

Rexroth IndraMotion MLC 13VRS RCL Programming Instruction

R911336297
Edition 01

Application Manual



Title	Rexroth IndraMotion MLC 13VRS RCL Programming Instruction								
Type of Documentation	Application Manual								
Document Typecode	DOK-MLC***-RCL*PRO*V13-AP01-EN-P								
Internal File Reference	RS-10b268d8de20fc020a6846a500eeae10-1-en-US-5								
Purpose of Documentation	This documentation provides information on the RobotControl. The programming language RCL (RobotControl Language) is focused. The program structure, variables, functions, motion statements, transformation rules and the required system parameters are described.								
Record of Revision	<table><tr><th>Edition</th><th>Release Date</th><th>Notes</th></tr><tr><td>Edition 01</td><td>05.2012</td><td>First edition for MLC 13VRS</td></tr></table>			Edition	Release Date	Notes	Edition 01	05.2012	First edition for MLC 13VRS
Edition	Release Date	Notes							
Edition 01	05.2012	First edition for MLC 13VRS							
Copyright	© Bosch Rexroth AG 2012 This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.								
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.								
Published by	Bosch Rexroth AG 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 http://www.boschrexroth.com/electrics DC-IA/EAY2, GBu, HBu, vha (MePe)								
Note	This document has been printed on chlorine-free bleached paper.								

Table of Contents

	Page
1 About this Documentation.....	11
1.1 Validity of the Documentation.....	11
1.2 Required and Supplementing Documentations.....	11
1.3 Information Representation.....	16
1.3.1 Safety Instructions.....	16
1.3.2 Symbols Used.....	17
1.3.3 Names and Abbreviations.....	17
2 Coordinates and Coordinate Systems.....	19
2.1 General Information.....	19
2.2 Joint Coordinate System (JCS).....	20
2.3 Basic Coordinate System (BCS).....	20
2.4 World Coordinate System (WCS).....	22
2.5 Workpiece Coordinate System (WpCS).....	23
2.6 Tool Coordinate System (TCS).....	24
3 Using Coordinate Systems.....	25
3.1 Defining Coordinate Systems.....	25
3.2 Defining the Basic Coordinate System Relative to the World Coordinate System.....	25
3.3 Defining the Workpiece Coordinate System Relative to the World Coordinate System.....	27
3.4 Workpiece Coordinate File.....	29
3.5 Workpiece Coordinate Editor.....	29
3.6 Library Functions.....	30
3.7 Defining the Tool Coordinate System.....	36
4 Travel Range Limits in the RobotControl.....	39
5 RCL Program Structure.....	41
5.1 General Information.....	41
5.2 Mode of the Compiler Operation.....	41
5.2.1 Use.....	41
5.2.2 Source Files.....	41
5.2.3 Output Files.....	41
5.3 Compiler Statements.....	42
5.3.1 General Information.....	42
5.3.2 Kinematic Definition.....	42
5.3.3 World Coordinate Names (WC Name Definition).....	42
5.3.4 Axis Names (JC Name Definition).....	43
5.3.5 Including Files.....	43
5.3.6 Selectable Extension in the Include Statement.....	44
5.3.7 Debug Information.....	44
5.4 Main Program Structure.....	44
5.4.1 General.....	44

Table of Contents

	Page
5.4.2 Declaration Part.....	45
5.4.3 Statement Part.....	46
5.5 Program Declaration.....	46
5.6 Main Program Call in Main Program.....	47
5.7 Subroutine Declaration.....	50
5.8 Subroutine Sequence.....	52
6 RCL Constants.....	57
6.1 RCL Constant Declaration	57
6.2 Predefined RCL Constants.....	57
7 RCL Variables.....	59
7.1 General Information.....	59
7.2 RCL Data Types.....	59
7.2.1 Overview.....	59
7.2.2 Simple RCL Data Types.....	60
7.2.3 Composed RCL Data Types.....	61
7.2.4 User-Defined RCL Types.....	61
7.3 RCL Variable Declaration.....	63
7.4 Global Variables.....	64
7.5 Point Variables.....	65
7.5.1 Declaration of Point Variables.....	65
7.5.2 Points and Point File "pkt".....	66
7.5.3 Complete Value Assignment.....	66
7.5.4 Assignment of Variables for Individual Components.....	66
7.6 Working with Text Variables.....	68
7.7 Working with Array Variables (Arrays).....	69
7.8 PLC Communication Channels.....	70
7.8.1 Communication Channels to PLC, General.....	70
7.8.2 Communication Channels to PLC, Declaration and Programming.....	73
8 RCL Program Control.....	77
8.1 WAIT Statement.....	77
8.2 PAUSE Statement.....	81
8.3 HALT Statement.....	81
8.4 REPEAT Statement.....	82
8.5 JUMP Statement.....	83
8.6 IF-THEN-(ELSE) Statement.....	85
8.7 CASE Statement.....	88
9 Value Assignments and Operations.....	89
9.1 Value Assignments.....	89
9.2 Operations.....	89
9.2.1 Arithmetic Expressions.....	89
9.2.2 Comparative Operations.....	91

Table of Contents

	Page
9.2.3 Logic Expressions.....	91
9.2.4 Sequence of Operations.....	92
10 RCL Functions.....	95
10.1 SIN Function.....	95
10.2 COS Function.....	95
10.3 ATAN Function.....	95
10.4 SQRT Root Function.....	96
10.5 ABS Function.....	97
10.6 TRUNC Function.....	97
10.7 ORD Function.....	97
10.8 CHR Function.....	97
10.9 ROUND Function.....	97
10.10 EOF Function.....	98
10.11 ASSIGN Function.....	98
10.12 INT_ASC Function.....	98
10.13 ASC_INT Function.....	99
10.14 SIZEOF Function.....	100
10.15 LIMIT_MIN, LIMIT_MAX, LIMIT_OFF Functions.....	101
10.16 JC Standard Function.....	101
10.17 WC Standard Function.....	101
11 RCL Special Functions.....	103
11.1 RCL Special Functions, General.....	103
11.2 RCL Special Functions, Listed as Table.....	104
11.3 Position-accurate Switching.....	105
11.3.1 Position-accurate Switching, General.....	105
11.3.2 SPC_FCT: 1, Position-accurate Switching of Digital Outputs on the Path.....	106
11.3.3 SPC_FCT: 2, Position-accurate Switching of Process Data on the Path.....	107
11.4 SPC_FCT: 21, Belt Type.....	108
11.5 SPC_FCT: 23, DateTime.....	109
11.6 SPC_FCT: 24, SysClk.....	109
11.7 SPC_FCT: 28, Set Belt Counter.....	110
11.8 SPC_FCT: 53, Belt Synchronous Operating Range of Belt Type 4.....	110
11.9 RCL Programming "Blending Mode".....	113
11.9.1 SPC_FCT: 60, Set Blending mode.....	113
11.9.2 SPC_FCT: 59, Query blending mode.....	114
12 RCL Motion Statements.....	117
12.1 General Information.....	117
12.2 Direct Motion Statements.....	117
12.2.1 Overview.....	117
12.2.2 Motion Commands.....	118
12.2.3 Interpolation Method	122
12.2.4 Mode of Operation of Motion-Influencing RCL Statements and Parameters.....	125

Table of Contents

	Page
12.2.5 Additional Information.....	127
Additional information, General.....	127
Additional Information: Velocity.....	127
Additional Information: Velocity Factor.....	129
Additional Information: Acceleration.....	130
Additional Information: Acceleration Factor.....	131
Additional Information: Deceleration Factor.....	132
Additional Information on the Jerk.....	133
12.3 Motion-influencing Statements	134
12.3.1 General Information.....	134
12.3.2 Belt Synchronization.....	135
12.3.3 Block Transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)	137
12.3.4 Look-Ahead Across Several Blocks at PROGR_SLOPE and CONT_SLOPE.....	141
12.3.5 Late Commanding.....	141
12.3.6 Spatial Blending.....	142
Overview.....	142
Spatial Blending for PROGR_SLOPE and CONT_SLOPE.....	143
Spatial Blending for BLOCK_SLOPE.....	147
Late Blending.....	149
Blending Mode.....	151
13 Setup Mode.....	153
13.1 Setup Mode, General Information.....	153
13.2 Jogging in Different Coordinate Systems.....	153
13.2.1 Parameters to be Set to Jog in Axis Coordinates.....	153
13.2.2 Parameter to Be Set to Jog in World Coordinates.....	155
13.2.3 Jogging in World Coordinates.....	156
14 RCL Parameters.....	157
14.1 Axis-specific Parameters Used by the Kinematics of an MLC.....	157
14.1.1 General Information.....	157
14.1.2 A-0-0028, Travel range limit switch.....	157
14.1.3 A-0-0029, Position polarities.....	157
14.1.4 A-0-0030, Position limit value, positive.....	158
14.1.5 A-0-0031, Position limit value, negative.....	158
14.1.6 A-0-0032, Velocity limit value, positive.....	159
14.1.7 A-0-0033, Velocity limit value, negative.....	159
14.1.8 A-0-0034, Acceleration limit value, bipolar.....	159
14.1.9 A-0-0036, Jerk limit value, bipolar.....	159
14.1.10 A-0-0038, Torque/force limit value, bipolar.....	160
14.1.11 A-0-0048, Velocity polarities.....	160
14.1.12 A-0-0049, Torque-/force polarities.....	161
14.1.13 A-0-0080, Jogging, short distance.....	161
14.1.14 A-0-0081, Jogging, long distance.....	162
14.1.15 A-0-0082, Jogging, low velocity.....	163

Table of Contents

	Page
14.1.16 A-0-0083, Jogging, high velocity.....	163
14.1.17 A-0-0084, Jogging, acceleration.....	164
14.1.18 A-0-0085, Jogging, deceleration.....	165
14.1.19 A-0-0086, Jogging, acceleration jerk.....	165
14.1.20 A-0-0087, Jogging, deceleration jerk.....	165
14.1.21 A-0-0224, Rapid stop deceleration.....	166
14.1.22 A-0-0225, Rapid stop jerk.....	166
14.1.23 A-0-0230, Blending distance PTP.....	166
14.1.24 A-0-0240, Feedrate override.....	167
14.1.25 A-0-0241, Feedrate override control word.....	167
14.1.26 A-0-0242, Position control, synchronization acceleration.....	167
14.1.27 A-0-0243, Position control, synchronization velocity.....	168
14.2 Common C-Parameters of All Kinematics of an MLC.....	169
14.2.1 General Information.....	169
14.2.2 C-0-0123, Maximum number of kinematics.....	169
14.2.3 C-0-2480, Configured kinematic number.....	169
14.2.4 C-0-2482, List of non-linked axes.....	170
14.2.5 C-0-2483, List of configured kinematics.....	170
14.2.6 C-0-2484, Kinematic configuration list.....	170
14.3 Parameters of a Kinematics in the MLC.....	171
14.3.1 General Information.....	171
14.3.2 K-0-0001, Logic kinematic number.....	171
14.3.3 K-0-0002, Kinematic name.....	171
14.3.4 K-0-0003, Maximum path speed.....	171
14.3.5 K-0-0004, Reserved.....	172
14.3.6 K-0-0005, Software limit switches, positive, WC.....	172
14.3.7 K-0-0006, Software limit switches, negative, WC.....	172
14.3.8 K-0-0007, Axis number of kinematics.....	172
14.3.9 K-0-0008, Axis configuration list.....	172
14.3.10 K-0-0009, Osci: list of all K-parameters that can be oscilloscoped.....	174
14.3.11 K-0-0010, List of all K-parameters.....	174
14.3.12 K-0-0011, List of all invalid K-parameters.....	175
14.3.13 K-0-0012, Checksum of all K-parameters.....	175
14.3.14 K-0-0013, IDN list of checksum K-parameters.....	175
14.3.15 K-0-0014, Path acceleration, change time.....	175
14.3.16 K-0-0015, Path deceleration, change time.....	178
14.3.17 K-0-0016, Project identification number.....	180
14.3.18 K-0-0019, Kinematic type.....	181
14.3.19 K-0-0020, Kinematic diagnostic message.....	181
14.3.20 K-0-0021, Kinematic status.....	181
14.3.21 K-0-0022, Extended kinematic status.....	183
14.3.22 K-0-0023, Kinematics diagnostic number.....	186
14.3.23 K-0-0031, Transformation scheme.....	186
14.3.24 K-0-0032, Referencing axes, command configuration.....	187
14.3.25 K-0-0033, IPOS, update at coordinate change.....	187
14.3.26 K-0-0036, WC axis monitoring.....	187

Table of Contents

	Page
14.3.27 K-0-0040, Transformation, lengths of the mechanics.....	188
14.3.28 K-0-0041, Transformation, factors.....	188
14.3.29 K-0-0042, WC system offset.....	188
14.3.30 K-0-0043, Assignment list of workpiece coordinates.....	189
14.3.31 K-0-0045, Tool selection.....	190
14.3.32 K-0-0050, Distance units WC.....	190
14.3.33 K-0-0051, Velocity units WC.....	191
14.3.34 K-0-0052, Acceleration units WC.....	191
14.3.35 K-0-0053, Jerk units WC.....	192
14.3.36 K-0-0054, Measuring unit for kinematic data.....	193
14.3.37 K-0-0070, Belt number of kinematics.....	193
14.3.38 K-0-0071, Coupling factors, belt 1.....	193
14.3.39 K-0-0072, Coupling factors, belt 2.....	194
14.3.40 K-0-0073, Coupling factors, belt 3.....	194
14.3.41 K-0-0074, Coupling factors, belt 4.....	195
14.3.42 K-0-0080, Jogging, distance short in WC system.....	195
14.3.43 K-0-0081, Jogging, distance long in WC system.....	196
14.3.44 K-0-0082, Jogging, velocity low in WC system.....	197
14.3.45 K-0-0083, Jogging, velocity high in WC system.....	197
14.3.46 K-0-0084, Jogging, acceleration in WC system.....	198
14.3.47 K-0-0085, Jogging, deceleration in WC system.....	199
14.3.48 K-0-0086, Jogging, acceleration jerk in WC system.....	199
14.3.49 K-0-0087, Jogging, deceleration jerk in WC system.....	200
14.3.50 K-0-0099, Kinematic configuration.....	201
14.3.51 K-0-0100, Current position, Cartesian original coordinates.....	201
14.3.52 K-0-0102, Current path velocity.....	201
14.3.53 K-0-0103, Current path acceleration.....	201
14.3.54 K-0-0104, Current path jerk.....	201
14.3.55 K-0-0105, Actual position, Cartesian workpiece coordinates.....	202
14.3.56 K-0-0106, Coordinate values of tool.....	202
14.3.57 K-0-0107, Coordinate values of workpiece.....	202
14.3.58 K-0-0121 to K-0-0136, Coordinate Names.....	202
14.3.59 K-0-0121, Coordinate name C1.....	202
14.3.60 K-0-0155, Safe zones on/off.....	202
14.3.61 K-0-0156, Safe zone 1 min. point.....	203
14.3.62 K-0-0157, Safe zone 1 max. point.....	203
14.3.63 K-0-0156/K-0-0157 Safezone1 min./max. Point till K-0-0218/K-0-0219 Safezone32 min./max. Point	203
14.3.64 K-0-0220, File transfer name.....	205
14.3.65 K-0-0221, File name to be compiled.....	205
14.3.66 K-0-0240, Feedrate override.....	206
14.3.67 K-0-0241, Feedrate override control word.....	206
14.3.68 K-0-0242, Acceleration override.....	206
14.3.69 K-0-0243, Deceleration override.....	206
14.3.70 K-0-0250, Continuous slope lead time.....	207
14.3.71 K-0-1060, Command: Load file from CF.....	207

Table of Contents

	Page
14.3.72 K-0-1061, Command: Save file to CF.....	207
14.3.73 K-0-1062, Command: Compile file to MLC.....	207
14.3.74 K-0-2200, Programmed position 1 (active).....	207
14.3.75 K-0-2201, Programmed position 2 (active).....	207
14.3.76 K-0-2202, Type of actively programmed position.....	208
14.3.77 K-0-2203, Interpolation type.....	208
14.3.78 K-0-2204, Currently active QLL file name.....	208
14.3.79 K-0-2205, Currently active line number.....	208
14.3.80 K-0-2300, PLC-Bit input (RCL).....	208
14.3.81 K-0-2301, PLC-Integer input (RCL).....	208
14.3.82 K-0-2302, PLC-Real input (RCL).....	208
15 Kinematic User Interface.....	209
15.1 General Information.....	209
15.2 Menu Structure of the Kinematics.....	210
15.2.1 Menu Item: Robot.....	210
15.2.2 Menu item: Kinematics.....	211
15.2.3 Menu Item "Global Data".....	211
15.2.4 Menu item: "Programs".....	211
15.2.5 Menu item of a Program.....	211
15.2.6 Menu Item "Debug".....	212
15.3 Global Data – Context Menu.....	212
15.3.1 Overview.....	212
15.3.2 Workpiece Coordinates.....	212
15.3.3 Tool Management.....	213
15.3.4 Library files.....	213
15.4 Kinematics - Context menu.....	213
15.4.1 Overview.....	213
15.4.2 Setup.....	214
15.4.3 Diagnostics.....	214
15.5 Kinematics – Dialogs.....	216
15.5.1 Overview on dialogs.....	216
15.5.2 Scaling/Units.....	216
15.5.3 Axis Lengths and Coupling.....	216
15.5.4 Motion Limit Values.....	217
15.5.5 Belt Coupling Factors.....	218
15.5.6 Coordinate Offset.....	219
15.5.7 Jogging.....	219
15.5.8 Safe Zones.....	221
15.5.9 Workpiece Coordinate Assignment.....	223
15.6 RCL Editor.....	223
15.6.1 General Information.....	223
15.6.2 Characteristics.....	223
15.6.3 Creating New RCL Program.....	224
15.6.4 Working with the Editor.....	224
15.7 Compiler.....	228

Table of Contents

	Page
15.8 Program for Tracing.....	229
15.8.1 General Information.....	229
15.8.2 Connecting for Tracing.....	229
15.8.3 Disconnecting when Tracing.....	229
15.8.4 Starting Program for Tracing.....	230
15.8.5 Stopping Program during Tracing.....	230
15.9 Debugger.....	230
15.9.1 General Information.....	230
15.9.2 Connecting in Debug mode.....	230
15.9.3 Disconnecting in Debug mode.....	230
15.9.4 Starting Program in Debug Mode.....	230
15.9.5 Stopping Program in Debug Mode.....	230
15.9.6 Setting and Deleting Breakpoints.....	230
15.9.7 Debugging a Program.....	231
15.9.8 Displaying Current Block Number.....	231
15.9.9 Monitoring Variables.....	232
15.10 Settable Options for RoCo.....	232
15.10.1 General Information.....	232
15.10.2 Connection settings.....	233
15.10.3 Compiler.....	233
15.10.4 Text Editor.....	235
General Information.....	235
Text Area.....	236
Page Margins.....	240
Keywords.....	241
16 Using the RoCo Multi-Kinematic Technology.....	243
16.1 General Information.....	243
16.2 Creating Kinematics.....	243
16.2.1 General Information.....	243
16.2.2 Defining the Kinematic Type (Robot Type, K-0-0031).....	244
16.2.3 Defining the Axis Assignment (K-0-0008).....	245
16.2.4 Deleting Kinematics.....	246
16.3 Coordinated Motion of the Kinematics.....	247
16.4 RCL Programs of the Kinematics.....	248
16.5 Point Tables of the Kinematics.....	249
16.5.1 General Information.....	249
16.5.2 Point Tables, Assignment and Response.....	250
16.6 PLC Shared Memory Interfaces (KinData) of the Kinematics.....	251
17 RobotControl Functionality Programmed in the PLC.....	253
18 Transformation Rules.....	255
18.1 Transformation Rules, Overview.....	255
18.2 Robot Type 00, Cartesian Robot.....	255

Table of Contents

	Page
18.3 Robot Type 07, 2-Axis Delta with 1 Optional Manual Axis.....	259
18.4 Robot Type 09, 3-Axis Scara with 1 Optional Manual Axis.....	261
18.5 Robot Type 10, 4-Axis Scara with 1 Optional Manual Axis.....	264
18.6 Robot Type 12, 2-Axis Parallelogram.....	267
18.7 Robot Type 13, 2-Axis Parallelogram.....	269
18.8 Robot Type 18, Contact Kinematics.....	271
18.9 Robot Type 19, 2-Axis Parallelogram V1.....	272
18.10 Robot type 20, 3-Axis Delta with 1 Optional Manual Axis,.....	274
18.11 Robot Type 21, 2-Axis Parallelogram V2.....	280
18.12 Robot Type 26, Robot with Offset Wrist with Six (Seven) Axes.....	282
18.13 Robot Type 27, Robot with Offset Wrist with Six Axes + Three Linear Axes.....	287
18.14 Robot Type 29, 2-Axis Delta with Rotary and Translatory Axis.....	294
18.15 Robot Type 30, Adabot.....	296
18.16 Robot Type 32, 2-Axis Delta with Asymmetric TCP.....	299
18.17 Robot Type 100, PLC.....	301
 19 RCL Keywords.....	 303
 20 Service and Support.....	 313
 Index.....	 315

1 About this Documentation

1.1 Validity of the Documentation

Target group This documentation provides an overview on the RobotControl of the IndraMotion MLC.

The programming language RCL (RobotControl Language) is focused.

The following is described

- Program structure
- Variables, functions, motion statements
- Required parameters of the system
- Programming references for further information
- Transformation rules
- Features at PLC programming

This documentation supports the user during all phases

- Programming the RCL program
- Debugging and commissioning

1.2 Required and Supplementing Documentations

Documentation titles with type codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software Installation DOK-IWORKS-SOFTINS*V13-CORS-EN-P, R911336880 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APRS-EN-P, R911336870 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Functional Description DOK-MLC***-FUNC****V13-APRS-EN-P, R911336295 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Functional Description DOK-XLC***-FUNC****V13-APRS-EN-P, R911336352 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X

About this Documentation

IndraWorks		MLC	XLC
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APRS-EN-P, R911336356 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Project Conversion DOK-XLCMLC-PROCONV*V13-APRS-EN-P, R911336366 This documentation describes the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks version 12 with IndraLogic 2G. Changes with regard to Motion and PLC are described in detail.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Commissioning DOK-MLC***-STARTUP*V13-CORS-EN-P, R911336308 This documentation describes the steps to commission and service the IndraMotion MLC system. It includes checklists for frequent tasks and a detailed description of the steps.	X	

Fig. 1-1: XCL/MLC documentation overview

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

About this Documentation

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

Fig. 1-2: XCL/MLC documentation overview

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

Fig. 1-3: XCL/MLC documentation overview

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

Fig. 1-4: XCL/MLC documentation overview

About this Documentation

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

Fig. 1-5: XCL/MLC documentation overview

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

Fig. 1-6: XCL/MLC documentation overview

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

About this Documentation

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

Fig. 1-7: XCL/MLC documentation overview

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

Fig. 1-8: XCL/MLC documentation overview

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

About this Documentation

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

Fig. 1-9: XCL/MLC documentation overview

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

Fig. 1-10: XCL/MLC documentation overview

1.3 Information Representation

1.3.1 Safety Instructions

If there are safety instructions, they contain certain signal words (Danger, Warning, Caution, Notice) and if applicable, signal alert symbols (acc. to ANSI Z535.6-2006).

The signal word draws the attention to the safety instruction and indicates the risk potential.

The signal alert symbol (warning triangle with exclamation mark) positioned in front of the signal words Danger, Warning and Caution indicates hazards for individuals.

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 or property damage can occur.

1.3.2 Symbols Used

Note Notes are represented as follows:



This is a note for the user.

Tip Tips are represented as follows:



This is a tip for the user.

1.3.3 Names and Abbreviations

Term	Explanation
JCS	Joint Coordinate System
BCS	Basic Coordinate System
RCL	Programming language: Robot Control Language
TCP	Tool Center Point
WCS	World Coordinate System
WpCS	Workpiece Coordinate System

Fig. 1-11: Names and abbreviations used

2 Coordinates and Coordinate Systems

2.1 General Information

In robotics, terms such as robot axes, coordinate axes and coordinate systems are central terms.

Therefore, the name specification of robot axes, the reference of coordinate systems and the use of coordinate transformations have significant importance.

In the following, the coordinate systems supported in the RobotControl are introduced.

In general, coordinate axes are Cartesian coordinates allocated in a right angle to each other. The positive directions of the coordinate axes are determined by a right-handed, rectangular coordinate system.

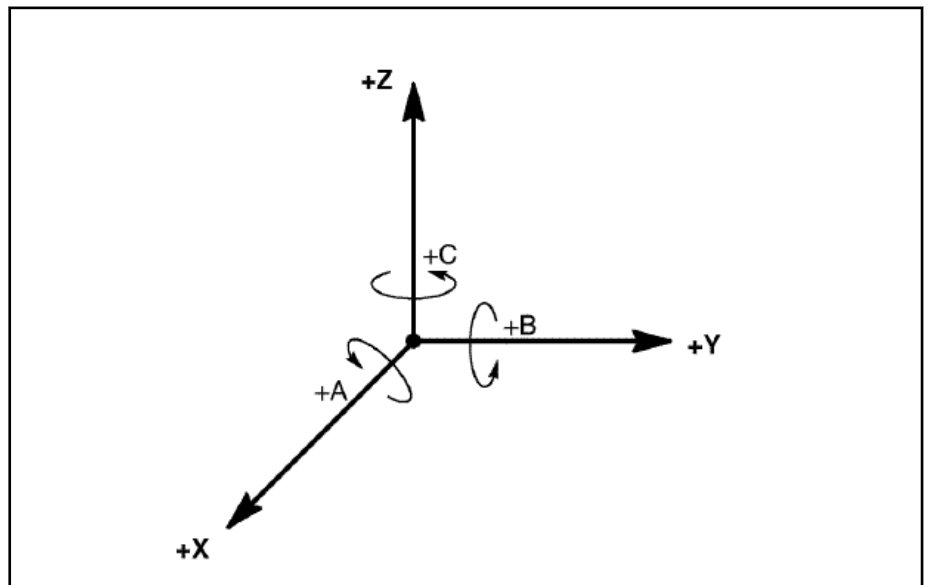


Fig. 2-1: Cartesian coordinate system, axis name

To freely position and orient an object in a Cartesian coordinate system, all six degrees of freedom are generally required. The position consists of three degrees of freedom of the positioning and three degrees of freedom of the orientation.



The allocation of the individual robot axes towards each other cannot be rectangular even in actual usage. Thus, the result is a non-orthogonal joint coordinate system.

The following coordinate systems are currently supported in the RobotControl:

Overview on coordinate systems

- [Joint Coordinate System \(JCS\), page 20,](#)
- [Basic Coordinate System \(BCS\), page 20,](#)
- [World Coordinate System \(WCS\), page 22,](#)
- [Workpiece Coordinate System \(WpCS\), page 23](#)
- [Tool Coordinate System \(TCS\), page 24](#)

2.2 Joint Coordinate System (JCS)

The joint coordinate system (JCS) of a robot is defined by the entity of all positioning and orientation axes assigned to one kinematics.

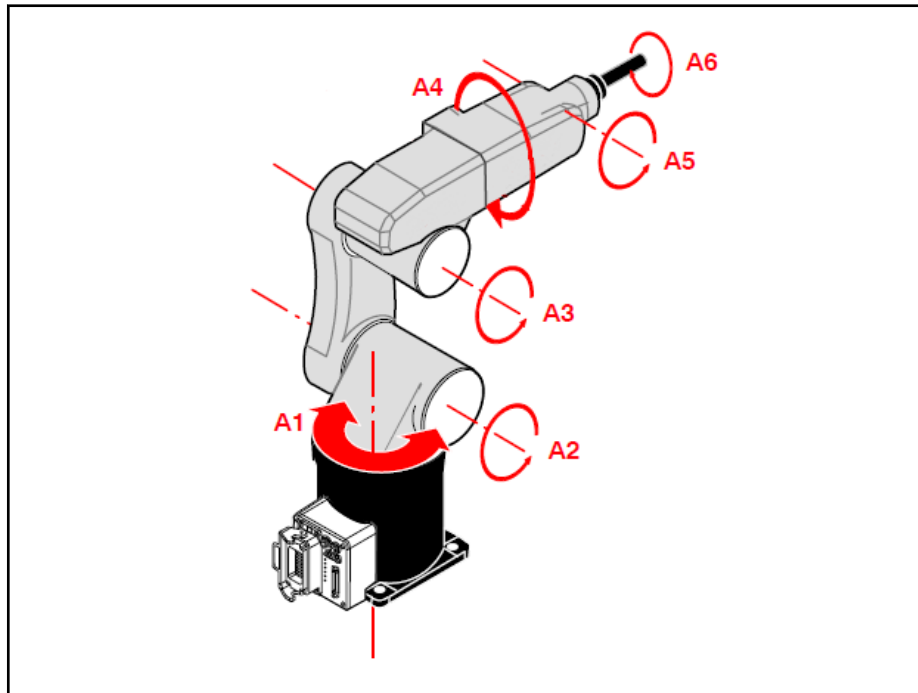


Fig. 2-2: Joint coordinates (A1,A2,A3,A4,A5,A6)

To setup the joint coordinate system, the robot axes are brought into zero positions. These can be found in the transformation guideline of the respective kinematics. Subsequently, the position measuring systems have to be referenced.

The basic requirements to perform the coordinate transformation correctly are:

- Exact settings of the axis zero positions
- Exact measurement of the arm lengths or center offsets
- exact determination of the coupling factors (if present)
- Correct directions of rotation of the axes



If one of these basic requirements is not fulfilled completely, a precise robot motion cannot be guaranteed. In worst case, this can cause incorrect and unexpected robot motions.

2.3 Basic Coordinate System (BCS)

The basic coordinate system (BCS) is a Cartesian coordinate system fixedly connected to the robot.

Coordinates and Coordinate Systems

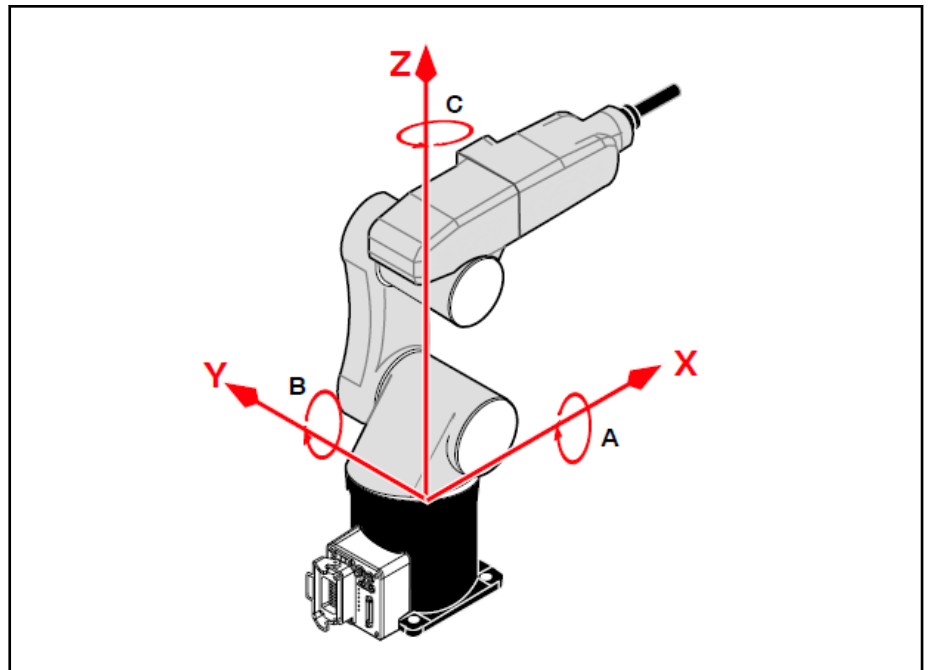


Fig. 2-3: Basic coordinates ($X_0, Y_0, Z_0, A_0, B_0, C_0$)

If the kinematic structure of the robot is not Cartesian, the relation between the joint and the Cartesian basic coordinates is established via a transformation.

Schematic representation of a kinematic transformation

Forward position kinematics/direct kinematic problem

If the current position of the tool working point has to be determined in space, the easiest way is to determine the joint coordinates and use these to calculate the position and orientation of the tool in the basic coordinate system.

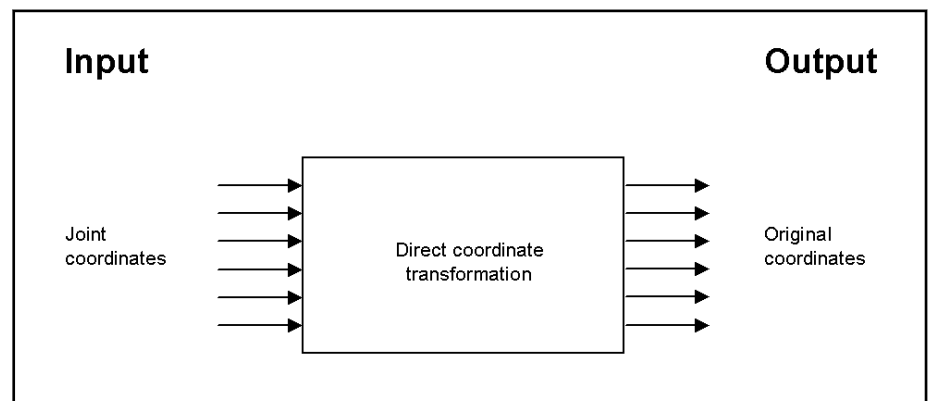


Fig. 2-4: Forward position kinematics

Inverse position kinematics

If the position of the tool working point is given by the position and the orientation in the basic coordinate system, this information is used to calculate the respective joint coordinates.

Coordinates and Coordinate Systems

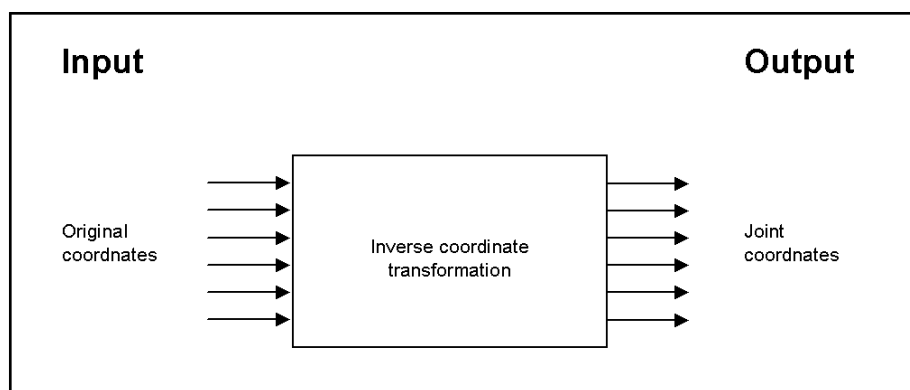
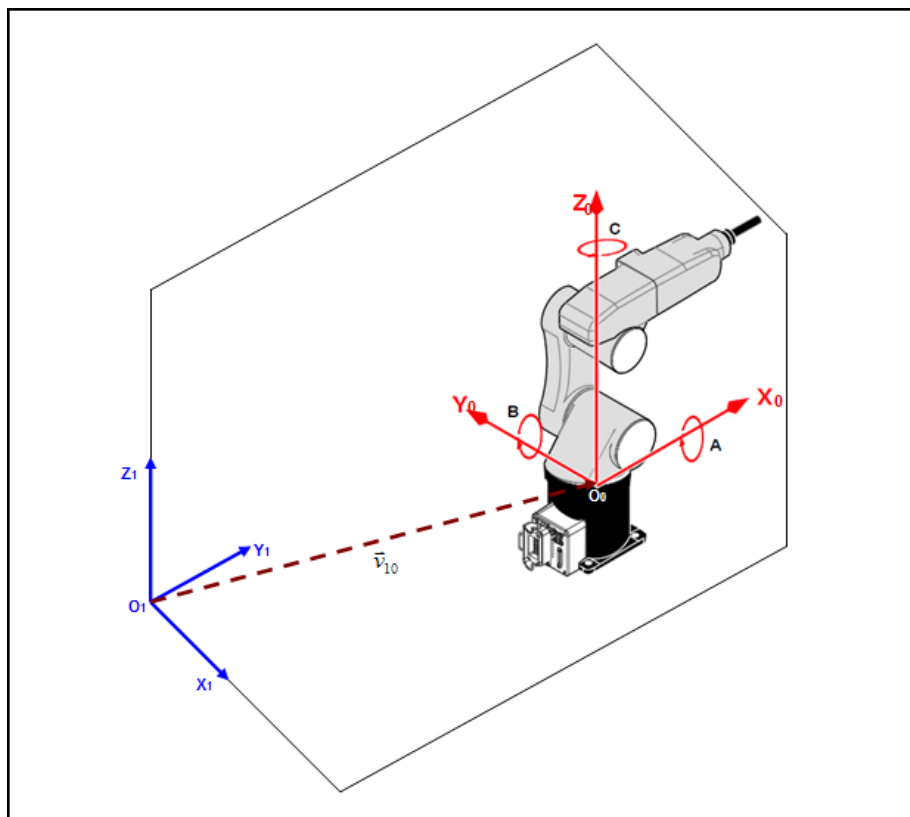


Fig.2-5: Inverse position kinematics

2.4 World Coordinate System (WCS)

The world coordinate system (WCS) is a coordinate system fix in location. The world coordinate system mostly orients at a distinctive point of the work cell of the robot.

Fig.2-6: World coordinates ($X_1, Y_1, Z_1, A_1, B_1, C_1$)

In practice, several robots are mounted within one work cell. If the workspaces of the respective robots overlap, the motion of the respective kinematics has to be controlled with regard to each other.

Therefore, the respective motions should be shown in the same coordinate system.

2.5 Workpiece Coordinate System (WpCS)

The workpiece coordinate system (WpCS = **W**orkpiece**C**oordinate **S**ystem) has a fixed position and orientation with regard to the world coordinate system.

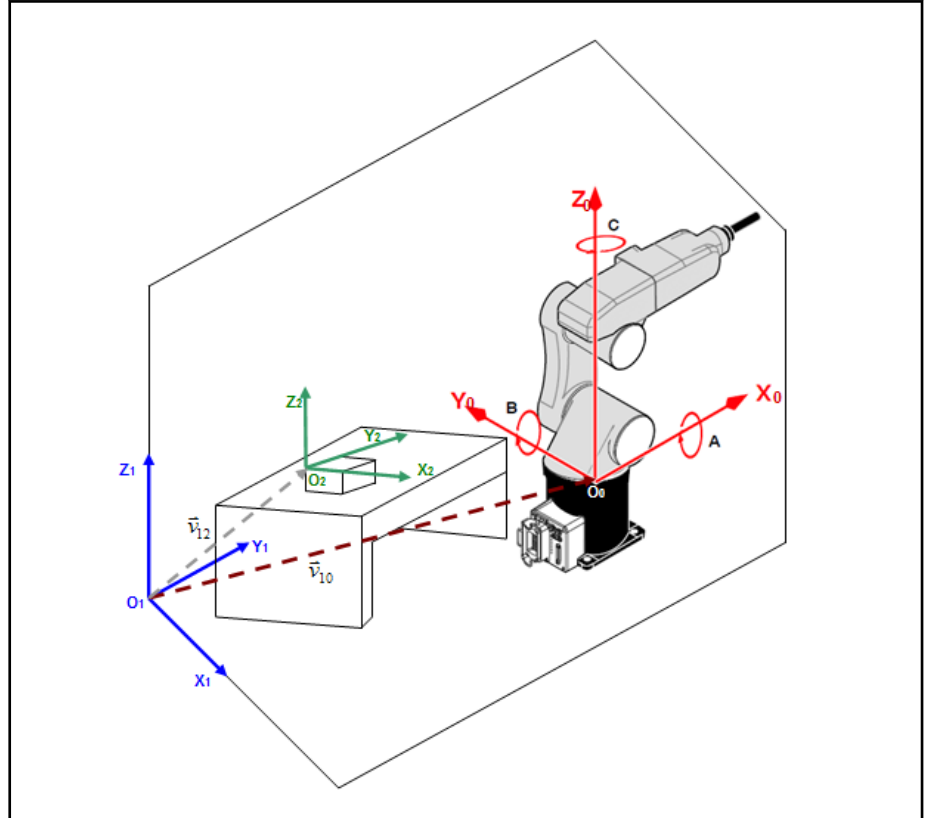


Fig.2-7: Workpiece coordinate system ($X_2, Y_2, Z_2, A_2, B_2, C_2$)

The workpiece coordinate system has special significance when programming machining tasks.

The workpiece coordinates can be easily used by the software engineer to exactly approach the different machining positions at the workpiece.



All programmed motions in the RCL program refer to the workpiece coordinate system.

The programmed coordinates are approached with the TCP (TCP = Tool Center Point).

The TCP is in the tool working point.

The workpiece coordinate system corresponds to the world coordinate system by default.

Coordinates and Coordinate Systems

2.6 Tool Coordinate System (TCS)

The tool coordinate system (TCS) is mounted at the flange of the tool (gripper) and always moves along with the tool.

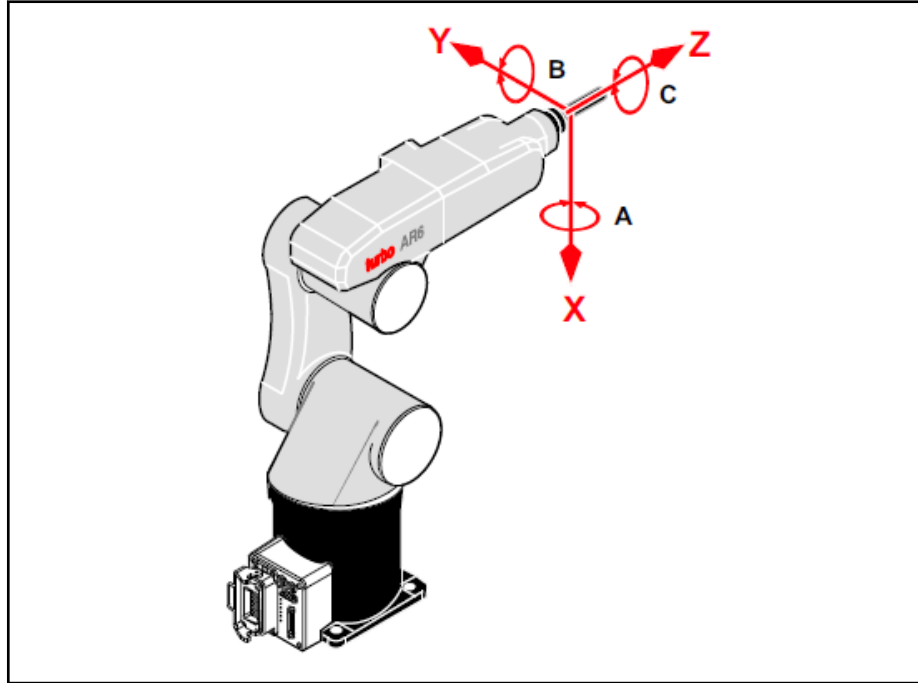


Fig.2-8: Tool coordinates (X, Y, Z, A, B, C)

The three axes of the TCS are virtually tightly connected to the tool and move along with every offset and rotation.



All programmed motions in the RCL program refer to the tool coordinate system.

The programmed coordinates are approached with the TCP (TCP = Tool Center Point).

The TCP is in the tool working point.

3 Using Coordinate Systems

3.1 Defining Coordinate Systems

The transition between coordinate systems, that is between world coordinates and basic coordinates or world coordinates and workpiece coordinates is carried out via transformations in the control.

The user is provided with the following options to define the coordinate systems.

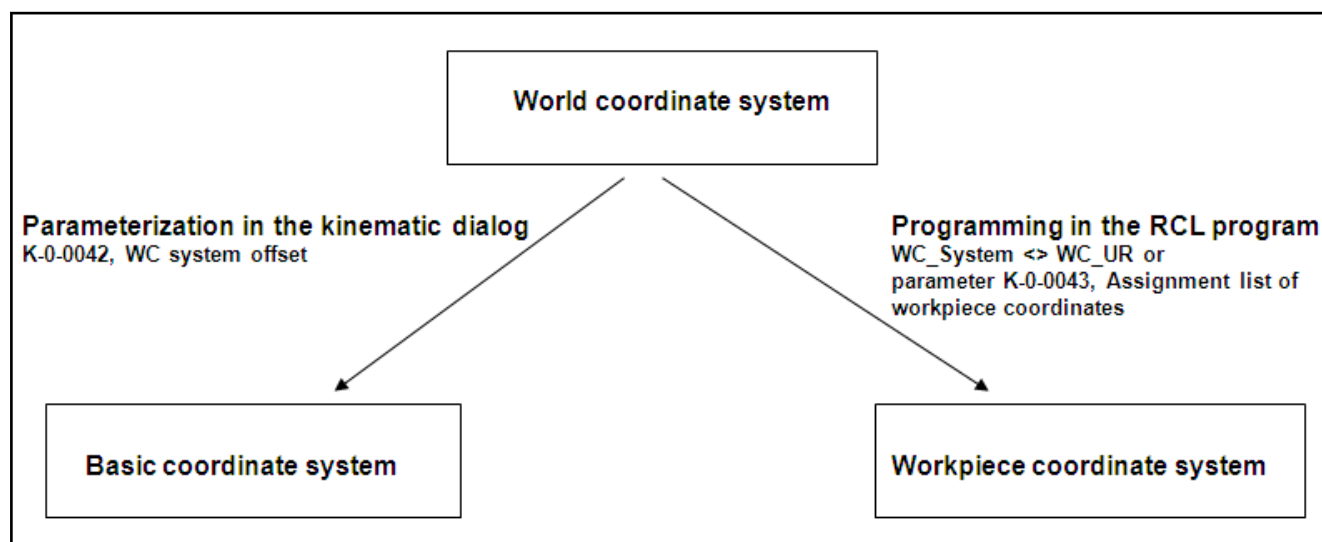


Fig.3-1: Overview on coordinate systems

3.2 Defining the Basic Coordinate System Relative to the World Coordinate System

Describing the function The relation between the world coordinate system and the basic coordinate system is described via the kinematic parameter K-0-0042, WC system offset.

The parameter K-0-0042 consists of six elements. Three elements are used for positioning and three elements for orienting the basic coordinate system relative to the world coordinate system.

Consider that the number of degrees of freedom depends on the kinematics used.

The **offset vector** starts at the origin of the world coordinate system and ends at the origin of the basic coordinate system.

The rotation of the coordinate system is only possible for particular transformation schemes (see K-0-0031). This is documented in the respective descriptions. (No note on the rotation of the coordinate system means that this option is not supported with this transformation schema).

Relevant parameters

K-0-0042	WC system offset
K-0-0031	Transformation scheme

The parameters are described in detail in [RCL Parameters, page 157](#).

Activate

The position of the basic coordinate system is determined with regard to the world coordinate system.

Using Coordinate Systems

Therefore, the offset vector has to be entered into the first three elements.

The orientation is determined by writing on the last three elements.

The offsets are specified in mm and the rotations in degrees.

To write on the parameter K-0-0042, the kinematic dialog "Coordinate offset" or the parameter editor can be used.



The parameters are applied when the phase is switched from P2 to BB.

According to the definition of the basic coordinate system relative to the world coordinate system, the coordinates refer in the KinData interface

`KinData[x].arPositionOriginalCoord_i`

refer to the world coordinate system.

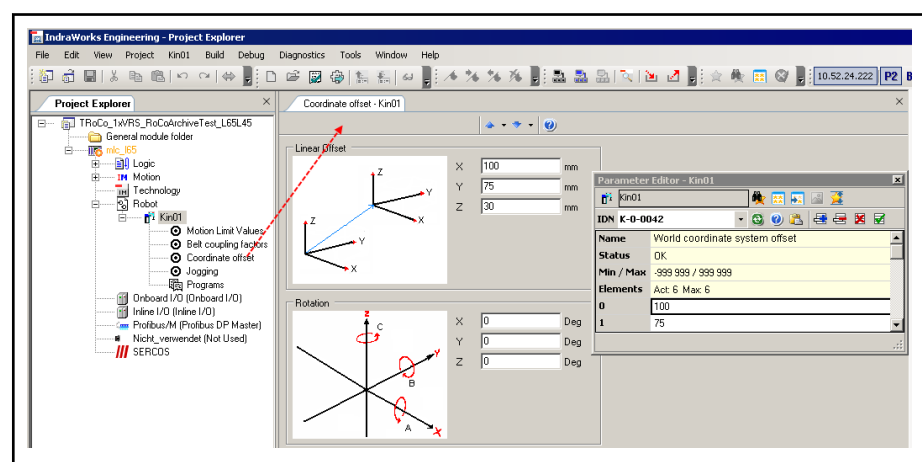


Fig.3-2: Options to activate the basic coordinate system

Deactivate 0 has to be assigned to all elements of the parameter K-0-0042, WC system offset.

After the phase switching from P2 to BB, the position of the basic coordinate system is identical to the position of the world coordinate system. Previously applied offsets and rotations are canceled.

After switching off the translation and rotation of the basic coordinate system, the coordinates in the KinData interface

`KinData[x].arPositionOriginalCoord_i`

remain related to the world coordinate system that is now identical to the basic coordinate system.

Using Coordinate Systems

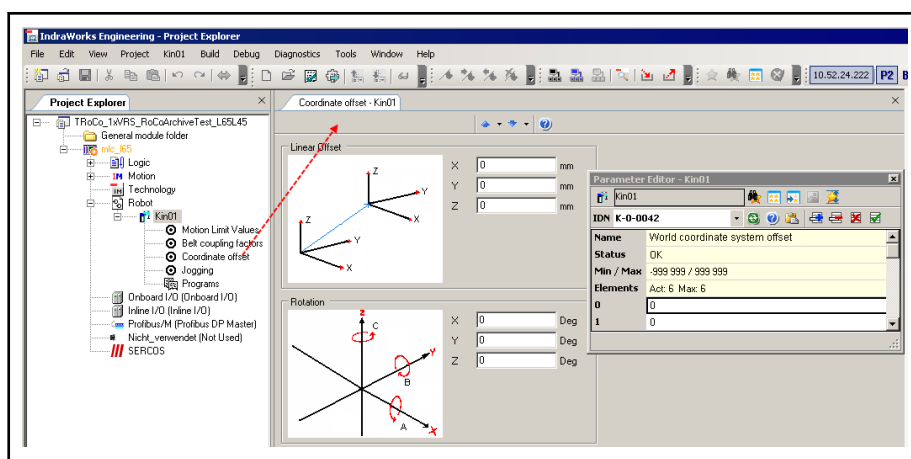


Fig.3-3: Options to deactivate the basic coordinate system

Restrictions

It has to be considered that the values of the parameter K-0-0042 become only active when switching the phase from P2 (P2 = Parameterization mode) to BB (BB = Ready for operation).

3.3 Defining the Workpiece Coordinate System Relative to the World Coordinate System

Describing the function

The relation between the workpiece coordinate system and the world coordinate system is described via the RCL command **WC_SYSTEM**. This command consists of six elements. Three elements are used to position the workpiece coordinate system and three elements to orientate the workpiece coordinate system. The option to apply rotations depends on the transformation scheme. Consider that the number of degrees of freedom depends on the kinematics used.

The translation vector starts at the origin of the world coordinate system and ends at the origin of the workpiece coordinate system.



The workpiece coordinate system can only be generally used for 6-axis kinematics. If the kinematics is provided with less than six degrees of freedom, the axes to be considered have to be specified via the parameter K-0-0043. In this case, the workpiece coordinate system can be shifted arbitrarily in all three directions, but only rotated around by one axis max.



The offsets are specified in mm and the rotations in degrees.

Relevant RCL commands

WC_SYSTEM is a kinematic-related default variable of the type **WC_FRAME**. The valid workpiece coordinate system is communicated and activated via this standard variable.

WC_FRAME is a kinematic-related default type consisting of six components of the type **REAL**. The six components describe the offset and the rotation of the workpiece coordinate system compared to the world coordinate system.

The sequence of the components is as follows:

1. Component	deltaX	x-direction
2. Component	deltaY	y-direction
3. Component	deltaZ	z-direction

Using Coordinate Systems

4. Component	deltaO1	First orientation axis
5. Component	deltaO2	Second orientation axis
6. Component	deltaO3	Third orientation axis

The special definition and sequence of the three orientations depend on the type of robot and eventually the orientation definition selected there (cf. documentation on the respective transformation rules).

WC_UR is a predefined constant of the type **WC_FRAME**. All its six components have a value of 0. If this constant is assigned to **WC_SYSTEM**, all offsets and rotations with regard to the world coordinate system become zero.

Activate

Variables, declaration, assignment

```
<KinematicsName>.WC_FRAME : <WC-VarName>
<KinematicsName>.WC_SYSTEM = <WC-VarName>
```

<WC-VarName> is the freely-selectable name of the workpiece coordinate system. Values can now be assigned to the declared variable <WC-VarName> of type **WC_FRAME** in the RCL program. After valid values have been set, the variable <WC-VarName> has to be assigned to the command **WC_SYSTEM**. By processing the command via block processing, the workpiece coordinate system is defined.

Transformation world coordinate system → Workpiece coordinate system

```

;-----
; Transformation world coordinate system -> Workpiece coordinate system
;-----
; Declaration of different workpiece coordinate systems
WC_FRAME      : pallet1, pallet2
REAL          : Comp1, Comp2, Comp3, Comp4, Comp5, Comp6
BEGIN
; Definition of the different workpiece coordinate systems
Pallet1 = WC_FRAME(100,100,0,0,0,0)
  Comp1=50
  Comp2=50
  Comp3=75
Pallet2 = WC_FRAME(Comp1,Comp2,Comp3,Comp4,Comp5,Comp6)
; Activating the workpiece coordinate system of Pallet1
WC_SYSTEM = Pallet1

```

The coordinates in the KinData interface

`KinData[x].arPositionWorkpieceCoords_i`

refer to the workpiece coordinate system defined relatively to the world coordinate system with pallet 1.

Deactivate

Deactivating the workpiece coordinate system

```
WC_SYSTEM = WC_UR
```

With this assignment, 0 is assigned to all six components of the **WC_SYSTEM** command. After deactivation, the workpiece coordinate system is identical to the world coordinate system.

The coordinates in the KinData interface

`KinData[x].arPositionWorkpieceCoords_i`

are equal to the coordinates

`KinData[x].arPositionOriginalCoords_i`.

Restrictions

A rotation of the workpiece coordinate system around a non-existent kinematic axis is not supported.

3.4 Workpiece Coordinate File

Describing the function

Alternatively to direct programming, there is the ASCII system file WCSYST.DAT in which all local workpiece coordinates are defined by name

Create

The file - if not available - can be created via the context menu of the "Global Data" folder. The "Global Data" folder is located below the robot node in the Project Explorer.

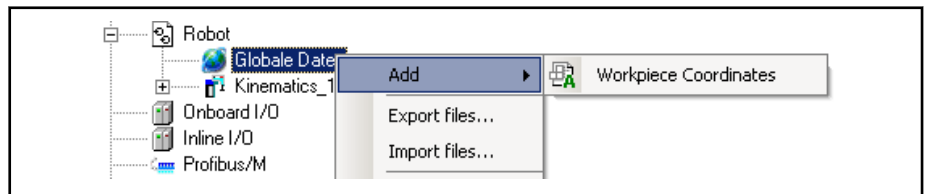


Fig. 3-4: Creating workpiece coordinate file

The "Global Data" folder can be deleted at any time and created again via the context menu of the "Robot" node.



When deleting the "Global Data" folder, all files stored in this folder are also deleted!

Edit

The editor to enter the input or to edit the workpiece coordinates can be opened via the context menu of the "WCSYST.DAT" file

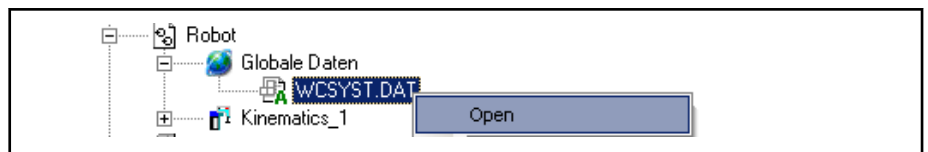


Fig. 3-5: Opening workpiece coordinate file

Activate

The workpiece coordinate file is used with the library functions.

3.5 Workpiece Coordinate Editor

General Information

Workpiece coordinates can be entered and edited with the editor.

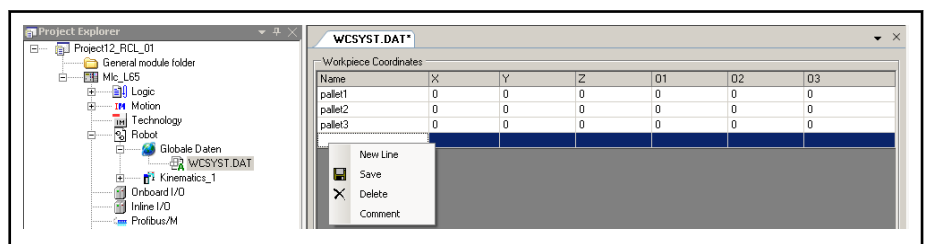


Fig. 3-6: Editor

Any name can be specified for the individual world coordinates. The respective coordinate values are stored under this name in the "WCSYST.DAT" file.



The name may not exceed 12 characters and has to start with a letter.

For coordinate values, only the entries "0" ... "9", ".", ",", "+", and "-" are allowed.

Using Coordinate Systems

The editor provides the following functions:

- Adding new line
- Saving workpiece coordinates,
- Deleting line
- Entering line comment

3.6 Library Functions

The class 2000 of the RCL library functions includes library functions facilitating the general use of world coordinate systems. For a detailed description of the parameter lists, refer to the "rk_En.inc" header file. To use the library functions, include the header file into the RCL program.

rKxGWcSystem

Provides name and coordinates of the active workpiece coordinate system. A structure of type "TKxGWcSystem" has to be transferred to the function. This structure looks as follows:

Transfer parameter "TKxGWcSystem" of the "rKxGWcSystem()" function

Name	Type	Data type	Meaning
ChannelId	Input	INTEGER	0 has to be set as channel ID.
KinNr	Input	INTEGER	Kinematic number Integers from 1 to n are allowed. The maximum kinematic number allowed depends on the setting of the parameter C-0-2484
WcSysCoord	Output	FARRAY_WC	Array contains six coordinate values of the workpiece coordinate system
WcSysName	Output	NAMSTRING	Name of the active workpiece coordinate system
Reserved	Reserved	ARRAY[1..32] INTEGER	Reserved

Fig.3-7: Transfer parameter of the "rKxGWcSystem()" function

Possible return values of the function

Value	Meaning
RK_NO_ERROR	No error. The function was executed correctly
RK_KIN_NR	Invalid kinematic number

Fig.3-8: Possible return values of the function

rKSWcSystem

Activates the workpiece coordinate system selected by name. Name and respective coordinates values have to be defined in the system file "WCSYST.DAT".

