396
User-defined transformation.....	393
Using coordinate systems.....	391
Robot Control application examples	
Belt synchronization.....	392
Robot Control use case	
Extend a point.....	402
Robot Control V1 to Robot Control V2.....	445
IEC commands.....	455
IndraWorks project.....	445

Index

Overview.....	445	Read next point.....	51
RCL commands.....	446	Read point.....	51
RCL programs.....	451	Save point file.....	51
Robot Control V2		Structure of the point file.....	50
ACS/BCS transformations.....	107	Write point.....	50
Application examples.....	391	Teach-In.....	49
Blending.....	52	Transformation	
Connecting third-party robotic systems.....	443	Axis coordinates (ACS)/base coordinates (BCS).....	28
Continuous operation.....	49	Base coordinates (BCS)/machine coordinates (MCS).....	29
Cyclic command value specification.....	80	Machine coordinates (MCS)/product coordinates (PCS).....	29
Errors and error reactions.....	78	Transformation rule	
Functional description.....	23	2-axis delta slide.....	152
Interface functions.....	85	2-axis delta with 1 optional orientation axis.....	109
Kinematics – Context menu.....	98	2-axis delta with asymmetric TCP.....	144
Manual mode.....	47	2-axis parallelogram.....	123
Performance measurement.....	82	2-axis parallelogram V1.....	129
Robot folder – Coordinate offset dialog.....	106	2-axis parallelogram V2.....	133
Robot folder – Kinematic dialogs.....	104	2-axis press.....	126
Robot folder– Axis lengths and couplings dialog.....	105	3-axis Canfield.....	170
Robot folder– Motion limit values dialog.....	105	3-axis delta linear with 1 optional orientation axis.....	134
Single-step operation.....	48	3-axis delta with 1 optional orientation axis.....	131
Slope types.....	53	3-axis delta with linear axes with 1 optional orientation axis.....	149
Switching on the belt.....	73	3-axis flex stamper.....	147
System overview.....	23	3-axis parallelogram V3.....	153
Transformation schemes.....	107	3-axis Scara with 1 optional orientation axis.....	115
Traversing range limits.....	76	3-axis twisting crane.....	145
Robot Control, commands and functions.....	34	4-axis Scara with 1 optional orientation axis.....	119
Robot Control, Move ID.....	59	4-axis Scara with crank.....	166
Robot-Control V2		4-axis speedpicker.....	159
Folder "Robot" – dialog: Scaling/Units.....	104	5-axis delta.....	137
S		6-axis Cartesian.....	154
Safety instructions.....	14	Adabot.....	141
Service hotline.....	459	Cartesian.....	108
Shared memory		Contact kinematics.....	128, 170
KinData[].....	185	Inverted 2-axis parallelogram.....	125
Shutdown.....	192	Stewart platform.....	162
Signal alert symbol.....	14	User-defined.....	174
Signal words.....	14	User-defined extended.....	176
Single-step operation.....	48	Transformation rules	
Standard interfaces at function blocks.....	190	Overview.....	27
State diagram		Transformation specification	
Kinematics.....	33	3-axis delta with 90 degree angles.....	156
StateDiagram		4-axis Scara with linear lifting axis.....	111
Axisgroup.....	33	Transformations	
Suggestions.....	21	Robot Control V2.....	27
Support.....	459	Traversing range limits.....	76
Symbols.....	15		
T			
Target group.....	13		
TE_KinInterfaceType01Prog.....	317		
Teach-in			
Delete point file.....	52		
Delete points.....	51		
Load point file.....	52		

U

uniVAL drive.....	443
Upgrading to an event-controlled kinematic interface (type 11).....	345
Use cases	
Registration of a user-defined transformation.....	410
Sequential programming with the kinematic interface.....	408
User extension.....	343, 344, 345
Extending the interface structure (MB_KinInterfaceType11).....	344
User extensions.....	343

V

Visualization	
Belt configuration.....	324
Homing overview.....	321
Jog mode/Setup.....	324
Kinematic overview.....	321
Overview.....	317
PCS configuration.....	327
Positioning overview.....	322
System overview.....	318
TeachIn.....	325

W

Warnings.....	14

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911341588