Using Coordinate Systems

CAUTION

When using this function, the kinematic properties are changed (Tool Center Point).

The user is responsible for taking the relevant safety measures and to comply with the applicable accident prevention regulations in order to protect individuals and equipment from any kind of hazard. The user is solely responsible for the appropriate usage.

A structure of type "TKSWcSystem" has to be transferred to the function. This structure looks as follows:

Transfer parameter "TKSWcSystem" of the "rKSWcSystem()" function

Name	Type	Data type	Meaning
ChannelId	Input	INTEGER	0 has to be set as channel ID.
KinNr	Input	INTEGER	Kinematic number Integers from 1 to n are allowed. The maximum kinematic number allowed depends on the setting of the parameter C-0-2484
WcSysName	Input	NAMSTRING	Name of the workpiece coordinate system to be activated The function searches for the name in the WCSYST.DAT file
Reserved	Reserved	ARRAY[1..32] INTEGER	Reserved

Fig.3-9: Transfer parameter of the "rKSWcSystem()" function

Possible return values of the function

Value	Meaning
RK_NO_ERROR	No error. The function was executed correctly
RK_KIN_NR	Invalid kinematic number
RK_HEAPOVERF	Heap overflow
RK_TOOL_FILE	Error in WCSYST.DAT: Name of the workpiece coordinate system missing in "WCSYST.DAT" or "WCSYST.DAT"
RK_WRONGK43	Invalid machine parameter K-0-0043

Fig.3-10: Possible return values of the function

rKCWpFrame

Calculated from the three positions P0, P1, P2, accepted in coordinates of the original coordinate system WC_{or} , the respective zero point of the local WC system WC_{loc} including rotation to the original coordinate system.



The function can only be used for kinematics with all six degrees of freedom. That means kinematics with at least six axes. Otherwise, the problem is mathematically underdetermined and output as return code (RK_AX_NR = -2).

The following prerequisites have to be met for the calculation:

Using Coordinate Systems

- The three points P0, P1 and P2 are located on one pallet level. That means that they have the same z_{loc} coordinate.
- The vector P0P1 is parallel to the x-axis.
- The coordinates (x, y, z) of point P0 in the local WC system WC_{loc} have to be known.

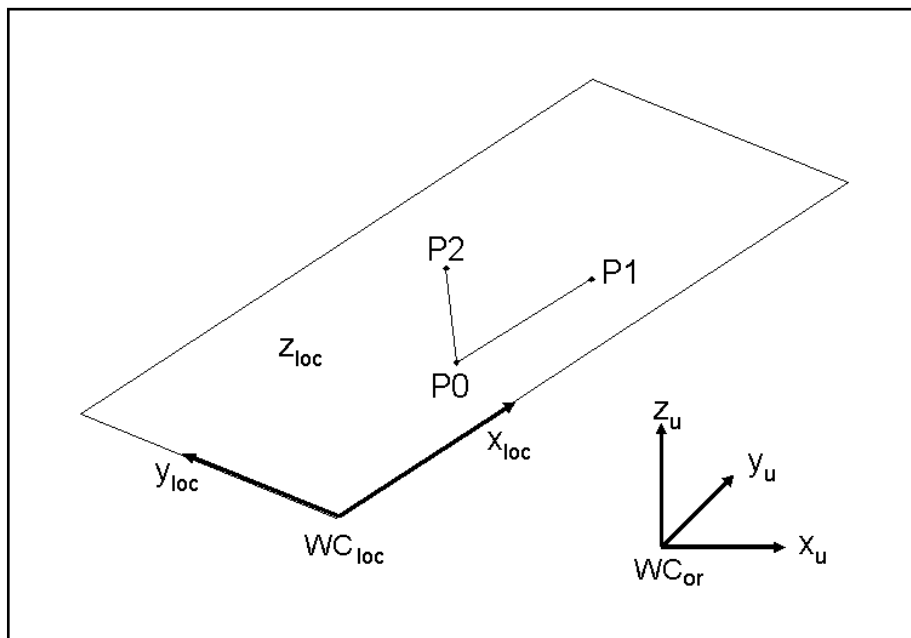


Fig.3-11: Parameters of the "rKCWpFrame" function shown schematically

Transfer parameter "TKCWpFrame" of the "rKCWpFrame()" function

Name	Type	Data type	Meaning
ChannelId	Input	INTEGER	0 has to be set as channel ID.
KinNr	Input	INTEGER	Kinematic number Integers from 1 to n are allowed. The maximum kinematic number allowed depends on the setting of the parameter C-0-2484
P0loc	Input	FARRAY_AXIS	Position of point P0; local workpiece coordinates
P0base	Input	FARRAY_AXIS	Position of point P0, robot original coordinates
P1base	Input	FARRAY_AXIS	Position of point P1, robot original coordinates
P2base	Input	FARRAY_AXIS	Position of point P2; robot original coordinates

Using Coordinate Systems

WpFrame	Output	FARRAY_WC	Coordinates of the local workpiece coordinate system. The array contains six values writing the position (three values) and the orientation (three values) of the local workpiece coordinate system in units of the robot original coordinate system. The basic level of the workpiece coordinate system is spanned by the three points P0, P1 and P2.
Reserved	Reserved	ARRAY[1..32] IN-TEGER	Reserved

Fig. 3-12: Transfer parameter of the "rKCWpFrame()" function

Possible return values of the function

Value	Meaning
RK_NO_ERROR	No error. The function was executed correctly
RK_KIN_NR	Invalid kinematic number
RK_AX_NR	Kinematics is not provided with all six degrees of freedom
RK_ANGLENOTC	Angles cannot be calculated. The points P0, P1, P2 are not suitable

Fig. 3-13: Possible return values of the function

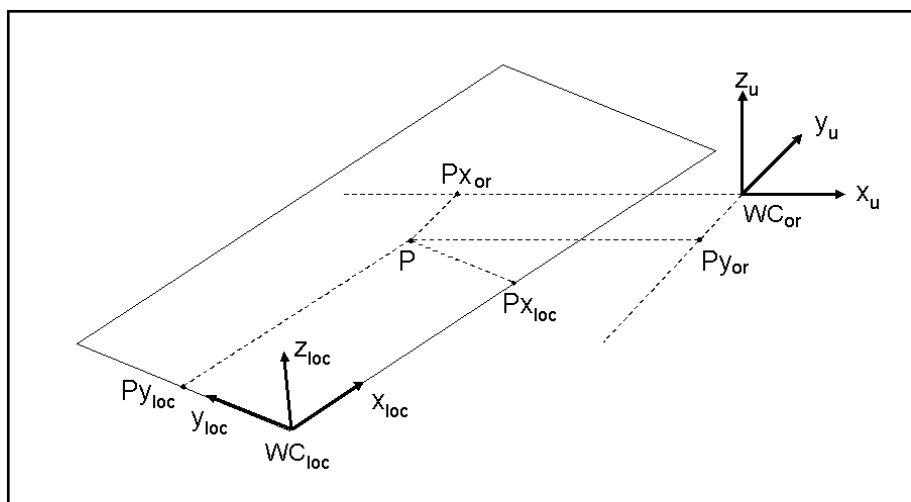
rKTrfLocOri Calculates the respective original coordinates from the local coordinates of a point.



The library function can only be used for 6-axis kinematics. The function is underdetermined at missing degrees of freedom.

The function works name-oriented with regard to the local workpiece coordinate system. The name with the respective coordinates is expected in the system file WCSYST.DAT. P_{loc} corresponds to the point in the coordinates of the local workpiece coordinate system. P_{or} is the same point in the coordinates of the original coordinate system WC_{or} .

Using Coordinate Systems

Fig.3-14: Point P in the coordinate system WC_{loc} and WC_{or} shown schematically**Transfer parameter "TKTrfLocOri" of the "rKTrfLocOri()" function**

Name	Type	Data type	Meaning
ChannelId	Input	INTEGER	0 has to be set as channel ID.
KinNr	Input	INTEGER	Kinematic number Integers from 1 to n are allowed. The maximum kinematic number allowed depends on the setting of the parameter C-0-2484
PntLoc	Input	FARRAY_AXIS	Point in the coordinates of the local workpiece coordinate system "WcSysName"
WcSysName	Input	NAMSTRING	Name of the local workpiece coordinate system The function searches for the name in the WCSYST.DAT system file
PntOri	Output	FARRAY_AXIS	Respective original coordinates of the point "PntLoc". That means transformation of the point "PntLoc" to the original system of the robot.
Reserved	Reserved	ARRAY[1..32] INTEGER	Reserved

Fig.3-15: Transfer parameter of the "rKTrfLocOri()" function

Possible return values of the function

Value	Meaning
RK_NO_ERROR	No error. The function was executed correctly

Using Coordinate Systems

RK_KIN_NR	Invalid kinematic number
RK_TOOL_FILE	Error in WCSYST.DAT: Name of the workpiece coordinate system missing in "WCSYST.DAT" or "WCSYST.DAT"

Fig. 3-16: Possible return values of the function

rKTrfOriLoc

Calculates the respective local coordinates from the original coordinates of a point.



The library function can only be used for 6-axis kinematics. The function is underdetermined at missing degrees of freedom.

The function works name-oriented with regard to the local workpiece coordinate system. The name with the respective coordinates is expected in the system file WCSYST.DAT. P_{loc} corresponds to the point in the coordinates of the local workpiece coordinate system. P_{or} is the same point in the coordinates of the original coordinate system WC_{or} .

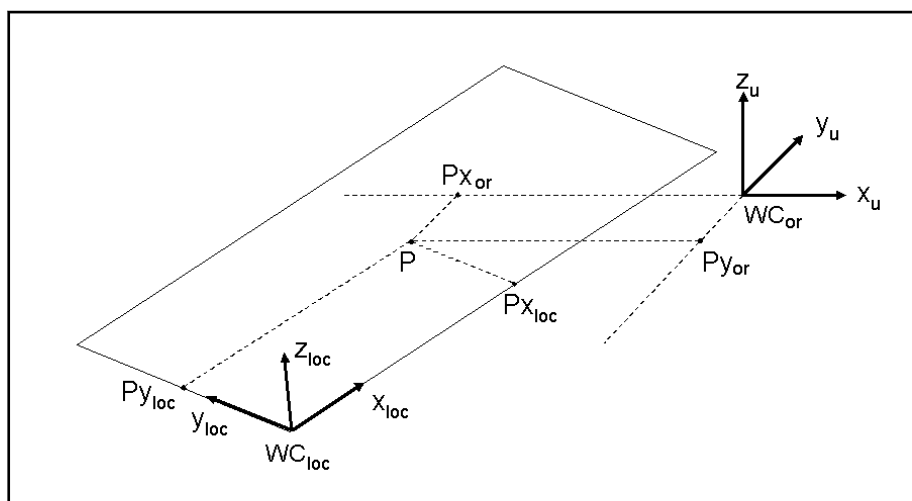


Fig. 3-17: Point P in the coordinate system WC_{loc} and WC_{or} shown schematically

Transfer parameter "TKTrfOriLoc" of the "rKTrfOriLoc()" function

Name	Type	Data type	Meaning
ChannelId	Input	INTEGER	0 has to be set as channel ID.
KinNr	Input	INTEGER	Kinematic number Integers from 1 to n are allowed. The maximum kinematic number allowed depends on the setting of the parameter C-0-2484
PntOri	Input	FARRAY_AXIS	Point in original coordinates of the robot
WcSysName	Input	NAMSTRING	Name of the local workpiece coordinate system The function searches for the name in the WCSYST.DAT system file

Using Coordinate Systems

PntLoc	Output	FARRAY_AXIS	Respective coordinates of the point "PntOri" in values of the workpiece coordinate system. That means transformation of the point "PntOri" to the local workpiece coordinate systems.
Reserved	Reserved	ARRAY[1..32] INTEGER	Reserved

Fig.3-18: Transfer parameter of the "rKTrfOriLoc()" function

Possible return values of the function

Value	Meaning
RK_NO_ERROR	No error. The function was executed correctly
RK_KIN_NR	Invalid kinematic number
RK_TOOL_FILE	Error in WCSYST.DAT: Name of the workpiece coordinate system missing in "WCSYST.DAT" or "WCSYST.DAT"

Fig.3-19: Possible return values of the function

3.7 Defining the Tool Coordinate System

Describing the function

To work with different tools (grippers) in the same world coordinate system, the coordinate transformation is split into several parts.

(1) The first part considers the robot kinematics to the flange via the machine parameters. This part determines the robot type, number of axes, arm lengths and coupling factors.

(2) The second part considers the kinematics of the tool from the flange to the tool center point. There are the following tool definition options:

Several tools can be defined via the TOOLS.DAT file. After control startup, none of these tool coordinate systems is active, the parameter K-0-0140 query can result in "none active". The coordinate systems can be enabled via an RCL program with the TOOL command or the parameter K-0-0045.

Three translations

(X, Y, Z)

and three rotations

(O1, O2, O3) are used for the general description.

The zero point of the tool coordinate system is determined by (1).



The alignment of the tool coordinate system and the definition of the orientations depend on the robot type. For details, refer to the respective transformation manual.

Structure of the TOOLS.DAT file

As mentioned above, all possible tool coordinate systems are stored in the TOOLS.DAT file.

Any name can be specified for the individual tools. The corresponding coordinates are saved under this name.

The tool name has to be at the beginning of the line and may not exceed twelve characters and may not contain any special characters. It can be freely selected.

Using Coordinate Systems

The sequence of the individual coordinates is as follows: first X, then Y ... and finally O3. This sequence may not be changed.

The individual coordinate specifications are decimal values. The decimal point does not have to be set. Only "0", "1" to "9", "+", "-", ",", and "." are valid entries for the coordinate values.

The translations (the first three words) are entered in [mm] and the rotations (the last three values) in [degree].

An example for TOOLS.DAT that can be edited in IndraWorks under "Robot Global Data":

Name	X	Y	Z	O1	O2	O3
gripper1	10	0	0	0	0	0
nogripper	0	0	0	0	0	0

Example of TOOLS.DAT

Activate

A tool is selected in the RCL program using the command TOOL kin1 grf_name

kin1 = Name of the kinematics for which the tool is activated.

grf_name = Name of the tool (gripper) to be activated is defined in TOOLS.DAT.

A tool change is only possible in automatic mode.

A gripper is effective until the next TOOL call.

The latest programmed tool remains active after the program end.

When the program is aborted (e.g. E-STOP input, Switching "automatic/manual", etc.) the tool active at the point of time of the abort remains active.

No tool is active after startup.



Ensure upon program start that the correct tool is activated. If programs run with incorrect tools, unexpected motions can occur. The same effect can be caused if a WCS point is approached and taught with different tools.

Deactivate

There is no direct command to switch off the gripper or the tool does not exist.

The desired effect can be implemented easily by defining a dummy gripper with the name "nogripper" and the corresponding coordinates X = ... = O3 = 0 in Werkzg.dat for example (see example). The RCL command "TOOL nogripper" causes the switch-off.

Alternative use of the parameter K-0-0045

Input of the tool name as string in the phase BB.

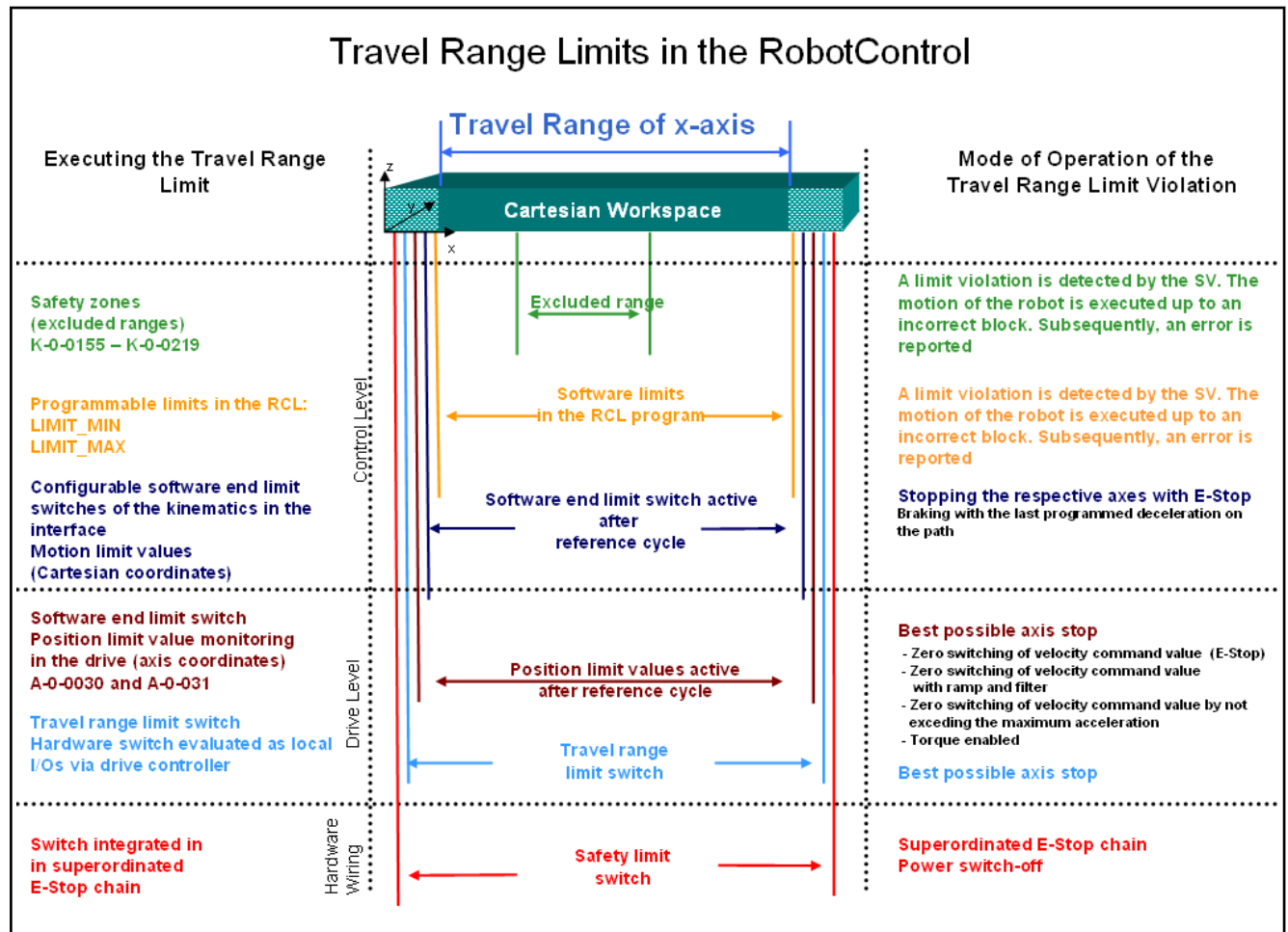
This allows the use of the functionality in the PLC without using the TOOL command in the RCL.

Error messages

"WCSYST/TOOLS.DAT missing" (display no.: F12C042A)

"WCSYST/TOOL name missing" (display no.: F12C042B)

4 Travel Range Limits in the RobotControl



SV
 Fig.4-1: Block preparation
 Travel range limits

5 RCL Program Structure

5.1 General Information

RCL stands for "Robot Control Language". RCL is a task-oriented program language used for programming the MLC control family.

As a programming language for robot and handling systems, RCL is an extensive, but easy to learn language. Tasks can be quickly and easily completed.

The language commands are available in English.

In this document, the general syntax of every statement is given before each detailed statement.

The following symbols are used for description purposes:

CAPITAL LETTERS	means part of the language element, i.e. an RCL keyword. It has to be written exactly like this
{ }	Can optionally be specified several times
[]	Can optionally only be specified once



Symbolic identifiers such as variable names may not exceed 12 characters.

The first character has to be a letter. Sequences of letters and digits are allowed. Umlauts are not allowed.

5.2 Mode of the Compiler Operation

5.2.1 Use

The RCL Compiler can be used in two different ways:

- An editor is provided in IndraWorks to create and compile an RCL program fast and easily.
- If a source file already exists (e.g. created with a text editor), the compiler can be started via the parameters K-0-0221 and K-0-1062.

5.2.2 Source Files

qll file This text file contains the RCL program code.
File name and program name have to be identical.

5.2.3 Output Files

The output files are created during the compilation. The name of the output files consists of the names of the qll file with the respective extension (e.g. "Prog1.qll" comes to "Prog1.ird").

ird file This file contains the program code to be processed by the control.
Additionally, the storage area required for the variables used in the program is determined. Storage location is also reserved in this file for point variables which are not declared with DEF and to which a value is assigned in the program.

This file is only generated if the program has been compiled without errors.

pkt file In this file, the storage area is reserved for point variables.
Storage is reserved for point variables

RCL Program Structure

- declared in the program with DEF.
 - not declared.
 - to which no value has been assigned in the program.
- sym file** This file contains information on the variable names used in the program.
This file is required for debugging.
- err file** This text file contains an overview on the errors that occurred during the compilation.

5.3 Compiler Statements

5.3.1 General Information

The qll file contains compiler statements. They control the compiler and reduce the time and effort required for writing.

The compiler statement always starts with two semicolons.

Compiler Statements:

```
;;COMPILER = RCL3 ; RCL Compiler Version
;;INCLUDE name ; optional
;;[kin_name.]INT = CIRCULAR | PTP | LINEAR
;;KINEMATICS: ({ integer_constant = kin_variable | , })
;;KINEMATICS = kin_variable
;;(kin_variable.JC_NAMES | WC_NAMES) = { name || , }
;;DEBUGINFO [+/-] ; optional
```

The compiler statement for the axis names and the target control has to be placed in front of the first source symbol, i.e. in front of the program declaration.

5.3.2 Kinematic Definition

In the MLC 04V12 or higher, up to 16 kinematics can be supported for each control instead of only one (see also [Using the RoCo Multi-Kinematic Technology, page 243](#)).

Each of these kinematics is controlled by its own program.

Only one kinematics should be assigned to each positioning and orientation axis.

Belt axes can be assigned to several kinematics.

Depending on the technology used, axes and belts can be transferred from one kinematics to another and they can be controlled.

5.3.3 World Coordinate Names (WC Name Definition)

World coordinate points contain the components for the position, the orientation and, if required, the belt coordinate(s). The component names can be defined.

Syntax

```
;;WC_NAMES=x_k,y_k,z_k,a_k ;WC_NAMES=wc_name,...
```

Example with kinematic names

```
;;Kin_1.WC_NAMES=x_k,y_k,z_k,a_k ;kinematics name.WC_NAMES=WC name,...
```



The coordinate declaration has to be in one line and may not be interrupted by a word wrap.

When creating a kinematics in IndraWorks, the names for the WC axes are entered. When a new qll file is generated, WC axes are automatically created.

They can be modified later on.

5.3.4 Axis Names (JC Name Definition)

The machine coordinate points contain the components for the individual axes and, if required, for the belt coordinate(s).

The axis names can be defined.

Syntax

```
;;JC_NAMES=a_1,a_2,a_3,a_4 ;JC_NAMES=JC name,...
```

Example with kinematic names

```
;;Kin_1.JC_NAMES=a_1,a_2,a_3,a_4
```



The axis name declaration has to be in one line and may not be interrupted by a word wrap.

When a new qll file is generated, the axis names are automatically accepted by IndraWorks.

They can be modified later on.

5.3.5 Including Files

The compiler statement `;;INCLUDE 'filename'` includes sections of source programs in the current program.

Example

```
;;INCLUDE rcl ;rcl.qll, File for automatic inclusion
;of the declaration during compilation
INPUT REAL: 1=grip_contact,4=measheight ;Declaration of inputs
INPUT: 11=gateswitch_1,15=li_barrier

OUTPUT REAL: 1=print,2=met_unit ;Declaration of outputs
OUTPUT: 7=alarm
```

The "rcl.qll" file contains the declarations of your inputs and outputs for example. These are defined with regard to their

- Data type
- Channel number
- Variable name of the signal

For details, refer to [PLC Communication Channels, page 70](#).



The rcl.qll file has to be in IndraWorks under "Robot Global Data".

RCL Program Structure

5.3.6 Selectable Extension in the Include Statement

The extension of include files can be freely selected. If only the file name is specified, the extension ".qll" is added due to compatibility purposes.

To include a file without extension, the file name has to end with a dot, refer to the "include statement" example below.

Syntax

```
;;INCLUDE filename      ;Compiler statement 'INCLUDE'
```

The following applies to the syntax above:

filename: File name to be included during compilation.

Example

```
;;INCLUDE head.inc
PROGRAM demo
;;INCLUDE constant      ;constant.qll file is included
;;INCLUDE types.INC     ;File types.INC are included
;;INCLUDE variable.     ;File variable. (without extension) is included
BEGIN

PROGRAM_END
```



The following files have to be stored in a directory: head.inc, demo.qll, constant.qll, types.inc and variable.



Include files cannot contain any INCLUDE commands (nesting is not possible).

5.3.7 Debug Information

For a more comfortable testing, the compiler statement ";;DEBUGINFO-" can be used to switch off the generation of debug information on the Irdata code.

In general, this is only practical for thoroughly tested application programs.

The additional debug information is missing in the Irdata code. The Irdata code is shorter and can be processed more quickly. However, debugging with the debugging system is no longer possible.

;;DEBUGINFO+ is preset.

Syntax

```
;;DEBUGINFO-      ;Test info generation is switched off
```

Further restrictions for "DEBUGINFO-":

Possible program errors are also displayed without line numbers.

Use ";;DEBUGINFO+" to switch on the generation of debug information again.

5.4 Main Program Structure

5.4.1 General

Each main program consists of

- declarations
- statements (instructions) and optional
- subroutine declaration(s) (subroutines)

RCL Program Structure

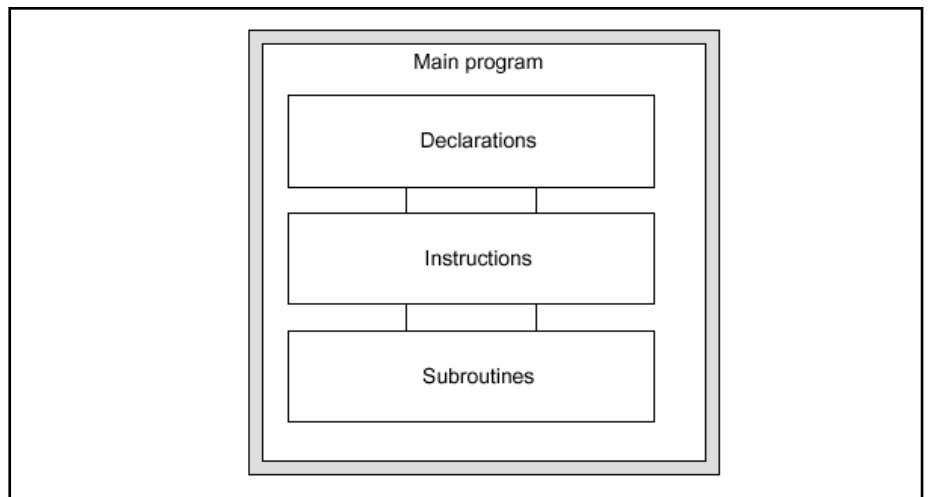


Fig. 5-1: Main program structure

5.4.2 Declaration Part

The declaration is located at the beginning of the main program. In the declaration, the names which occur in the main program are declared.

This applies to the:

- program head
name of the main program
- external declaration
names of the external main programs called in the main program
- channel declarations
names of the input and output channels used in the program
- variable and constant declaration
names of variables and constants in the program

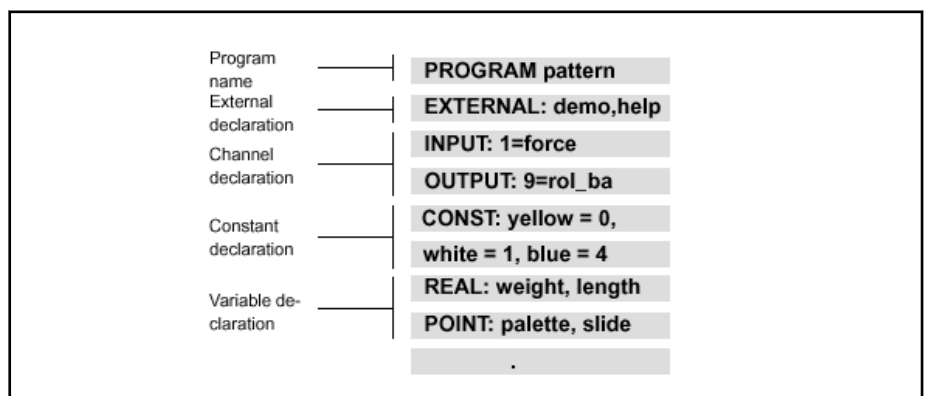


Fig. 5-2: Declaration part of the main program

Optionally, the keyword "VAR" can introduce the declaration part of variables.

RCL Program Structure

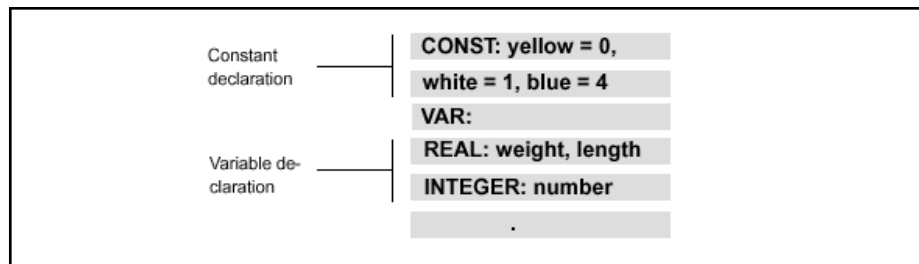


Fig.5-3: Declaration of variables

The keyword "BEGIN" has to be used to separate the statement from the declaration.



Undeclared variables are considered to be "POINT" or "JC_POINT" variables.

5.4.3 Statement Part

In the statement part, the statements to be executed are programmed.

These are for example

- motion commands
- decelerations and stop
- main program calls
- subroutine calls
- program part repetitions
- program jumps
- arithmetical operations and
- function calls

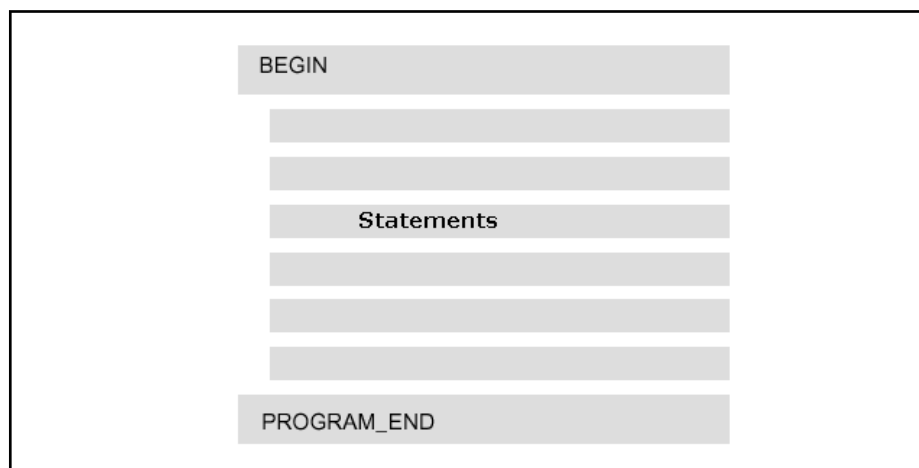


Fig.5-4: Statement part

The statement is located between the keywords "BEGIN" and "PROGRAM_END".

5.5 Program Declaration

A main program is identified at the beginning by the RCL word "PROGRAM" and its program name.

The program name consists of a maximum of eight characters. Letters, digits and underscores are permitted.

RCL Program Structure



The first character has to be a letter. Upper case and lower case letters are of equal value

The program end is identified by the RCL word "PROGRAM_ END" or "SUB_ END" if subroutines are listed.



The program name and the file name have to be identical.

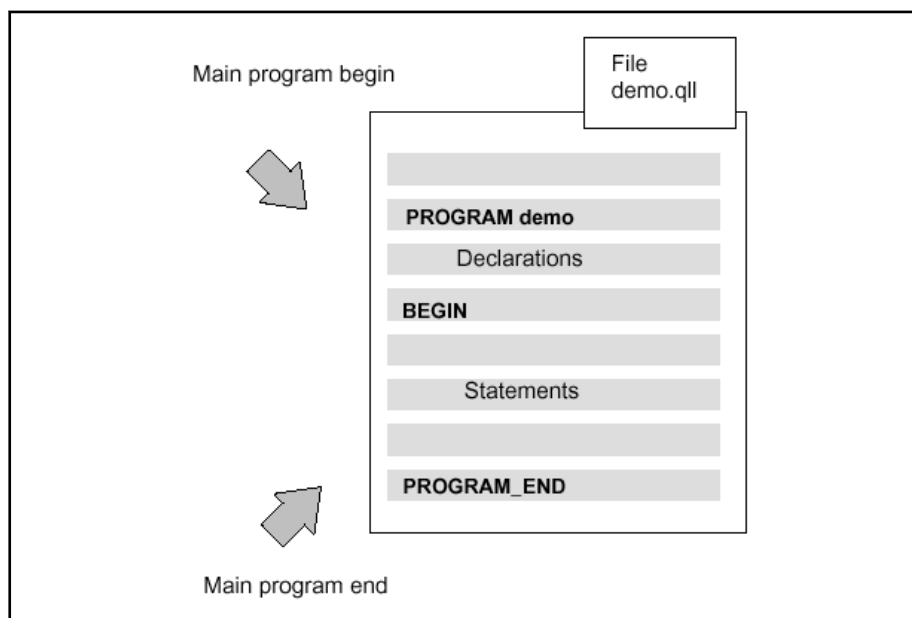


Fig.5-5: Structure of the main program

5.6 Main Program Call in Main Program

A main program can consist of several external individual main programs. Each main program is a separate QLL file.

External declaration:

The external main programs has to be declared with "EXTERNAL" at the beginning of the declaration part and has to be available in the control as ird file.

Transfer parameters for external main programs are optional. The number, order and data types has to match the declaration at parameter transfer.

A parameter can be transferred as value or reference. When a parameter is transferred as value, the keyword "VALUE" has to be used. This type of parameter transfer only transfers the value. When a parameter is transferred as reference and not as value, a reference on the value is transferred. Thus, it is possible to change transferred parameters in an external program and to process this change in the calling program. The following data types are allowed as parameters.

Types for parameters of external programs

RCL definition	Transfer type
VALUE CHAR	Character as value
VALUE REAL	Floating point number as value
VALUE BINARY	Binary number as value

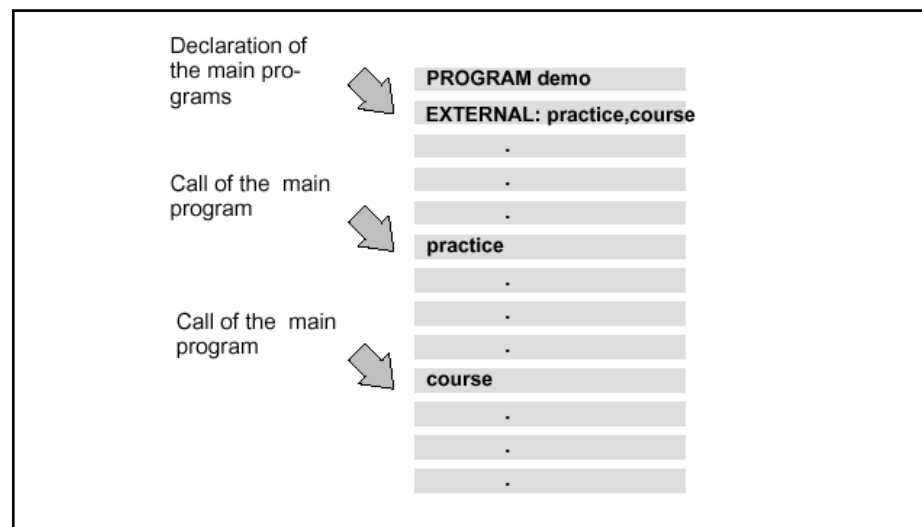
RCL Program Structure

RCL definition	Transfer type
VALUE INTEGER	Integer as value
VALUE TEXT	Character string as value
VALUE POINT	World coordinates as value
VALUE JC_POINT	Joint coordinates as value
CHAR	Character as reference
REAL	Floating point number as reference
BINARY	Binary number as reference
INTEGER	Integer as reference
POINT	World coordinates as reference
JC_POINT	Joint coordinates as reference

Fig.5-6: *Types for parameters of external programs*

External main programs with transfer parameters cannot be started as independent programs. Instead, they have to be started by a program call of a superordinate main program.

Only the specification of the declared program name is required to call external programs in the active main program.

Fig.5-7: *Working with several main programs*

The main program and the external program to be called are compiled independently of each other. During compilation, the transferred parameters cannot be checked with respect to their matching with the declaration in the external main program. It is performed, when the program is running. The number, types, order and nature (VALUE or reference) of the transfer parameters have to correspond to the declaration of the called external main program.

Programming

The program names of the external main programs are declared with the statement "EXTERNAL:"

Example

```
EXTERNAL: practice,course
```

RCL Program Structure

External declaration with parameter transfer

EXTERNAL: withpar(VALUE INTEGER: number REAL: r2)

It is defined that a main program exists. The first parameter transfers an integer as reference and the second parameter transfers a floating point number as value. The main program has to be declared accordingly.

Main Program

PROGRAM withpar(VALUE INTEGER: number REAL: r2)

Main program call in the main program

The control executes the active main program up to the external program call.

This is followed by a jump to the start of the program "practice".

The program "practice" is executed up to the "HALT" statement (or up to "PROGRAM_END").

"HALT" (or "PROGRAM_END") triggers a return to the main program "demo".

The control continues the program run with the statement following the call.

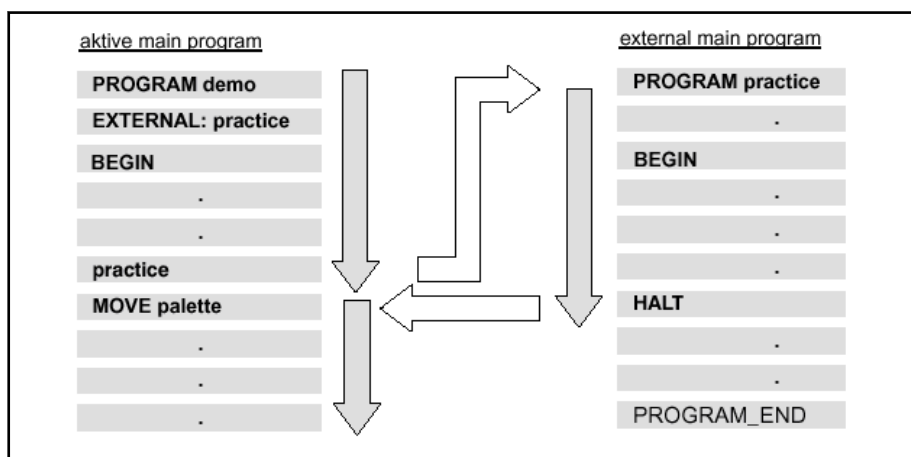


Fig. 5-8: Sequence of a main program call in the main program

Effects of global specifications

If a main program is called externally, the current global specifications such as V, A, J or the slope type are applied. Changes of these values in the calling program are not transferred to the calling program at the return from the program. For example:

RCL Program Structure

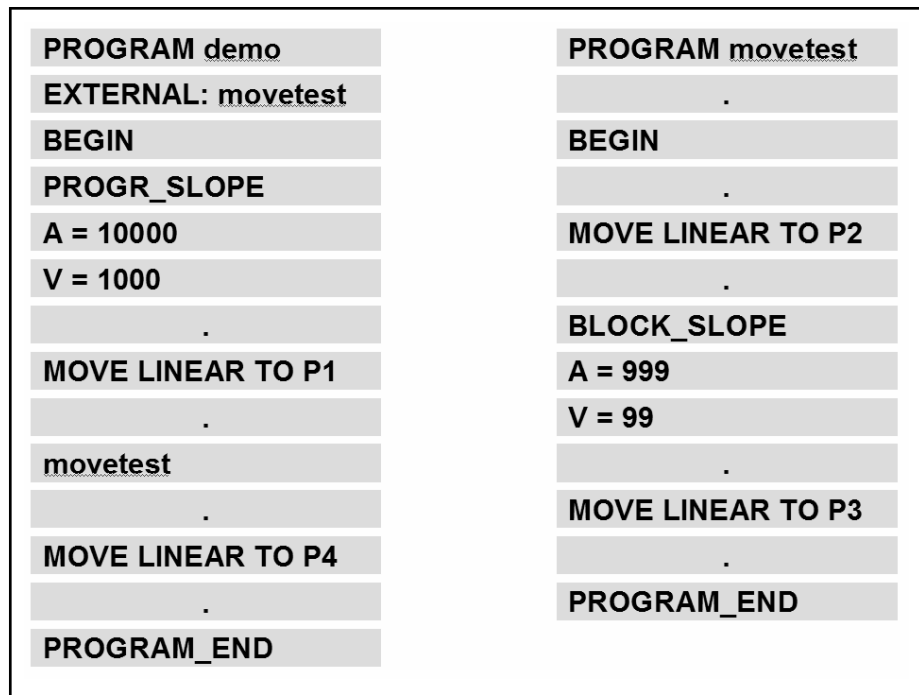


Fig.5-9: Example of an external program call with changed global specifications

The following table shows the global specifications used to approach the individual points.

Global specifications for the motion

Point	A	V	Slope type
P1	10000	1000	PROGR_SLOPE
P2	10000	1000	PROGR_SLOPE
P3	999	999	BLOCK_SLOPE
P4	10000	1000	PROGR_SLOPE

Fig.5-10: Global specifications for the motion

5.7 Subroutine Declaration

If the same work procedures are to be performed at different points in the program, these procedures can be combined into subroutines. With these subroutines, storage area is saved while increasing the user-friendliness of the program.

Variables defined in the main program, i.e. global variables, can also be processed in the subroutine.

Variables defined in the subroutine, i.e. local variables, can only be processed in the subroutine. There is no transfer to the main program.

The subroutine declarations are located after the main program after the PROGRAM_END statement.

Identification A subroutine is identified at the start by the word "SUBROUTINE" and the subroutine name.

The subroutine name contains a maximum of twelve characters. Letters and digits are permitted.

RCL Program Structure



The first character has to be a letter. Upper case and lower case letters are of equal value.

The subroutine is ended by the RCL keyword "SUB_END"

Programming

A subroutine contains statements for the gripper and should get the name "gripper".

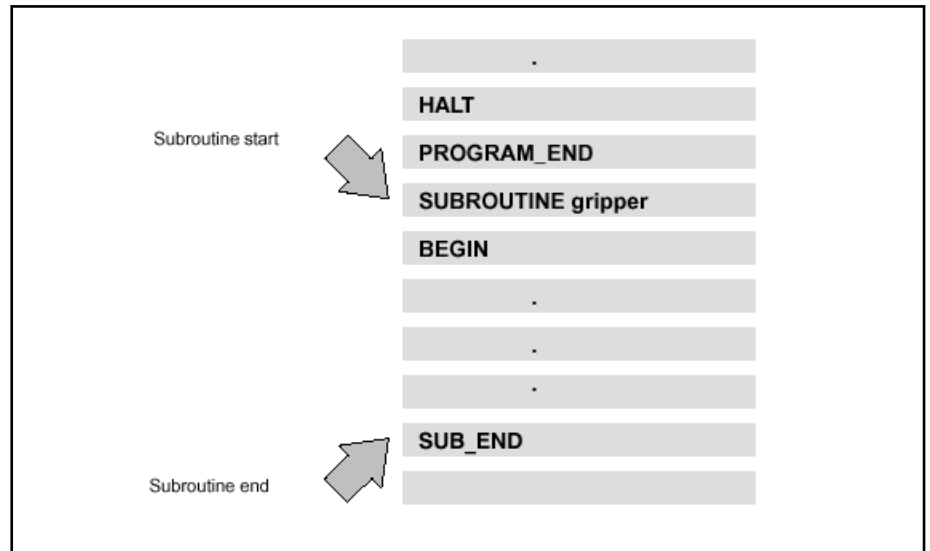


Fig.5-11: Position of the subroutine after the main program



"RETURN" can be used several times in a subroutine, e.g. with program jumps and conditional statements.

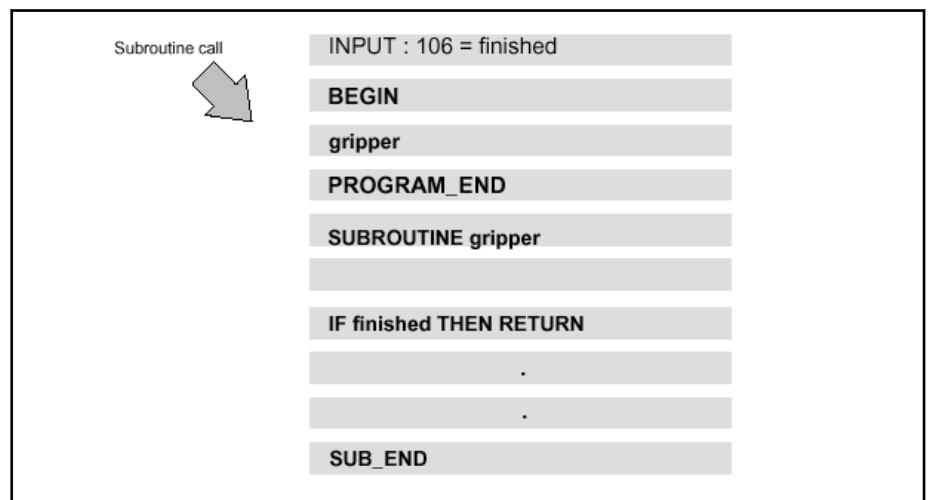


Fig.5-12: Subroutine call and conditional return

If the subroutine return can be identified from the program structure, e.g. at the end of the subroutine, the compiler automatically generates the command "RETURN".

Subroutine call

Only the declared subroutine name has to be specified, "gripper" in the example, for the subroutine call.

RCL Program Structure



The program and the subroutine to be called are compiled independently of each other. During compilation, the transferred parameters are not checked for matching. It is performed, when the program is running. The number, types, order and nature ("VALUE" or addressing) of the transfer parameters have to correspond to the declaration of the called external main program.

5.8 Subroutine Sequence

The control processes the main program up to a subroutine call.

This is followed by a jump to the start of the called subroutine. The subroutine is processed up to RETURN (or SUB_END). The command "RETURN" triggers a return to the main program.

The control continues the program run with the statement following the subroutine call.

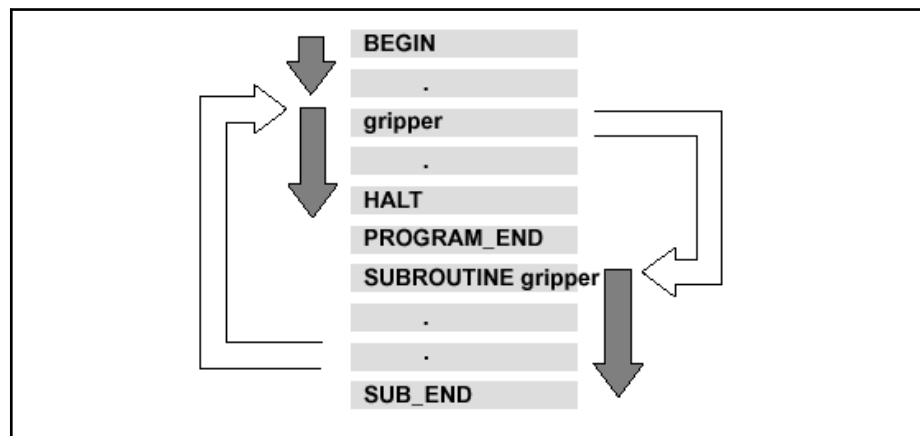


Fig.5-13: Program execution, main program/subroutine

Transfer variables or values with a subroutine call. For this purpose, the variables have to be declared accordingly in the subroutine declaration.

Example

```
gripper (1.5,6.0)           ;Value assignment
SUBROUTINE grip(VALUE REAL: force1,force2) ;when calling the subroutine,
                                ;Variable force1 has the value 1.5 and
                                ;Variable force2 has the value 6.0.
```

The value is transferred if the declaration includes the preceding RCL command "VALUE". Otherwise, the address of the variable is transferred, i.e. the variable has to be declared in the main program.

If the declaration is preceded by "VALUE", the calling program transfers information to the called subroutine. However, the called subroutine does not return information this way.

If the address is transferred, the assignments in the subroutine also apply to this variable after returning to the calling program.

Example

```
;;INCLUDE head           ;Include the compiler statements such as control type, TESTINFO+ etc.
PROGRAM demo             ;from the "head" file into the program
REAL: x                  ;Test program for subroutine with parameter transfer
BEGIN                   ;Variable declaration
  x=5
  anix(X)                ;Calling of the subroutine anix
                          ; x has the value 10
  awas(X)                ;Calling of the subroutine awas
                          ; x still has the value 10
```

RCL Program Structure

```

HALT
PROGRAM_END
SUBROUTINE awas(REAL: aw) ;Subroutine declaration awas. Transferring the value and the
                           ;variable. The subroutine calculates the variable and
                           ;returns it to the main routine.

BEGIN
  aw=aw*2                  ; aw and x have the value 10 now
  RETURN
SUB_END

SUBROUTINE anix(VALUE REAL: aw) ;Subroutine declaration anix. Transferring the value and the
                                ;Variable x remains unchanged in the main routine.
                                ;The subroutine processes the aw variable (individual storage location).
                                ;After exiting the subroutine, the variable remains unchanged
                                ;in the main routine.

BEGIN
  aw=aw*2                  ;aw has the value 20 and x has the value 10
  RETURN
SUB_END

```

Nesting Additional main program calls or subroutine calls can be programmed in called main programs or subroutines. This is called nesting.

Program examples of subroutine nesting

- A call of the subroutine "stacker" is programmed in the main program.
- In the subroutine "stacker", a further subroutine call is programmed, the subroutine "gripper".
- The control executes the main program up to the "stacker" call.
- This is followed by a jump to the subroutine "stacker".
- The control executes the subroutine "stacker" up to the call "gripper".
- This is followed by a jump to the next subroutine "gripper".
- The subroutine "gripper" is executed completely in the example shown.

With the "RETURN" command, the control jumps back to the subroutine "stacker", continues the program run up to "RETURN" and finally jumps back to the main program.



In case of nesting programs and subroutines, ensure that no end-less loops are created.

Any nesting depth is possible. The nesting depth is only limited by the stack size (4 KB).

RCL Program Structure

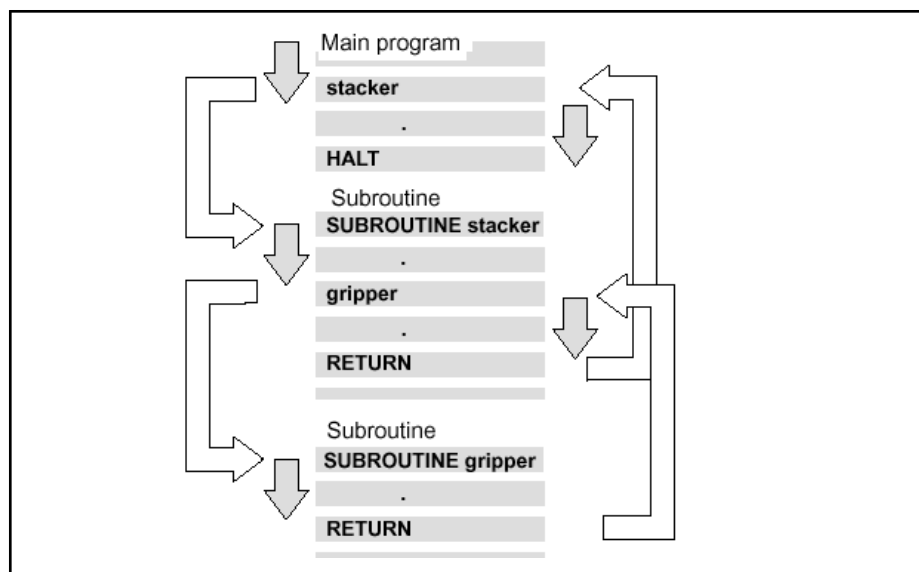


Fig.5-14: Example 1

Program part repetitions can also be nested. A second repetition is programmed in a program part repetition.

The program is executed as follows:

- The control executes the program part once up to the start of the second program part.
- The second program part is repeated three times.
- Subsequently, the control continues the program run up to the end of the first program part. The first program part has been executed once.
- For the second run, the control returns to the start of the first program part.
- The entire sequence is repeated a second time.

Any nesting depth is possible. The nesting depth is only limited by the stack size (4 KB).

RCL Program Structure

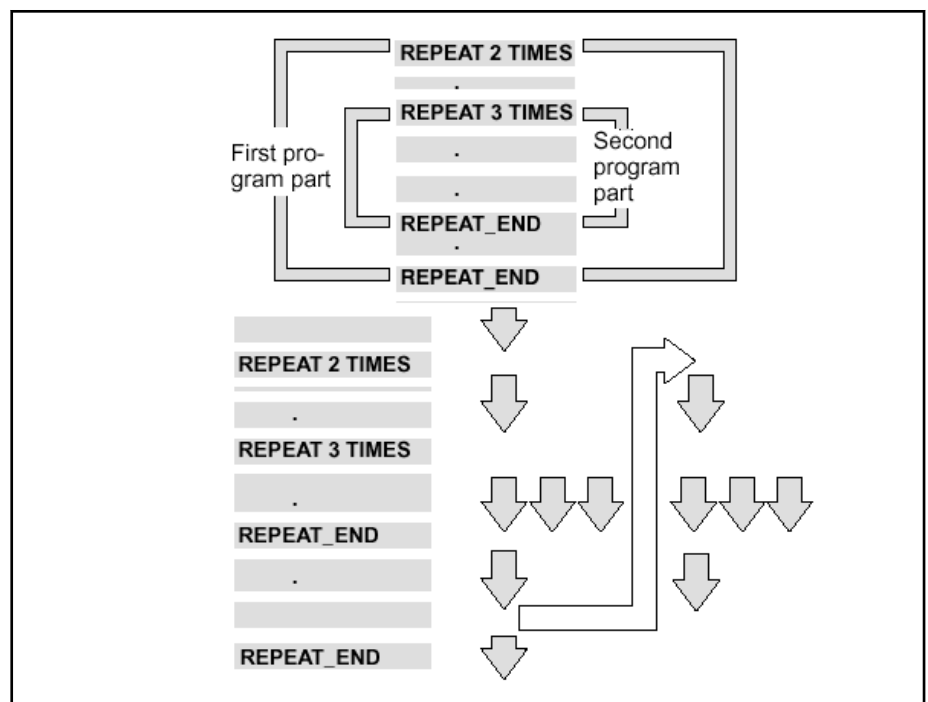


Fig. 5-15: Example 2

6 RCL Constants

6.1 RCL Constant Declaration

In one program, numeric values, e.g. the number of repetitions, character sequences or characters can be specified by constants.

The value of a constant can also be specified by a computing rule.

The constants have to be defined in the constant declaration before the variables of a program are defined.

Example

```
CONST: red=1,yellow=2*red,green=2*yellow,lines=4,           ;Constants
      columns=20,limit_value=2.5*7.25
INTEGER: color, l_index, c_index                             ;Variables
REAL: meas_value
ARRAY[lines .. columns] REAL: distance
BEGIN....
IF meas_value > limit_value THEN ....                       ;Implementation
```

6.2 Predefined RCL Constants

Default constants are available in the RCL. These constants are part of the language scope and do thus not have to be declared.

WC_UR "WC_UR" is a constant of type WC_FRAME predefined in RCL. All its six components have a value of 0.

If this constant is assigned to WC_SYSTEM, all displacements and rotations to the original coordinate system become zero, e.g. by WC_SYSTEM = WC_UR.

The workpiece coordinate system is again equal to the original world coordinate system (also refer to [Defining the Workpiece Coordinate System Relative to the World Coordinate System, page 27](#)).

7 RCL Variables

7.1 General Information

Numerical values, e.g. coordinate values for positions or the number of repetitions can be replaced in a program by variables.

A variable reserves storage area under its name.

A numerical value can be assigned any number of times to this storage area. The control always stores the latest numerical value assigned.

If a variable name is displayed during the program run, the control replaces the variable with the value stored under its name at that point in time.

Each variable has a name. Different variables have to be provided with different names. If possible, the names should be chosen in a way so that the meaning of the variable can be identified, e.g. for position designations "bore_1" is better than "b1" or "counter" for arithmetic variables should be chosen over "i".



The name consists of 12 characters max.

Letters, digits and underscores "_" are permitted. There is no case sensitivity.

The first character has to be a letter.

Space characters are not permitted.

Point variables of type "JC_POINT" are an exception.

These variables have to start with "@".

7.2 RCL Data Types

7.2.1 Overview

The data type of a variable determines its value range and the permissible assignment and arithmetic operations. The permissible operations are listed in [Complete Value Assignment, page 66](#).

It has to be distinguished between simple data types and composed, structured data types.

Simple data types

- INTEGER
- REAL
- BINARY
- CHAR

Structured data types

- POINT
- JC_POINT
- BELT
- TEXT
- ARRAY
- WC_FRAME
- FILE.

A composed data type consists of two or more simple data types.

RCL Variables

7.2.2 Simple RCL Data Types

Variables are declared before using them in the declaration (before "BEGIN"). The keyword "VAR" separates this section from a type definition.

INTEGER Only positive or negative integers, i.e. numbers with a decimal point, can be assigned to "INTEGER" type variables.

Value range -2^{31} to $+(2^{31} - 1)^{1)}$



The RCL data type INTEGER corresponds to the PLC data type DINT!

REAL Only positive or negative real numbers, i.e. numbers with a decimal point, can be assigned to "REAL" type variables.

Value range: 4 byte floating point number with simple accuracy²⁾.



The values for "REAL" type variables has to be entered with decimal points, not commas.

BINARY Only 0 or 1 may be assigned to variables of the type "BINARY".



Integral arithmetic operations cannot be performed with "BINARY" type variables.

The digits 0 and 1 do not represent any conventional values.

They describe two defined states, e.g.

Variable designation	0 (FALSE)	1 (TRUE)
Switch	Off	On
Question	No	Yes
Bowl	Empty	Full
Filled	False	True

Fig. 7-1: Interpretation of binary values

Variables of type "BINARY" are for example signals of binary input and output channels.

CHAR Variables of type "CHAR" may only be assigned to ASCII characters³⁾.

¹⁾ INTEGER type variables (long integers) are saved as 32 bit numbers (4 bytes) with signs in the range from -2,147,483,648 to 2,147,483,647.

²⁾ REAL data type variables (long integers) are saved as 32 bit numbers (4 bytes) -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.

³⁾ The ASCII (American Standard Code for Information Interchange) character encoding (7 bits) is used to represent letters and symbols used on a standard US keyboard. The ASCII character encoding corresponds to the first 128 characters (0-127) of the ANSI character set.

7.2.3 Composed RCL Data Types

POINT Variables of type "POINT" may only be assigned to positions in world coordinates. The individual coordinate values of a position of "POINT" type has to be of type "REAL".

JC_POINT Variables of type "JC_POINT" may only be assigned to positions in joint coordinates.
The individual coordinate values of a position of type "JC_POINT" has to be of type "REAL".



Variable names of type "JC_POINT" have to start with "@".

Variables of the types "POINT" and "JC_POINT" are also called point variables.

The number of components depends on the axis number of the specified kinematics.

TEXT Variables of type "TEXT" may only be assigned to texts that consist of 80 ASCII characters max. The individual characters can be addressed directly like array elements using the index.

Text variables, declaration, assignment

```
TEXT: char_string           ;Declaration of text variables
char_string[1]='A'         ;Assigning individual components
char_string[2]='B'
char_string='This is ASCII text' ;Assigning a character string
```

FILE *Files, declaration*

FILE: cad_dat

A file name is defined with the "FILE" data type. It is used as a parameter when accessing with "WRITE"/"READ" or "OPEN"/"CLOSE".

7.2.4 User-Defined RCL Types

In the RCL, users can declare their own types by extending the declaration in an RCL program to include the type definition "TYPE".

ALIAS name New names can be assigned to any data types.

This applies to both standard types

- INTEGER
- REAL
- BINARY
- CHAR

as well as for the structured types based on the standard types.

- TEXT
- ARRAY
- RECORD

RCL Variables

Examples

```

PROGRAM types
TYPE:
  tmatrix=  ARRAY[0..7] ARRAY[0..7] INTEGER
  myinteger= INTEGER
;Type definition
;2 dim. ARRAY "tmatrix"
;Direct derivation from "INTEGER"

VAR:
  ARRAY [0..7] ARRAY[0..7] INTEGER: matrixone
  tmatrix: matrixtwo
  myinteger: integervar
;Variable declaration

BEGIN
  initold(matrixone)
  initnew(matrixtwo)
PROGRAM_END
SUBROUTINE initold(ARRAY[0..7] ARRAY[0..7] INTEGER: matrix) ;Long spelling
  BEGIN...
SUB_END
SUBROUTINE initnew(tmatrix: matrix) ;Short spelling
  BEGIN...
SUB_END

```

Record Types, RECORDs Record types can be used to further improve the usability and structure of programs.

A record has a name and consists of 1..n variables (components).

Each component has its own name and data type.

A component itself can also be a record.

In PASCAL, these record types are known as "RECORD". In C, they are called "STRUCT".

Barcode identification example

Barcode information (r_barcode) consists of

- Name (name, TEXT)
- Weight (REAL)
- Volume (r_volume), itself a record of
 - Length (REAL)
 - Width (REAL)
 - Height (REAL)

Records are useful to organize complex data, particularly in large programs. They allow access to the complete data set and to each component.

The name in the type declaration on the left side of the assignment character in front of the reserved word "RECORD" stands for the entire record and can be used as abbreviation for the detailed declaration.

Variables mentioned in a record are called components.

A component can have the same name as an ordinary variable without creating a conflict. They can always be distinguished by context.

A component can only be accessed using the record name. A dot separates the component name from the record name.

If the component represents a record, its components are also separated by a dot.

Example

```
PROGRAM records
TYPE:                                ;Type definition part

r_volume=RECORD
    REAL: lenght
    REAL: width
    REAL: height
RECORD_END

r_barcode=RECORD
    TEXT: name
    REAL: weight
    t_volume: volume
RECORD_END

r_barcode: code                      ;Variable declaration for the "code" of a packing
BEGIN
;
; ..
code.name='cuboid'                   ;Accessing a component
code.volume.lenght= 12.5             ;Accessing a subcomponent
code.weight= 155.5
; ..
PROGRAM_END
```

When using record variables in WRITE or READ statements, the following has to be complied with:

- Record variables can only be read or written to "FILE" type files (text files) component by component.
- Record variables can be read or written to "BNR_FILE" type files (binary files) either completely or component by component.

ARRAY Variables of the same type can also be combined in one array.

These variables have the same name; only their index is different. Thus, these variables are also called indexed variables.

Syntax

```
ARRAY[( $\pm$ ] integer_constant)...( $\pm$ ] integer_constant)] <Type> : variablename
```

Example

```
ARRAY[-10..10] INTEGER: variablename
```

7.3 RCL Variable Declaration

Before executing a statement with variables, the control has to be informed which values or characters can be assigned to the variable.

Thus, every variable used in the program has to be declared first, i.e. the data type of the variable has to be defined.

Syntax

```
[DEF] Type: [channel =]name{ , [channel =]name}
```

- "DEF" can be located in front of the point variables "POINT" and "JC_POINT"
- "Type" has to be replaced by the actual type, e.g. "INTEGER".
- "channel" is only provided for the types "INPUT", "OUTPUT" or "BELT".

RCL Variables



All points declared with "DEF" and all points to which no value is assigned anywhere else in the program, are stored in a point file with the identification "pkt".

Points transferred to subroutines as parameters (spc.fct. and rho.fct) are not shown in the point file.

Programming The declaration consists of the data type and the variable names assigned. The data type is separated from the variable name by a colon. Several variable names of the same type are separated by commas.

Declarations can be made between:

- "PROGRAM" and "BEGIN" and
- "SUBROUTINE" and "BEGIN".

Example

```
PROGRAM Prog2
POINT: BasePnt
OUTPUT: 1=valve_1,2=valve_2
BINARY: ready
INTEGER: counter
REAL: divisor,xvalue,yvalue
TEXT: message_1,message_2
BEGIN
```



Variables of the type "POINT" and "JC_POINT" do not have to be declared, i.e. the compiler interprets undeclared variables as point variables.



"JC_POINT" variables start with "@".



RCL Keywords (see page 303), may not be used as names for user variables.

7.4 Global Variables

Global variables are used to simply exchange data between several individual user programs.

The basic concept is to summarize programs using identical global variables in one program group. A program can export data in this group while the other programs of this group may only import this data.

Any number of program groups can be created on the control (only limited by the available memory).

Program example: Program to be exported:

```
PROGRAM exp_var                ;Exporting program
                                ;Declaration part in the exporting program
PUBLIC DEF POINT: start_pos    ;Global data
PUBLIC INTEGER: index          ;Global variables are declared by adopting
                                ;the keyword PUBLIC in the type declaration
                                ;of these variables. The data range for
                                ;these variables is reserved in the IRD file.
                                ;Teachpoints, e. g. identified by the
                                ;keyword DEF, are stored in the PKT file.
REAL: mvalue                  ;Local data
BEGIN                        ;RCL statements.
```

```

start_pos = Pos           ;Actual position
index = 0                 ;Start index
                           ;Additional statements
PROGRAM_END

```

Program example: Program to be imported:

```

PROGRAM imp_var           ;Importing program
                           ;Declaration part in the importing program
                           ;Global data
EXTERNAL exp_var: start_pos ;The variables can be accessed from other
                           ;RCL programs, if the variables are declared
                           ;with EXTERNAL und the name of the exporting
                           ;program
EXTERNAL exp_var: index
EXTERNAL exp_var: write_prot
POINT: end_pos            ;Local data
BEGIN                     ;RCL statements

  REPEAT index TIMES
    MOVE TO start_pos
    MOVE TO end_pos
  REPEAT_END

PROGRAM_END

```

The following restrictions have to be considered when using global data:

- Data cannot be imported and exported in a program at the same time.
- Only data from one program may be imported.
- Global data may only be exported or imported in the declaration part of the main program, not in subroutines.

7.5 Point Variables

7.5.1 Declaration of Point Variables

Point variables are composed, structured data types that consist of components.

The components are the coordinates or the axes of the point variables.

In addition to the complete value assignment, the point variables can also be assigned component by component, e.g. corner.z_k = height.

The component designation is indicated with the following compiler statements:

- `;;JC_NAMES = axis name, ...`
as well as
- `;;WC_NAMES = coordinate name, ...`

Example

```

;;KINEMATICS: ( 1 = Kin_1)
;;KINEMATICS= Kin_1      ; Default Kinematics here
;;WC_NAMES= kx, ky, kz
;;JC_NAMES= a1, a2, a3

PROGRAM Prog2
  POINT: corner
  JC_POINT: @wpos
BEGIN
  corner.kx = 100.0
  corner.ky = 200.0
  corner.kz = 300.0
  @wpos.a1 = 45.3
  @wpos.a2 = 0.0
  @wpos.a3 = 10.2

  MOVE TO @wpos

```

RCL Variables

```
MOVE TO corner
PROGRAM_END
```



Point variable names of type "POINT" start with a letter.

Point variable names of type "JC_POINT" start with a special character "@".

Values are assigned for point variables only after "BEGIN".

7.5.2 Points and Point File "pkt"

While all other variables have to be declared, point variables do not need to be declared explicitly.



The RCL compiler interprets all undeclared variables as POINT or JC_POINT.

All points declared with "DEF" and all points to which no value is assigned anywhere else in the program, are stored in a point file with the identification "pkt". For details regarding multi-kinematic operation, see [Point Tables of the Kinematics, page 249](#).

Points transferred to subroutines as parameters (spc.fct.) do not appear in the point file.

This also applies to the transfer of individual point components (e.g. "Pos.X_K").



When the program is running, values can only be assigned to points in the pkt file if they were declared with "DEF".



Points that are not declared but to which values are assigned in the program, are not stored in the point file, but in the ird file.

7.5.3 Complete Value Assignment

In case of complete value assignments, values are assigned to all coordinates of the point variable. In a particular assignment, all point variables must be of one data type.

Exception:

A particular assignment can contain mixed operations with the standard functions "WC" or "JC".

Assignment of numerical values

```
position=(50,0,100,0,15,10)
@edge=@(45,5.8,70,10,5.8,0)
```

7.5.4 Assignment of Variables for Individual Components

Assignment for individual components

```
hole=(xvalue,yvalue,zvalue,0,0,0)
@seat=@(50,95.8,height,40,38,0)
hole.z_c=height
```

The variables "xvalue", "yvalue", "zvalue" and "height" are of type "REAL".

RCL Variables

Assignment by addition and subtraction

The total of the point variable "position" and the constant has been assigned to the point variable "shelf".

Constants consist of the enumeration of their coordinates placed in parentheses. In contrast to variables, constants cannot be modified while the program is running.

For addition and subtraction, the individual components are either added or subtracted.

Assignment Basis ± Offset

```
shelf= position + (10,-30,100,5,0,0,0)
```

Assignment by multiplication and division

Modify any of the coordinate values of point variables or constants by multiplying or dividing with numerical values or "REAL" type variables.

Assignment via expression (1)

```
pos_1=(10,10,10,0,0,0,0)
pos_2=pos_1*2
```

Each individual component of pos_1 is doubled by multiplying with 2 and is assigned to the new point variable pos_2, which then has the coordinates: pos_2 = (20, 20, 20, 0, 0, 0, 0)

Assignment via expression (2)

```
div=4.0
pos_3=pos_2/div
```

Subsequently, pos_3 has the coordinates: pos_3 = (5, 5, 5, 0, 0, 0, 0).

Mixed operation with point variables

Using the standard functions "WC" and "JC", mixed operations with "POINT" and "JC_POINT" type variables can be executed.

If the result of the arithmetic operations is to be assigned to a point variable of the type "POINT", the operation has to be executed using world coordinates.

If the result of the arithmetic operations is to be assigned to a "JC_POINT" type variable, the operation must be carried out using joint coordinates.

Mixed operations

```
PROGRAM Prog2
  DEF POINT: p2
  DEF JC_POINT: @p1, @p3
  BEGIN
    @p3=@(0,100,0)
    @p1= JC(p2)+@p3
  PROGRAM_END
```

The control converts the world coordinates of the point "p2" into joint coordinates and adds them to the coordinates of @p3.

The result is assigned to the point variable "@p1".

Reading the actual position POS, @POS

The standard point variables POS (world coordinates) and @POS (joint coordinates) can be used to assign the current position of the robot to a point variable while the program is running.

Reading the actual position

```
act_pos= POS
@joint_pos= @POS
```

RCL Variables

The standard point variables are located on the right side of the assignment. A component by component assignment is also possible.

Reading the actual position, component-by-component

```
cpos.c_3= POS.c_3
xvalue= POS.c_1
```



However, a query can only be made when the robot is in standstill.

If the query is made following a "MOVE..TO" command, a velocity-related position error is generally reported in the entire causal loop.

Assignment by component

For point variables, assign new values to individual coordinates.

Conversely, it is also possible to assign the coordinate values of point values to "REAL" type variables.

The coordinate names are defined in the machine parameters or by a compiler statement. In the following examples, the coordinates of a position in world coordinates are designated by c1, c2, c3, etc. (c = coordinates) and the coordinates of a position in joint coordinates are designated with a1, a2, a3, etc. (a = axis).

The coordinate name is added as a suffix to the name of the point variable using a dot.

Example

```
position.k3= 100      ;the value 100 is assigned to the coordinate k3
zvalue= anchor.k3    ;the third coordinate value of the point variable anchor
                     ;is assigned to the variable zvalue
@pal_pos.a4= radius  ;the coordinate a4 of the position described in joint coordinates
                     ;is assigned the value of the variable radius of type REAL
```

7.6 Working with Text Variables

Texts can be assigned to text variables in a program.

Declare 'message' and 'note' as TEXT variable

```
TEXT: message,note
```

Text assignment

The text to be assigned has to be placed between apostrophes and should not exceed 80 characters. The text must be in one line.

Character string is assigned to TEXT variable

```
message= 'gripper is defective'
note= 'change pallet'
```

Use of variables

In the program text variables can be output to an output channel, e.g. as a file with the RCL command WRITE and input with the command READ.

READ/WRITE on input/output channel

```
WRITE File1, note
READ File1, entry
```



The variable itself cannot be placed between apostrophes.

For **WRITE 'note'** the control displays the word "note".

7.7 Working with Array Variables (Arrays)

Variables of the same type can be combined to arrays.

Arrays consist of a freely selectable number of array positions designated by ascending numbers. A variable can be assigned to each array position by specifying a number (index). Array variables all have the same name; only the index is different. The index is identical to the number of the assigned array position.

All types of variables can be stored in arrays.

Array declaration

The array declaration consists of:

- the declaration command "ARRAY"
- array limits (position numbers)
- the declaration of array variables.

The array limits are indicated by square brackets and are generated from the first position number (first index = lower limit) and the last position number (final index = upper limit).

Declaration of array instances

```

ARRAY[1..9] POINT: placing_pos           ;1-dim. ARRAY
ARRAY[0..10] TEXT: message
ARRAY[-10..10] INTEGER: number_pos
ARRAY[1..9]ARRAY[5..10]POINT: picking_pos ;multidim. ARRAY
  
```

The lower limit must be separated from the upper limit by two dots, e.g. [3..8].

The upper limit of the array cannot be less than the upper limit. The index is of type "INTEGER".

Example

```

ARRAY[1..5] POINT:hole;declaration of an array with five positions for the
                        ;point variable 'hole', the first index should be 1
  
```

Value assignment

Each array variable is addressed using its name and the index placed in square brackets. For the index, a variable or an expression of type "INTEGER" can also be inserted. A value assignment is possible using the "REPEAT" statement or other program loops.

Value assignment for array elements

```

PROGRAM Prog2
ARRAY[-5..0] INTEGER: number
INTEGER: i,k           ;index i and value k
BEGIN
;example of direct assignment
number[-5]=1           ;the variable number on the array position -5 has the value 1,
number[-4]=2           ;the variable number on the array position -4 has the value 2,
number[-3]=4           ;etc.
number[-2]=8
number[-1]=16
number[0]=32

;example with REPEAT statement
i=-5                   ;first index
k=1                    ;first value
REPEAT 6 TIMES
number[i]=k            ;value assignment
i=i+1                  ;increase index
k=k*2                  ;value change
                        ;user-defined assignments
REPEAT_END

;example program loop
  
```

RCL Variables

```
i=-5                ;first index
k=1                ;first value
mark:              ;jump target
number[i]=k        ;value assignment
i=i+1              ;index increase
k=k*2              ;value change
                  ;user-defined assignments
IF i<=0 THEN JUMP mark;conditioned assignm. f. program loop
PROGRAM_END
```

Specification of pallet positions

```
;;INCLUDE head      ;inserting compiler statements
PROGRAM palpos      ;determining pallet positions
INTEGER: line,column,c,l,k ;declarations
ARRAY[1..12] POINT: palpos;the pallet contains 4 lines and 3 columns,
                        ;as well as 12 positions
BEGIN              ;program start
  dx=(30,0,0,0,0)   ;distance of positions
  dy=(0,20,0,0,0)   ;specified in increments under dx and dy
  line=4
  column=3
  c=0
  l=0
  k=0
  mark_1:
    k=k+1            ;statements, point assignments
    palpos[k]=pos+c*dx+l*dy ;the position pos is a teach point, i.e.
                        ;it is defined by approaching and saving,
                        ;it is the first position palpos[1].
                        ;thus, s=0 and z=0 is entered in the initialization part.
                        ;increased number of columns
    c=c+1            ;increased number of columns
    IF c<column THEN JUMP mark_1;the next position palpos[2] is in
    ELSE c=0          ;the adjacent column.
                        ;The number of columns has to be increased.
                        ;However, the number of columns c cannot
                        ;exceed the entire number of columns.
                        ;if c is smaller than column, the
                        ;control jumps to the mark mark_1,
                        ;increases the index k by 1
                        ;and assigns the value pos+1*dx+0*dy to the
                        ;position palpos[2] . If c becomes greater than
                        ;column, the control assigns the value zero to the control
                        ;and increases the number of lines.
    l=l+1            ;increase line number
    IF l<line THEN JUMP mark_1;the increase of the line number runs analogously
                        ;to the increase of the number of columns. In the
                        ;IF-THEN statement without ELSE statement
                        ;the control continues with the travel statement
                        ;in case of non-compliance with condition l<line

;;INT= LINEAR
V=1000
FACTOR=9.999
k=0

REPEAT 12 TIMES
  k=k+1
  MOVE TO palpos[k] ;travel statement
  MOVE_REL with V=36 EXACT (0,0,-20,0,0)
  WAIT 2
  MOVE_REL with V=59 EXACT (0,0,20,0,0)
REPEAT_END
MOVE_REL CIRCULAR((-50,-50,100,0,0),(-100,-100,0,0,0))
HALT
PROGRAM_END
```

7.8 PLC Communication Channels

7.8.1 Communication Channels to PLC, General

With the RCL, it is possible to write to the inputs of the PLC and read from outputs of the PLC:

RCL Variables

- **BINARY** The interface is provided with 32 binary inputs and outputs,
The extended interface is provided with $32 + 160 = 193$ binary inputs to and outputs from the PLC.
- **INTEGER** The interface is provided with four inputs and outputs of type "INTEGER",
32 bits
The extended interface is provided with $4 + 46 = 50$ binary inputs to and outputs from the PLC of type "INTEGER".
- **REAL** The interface is provided with four inputs and outputs of type "REAL",
The extended interface is provided with $4 + 46 = 50$ binary inputs to and outputs from PLC of type "REAL".
- **POINT** Points can be transferred from the PLC to the RCL program, but not from the RCL program to the PLC.
Example
INPUT POINT: 701 = point1
OUTPUT POINT... is not available!
The interface is provided with four inputs of type "POINT",
The extended interface is provided with $4 + 46 = 50$ inputs of type POINT.
- **BELT** Belt channels allow for synchronization with any axis, i.e. a real axis, virtual axis, link axis or encoder axis (standard position measuring system).
Belt channels are inputs of type "INTEGER". No extended interface.



Data exchange with the PLC

1. Inputs on the RCL side are outputs on the PLC side and vice versa.
2. Interface data can be accessed on the PLC side via the Kin-Data interface and the function.
Data of the extended interface can only be accessed on the PLC side only via function.
3. The function access on user data does not depend on the block preparation of the RCL program, but is directly executed.

ML_KINEMATICDATA_SM	FUNCTION	INDEX
dwUserBits_i	ML_KinGetUserBits	1..32
	ML_KinGetUserBits	33..192
dwUserBits_i	ML_KinGetUserBit_DWord	1
	ML_KinGetUserBit_DWord	2..6
adiUserDInt_i[1..4]	ML_KinGetUserDInt	1..4
	ML_KinGetUserDInt	5..50
arUserReals_i[1..4]	ML_KinGetUserReal	1..4
	ML_KinGetUserReal	5..50

RCL Variables

ML_KINEMATICDATA_SM	FUNCTION	INDEX
dwUserBits_q	ML_KinSetUserBits	1..32
	ML_KinSetUserBits	33..192
adiUserDInt_q[1..4]	ML_KinSetUserDInt	1..4
	ML_KinSetUserDInt	5..50
arUserReals_q[1..4]	ML_KinSetUserReal	1..4
	ML_KinSetUserReal	5..50
arUserPoints_q[1..4]	ML_KinSetUserPoint	1..4
	ML_KinSetUserPoint	5..50

Fig. 7-2: Erweiterung des Shared Memory-Bereiches durch Funktionszugriff aus Sicht der SPS

The respective input or output is addressed by specifying a channel number in the declaration of input or output variables.

Synchronous and asynchronous processing

User data can be accessed in the RCL program via INPUT / OUTPUT variables.

Synchronous as well as asynchronous access (channel number + 1000) is possible; synchronous $\Leftrightarrow$ effective in block processing, asynchronous $\Leftrightarrow$ effective in block preparation.

The user data can be accessed as follows:

- INPUT / OUTPUT BINARY,
- INPUT / OUTPUT REAL,
- INPUT / OUTPUT INTEGER ,
- INPUT POINT.

RCL access examples:

Access on 108. user bit, output, asynchronous:

```
OUTPUT BINARY:    1108 = bool1
bool1 = TRUE
```

Access on 41. user point, input, synchronous:

```
INPUT POINT:      741 = point1
varPoint = point1
```

The following channel numbers (for asynchronous access + 1000) are provided for each kinematics in MLC 10VRS:

Channel number ⁴⁾	Type and meaning
1 to 32, to 192	BOOL, binary inputs and outputs
201 to 204, to 250	REAL, inputs and outputs
401 to 404, to 450	INTEGER, inputs and outputs (on the PLC side DINT!)
501 to 504	Belt inputs (no outputs)
701 to 704, to 750	Point inputs (no outputs)

Fig. 7-3: Channel declarations

⁴⁾ Third digit: extended interface.

Belt inputs RCL programmers can access belt data using the definition

BELT: 501...504.

From the **parameter** perspective, the "BELT: 501" data can be found in parameter K-0-0008, Axes configuration list, first belt axis, etc.

The belt coordinates are added to the coordinates of the Tool Center Point (TCP) in this way.

The **PLC** addresses the same shared memory via an instance of the data type "ML_KINEMATICSDATA_SM" (for details see Rexroth IndraMotion MLC10VRS, ML_Robot).

The current belt position can be taken from "KinData[n].arPositionOriginalCoord_i" (first TCP coordinates then belt coordinates).

7.8.2 Communication Channels to PLC, Declaration and Programming

In channel declaration, the data type "BINARY", "INTEGER" or "REAL" and the variable name of the signal to be transferred are assigned to a channel number. It is necessary to define whether input or output signals are involved.

Individual channel assignments must be separated by a comma. Do not place a comma after the last assignment.



If the data type is not specified as "INPUT" or "OUTPUT", the control automatically assumes "BINARY".

Communication channels, declaration and usage

```
PROGRAM Prog1
INPUT:
  1=gate_switch1,
  2=met_unit,
  5=li_barrier
OUTPUT:
  7=alarm
BEGIN
  IF NOT gate_switch1 AND li_barrier THEN alarm=1
PROGRAM_END
```

When querying binary signals for 1 it is not necessary to specify "=1". The control then automatically queries for "1".

Example: Test on "1" (TRUE)

```
IF gate_switch1 THEN ..
IF NOT gate_switch1 THEN ..
```

Example: Signals of type "INTEGER"

```
PROGRAM Prog2
INPUT:
  1=gate_switch1
INPUT INTEGER:
  401=grip_force,
  403=meas_height
OUTPUT INTEGER:
  401=pressure,
  402=print
BEGIN
  WAIT UNTIL gate_switch= 1
  IF grip_force>=26 THEN...

WAIT UNTIL meas_height >= 212
MOVE LINEAR UNTIL meas_height >= 200 TO pos ; ok
PROGRAM_END
```

RCL Variables

Example: Signals of type "REAL"

```

PROGRAM Prog3
INPUT REAL:
  201=torque,
  206=force
OUTPUT REAL:
  203=speed
BEGIN
;...
PROGRAM_END

```

Example: Point inputs

```

PROGRAM Prog4
;handshake for the points pick and place by the PLC
INPUT BINARY: 1= ppavail
INPUT POINT: 701= pick, 702= place
;handshake to PLC, points processed pick and place
OUTPUT BINARY: 1= ppready

BEGIN
Loop:
;Wait till PLC sets pick and place
WAIT UNTIL ppavail= 1
MOVE TO pick
MOVE TO place
;handshake to PLC, points processed pick and place
ppready= 1
JUMP Loop
PROGRAM_END

```

Querying channels/signals

The query and evaluation of channels or their assigned names occurs in conditions.

Example: Querying channels/ signals in the program

```

PROGRAM Prog5
INPUT:          2=met_unit
INPUT INTEGER: 401=grip_force
BEGIN
  IF met_unit THEN ...
  IF grip_force>=26 THEN...
PROGRAM_END

```



- Output signals cannot be queried.
 - Only input signals can be queried.
-

Example: incorrect query

```

MOVE UNTIL grip_force=100 AND meas_height=200 TO pallet

```

Incorrect is:

- MOVE UNTIL grip_force=100 AND meas_height=200 TO pallet

The compiler interprets this statement as follows:

- MOVE UNTIL grip_force= ((100 AND meas_height) =200) TO pallet
-



The query in combination with a "MOVE UNTIL" statement is only permitted for 1 input signal.

Setting signals Signals are set in value assignments; see [Arithmetic Expressions, page 89](#).

Example: Setting signals

```
PROGRAM Prog6
OUTPUT INTEGER: 402=print
BEGIN
  print= 75
PROGRAM_END
```



- Input signals cannot be set.
- Only output signals can be set.

Binary signals can be combined with "AND", "OR" or "NOT".

Exceptions include the conditions "WAIT UNTIL" and "MOVE UNTIL". No combinations are permitted here.

8 RCL Program Control

8.1 WAIT Statement

Syntax:

WAIT expression

Use the "WAIT" statement to program delays and interruptions into the program execution.

Dwell time If the robot is supposed to remain in one position for a certain period, specify this period directly.

Dwell time for "WAIT"

- Measuring unit: Second
- Input range: 0.0 to MAX_REAL s

Programming:

The time is specified as numerical value following the "WAIT" statement, e.g. WAIT 8.5.

Example:

The robot transports a container to the feeder for filling purposes.

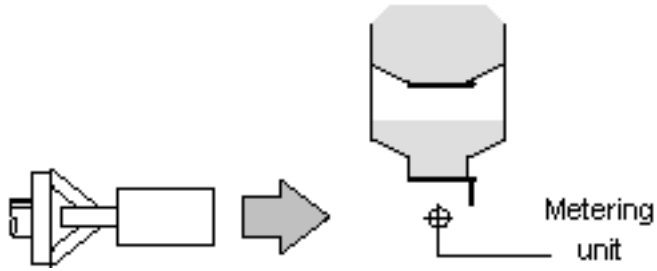
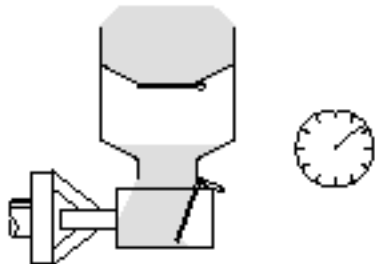
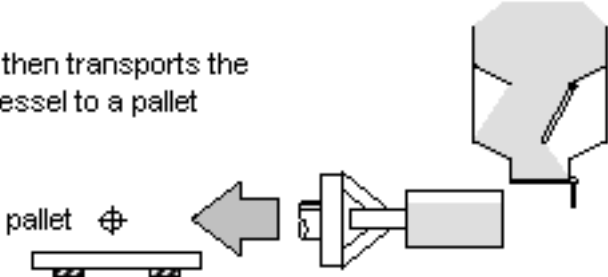
Syntax	Description
MOVE TO met_unit	
WAIT 8.5	The filling time is approx. 8.5 sec. 
MOVE TO pallet	It then transports the vessel to a pallet 

Fig.8-1: Program example 1 for WAIT statement

RCL Program Control

Waiting for the condition

Syntax:

```
WAIT UNTIL variable rel_operator expression [max_time=expression [ERROR statement]]
```

rel_operator: =, <>, <=, >=, <, >

If the robot is supposed to wait in one position for a condition, specify the condition together with the "WAIT" statement.

The program execution is interrupted until the condition is fulfilled.



Conditions can only be set via the input channel variables, also refer to [Communication Channels to PLC, Declaration and Programming](#), page 73.

Programming:

Attach the condition with the word "UNTIL" to the "WAIT" statement, e.g.

Program:

```
WAIT UNTIL pal_empty=1
```

If the condition consists of several input channels, the "WAIT" statement has to be divided into several steps.

Examples:

```
WAIT UNTIL signal=1  
WAIT UNTIL light=1  
MOVE TO pallet
```

Incorrect is:

- WAIT UNTIL signal=1 AND light=1

The compiler interprets this statement as follows:

- WAIT UNTIL signal= ((1 AND light) =1)

Example:

The robot transports a workpiece to a pallet changer.

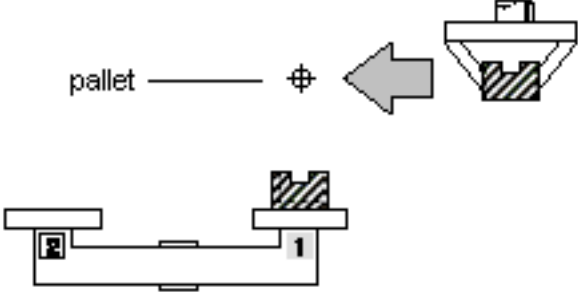
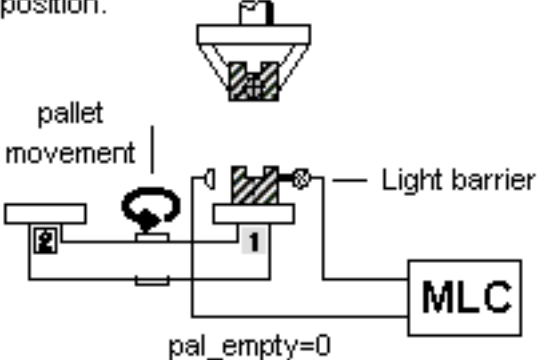
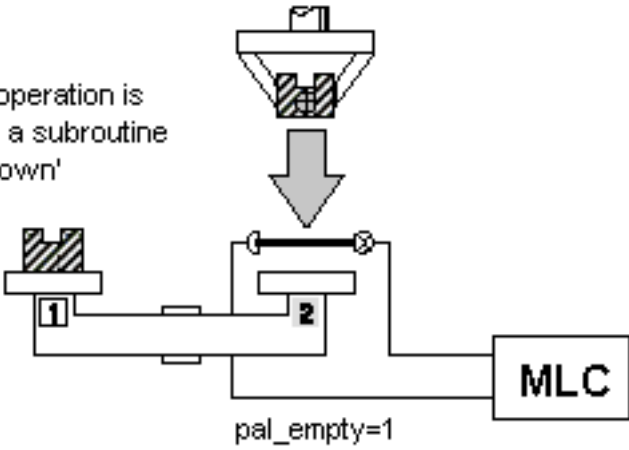
Syntax	Description
MOVE TO pallet	
WAIT UNTIL pal_empty=1	It waits until the pallet is empty. The pallet changer then puts the signal pal_empty=1, when an empty pallet has arrived at the change position. 
put_down	The deposit operation is programmed in a subroutine 'put_down'. 

Fig. 8-2: Programming example 2 for WAIT statement

Maximum waiting time *MAX_TIME*

- Measuring unit: Second
- Input range: 0.5 to 32000 s

A maximum waiting time can be specified with a WAIT condition.

RCL Program Control

Program execution is interrupted until the condition is fulfilled or the maximum time is exceeded.

Use the maximum waiting time to program an error statement. If the maximum waiting time is exceeded, the control executes the error statement.

Programming:

Specify the maximum waiting time with the RCL command MAX_TIME.

This is programmed after the WAIT condition.

Example:

```

WAIT UNTIL sig=1 MAX_TIME=60

```

The error statement is programmed after "MAX_TIME".

Example (error):

```

WAIT UNTIL sig=1 MAX_TIME=60 ERROR PAUSE

```

Sequence example

A container is filled with liquid.
 A sensor measures the weight of the container.
 If the container is full, the sensor outputs the signal "weight= 1". The filling system closes the valve and the robot removes the container. The average filling time is 25 seconds.
 The container may only remain below the filling device for a maximum of 45 seconds without putting the production at risk.

Syntax	Description
WAIT UNTIL weight=1 MAX_TIME=45 ERROR JUMP f_end ... f_end:	Error handling

Fig.8-3: Error handling in the sequence example

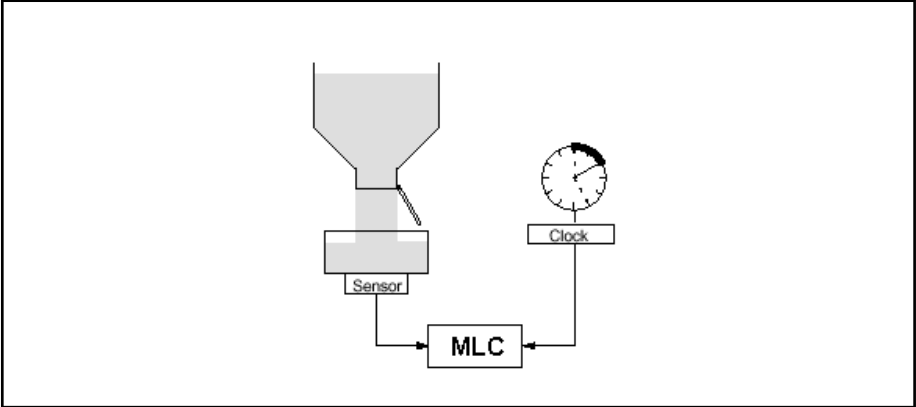


Fig.8-4: Filling procedure has begun

Case 1

- The container is completely filled.
The sensor reports "weight = 1" to the MLC.
- The robot removes the container.
- "MAX_TIME" was not reached.

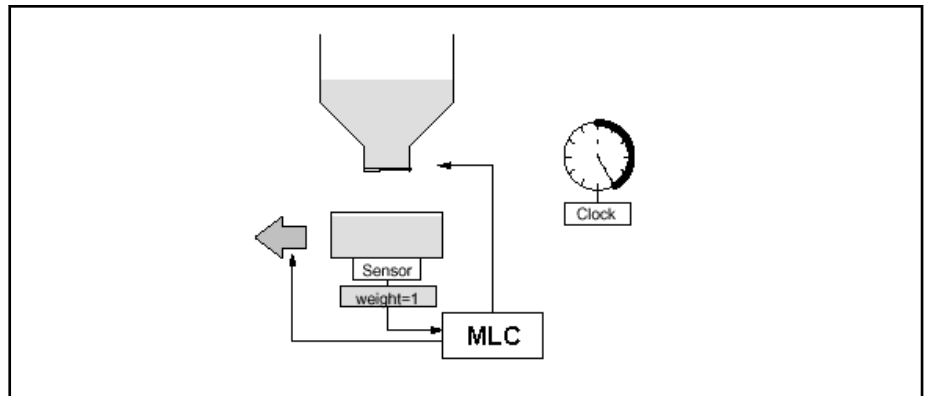


Fig. 8-5: Case 1, sufficient supply

Case 2

- The filling system is empty and the container is not completely filled.
- "MAX_TIME" is reached.
The program continues at "f_end".
- The "f_end" program instructs the robot to isolate the partially filled container.

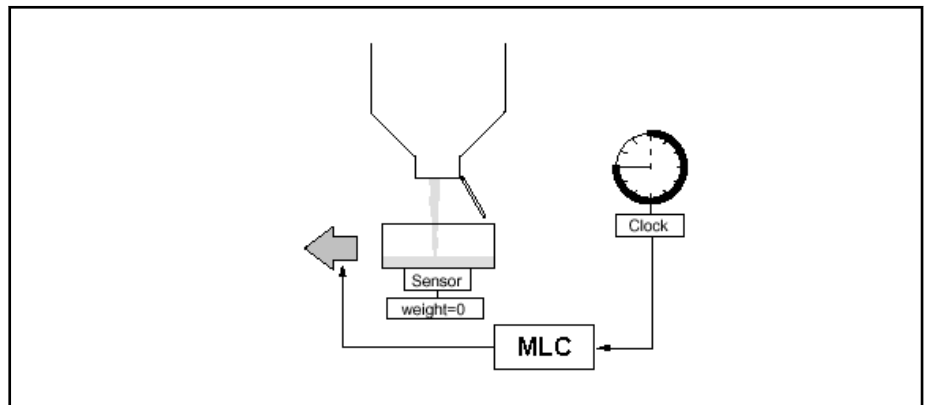


Fig. 8-6: Case 2, empty container

8.2 PAUSE Statement

The "PAUSE" statement can stop the program execution.

To continue the program, the PLC program has to call the MB_IRDProgStart function block again "MB_IRDProgStart".

Bit 26, IRD_PROGRAM_PAUSE is set in K-0-0021, Kinematic status.

KinData[n].dwKinStatus_i.26 is also influenced at the time.

Programming The "PAUSE" statement consists of the RCL command "PAUSE".



It is recommended to set an output before the "PAUSE" command to inform the operator on the program flow.

8.3 HALT Statement

The "HALT" statement ends the execution of a statement sequence in the main program.

RCL Program Control

Based on the "HALT" command, the control detects that the program has been completed during the program run.



In case of called external subroutines, "HALT" causes in a return to the calling, active main program.

Programming Enter the "HALT" command in front of the subroutine declaration or program end. The HALT command can be used several times in one program.



If the program halt can be detected from the program structure, e.g. in case of the RCL statement "PROGRAM_END", the control automatically generates the "HALT" command during compilation.

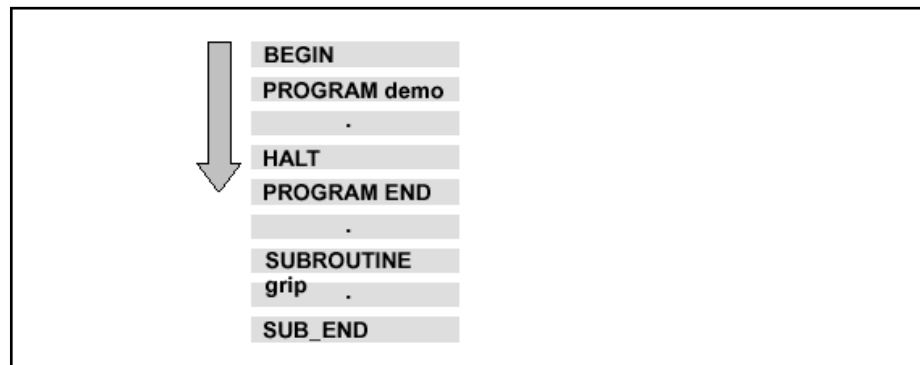


Fig.8-7: HALT statement in the program run

8.4 REPEAT Statement

Syntax:

```

REPEAT [expression] TIMES
statement(s)
REPEAT_END
  
```

With a "REPEAT" statement, a program section can be executed several times.

This is called program section repetition.

The program section is labeled by the repeat statement

"REPEAT n TIMES".

Numbers, variables or "INTEGER" type expressions (see [Overview, page 59](#)) can be used for the number of repetitions.

If the number= 0 or negative, the loop is not executed.

The program section ends with the RCL command "REPEAT_END".

Programming A certain program section is to be executed eight times, i.e. the RCL statement is "REPEAT 8 TIMES".

The program section is completed with "REPEAT_END".

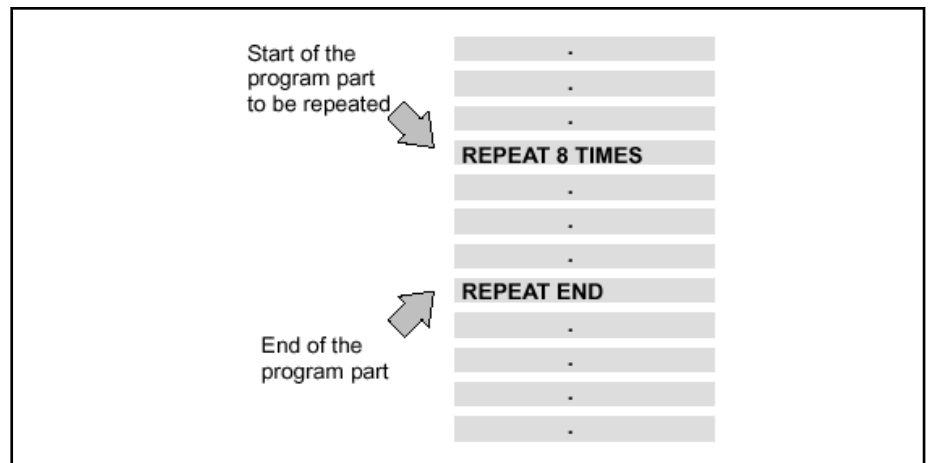


Fig. 8-8: REPEAT statement in the program run

8.5 JUMP Statement

Labels can be set in main programs and subroutines. These are used to jump to and from this position using a jump statement. Forward and backward jumps are possible.

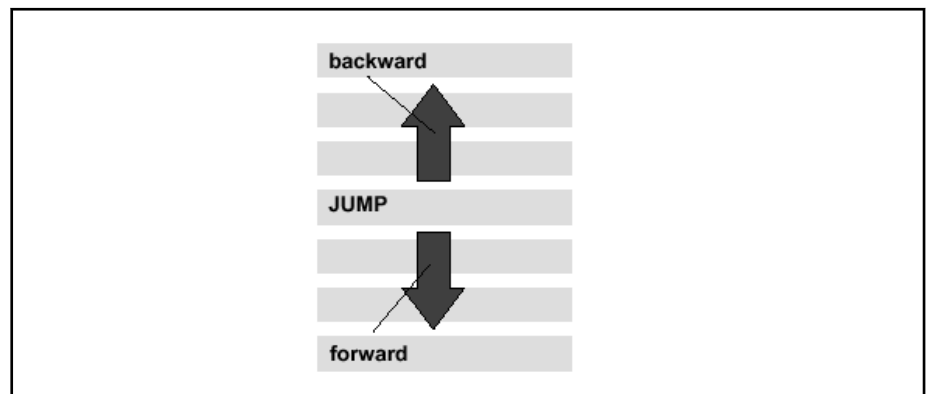


Fig. 8-9: Forward and backward jumps

The jump statement consists of the RCL command "JUMP" and the name of the set jump label.

The labels (jump targets) are identified with names. The name contains of twelve characters max.



The first character has to be a letter. Only the underscore "_" is allowed as special character. There is no case sensitivity.

For distinguishing purposes with regard to subroutine calls, the jump label is identified with a colon ":"



A specific jump label may only be used once! There can be any number of jump labels with different names.



Jumps from the main program to a subroutine and vice versa are not permitted!

RCL Program Control

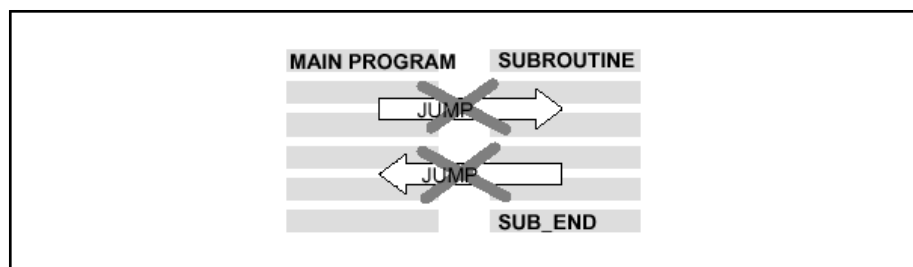


Fig.8-10: Jumps between the main program and a subroutine that are not permitted



Jumps to program section repetitions are also not permitted.
However, jumps out of program section repetitions are permitted.

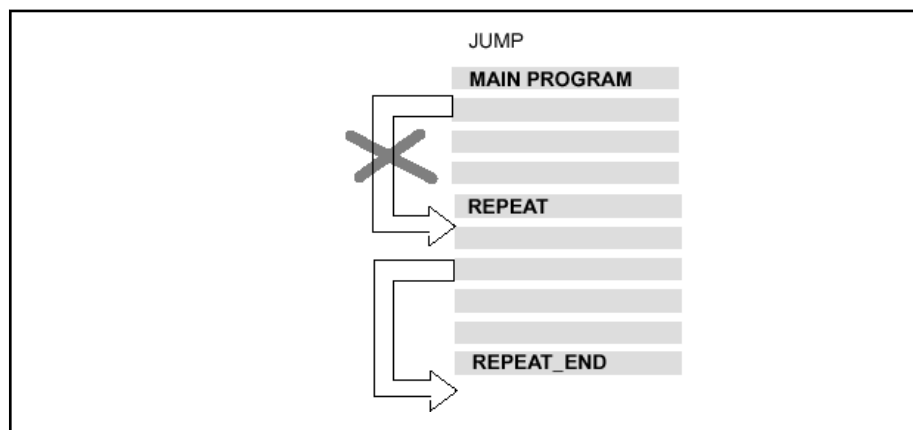


Fig.8-11: A jump to a loop is not permitted. A jump from the loop is permitted

Programming In a program, it is to be jumped forward to the jump label "table".

Syntax:

```
JUMP table ;Jump statement to jump label 'table'
```

```
table: ;Set jump label called table
```

The program is executed up to the jump statement "JUMP table".
This is followed by a jump to the jump label "table".
Subsequently, the control continues the program from this label.

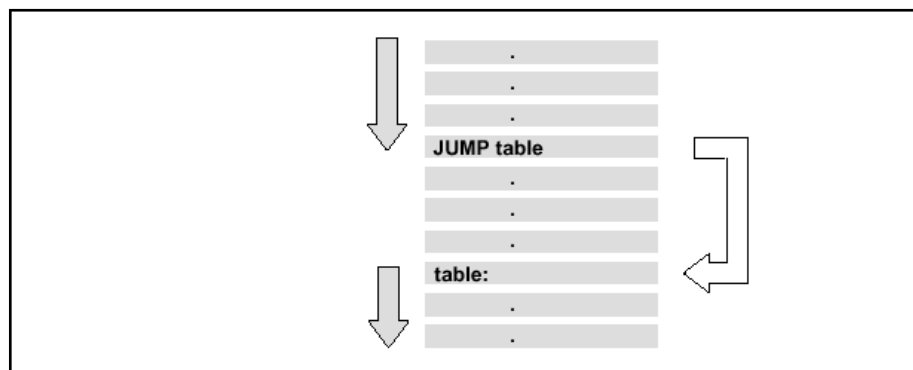


Fig.8-12: JUMP statement in the program run

8.6 IF-THEN-(ELSE) Statement

Syntax:

```
IF condition THEN statement
[ELSE statement]
```

```
IF condition THEN
  BEGIN
    statement(s)
  END
[ELSE statement]
```

```
IF condition THEN
  BEGIN
    statement(s)
  END
ELSE
  BEGIN
    statement(s)
  END
```

At freely selectable locations in a main program or subroutine, the remaining execution of the program can be conditional (based on conditional statements).

A condition is a "BINARY" type expression.

The condition is fulfilled if the statement is true, i.e. the variable actually has the specified value or a value in the specified range.

The condition is not fulfilled if the statement is false.

channel	=	1
l	=	15
sig1	AND	sig2
word	=	j
force	>	115.0
torque	>=	thresholdval



Conditions with AND, OR or NOT do not have to be placed in parentheses, see [Logic Expressions, page 91](#) for the desired priorities for condition operation.

Programming

The conditional statement is as follows:

```
IF condition THEN statement 1
  ELSE statement 2
```

"IF" the condition is fulfilled, "THEN" the control executes statement 1; "ELSE", if the condition is not fulfilled, the control executes statement 2.

RCL Program Control

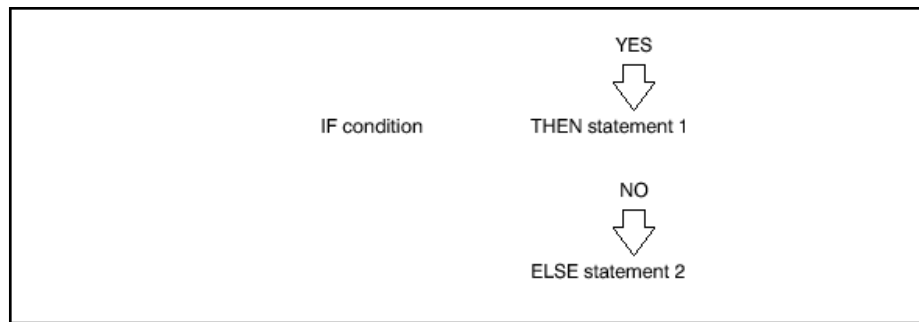


Fig.8-13: IF..THEN..ELSE

For example, possible statements include:

- Program calls (main program or subroutine calls)
- Jump statements (JUMP)
- Pause statements (PAUSE), Halt statements (HALT), delays (WAIT)
- Repetitions (REPEAT)
- Motion instructions (MOVE, MOVE UNTIL)
- Conditional statements (IF...THEN...ELSE), etc.

If no jump is programmed in the "THEN" or "ELSE" statements, the control continues the program run with the program steps following the conditional statement.

The "ELSE" statement can be omitted. In this case, the control also continues the program run with the program steps following the conditional statement.

Example

The robot is supposed search in a shelf for a pallet loaded with a workpiece, approx. weight 200 kg. It has a sensor in its lifting device for this purpose, which informs the control on the pallet weight.

If it has found the workpiece, it is supposed to bring it to the machine.

Starting position: The robot is positioned in front of the top pallet.

Program:

```

next:
  in                ;Calling subroutine 'in': Moving in and lifting the pallet
  IF weight >= 180.0
    THEN MOVE TO machine
    ELSE JUMP search
  sig= 1            ;Signal to PLC, pallet delivered
  PAUSE
  search
  down              ;SUBROUTINE put down, move back and down
  JUMP next
  
```

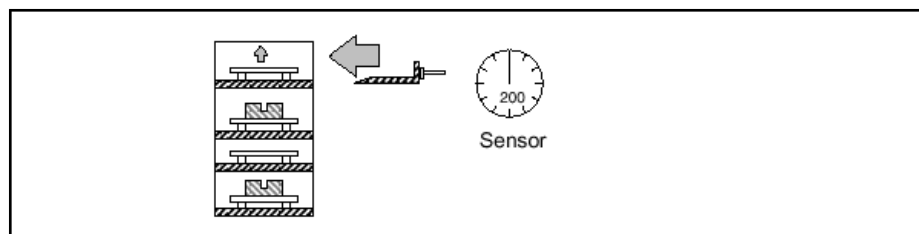


Fig.8-14: Traveling to the pallet rack

RCL Program Control

Program:

```
IF weight >= 180.0      ;Evaluating signal < 180.0 kg, empty pallet
  THEN MOVE TO machine
  ELSE JUMP search      ;Control decides for ELSE statement
                        ;Program jump on jump label 'search:'
```

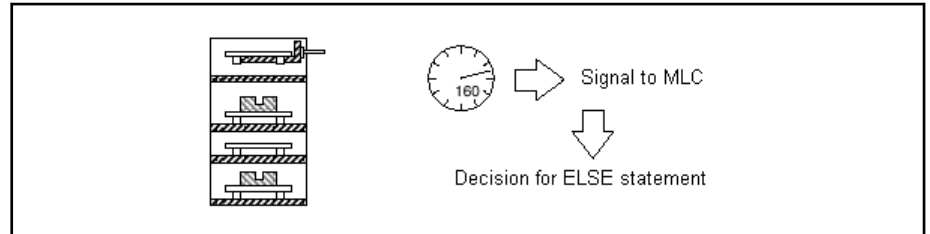


Fig. 8-15: Lifting an empty pallet

Program:

```
search
down                      ;Calling subroutine 'down': Put down the pallet,
                          ;Move out and move lower.
```

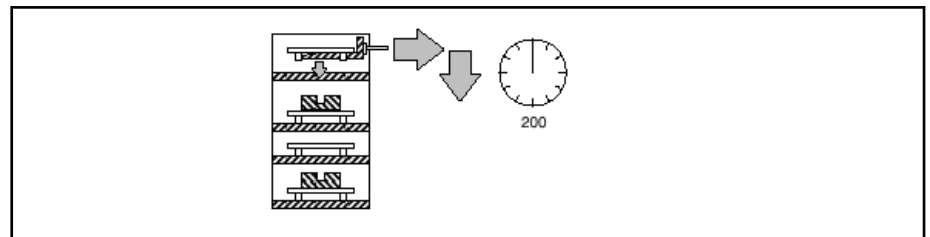


Fig. 8-16: Off-loading the empty pallet again

Program:

```
JUMP next                ;Program jump on jump label 'next:'

next:
in                        ;Calling subroutine 'in': Moving in and lifting the pallet
```

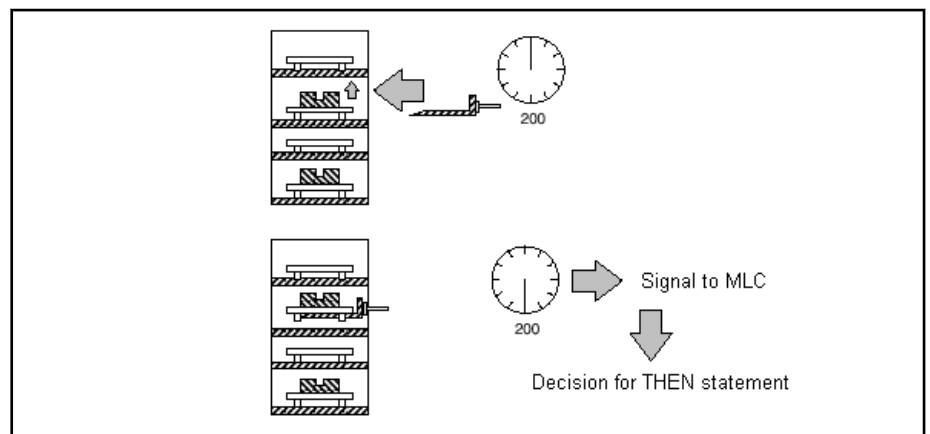


Fig. 8-17: Repetition, full pallet...

Program:

```
IF weight >= 180.0      ;Evaluating signal 200 kg ==> THEN
  THEN MOVE TO machine  ;Traveling towards 'machine'
  ELSE JUMP search
```

Program:

```
sig= 1                  ;Signal to PLC, pallet delivered
PAUSE                   ;Pause program and wait for PLC
```

RCL Program Control

8.7 CASE Statement

The branching statement "CASE" can be used to create a selection from several alternatives. Thus, nested "IF-THEN" queries can be avoided and the program execution time can be shortened.

Programming The conditional statement is as follows

Syntax:

```
CASE selection_expression
  EQUAL: statement
  EQUAL: statement
  DEFAULT: statement
CASE_END
```

Types allowed for **selection_expression** are "BINARY", "INTEGER", "CHAR" and "TEXT".

The constants in the constant list have to be compatible with the selection expression. A constant may only be used once in the branching statement.

Specifying the "DEFAULT" section is optional.



If the "DEFAULT" section is not in the branching statement, no runtime error is be output during the program execution.

This also applies if none of the statements are valid.

Program:

```
PROGRAM case_ex
INPUT INTEGER: amount      ;Value received from the PLC
INTEGER: offset            ;Declarations
BEGIN                     ;Program start
  offset=0
  CASE amount
    EQUAL 9,10: offset=0
    EQUAL 11,12: offset=1
    EQUAL 13: offset=2
    DEFAULT offset=5
  CASE_END
PROGRAM_END
```

9 Value Assignments and Operations

9.1 Value Assignments

Value assignments are used to assign values to variables for velocity and acceleration while a program is running. This also applies to default variables.

The value assigned has to be of the same data type as the variable.

The following section describes value assignments for general variables. Value assignment for positions can be found in [Point Variables, page 65](#).

Programming Syntax:

variable=expression

The "=" sign is used for the assignment.

The name of the variable to which a value is assigned has to be located on the left side of the assignment. An expression is located on the right side.

It is possible to enter for the expression:

- Numerical values (constants)
- Variable
- Arithmetic expressions
- Standard functions

The "+" sign can be omitted.

The "-" sign is located directly in front of the variables or constants. If two operators directly follow each other, the negative expression has to be placed in parentheses.

If all components of a variable of type "POINT" or "JC_POINT" are to be negated, all components have to be multiplied by -1. -1.0 also has to be placed in parentheses in this case.

Syntax:

counter=1	;Variable and numerical value
xvalue=yvalue+2.5	;Variable and arithmetic expression
distance=SIN(alpha)	;Variable and default function
value=-value	;Variable of type REAL and negation
d=d*(-2.0)	;Variable and negative expression
corner=(-1.0)*corner	;Variable of type POINT and negation

9.2 Operations

9.2.1 Arithmetic Expressions

Arithmetic expressions are operations with:

- Numerical values and/or
- Variables and/or
- Standard functions and/or
- Other arithmetic expressions

The operator defines the operation type.

RCL is familiar with five arithmetic operators:

+	Addition, e.g. k = 1 + 5
-	Subtraction, e.g. value = scaled-1

Value Assignments and Operations

*	Multiplication, e.g. length = width * 2
/	Division, e.g. height_new = height/2.0
MOD	Modulo calculation, only for INTEGER type, e.g. remaining = number MOD divisor



- The characters + and – can also be used as signs for variables and numerical values.
- Variables and numerical values with signs have to be placed in parentheses so that two operators do not directly follow each other.

No sign may be displayed on the left side of an assignment.

The addition and subtraction operations can be performed with variables and numeric values (constants) of the types "INTEGER", "REAL", ""POINT"" and "JC_POINT".

The multiplication and division operations can be performed with variables of the types "INTEGER" and "REAL".

"REAL" type variables and numerical values can also be used for multiplication and division of point variables, see [Point Variables, page 65](#).

The operators "*", "/" and "MOD" are executed before + and –.

Calculation takes place from left to right in these classes.

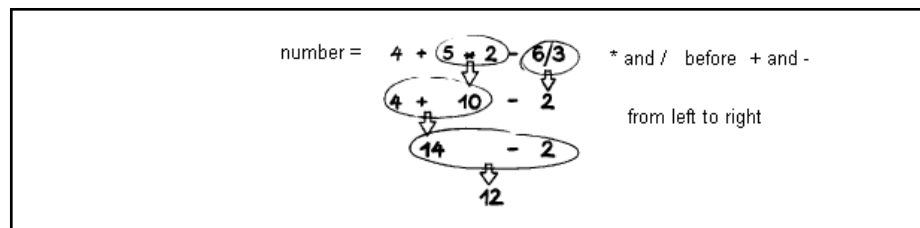


Fig.9-1: Calculation example 1

In addition, expressions that belong together can be placed in parentheses.

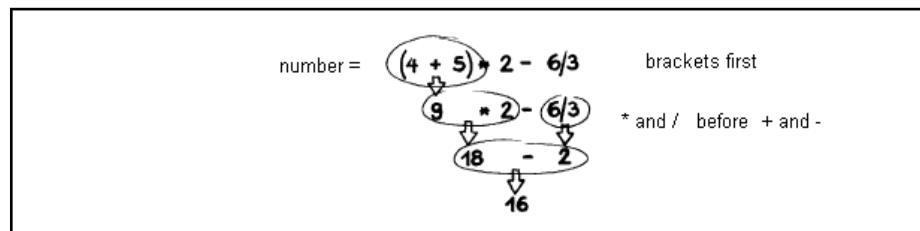


Fig.9-2: Calculation example 2

Modulo function value1 MOD value2

The modulo function, "INTEGER" data type, calculates the integral remainder calculated from the division of "value1" by "value2".

"value1", "value2" and the result are of type "INTEGER".

Determining the column of a pallet after specifying position number and column number.

Value Assignments and Operations

	Col. 0	Col. 1	Col. 2	Col. 3	Col. 4
Line 0	0	1	2	3	4
Line 1	5	6	7	8	9
Line 2	10	11	12	13	14
Line 3	15	16	17	18	19
Line 4	20	21	22	23	24

Fig.9-3: Calculation example 3

The pallet has 5 columns (0, 1, 2, 3, 4). In which column and which row can position 13 be found?

The position number has to be entered in "value1". The number of columns has to be entered in "value2".

Program:

```
column=13 MOD 5 ;Calculation: 13/5 = 2 rest 3. The rest indicates the column. Therefore,
                 ;column 3 is the wanted
line=13 / 5      ;Calculation: 13/5 = 2 rest 3. The quotient indicates the line. Therefore,
                 ;line 2 is the wanted
```

When "column" is called, the Modulo function determines where (in which column) the respective position is located based on the value of the remainder (=3 in this example).

When the line is called, the division function determines where (in which line) the respective position is located based on the value of the result (= 2 in this example).

9.2.2 Comparative Operations

The control queries values and states in conditions, e.g. UNTIL or IF conditions. These queries are made via comparisons.

Programming

The following characters can be used:

=	equal to, e.g. p = 1
<>	not equal to, e.g. p <> 1
>	greater than, e.g. p > 1
>=	greater than or equal to, e.g. p >= 1
<	less than, e.g. p < 1
<=	less than or equal to, p <= 1



"POINT", "JC_POINT" and "RECORDS" type variables (record variables) can only be queried with respect to "=" (equal to) or "<>" (not equal to).

9.2.3 Logic Expressions

The control checks conditions for validity, see also [Comparative Operations, page 91](#), and [IF-THEN-\(ELSE\) Statement, page 85](#). Conditions can only assume two values.

Value Assignments and Operations

"Value 1" for TRUE and "value 0" for FALSE.

This also applies to "BINARY" type variables.

Only one of the two values, 0 or 1, can be applied to such a variable.

Operation of conditions

The remaining program sequence often depends on several conditions at the same time.



"BINARY" type variables and expressions (conditions) can be combined with the logic operations AND, OR and NOT.

cond_1	cond_2	cond_1 AND cond_2
1	1	1
1	0	0
0	1	0
0	0	0

Fig. 9-4: Operation of two conditions cond_1 and cond_2 with AND truth table

cond_1	cond_2	cond_1 OR cond_2
1	1	1
1	0	1
0	1	1
0	0	0

Fig. 9-5: Operation of two conditions cond_1 and cond_2 with OR truth table

Negation of conditions

The validity of type "BINARY" conditions and variables can be negated with the word "NOT".

Example:

Condition cond_1 is true, so (cond_1) = 1

If the word "NOT" is placed before the condition cond_1, the following applies to the validity of "NOT" cond_1:

(NOT cond_1) = 0

or

Example:

The condition cond_2 is false, (cond_2) = 0 applies

Applies to the inversion with "NOT":

(NOT cond_2) = 1

9.2.4 Sequence of Operations

Programming operations with several conditions

Results of comparative operations are always of the "BINARY" data type. When combining several conditions, observe the following operator sequence.

1. NOT
2. *, /, MOD, AND

Value Assignments and Operations

3. +, -, OR
4. =, <>, >, >=, <, <=.

Example:

Query according to numerical values of the variables i and j of type "INTEGER"

IF i = 10 AND j = 50 THEN...

In this example, the control first processes the expression 10 AND j.

However, 10 AND j represents a type conflict for the control, since the constant 10 is of the "INTEGER" type and not of the "BINARY" data type.

Parentheses are used to define the sequence for processing expressions.

IF (i = 10) AND (j = 50) THEN...

10 RCL Functions

10.1 SIN Function

The sine function of data type "REAL" establishes the mathematical relationship between an angle and the lengths of the sides of a right triangle.

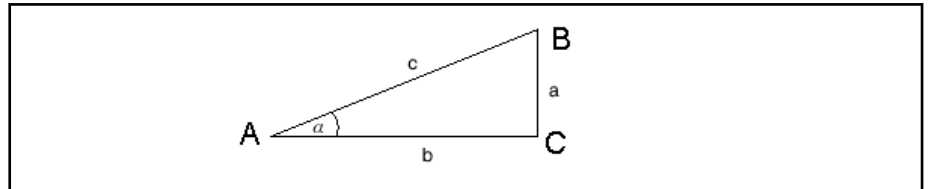


Fig. 10-1: Right-angle triangle

$$\sin \alpha = \frac{\text{opposite leg}}{\text{hypotenuse}} = \frac{a}{c}$$

Programming The angle α must be specified as radian (rad). The number is of type "REAL".

$$\text{rad} = \alpha \frac{\pi}{180^\circ} \Leftrightarrow \alpha = \text{rad} * \frac{180^\circ}{\pi}; \pi = 3,14$$

Syntax:

avalue= c*SIN(rad)



The designation "avalue" was chosen instead of "a" as "A" has already been reserved for acceleration.

10.2 COS Function

The Cosine function of data type "REAL" establishes the mathematical relationship between an angle and the lengths of the sides of a right triangle.

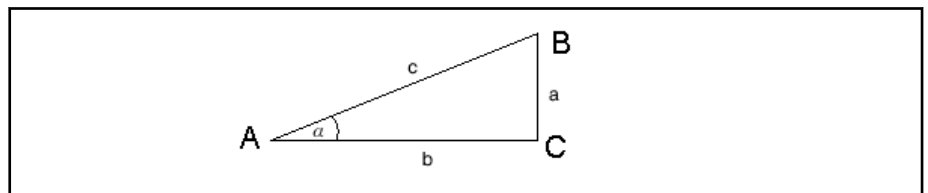


Fig. 10-2: Right-angle triangle

$$\cos \alpha = \frac{\text{adjacent leg}}{\text{hypotenuse}} = \frac{b}{c}$$

Programming The angle α must be specified as radian (rad). The number is of type "REAL".

Syntax:

b= c*COS(rad)

10.3 ATAN Function

The Arcus tangent function, of data type "REAL", determines the angle of a right triangle based on the lengths of the sides.

RCL Functions

The arc tangent function is the inverse of the tangent function, which can be described as follows:

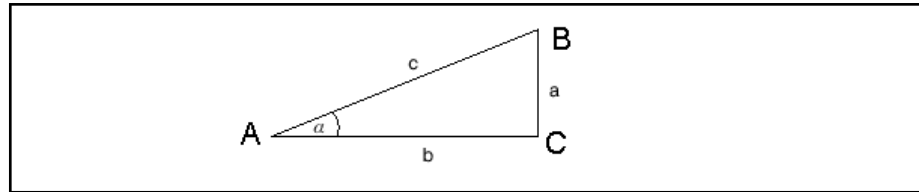


Fig.10-3: Right-angle triangle

$$\tan \alpha = \frac{\text{opposite leg}}{\text{adjacent leg}} = \frac{a}{b}$$

The inverse of the tangent function is

$$\alpha = \text{atan} \frac{a}{b}$$

The angle is established as a radian value "rad" and can be converted.

$$\text{rad} = \alpha \frac{\pi}{180^0} \Leftrightarrow \alpha = \text{rad} * \frac{180^0}{\pi}; \pi = 3,14$$

Syntax:

alpha= c***ATAN**(avalue/b)



The designation "avalue" was chosen instead of "a" as "A" has already been reserved for acceleration.

10.4 SQRT Root Function

The root function, "REAL" data type, determines the value of the square root of an expression.

Calculation of length c in a right triangle.

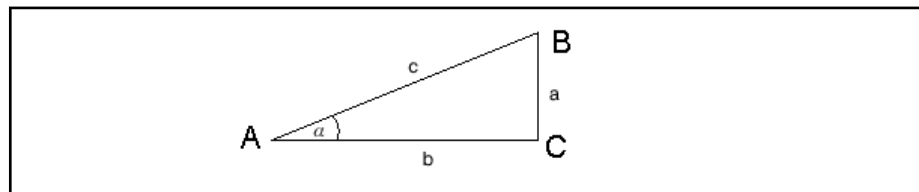


Fig.10-4: Right-angle triangle

Pythagorean theorem:

$$a^2 + b^2 = c^2 \Rightarrow c = \sqrt{a^2 + b^2}$$

Syntax:

c=**SQRT**(avalue*avalue+b*b)



The designation "avalue" was chosen instead of "a" as "A" has already been reserved for acceleration.

10.5 ABS Function

Syntax:

ABS(argument)

The result of the function is the absolute value of the transferred argument. The argument may be of type "INTEGER" or "REAL". The result is of the same type as the argument.

Example:

deviation=**ABS**(delta)

10.6 TRUNC Function

Syntax:

TRUNC(argument)

By cutting, the function converts the argument of type "REAL" to a value of type "INTEGER".

If there is a positive argument, the largest integer less than or equal to the argument results, e.g. 3.7 becomes 3.

If there is a negative argument, the smallest integer greater than or equal to the argument results, e.g. -3.7 becomes -3.

10.7 ORD Function

Syntax:

ORD(char_variable)

This function provides the INTEGER value of type "CHAR".

Example:

char_num=**ORD**(asc_zeich)

char_num=**ORD**('i')

10.8 CHR Function

Syntax:

CHR(integer)

This function provides a "CHAR" type value based on the "integer MOD 256" calculation.

Example:

asc_char=**CHR**(34)

10.9 ROUND Function

Syntax:

ROUND(argument)

The function converts the argument of type "REAL" to a value of type "INTEGER".

RCL Functions

The rounding is defined as follows:

- If the first number of decimals is either 0, 1, 2, 3 or 4, it is rounded off.
- If the first number of decimals is 5, 6, 7, 8 or 9, it is rounded up.

Example:

```
value_3=ROUND(7.81) ;value_3 is assigned to 8
value_4=ROUND(-5.43) ;value_4 is assigned to -5
value_5=ROUND(6.50) ;value_5 is assigned to 7
```

10.10 EOF Function

Syntax:

```
END_OF_FILE(argument)
```

This function queries whether the end of the file has been reached when reading a dat file.

The argument is a "FILE" type variable. The function supplies the value 1 (true) if the end of the file has been reached and the value 0 (false) if the end of the file has not been reached.

Example 1:

```
IF END_OF_FILE(dat_values) ;Using the function in a conditional statement
  THEN statements
  ELSE statements
```

Example 2:

```
EOF=END_OF_FILE(dat_values) ;Using the function in an assignment
```

10.11 ASSIGN Function

With the "ASSIGN" statement, the user program can be used to change file names while the program is running.



The ASSIGN statement can only be applied to closed files and standard channels.

Example:

```
FILE datvar,wrvar ;Files datvar, wrvar
ASSIGN wrvar,wname ;File variable wrvar <-- text variable wname
ASSIGN datvar,datname ;File variable datvar <-- text variable datname
READ_BEGIN datvar ;Read pointer to the start of the datvar file
READ datvar, oneline ;Reading a line
WRITE_END wrvar ;Write pointer to the end of the wrvar file
WRITE wrvar, oneline ;Writing a line
CLOSE datvar ;Closing the file
CLOSE wrvar ;Closing the file
PROGRAM_END
```

10.12 INT_ASC Function

The standard procedure "INT_ASC" converts an integer to an array of characters.



The format is right-justified, possibly with leading space characters. If an error is detected during the conversion, the character array remains unchanged and the corresponding error code is returned.

Syntax:

```

OUTPUT INTEGER: 401= toPLC
INTEGER: int_number      ;Specification
INTEGER: index_na       ;int_number = Number to be converted.
INTEGER: length_char     ;Only of type INTEGER, e.g.
INTEGER: error_ret       ;INT number <= 2147483647
ARRAY[1..10] CHAR: ascii_array ;Event feedback ascii_feld = Event array of type ARRAY[ ] CHAR
BEGIN
  int_number=-1234      ;Specification
  index_zf=1           ;Starting index in the character array
  length_char=5         ;Maximum number of characters
                        ;reserved for the number to be converted.
  INT_ASC(int_number,ascii_array,index_na,length_char,error_ret)
                        ;Previous initialization using blanks before conversion is carried out.
  IF error_ret<>0       ;Event feedback
    THEN toPLC= error_ret ;error_ret= Error number output
    ELSE toPLC= int_number

```

Description of the error numbers		
	0	No errors
	-1	Start index exceeds array limits
	-2	End index (start index + length) exceeds array limits
	-3	Reserved length is too small
	-4	Range exceeded (value too high)
	-5	Array length < 0
	-6	Array length = 0

10.13 ASC_INT Function

The standard procedure "ASC_INT" converts an array of characters to an integer.

Starting with the start position, the procedure reads characters until a character is detected that is not a number or until the maximum number of characters has been read or the end of the character array has been reached.

Syntax:

```

OUTPUT INTEGER: 401= toPLC
INTEGER: int_number      ;Result: converted number only as INTEGER
INTEGER: index_na       ;Array index from where the number is to be read
INTEGER: length_char     ;Maximum number of characters to be read
INTEGER: error_ret
ARRAY [1..10] CHAR: ascii_array ;Array to be converted; of type CHAR
BEGIN
  ascii_array='AaBc'
  index_na=1
  length_char=5
  ASC_INT(ascii_array,int_number,index_na,length_char,error_ret)
  IF error_ret<>0       ;Event feedback error_ret= Error number output
    THEN toPLC= error_ret ;error_ret= Error number output
    ELSE toPLC= int_number

```

RCL Functions

Description of error number	0	No errors
	-1	Start index exceeds array limits
	-2	End index (start index + length) exceeds array limits
	-3	Unused error codes
	-4	Range exceeded (value too high)
	-5	Array length < 0
	-6	Array length = 0
	-7	Character string does not start with a number or sign

10.14 SIZEOF Function

The standard function "SIZEOF" serves to determine memory space required for any RCL variables or types. Knowing the memory requirement for a given variable (data buffer) is important, especially when dimensioning transfer buffers, e.g. for communication with TCP/IP connections.

The function returns the memory requirement for the specified type or variable in bytes. The standard function "SIZEOF" can be used within the constant definition section.

Syntax of the implicit declaration:

```
sizeof=INTEGER: SIZEOF(type_or_variable) ;Standard function
```

The following applies to the syntax above:

- **type_or_variable:**
Any type or variable name.

The following table shows the storage area for standard types.

Data type	Size [BYTES]	Corresponding data type (C/C++)	Value range in RCL
BINARY	4	long	[0, 1]
INTEGER	4	long	[-2147483648 ... 2147483647]
REAL	4, simple precision	float	3.4E +/-38 (7 digits)
CHAR	1	unsigned char	[CHR(0)...CHR(255)]
POINT	axis number * 4 (REAL)	typedef float POINT [AxisNumber]	-
JC_POINT	axis number * 4 (REAL)	typedef float JC_POINT [AxisNumber]	-
WC_FRAME	6 * 4 (REAL)	typedef float WC_FRAME[6]	-
TEXT	80 * 1 (CHAR)	typedef char TEXT[80]	-
INPUT <type>	4 + SIZEOF <type>	typedef struct IN<type> {<type> value; long address} IN<type>	see <type>
OUTPUT <type>	4 + SIZEOF <type>	typedef struct OUT<type> {<type> value; long address} OUT<type>	see <type>

Data type	Size [BYTES]	Corresponding data type (C/C++)	Value range in RCL
FILE	28	-	-
BNR_FILE	28	-	-

Fig. 10-5: Memory requirement of RCL data types

Examples of the **SIZEOF** function:

```

;;JC_NAMES=a1,a2,a3,a4           ;Axis names
;;WC_NAMES=c1,c2,c3,c4          ;Coordinates
OUTPUT INTEGER: 401= szofReal, 402= szofPoint ;Transferring to PLC
PROGRAM size
POINT: EndPos
BEGIN
  szofReal= SIZEOF (REAL)           ;Edition: 4, Type REAL
  szofPoint= SIZEOF (EndPos)        ;Edition: 16 (4 axes * size of REAL)
PROGRAM_END

```

10.15 LIMIT_MIN, LIMIT_MAX, LIMIT_OFF Functions

With the assignments,

Program:

```

LIMIT_MIN= min_position
LIMIT_MAX= max_position

```

the workspace limitations can be defined.

The coordinates of the points **min_position** and **max_position** represent the spatial diagonals of the spanned cuboid.

in a formal sense, MIN_POSITION and MAX_POSITION are regular world coordinate points.

As all WC points, they refer to the local workpiece coordinate system, with the result that LIMIT_MIN and LIMIT_MAX are programmed in workpiece coordinates.

LIMIT_OFF switches the monitoring off.

10.16 JC Standard Function

The standard function "JC" provides a conversion from world to joint coordinates, e.g.

Program:

```
@start_pos= JC(start_pos)
```

The result is the arm position of the kinematics. Since the starting point is a world coordinate point, the result depends on the active workpiece coordinate system.



"JC" always supplies values from the main area if joint or world coordinates could be ambiguous.

10.17 WC Standard Function

The standard function "WC" supplies a conversion from joint to world coordinates, e.g.

RCL Functions

Program:

```
start_pos= WC(@start_pos)
```

This results in coordinates that refer to the local workpiece coordinate system, i.e. the result depends on the active workpiece coordinate system.



"WC" always supplies values from the main area if joint or world coordinates could be ambiguous.

11 RCL Special Functions

11.1 RCL Special Functions, General

Special functions allow the RCL software engineer to access the functions in the control for which no RCL language elements have been reserved. Special functions represent an extension of the RCL language scope.



They can be called in a program if they have been declared before the call similar to variables.

Currently, the following special functions are available for the IndraMotion MLC:

SPC_FCT	Brief functional description
1	iol (position-accurate switching of digital outputs on the path)
2	ppo (position-accurate switching of process data on the path)
21	belt_kind (belt type)
23	clock_date (system date and time)
24	sys_count (system counter, real-time counter)
28	set_belt (set belt counter reset value)
53	belt_area (belt range at belt type 4)
59	Query blending mode
60	Set blending mode

Fig. 11-1: RCL Special Functions

Declaring special functions

The declaration of a special function includes function code and name as well as name and data type of the transfer parameters. The transfer parameters are used to define when, where and how the function is to take effect. The declaration has to be made in the program declaration.

The name of the special function and the names of the transfer parameters can be freely selected by the user.



The specification of the respective special function fixedly determines the data types.

Example:

SPC_FCT: x = example (VALUE INTEGER: wap)

SPC_FCT: x	Function number
Example	Special function name
VALUE INTEGER:	Data type designation for the transfer parameter, e.g. decimal value
wap	Name of the transfer parameter (placeholder)

Fig. 11-2: Example of a special function

Calling the special function

The call in the program statement is executed by entering the special function name and defining the declared transfer parameters.

RCL Special Functions

In the call, name of the special function and transfer parameter type have to be followed as specified in the program declaration.

11.2 RCL Special Functions, Listed as Table

Syntax	Description
SPC_FCT:1=IOL (VALUE INTEGER : IONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: RateTime)	Position-accurate switching of digital outputs on the path The declaration of the special function includes code 1 and the special function designation as well as the names and type designations of the function sizes used in the special function call to define which control value should be output when the corresponding axis is at a given position.
SPC_FCT: 2=PPPO (VALUE INTEGER : PPONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: RateTime)	Position-accurate switching of process data on the path The declaration of the special function includes the code 2 and the special function designation as well as the names and type designations of the function sizes used in the special function call to define which process value should be output when the corresponding axis is at a given position.
SPC_FCT: 21 = belt_kind (VALUE INTEGER : belt_no VALUE INTEGER: kind_belt)	Belt-synchronous workspace of belt type 4 This special function can be used to select the desired type of belt synchronization. At this time, the following synchronizations are possible: kind_belt = 4: Default
SPC_FCT: 23 = clock_date (INTEGER: hours,minutes,day,month,year)	System clock Using the special function 23, the MLC system clock can be accessed from a RoCo program. Day, month, year, hours and minutes are determined.
SPC_FCT: 24= sys_count (INTEGER: c_count)	System counter The variable transferred when special function 24 is called, is filled with the value of the system counter. The variable value is given in milliseconds [ms]. The counter matrix corresponds to the clock time. Every time the control is booted, the real-time counter is reset to zero.
SPC_FCT: 28= belt_set (VALUE INTEGER : belt_no VALUE REAL: reset_value)	Set belt counter Special function 28 assigns a reset value to the belts. The RCL command "SYNC" can set the belt counter to its reset value (default= 0).

RCL Special Functions

Syntax	Description
SPC_FCT: 53 = belt_area (VALUE INTEGER : belt_no JC_POINT: @Base REAL: FullLength REAL: BeginLength)	Geometric position of the belt range with belt type 4 belt_no: Number of the belt for which the value is to be set or read (1..4). If belt_no is negative, the values are read out. @Base: JC_POINT of the beginning of the belt-synchronized workspace. FullLength: Total length of the belt-synchronous workspace in belt direction. The value must be greater than zero BeginLength: Latest start for belt-synchronized operation. The value must be greater than zero and less than or equal to the total length (FullLength).
SPC_FCT: 59 = getBlendMode (VALUE INTEGER : blMode)	Query blending mode blMode=0: The blending radius is limited to half the distance of the total distance (default) blMode=1: The blending radius can reach the entire distance of the shorter distance if possible
SPC_FCT: 60 = setBlendMode (VALUE INTEGER : blMode)	Set blending mode blMode=0: The blending radius is limited to half the distance of the total distance (default) blMode=1: The blending radius can reach the entire distance of the shorter distance if possible

Fig. 11-3: Special functions

11.3 Position-accurate Switching

11.3.1 Position-accurate Switching, General

The MLC allows switching on one path. During a travel motion, signals can be transferred from a RoCo program to the PLC in advance and with accurate position information using the KinData interface.

See also [RobotControl Functionality Programmed in the PLC, page 253](#).

Two special functions are available for this task.

- With special function 1, digital outputs can be enabled at the interface.
In the following, special function 1 is identified as I/O logic (switch on/switch off logic, IOL).
- With special function 2, integers can be output.
In the following, special function 2 is identified as PPO logic (process parameter output logic). Use these values to control your technology, e.g. measuring the amount of paint in painting applications or switching on or off the spray gun.

Position-accurate output Accurate position output means that a specific position of the tool center point (gripper position) is defined in the program that sets the output to the KinData interface at the PLC.

RCL Special Functions

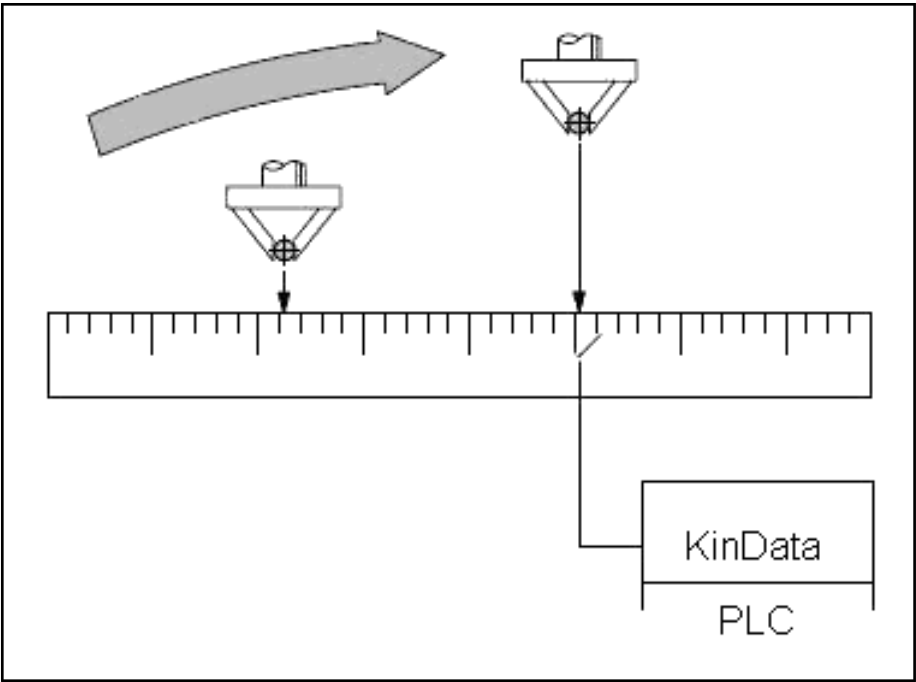


Fig.11-4: Accurate position output at the KinData interface

Programming Before the special functions are called, they have to be declared in the RoCo program declaration similar to variables. It must be ensured that the parameter list corresponds exactly to the following declaration.

11.3.2 **SPC_FCT: 1, Position-accurate Switching of Digital Outputs on the Path**

The declaration of the special function includes code 1 and the special function designation as well as the names and type designations of the function sizes used in the special function call to define which control value should be output when the corresponding axis is at a given position.

Declaration Program:

```
SPC_FCT:1=IOL(VALUE INTEGER: IONo
VALUE INTEGER: KinNo
VALUE INTEGER: CoordNo
VALUE REAL: SwitchPos
VALUE REAL: ParamValue
VALUE INTEGER: RateTime)
```

IONo	The KinData interface has 32 available outputs that can be enabled with special function 1 (OUTPUT BINARY 1..32).
KinNo	Number of the kinematics (1..16) to which the switching position is referred.
CoordNo	Number of the coordinate within the kinematics to which the switching position is referred.
SwitchPos	Position switched to, i.e. where the parameter value is output.

RCL Special Functions

ParamValue	The parameter value is preset as a decimal value where a value < 0.5 is output as logic 0 (low) and a value ≥ 0.5 is output as logic 1 (high).
RateTime	Time, in [ms], which elapses between the actual output of the parameter value and the reaching of the switching position. This means that the digital output is already switched before reaching the desired position (lead time). Thus, peripheral reaction times can be compensated.



No belt-synchronous coordinates may be used with this function!

Enabling signals *For the special function to operate correctly, the following states must be set:*

- Automatic,
- No emergency stop
- All axes are assigned to the kinematics with MB_AxisToGroup,
- In K-0-0241 feed halt cannot be set.

Program:

```
PROGRAM prog1
VAR:
SPC_FCT:1=ioout(VALUE INTEGER: chan,kin,coord
                VALUE REAL: po, va
                VALUE INTEGER: tm)
OUTPUT BINARY: 7= ToVaOn      ; corresponds to KinData.dwUserBits_i.6
BEGIN
  ioout(7,1,3,-300,1.0,50) ; 50ms (tm) before the Cartesian z-axis (coord)
                           ; reached position (po) -300, value (va) 1 is output on
                           ; output channel (chan) 7
MOVE LINEAR TO (10,20,-300)
PROGRAM_END
```

11.3.3 SPC_FCT: 2, Position-accurate Switching of Process Data on the Path

The declaration of the special function includes code 2 and the special function designation as well as the names and type designations of the function sizes used in the special function call to define which control value should be output when the corresponding axis is at a given position.

Declaration *Program:*

```
SPC_FCT:2=PPO(VALUE INTEGER: PPONo
              VALUE INTEGER: KinNo
              VALUE INTEGER: CoordNo
              VALUE REAL: SwitchPos
              VALUE REAL: ParamValue
              VALUE INTEGER: RateTime)
```

PPONo	The KinData interface is provided with 4 integer outputs that can be switched with special function 2 (OUTPUT INTEGER 1..4).
KinNo	Number of the kinematics (1..16) to which the switching position is referred.
CoordNo	Number of the coordinate within the kinematics to which the switching position is referred.
SwitchPos	Position switched to, i.e. where the parameter value is output.

RCL Special Functions

ParamValue	The parameter value is preset as decimal value.
RateTime	Time, in [ms], which elapses between the actual output of the parameter value and the reaching of the switching position. This means that the digital output is already switched before reaching the desired position (lead time). Thus, peripheral reaction times can be compensated.



No belt-synchronous coordinates may be used with this function!

Enabling signals *For the special function to operate correctly, the following states must be set:*

- Automatic,
- No emergency stop
- All axes are assigned to the kinematics with MB_AxisToGroup,
- In K-0-0241 feed halt cannot be set.

Program:

```
PROGRAM prog1
VAR:
SPC_FCT:2=ppo(VALUE INTEGER: chan,kin,coord
               VALUE REAL: po, va
               VALUE INTEGER: tm)
               OUTPUT INTEGER: 403=ToVaOn ; corresp. to KinData.adiUserDInt_i[3]
BEGIN
ioout(403,1,3,-300,42.0,50) ; 50ms (tm) before the Cartesian z-axis (coord)
                           ; reached position (po) -300; value (va) 42 is output on output channel
MOVE LINEAR TO (10,20,-310)
PROGRAM_END
```

11.4 SPC_FCT: 21, Belt Type

With special function 21, the type of belt synchronization is selected. After the control is booted, synchronization type 1 is active.



Currently, only synchronization type 4 is approved for use.

If the synchronization type is changed, this setting is valid (for all processes) until it is modified again or the control is rebooted.

Declaration *Program:*

```
SPC_FCT:21 = belt_kind (VALUE INTEGER: belt_no
                       VALUE INTEGER: kind_belt)
```

belt_no	Belt number (1 to 4)
kind_belt	Belt synchronous type: 4= belt type 4 with an axis parallel to the belt and with the possibility for a belt-synchronized PTP travel

Example *Program:*

```
PROGRAM SPZ21 ;Program name
               ;declaration of the special function
SPC_FCT : 21 = belt_kind (VALUE INTEGER: belt_no
                         VALUE INTEGER: kind_belt)
```

```
BEGIN belt_kind(1,4)      ;selecting the belt type 4 for belt 1
PROGRAM_END
```

11.5 SPC_FCT: 23, DateTime

With special function 23, the MLC system clock can be accessed from an RCL program. Day, month, year, hours and minutes are determined.

Declaration *Program:*

```
SPC_FCT:23 = DateTime(INTEGER: hour, minute,
                     day, month, year)
```

hour	hour
minute	minutes
day	day
month	month
year	year

Example *Program:*

```
PROGRAM SPZ23                      ;program name
VAR:
INTEGER: hr, mi, dy, mo, ye      ;declaration of the special function
SPC_FCT : 23 = DateTime(
INTEGER: hour,
                minute,
                day,
                month,
                year)
BEGIN
    DateTime(hr, mi, dy, mo, ye)   ;returns the current data and time in
                                ;hr, mi, dy mo, and ye
PROGRAM_END
```

11.6 SPC_FCT: 24, SysClk

The variable transferred when special function 24 is called is specified by the system counter value.

The variable value is given in milliseconds [ms]. The counter matrix corresponds to the MC cycle time (see C-0-0410). Every time the control is booted, the real-time counter is reset to zero.

Declaration *Program:*

```
SPC_FCT:24 = SysClk(INTEGER: clks)
```

clks	Number of motion cycles (MC) passed after the reboot

Example *Program:*

```
PROGRAM SPZ24                      ;program name
VAR:
INTEGER: clks                      ; declaration of special functions
SPC_FCT : 24 = SysClk(INTEGER: clks)
BEGIN S
    SysClk( clks )                 ;supplies number of motion cycles in clks
PROGRAM_END
```

RCL Special Functions

11.7 SPC_FCT: 28, Set Belt Counter

Special function 28 assigns a reset value to the belts.

The belt counter can be set using the RCL command "SYNC".

Declaration *Program:*

```
SPC_FCT:28=belt_set(VALUE INTEGER: belt_no  
                   VALUE REAL: reset_value)
```

belt_no	Belt number (1 to 4)
reset_value	Reset value in [mm] for the belt counter.

Enabling signals *For the special function to operate correctly, the following states must be set:*

- Automatic,
- No emergency stop
- All axes are assigned to the kinematics with MB_AddAxisToGroup
- In K-0-0241 feed halt cannot be set.

Example *Program:*

```
;;INCLUDE head           ;head contains compiler statements  
PROGRAM SPC28            ;program name  
SPC_FCT:28=belt_set(VALUE INTEGER: belt_no ;declar. of special function  
                   VALUE REAL: reset_value)  
  
BEGIN  
belt_set(2,200)  
PROGRAM_END
```

The reset value remains active until special function 28 is called again for the same belt. If the control is rebooted, the reset value is set to 0.

For the belt synchronization in an RCL program, the internal belt counters are also set to the reset value active at the time the function is called. An example is

Program:

```
SYNC band1, E1=1 ; belt 1 is set to the reset value
```

Program:

```
SYNC belt3 ≥ 200 ; 200 is once subtracted from the current belt counter value  
                ; of belt3.  
                ; if the current belt counter is < 200, the function waits  
                ; until the counter is increased to 200 and subsequently subtracts 200.
```

11.8 SPC_FCT: 53, Belt Synchronous Operating Range of Belt Type 4

Before using SPC_FCT: 53, belt-synchronized workspace, the kinematics must be defined with SPC_FCT: 21, belt type 4.

The working principle is shown in the following illustration.

RCL Special Functions

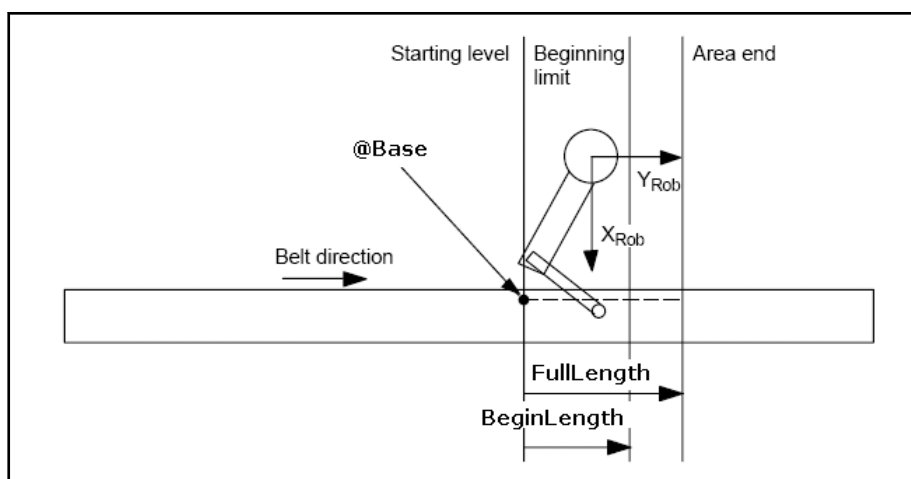


Fig. 11-5: Belt-synchronized workspace

The belt direction is defined by parameter K-0-0071, Coupling factors belt 1 (K-0-0072/belt 2... K-0-0074/belt 4).

In the figure the belt is rotated by +90 degrees to the robot coordinate system, i.e., Coord1=0, Coord2=1, Coord3=0. (A coordinate's value is the cosine (cos a) of the angle generated by the belt and the corresponding coordinates).

The starting level runs through the machine coordinate point @Base and is perpendicular to the belt direction.

The start and end of the area are parallel to the starting level.

A check is made during the first travel motion in the belt-synchronized section of belt type 4, to see whether the end point of the travel motion is located between the starting level and the beginning limit. If it has not yet reached the starting level, the travel motion is delayed until the end point has passed the starting level. If the end point of the first belt synchronized traveling motion has already exceeded the start limit, an error message is displayed (F12C00B4, belt sync.: 1. BS travel block activated too late).

When executing the belt-synchronized section of belt type 4, the tool center point (TCP) to the area end is monitored. If it is determined that the TCP will pass the end of the area or has already passed it, an error message is output (F12C00AE, Belt-sync.: travel block activated too late).

During the final belt-synchronized travel block (desynchronization and travel to fixed position), the area end is not monitored.

Declaration Program:

```
SPC_FCT:53 = belt_area (VALUE INTEGER : belt_no
                        JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)
```

belt_no	Number of the belt for which the belt value is to be set (1 to 4) or to be read (-1 to -4).
@Base	JC_POINT of the start of the belt-synchronized workspace
FullLength	Full length of the belt synchronized workspace in the belt direction. The value must be greater than zero.
BeginLength	Latest start for belt-synchronized operation. The value must be greater than zero and less than or equal to the full length.

RCL Special Functions

After the control is booted, the initial values for the belt range are:

- @Base all components 0.0
- FullLength= 0.0
- BeginLength= 0.0

Belt type 4 cannot be used with these values.

They must be set to reasonable values at least once with SPC_FCT: 53 = belt_area (...) to reasonable values. They are globally valid for all processes until they are modified again.

If a kinematics has more than one belt, each belt has its own belt range.

The currently valid values for the belt ranges can be read out by calling SPC_FCT: 53 = belt_area (...) with negative belt numbers. The values are then written to the transfer variables.

Requirements for input data:

- FullLength must be greater than zero.
- BeginLength must be greater than zero and less than or equal to FullLength.
- All components of the machine coordinate point @Base must be defined.
- If there is a process in the belt-synchronized section with belt type 4 for the kinematics, belt_area should not be modified.
- The machine coordinate point @Base must be located within the work-space.
- Five equidistant machine coordinate points from the machine coordinate point @Base in the belt direction up to the end of the area are specified internally.

These five machine coordinate points must also be located within the work-space of the kinematics. The geometric dependence of belt motions on axis motions is determined based on this total of six machine coordinate points.

Example Program:

```

PROGRAM beltsyn
VAR:
  CONST :
    C_BELT_NO_1 = 1,           ; belt number
    C_BK_PTP    = 4           ; belt type PTP
  BELT :
    500 + C_BELT_NO_1 = belt1

                                ; declaration of special functions
SPC_FCT : 21 = belt_kind (VALUE INTEGER: belt_no
                                VALUE INTEGER: kind_belt)
SPC_FCT : 53 = belt_area (VALUE INTEGER: belt_no
                                JC_POINT: @Base
                                REAL: rTotalLength
                                REAL: rBeginLength)

BEGIN

                                ; belt type specification
                                ; specifying belt-synchronous operating range
belt_kind ( C_BELT_NO_1, C_BK_PTP )
rTotalLength = 560
rBeginLength = 400
base = (280, 140, -600, 0, 0, 0)
@Base = JC(Base)
belt_area ( C_BELT_NO_1, @Base, rTotalLength, rBeginLength)

```

Several belts within one kinematics

If a kinematics has several belts, the special function can be used with the same declaration for all of the kinematic belts. "belt_no" is used for distinguishing purposes.

Several kinematics



An RCL program can move exactly one kinematic. The RoCo is based on the following philosophy:

1. the RCL program is responsible for the robot's motion
2. the PLC program takes care of all other tasks (communication, coordination, system/process control...)

"JC_POINT" type variables are related to a kinematics. If this special function is to be used for several kinematics, it must be declared with different names, e.g. "belt_area1".

The number (53 = ...) has to be the same for all kinematics with kinematic-related "JC_POINT" type variables, e.g. Robi_1.JC_POINT.

Example:

Program:

```
SPC_FCT :53=belt_area1 (VALUE INTEGER: belt_no
                        Robi_1.JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)

SPC_FCT :53=belt_area2 (VALUE INTEGER: belt_no
                        Robi_2.JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)
```

11.9 RCL Programming "Blending Mode"

11.9.1 SPC_FCT: 60, Set Blending mode

The blending mode can be selected in the RCL program using SPC_FCT 60. This mode applies to all subsequent commands in the block buffer.

Program:

```
SPC_FCT: 60 = setBlendMode(VALUE INTEGER : blMode)
```

blMode=0: The blending radius is limited to half the distance of the total distance (default)

blMode=1: The blending radius can reach the entire distance of the shorter distance if this is possible

The programmed blending radius R is taken into consideration in both modes.

In case of **blMode** = 0

the effective blending radius R_w results from the minimum of the programmed radius R , half the distance s_1 and half the distance s_2 .

$$R_w = \min \left(R, \frac{1}{2}s_1, \frac{1}{2}s_2 \right)$$

Fig. 11-6: Effective radius when blending with blending mode 0

In case of **blMode** = 1

the effective blending radius R_w results from the minimum of the programmed radius R , the total distance s_1 and the total distance s_2 , if this is possible.

RCL Special Functions

$$R_w = (R, n_1 \cdot s_1, n_2 \cdot s_2)$$

Fig. 11-7: Effective radius when blending with blending mode 1

$n_{(1,2)} = 1$ or $1/2$

s_1 = distance 1

s_2 = Path2

$n_{(1,2)}$ is automatically selected by the control by means of the programmed travel commands.

If the travel command is a "MOVE TO" command, $n = 1$ for this distance.

If the travel command is a "MOVE VIA" command, n depends on the specified command.

11.9.2 SPC_FCT: 59, Query blending mode

The blending mode set in the block preparation can be queried in the RCL program using SPC_FCT 59.

Program:

```
SPC_FCT: 59 = getBlendMode(VALUE INTEGER : blMode)
```

blMode=0: The blending radius is currently limited to half the distance of the total distance (default)

blMode=1: The blending radius can reach the entire distance of the shorter distance if this is geometrically possible.

Example:

```
;;COMPILER = RCL3
;;INT = LINEAR
;;KINEMATICS: ( 1= Kin1)
;;Kin1.JC_NAMES = vAchse1,vAchse2,vAchse3,vAchse4,vAchse5,vAchse6
;;Kin1.WC_NAMES = kx,ky,kz,kc,kb1,kb2
```

```
PROGRAM halfFull
```

```
VAR:
SPC_FCT : 60 = setBlendMode (VALUE INTEGER : blMode)
INTEGER : blMode
POINT : P1, P2, P3, P4
BEGIN
```

```
V=200 ; 0.2 m/s
A=1000 ; 1 m/s^2
R=1000 ; 1 meter = control calculates the max. blend radius
```

```
; because the moved distances will be lower then 1 meter
```

```
BLOCK_SLOPE ; has no effect because this is the default
setBlendMode( 0 ) ; has no effect because this is the default
```

```
; points for the rectangle
P1 = (-300, 0, -650, 0, 0, 0)
P2 = (-300, 200, -650, 0, 0, 0)
P3 = ( 300, 200, -650, 0, 0, 0)
P4 = ( 300, 0, -650, 0, 0, 0)
```

```
; no blending
MOVE LINEAR TO P1
MOVE LINEAR TO P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4
```

RCL Special Functions

```
; blending with half distances

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with half distances

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR VIA P3
MOVE LINEAR TO P4

; blMode = 1
setBlendMode( 1 )

; no blending

MOVE LINEAR TO P1
MOVE LINEAR TO P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with complete distances of P1P2 and P2P3

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with complete distance of P1P2
; and half distance of P2P3
; blending with half distance of P2P3
; and complete distance of P3P4

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR VIA P3
MOVE LINEAR TO P4

PROGRAM_END
```

12 RCL Motion Statements

12.1 General Information

Robot motions are initiated by motion statements. Motion statements describe the motion of a robot from a current position and orientation to a target point.

It is distinguished between:

- Direct motion statements (specification of **V**, **A** etc.)
- Motion-influencing motion statements

12.2 Direct Motion Statements

12.2.1 Overview

A direct motion statement consists of the following single statements:

- Motion command
- Interpolation method
- Additional information
- Abort condition (opt.)
- Approaching method
- Target position

The following table provides an overview on possible entries. The possible combinations result from the following individual descriptions.

Motion statements				
Motion command	Interpolation method	Additional information	Abort condition	Approaching method
MOVE	LINEAR	V	UNTIL	TO
MOVE_REL	CIRCULAR	A		VIA
	PTP	J		APPROX
		R		EXACT
		VFACTOR		
		AFACTOR		
		DFACTOR		
		V_PTP		
		R_PTP		

Fig. 12-1: Overview on motion statements

Programming an abort condition is optional.

Approaching method and target position have to be specified for each motion command.

Statement-specific information

The WITH command defines the additional information **V**, **A**, **J** and **R** only statement-specifically. The definition of the interpolation method is executed directly after the motion command without the WITH command.

RCL Motion Statements

These specifications do not apply anymore for the following motion commands.

It is not possible to program the **AFACTOR**, **VFACTOR** and **DFACTOR** statement-specifically.

Global information If no global statement-related additional information is commanded with the motion command, the global commands are valid. They include all additional information (V, A, J, R etc.) and the interpolation method (LINEAR, CIRCULAR and PTP).

At the beginning, the respective default values are valid. As soon as these are defined, the following motion commands apply.



If no additional information is specified for the motion command, the global additional commands apply.

12.2.2 Motion Commands

The control is familiar with the following motion commands:

- MOVE
- MOVE_REL

MOVE

Syntax:

```
MOVE[Interpolation method][additional information][abort condition]
[TO]target point[VIA target point[TO target point]
```

The control interprets all positions programmed after MOVE as absolute positions. The coordinate values refer to the zero point of the world coordinate system or joint coordinate system.



The motion command MOVE can be combined with all interpolation methods.

Program:

```
target=(10,10)
MOVE TO target
```

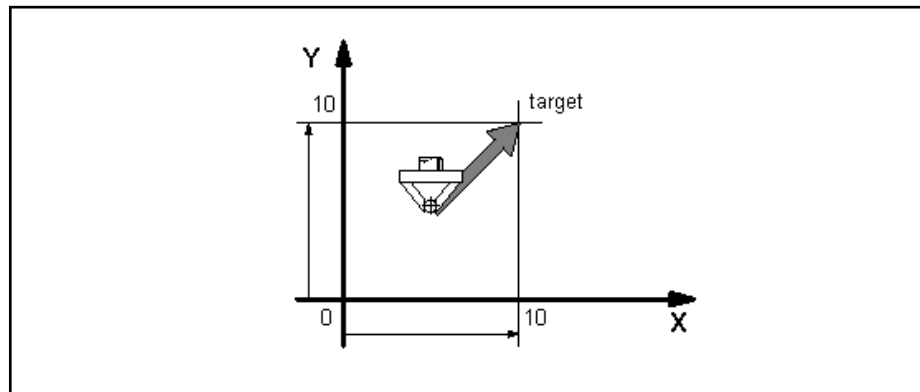


Fig.12-2: MOVE command

Additional information in the travel command are the decisive factor whether the robot travels exactly to the programmed points – i.e., within the defined tolerance - or whether it travels past the points without stopping.

RCL Motion Statements

When traveling in absolute position with MOVE, the TO and VIA specifications apply.

MOVE VIA ...(blending) The robot travels past the positions without stopping. See "R" and "blending".

Program:

```
MOVE VIA pos1, pos2, pos3
```

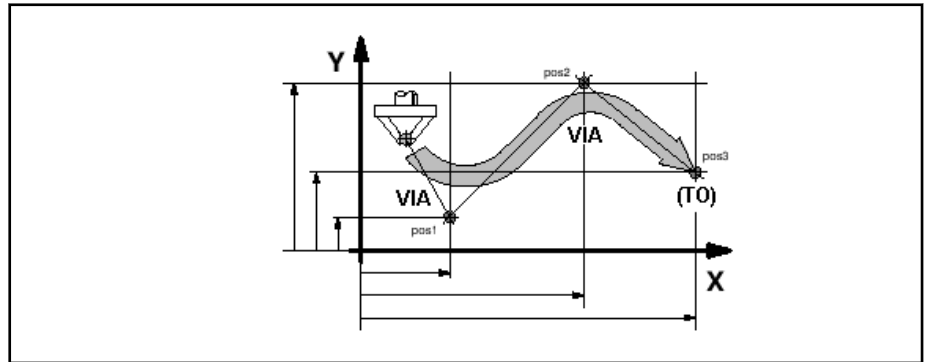


Fig. 12-3: MOVE VIA ...(blending)

MOVE TO ... The robot travels successively to the positions with intermediate stop. In this case, "TO" is selected as default addition.

Program:

```
MOVE TO pos1, pos2, pos3
```

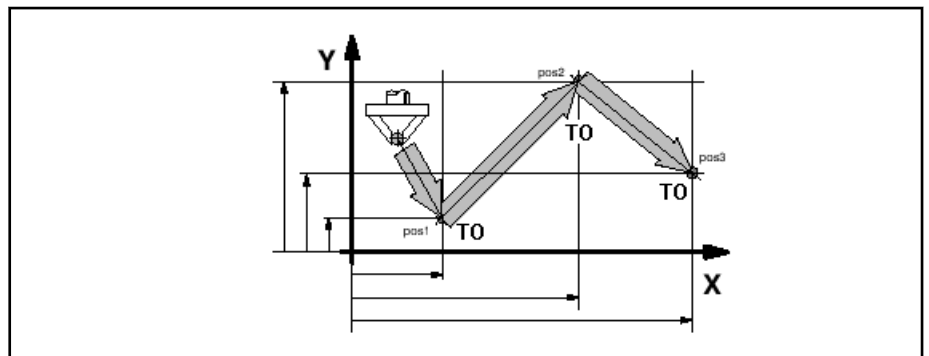


Fig. 12-4: MOVE TO ...

It is possible to link VIA and TO in a travel command.

MOVE VIA ... TO ... The robot travels past position 1 without stopping and then travels successively to the positions 2 and 3 with intermediate stop.

Program:

```
MOVE VIA pos1 TO pos2, pos3
```

RCL Motion Statements

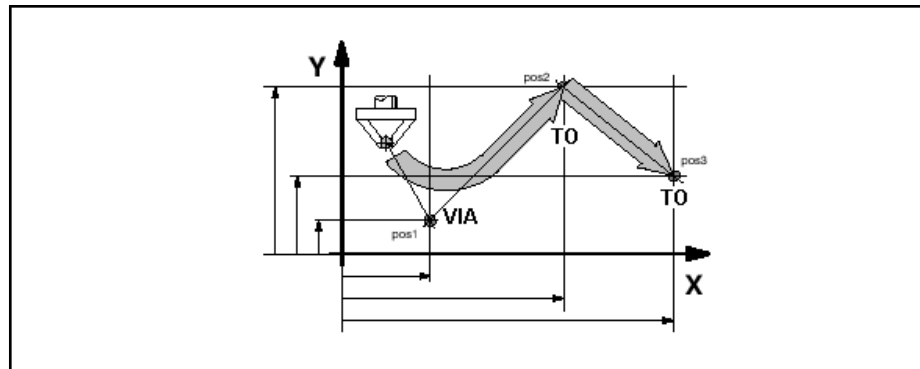


Fig.12-5: MOVE VIA ... TO ...

MOVE_REL*Syntax:*

```
MOVE_REL[Interpolation method][additional information][abort condition]
([EXACT]target point|APPROX target point[EXACT target point] )
```

The control interprets all positions programmed after MOVE_REL as incremental dimensions.

The coordinate values are distance values in the respective coordinate system and refer to the last commanded position of the robot.



The motion command MOVE_REL can be combined with all interpolation methods.

Program:

```
target=(10,10)
MOVE_REL target
```

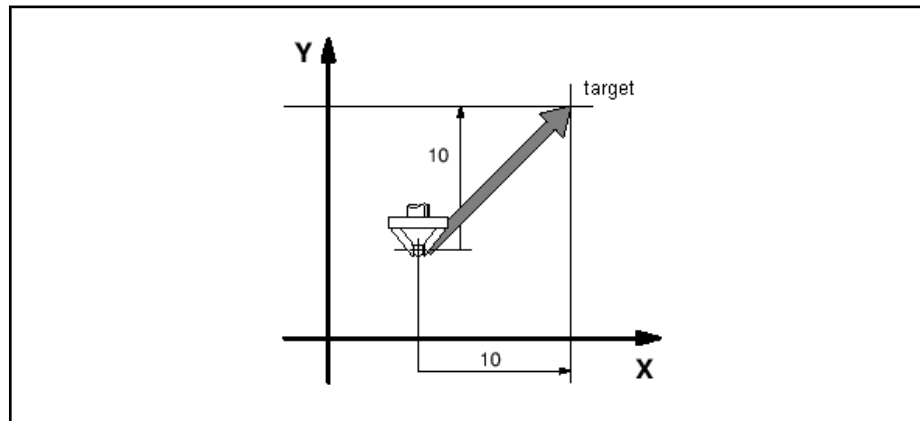


Fig.12-6: MOVE_REL

When traveling in incremental dimensions with MOVE_REL, the words "APPROX" and "EXACT" are used to specify an exact stop or traveling past.

MOVE_REL APPROX...

The robot travels past the positions defined in incremental dimensions without intermediate stop. See "R" and "blending".

Program:

```
MOVE_REL APPROX dis1,dis2,dis3
```

RCL Motion Statements

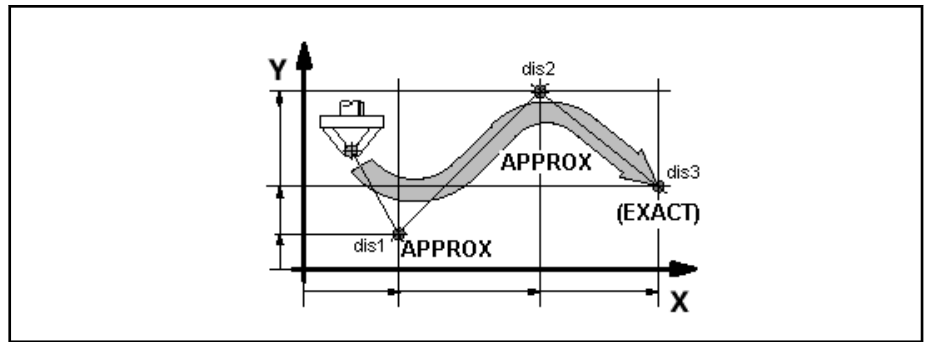


Fig. 12-7: MOVE_REL APPROX...

MOVE_REL EXACT...

The robot successively travels to the positions defined in incremental dimensions with intermediate stops. The word "EXACT" can be omitted when programming the travel command. The control automatically generates the command.

Program:

```
MOVE_REL EXACT dis1,dis2,dis3
```

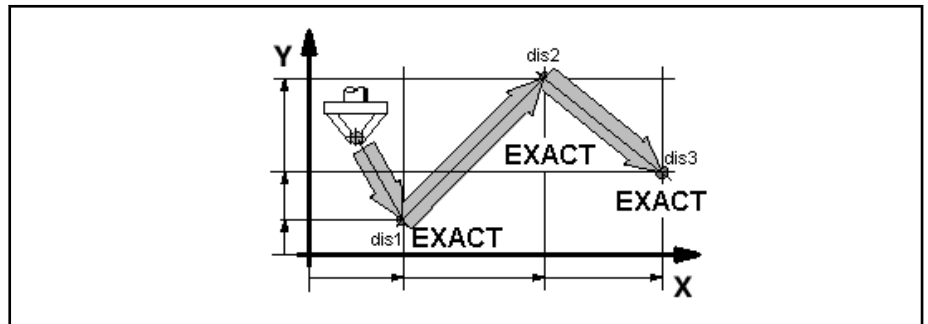


Fig. 12-8: MOVE_REL EXACT...

It is possible to link "APPROX" and "EXACT" in the travel command.

MOVE_REL APPROX ... EXACT ...

The robot travels past the first position without intermediate stop and then travels to the next positions 2 and 3 with intermediate stop.

Program:

```
MOVE_REL APPROX dis1 EXACT dis2,dis3
```

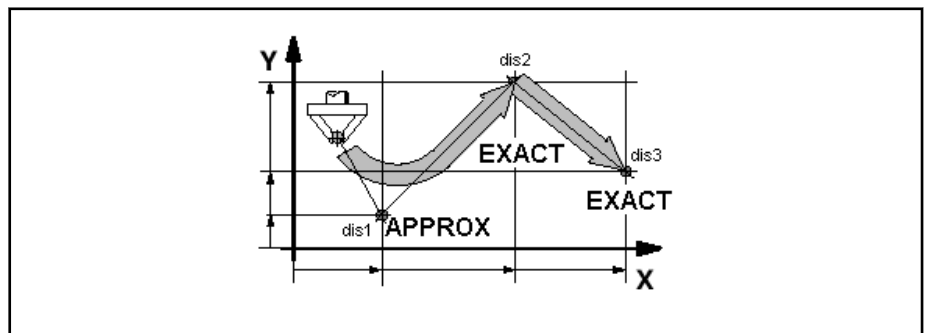


Fig. 12-9: MOVE_REL APPROX ... EXACT ...

RCL Motion Statements

Target point names The names of the point variables can be freely selected, see [Point Variables, page 65](#). For example, the names "palett1", "palett2" ... can be assigned to pallet points.

Due to reasons of clarity, the examples on the following pages, point specifications in absolute position are named with "pos", e.g. "MOVE pos".

Point specifications in incremental dimensions are named with "dis" (distance values), e.g. "MOVE_REL dis".

12.2.3 Interpolation Method

The MLC RobotControl supports one kinematic technique in three interpolation methods:

- LINEAR (straight line in space)
- CIRCULAR (circular path in space)
- PTP (synchronized point-to-point)

The interpolation methods refer to the interpolation in the space. Depending on the kinematics of the robot, the axis motions are executed after the inverse position kinematics (from space coordinates to axis coordinates).



Orientation axes of a kinematics are not transformed. For the interpolation, they are triggered in a fully synchronized (identical ramp times) way.

LINEAR interpolation method

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

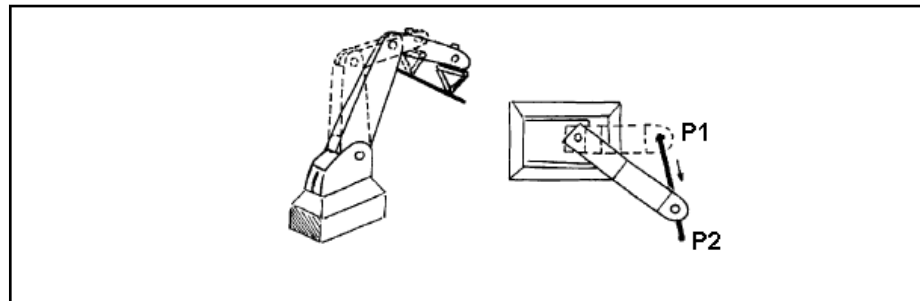


Fig. 12-10: LINEAR interpolation method

CIRCULAR interpolation method

The robot approaches the target point on a circular path in the space.

The circle is geometrically defined by three points.

Apart from specifying target point P2, an intermediate point has to be specified "PZ" so that the control can correctly calculate the circular path.

The control is familiar with point P1 as it was the last point approached. The intermediate point PZ is a point on the circular arc passed by the robot.

RCL Motion Statements

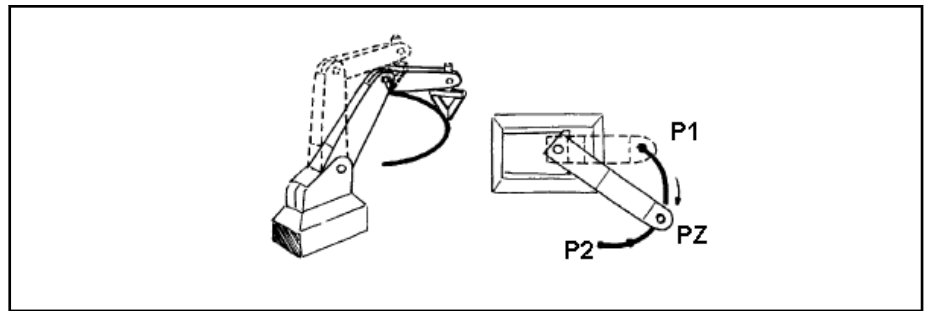


Fig. 12-11: CIRCULAR interpolation method

PTP interpolation method

PTP: Point-To-Point

The control calculates the motion of all axes. Thus, they start and stop moving simultaneously. The same ramp times apply to all axes. In general, this is referred to as fully synchronous PTP. Depending on the lever ratio of the robot arms and the points P1 and P2, this can result in a spatial distance not defined in detail.

Specifying a target point is sufficient for the PTP method.

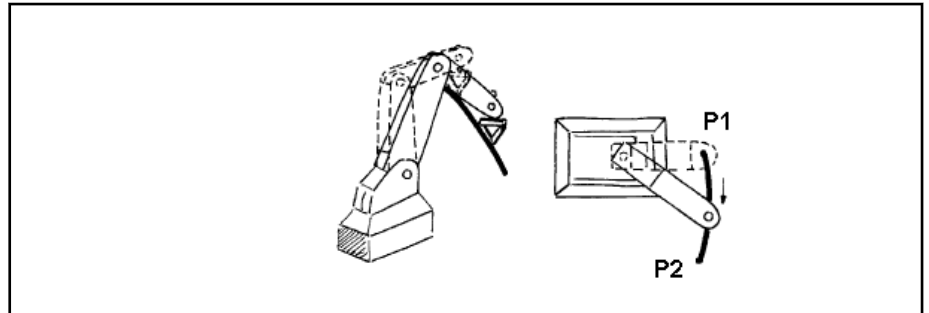


Fig. 12-12: PTP interpolation method

Statement-related interpolationmethod

If a certain interpolation method is supposed to apply to only one motion statement, the interpolation method is programmed in the motion statement.

The interpolation method is included in the travel command, directly following "MOVE" or "MOVE_REL".



Only one interpolation method can be in one motion function!

Example: LINEAR

Program:

```
MOVE_LINEAR TO pos1
MOVE_REL_LINEAR APPROX dis1 EXACT dis2
```

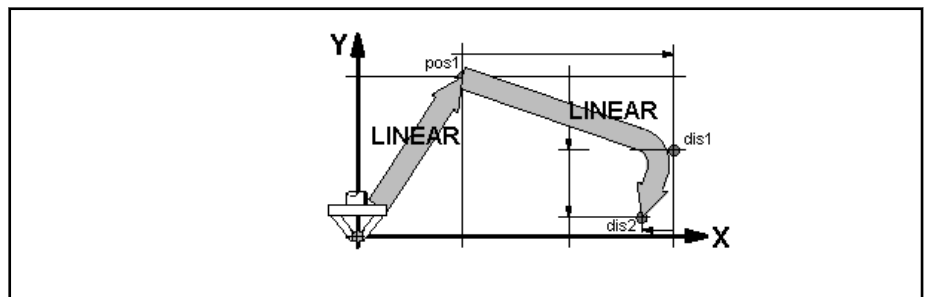


Fig. 12-13: Example: LINEAR

RCL Motion Statements

Example: CIRCULAR

The point specifications required for the CIRCULAR method, such as the intermediate point "pz" or "dz", end position "pos1" or dis1 have to be separated by a comma and placed in parentheses.

Program:

```
MOVE CIRCULAR TO (pz,pos1)
```

```
MOVE_REL CIRCULAR EXACT (dz,dis1)
```

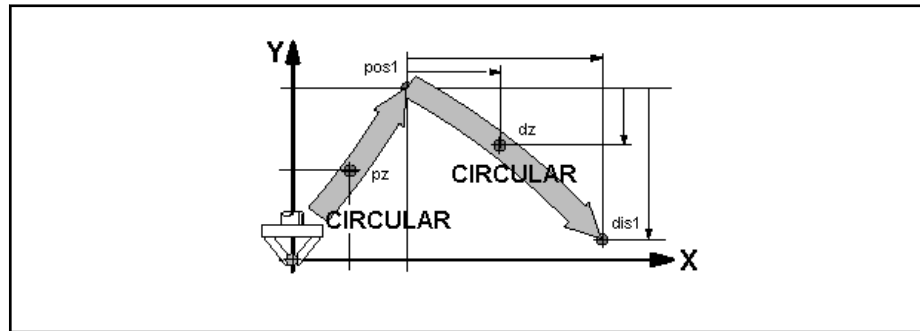


Fig. 12-14: Example: CIRCULAR

Example: PTP

Program:

```
MOVE PTP TO pos1
```

```
MOVE_REL PTP EXACT dis1
```

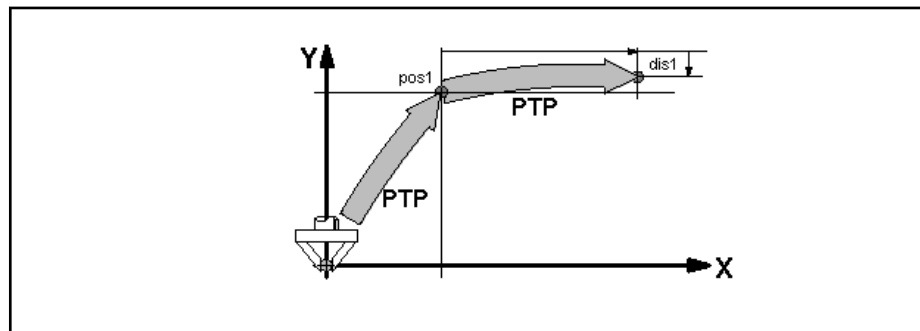


Fig. 12-15: Example: PTP

Global interpolation method

If an interpolation method is to apply to several motion statements, it can be specified as global interpolation method. The global interpolation method is specified by the following compiler statement.

Syntax:

```
;;INT= [LINEAR, CIRCULAR, PTP] ;Interpolation mode, all kinematics
```

This is integrated into the RCL program header by default. The defined interpolation method applies to all subsequent motion statements without any defined interpolation specification.

Example:

```
;;INT=LINEAR
MOVE pos1 ;The position pos1 and the point defined with the help of dis1,
           ;are approached by a robot on a straight line each.
MOVE_REL dis1
MOVE CIRCULAR(pz,pos2) ;Point pos2 is approached on a circular path.
```

RCL Motion Statements

The global interpolation method remains valid until it is replaced by another interpolation method.

Example:

```
;;INT=LINEAR

MOVE pos1      ;The robot approaches the axes of the kinematics 1 to the positions
MOVE pos2      ;pos1 and pos2 on a straight line each

;;INT=CIRCULAR

MOVE (pz1,pos3) ;It is approached towards the positions pos3 and pos4 and all remaining positions on a circular
MOVE (pz2,pos4) ;according to the interpolation specification CIRCULAR
```



If a CIRCULAR interpolation is specified as global method, the point values within the motion statement to which the interpolation specification is supposed to apply, must be valid point pairs.



If no interpolation method is specified and there is no global specification regarding an interpolation method, the control automatically selects PTP.



If several global interpolation methods are programmed, the specification of the interpolation method occurs in ascending line sequence and not according to the program sequence (subroutine, jumps, etc.).

Example:

PTP is used to approach "pos2" although the compiler statement ";;INT = PTP" was skipped in the program sequence.

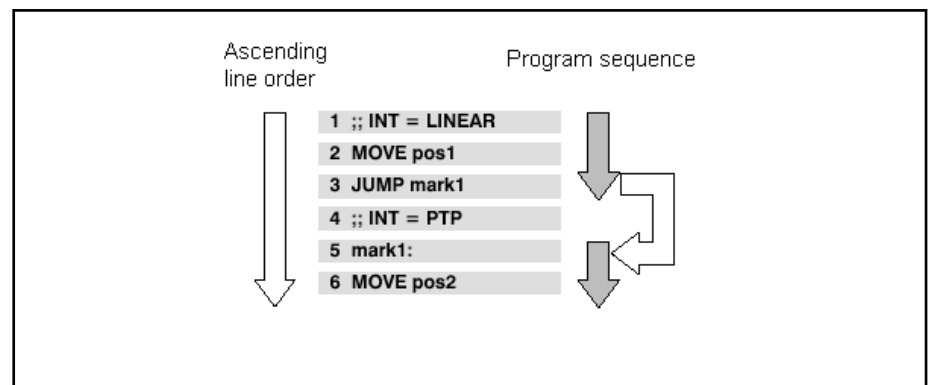


Fig. 12-16: Ascending line sequence versus program sequence

12.2.4 Mode of Operation of Motion-Influencing RCL Statements and Parameters

Interpolation method: LINEAR/ CIRCULAR (RCL syntax)

	Explanation	Calculation	Effect
V	Path velocity		SV
VFACTOR	Path velocity override	$V_{eff} = V * VFACTOR$	SV
VFIX	Deactivates K-0-0240, RoCoFeedrate override		SV
A	Path acceleration		SV

RCL Motion Statements

	Explanation	Calculation	Effect
J	Path jerk		SV
AFACTOR	Path acceleration override	$A_{eff} = A * AFACTOR$	SV
DFACTOR	Path deceleration override	$Deff = A * DFACTOR$	SV
AFIX	Deactivated K-0-0242, RoCoAcceleration override		SV
R	Blending radius		SV

SV Block preparation
 Fig. 12-17: Mode of operation of motion-influenced RCL statements

K-parameter (PLC/ parameter editor)

	Explanation	Calculation	Effect
K-0-0240	RoCoFeedrate override	$V_{eff} = V_{eff} * (K-0-0240 / 100)$	SA
K-0-0242	RoCoAcceleration override	$A_{eff} = A_{eff} * (K-0-0242 / 100)$	SA
K-0-0243	RoCoDeceleration override	$Deff = Deff * (K-0-0243 / 100)$	SA
K-0-0014	Path acceleration, modification time	$Jeff = A_{eff} / K-0-0014$ (Acceleration in triangle mode) $Jeff = A_{eff} / t$ for $t \leq K-0-0014$ (Acceleration in trapezoidal mode)	SA
K-0-0015	Path deceleration, modification time	$Jeff = Deff / K-0-0014$ (Acceleration in triangle mode) $Jeff = Deff / t$ for $t \leq K-0-0014$ (Acceleration in trapezoidal mode)	SA

SA Block execution, block processing
 Fig. 12-18: Mode of operation of motion-influenced parameters

Interpolation method: PTP (RCL syntax)

	Explanation	Calculation	Effect
V_PTP	decimal factor for the smaller value in A-0-0032/ A-0-0033 of the slowest axis participating in the motion	$V_{PTPeff} = (0.8 * A-0-0032/33) * V\_PTP$	SV
AFACTOR	decimal factor for A-0-0034, Acceleration limit value bipolar, of the slowest axis participating in the motion	$A_{PTPeff} = (0.8 * A-0-0034) * AFACTOR$	SV
R_PTP	decimal factor for A-0-0230, Blending distance PTP	$R_{PTPeff} = A-0-0230 * R\_PTP$	SV

SV Block preparation
 Fig. 12-19: Mode of operation of motion-influenced RCL statements



A-0-0032/ A-0-0033/ A-0-0034: The axis limit values can be modified in P2 and BB.

The modification of the values becomes only effective when switching from P2 to BB.

12.2.5 Additional Information

Additional information, General

Apart from informing the control about the position of the target point and the path, information on the period or the velocity in which the robot has to approach the target point has to be specified.

The control also requires specifications on the motion period or the path velocity of the robot.

The value entered for acceleration indicates how fast the robot is supposed to reach this path velocity.

Additional Information: Velocity

The travel velocity is programmed in mm/s for LINEAR and CIRCULAR interpolation. In case of a PTP interpolation, programming is carried out as factor of the maximum axis velocity.



If no velocity specifications are programmed, the robot travels at default velocity values. These values are 10 mm/s or 10 in/s (see K-0-0054) for path interpolation or 10 % of the maximum axis velocity for synchronous PTP interpolation.

The velocity V_{PTP} is specified as a decimal factor of the maximum axis velocity for the synchronous PTP interpolation.

It is also possible to enter values in per cent, i.e. $V_{PTP} = 80\%$ is equivalent to $V_{PTP} = 0.8$.

V

Designation	V
Measuring unit	mm/s or in/s (see K-0-0054)
Input range	1 to 99999
Switch-on state	25

Fig. 12-20: Velocity for LINEAR and CIRCULAR

V_{PTP}

Designation	V_{PTP}
Measuring unit	Decimal factor
Input range	0.0001 to 9.9999 0.01% to 999.99%
Switch-on state	0.1

Fig. 12-21: Velocities for PTP

Motion optimization for orientation axes

The interpolator calculates the accelerations and velocities of the orientation axes in advance that result when the path velocity is not exceeded.

If these limit values are violated, the resulting path velocity and path acceleration reduce accordingly. This also applies to linear interpolations.



The motion specification is always carried out with the limit values of all orientation axes maintained!

Programming options

The velocity can be programmed as a

RCL Motion Statements

- global velocity specification,
- statement-specific velocity specification,
- statement-specific time specification,

Global velocity specification

If the path velocity remains constant for the entire program or for a larger section, a global velocity is to be specified.

The velocity value remains valid until it is modified by another global velocity specification.

Example:

```

V=750                                ;V = decimal expression. All the following
                                     ;motion functions with the interpolation method LINEAR and
MOVE LINEAR pos1                     ;CIRCULAR are provided with the specified velocity value of V = 750 mm/s
MOVE CIRCULAR (pz, pe)

V=750                                ;The velocity when approaching linearly towards the points pos1
MOVE LINEAR pos1                     ;and pos2 is V = 750 mm/s
MOVE LINEAR pos2

V=300                                ;It is approached towards the points place3 und place4 as well
MOVE LINEAR place3                   ;towards the remaining ones
MOVE LINEAR place4                   ;with V= 300

```

Statement-related velocity specification (WITH)

If the path velocity is supposed to apply to just one motion statement, the velocity specification is statement-specific, which means that it is specified with the motion statement.

Velocity designation and value assignment are part of the motion function to which they apply.

Programming occurs following the interpolation method with the keyword WITH.

Example: LINEAR:

```

MOVE_REL LINEAR WITH V=500 TO pos

```

Example: PTP:

```

MOVE PTP WITH V_PTP=70% TO pos

```

The global velocity specifications do not influence the statement-specific specifications.

Override

Via the kinematic parameters

K-0-0240, RoCoFeedrate override,

K-0-0241, RoCoFeedrate override control word,

K-0-0242, RoCoAcceleration override and

K-0-0243, RoCoDeceleration override,

globally valid override parameters can be set for the kinematics. They directly affect the motion specification.

VFIX, VFIX_PTP, AFIX: Disabling override

The RCL commands VFIX, VFIX_PTP and AFIX can disable the following override parameters valid for the kinematics:

RCL Motion Statements

Designation	Disables parameter
VFIX, VFIX_PTP	K-0-0240, RoCoFeedrate override
AFIX	K-0-0242, RoCoAcceleration override

Fig. 12-22: VFIX, AFIX: Disable override

VFIX, AFIX and VFIX_PTP are only effective block by block. That means that these values can/may only be programmed together with the keyword WITH within a MOVE command. After this block, locking is automatically canceled. Influencing the velocity or acceleration is not effective/possible for this block via potentiometer.

Examples:

```
MOVE PTP WITH VFIX_PTP = 0.8 TO @pos1 ; potentiometer will have no effect
MOVE PTP WITH V_PTP = 0.8 TO @pos1 ; potentiometer will be considered
```

Additional Information: Velocity Factor

Velocity and time specifications can be modified using the programmed velocity factor "VFACTOR".

All velocity specifications are multiplied by the velocity factor.

Time specifications are divided by the "VFACTOR". The values determined apply to the subsequent motion functions.

The factor refers to the velocity and time specifications defined in the program.

The factor can also be specified in per cent, i.e. VFACTOR = 180 % is equivalent to VFACTOR = 1.8.

VFACTOR

Designation	VFACTOR
Measuring unit	1 or %
Input range	0.0001 (0.01%) to 6 (600%)
Switch-on state	1.0 (100%)

Fig. 12-23: Velocity factor



When starting an RCL program, AFACTOR, DFACTOR and VFACTOR are set to 1.0 (100 %)

Example:

```
VFACTOR=180%
;The VFACTOR of 180% (= Factor 1.8) increases the
;the path velocity to the positions 1, 2,
;3 and 4. It is moved to 180% of Vmax to pos1
;(initial state 10%, Factor 1.8)

MOVE PTP TO pos1

V=200
MOVE LINEAR TO pos2 ;It is moved towards pos 2 with a velocity of 360 mm/s
MOVE LINEAR WITH V=100 TO pos3 ;It is moved towards pos3 with a velocity of 180 mm/s
MOVE LINEAR WITH T=10 TO pos4 ;It is moved towards pos4 within 5.6 s
;(10/1.8= 5.6)
```

Effective velocity

The actual effective velocity consists of the programmed velocity (V), the velocity factor (VFACTOR) and the velocity override (K-0-0240).

Example:

RCL Motion Statements

K-0-0054 Bit 0 = 0 (mm)

Programmed V = 1000.00 mm/s, VFACTOR = 80% and K-0-0240 = 100 %.
An effective

$V_{\text{eff}} = V * VFACTOR * K-0-0240 = 800 \text{ mm/s}$ results

Additional Information: Acceleration

From the acceleration specifications, the control identifies how fast the robot has to reach its specified velocity.



Acceleration specifications can only be programmed for LINEAR and CIRCULAR interpolation.

In the PTP method, the robot accelerates based on the maximum values defined in the axis data set. The AFACTOR can be used to modify the acceleration.

Programming options for the acceleration specification

For LINEAR and CIRCULAR interpolation, the acceleration specification can be programmed as

- global specification
- statement-specific specification
- acceleration factor.

The acceleration factor is also effective for the PTP method.

Global acceleration specification
(Global specification)

Interpolation method: LINEAR and CIRCULAR

If the acceleration remains constant for the entire program or for a large section, a global acceleration is to be specified.

Designation and value assignment are part of an individual statement at the beginning of the program or program section.

The acceleration value remains valid until it is modified by another global specification.

Example:

```

O=30          ;All the following motion functions with the
              ;interpolation method LINEAR and CIRCULAR are provided with an
              ;acceleration value of A=30 mm/s2
V=100         ;When moving towards the positions pos1 to pos3, the acceleration till a
              ;path velocity of V=100 is reached
              ;is A=30 mm/s2
MOVE CIRCULAR (pz, pos1)
MOVE LINEAR pos2
MOVE LINEAR pos3

O=15          ;The positions pos4, pos6 and all the remaining ones with LINEAR and
              ;CIRCULAR interpolations are provided with a specified value of
              ;A=15 mm/s2
MOVE LINEAR pos4
MOVE PTP pos5
MOVE LINEAR pos6
  
```



pos5 is approached with the PTP interpolation. The acceleration specification A = 15 mm/s² does not apply. (K-0-0054 Bit 0 = 0 (mm))

RCL Motion Statements

Statement-related acceleration specification (WITH)

A

If a path acceleration is supposed to apply to just one motion command, the acceleration is specified locally.

Designation	A
Measuring unit	mm/s ² or in/s ² (see K-0-0054)
Switch-on state	10

Fig. 12-24: Acceleration for LINEAR and CIRCULAR interpolation



Local acceleration specifications can only be used for LINEAR and CIRCULAR interpolation.

In the PTP method, the robot accelerates based on the maximum values defined in the axis data set. The AFACTOR can be used to modify the acceleration.

Name and value assignment are part of the motion function for which the acceleration should apply.

Programming occurs after the interpolation method with the keyword WITH.

Example: LINEAR:

```
MOVE LINEAR WITH J=1000 TO pos
```

Example: CIRCULAR:

```
MOVE_REL CIRCULAR WITH J=2000 EXACT (dz,dis1)
```



The global acceleration specifications do not influence the statement-specific specifications.

It is possible to specify velocity and acceleration locally within a motion function. When programming, the two specifications are separated by a comma. There is no specified order.

Program:

```
MOVE_REL CIRCULAR WITH V=120, A=35, J=1000 EXACT (dz,dis1)
```

Additional Information: Acceleration Factor

The acceleration factor "AFACTOR" can influence the acceleration specifications again. The acceleration factor is multiplied by all acceleration specifications. The values calculated this way apply to all subsequent motion functions.



The AFACTOR is not effective for the statement-specific acceleration specification (in contrast to the DFACTOR).

Example:

K-0-0054 Bit 0 = 0 (mm)

Programmed A= 300.00 mm/s² and AFACTOR= 90%. Result

$A_{\text{eff}} = A * \text{AFACTOR} = 270 \text{ mm/s}^2$.

RCL Motion Statements

AFACTOR

Designation	AFACTOR
Measuring unit	%
Input range	0.0001 (0.01%) to 1.2 (120%)
Switch-on state	1.0 (100%)

Fig. 12-25: Acceleration factor



The AFACTOR is not effective if the acceleration is programmed statement-specifically.



After the program start, the AFACTOR, DFACTOR and VFACTOR are reset to 1.0 (100 %).

Example:

```

O = 1000 ;1000 [mm/s²]
V = 200 ;200 [mm/s]

AFACTOR = 50% ;The AFACTOR of 50 %
               ;(=Factor 0.5) increases the path acceleration

MOVE PTP TO pos1 ;The path acceleration has no influence on the PTP
                 ;interpolation
MOVE CIRCULAR TO (pz,pos2) ;Effective A = 500 mm/s² (0.5 * 1000)
MOVE LINEAR WITH O=100 TO pos3 ;Effective A = 100 mm/s², AFACTOR has no effect

```

Effective acceleration

The actual effective acceleration consists of the programmed acceleration (A), the acceleration factor (AFACTOR) and the acceleration override (K-0-0242).

Example:

K-0-0054 Bit 0 = 0 (mm)

Programmed A = 300.00 mm/s², AFACTOR= 90% and K-0-0242 = 100 %.
An effective

$A_{\text{eff}} = A * \text{AFACTOR} * \text{K-0-0242} = 270 \text{ mm/s}^2$.

Additional Information: Deceleration Factor

Analog to the acceleration factor, the deceleration factor can be set as impact on the deceleration via the command "DFACTOR".

The deceleration factor is multiplied with all deceleration specifications. The values calculated this way apply to all subsequent motion functions.

Example:

K-0-0054 Bit 0 = 0 (mm)

Programmed A= 300.00 mm/s² and DFACTOR= 90%. Result: an effective
 $D_{\text{eff}} = A * \text{DFACTOR} = 270 \text{ mm/s}^2$.

DFACTOR

Designation	DFACTOR
Measuring unit	%
Input range	0.0001 (0.01%) to 1.2 (120%)
Switch-on state	1.0 (100%)

Fig. 12-26: Deceleration factor

RCL Motion Statements

Effective deceleration The actual effective deceleration consists of the programmed acceleration (A), the deceleration factor (DFACTOR) and the deceleration override (K-0-0243).

Example:

K-0-0054 Bit 0 = 0 (mm)

Programmed A = 300.00 mm/s², DFACTOR= 90 % and K-0-0243 = 100 %.
An effective

$$D_{\text{eff}} = A * \text{DFACTOR} * K-0-0243 = 270 \text{ mm/s}^2$$



Acceleration and deceleration are based on the same **A** value.

The overlay occurs separately with the AFACTOR and the DFACTOR.

Example:

<pre>A = 1000 V = 200 DFACTOR = 50% MOVE PTP TO pos1 MOVE CIRCULAR TO (pz,pos2) MOVE LINEAR WITH O=100 TO pos3</pre>	<pre>;1000 [mm/s²] ;200 [mm/s] ;The DFACTOR of 50 % ; (=Factor 0.5) increases the path deceleration ;The path acceleration has no influence on the PTP ;interpolation ;Effective D = 500 mm/s² (0.5 * 1000) ;Effective D = 50 mm/s² (0.5 * 100)</pre>

Additional Information on the Jerk

From the jerk specifications, the control identifies how fast the robot has to reach its specified acceleration.

J

Designation	J
Measuring unit	mm/s ³ or in/s ³ (see K-0-0054)
Switch-on state	1000000

Fig. 12-27: Path jerk for LINEAR and CIRCULAR interpolation



Jerk specifications can only be programmed for LINEAR and CIRCULAR interpolation in the block transition modes BLOCK_SLOPE and CONT_SLOPE.

In the block transition mode PROGR_SLOPE, an acceleration change time can be set with the parameter K-0-0014 and a deceleration change time can be set with the parameter K-0-0015.

Programming options for the jerk specification

The jerk specification can result as

- global specification
- statement-specific specification

Global jerk specification (Global specification)

If the jerk remains constant for the entire program or for a large section, a global jerk is to be specified.

Designation and value assignment are part of an individual statement at the beginning of the program or program section.

RCL Motion Statements

The jerk value remains valid until it is modified by another global specification.

Example:

```

J=1000           ;All the following motion functions with the
                  ;interpolation method LINEAR and CIRCULAR are provided with a
O=30             ;jerk of J=1000 mm/s³.
                  ;The jerk until reaching the path acceleration of A=30 mm/s² is
MOVE CIRCULAR (pz, pos1) ;When approaching the points pos1 to pos3 J=1000 mm/s³
MOVE LINEAR pos2
MOVE LINEAR pos3

J=500            ;the positions pos4, pos5 and all remaining motion functions
                  ;with LINEAR and CIRCULAR interpolation have a value specification of
MOVE LINEAR pos4   ;J=500 mm/s³
MOVE LINEAR pos5
MOVE PTP pos6      ;Pos6 is approached with PTP. Thus, the jerk specification is not effective.

PROGR_SLOPE       ;The block transition mode PROGR_SLOPE becomes ineffective for all following
MOVE LINEAR pos7   ;motion functions.
                  ;Pos7 is approached with the block transition mode PROGR_SLOPE.
                  ;Thus, the jerk specification is not effective.

```

Statement-related jerk specification (WITH)

If a path jerk is supposed to apply to just one motion command, the jerk is specified locally.

Name and value assignment are part of the motion function for which the jerk should apply.

Programming occurs after the interpolation method with the keyword WITH.

Example: LINEAR:

```
MOVE LINEAR WITH J=1000 TO pos
```

Example: CIRCULAR:

```
MOVE_REL CIRCULAR WITH J=2000 EXACT (dz,dis1)
```



The global jerk specifications do not influence the statement-specific specifications.

It is possible to specify velocity, acceleration and jerk locally within a motion function. When programming, the specifications are separated by a comma. There is no specified order.

Program:

```
MOVE_REL CIRCULAR WITH V=120, A=35, J=1000 EXACT (dz,dis1)
```

12.3 Motion-influencing Statements

12.3.1 General Information

In addition to the direct motion statements described above, the following statements also influence the motion sequences:

- Synchronization statements SYNC, SYNCHRON, SYNCHRON_END
- Acceleration and deceleration between motion statements (block transitions) BLOCK_SLOPE, PROGR_SLOPE and CONT_SLOPE

12.3.2 Belt Synchronization

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

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

The presettings are selected using the special functions [SPC_FCT: 21, belt type, page 108](#), and [SPC_FCT: 53, belt-synchronous workspace with belt type 4, page 110](#).

The belt must be a straight line. The line can be arbitrarily positioned in space; see MLC parameter "K-0-0070" and the following. The belt motion is registered by a distance measuring system or commanded by the MLC with "MC_MoveVelocity" for example.

The statements SYNC, SYNCHRON and SYNCHRON_END are available in the RCL for functionality programming.

Syntax:

```
SYNC belt variable[, variable] condition
SYNCHRON [kinematic name] belt variable
SYNCHRON_END [kinematic name] belt variable
```

The belt variable has to be declared in the program's declaration; see also [Communication Channels to PLC, General, page 70](#) and has to be assigned to a kinematics.

Example:

Declaration of belt variables of several belts to one kinematics

```
Kin1.belt:    501=belt1, 502=belt2
screwdr.belt: 503=screw_belt ;same belt like belt1
```



Altogether, up to 4 belts can be declared per kinematics. The belt names and belt numbers must be different, even if several kinematics use the same measuring system for one belt variable.

Synchronized motion can only occur on belts that have also been declared for this kinematics.

The component names and axis names must be specified accordingly for the assigned belt.



The belt synchronization with the block transition modes PROGR_SLOPE and CONT_SLOPE is currently not supported.

Programming belt synchronization

The belt variable of data type "REAL" contains the counter reading of the belt measuring system.

The belt variable can only be queried with comparative operations $\geq$ and $\leq$ or used in the program with "WAIT UNTIL", "IF" or "SYNC".

Example:

```
WAIT UNTIL belt_1>=6 ;belt variable in WAIT UNTIL
```



The program waits at the SYNC command till the belt variable has met the criterion.

RCL Motion Statements

The "SYNC" command sets the belt variable to zero or to the reset value (SPC_FCT 28).

Zeroing can take place depending on a condition.¹⁾

The following example shows possibilities for using the SYNC command.

Example:

```

SYNC belt_1>=300 ;subtracts 300 from the belt counter if the current
                  ;counter reading is ≥ 300
SYNC belt_1, li_scr=1 ;resetting to zero is executed depending on a condition

```

The following example shows how to execute the non-waiting modulo processing.

Example:

```

IF belt_1>=300      ;initially checks if
                   ;counter reading is ≥ 300
THEN
BEGIN
  SYNC belt_1>=300 ;belt counter is reset in any case, as condition is correct.
END

```

The "SYNCHRON" command enables the belt synchronization. From now onwards, all programmed motions are synchronized with the belt in terms of position and orientation. The synchronization takes place as soon as the programmed target point (moved with the counter) is located within the belt-synchronous workspace.

"SYNCHRON_END" is used to switch off the synchronization.

Example:

```

SYNCHRON Kin1 belt_1 ;belt variable and SYNCHRON statement
MOVE LINEAR TO pos_1 ;(the pos_1 belt counter has a valid value > 0)
SYNCHRON_END belt_1

```

Example:

```

;;COMPILER= RCL3
;;KINEMATICS: (1=robi_1)      ;definition of kinematic names
;;robi_1.JC_NAMES=a1,a2,a3,b1 ;b1 is a dummy for belt variables
;;robi_1.WC_NAMES=k1,k2,k3,b_c1

PROGRAM beltsyn
VAR:
  CONST :
    C_BELT_NO_1 = 1,      ; belt number
    C_BK_PTP = 4          ; belt type PTP
BELT :
  500 + C_BELT_NO_1 = belt1 ; channel / belt name

; declaration of special functions
SPC_FCT : 21 = belt_kind (VALUE INTEGER: belt_no
                          VALUE INTEGER: kind_belt)
SPC_FCT : 53 = belt_area (VALUE INTEGER: BeltNo
                          JC_POINT: @Base
                          REAL: rTotalLength
                          REAL: rBeginLength)

INPUT: 1=i1,2=i2

BEGIN ; belt type specification
belt_kind ( C_BELT_NO_1, C_BK_PTP ) ; specifying belt-synchronous operating range
rTotalLength = 560
rBeginLength = 400
Base = (280, 140, -600, 0, 0, 0)
@Base = JC(Base)
belt_area ( C_BELT_NO_1, @Base, rTotalLength, rBeginLength)

  SYNC belt1,i1=1 ;waits until the PLC has set i1=1 to reset the
                  ;belt counter belt1 (501)
SYNCHRON robi_1 belt1 ;synchronization of kinematics robi_1 to belt1

```

¹⁾ Subtracts a value (similar to modulo).

```
MOVE rob1_1 LINEAR TO rob1_1.POS
WAIT UNTIL belt1>=1000
SYNCHRON_END rob1_1 belt1
```

PROGRAM_END

12.3.3 Block Transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)

Syntax:

BLOCK_SLOPE

PROGR_SLOPE

CONT_SLOPE

In the normal motion sequence, the triggered axes are accelerated to the programmed velocity with each MOVE statement. They travel at the programmed velocity and are then decelerated again to $V = 0$ to reach the programmed position.

The velocity change is referred to as "SLOPE". Accordingly, the acceleration phase is known as up-slope and deceleration as down-slope.

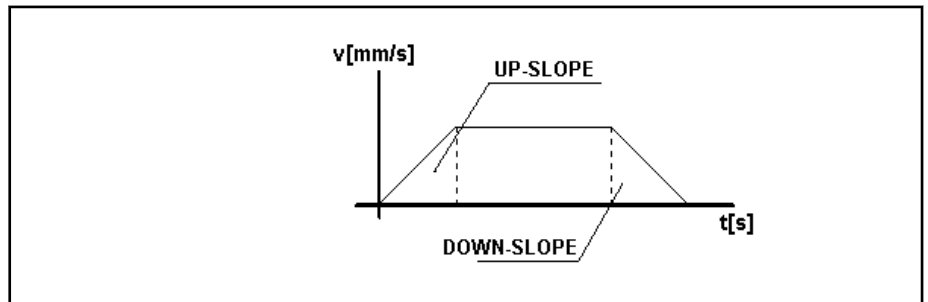


Fig. 12-28: Up-slope and down-slope

General Information

The block transition types PROGR_SLOPE and CONT_SLOPE are used to travel the blending motion commands altogether without intermediate stops. The block transitions are executed at a constant command velocity. If the blocks are programmed with different velocities, a ramp-shaped velocity change results between the motion commands.

The block transition mode BLOCK_SLOPE is used for a block by block acceleration and deceleration.

Enabling SLOPE mode

The initial setting for the slope mode is BLOCK_SLOPE.

This slope mode can be switched in the RCL program. The RCL standard functions BLOCK_SLOPE, PROGR_SLOPE and CONT_SLOPE are defined for this purpose.

When enabling a SLOPE type, the previous one is switched off.



For each slope mode change, the transition velocity is always 0 !

BLOCK_SLOPE

The operation mode BLOCK_SLOPE handles all velocity specifications and acceleration specifications in a block-specific way. Acceleration and deceleration take place at the beginning and at the end of a motion command. Thus, there is no transition velocity between the two blocks. Via the blending radius, a velocity blending is achieved when the subsequent block was started too early.

RCL Motion Statements

PROGR_SLOPE	The operation mode PROGR_SLOPE is a block-comprehensive velocity and acceleration specification. The robot accelerates at the start of a coherent motion sequence and decelerates at its end. At block transitions, the velocity is kept constantly if no velocity change has been programmed. Spatial blending via circular segments is executed via the blending radius.
CONT_SLOPE	The operation mode CONT_SLOPE is a block-comprehensive velocity and acceleration specification. The robot accelerates at the start of a coherent motion sequence and decelerates at its end. At block transitions, the velocity is kept constantly if no velocity change has been programmed. Spatial blending via spline segments is executed via the blending radius.



For more information on the spatial blending via blending radius, see below: [chapter 12.3.6 "Spatial Blending" on page 142](#)

Programming the SLOPE mode is explained below using a few RCL program examples. Its effect is shown in the figures below.

Example 1:

PROGRAM ex_1	; (1)
BEGIN	; (2)
;;INT=LINEAR	; (3)
V=800,A=1000	; (4)
MOVE TO beg_point	; (5) Programmslope nicht aktiv, d.h. der Roboter wird in jedem Satz mit A=1000 mm/s ² beschleunigt und am Ende des Satzes wieder abgebremst.
	; (6)
MOVE TO point_center	; (7)
MOVE TO end_point	; (8) PROGR_SLOPE wird aktiviert
PROGR_SLOPE	; (9)
MOVE VIA beg_point	; (10)
MOVE VIA point_center	; (11)
MOVE TO end_point	; (12)
MOVE VIA RAMP_1	; (13)
MOVE VIA RAMP_2	; (14)
MOVE VIA beg_point	; (15)
MOVE VIA point_center	; (16)
MOVE TO end_point	; (17)
BLOCK_SLOPE	; (18)
HALT	; (19)
PROGRAM_END	

In the example, bit 0 = 0 (mm) is configured in the parameter K-0-0054.

In the first part of the program, blocks 5 to 7, the default slope mode BLOCK_SLOPE is active. The robot accelerates in each block at $A = 1000 \text{ mm/s}^2$ and decelerates at the end of the block, see slope, Fig. 1.

In block 8, PROGR_SLOPE is switched on. As a result, the robot accelerates in block 9 at $A = 1000 \text{ mm/s}^2$.

The block transitions from 9 to 10 and 10 to 11 are executed at constant velocity.

At the end of block 11, the robot decelerates with the programmed deceleration of 1000 mm/s^2 ; see slope, Fig. 2.

If the travel distance in the MOVE TO block (block 11) is not sufficient to decelerate the robot with the slope function, the deceleration has been initiated in the previous block (block 10, slope, Fig. 3).

The acceleration phase can run across any number of blocks, e.g. block 12 to block 13, slope, Fig. 3. This also applies to the deceleration phase, e.g. block 15 to block 16, Fig. 3.

RCL Motion Statements

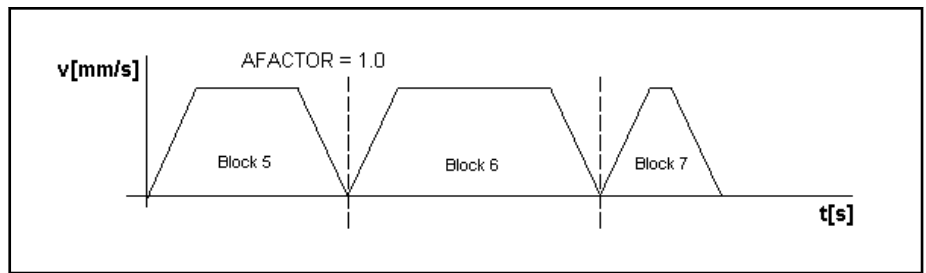


Fig. 12-29: Slope, Fig. 1 - BLOCK_SLOPE

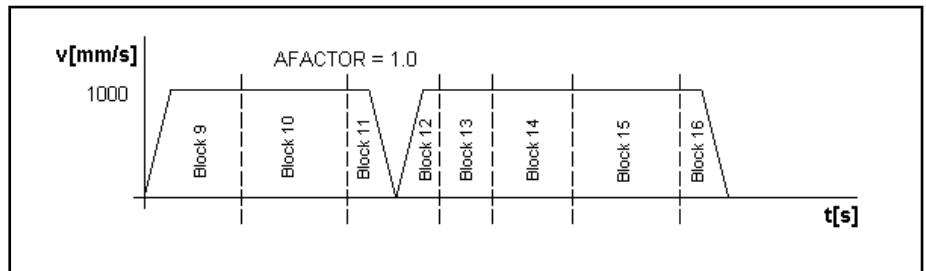


Fig. 12-30: Slope, Fig. 2 - PROGR_SLOPE

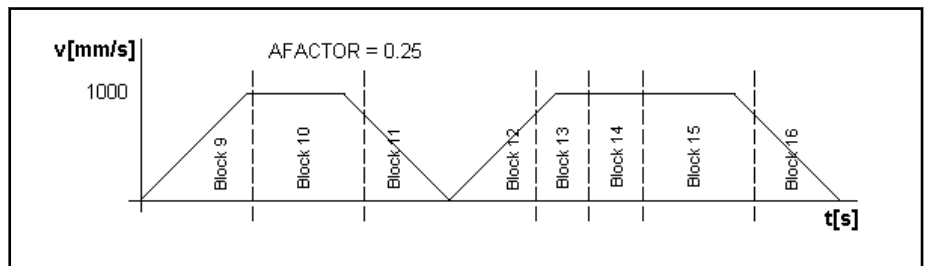


Fig. 12-31: Slope, Fig. 3 - PROGR_SLOPE

Motion abort

Abort of a coherent motion in the RCL program

A motion sequence with the slope modes PROGR_SLOPE and CONT_SLOPE is interrupted by the following RCL commands:

- Synchronous input prompt
- WAIT
- PAUSE
- HALT
- BLOCK_SLOPE
- Several blocks without travel information, e.g. calculations, variable assignments, setting of output signals

The number of possible blocks depends on the length of the preceding travel blocks and the type of the assignments or calculations performed.

Example 2:

```
PROGRAM ex_3
;;INT=LINEAR
BEGIN
V=800, O=1000
PROGR_SLOPE

MOVE LINEAR VIA beg_point

WAIT 1
MOVE LINEAR VIA point_center
```

```
;Interpolation method LINEAR applies to the following motion commands

;PROGR_SLOPE is switched on

;End velocity = 0, since the next command is a synchronous
;command

;It is traveled in a velocity-blending manner.
;The end velocity is calculated.
```

RCL Motion Statements

```
MOVE TO end_point
PROGRAM_END
```

Changing the interpolation method

The global acceleration and deceleration behavior can be enabled irrespective of the interpolation method. Block transitions without any change in the interpolation method are executed as described under [Block Transitions \(BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE\)](#), page 137..

Switching between linear and circular interpolation

When switching from linear to circular interpolation and vice versa, the block transitions are executed as described in [Block Transitions \(BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE\)](#), page 137.

Example 3:

```
PROGRAM ex_4
;;INT=LINEAR
BEGIN
V=800,A=1000
PROGR_SLOPE
MOVE VIA beg_point
MOVE CIRCULAR VIA (int_point1,circular_end1)
MOVE CIRCULAR TO (int_point2,circular_end2)
MOVE TO pnt_center
MOVE CIRCULAR TO (int_point3, circular_end3)
HALT
PROGRAM_END
```

Switching between LINEAR and PTP interpolation

Before switching from LINEAR to PTP interpolation and vice versa, the motion sequence has to be completed by a "MOVE TO" block so that a controlled transition can be executed.

If the switching occurs during a coherent motion, the velocity is modified in jumps.

Example 4:

```
PROGRAM ex_5
BEGIN
PROGR_SLOPE
V_PTP=1
MOVE PTP TO beg_point
MOVE LINEAR TO p1
MOVE PTP VIA p2
MOVE LINEAR VIA end_pnt
V_PTP=0.5
MOVE PTP TO end_pnt_1
HALT
PROGRAM_END
```

Exceeding axis limit values

Due to the coordinate transformation, it cannot be excluded that the limit values of individual machine axes are exceeded during the path operation. While the program is running, it can only be monitored.

If an error occurs, the monitoring function triggers one of the following error messages:

- Interpolator stop, axis X
- Axis velocity too high, axis X

X is the number of the respective machine axis.

In this case, the software engineer has to modify the program at the respective positions. Automatic velocity adjustment is not possible, as it would contradict the requirement for keeping the path velocity as constant as possible.

12.3.4 Look-Ahead Across Several Blocks at PROGR_SLOPE and CONT_SLOPE

In the slope modes PROGR_SLOPE and CONT_SLOPE, the calculation of possible block transition velocities is calculated before across several blocks. It is considered, that the robot is provided with sufficient distance to stop at the end of the travel motion.

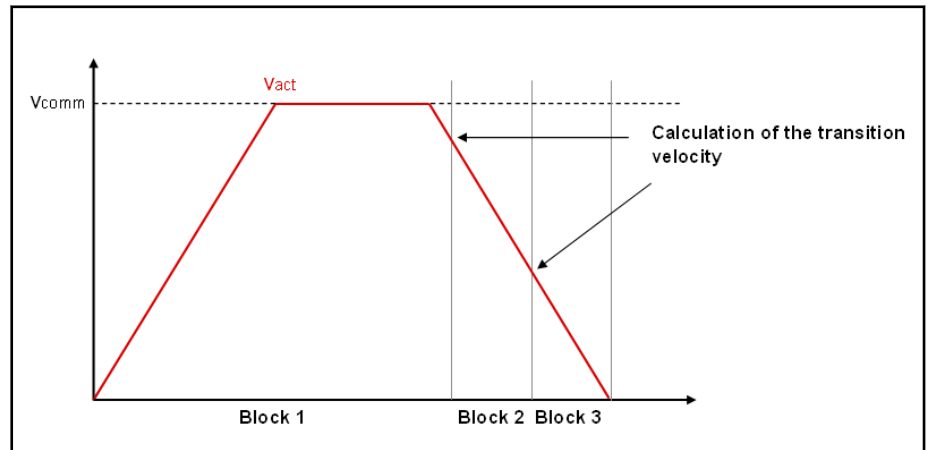


Fig. 12-32: Look-ahead across several blocks

The figure above shows the operating principle of an integrated look-ahead. The programmed command velocity is not reached in block 2 and block 3. The end velocity of the first block is adapted to the deceleration distance.

12.3.5 Late Commanding

The MLC RobotControl provides the option to transmit motion commands late. This is possible for PROGR_SLOPE and CONT_SLOPE as well as for BLOCK_SLOPE. If another block is commanded during block traveling, the end velocity of the current block might also be influenced.

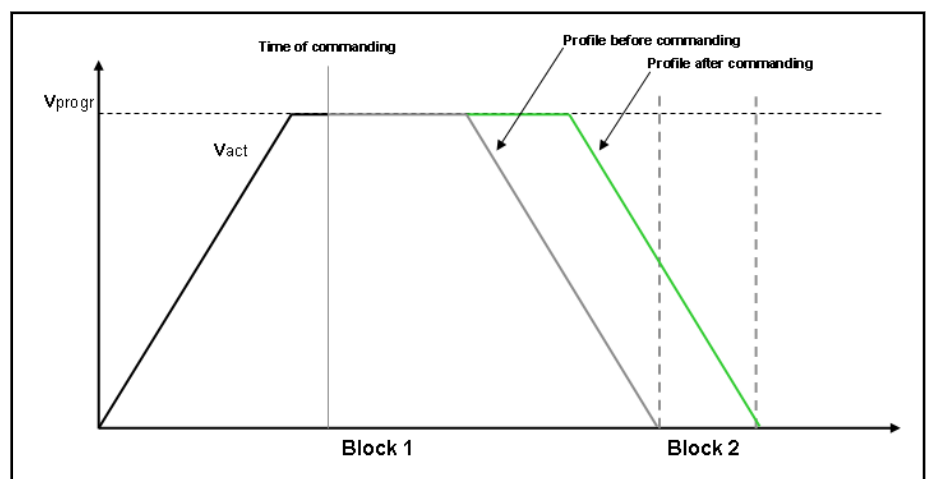


Fig. 12-33: Increasing end velocity by late commanding

The figure describes such a procedure. Block 1 starts with the acceleration to the programmed velocity. The block is interpolated so that the kinematics is in standstill after the programmed block. At the time "Time of commanding", block 2 is commanded while block 1 is active. This increases the possible

RCL Motion Statements

end velocity of the first block, even though the block has already been started.

Adapting the end velocity includes several blocks beforehand. This mechanism is completely running in the background.

12.3.6 Spatial Blending

Overview

To carefully treat the mechanics of the robot and to decrease the time needed for the motion sequence at the same time, the programmed target points can be blended. The consequence is a continuous and harmonic motion at block transitions.

Definition of "blending": Ensuring uninterrupted velocity characteristics for block transitions using a defined deviation from the programmed path.

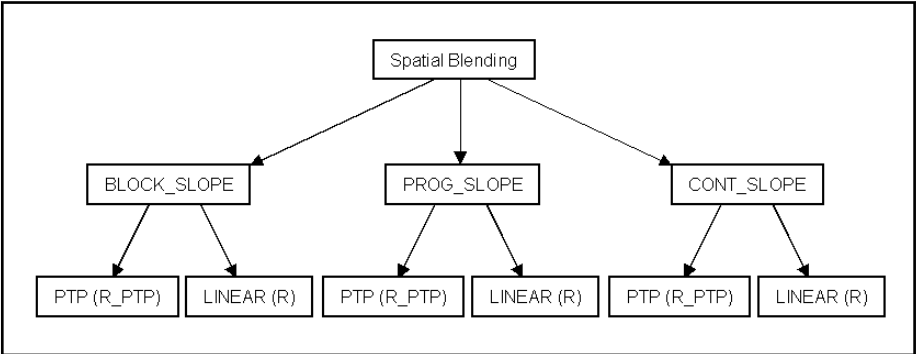


Fig. 12-34: Spatial blending

Blending radius Depending on the interpolation method of the default variables "R" or "R_PTP", spatial blending is influenced in the RCL program.

The default variable "R_PTP" is programmed for the PTP interpolation. This factor can be programmed in decimals or in per cent. It is calculated with parameter A-0-0230, Blending distance PTP, of all axes.

The default variable "R" like in radius is used for the LINEAR interpolation. Absolute values are programmed for "R". The robot type selected defines the unit for "R".

R, R_PTP	Designation	R	R_PTP
	Measuring unit	mm or in or ° (see K-0-0054)	-
	Switch-on state	0	0
	Effect	absolute	A-0-0230, Blending distance PTP

Fig. 12-35: Blending radius

Example MOVE statement with R or R_PTP

```
;;INT=LINEAR ;interpolation method LINEAR applies for the following motion statements
R = 50 ;R = 50 mm now applies to all blocks

MOVE LINEAR WITH R=20 VIA p1;R = 20 mm, as block-related
MOVE LINEAR TO p2 ;R = 50 mm

;;INT=PTP ;interpolation method presetting
R_PTP = 1.4 ;R_PTP = 140% of A-0-0230 of the participating axes
;now applies globally

MOVE PTP WITH R_PTP=1.2 VIA p3 ;R = 120%, since defined block-related
MOVE PTP TO p4 ;R = 140%
```




If $R = 0$ or $R\_PTP = 0$ are programmed, spatial blending is switched off.

In general, the following applies to BLOCK_SLOPE

- The spatial deviation in the blending range **depends on velocity**.
- The individual motion blocks are started earlier in terms of the preset blending distances. The path velocity results from the geometric addition of single axis velocities.

In general, the following applies to the PROGR_SLOPE and CONT_SLOPE

- The spatial deviation in the blending range **does not depend on the velocity**.
- In the blending range, the path is specified by a block generated by the control. In a related motion sequence, attempts are made to keep the velocity constant.



Points approached in the circular interpolation or from which approaching starts with the circular interpolation, are not spatially blended.

This applies to BLOCK_SLOPE as well as to PROGR_SLOPE and CONT_SLOPE.

Spatial Blending for PROGR_SLOPE and CONT_SLOPE

PTP Blending for PROGR_SLOPE and CONT_SLOPE

A blending distance in degrees or mm can be saved for each axis for its end position in parameter A-0-0230, Blending distance PTP. A blending range around the target point is defined based on these distances. These distances can be changed with a factor (R_PTP) in the RCL program by the user. All of the stored distances for the individual axes are equally influenced so that blending ranges are either compressed or extended.

A unique blending range can thus be defined around each target position. This range can be optimally adjusted based on the respective requirements.

If the tool working point enters this blending range during the travel motion, the robot travels on a path generated by the control ensuring that the individual axis velocities do not change in jumps. After leaving the blending range, the robot acts as in case of travel motions without programmed blending.

Each motion block contains information on velocity and distance to be traveled. If a different velocity is programmed in successive blocks, the velocity change is carried out at the block transition. If the defined blending range is relatively small and the difference of the programmed block velocity is relatively high, the difference velocity of the successive motions has to be balanced in a short period. The deviation from the originally programmed path, that is without blending, is limited to a small range. If large blending ranges are selected, the velocity transition is smoother, but the deviation occurs over a longer distance from the path.

The transitions between the programmed path and the blending path are tangential.

The following example tries to clarify the continuous velocity change in one block transition. It is based on a Cartesian system with two axes, one in x-direction and one in y-direction. The slope mode is PROGR_SLOPE. The blending range in the x- and y-direction is determined using the parameter

RCL Motion Statements

A-0-0230 of the respective axis. If it is traveled without blending, there is a velocity jump from +V to -V at the block transition for the y-axis.

If blending is enabled, the velocity is changed continuously from +V to -V within the blending range. Smoother block transitions are generated.

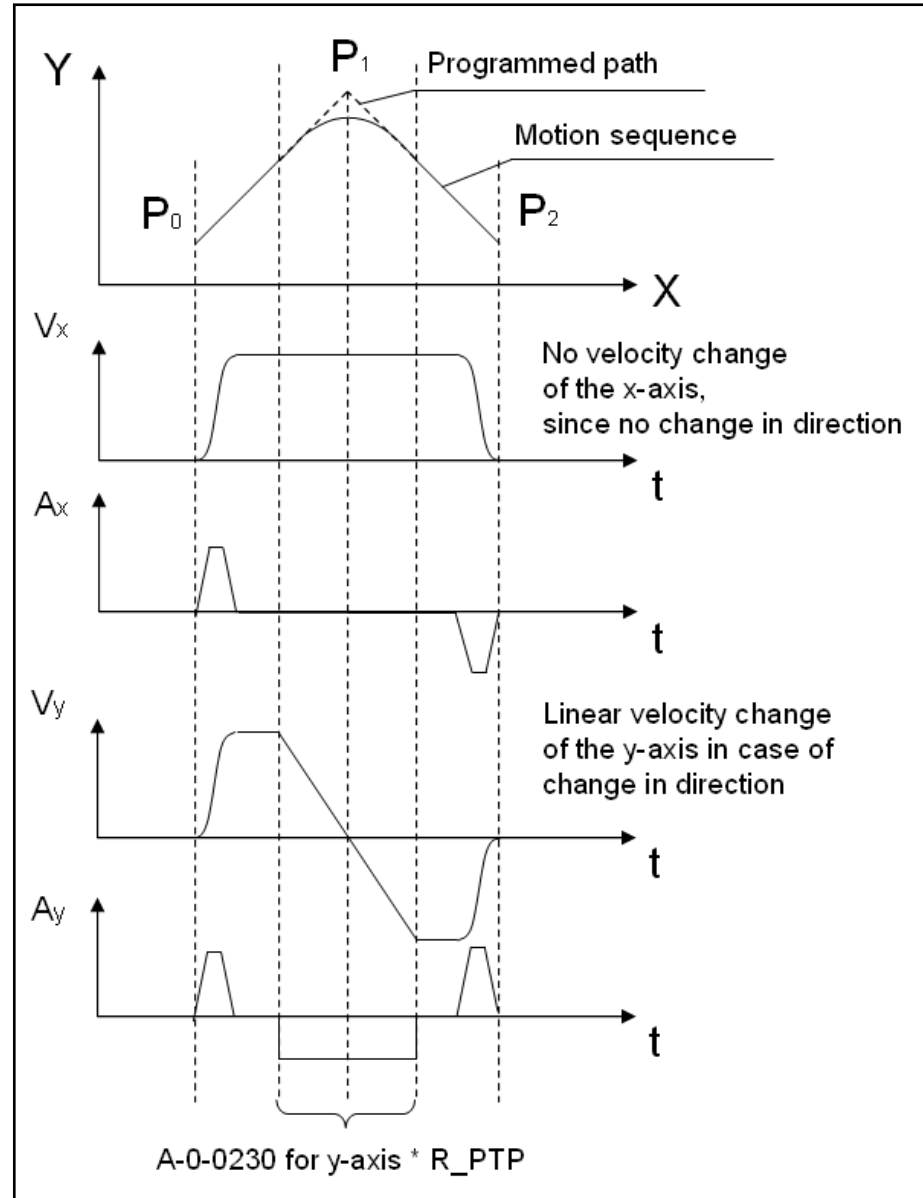


Fig. 12-36: Velocity characteristics of the x-axis and y-axis during PTP blending in PROGR_SLOPE

Linear Blending for PROGR_SLOPE and CONT_SLOPE

In path blending, a spatial sphere is defined around each point to be blended. The radius of this sphere is defined in the RCL program with the default variable R. Thus, for path blending, each programmed spatial point can be provided with its individual blending range. During entry and exit from this spatial sphere, the original path is exited for a constant path velocity. The motion within the sphere is carried out on a path generated by the control. Outside this sphere, the robot moves on the programmed path.

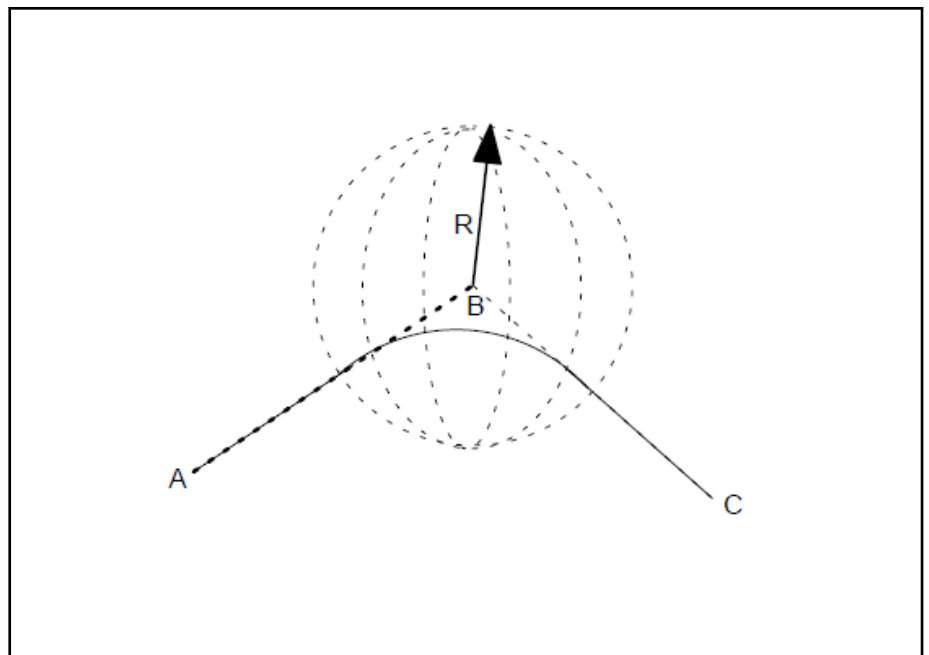


Fig. 12-37: Blending sphere for linear interpolation in *PROGR_SLOPE*

Blending radius: *PROG_SLOPE*

If the distances between the points to be blended do not meet the programmed blending criterion, the control automatically reduces to the maximum possible blending range. This is shown in

[fig. 12-38 "Overlapping blending range " on page 146](#)

RCL Motion Statements

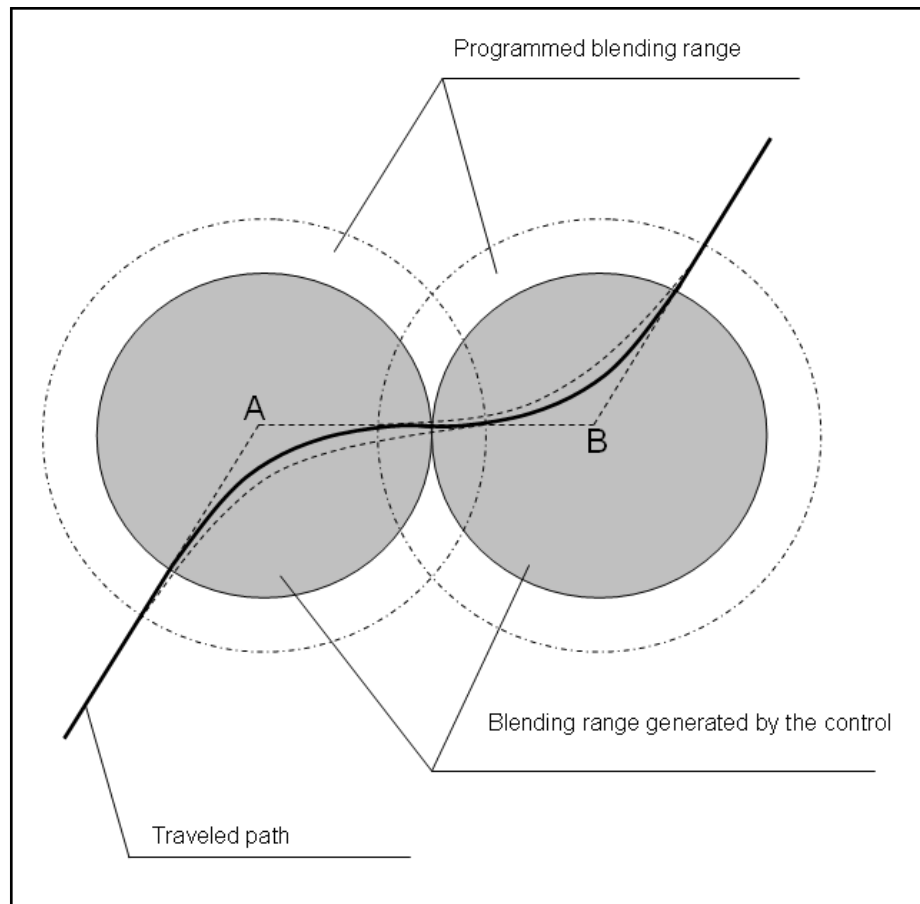


Fig.12-38: Overlapping blending range

Explanation of the figure:

Overlapping blending ranges are automatically reduced by the control in such a way that the maximum possible blending range results for this case. It is shown by the continuous line. For more details on the effective radius while blending, please refer to [Blending mode, page 151](#).

Blending radius: CONT_SLOPE

The maximum radius possible for blending is half the distance between two points. If the distances between the points to be blended do not meet the programmed blending criterion, the control automatically reduces to the maximum possible blending range. In contrast to the slope type PROG_SLOPE, asymmetric blending radii are possible. The following applies: The maximum blending radius between several blocks to be blended is the respective half of the distance between two points. For start and end blocks of a motion, the complete distance can be used for blending.

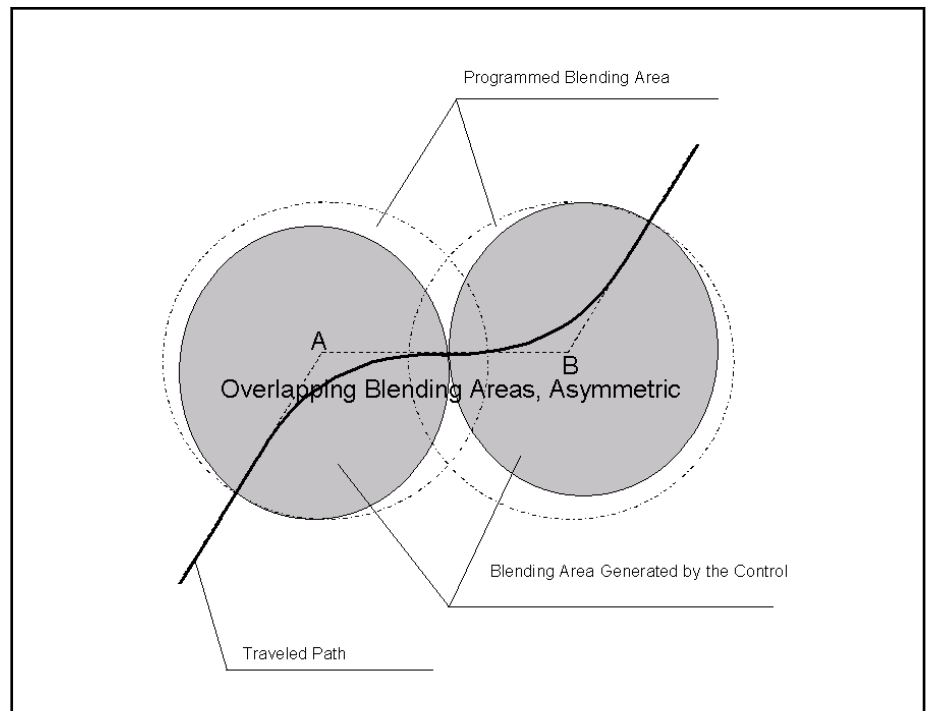


Fig. 12-39: Overlapping blending range

Explanation of the figure:

Overlapping blending ranges are automatically reduced by the control in such a way that the maximum possible blending range results for this case. It is possible for CONT_SLOPE that asymmetric blending areas result. The continuous line represents the actual path. For more details on the effective radius while blending, please refer to [Blending mode, page 151](#).



The belt synchronization with the block transition modes PROGR_SLOPE and CONT_SLOPE is currently not supported.

Blending when Changing the Interpolation Method

Blending in case of PROGR_SLOPE active, is also effective when changing the interpolation methods. PTP can be used to move to a point and LINEAR can be used to move away from a point for example. This is also possible vice versa. Before entering and after leaving the blending envelope, the robot does not deviate from the originally programmed path. In this case, it is irrelevant whether it is changed from PTP to LINEAR or vice versa.



For the operation mode CONT_SLOPE, blending is only supported for the motion combinations Linear - Linear and PTP - PTP. When changing the interpolation method, no blending takes place.

Spatial Blending for BLOCK_SLOPE

PTP Blending for BLOCK_SLOPE

For the slope type BLOCK_SLOPE, the next motion block is started too early depending on A-0-0230, blending distance and R_PTP factor programmed in the RCL program. Distances is specified in degrees or mm for each axis with regard to its end position. From these distances, a common point in time is determined to start the following motion block. Both block are now parallelly

RCL Motion Statements

executed. The path velocity results from the geometric addition of single axis velocities. This ensures that after the robot's tool working point exited the blending range, it reaches again the path originally taken without blending. To prevent the simultaneous activation of more than two motion blocks, the control automatically limits the earliest starting point to the center of the previous motion. A started motion block is ended until the center of the subsequent block. Depending on the effective blending mode, the behavior described can deviate with regard to the mentioned distances. For more details on the effective radius while blending, please refer to [Blending mode, page 151](#).

The following example tries to clarify the travel behavior.

It is based on a Cartesian system with 3 positioning axes, in which the z-axis will first travel upwards from the workspace. Subsequently, the x-axis and the y-axis travel to emerge again with the z-axis in the workspace.

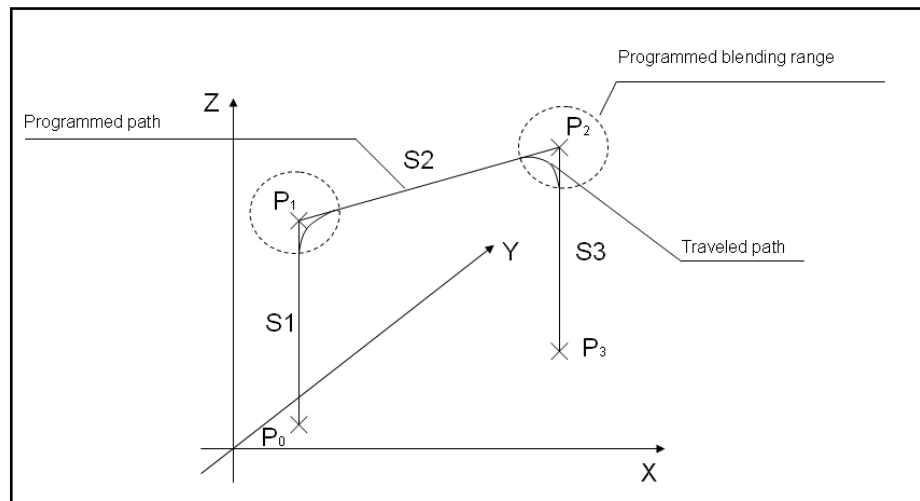


Fig.12-40: Path characteristics of Pick&Place application

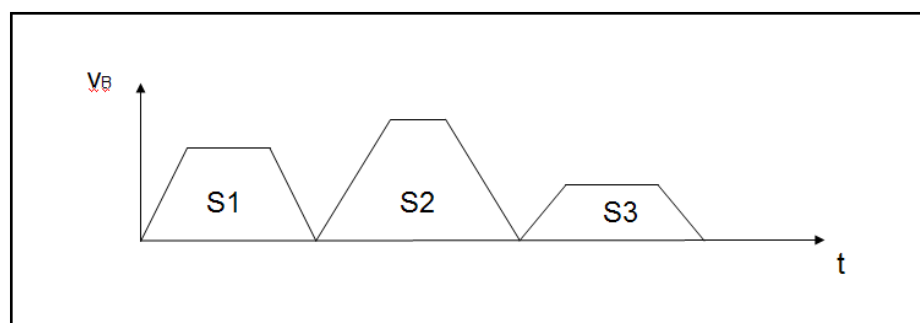


Fig.12-41: Path velocity characteristics of Pick&Place application without blending

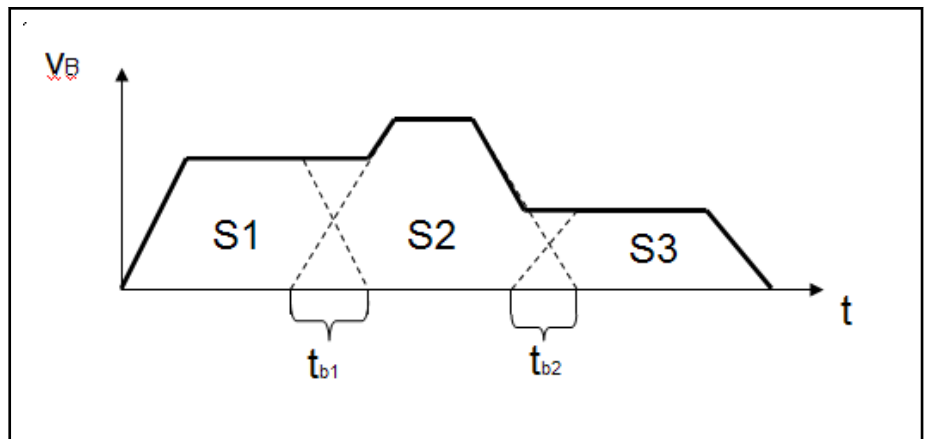


Fig. 12-42: Path velocity characteristics of Pick&Place application with blending

Time saved: $t_{b1} + t_{b2}$.

Linear Blending for BLOCK_SLOPE

Path blending with BLOCK_SLOPE has, with regard to the path, principally the same effect as PTP blending with BLOCK_SLOPE.

Blending when Changing the Interpolation Method

In case of an active BLOCK_SLOPE, it is blended from LINEAR to PTP and vice versa when changing the interpolation method.

Late Blending

The mechanism of the "Late blending" refers to the spatial blending in BLOCK_SLOPE, CONT_SLOPE as well as in PROGR_SLOPE.

In practice, the target point of the travel motion is often posted at a later point in time. To optimize the motion sequence from a temporal point of view, the robot can already be commanded to an intermediate point before this point in time. If the target point becomes known on the way to the intermediate point, late blending starts. If necessary, the programmed radius is approximately reduced to the distance between the current position and the intermediate point.

RCL Motion Statements

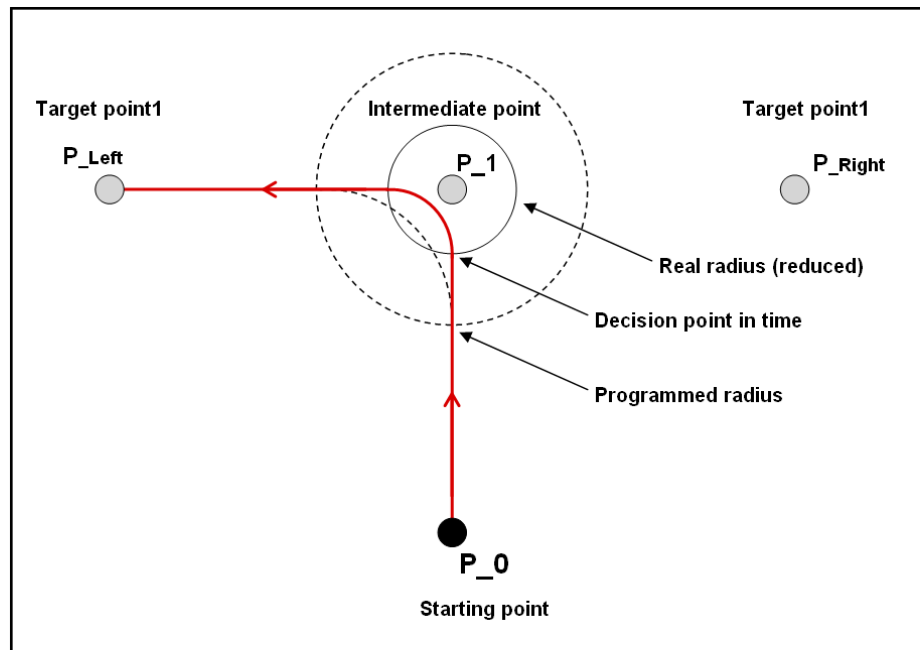


Fig.12-43: Late blending for PROGR_SLOPE

The example above shows a motion sequence that will be completely defined at a later point in time using the "Late blending". It is first traveled to the starting point P_0. The motion is started when traveling to the intermediate point P_1. While traveling, it is decided to approach the point "P_Left" at a later point in time. Since the motion is already in a progressed state and the initially programmed radius range has already been exceeded, the radius is reduced automatically.

Program:

```

PROGRAM LBlend
  INPUT BINARY : 1001 = bAsyncGo
  INPUT BINARY : 1002 = bAsyncIn
  POINT : P_0, P_1, P_Left, P_Right
BEGIN
  ;;INT=LINEAR

  P_0 = (0,0)
  P_1 = (0,100)
  P_Left = (-100,100)
  P_Right = (100,100)

  PROGR_SLOPE

  A = 10000
  V = 2000
  R = 20

  MOVE TO P_0
  MOVE VIA P_1

  ; Decide whether to turn left or right
  ; By the time when this decision is done, switch on the bAsyncGo input.
  ; The program shall wait here for this plc signal.
  WAIT UNTIL bAsyncGo

  ; The decision was made. Check the result.
  ; 1: Left, 0: Right
  IF bAsyncIn = 1
  THEN
    MOVE TO P_Left
  ELSE
    MOVE TO P_Right

```


PROGRAM_END

Blending Mode



This blending behavior applies to BLOCK_SLOPE and to PROGR_SLOPE. Blending with point-to-point movements (PTP) and blending with linear interpolation (LINEAR) are equally implemented. At CONT_SLOPE, this behavior only applies for the blending at linear interpolation (LINEAR).

For many applications, it is recommended not to limit the blending radius of the first and last motion by half the distance of the shortest distance (upper blue curve).

Optionally, the entire distance of the shortest distance can be used as blending radius from version MLC-FW 10V06 onwards if this is geometrically possible (lower curve shown in green).

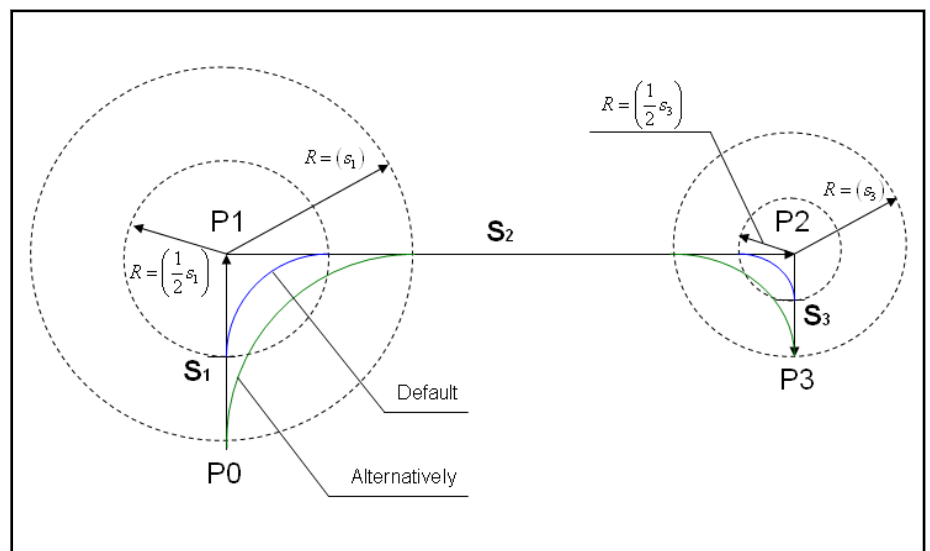


Fig.12-44: Adapting the effective radius via blending mode



An overlapping of the blending radii is not allowed in both the cases and is impeded by the control. In this case, the effective radius

$$R_w = \min \left(R, \frac{1}{2} s_1, \frac{1}{2} s_2 \right)$$

us is limited to



The term "**half distance**" describes a little bit less than half the distance (1/2,3) due to technical reasons. This is also the case for the "**complete distance**" that is also a bit less (1/1,1).

Programming this function can take place in the PLC as well as in the RCL program. However, it is strongly recommended not to use both access options at the same time.

See also [RobotControl Functionality Programmed in the PLC](#), page 253.

PLC Programming "Blending Mode" see *ML_Robot, Library Description*:

- PLC function ML_KinGetDistanceBlendOption,

RCL Motion Statements

- PLC function ML_KinSetDistanceBlendOption.

RCL programming "Blending mode"

- [SPC_FCT: 59, Query blending mode, page 114,](#)
- [SPC_FCT: 60, Set blending mode, page 113](#)

13 Setup Mode

13.1 Setup Mode, General Information

An interface to the PLC to set up kinematics is provided.

Data is exchanged via a shared memory between PLC and Motion. The data are stored in the KinData array according to the control type for the maximum number of kinematics (axis groups).

The structure ML_KINEMATICS_DATA_SM, ML_Robot.library, is available for each individual kinematics.

For the required elements to jog along the individual axes in this structure, refer to "Command interface to jog"

stJogCmd_q :ML_KIN_JOG; // x-x-xxxx -RW- Command interface for jog mode

See also [RobotControl Functionality Programmed in the PLC](#), page 253.

13.2 Jogging in Different Coordinate Systems

13.2.1 Parameters to be Set to Jog in Axis Coordinates

Kinematic axes can be moved manually in the axis coordinates. To jog a single axis in the setup mode, the parameters have to be set according to the jog mode.

	Incremental jogging,	distance [mm] or inch [in]
Short:	A-0-0080, Jogging, short distance	
	Velocity [mm/s] or [in/s]	
	Fast:	A-0-0083, Jogging, high velocity
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	A-0-0084, Jogging, acceleration
	Deceleration:	A-0-0085, Jogging, deceleration
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	A-0-0086, Jogging, acceleration jerk
	Deceleration jerk:	A-0-0087, Jogging, deceleration jerk
Long:	A-0-0081, Jogging, long distance	
	Velocity [mm/s] or [in/s]	
	Fast:	A-0-0083, Jogging, high velocity
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	A-0-0084, Jogging, acceleration
	Deceleration:	A-0-0085, Jogging, deceleration
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	A-0-0086, Jogging, acceleration jerk
	Deceleration jerk:	A-0-0087, Jogging, deceleration jerk
	Continuous jogging,	velocity [mm/s] or [in/s]
Slow:	A-0-0082, Jogging, low velocity	
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	A-0-0084, Jogging, acceleration

Setup Mode

Deceleration: A-0-0085, Jogging, deceleration

Jerk [mm/s³] or [in/s³]

Acceleration jerk: A-0-0086, Jogging, acceleration jerk

Deceleration jerk: A-0-0087, Jogging, deceleration jerk

Fast: A-0-0083, Jogging, high velocity

Acceleration [mm/s²] or [in/s²]

Acceleration: A-0-0084, Jogging, acceleration

Deceleration: A-0-0085, Jogging, deceleration

Jerk [mm/s³] or [in/s³]

Acceleration jerk: A-0-0086, Jogging, acceleration jerk

Deceleration jerk: A-0-0087, Jogging, deceleration jerk

13.2.2 Parameter to Be Set to Jog in World Coordinates

Kinematic axes can be moved manually in the world coordinates. To jog along a single coordinate in the setup mode, the parameters have to be set according to the jog mode.

Incremental jogging, distance [mm] or [in]

Short:	K-0-0080, Jogging, distance short in WC system
	Velocity [mm/s] or [in/s]
	Fast: K-0-0083, Jogging, velocity high in WC system
	Acceleration [mm/s²] or [in/s²]
	Acceleration: K-0-0084, Jogging, acceleration in WC system
	Deceleration: K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]
	Acceleration jerk: K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk: K-0-0087, Jogging, deceleration jerk in WC system
Long:	K-0-0081, Jogging, distance long in WC system
	Velocity [mm/s] or [in/s]
	Fast: K-0-0083, Jogging, velocity high in WC system
	Acceleration [mm/s²] or [in/s²]
	Acceleration: K-0-0084, Jogging, acceleration in WC system
	Deceleration: K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]
	Acceleration jerk: K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk: K-0-0087, Jogging, deceleration jerk in WC system

Continuous jogging, velocity [mm/s] or [in/s]

Slow:	K-0-0082, Jogging, velocity low in WC system
	Acceleration [mm/s²] or [in/s²]
	Acceleration: K-0-0084, Jogging, acceleration in WC system
	Deceleration: K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]
	Acceleration jerk: K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk: K-0-0087, Jogging, deceleration jerk in WC system
Fast:	K-0-0083, Jogging, velocity high in WC system
	Acceleration [mm/s²] or [in/s²]
	Acceleration: K-0-0084, Jogging, acceleration in WC system
	Deceleration: K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]
	Acceleration jerk: K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk: K-0-0087, Jogging, deceleration jerk in WC system

Setup Mode

13.2.3 Jogging in World Coordinates

Kinematic axes can be moved manually in the world coordinates. To jog along a single coordinate in the setup mode, the parameters have to be set according to the jog mode.

Incremental jogging, distance [mm] or [in]

Short:	K-0-0080, Jogging, distance short in WC system	
	Velocity [mm/s] or [in/s]	
	Fast:	K-0-0083, Jogging, velocity high in WC system
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	K-0-0084, Jogging, acceleration in WC system
	Deceleration:	K-0-0085, Jogging, deceleration in WC system
Long:	K-0-0081, Jogging, distance long in WC system	
	Velocity [mm/s] or [in/s]	
	Fast:	K-0-0083, Jogging, velocity high in WC system
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	K-0-0084, Jogging, acceleration in WC system
	Deceleration:	K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk:	K-0-0087, Jogging, deceleration jerk in WC system
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	K-0-0084, Jogging, acceleration in WC system
	Deceleration:	K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk:	K-0-0087, Jogging, deceleration jerk in WC system

Continuous jogging, velocity [mm/s] or [in/s]

Slow:	K-0-0082, Jogging, velocity low in WC system	
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	K-0-0084, Jogging, acceleration in WC system
	Deceleration:	K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	K-0-0086, Jogging, acceleration jerk in WC system
Fast:	K-0-0083, Jogging, velocity high in WC system	
	Acceleration [mm/s²] or [in/s²]	
	Acceleration:	K-0-0084, Jogging, acceleration in WC system
	Deceleration:	K-0-0085, Jogging, deceleration in WC system
	Jerk [mm/s³] or [in/s³]	
	Acceleration jerk:	K-0-0086, Jogging, acceleration jerk in WC system
	Deceleration jerk:	
	K-0-0087, Jogging, deceleration jerk in WC system	

14 RCL Parameters

14.1 Axis-specific Parameters Used by the Kinematics of an MLC

14.1.1 General Information

Limit values and axis-specific data are stored in the A-parameters of the respective axis.

For details, refer to "IndraMotion MLC Parameter Description".

14.1.2 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

14.1.3 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 acts on the parameter S-0-0055 of the drive device.



The position polarity has to be specified at first commissioning of an axis before creating the machine zero point reference for measuring systems, since other actual position values result when changing the polarity.

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 apply:

Motion of the primary component towards the cable connection side at positive position command difference and positive polarity.

See also: A-0-0048 and A-0-0049

Function The software position limit value, "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.

RCL Parameters

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

14.1.4 A-0-0030, Position limit value, positive

Function An axis's permitted travel range can be defined by the position limit values. The maximum allowed positive actual position value (positive traveling direction) is entered in this parameter.

The positive limit value is only active if all position data refers to the axis zero point, i.e. the axis is referenced "A-0-0021, Axis status, bit 10".

Use If the "A-0-2200, Command position" of the axis is set outside the positive position limit value, the error bit in "A-0-0021, Axis status, bit 9" is TRUE and generates the error "F2112053, Command target position outside of travel range".

If the "A-0-0100, Actual position value" exceeds the "A-0-0030, Position limit value, positive" or the "A-0-0031, Position limit value, negative", the axis sets the error bit in the "A-0-0021, Axis status, bit 9" and generates the error "F2116029, Positive position limit value exceeded" or "F2116030, Negative position limit value exceeded".

See also Functional Description: "Position Limit Values".

14.1.5 A-0-0031, Position limit value, negative

Function An axis's permitted travel range can be defined by the position limit values. The maximum allowed negative actual position value (negative travel direction) is entered in this parameter.

The negative limit value is only active if all position data refers to the axis zero point, i.e. the axis is referenced "A-0-0021, Axis status, bit 10".

Use If the "A-0-2200, Command position" of the axis is set outside the negative position limit value, the error bit in "A-0-0021, Axis status, bit 9" is TRUE and generates the error "F2112053, Command target position outside of travel range".

If the "A-0-0100, Position value" exceeds the "A-0-0030, Position limit value, positive" or the "A-0-0031, Position limit value, negative", the axis sets the

error bit in the "A-0-0021, Axis status, bit 9" and generates the error "F2116029, Positive position limit value exceeded" or "F2116030, Negative position limit value exceeded".

See also Functional Description: "Position Limit Values".

14.1.6 A-0-0032, Velocity limit value, positive

Function The parameter describes the maximum admissible velocity in the positive direction.

Use If the "A-0-0102, Actual velocity value" exceeds the absolute limit value "A-0-0032, Velocity limit value, positive" or "A-0-0033, Velocity limit value, negative", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112063, Velocity greater than limit A-0-0032 / A-0-0033".

The entered value is used as the maximum value for velocity variables in the positive direction.

See also Functional Description "Velocity Limitation".

14.1.7 A-0-0033, Velocity limit value, negative

Function The parameter describes the maximum admissible velocity in the negative direction.

Use If the "A-0-0102, Actual velocity value" exceeds the absolute limit value "A-0-0032, Velocity limit value, positive" or "A-0-0033, Velocity limit value, negative", the axis sets the warning bit "A-0-0021, Axis status, bit 8" and generates the warning "E0112063, Velocity greater than limit A-0-0032 / A-0-0033".

The entered value is used as the maximum value for velocity variables in the negative direction.

See also Functional Description "Velocity Limitation".

14.1.8 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".

14.1.9 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".

RCL Parameters

14.1.10 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".

14.1.11 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 become effective 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** apply: Motion of the primary component towards the cable connection side causes a positive actual velocity value and (positive polarity).

14.1.12 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 become effective 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.

14.1.13 A-0-0080, Jogging, short distance

Function The short distance for the incremental axis jogging is stored in this parameter.



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

Structure

- Unit:
Millimeters [mm] or inches [in] for axes with translatory scaling
Degrees for axes with rotary scaling
- Value range: >0.0 (floating point number)

RCL Parameters

- Default: 1.0 mm or inch or degree.

Use This parameter is used by the robot control for the incremental axis jogging. If an axis is coupled with another axis (e.g. lifting and rotating spindle), the other axis is controlled without its logic position being changed.

With the jog bits (KinData[1].stJogCmd_q), the PLC program can activate incremental jogging with short distance:

E.g. incremental jogging with short distance of the 3rd axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SMALL;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The maximum velocity is listed in A-0-0083 (always fast for incremental jogging).
- The maximum acceleration is listed in A-0-0084 and
- the maximum deceleration is listed in A-0-0085.
- The maximum jerk for the acceleration is listed in A-0-0086 and
- the maximum deceleration jerk is written in A-0-0087.

The jog motion can be prematurely completed by setting enJogMode = KIN_JOG_STOP.

Program:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.1.14 A-0-0081, Jogging, long distance

Function The long distance for the incremental axis jogging is stored in this parameter.



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

- Structure**
- Unit:
Millimeters [mm] or inches [in] for axes with translatory scaling
Degrees for axes with rotary scaling
 - Value range: >0.0 (floating point number)
 - Default: 10.0 mm or inch or degree.

Use This parameter is used by the robot control for the incremental axis jogging. If an axis is coupled with another axis (e.g. lifting and rotating spindle), the other axis is controlled without its logic position being changed.

With the jog bits (KinData[1].stJogCmd_q), the PLC program can activate incremental jogging with long distance:

E.g. incremental jogging with long distance of the 4th axis in negative direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_LARGE;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*4. bit for 4. axis of kinematics or group*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*no axis in pos.*)
```

- The distance to the jog the axis with long distance is listed in A-0-0081.

- The maximum velocity is listed in A-0-0083 (always fast for incremental jogging).
- The maximum acceleration is listed in A-0-0084 and
- the maximum deceleration is listed in A-0-0085.
- The maximum jerk for the acceleration is listed in A-0-0086 and
- the maximum deceleration jerk is written in A-0-0087.

The jog motion can be prematurely completed by setting enJogMode = KIN_JOG_STOP.

Program:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.1.15 A-0-0082, Jogging, low velocity

Function The low velocity for the continuous axis jogging is stored in this parameter.



It can also be written on this parameter in phase 4. The new value is only applied after the next phase switching of the control from P2 to BB.

- Structure**
- Unit:
mm/s or in/s for axes with translatory scaling,
degree/s for axes with rotary scaling.
 - Value range: >0.0 (floating point number)
 - Default: 10.0 mm/s or in/s or degree/s.

Use This parameter is used by the robot control for the continuous axis jogging.
If an axis is coupled with another axis (e.g. lifting and rotating spindle), the other axis is controlled without its logic position being changed.

With the jog bits (KinData[1].stJogCmd_q) the PLC program can activate continuous jogging at low velocity:

E.g. continuous jogging with low velocity of the 3rd axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;  
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_SLOW;  
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The maximum acceleration is listed in A-0-0084
- The maximum deceleration is listed in A-0-0085
- The maximum jerk for the acceleration is listed in A-0-0086
- The maximum deceleration jerk is written in A-0-0087

The continuous jogging can be completed by 2 different methods:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

and/or

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*no axis in pos.*)
```

14.1.16 A-0-0083, Jogging, high velocity

Function The high velocity for the continuous axis jogging is stored in this parameter.

RCL Parameters



It can also be written on this parameter in phase 4. The new value is only applied after the next phase switching of the control from P2 to BB.

- Structure**
- Unit:
mm/s or in/s for axes with translatory scaling,
degree/s for axes with rotary scaling.
 - Value range: >0.0 (floating point number)
 - Default: 25.0 mm/s or in/s or degree/s.
- Use** This parameter is used by the robot control for continuous **and incremental** axis jogging.
- If an axis is coupled with another axis (e.g. lifting and rotating spindle), the other axis is controlled without its logic position being changed.
- With the jog bits (KinData[1].stJogCmd_q), the PLC program can activate continuous jogging at high velocity:
- E.g. continuous jogging at higher fast velocity of the 4th. axis in negative direction:*

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_FAST;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*4. bit for 4. axis of kinematics or group*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*no axis in pos.*)
```

- The maximum acceleration is listed in A-0-0084
- The maximum deceleration is listed in A-0-0085
- The maximum jerk for the acceleration is listed in A-0-0086
- The maximum deceleration jerk is written in A-0-0087

The continuous jogging can be completed with two different methods:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

and/or

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*no axis in pos.*)
```

14.1.17 A-0-0084, Jogging, acceleration

- Function** The maximum acceleration to jog the axis (incremental and continuous) is stored in this parameter.



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- Unit:
mm/s² or in/s² for axes with translatory scaling,
degree/s² for axes with rotary scaling.
 - Value range: > 0.0 (floating point number, has to be ≤ A-0-0034)
 - Default: 370 mm/s² or in/s² or degree/s².
- Use** This parameter is used by the robot control to jog the axis. The jog acceleration is valid for all jog motions (incremental and continuous). The jog deceleration is listed in A-0-0085.

To jog in the world coordinate system (WC), the deceleration and acceleration values are determined in K-0-0084 and K-0-0085.

14.1.18 A-0-0085, Jogging, deceleration

Function The maximum deceleration to jog the axis (incremental and continuous) is stored in this parameter.



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- Unit:
mm/s² or in/s² for axes with translatory scaling,
degree/s² for axes with rotary scaling.
 - Value range: > 0.0 (floating point number, has to be ≤ A-0-0034)
 - Default: 100 mm/s² or in/s² or degree/s².

Use This parameter is used by the robot control to jog the axis.
The jog deceleration is valid for all jog motions (incremental and continuous).
The jog acceleration is listed in A-0-0084.
To jog in the world coordinate system (WC), the deceleration and acceleration values are determined in K-0-0084 and K-0-0085.

14.1.19 A-0-0086, Jogging, acceleration jerk

Function The maximum acceleration jerk to jog the axis (incremental and continuous) is stored in this parameter (indirect definition of the acceleration change time).



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- Unit:
mm/s³ or in/s³ for axes with translatory scaling,
degree/s³ for axes with rotary scaling.
 - Value range: ≥ 0.0 (floating point number, ≤ A-0-0036)
 - Default: 0.0 (no jerk limitation).

Use This parameter is used by the robot control to jog the axis. The jog acceleration jerk is valid for all jog motions (incremental and continuous).

Off $\Delta a = j * t$ The acceleration change time can be calculated from $t = \frac{\Delta a}{j}$

or $t = \frac{A-0-0084}{A-0-0086}$

The maximum jerk j for the deceleration is listed in A-0-0087.

The maximum jerk values to jog in world coordinates are applied with K-0-0086 and K-0-0087.

14.1.20 A-0-0087, Jogging, deceleration jerk

Function The maximum deceleration jerk to jog the axis (incremental and continuous) is stored in this parameter (indirect definition of the deceleration change time).

RCL Parameters



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- Unit:
mm/s³ or in/s³ for axes with translatory scaling,
degree/s³ for axes with rotary scaling.
 - Value range: ≥ 0.0 (floating point number, $\leq A-0-0036$)
 - Default: 0.0 (no jerk limitation).
- Use** This parameter is used by the robot control to jog the axis. The jog deceleration jerk is valid for all jog motions (incremental and continuous).

$$\Delta a = j * t \quad \text{The deceleration change time can be calculated from} \quad t = \frac{\Delta a}{j}$$

$$t = \frac{A-0-0085}{A-0-0087}$$

or

The maximum jerk for the acceleration is specified in A-0-0086.

The maximum jerk values to jog in world coordinates are applied with K-0-0086 and K-0-0087.

14.1.21 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"

14.1.22 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.

14.1.23 A-0-0230, Blending distance PTP

- Function** This parameter is relevant only for axes which are proceeded within a kinematics with the command " MOVE PTP ".
- With these axis movements the strength of the passing can be programmed with movement block transitions by the factor R_PTP.
- The value of this parameter indicates the distance to which the factor R_PTP refers.
- If the value is 0, no passing of the axis can take place.

14.1.24 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.

14.1.25 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"

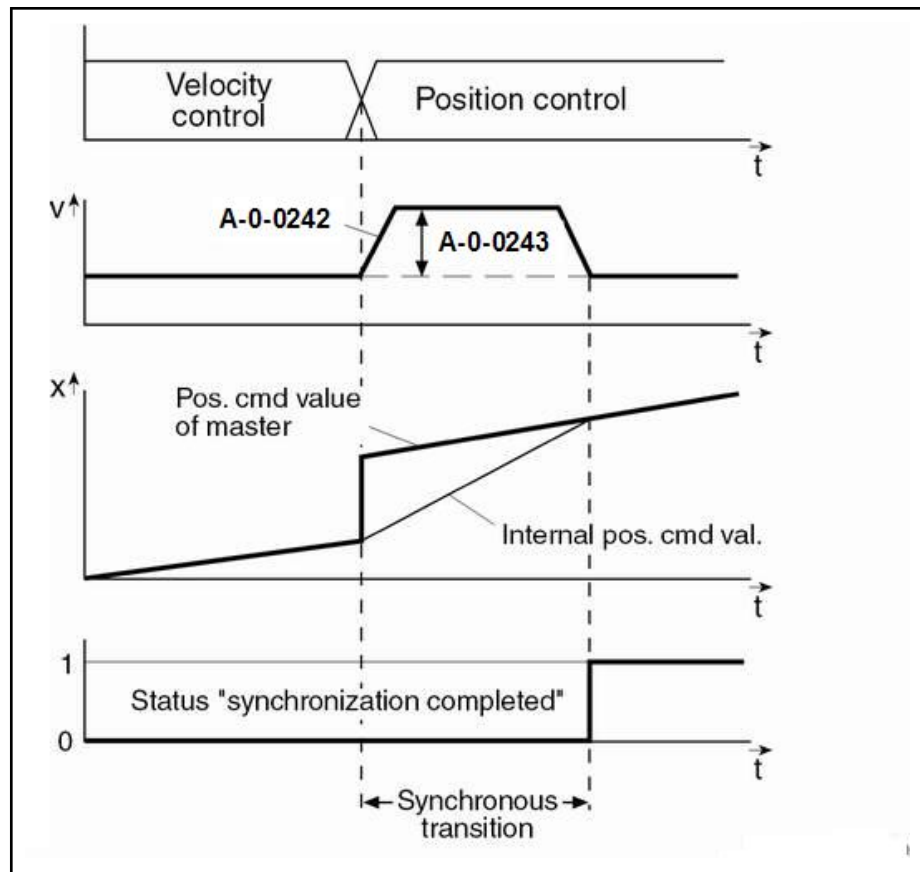
14.1.26 A-0-0242, Position control, synchronization acceleration

- Function** The parameter contains the acceleration value considered as drive-controlled when switching the operation mode from velocity control to position control during synchronization.

The drive synchronizes when changing the operation mode to the cyclic position control. That means that it creates a smooth transition of the internal position command value from the actual position to the new command value characteristic specified by the control. This creation is based on internal synchronization parameters (A-0-0242, A-0-0243) After the synchronization, the drive follows the position command values specified by the master in the Motion cycle

It is ensured drive-internally, that the operation mode switching is also switched in case of command value changes in jumps.

RCL Parameters



A-0-0242 Synchronization acceleration

A-0-0243 Synchronization velocity

Fig. 14-1: Time diagram "Drive-controlled operation mode switching"



The value of this parameter corresponds in its default state to the value of the parameter A-0-0034, Acceleration limit value, bipolar.

Individual values can be configured if bit 4 is set in the parameter A-0-0007, Axis configuration.

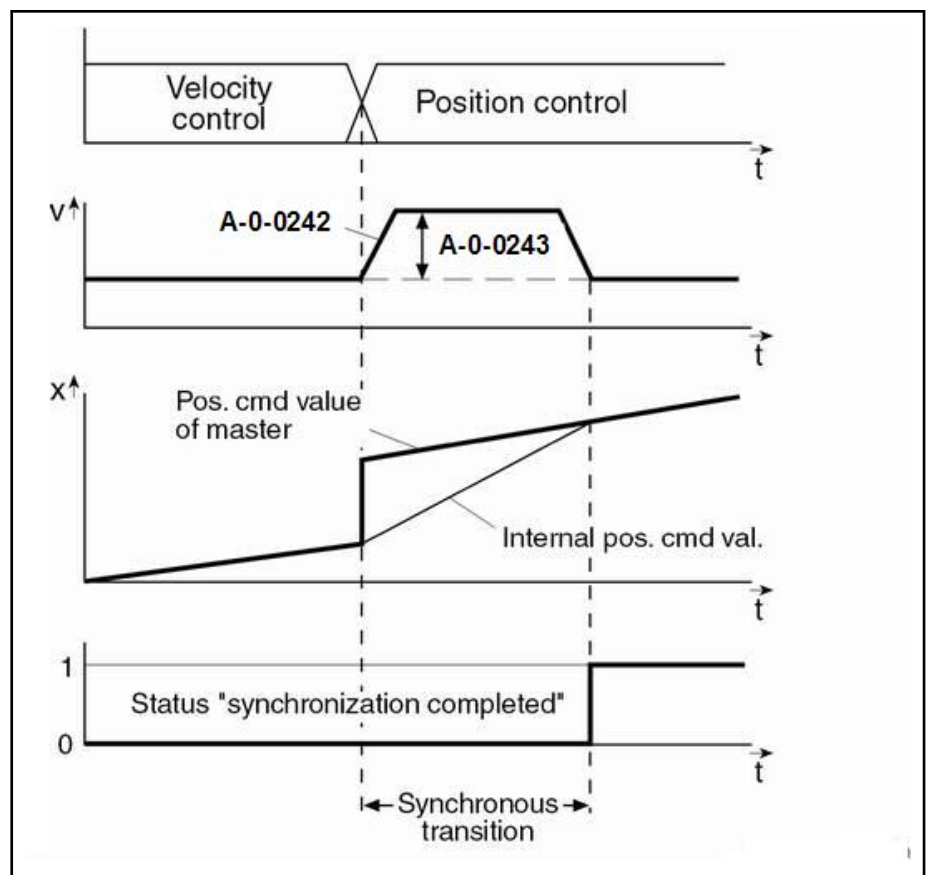
Use This value is synchronized to the drive when switching from P2 to BB.

14.1.27 A-0-0243, Position control, synchronization velocity

Function The parameter contains the velocity value considered as drive-controlled when switching the operation mode from velocity control to position control during synchronization.

The drive synchronizes when changing the operation mode to the cyclic position control. That means that it creates a smooth transition of the internal position command value from the actual position to the new command value characteristic specified by the control. This creation is based on internal synchronization parameters (A-0-0242, A-0-0243) After the synchronization, the drive follows the position command values specified by the master in the Motion cycle

It is ensured drive-internally, that the operation mode switching is also switched in case of command value changes in jumps.



A-0-0242 Synchronization acceleration

A-0-0243 Synchronization velocity

Fig. 14-2: Time diagram "Drive-controlled operation mode switching"



This parameter value corresponds by default to the maximum absolute value of the parameters A-0-0032/A-0-0033, Velocity limit value, positive/negative.

Individual values can be configured if bit 4 is set in the parameter A-0-0007, Axis configuration.

Use This value is synchronized to the drive when switching from P2 to BB.

14.2 Common C-Parameters of All Kinematics of an MLC

14.2.1 General Information

The following C-parameters are created to manage **all** kinematic data of one control.

For details, refer to "IndraMotion MLC Parameter Description".

14.2.2 C-0-0123, Maximum number of kinematics

Function This parameter indicates the maximum number of kinematics.

14.2.3 C-0-2480, Configured kinematic number

Function This parameter indicates the number of the currently configured kinematics.

RCL Parameters

14.2.4 C-0-2482, List of non-linked axes

- Function** An axis can be assigned only to one kinematics.
This lists parameter indicates the axes which are still not assigned to a kinematics.
If no kinematics is configured, this list corresponds to parameter "C-0-0483, List of the configured axes".
The allocation of the axes to the kinematics takes place via parameter "K-0-0008, Axis configuration list".

14.2.5 C-0-2483, List of configured kinematics

- Function** This parameter indicates the configured kinematics of the system.
Use Example:
3 kinematics (kinematics 1, 2 and 3) are configured in the system.
C-0-2483 indicates:

Element	Value	Comment
1	1	Kinematics 1 is configured
2	2	Kinematics 2 is configured
3	3	Kinematics 3 is configured

14.2.6 C-0-2484, Kinematic configuration list

- Function** This parameter defines the kinematic 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 kinematics configured
1	Kinematics configured

- Use** The list must not contain any gaps with non-configured kinematics. This is checked when switching to phase 4.

If this is not the case, the parameter "C-0-2484, Axis configuration list" is entered into the parameter "C-0-0011, List of all invalid C-parameters" and an incorrect configuration is displayed (F0160040, Error in cross-check of C-parameters).

Writing on the parameters "C-0-2484, Configuration list of the kinematics" updates the parameters "C-0-2480, Configured number of kinematics" and "C-0-2483, List of the configured kinematics".

Example A:

A kinematics is to be configured.

C-0-2484 indicates:

RCL Parameters

Element	Value	Comment
1	1	Kinematics 1 is configured
2	0	No kinematics 2 used/required
3	0	No kinematics 3 used/required
4	0	No kinematics 4 used/required

Example B:

Three kinematics are to be configured (version 04V12 and higher).
C-0-2484 indicates:

Element	Value	Comment
1	1	Kinematics 1 is configured
2	1	Kinematics 2 is configured
3	1	Kinematics 3 is configured
4	0	No kinematics 4 used/required

14.3 Parameters of a Kinematics in the MLC

14.3.1 General Information

The following K-parameters are created to manage the data of **one** kinematics in the control. Each parameter exists once in each kinematics.

For details, refer to "IndraMotion MLC Parameter Description".

14.3.2 K-0-0001, Logic kinematic number

Function This parameter shows the actual logical kinematics number.

14.3.3 K-0-0002, Kinematic name

Function The name of the kinematics (e.g. Robi_1) is written in this parameter.

A global variable of type AXIS_GROUP_REF is automatically created with this name.



The following restrictions apply:

- maximum of 12 characters,
- A...Z, a...z, 0..9, '_'
- first character must be a letter,
- "Robi_1" is identical to "ROBI_1".

14.3.4 K-0-0003, Maximum path speed

Function Maximum programmable motion velocity in world coordinates (WC) at the Tool Center Point (TCP).

Depending on the parameter K-0-0054, the entry is made in mm/s or in/s.

Determining a reasonable maximum path velocity depends on the resolution of the measuring system, the minimum programmed point distance, the robot mechanics and the sample time of the position control loop (clock starting time).

RCL Parameters

14.3.5 K-0-0004, Reserved

Function This parameter is reserved for internal applications.

14.3.6 K-0-0005, Software limit switches, positive, WC

Function This parameter determines the values of the positive software limit switch in world coordinates.

Depending on the parameter K-0-0054, the entry is made in millimeters [mm] or inches [in] or degrees.

Only effective in jog mode.

Due to the kinematics, an internal travel range limit can be reached when traversing in world coordinates in automatic or manual mode.

The motion is immediately stopped and "travel range limit reached" is displayed in the diagnostic error mode.

It can also be written on this parameter in phase 4. The new value is only applied after the next phase switching of the control from P2 to BB.

14.3.7 K-0-0006, Software limit switches, negative, WC

Function This parameter determines the values of the negative software limit switch in world coordinates.

Depending on the parameter K-0-0054, the entry is made in millimeters [mm] or inches [in] or degrees.

Only effective in jog mode.

Due to the kinematics, an internal travel range limit can be reached when traversing in world coordinates in automatic or manual mode.

The motion is immediately stopped and "travel range limit reached" is displayed in the diagnostic error mode.

It can also be written on this parameter in phase 4. The new value is only applied after the next phase switching of the control from P2 to BB.

14.3.8 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.

14.3.9 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.

RCL Parameters

Value	Comment
0	No axis configured
8	Positioning axis configured
16	Orientation axis configured
32	Belt axis configured

Fig. 14-3: Valid values of an axis configuration

Value	Comment
0	Individual control
1-64	Control number in C2C link

Fig. 14-4: 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 must be located behind the positioning axes.

Belt axes must be located behind 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."



When switching to phrase 4, it is checked whether at least one axis is configured for each configured kinematics and whether the configured axes are present.

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.

If this is not the case, the parameter "K-0-0008, Axis configuration list" is entered in the parameter "K-0-0011, List of all invalid K-parameters" and an incorrect configuration is displayed (F0160040, Error in cross-check of K-parameters).

RCL Parameters

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).

14.3.10 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.

14.3.11 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:

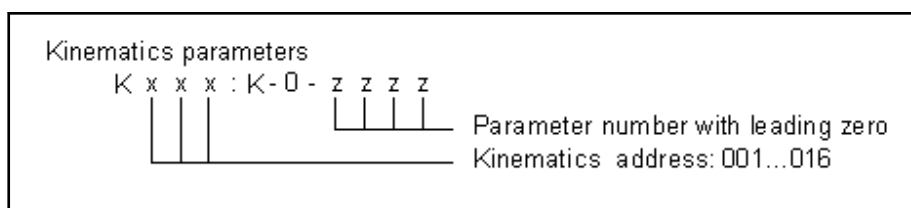


Fig. 14-5: K-parameter structure

Example:

K001: K-0-0004 - Kinematics 1; parameter K-0-0004.

14.3.12 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.

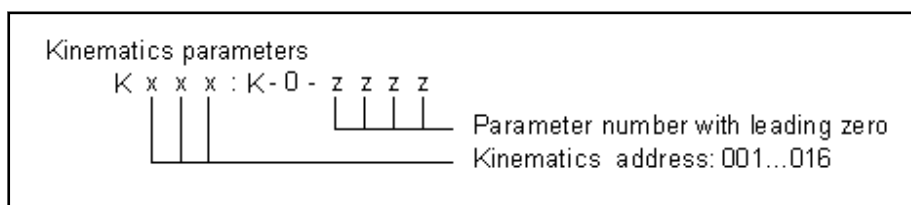


Fig. 14-6: Structure of a K-parameter

Use K001: K-0-0008 - cinematics 1; parameter K-0-0008.

14.3.13 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".

14.3.14 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.

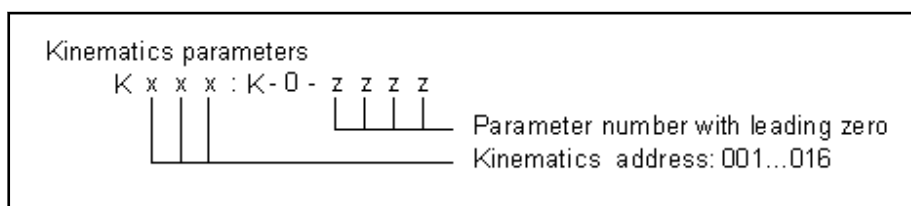


Fig. 14-7: Structure of a K-parameter

Use K001: K-0-0008 - cinematics 1; parameter K-0-0008.

14.3.15 K-0-0014, Path acceleration, change time

- Function** This parameter is only effective with the RCL commands MOVE LINEAR and MOVE CIRCULAR in the block transition mode PROGR_SLOPE.
- In the block transition mode BLOCK_SLOPE, a jerk limitation can be programmed using the RCL keyword "J".

RCL Parameters

The change time only affects the acceleration, but not the deceleration.

The deceleration change time is set using parameter K-0-0015.



The deceleration change time specified here is the maximum time until the acceleration/ braking deceleration/ programmed in the RCL program is reached.

We distinguish 3 movement forms:

- Acceleration in rectangle mode:
K-0-0014 and K-0-0015 are 0.

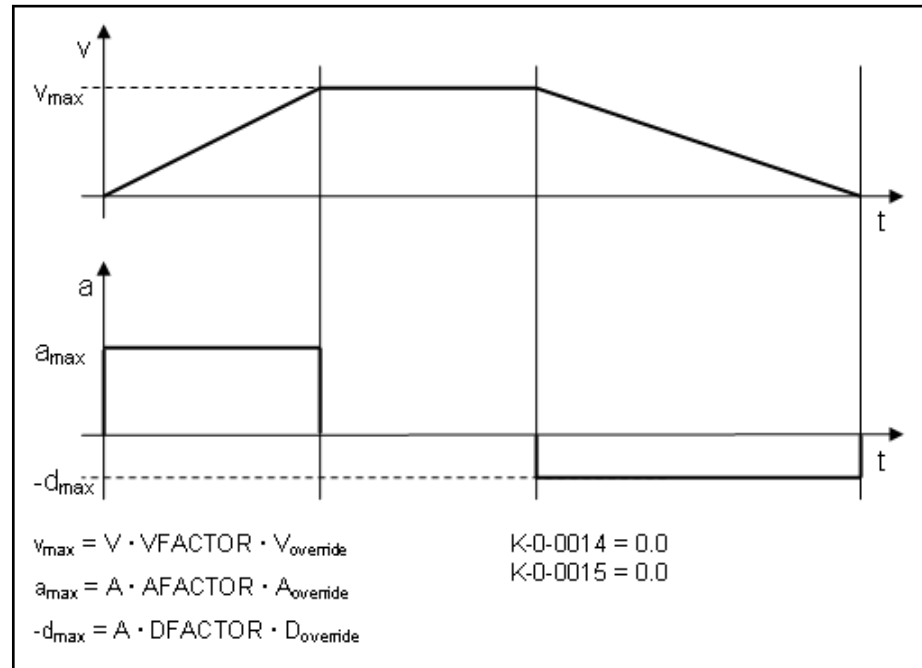


Fig. 14-8: Acceleration/deceleration in rectangle mode

The resulting jerk j is mathematically infinite.

- Acceleration in trapezoid mode:
K-0-0014 and/or K-0-0015 are greater or equal to the sercos cycle time (see also C-0-0400). The programmed acceleration must be constantly available at a certain time to reach the programmed velocity.

RCL Parameters

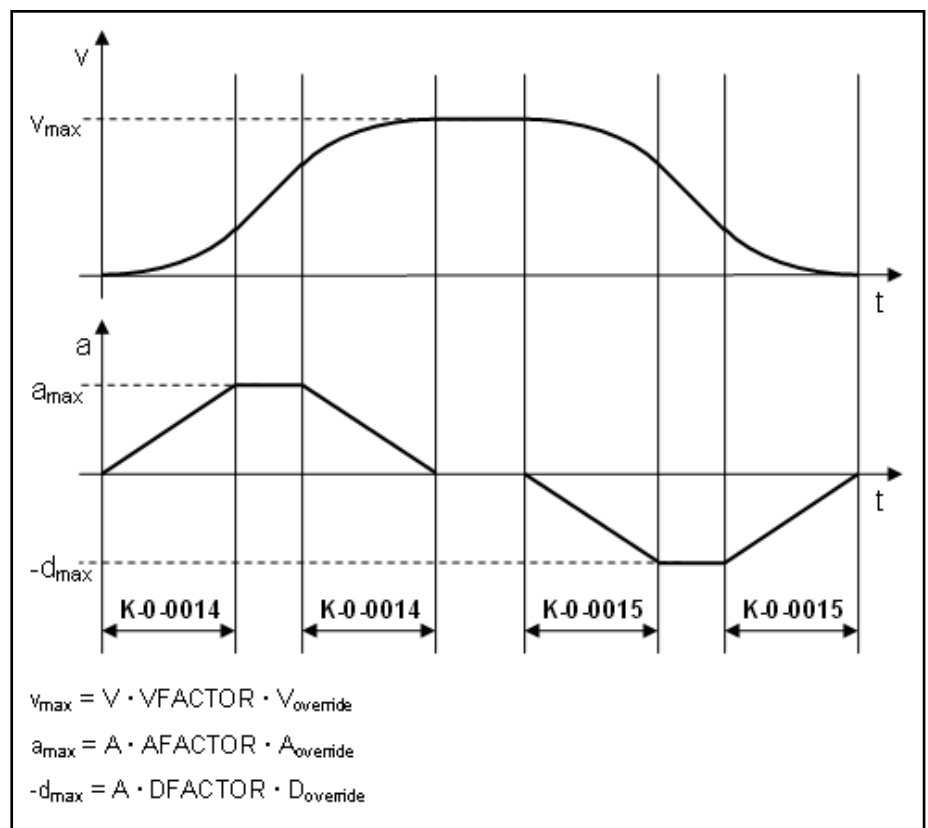


Fig. 14-9: Acceleration/deceleration in trapezoid mode

$$j = \frac{a_{\max}}{K-0-0014}$$

The resulting jerk j is

- Acceleration in triangle mode:
The times specified in K-0-0014 and K-0-0015 cannot be maintained to reach the programmed acceleration.

RCL Parameters

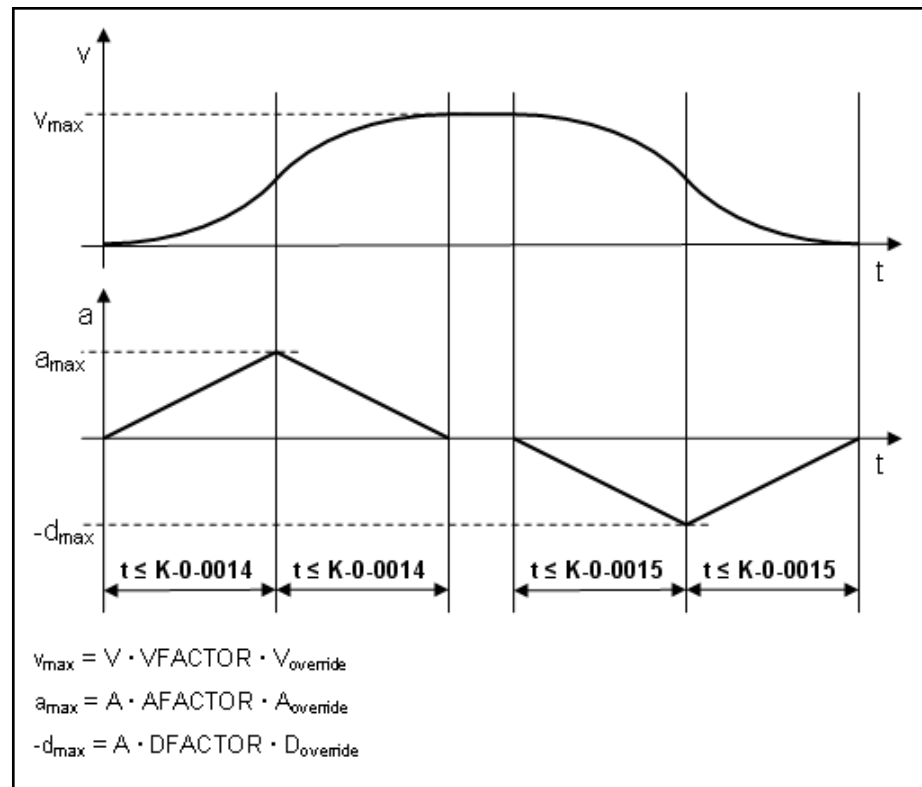


Fig. 14-10: Acceleration/deceleration in triangle mode

$$j = \frac{a_{\max}}{t}$$

The resulting jerk j is

Use The greater the change times, the smoother the motion profile. If the programmed acceleration/deceleration value is too large, the path planning is switched over to the aforementioned triangle mode.



The greater the programmed acceleration and the smaller the acceleration change time, the greater the jerk.

Low velocities can result in the very small acceleration change time t in triangle mode, which, in turn, produces a large jerk.

14.3.16 K-0-0015, Path deceleration, change time

Function This parameter is only effective with the RCL commands MOVE LINEAR and MOVE CIRCULAR in the block transition mode PROGR_SLOPE.

In the block transition mode BLOCK_SLOPE, a jerk limitation can be programmed using the RCL keyword "J".

The change time only affects the deceleration, but not the acceleration.

The acceleration change time is set using parameter K-0-0014.



The deceleration change time specified here is the maximum time until the acceleration/ braking deceleration/ programmed in the RCL program is reached.

We distinguish 3 movement forms:

- Acceleration in rectangle mode:
K-0-0014 and K-0-0015 are 0.

RCL Parameters

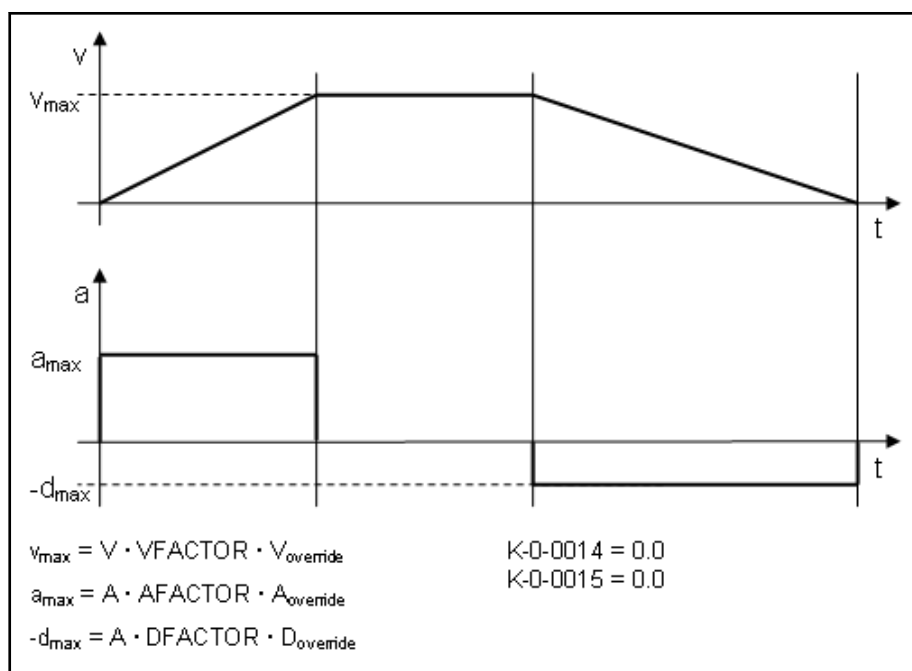


Fig. 14-11: Acceleration/deceleration in rectangle mode

The resulting jerk j is mathematically infinite.

- Acceleration in trapezoid mode:
K-0-0014 and/or K-0-0015 are greater or equal to the sercos cycle time (see also C-0-0400). The programmed acceleration must be constantly available at a certain time to reach the programmed velocity.

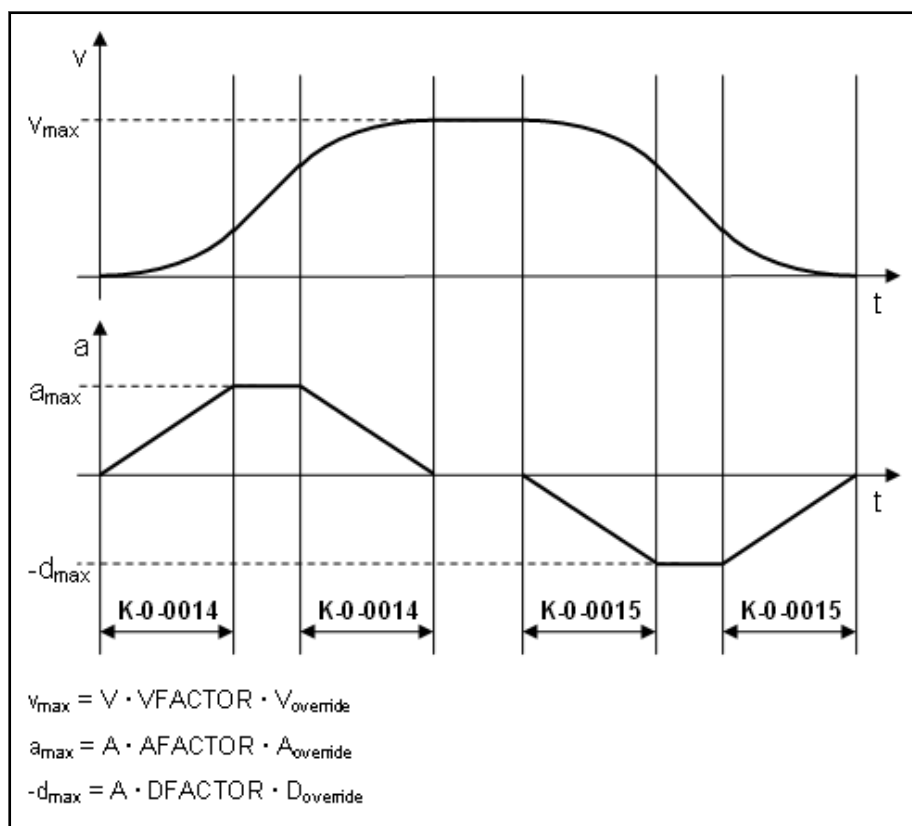


Fig. 14-12: Acceleration/deceleration in trapezoid mode

RCL Parameters

$$j = \frac{a_{\max}}{K-0-0015}$$

The resulting jerk j is

- Acceleration in triangle mode:

The times specified in K-0-0014 and K-0-0015 cannot be maintained to reach the programmed acceleration.

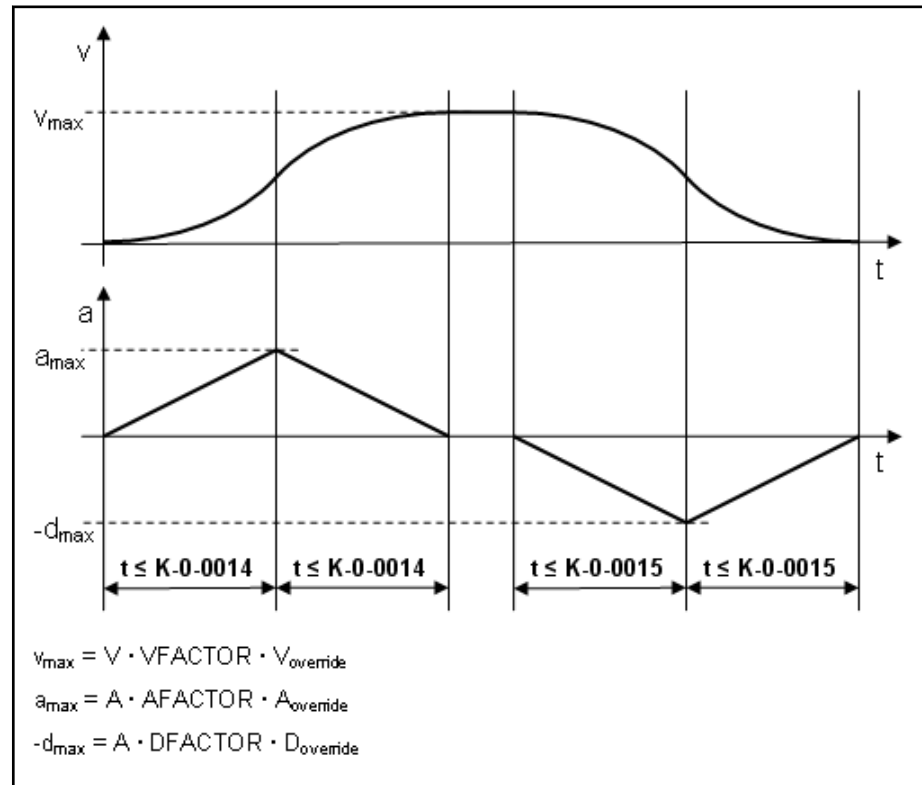


Fig. 14-13: Acceleration/deceleration in triangle mode

$$j = \frac{a_{\max}}{t}$$

The resulting jerk j is

Use The greater the change times, the smoother the motion profile. If the programmed acceleration/deceleration value is too large, the path planning is switched over to the aforementioned triangle mode.



The greater the programmed acceleration and the smaller the acceleration change time, the greater the jerk.

Low velocities can result in the very small acceleration change time t in triangle mode, which, in turn, produces a large jerk.

14.3.17 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.

14.3.18 K-0-0019, Kinematic type

Function This parameter indicates the kinematic type.
Structure

Bit	Content	Bit name
1..0	01 - Robot Control V1 10 - Robot Control V2	Kinematic type
31..2	Reserved	

Fig. 14-14: K-0-0019 Kinematic type



Bits 0..2 for the operation modes are evaluated as numerals (0..7).

14.3.19 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 parameter "K-0-0021, Kinematics status" is set to 1, this parameter contains the warning (bit 8) or error (bit 9) text.

From the perspective of the PLC, this information is available to the kinematic "*" as

- KinData[*].dwKinStatus_i.8= TRUE, diagnostic message in K-0-0020 is a warning text or
- KinData[*].dwKinStatus_i.8= TRUE, diagnostic message in K-0-0020 is an error text.

14.3.20 K-0-0021, Kinematic status

Function This parameter describes the current status of the kinematics.

This is internal system information that is triggered via PLC function blocks for example. This is no information on the state of the kinematic state machine (see also K-0-0022).

Structure

Bit	Content	Bit name
2...0	001 - Automatic 010 - Manual	Kin_OpMode_Auto Kin_OpMode_Manual
4..3	Jog coordinate system 00 - JC 01 - WC original 10 - WC workpiece 11 - WC tool	Kin_JogCoord1 (Bit 3) Kin_JogCoord2 (Bit 4)
5	All kinematic axes are in standstill. (Belt axes are not considered)	Kin_StandStill
6	Command position = actual position The programmed end position is reached.	Kin_InPosition
7	Belt-synchronous	Kin_BeltSynchronous
8	Warning	Kin_Warning
9	Error	Kin_Error

RCL Parameters

Bit	Content	Bit name
10	Homing required; not all axes are referenced.	Kin_NotHomed
12	Transformed world coordinates are invalid	Kin_TrafoValuesInvalid
13	Operating mode MANUAL_JOG is enabled KinData[].stJogCom_q.enJogCmdVel $\neq$ KIN_JOG_STOP	Kin_JogEnabled
14	All axes are ready for operation (AH or AF)	Kin_DrivesOn
15	Feed enabled Enabled for traversing motions.	Kin_FeedEnabled
16	Block processing enabled Processing the next block enabled	Kin_BlockEnabled
17	Error reaction enabled	Kin_ErrorReaction
19..18	Jog modes 00 - small steps 01 - big steps 10 - slow 11 - fast	Kin_JogMode1 (bit 18) Kin_JogMode2 (bit 19)
21	Program selected	Kin_ProgSelected
23	Reset enabled	Kin_ResetActive
24	Program ended	Kin_ProgEnded
25	Program started	Kin_ProgStarted
26	Program: PAUSE command enabled	Kin_ProgPaused
28	Spline is calculated for blending motion	
29	Spline is valid	

Fig. 14-15: K-0-0021 Kinematic status



Bits 0..2 for the operation modes are evaluated as numerals (0..7).

Use Bit 5: All kinematic axes are in standstill

This bit signals that the axes positioned by the kinematics are in standstill. This means that the velocity is zero.

This bit differs from Kis_Standstill (see parameter K-0-0022) which refers to the kinematic state machine.

Bit 8: Warning

If an incorrect state is detected which still permits the correct operation, a warning is output.

Example:

The programmed path velocity cannot be reached due to the maximum velocity of one or more axes.



If they are ignored, some warnings cause an error.

Bit 9: Error

If an incorrect state detected which does not allow any correct operation any longer, an error is output.

Example:

The programmed position is located beyond the traversing limit of one or more axes.

Bit 12: Transformed world coordinates are invalid

The current world coordinates 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 17: Error reaction active

When errors with error reaction capability occur (F0 errors do not have error reaction capability), all of the axes assigned to the axes are stopped as quickly as possible keeping the programmed path.

Bit 17 is deleted when all axes are stopped.

Bit 26: Program: PAUSE command enabled

In the robot program, a PAUSE command is enabled (see PAUSE command).

Bit 28: Spline is calculated for blending motion

The bit is active until a spline is calculated for a blending motion with CONT_SLOPE.

Bit 29: Spline is valid

The bit is active if the spline (to blend with CONT_SLOPE) of the currently selected motion command is valid.

14.3.21 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.

The abbreviation KIS in the bit name stands for "Kinematic State Machine" in contrast to the KIN bits. The KIN bits describe individual kinematic messages.

Structure The kinematic state machine is explained in the following figure:

RCL Parameters

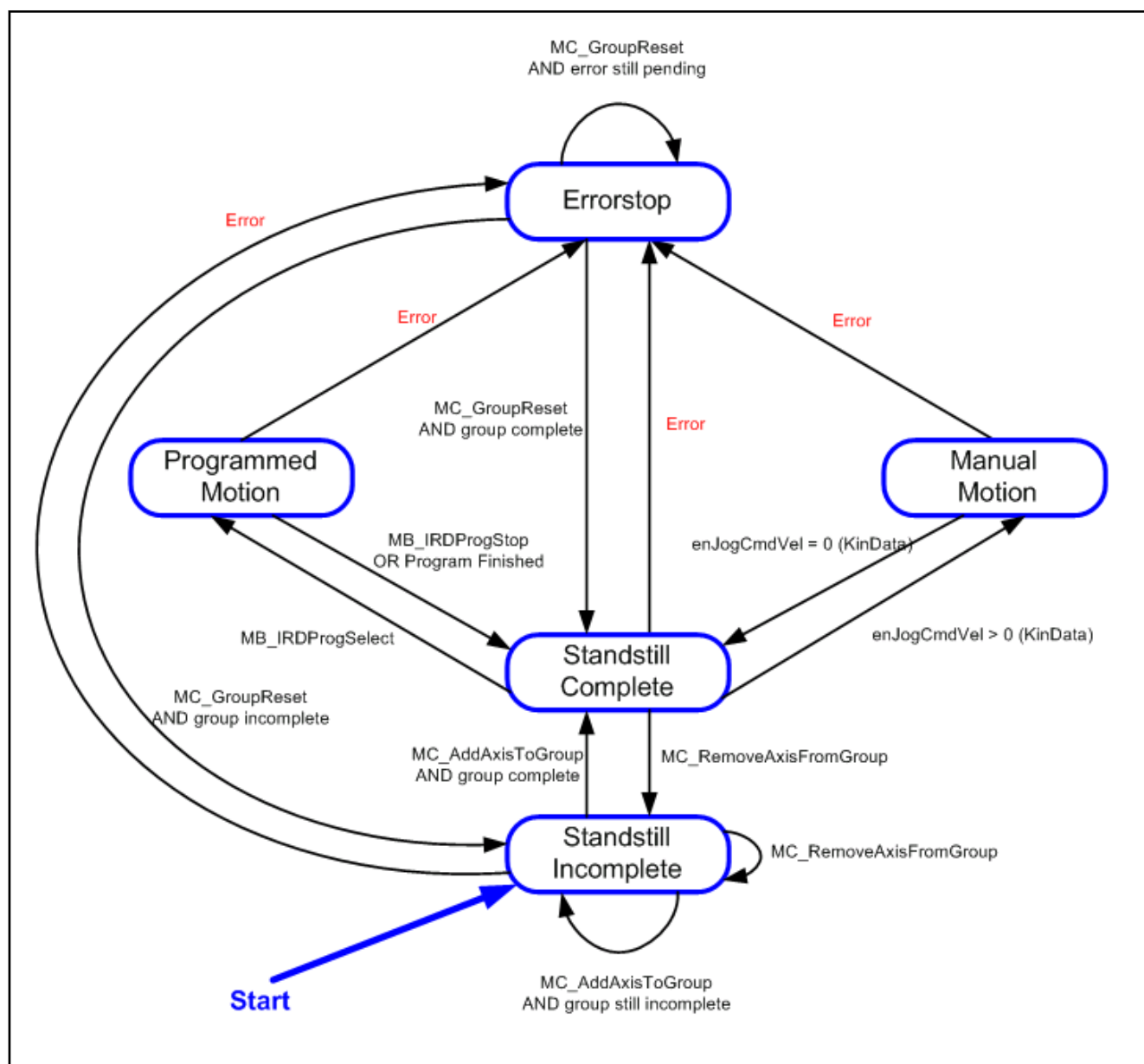


Fig. 14-16: Kinematic state machine

"Programmed Motion" has the following substates:

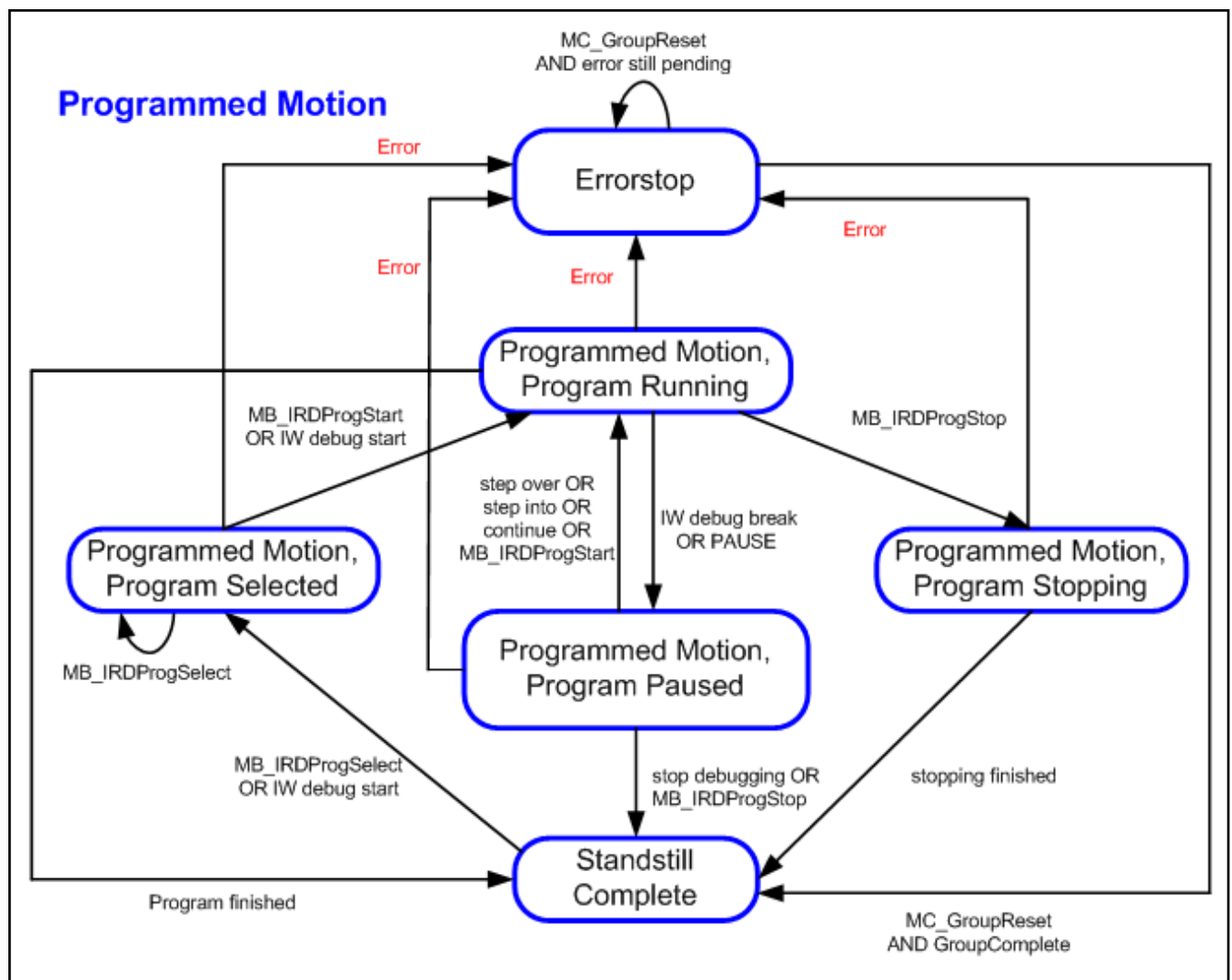


Fig. 14-17: "Programmed Motion" state machine

"Errorstop" state

The "Errorstop" state is reached when an error occurs while the kinematics is running. The axes remain in the "Coordinated Motion" state and remain assigned to the kinematics. Except for axes that have caused an error or that are currently in error state.

The "Errorstop" state can be reset via the MB_GroupReset function block or the command parameter "C-0-1030, Command: Clear all control errors". Afterwards, it is switched to the "Standstill Complete" state for complete kinematics or to the "Standstill incomplete" state for incomplete kinematics.

Bit field definition

In parameter K-0-0022, the following bits are available:

Bit	Content	Name of the bit
0	Kinematic is in "error stop"	Kis_Errorstop
1	Kinematic is "stopping"	Kis_Stopping
3	Not all axes are assigned to the kinematic (incomplete)	Kis_Incomplete
5	Kinematic is in "programmed motion"	Kis_ProgrammedMotion
7	Kinematic is in "StandStill"	Kis_StandStill

RCL Parameters

Bit	Content	Name of the bit
8	Program is selected	Kis_ProgramSelected
9	Kinematic is enabled (running)	Kis_Running
10	Kinematic is in jog status KinData[].stJogCom_q.enJogCmdVel KIN_JOG_STOP	Kis_ManualMotion
11	Kinematic is in "Pause"	Kis_ProgramPaused

Fig. 14-18: K-0-0022 Extended kinematics status

Bit assignment/state

The assignment between the outlined states and their corresponding bits is shown in the following table.

1 : Bit set

0 : Bit not set

x : Bit irrelevant for state

State	Bits 12...8	Bits 7...4	Bits 3...0
Standstill incomplete	00000	1x0x	1x00
Standstill complete	00000	1x0x	0x00
Programmed Motion, program selected	00001	0x1x	0x00
Programmed Motion, program running	00010	0x1x	0x00
Programmed Motion, program stopping	00000	0x1x	0x10
Programmed Motion, program paused	01000	0x1x	0x00
Errorstop	00000	0x0x	0x01
Manual Motion	00100	0x0x	0x00
GroupMoving	10000	0x0x	0x00

Fig. 14-19: Assigning kinematic state to status bits

14.3.22 K-0-0023, Kinematics 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.

14.3.23 K-0-0031, Transformation scheme

Function This parameter describes the scheme used to couple the kinematic axes mechanically. It identifies how pivot angles or axis positions in Cartesian coordinates are to be transformed.

Use The meaning of the individual values of parameter "K-0-0040, Transformation, lengths of the mechanics " and "K-0-0041, Transformation, factors" depends on this set transformation scheme.

More specific information can be found in the special transformation documentation.

Value	Comment
0	Robot type 00, Cartesian robot
9	Robot type 09, 3 axis Scara with 1 optional manual axis
10	Robot type 10, 4 axis Scara with 1 optional manual axis

RCL Parameters

Value	Comment
20	Robot type 20, 3 axis Delta with 1 optional manual axis
≥1000	User specified transformation

Fig. 14-20: Valid values for the transformation scheme



This parameter must be set when starting operation according to DCC service specifications.

When switching after phase 4, a check is made to see if enough axes are configured for the transformation scheme in parameter "K-0-0008, Axis configuration list."

If this not the case, this parameter is entered in parameter "K-0-0011, List of all invalid K-parameters" and

a configuration error is displayed ("F0160040, Error in cross-check K-parameters").

The kinematic wizard does not make any parameter checks for user-defined kinematic types.

14.3.24 K-0-0032, Referencing axes, command configuration

Function A presetting to reference all kinematic axes can be set for various robot types.

Referencing all axes:

- 0 - no, default setting
- 1 - yes, only allowed for specific transformations.



This parameter is to be set when commissioning according to DCC service specifications.

14.3.25 K-0-0033, IPOS, update at coordinate change

Function A presetting can be made for IPOS updater at coordinate change for various robot types.

IPOS, update when coordinate change:

- 0 - no, default setting,
- 1 - yes, only allowed for specific transformations.



This parameter must be set when commissioning according to DCC service specifications.

14.3.26 K-0-0036, WC axis monitoring

Function WC axis monitoring can be configured for various robot types (monitoring axis velocity).

Use *WC axis monitoring:*

- 0 = No monitoring
- 1 = Monitoring with motion abort.
- 2 = Only in manual mode, monitoring with limitation of axis velocities. No monitoring in automatic mode.
- 3 = Do not use this setting.

RCL Parameters

- 4 = In manual and automatic mode, monitoring with limitation of axis velocities. The limitation applies for the complete motion.
- 5 = Continuous monitoring of the axis velocity during linear motion.

Please note the following:

- The high JC jog velocities set in parameter A-0-0083 are monitored in manual mode. The maximum axis velocities set in $\text{MIN } \{|A-0-0032| \vee |A-0-0033|\}$ are monitored in automatic mode.
- If the axis limitation occurs during the motion, the velocity at the TCP does not match the programmed velocity.
- If the axis limitation occurs, there are path inaccuracies.

14.3.27 K-0-0040, Transformation, lengths of the mechanics

Function Lengths required to transform axis positions into Cartesian coordinates are configured in this parameter.

The meaning of the individual values depends on the set "K-0-0031, Transformation scheme".

More specific information can be found in the special transformation documentation.



When switching to phase 4, the values are checked depending on parameter "K-0-0031, Transformation scheme". (For transformation scheme SCARA, it is checked for example whether the first two lengths are not equal to zero.)

If an error occurs when the values are checked,

- this parameter is entered in the parameter "K-0-0011, List of all invalid K-parameters" and
- an incorrect configuration is displayed ("F0160040, Error in cross-check K-parameters").

14.3.28 K-0-0041, Transformation, factors

Function Factors required to transform axis positions to Cartesian coordinates are configured in this parameter.

The meaning of the individual values depends on the set "K-0-0031, Transformation scheme". More specific information can be found in the special transformation documentation.



When switching to phase 4, the values are checked depending on parameter "K-0-0031, Transformation scheme".

If an error occurs when the values are checked,

- the parameter "K-0-0041, Transformation factors" is entered in the parameter "K-0-0011, List of all invalid K-parameters" and
- an incorrect configuration is displayed (F0160040, Error in cross-check K-parameters).

14.3.29 K-0-0042, WC system offset

Function This parameter allows the coordinate system origin of the transformation scheme to be displaced to allow editing of coordinate values that refer to a reference point in the editing cell.

RCL Parameters

Structure This parameter is a list parameter with six values.

Element	Description
1	Displacement in Cartesian x-direction [in millimeters or inches (see K-0-0054)]
2	Displacement in Cartesian y-direction [in millimeters or inches (see K-0-0054)]
3	Displacement in Cartesian z-direction [in millimeters or inches (see K-0-0054)]
4	Rotation along the Cartesian x-axis [in degrees] (see note)
5	Rotation along the Cartesian y-axis [in degrees] (see note)
6	Rotation along the Cartesian z-axis [in degrees] (see note)

Use The first three values indicate the position of the coordinate system origin from the perspective of the cell coordinate system. The elements 4 to 6 indicate the rotation of the coordinate system.



The coordinate system can only be rotated for particular transformation schemes (see K-0-0031). This is documented in the corresponding descriptions (no information on the rotation of the coordinate system means that this option is not supported with this particular transformation scheme).

14.3.30 K-0-0043, Assignment list of workpiece coordinates

Function This parameter describes which robot coordinate affects which displacement/rotation of the workpiece coordinate system.

If the kinematics is provided with less than six degrees of freedom, not all displacements/rotations of the workpiece coordinate system are compensated. Using this parameter, it is reported to the control

- which of the six general workpiece coordinates from the mechanics are considered
- which robot coordinate corresponds to which workpiece coordinate

Structure This parameter is a list parameter with six values.

Element	Description
1	Robot coordinate including the displacement in Cartesian x-direction
2	Robot coordinate including the displacement in Cartesian y-direction
3	Robot coordinate including the displacement in Cartesian z-direction
4	Robot coordinate including the rotation along the Cartesian x-axis
5	Robot coordinate including the rotation along the Cartesian y-axis
6	Robot coordinate including the rotation along the Cartesian z-axis



If a rotation is mechanically impossible, 0 has to be entered in the respective element.

RCL Parameters



The coordinate system can only be rotated for particular transformation schemes (see K-0-0031). This is documented in the corresponding descriptions (no information on the rotation of the coordinate system means that this option is not supported with this particular transformation scheme).

Use *Example:***4-axis Scara**

A 4-axis Scara can traverse in x-, y-, z- direction and rotate along the z-axis. Thus, the following assignment results for the parameter K-0-0043:

K-0-0043 assignment at a 4-axis Scara

Element	Value	Description
1	1	The first axis of the kinematics is used for displacement in x-direction
2	2	The second axis of the kinematics K-0-0042 is used for displacement in y-direction
3	3	The third axis of the kinematics K-0-0042 is used for displacement in z-direction
4	0	Rotation of the workpiece along the x-axis is mechanically not possible
5	0	Rotation of the workpiece along the y-axis is mechanically not possible
6	4	The fourth axis of the kinematics is used for rotating the workpiece coordinate system along z-direction

Fig. 14-21: K-0-0043 assignment at a 4-axis Scara

14.3.31 K-0-0045, Tool selection

- Function** The parameter enables the tool by entering the name as string in phase BB. The name has to be configured with the corresponding coordinate values in the TOOLS.DAT file.
- This allows the use of a tool coordinate system in the PLC without using the TOOL command in the RCL.

14.3.32 K-0-0050, Distance units WC

- Function** The distance and orientation 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 type of transformer K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm
 - For alpha, beta, gamma: degrees

Use *Examples*

1. Cartesian robot with X, Y, Z, C
K-0-0050 = "mm,mm,mm,degree"
2. Scara with distance lathe spindle (X, Y, Z, C)
K-0-0050 = "mm,mm,mm,degree"
3. Cartesian robot with X, Y, Z, C, B
K-0-0050 = "mm,mm,mm,degree,degree"
4. Cartesian robot with X, Y, C
K-0-0050 = "mm,mm,degree"
5. Delta with C-axis and 2 belts
K-0-0050 = "mm,mm,mm,degree"



The belts are not included in the lists for the world coordinate system.

14.3.33 K-0-0051, Velocity units WC

Function The velocity units for each element of a list parameter in the world coordinate system are separated with a command and saved in this text parameter.

- Only legible entries are automatically determined.
- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the type of transformer K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/s
 - 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"
2. Scara with distance lathe spindle (X, Y, Z, C)
K-0-0051 = "mm/s,mm/s,mm/s,degree/s"
3. Cartesian robot with X, Y, Z, C, B
K-0-0051 = "mm/s,mm/s,mm/s,degree/s, degree/s"
4. Cartesian robot with X, Y, C
K-0-0051 = "mm/s,mm/s,degree/s"
5. Delta with C-axis and 2 belts
K-0-0051 = "mm/s,mm/s,mm/s,degree/s"



The belts are not included in the lists for the world coordinate system.

14.3.34 K-0-0052, Acceleration units WC

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.

RCL Parameters

- One entry (with comma separator) for each coordinate axis (x, y, z, alpha, beta, gamma),
- depending on the type of transformer K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/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"
2. Scara with distance lathe spindle (X, Y, Z, C)
K-0-0052 = "mm/s²,mm/s²,mm/s²,degree/s²"
3. Cartesian robot with X, Y, Z, C, B
K-0-0052 = "mm/s²,mm/s²,mm/s²,degree/s²,degree/s²"
4. Cartesian robot with X, Y, C
K-0-0052 = "mm/s²,mm/s²,degree/s²"
5. Delta with C-axis and 2 belts
K-0-0052 = "mm/s²,mm/s²,mm/s²,degree/s²"



The belts are not included in the lists for the world coordinate system.

14.3.35 K-0-0053, Jerk units WC

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 type of transformer K-0-0031 and the axes used K-0-0008.
 - For x, y, z-axes: mm/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"
2. Scara with distance lathe spindle (X, Y, Z, C)
K-0-0053 = "mm/s³,mm/s³,mm/s³,degree/s³"
3. Cartesian robot with X, Y, Z, C, B
K-0-0053 = "mm/s³,mm/s³,mm/s³,degree/s³, degree/s³"
4. Cartesian robot with X, Y, C
K-0-0053 = "mm/s³,mm/s³,degree/s³"
5. Delta with C-axis and 2 belts
K-0-0053 = "mm/s³,mm/s³,mm/s³,degree/s³"



The belts are not included in the lists for the world coordinate system.

14.3.36 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

14.3.37 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.

14.3.38 K-0-0071, Coupling factors, belt 1

Function The belt coupling factors specify how belt position changes of the first belt affect coordinate changes in the program sections between SYNCHRON and SYNCHRON_END.

Structure These parameter values are only important if at least a value of 1 is written in the parameter "K-0-0070, Belt number of kinematics".

This is a list parameter. One element in the list is responsible for each coordinate.

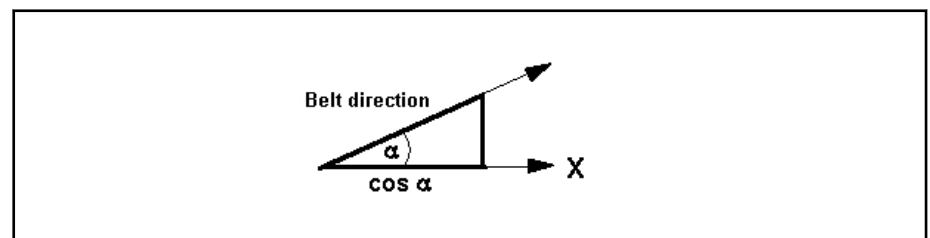


Fig. 14-22: Phase: Belt direction/axis

The value for a coordinate corresponds to the cosine value ($\cos \alpha$) of the phase between the belt and the respective coordinates.



Use the Windows calculator (View -> Scientific) with the setting DEG (not (new) Grads).

Use The assignment of the coordinates to the spatial direction does generally depend on the parameter "K-0-0031, Transformation scheme". Generally, the first coordinate is assigned to the x-direction, the second coordinate to the y-direction and the third coordinate to the z-direction. The following examples are based on this definition.

Example 1:

The belt is exactly parallel to the x-axis. Increasing belt values cause the part on the belt moving in positive x-direction.

K-0-0071 displays:

RCL Parameters

Element	Value	Comment
1	1.000000	Cos (0 degrees)
2	0.000000	Cos (90 degrees)
3	0.000000	Cos (90 degrees)

Example 2:

The belt is exactly parallel to the y-axis. Increasing belt values cause the part on the belt moving in negative y-direction.

K-0-0071 displays:

Element	Value	Comment
1	0.000000	Cos (90 degrees)
2	-1.000000	Cos (180 degrees)
3	0.000000	Cos (90 degrees)

Example 3:

The belt and the x-axis are in an angle of 30 degrees to each other; the belt and the y-axis in an angle of 60 degrees. Increasing belt values cause the part on the belt moving in positive x-direction and in positive y-direction.

K-0-0071 displays:

Element	Value	Comment
1	0.866025	Cos (30 degrees)
2	0.500000	Cos (60 degrees)
3	0.000000	Cos (90 degrees)



The root from the sum of the square of the cosine values has to be as exactly 1 as possible.

If the value is higher than 1, it corresponds to a length elongation in belt direction. If the value is smaller than 1, there is a length compression.

14.3.39 K-0-0072, Coupling factors, belt 2

Function The belt coupling factors specify how belt position changes of the second belt affect coordinate changes in the program sections between SYNCHRON and SYNCHRON_END.

These parameter values are only important if at least a value of 2 is written in the parameter "K-0-0070, Belt number of kinematics".

The parameter structure corresponds to the structure of the parameter "K-0-0071, Coupling factors, belt 1".

14.3.40 K-0-0073, Coupling factors, belt 3

Function The belt coupling factors specify how belt position changes of the third belt affect coordinate changes in the program sections between SYNCHRON and SYNCHRON_END.

These parameter values are only important if at least a value of 3 is written in the parameter "K-0-0070, Belt number of kinematics".

The parameter structure corresponds to the structure of the parameter "K-0-0071, Coupling factors, belt 1".

14.3.41 K-0-0074, Coupling factors, belt 4

Function The belt coupling factors specify how belt position changes of the fourth belt affect coordinate changes in the program sections between SYNCHRON and SYNCHRON_END.

These parameter values are only important if at least a value of 4 is written in the parameter "K-0-0070, Belt number of kinematics".

The parameter structure corresponds to the structure of the parameter "K-0-0071, Coupling factors, belt 1".

14.3.42 K-0-0080, Jogging, distance short in WC system

Function The short distance for the incremental jogging in the world coordinate system is stored in this parameter.



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

- Structure**
- List parameter:
One entry for every coordinate axis (e.g. x, y, z, alpha, beta, gamma),
 - Unit:
Depends on the transformation type K-0-0031 and the axes used (see K-0-0008).
The units are stored in K-0-0050.
Example:
for x, y, z in millimeters [mm] or inches [in] (see K-0-0054),
for alpha, beta, gamma in degrees.
 - Value range: >0.0 (floating point number)
 - Default: 1.0 mm or inch or degree.

Use The robot control uses this parameter for the incremental axis jogging in the world coordinate system.

The PLC program can use the jog bits (KinData[1].stJogCmd_q) to enable the incremental jogging in the world coordinate system (WC) with short distance:

E.g. incremental jogging in WC with short distance of the 3rd axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_ORIGIN_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SMALL;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The short distance to jog the axis is written in K-0-0080.
- The maximum velocity is written in K-0-0083 (always high for incremental jogging)
- The maximum acceleration is written in K-0-0084
- The maximum deceleration is written in K-0-0085
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

RCL Parameters

The jog motion can be prematurely completed by setting enJogMode = KIN_JOG_STOP.

Program:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.3.43 K-0-0081, Jogging, distance long in WC system

Function The long distance for the incremental jogging in the world coordinate system is stored in this parameter.



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

- Structure**
- List parameter:
One entry for every coordinate axis (e.g. x, y, z, alpha, beta, gamma),
 - Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0050.
Example:
for x, y, z in millimeters [mm] or inches [in] (see K-0-0054),
for alpha, beta, gamma in degrees.
 - Value range: >0.0 (floating point number)
 - Default: 10.0 mm or inch or degree.

Use The robot control uses this parameter for the incremental axis jogging in the world coordinate system.

The PLC program can use the jog bits (KinData[1].stJogCmd_q) to enable the incremental jogging in the world coordinate system (WC) with long distance:

E.g. incremental jogging in WC with long distance of the 3rd axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_ORIGN_COORD;  
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_LARGE;  
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The long distance to jog the axis is written in K-0-0081
- The maximum velocity is written in K-0-0083 (always high for incremental jogging)
- The maximum acceleration is written in K-0-0084
- The maximum deceleration is written in K-0-0085
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

The jog motion can be prematurely completed by setting enJogMode = KIN_JOG_STOP.

Program:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.3.44 K-0-0082, Jogging, velocity low in WC system

Function This parameter provides the low velocity for the continuous jogging in the world coordinate system.



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

- Structure**
- List parameter:
One entry for every coordinate axis (x, y, z, alpha, beta, gamma),
 - Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0051.
Example:
for x, y, z in mm/s or in/s (see K-0-0054),
for alpha, beta, gamma in degrees/s.
 - Value range: >0.0 (floating point number)
 - Default: 10.0 mm/s or in/s or degree/s.

Use The robot control uses this parameter for the continuous axis jogging in the world coordinate system.

The PLC program uses the jog bits (KinData[1].stJogCmd_q) to enable the continuous jogging in the world coordinate system (WC) with lower velocity:

E.g. continuous jogging in WC with low velocity of the 3. axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_ORIGIN_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SLOW;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The low velocity is written in K-0-0082
- The maximum acceleration is written in K-0-0084
- The maximum deceleration is written in K-0-0085
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

The continuous jogging can be completed with two different methods:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

and/or

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*no axis in pos.*)
```

14.3.45 K-0-0083, Jogging, velocity high in WC system

Function The high velocity for continuous jogging in the world coordinate system is stored in this parameter.

RCL Parameters



It can also be written on this parameter in phase 4. The new value is applied at the next jog command. It is not possible to change the parameter for the active jog command.

Structure

- List parameter:
One entry for every coordinate axis (x, y, z, alpha, beta, gamma).
- Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0051.
Example:
for x, y, z in mm/s or in/s (see K-0-0054),
for alpha, beta, gamma in degrees/s.
- Value range: >0.0 (floating point number)
- Default: 25.0 mm/s or in/s or degree/s.

Use

The robot control uses this parameter for the continuous axis jogging in the world coordinate system.

The PLC program uses the jog bits (KinData[1].stJogCmd_q) to enable the continuous jogging in the world coordinate system (WC) with higher velocity:

E.g. continuous jogging in WC with higher velocity of the 3. axis in positive direction:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_ORIGN_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_FAST;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. bit for 3.axis of kinematics or group*)
```

- The high velocity is written in K-0-0083
- The maximum acceleration is written in K-0-0084
- The maximum deceleration is written in K-0-0085
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

The continuous jogging can be completed with two different methods:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

and/or

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*no axis in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*no axis in pos.*)
```

14.3.46 K-0-0084, Jogging, acceleration in WC system

Function

The acceleration to jog (incremental and continuous) in the world coordinate system is stored in this parameter.



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

Structure

- List parameter:
One entry for every coordinate axis (x, y, z, alpha, beta, gamma),

- Unit:
Depends on the transformation type K-0-0031 and the axis, see K-0-0008.
The units are stored in K-0-0052.
Example:
for x, y, z in mm/s² or in/s² (see K-0-0054),
for alpha, beta, gamma in Grad/s².
- Value range: >0.0 (floating point number)
- Default: 370 mm/s² or in/s² or degree/s².

Use The robot control uses this parameter to jog (incremental and continuous) in the world coordinate system.

- The maximum deceleration is written in K-0-0085
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

Acceleration A-0-0084 and deceleration A-0-0085 are valid for axis jogging (JC).

14.3.47 K-0-0085, Jogging, deceleration in WC system

Function The deceleration to jog (incremental and continuous) in the world coordinate system is stored in this parameter.



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

Structure

- List parameter:
One entry for every coordinate axis (x, y, z, alpha, beta, gamma).
- Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0052.
Example:
for x, y, z in mm/s² or in/s² (see K-0-0054),
for alpha, beta, gamma in Grad/s².
- Value range: ≥ 0.0 (floating point number)
- Default: 370 mm/s² or in/s² or degree/s².

Use The robot control uses this parameter to jog (incremental and continuous) in the world coordinate system.

- The maximum acceleration is written in K-0-0084
- The maximum jerk for acceleration is written in K-0-0086
- The maximum deceleration jerk is written in K-0-0087

Acceleration A-0-0084 and deceleration A-0-0085 are valid for axis jogging (JC).

14.3.48 K-0-0086, Jogging, acceleration jerk in WC system

Function This parameter stores the maximum acceleration jerk to jog (incremental and continuous) in the world coordinate system (indirect definition of the acceleration change time).

RCL Parameters



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- List parameter:
One entry for each coordinate axis (x, y, z, alpha, beta, gamma)
 - Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0053.
Example:
for x, y, z in mm/s³ or in/s³ (see K-0-0054),
for alpha, beta, gamma in degree/s³.
 - Value range: ≥ 0.0 (floating point number)
 - Default: 0.0 (no jerk limitation).
- Use** The robot control uses this parameter to jog (incremental and continuous) in the world coordinate system.

$$\Delta a = j * t \quad \text{The acceleration time can be calculated from} \quad t = \frac{\Delta a}{j} \quad \text{or}$$

$$t = \frac{K-0-0084}{K-0-0086}$$

The maximum deceleration jerk in the world coordinate system is written in K-0-0087.

The jerk limit values for jogging axis-by-axis are specified in A-0-0086 and A-0-0087.

14.3.49 K-0-0087, Jogging, deceleration jerk in WC system

- Function** This parameter stores the maximum deceleration jerk to jog (incremental and continuous) in the world coordinate system (indirect definition of the deceleration change time).



With the MLC 04VRS, writing to the parameter is only possible in phase 2.

- Structure**
- List parameter:
One entry for every coordinate axis (x, y, z, alpha, beta, gamma).
 - Unit:
Depends on the transformation type K-0-0031 and the axis (see K-0-0008).
The units are stored in K-0-0053.
Example:
for x, y, z in mm/s³ or in/s³ (see K-0-0054),
for alpha, beta, gamma in degree/s³.
 - Value range: ≥ 0.0 (floating point number)
 - Default: 0.0 (no jerk limitation).

RCL Parameters

Use The robot control uses this parameter to jog (incremental and continuous) in the world coordinate system.

Off $\Delta a = j * t$ The deceleration time can be calculated from $t = \frac{\Delta a}{j}$ or

$$t = \frac{K-0-0085}{K-0-0087}$$

The maximum acceleration jerk in the world coordinate system is specified in K-0-0086.

The jerk limit values for jogging axis-by-axis are specified in A-0-0086 and A-0-0087.

14.3.50 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..31	Reserved

Use This parameter can only be used for Robot Control V2 and higher (see C-0-2484).

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.

14.3.51 K-0-0100, Current position, Cartesian original coordinates

Function The parameter indicates the current position of the kinematics in the original coordinate system.

Depending on the kinematic type, the current position consists of a list of Cartesian world coordinates, the tool orientation and the belt counters.

14.3.52 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).

14.3.53 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.

14.3.54 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.

RCL Parameters

The unit is mm/s³ or in/s³.

14.3.55 K-0-0105, Actual position, Cartesian workpiece coordinates

Function This parameter is reserved and has no functionality.

14.3.56 K-0-0106, Coordinate values of tool

Function The parameter indicates the coordinate values of the active tool, the three translations X, Y Z and the three rotations O1, O2, O3.

14.3.57 K-0-0107, Coordinate values of workpiece

Function This parameter is reserved and has no functionality.

14.3.58 K-0-0121 to K-0-0136, Coordinate Names

The parameters K-0-0121 ... 0136 contain the names of the individual coordinates. When an RCL program is created in the IndraWorks environment, the compiler statement applies

```
;;WC_NAMES
```

.

For uniform and detailed description, refer to "K-0-0121, Coordinate name C1".

14.3.59 K-0-0121, Coordinate name C1

Function This parameter contains the name of the first coordinate of the world coordinate system of the kinematics.



The following restrictions apply:

- maximum of 12 characters
- A...Z, a...z, 0..9, '_'
- first character must be a letter
- "K_1" is identical to "k_1"; upper-case and lower-case letters are not differentiated.

14.3.60 K-0-0155, Safe zones on/off

Function Structure This parameter specifies which of the 32 safe zones are enabled or disabled.

Bit	Content
0	Enabling the 1st safe zone
1	Enabling the 2nd safe zone
...	...
31	Enabling the 32nd safe zone

Fig.14-23: K-0-0155 Safe zones on/off

Use Parameter K-0-0155 can be directly modified in P4 (BB RUN). Switching to P2 and back to BB is not necessary.

When running a RCL program, it must still be remembered that enabling/disabling the safe zones can be delayed, since safe zones are checked during the block preparation.

Example:

K-0-0155: 0000 0000 0000 0000 0000 0000 0000 0100 ; the 3rd safe zone is enabled.

14.3.61 K-0-0156, Safe zone 1 min. point

- Function** This parameter determines the minimum point of the 1st safe zone.
A cuboid safe zone is determined with the maximum point from K-0-0157.
- Structure** The parameter is a list parameter with the following structure:
- Element 1: Position of the 1st axis
 - Element 2: Position of the 2nd axis
 - Element 3: Position of the 3rd axis
- Use** The parameter can be directly modified in P4 (BB RUN). Switching to P2 is not necessary.
- If an RCL program is running, note that the changes in the safe zone can be delayed since the safe zone check is executed in the block preparation.
- In addition, the maximum point has to be greater than the minimum point, since the safe zone check is otherwise disabled.

14.3.62 K-0-0157, Safe zone 1 max. point

- Function** This parameter determines the maximum point of the 1st safe zone.
A cuboid safe zone is determined with the maximum point from K-0-0156.
- Structure** The parameter is a list parameter with the following structure:
- Element 1: Position of the 1st axis
 - Element 2: Position of the 2nd axis
 - Element 3: Position of the 3rd axis
- Use** The parameter can be directly modified in P4 (BB RUN). Switching to P2 is not necessary.
- If an RCL program is running, note that the changes in the safe zone can be delayed since the safe zone check is executed in the block preparation.
- In addition, the maximum point has to be greater than the minimum point, since the safe zone check is otherwise disabled.

14.3.63 K-0-0156/K-0-0157 Safezone1 min./max. Point till K-0-0218/K-0-0219 Safezone32 min./max. Point

- Function:** Safe zones can define areas blocked for the robot motion. 32 safe zones are available for each kinematics.
- Functional description:** A motion command in the RCL program consists of a start point A and the programmed end point B.
- In the block preparation, a linear distance AB is interpolated from the points A and B. It is checked additionally, whether the distance AB has one intersection or one boundary point in common with one side of the safe zone. This is checked for each motion command and for all workspaces assigned to the kinematics.
- If the block preparation identifies a violation of a safe zone in the next block, the kinematics is traveled up to the next start point of the incorrect block and then, the RCL program is aborted. Additionally, the error message „F12C0433 Workspace violation“ is output.

RCL Parameters

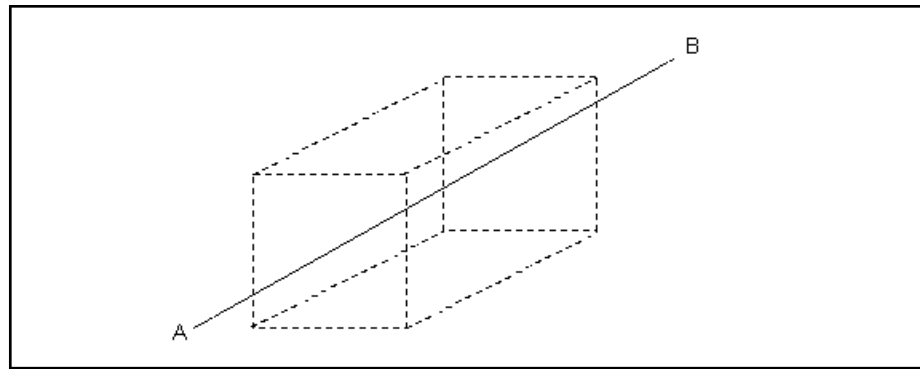


Fig. 14-24: Safe zone with minimum point (A) and maximum point (B)

Conditions: When programming the safe zones, the following has to be considered:

- Each kinematics considers only its individual safe zones.
- The safe zones consider only the TCP of the robot, 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 checking of the safe zones is restricted on the linear distance AB as well. The circular path and the programmed interpolation point are not checked.
- When blending with VIA or APPROX, deviations from the linear distance AB are not detected. Deviations due to blending radii are not considered.
- The safe zone monitoring is only effective in the automatic mode. The safe zones are not monitored in the manual mode.
- In case of a belt-synchronous traveling motion, the belt-synchronous motion part is not considered when checking the safe zones.
- Since the safe zones are checked in the block preparation, it is not possible to dynamically adapt the safe zones. A change of the safe zones becomes only effective if the block buffer was processed. Therefore, a change or activation / deactivation of a safe zone can delay when the RCL program is running.
- The maximum point must be bigger than the minimum point, since the safe zone check would become inactive otherwise.

Use: The activation and adaptation of the safe zone is executed via the parameters K-0-0155 to K-0-0219.

The K parameters can be switched via the respective PLC FBs or directly via the parameter editor to P4 (BB RUN).

Switching to P2 is not necessary.

The parameter K-0-0155 activates the checking of the safe zones (from bit 1 for 1st safe zone up to bit 32 for 32nd safe zone).

Example: K-0-0155 0000 0000 0000 0000 0000 0000 0100 $\equiv$ decimal: 4 $\Rightarrow$ Safe zone 3 is active.

The following parameters K-0-0156 to K-0-0219 can be used to specify the maximum and minimum points by which cuboids are defined as safe zones.

K-0-0155 0000 0000 0000 0000 0000 0000 0100 decimal: 4 $\Rightarrow$ Safe zone 3 is active.

K-0-0156 minimum point safe zone 1 (0, 0, 0),

RCL Parameters

K-0-0157 maximum point safe zone 1 (20, 10, 10),
K-0-0158 minimum point safe zone 2 (... , ... , ...),
K-0-0159 maximum point safe zone 2 (... , ... , ...),
....
K-0-0219 maximum point safe zone 32 (... , ... , ...).

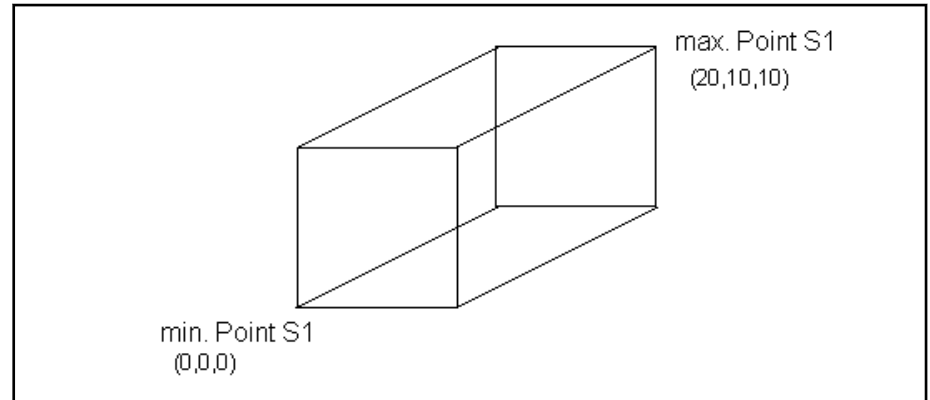


Fig. 14-25: Maximum and minimum point of safe zone 1

14.3.64 K-0-0220, File transfer name

Function The name of the RCL program is written in this parameter. It can be copied using the command parameters K-0-1060 and K-0-1061.

If the extension is `.*`, all files with the same program name - irrespective of the extension - are copied when executing one of the mentioned command parameters.

If the program name is specified as `*.*`, all files are copied when executing one of the mentioned command parameters.



The following restrictions apply:

- A maximum of 12 characters
- A...Z, a...z, 0..9, ' _ ' '*'
- The first character must be a letter
- "Robi_1" is identical to "ROBI_1"

14.3.65 K-0-0221, File name to be compiled

Function The name of the QLL file to be compiled is in this parameter.

A suffix, i.e. `.qll`, is automatically removed. The suffix `.QLL` is always added internally.

If a compiler process is started using the command parameter K-0-1062, the QLL file specified in K-0-0221 is compiled.



The following restrictions apply:

- a maximum of 12 characters with suffix
- a maximum of 8 characters without suffix
- only A...Z, a...z, 0..9, ' _ ' are permissible
- the first character must be a letter
- upper-case and lower-case letters are not differentiated, "Robi_1" and "ROBI_1" are identical

RCL Parameters

14.3.66 K-0-0240, Feedrate override

Function By means of this parameter, the target velocity can be changed as percentage of the runtime.

A value of 100% means that the axes are moved using the specified (programmed) velocity.

The percentage can be set in a range between 0% and 600%.



With 0%, the axes stop without changing the status of the kinematics status machine.

As long as the VFIX command is active in the RCL program, this parameter will have no effect

14.3.67 K-0-0241, Feedrate override control word

Function Using this parameter, the RoCoFeedrate override is controlled.
Structure

Bit	Designation/ function
0	reserved
1	Bit 1 = 0: RoCoFeed override in the K-0-0240 is effective, Bit 1 = 1: Feed halt, the value in K-0-0240 remains unchanged.
15..2	reserved

Use **Bit 1:**

By setting bit 1, the feed halt becomes effective.

14.3.68 K-0-0242, Acceleration override

Function By means of this parameter, the acceleration can be changed as percentage of the runtime.

A value of 100% means that the axes are moved using the specified (programmed) acceleration.

The percentage can be set in a range between 1% and 600%.



Setting of K-0-0242 takes effect in the block processing, i.e. not immediately.

As long as the AFIX command is active in the RCL program, this parameter will have no effect

14.3.69 K-0-0243, Deceleration override

Function By means of this parameter, the deceleration can be changed as percentage of the runtime.

A value of 100% means that the axes are moved using the specified (programmed) deceleration.

The percentage can be set in a range between 1% and 600%.



Setting of K-0-0243 takes effect in the block processing, i.e. not immediately.

Small values in K-0-0243 cause very long braking paths.

If the PROGR_SLOPE passing function is active in the RCL program, this may cause a jerky halt in case of a short braking path.

14.3.70 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.

14.3.71 K-0-1060, Command: Load file from CF

Function This command parameter copies the file(s) specified in parameter K-0-0220 from the Compact Flash card to the control's internal memory.
The file source directories "<User-Partition> / RC_Files" are specified on the Compact Flash.

- Use**
- Enter the binary value "00000000000000011" (=0x0003) in this parameter. This triggers the copying process.
 - After the copy process is complete, the control automatically resets the parameter to "0000000000000000".

14.3.72 K-0-1061, Command: Save file to CF

Function This command parameter copies the file(s) specified in parameter K-0-0220 from the control's internal memory to the Compact Flash card.
The file target directories "<User-Partition> / RC_Files" are specified on the Compact Flash.

- Use**
- Enter the binary value "00000000000000011" (=0x0003) in this parameter. This triggers the copying process.
 - After the copy process is complete, the control automatically resets the parameter to "0000000000000000".

14.3.73 K-0-1062, Command: Compile file to MLC

Function This parameter compiles the QLL file specified in parameter K-0-0221.



The QLL file to be compiled must be saved in the SRAM/RAM.

The QLL file is loaded in the SRAM/RAM using parameter K-0-1060.

- Use**
- Enter the binary value "00000000000000011" (=0x0003) in this parameter. This triggers the compile process.
 - After the compile process is complete, the MLC automatically resets the parameter to "0000000000000000".

14.3.74 K-0-2200, Programmed position 1 (active)

Function This parameter is reserved and has no functionality.

14.3.75 K-0-2201, Programmed position 2 (active)

Function This parameter is reserved and has no functionality.

RCL Parameters

14.3.76 K-0-2202, Type of actively programmed position

Function This parameter is reserved and has no functionality.

14.3.77 K-0-2203, Interpolation type

Function This parameter is reserved and has no functionality.

14.3.78 K-0-2204, Currently active QLL file name

Function This parameter is reserved and has no functionality.

14.3.79 K-0-2205, Currently active line number

Function This parameter is reserved and has no functionality.

14.3.80 K-0-2300, PLC-Bit input (RCL)

Function The parameter indicates the bit inputs that are transferred from RCL toward the PLC.

14.3.81 K-0-2301, PLC-Integer input (RCL)

Function The parameter indicates the integer inputs that are transferred from RCL toward the PLC.

14.3.82 K-0-2302, PLC-Real input (RCL)

Function The parameter indicates the floating point inputs that are transferred from RCL toward the PLC.

15 Kinematic User Interface

15.1 General Information

Predefined kinematics (Cartesian, Scara, Deltapack or user-defined kinematics) can be created in the "Robot" folder.

16 kinematics with the following maximum number of axes (including auxiliary axes (belt) and orientation axes) are possible:

- 16 axes for an IndraMotion MLC L25
- 32 axes for an IndraMotion MLC L45
- 64 axes for an IndraMotion MLC L65

The user-defined kinematics can only be selected and parameterized via an assigned kinematic number.

The screenshot shows a Windows-style dialog box titled "Insert Kinematics". Inside, there's a section "Step 2: Configuration" with the instruction "Insert the kinematics type". On the left is a large empty box labeled "Kinematics type". On the right, there are input fields: "Name" with "Kin1", "Number" with "1", "Type" with a dropdown menu showing "User defined", and "Transformation Number" with "1000". At the bottom are four buttons: "<< Back", "Next >>", "Cancel", and "Help".

Fig. 15-1: Options for user-defined kinematics

Programming environment

RobotControl is the programming environment for the Bosch Rexroth robot family. The advantages are:

- Faster implementation of robot applications
- Visualization of the implementation
- Diagnostics of the implementation

Characteristics

The programming environment contains the following components:

- **User interface**

The user interface is oriented at the Windows applications and thus allows the user to work intuitively and effectively.

All RobotControl components are integrated into the interface and can be optionally shown or hidden. All toolbar icons are provided with help

Kinematic User Interface

texts. These are displayed when holding the mouse cursor on the icon for a moment

Additionally, information on the respective functions is shown in the status bar

- **Text editor**

The text editor is used to create and check user programs in the programming language RCL (**R**obot **C**ontrol **L**anguage). The text editor can also be used for any other ASCII text file

- **Compiler and Debugger**

The compiler compiles a source code to a machine-readable code. The output texts and the error messages are redirected to an output window of the RobotControl and displayed. Double-click to jump to the position in the program.

The debugger facilitates the finding of program errors. The values of previously declared variables can be displayed in the debug mode and monitored during the program execution

- **Point editor**

The point editor enters, deletes or modifies points. The points are loaded from the point file (online or offline) and represented in a table. Points can also be declared as point arrays

- **File transfer**

Files can be transferred to the control and deleted in the control

- **Control diagnostics**

All relevant status information of the respective kinematics is displayed in the "Status" dialog

15.2 Menu Structure of the Kinematics

15.2.1 Menu Item: Robot

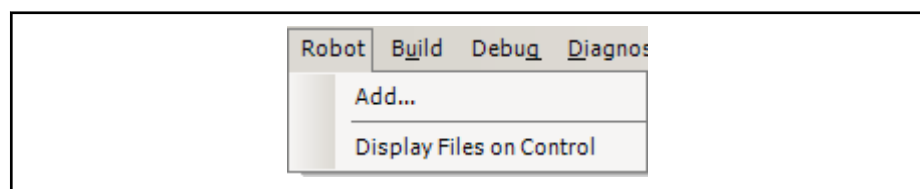


Fig. 15-2: Menu Item: Robot

Select the "Robot" node to enable the menu item **Robot** in the IndraWorks main menu. **Add...** starts the wizard to configure a new kinematics and the folder for "Global Data" can be created. Display the RCL programs in the control via **Display Files on Control**. Open the previously described functions also via the context menu of the "Robot" node.

15.2.2 Menu item: Kinematics

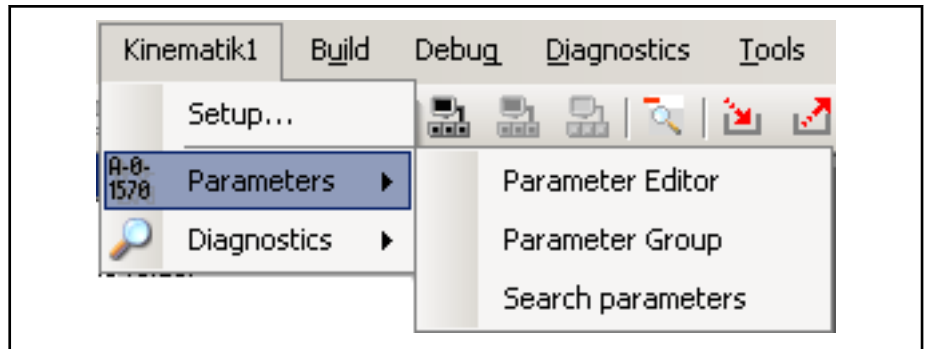


Fig. 15-3: Menu Item of a Kinematics (in the example "Kinematik1")

The menu item (in the example **Kinematik1** in the IndraWorks main menu) is activated by selecting the a node of a kinematics. The name of the respectively created selected kinematics is displayed as name in the menu item or node.

15.2.3 Menu Item "Global Data"

Select the "Global Data" node to enable the menu item **Global Data** in the IndraWorks.

Click on **Add...** to create the following global data:

- Workpiece coordinates
- Tool management
- Library files

15.2.4 Menu item: "Programs"

Select the "Programs" node to enable the menu item **Programs** in the IndraWorks main menu. The following menu items are available:

- **Add...** - Creates a new RCL program in the editor
- **Open External File...** - Opens an RCL program outside the project in the editor.
- **Upload from MLC** - Loads a selected RCL program from the control to the project
- **Download to MLC** - Downloads the RCL programs to the control

15.2.5 Menu item of a Program

Select a program in the project (below the node /<KinematicName>/ Programs) to enable the menu item of the program (in the example **Prog1.qll**) in the IndraWorks main menu. The following menu items are available:

- **Compile** - Creates an executable robot program from the source code (.QLL)
- **Open** - Opens the selected RCL program in the editor
- **Point File** - Opens and saves a point file
- **Upload from MLC** - Loads a selected RCL program from the control to the project
- **Download to MLC** - Downloads the RCL programs to the control

Kinematic User Interface

15.2.6 Menu Item "Debug"

Select an open RCL program in the RCL editor to enable the **Debug** menu item in the IndraWorks main menu. The following menu items are available:

- **Login** - Establish online mode
- **Logout** - Exits the online mode
- **Single Step** - Executes current instruction
- **Debug Step** - Executes current instruction, skips subroutines
- **New breakpoint...** - Sets new breakpoint
- **Toggle Breakpoint** - Switches on/off breakpoint
- **Delete All Breakpoints** - Deletes all breakpoints
- **Activate/Deactivate Breakpoint** - Enables/disables breakpoint

15.3 Global Data – Context Menu

15.3.1 Overview

The "Global Data" context menu includes the following menu items:

- **Add**
 - **Workpiece Coordinates** - Creates and edits the workpiece coordinate file
 - **Tool Management** - Creates and edits the tool management file
 - **Library File** - Creates and edits RCL library files
- **Export...** - Save "Robot" elements in a file
- **Import...** - Import robot elements from a file
- **Compare...** - can compare and merge objects
- **Delete** - Deletes the folder "Global Data"
- **Find element...** - Searches for elements in the project

15.3.2 Workpiece Coordinates

General Information

Use the "workpiece coordinate system" to define any world coordinate systems.

Workpiece Coordinate File

Define all local workpiece coordinates by name in the "WCSYST.DAT" file.

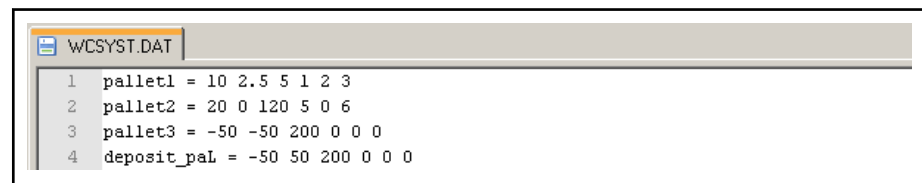


Fig.15-4: Example for WCSYST.DAT (opened in text editor)

The name of the respective world coordinate has to start in the first column and may not exceed 12 characters.

Workpiece Coordinate Editor

Use the editor to create, edit and delete the workpiece coordinates.

WCSYST.DAT*						
Workpiece Coordinates						
Name	X	Y	Z	01	02	03
pallet1	10	2.5	5	1	2	3
pallet2	20	0	120	5	0	6
pallet3	-50	-50	200	0	0	0
deposit_pal	-50	50	200	0	0	0

Fig. 15-5: WCSYST.DAT dialog

The specified data sets are stored in the "WCSYST.DAT" file in ASCII format. Close the dialog for saving. By default, the storage location is the project.

15.3.3 Tool Management

General Information

If different grippers or tools are used during machining, the individual gripper parameters are stored in the "TOOLS.DAT" file.

Tool management file

Three translations (G_X, G_Y, G_Z) and three rotations (G_D1, G_D2, G_D3) each referring to the tool center point (TCP) are used for a general description.

TOOLS.DAT	
1	gripper1 = 10 2.5 5 1 2 3
2	gripper2 = 20 0 120 5 0 6
3	gripper3 = 0 0 0 0 0 0
4	OFF = 0 0 0 0 0 0

Fig. 15-6: Example for TOOLS.DAT (opened in text editor)

The sequence of the individual coordinates (G_X, G_Y, to G_D3) must not be changed. Zeros have to be set for missing values. The gripper name has to start in the first column and may not exceed 12 characters.

Tool management editor

Use the editor to create, edit and delete the tool data sets.

TOOLS.DAT						
Tool agreement						
Name	X	Y	Z	01	02	03
gripper1	10	2.5	5	1	2	3
gripper2	20	0	120	5	0	6
gripper3	0	0	0	0	0	0
OFF	0	0	0	0	0	0

Fig. 15-7: TOOLS.DAT dialog

The specified data sets are stored in the "TOOLS.DAT" file in ASCII format. Close the dialog for saving. By default, the storage location is the project.

15.3.4 Library files

Global RCL libraries can be created and edited.

15.4 Kinematics - Context menu

15.4.1 Overview

The context menu of a kinematics comprises the following menu items:

Kinematic User Interface

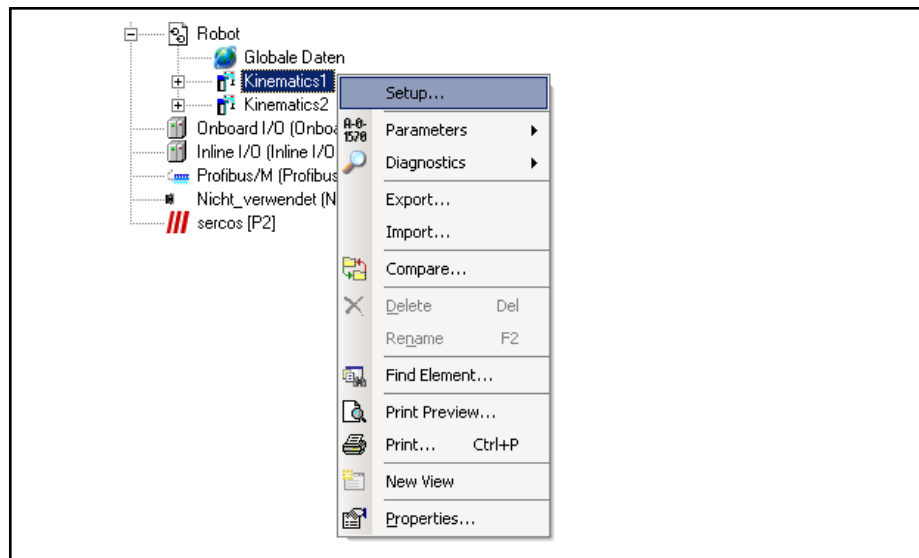


Fig. 15-8: Context menu of a kinematics in Online mode

- **Setup...** - Commissioning cycle of the kinematics, see [Setup, page 214](#)
- **Parameter** - Edits parameters
- **Diagnostics** - Diagnostic functions of the kinematics, see [Diagnostics, page 214](#)
- **Export...** - Save "Robot" elements in a file
- **Import...** - Import robot elements from a file
- **Compare...** - can compare and merge objects
- **Find element...** - Searches for elements in the project
- **Print preview...** - Provides a preview of the kinematic data
- **Print...** - Allows the expression of the current kinematic data
- **New view** - Displays the kinematics in an individual window
- **Properties...** - Provides information on the kinematics (such as name or type)

15.4.2 Setup

To commission a kinematics for the first time, make different settings using a wizard (see [chapter 15.5.1 "Overview on dialogs" on page 216](#)).

15.4.3 Diagnostics

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

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

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

Kinematic User Interface

The screenshot shows the Kinematic User Interface for 'Kinematik 1'. At the top, it displays 'Kinematics number: 1', 'Kinematics name: Kinematik 1', and 'Kinematics type: Carthesis'. Below this, the 'Status' section indicates 'Kinematik ist in "Standstill Incomplete"'. A large table area is currently empty. To the right, the 'Coordinate system' section has two radio buttons: 'Joint coordinates (JC)' (selected) and 'World coordinates (WC)'. Below this is a 'Details >>' button. The 'Status word' section shows 'Standstill is not complete'. There are two columns of status indicators: 'Power' (Axes ready, Feed enable) and 'Errors/Warnings' (Warning exists, Error exists). On the far right, a 'Messages' section lists several options: Automatic, Manual, Target position reached, Block processing enabled, and Error reaction is active.

Fig. 15-9: Diagnostic of a kinematics

Coordinate system

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

There are two settings:

- **JC** = Joint coordinates - They describe the individual axis positions in [°] (rotary axes) or [mm] (linear axes)

Axis No.	Axis name	Position[°]	End position[°]	InPos	Lag[°]
1	A01	0	0	☐	0
2	A02	0	0	☐	0

Fig. 15-10: Display in joint coordinates

- **WC** = World coordinates - They describe a freely definable local world coordinate system, also called workpiece coordinate system

Default setting: Coordinate origin at the base point of the robot.

Axis No.	Axis name	Position[mm]	End position[mm]	InPos	Lag[mm]
1	X1	2	0	☐	0
2	Y1	0	0	☐	0

Fig. 15-11: Display in world coordinates

List of axis coordinates

The list of the axis coordinates shows

- Axis number
- Axis name
- Axis position
- End point
- Lag

An enabled checkbox in the "InPos" provides information whether the respective axis is "in Position".

Kinematic User Interface

15.5 Kinematics – Dialogs

15.5.1 Overview on dialogs

When opening the folder of the kinematics in the Project Explorer, individual dialogs - display of data from the kinematics and adjustment possibilities for kinematic parameters - are provided. These dialogs are disabled at first and enabled when the project is switched to online mode.

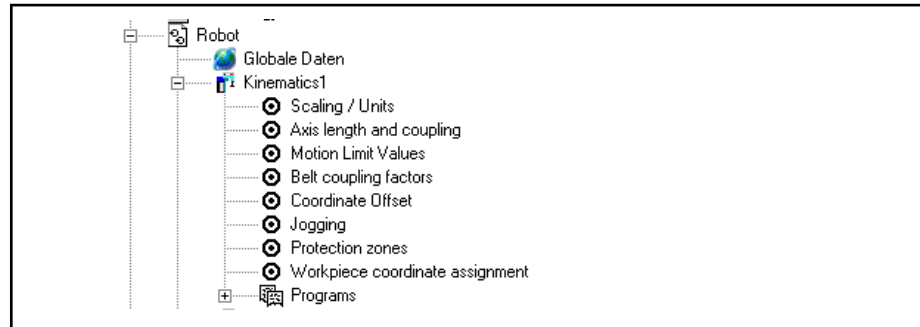


Fig. 15-12: Kinematic dialogs

- [Scaling/Units, page 216](#)
- [Axis Lengths and Coupling, page 216](#)
- [Motion Limit Values, page 217](#)
- [Belt Coupling Factors, page 218](#)
- [Coordinate Offset, page 219](#)
- [Jogging, page 219](#)
- [Safe Zones, page 221](#)
- [Workpiece Coordinate Assignment, page 223](#)
- [Creating New RCL Program, page 224](#)

15.5.2 Scaling/Units

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

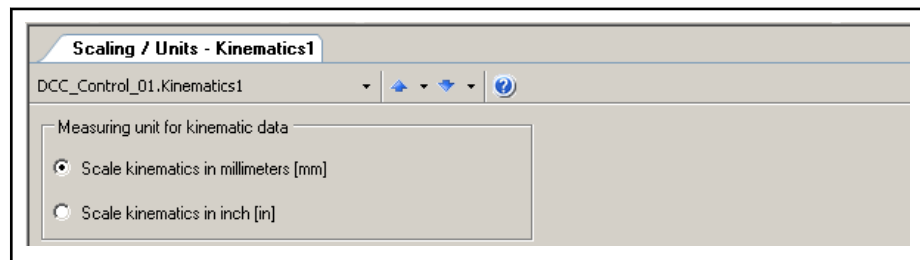


Fig. 15-13: Toggling units

The following parameter is evaluated:

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

15.5.3 Axis Lengths and Coupling

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

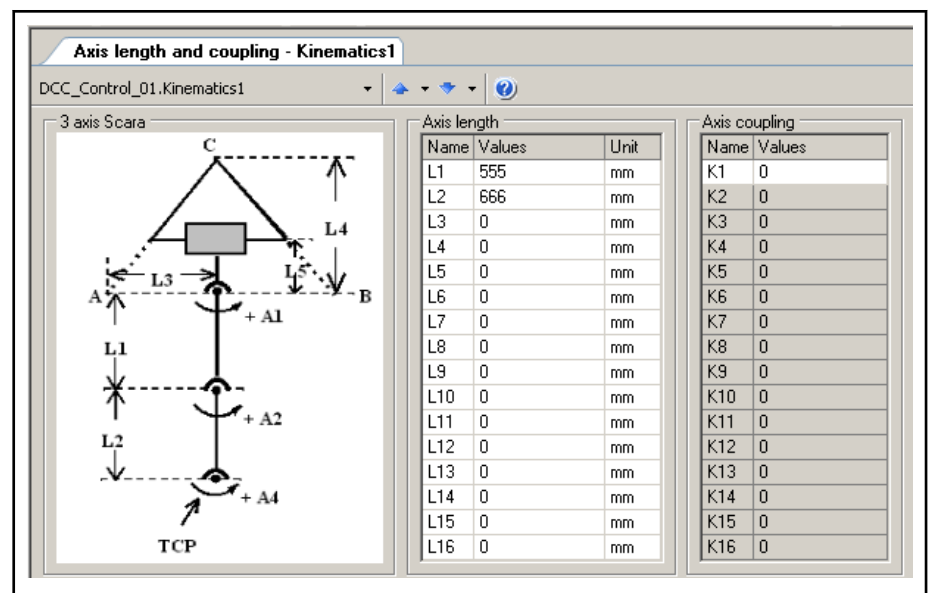


Fig. 15-14: Defining axis lengths and axis couplings

The following parameters are evaluated:

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

15.5.4 Motion Limit Values

The following limit values can be entered:

- Maximum path velocity
- Path acceleration, modification time
- Path deceleration, modification time
- Axis monitoring
- Software limit switch, positive WC
- Software limit switch, negative WC

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

Kinematic User Interface

	Positive	Negative
Limit value 1	1 mm	-1 mm
Limit value 2	1 mm	-1 mm
Limit value 3	1 ..	-1 ..
Limit value 4	1 ..	-1 ..
Limit value 5	1 ..	-1 ..
Limit value 6	1 ..	-1 ..
Limit value 7	1 ..	-1 ..
Limit value 8	1 ..	-1 ..

Fig.15-15: Overview on motion limit values

The following parameters are evaluated:

- K-0-0003, Maximum path velocity
- K-0-0014, Path acceleration, change time
- K-0-0015, Path deceleration, change time
- K-0-0036, WC axis monitoring
- K-0-0005, Positive software limit switch WC
- K-0-0006, Negative software limit switch WC

15.5.5 Belt Coupling Factors

The belt coupling factors specify how belt position changes of the respective belt affect coordinate changes in the program sections between SYNCHRON and SYNCHRON_END.

The dialog box provides an overview on the preset belt coupling factors of the kinematics.

Belt	Value 1 (cos α)	Value 2 (cos β)	Value 3 (cos γ)
Belt1	0	0	0
Belt2	0	0	0
Belt3	0	0	0
Belt4	0	0	0

Fig.15-16: Overview on the belt coupling factors

The following parameters are evaluated:

- K-0-0071, Coupling factors, belt 1

- K-0-0072, Coupling factors, belt 2
- K-0-0073, Coupling factors belt 3
- K-0-0074, Coupling factors, belt 4

15.5.6 Coordinate Offset

With this dialog, the coordinate system origin of the transformation schema can be shifted in order to work with the coordinate values corresponding to a reference point in the editing cell.

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

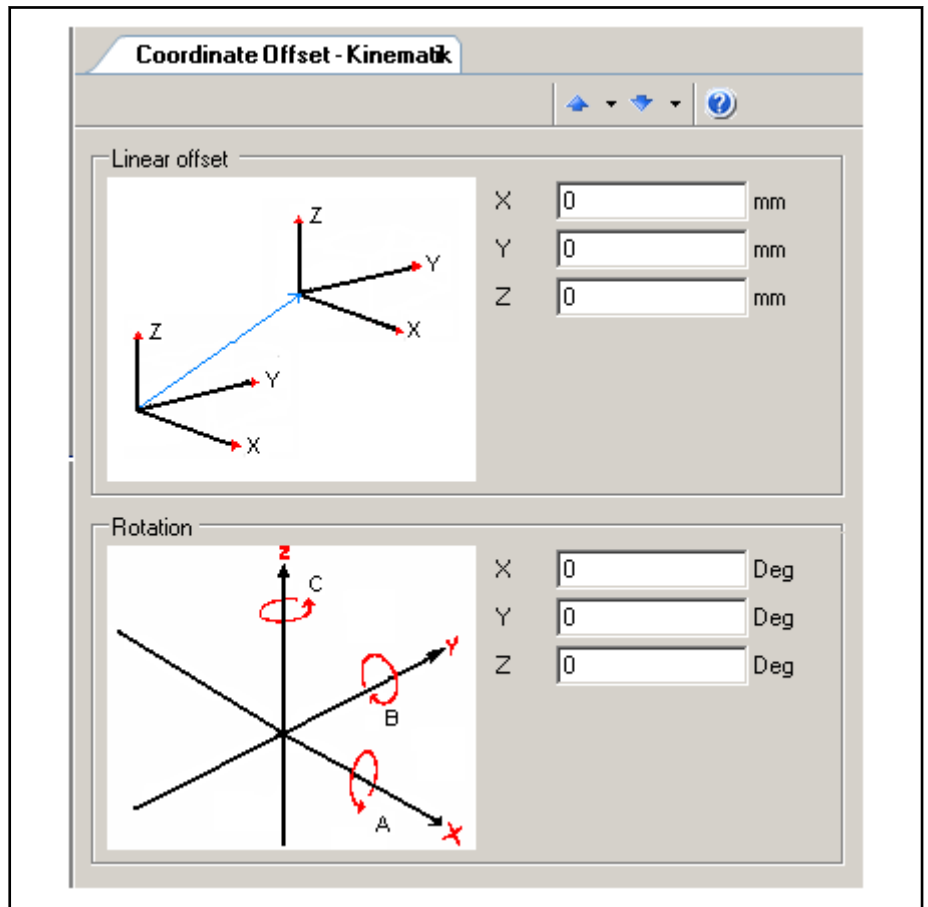


Fig. 15-17: Overview on the coordinate offset

The following parameter is evaluated:

- K-0-0042, World coordinate system offset

15.5.7 Jogging

In this dialog, the velocities settings for the manual operation can be made. The settings are made individually for each coordinate (axis).

Kinematic User Interface

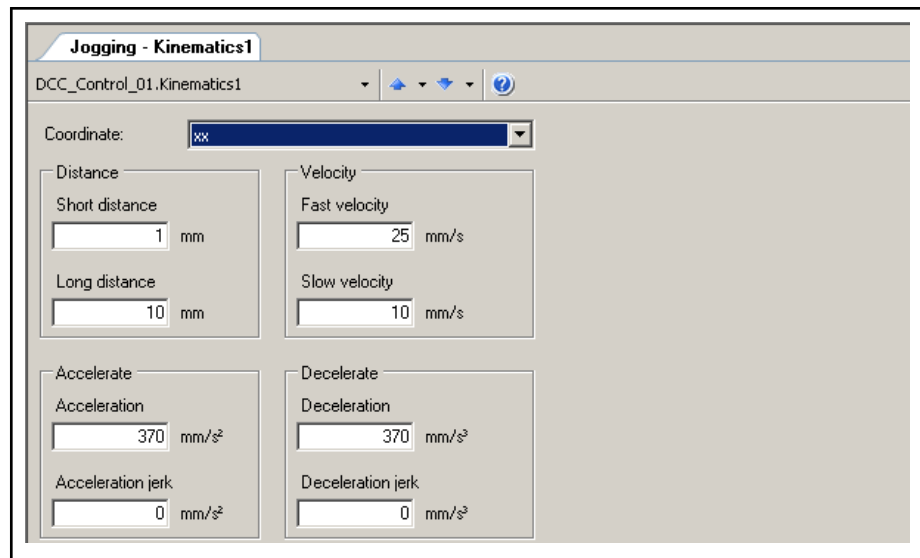


Fig. 15-18: Overview on jogging

The following parameters are evaluated:

Coordinates

- Selection of coordinates (axis) for which the corresponding jog values are to be displayed or changed.

Distance

- K-0-0080, Jogging, distance short in WC system
This parameter provides the short distance for the incremental jogging in the world coordinate system
- K-0-0081, Jogging, distance long in WC system
This parameter provides the long distance for the incremental jogging in the world coordinate system

Velocity

- K-0-0082, Jogging, velocity low in WC system
This parameter provides the low velocity for continuous jogging in the world coordinate system
- K-0-0083, Jogging, velocity high in WC system
This parameter provides the fast velocity for continuous jogging in the world coordinate system

Acceleration

- K-0-0084, Jogging, acceleration in WC system
The acceleration for jogging (incremental and continuous) in the world coordinate system is stored in this parameter
- K-0-0086, Jogging, acceleration jerk in WC system
The maximum acceleration jerk for jogging (incremental and continuous) in the world coordinate system is stored in this parameter (indirect definition of the acceleration change time)

Deceleration

- K-0-0085, Jogging, deceleration in WC system
The deceleration for jogging (incremental and continuous) in the world coordinate system is stored in this parameter.
- K-0-0087, Jogging, deceleration jerk in WC system

The maximum deceleration jerk for jogging (incremental and continuous) in the world coordinate system is stored in this parameter (indirect definition of the deceleration change time)

15.5.8 Safe Zones

This dialog allows setting the safe zones. Safe zones are areas blocked for the robot motion.

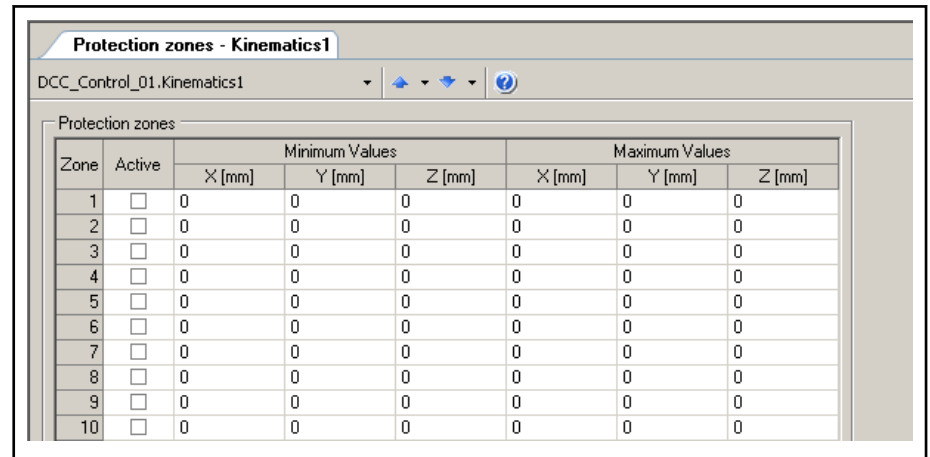


Fig. 15-19: Specifying the safe zones

The following parameters are evaluated:

- K-0-0155, Safe zone on/off
- K-0-0156, Safe zone 1 min. point
- K-0-0157, Safe zone 1 max. point

The parameter for the safe zones 2 - 32 are numbered consecutively. A maximum of 32 safe zones can be parameterized for each kinematics.

Functional description:

A motion command in the RCL program consists of a starting point A and the programmed end point B.

In the block preparation, a linear distance AB is interpolated from the points A and B. It is checked additionally, whether the distance AB has one intersection or one boundary point in common with one side of the safe zone. This is checked for each motion command and for all workspaces assigned to the kinematics.

If the block preparation identifies a violation of a safe zone in the next block, the kinematics is traveled up to the next starting point of the incorrect block and then, the RCL program is aborted. The error message "F12C0433 Workspace violation" is also output.

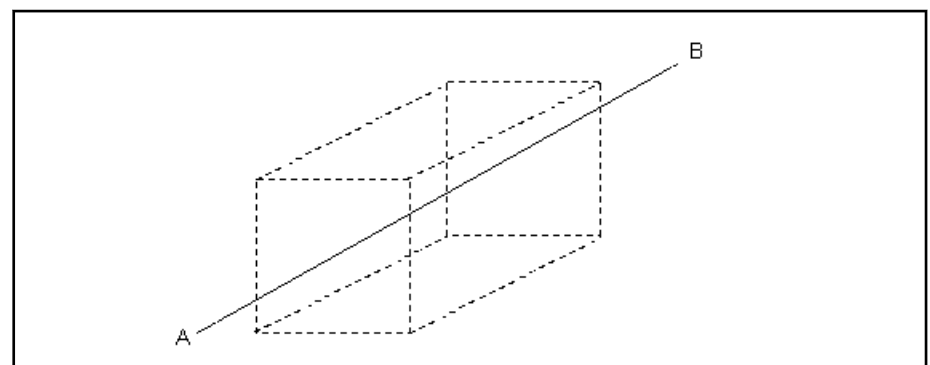


Fig. 15-20: Safe zone with minimum point (A) and maximum point (B)

Kinematic User Interface

- Conditions:** When programming the safe zones, the following has to be considered:
- Each kinematics considers only its individual safe zones
 - The safe zones consider only the TCP of the robot, 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 checking of the safe zones is restricted on the linear distance AB as well. The circular path and the programmed interim point are not checked
 - When blending with VIA or APPROX, deviations from the linear distance AB are not detected. Deviations due to blending radii are not considered
 - The safe zone monitoring is only effective in Automatic mode. The safe zones are not monitored in Manual mode
 - In case of a belt-synchronous travel motion, the belt-synchronous motion part is not considered when checking the safe zones
 - Since the safe zones are checked in the block preparation, safe zones cannot be dynamically adapted. A change of the safe zones becomes only effective if the block buffer was processed. Therefore, a change or activation / deactivation of a safe zone can delay when the RCL program is running
 - The maximum point has to be bigger than the minimum point to ensure that the safe zone check is not active
- Use:** The activation and adaptation of the safe zone is executed via the parameters K-0-0155 to K-0-0219.
- The K-parameters can be switched via the respective PLC functions blocks or directly via the parameter editor to P4 (BB RUN). Switching to P2 is not required.
- The parameter K-0-0155 activates the checking of the safe zones (from bit 1 for 1st safe zone up to bit 32 for 32nd safe zone).
- The following parameters K-0-0156 to K-0-0219 can be used to specify the maximum and minimum points by which cuboids are defined as safe zones.
- Example:** K-0-0155 0000 0000 0000 0000 0000 0000 0100 $\equiv$ decimal: 4 $\Rightarrow$ safe zone 3 is active.
- K-0-0156 minimum point of safe zone 1 (0, 0, 0)
- K-0-0157 maximum point of safe zone 1 (20, 10, 10)
-
- K-0-0219 maximum point of safe zone 32 (..., ..., ...)

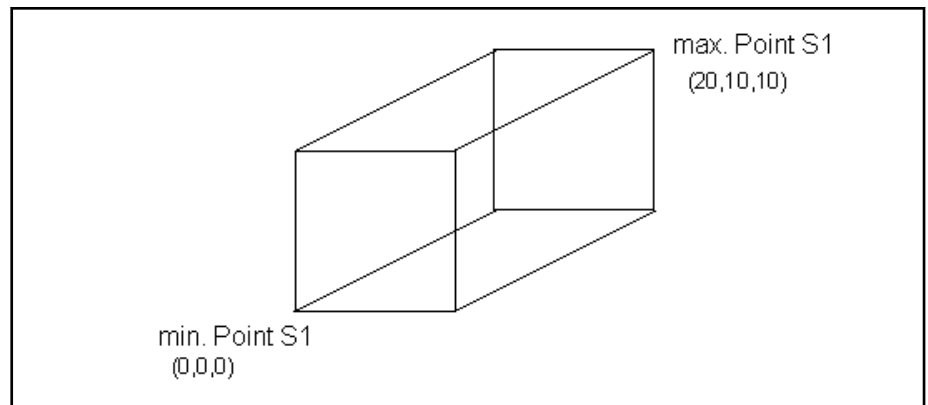


Fig. 15-21: Maximum and minimum point of safe zone 1

15.5.9 Workpiece Coordinate Assignment

This dialog allows assigning axis coordinates to workpiece coordinates.

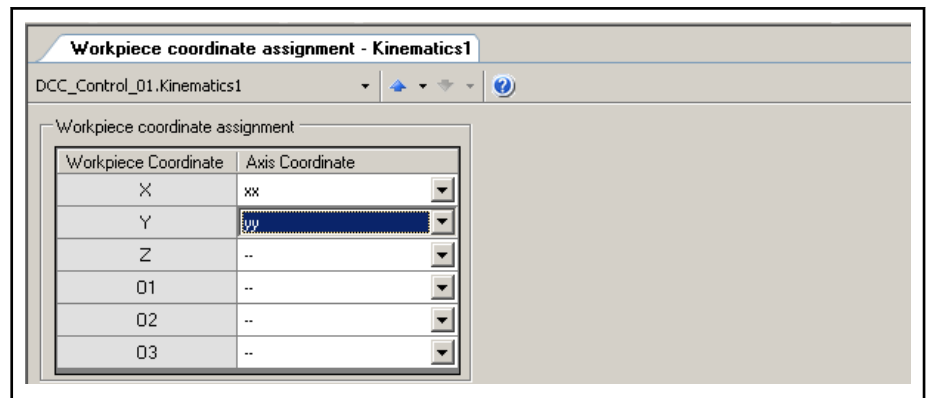


Fig. 15-22: Specifying the workpiece coordinate assignment

The following parameter is evaluated:

- K-0-0043, Assignment list of workpiece coordinates

15.6 RCL Editor

15.6.1 General Information

The text editor is used to create, check and debug user programs in the programming language RCL (**R**obot **C**ontrol **L**anguage).

Options can be used to set the behavior, appearance, colors, line numbering, tabulator widths, indentations, etc. (see [chapter 15.10 "Settable Options for RoCo" on page 232](#)).

15.6.2 Characteristics

Special editor characteristics:

- Syntax highlighting (can be freely configured in the "Options" dialog)
- Copy, Paste, Find, Replace, etc.
- Bookmark management
- Line numbering
- Tooltip
- Automatic keyword specification

Kinematic User Interface

- Code folds
- Intellisense
- Code templates
- Debugging

15.6.3 Creating New RCL Program

Create a new RCL program via the following menus:

- In the IndraWorks main menu via **Programs ► Add** or
- via the program context menu item **Add...** of the kinematics

Depending on the execution of the respective menu item, the specified program name can be overwritten by default. If the entry is confirmed with **OK**, the RCL Editor is started containing the program body of the new RCL program.

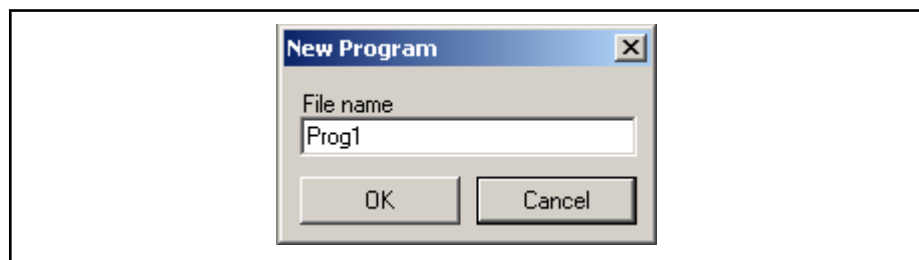


Fig. 15-23: "Add New Program" dialog

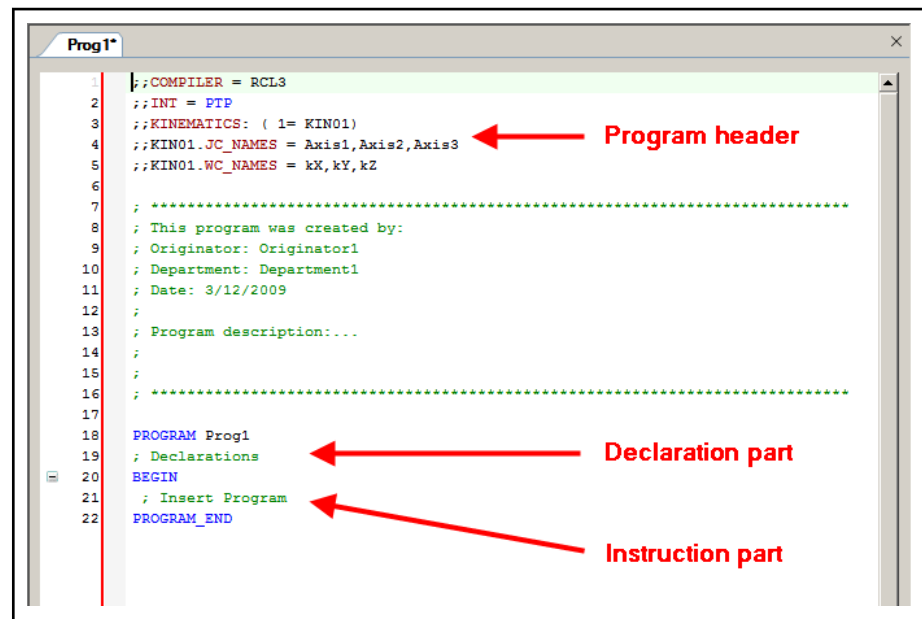


Fig. 15-24: General structure of a RoCo program

15.6.4 Working with the Editor

General editor functions

The RoCo editor integrated the following useful techniques of common Windows editors:

- Find
- Find files
- Replace

- Block highlighting
- Match case
- Drag&drop

Find and Replace

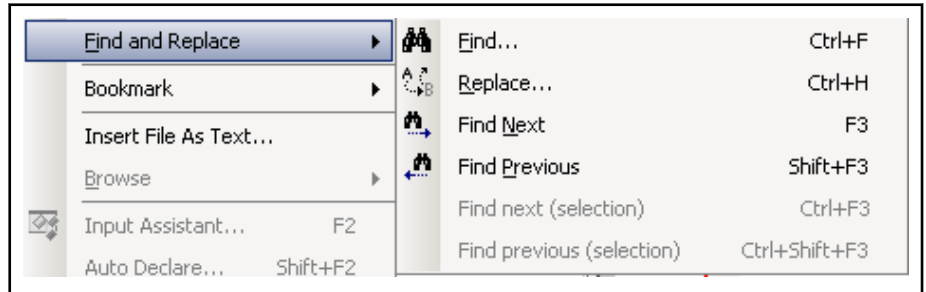


Fig. 15-25: Menu item: "Find and Replace"

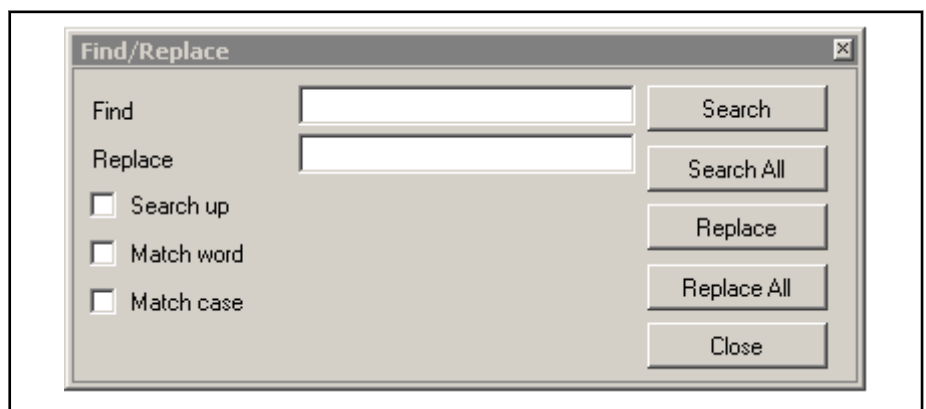


Fig. 15-26: Dialog "Find/Replace"

Bookmark

The text editor provides bookmark management. Add bookmarks to lines to facilitate navigation in long programs. The commands described below allow the setting and navigation of bookmarks. They are included by default in the "Bookmarks" submenu in the "Edit" menu. Additionally, they can be found via "Bookmarks" in the toolbar.

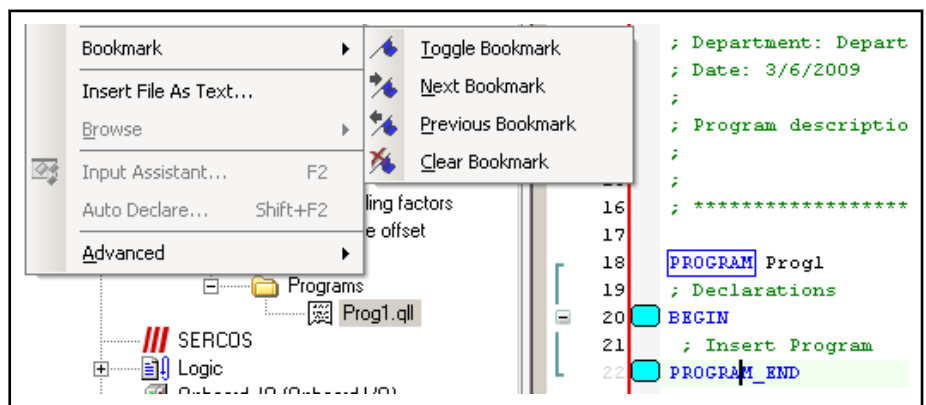


Fig. 15-27: Menu item: "Bookmark"

Comments

Comments can be inserted. The comment starts with ";" and ends at the end of the line.

Cut, Copy, Paste

Kinematic User Interface

Highlight the area in the editor with the left mouse button or via the Shift cursor shortcut. The commands "Cut", "Copy", "Paste" can then be used in these areas.

Find the commands in the "Edit" menu or via the context menu of the editor.

Supporting the source text input

- Parameter information**

If the mouse pointer is set on a subroutine, the transfer parameters are displayed.

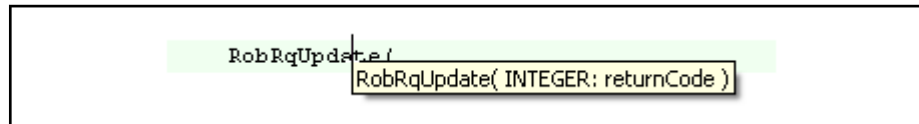


Fig. 15-28: Parameter information

If a subroutine name and an open parenthesis are entered, the transfer parameters are also displayed

- Code completion**

This function supplements the user input. The input options reduce with the progress and might also with the semantics of the user input. Enter the first character or press <CTRL> + <Space> and a selection list is shown containing the RoCo command block as well as the already declared variables. The object selected in the list is applied with the <Return> key (also refer to [fig. 15-51 "Auto completion" on page 236](#))

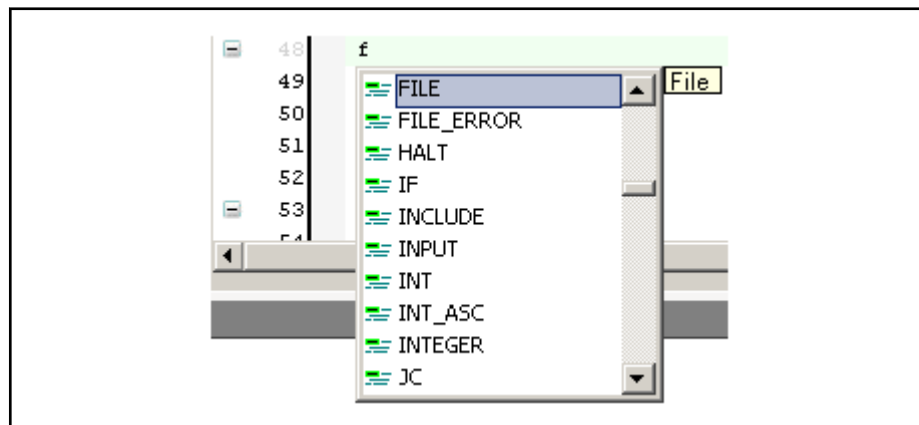


Fig. 15-29: Code completion

- Element listing**

Enter a link variable and a subsequent point and the elements of the link variables are displayed for selection. Select the desired element using the arrow keys. Apply the selection with <Return>

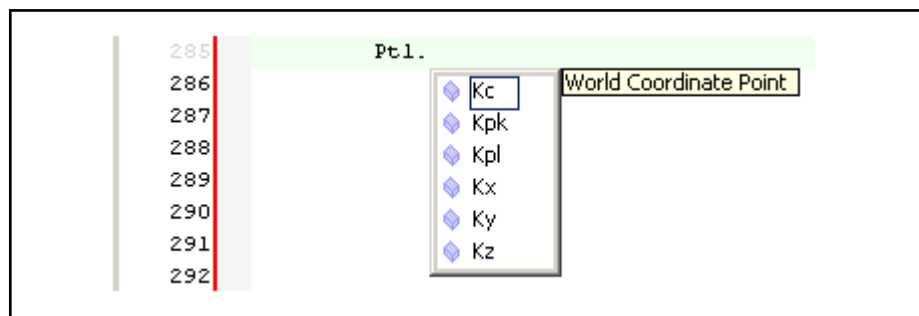


Fig. 15-30: Element listing

- **Type information**

Move the mouse cursor across the variable and the type information is shown as information.

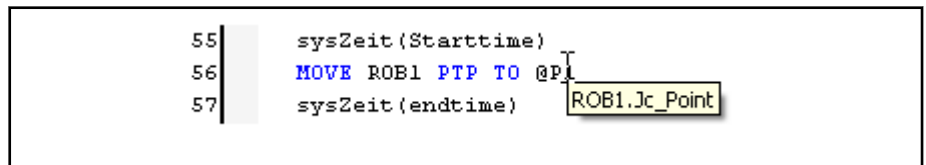


Fig.15-31: Type information

- **Indenting the cursor automatically**

When writing a source text, the indentation of the previous line is automatically applied to the next line

- **Tabulators**

The length of the tabulator jumps can be set in the "Options" dialog (see [fig. 15-50 "Text editor settings" on page 235](#)).

- **Syntax highlighting**

The keywords in the source text are highlighted. The default specifications can be changed in the "Options" dialog (see [fig. 15-66 "Keyword settings" on page 242](#))

- **Special functions**

Entering special functions for the code completion is supported.

Enter the RoCo command for special functions and it is selected in the drop-down list.

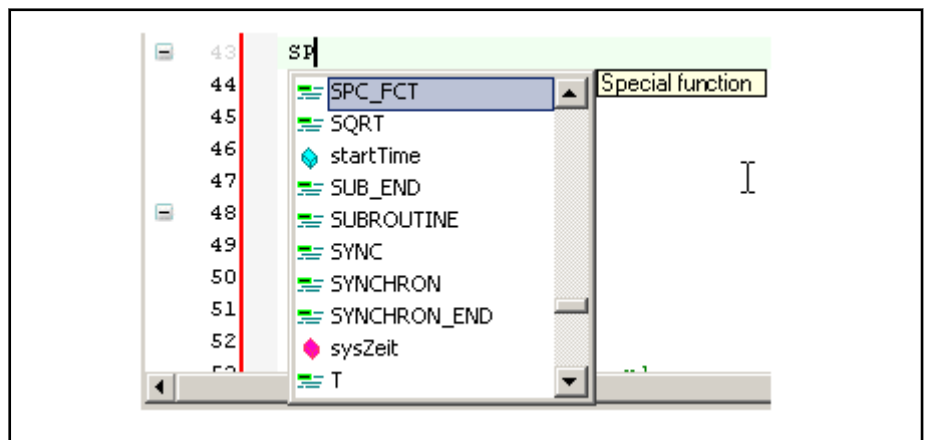


Fig.15-32: Selecting the "Special Function" command

Apply the object selected in the list with <Return>. A drop-down list displays the special functions available for selection.

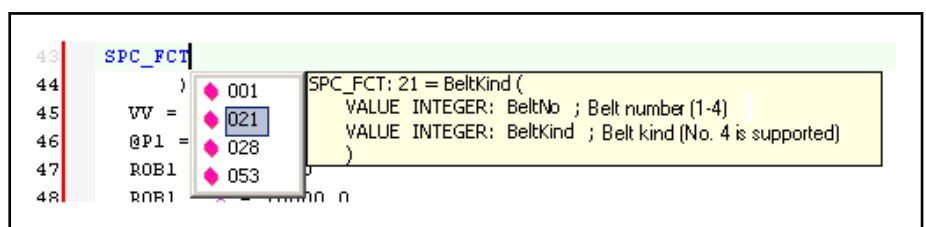


Fig.15-33: Selecting the special function

<Return> applies the special function selected in the list. The default transfer parameter names can be changed.

Kinematic User Interface

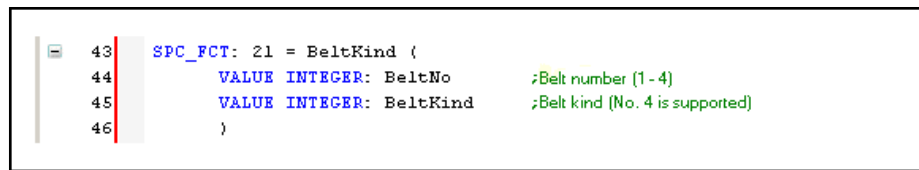


Fig. 15-34: Inserted special function

15.7 Compiler

The compiler creates the program in a machine-readable form from the source text and checks each single line for a correct command syntax. The source text can be compiled in the editor if an RCL program is selected.

Start the compilation as follows:

- via the main menu

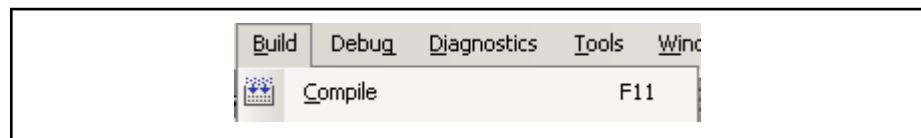


Fig. 15-35: Selecting the compiler

- via the context menu in the editor

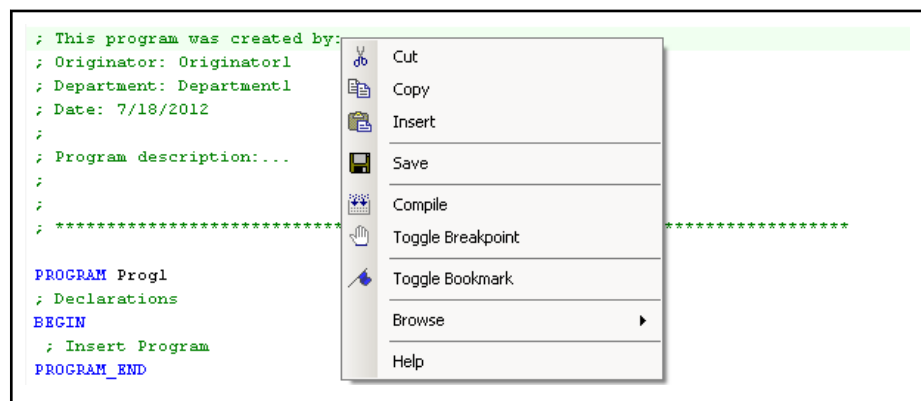


Fig. 15-36: Selecting the compiler

- via <F11>
- a .QLL program can also be compiled by selecting the menu item "Compile" in the context menu of the program.

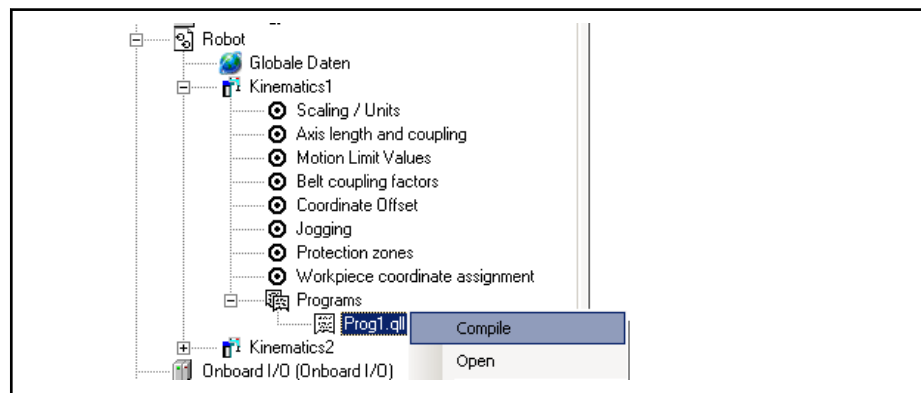


Fig. 15-37: Calling the compiler via the context menu of the program in the project navigator

Kinematic User Interface

The selected file is compiled in machine language and saved in the project. An additional immediate saving in the control can be enabled via the option "RoCo Compiler" (see [chapter 15.10.3 "Compiler" on page 233](#)).

The output texts as well as error messages are redirected to the dockable "Output" window.

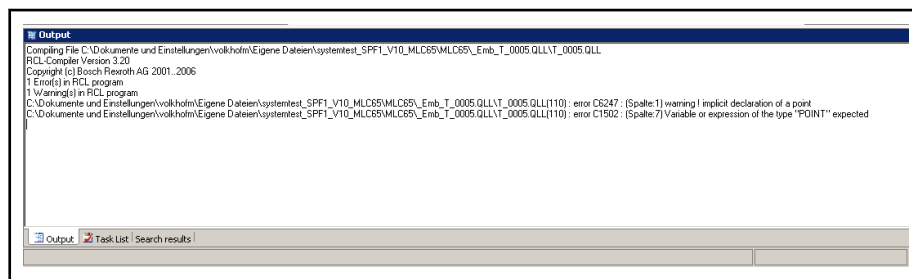


Fig. 15-38: Output window of the compiler

If error messages are displayed in the output range, you can jump directly to the position in the program where the error was identified via a double-click on the error message in the **Task list** window.

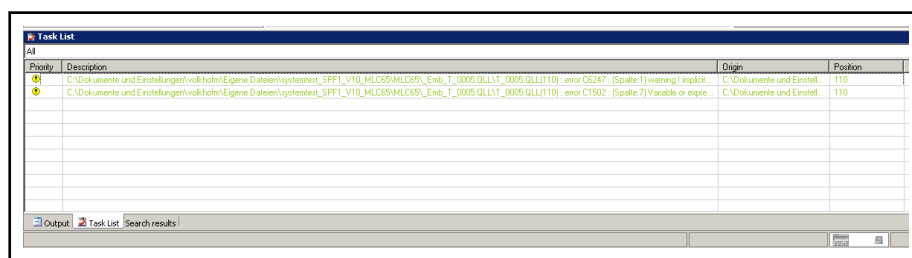


Fig. 15-39: Outputting compiler error messages

15.8 Program for Tracing

15.8.1 General Information

In this mode, the block number of an active block of a running RCL program is graphically displayed.

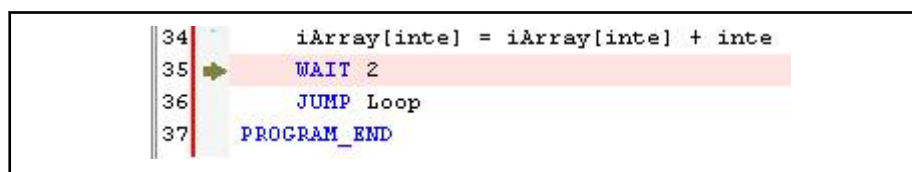


Fig. 15-40: Displaying current block

15.8.2 Connecting for Tracing

Use the **Connect for Tracing** function in the **Debug ► Log in** menu item to connect the RCL programming system to a running RCL program and thus establish the online mode.

This command enables the graphical display of the active block in the RCL editor.

Editing is not possible in this mode and breakpoints cannot be set either!

15.8.3 Disconnecting when Tracing

The control is disconnected via the function **Disconnect program** in the menu item **Debug ► Log off**.

It is returned to offline mode.

15.8.4 Starting Program for Tracing

Use the function **Start Program for Tracing** in the menu item **Debug ► Log in** to connect the RCL programming system to the control and start the RCL program in the "Program tracing" mode.

This command enables the graphical display of the active block in the RCL editor.

Editing is not allowed in this mode. Not even breakpoints can be set!

15.8.5 Stopping Program during Tracing

The running program is stopped and the control is disconnected via the function **Stop program** in the menu item **Debug ► Log off**.

It is returned to offline mode.

15.9 Debugger

15.9.1 General Information

The execution of a program in "Debug" mode can be checked for logic programming errors and the content of the variables in the dockable "Monitoring" window can be displayed.

15.9.2 Connecting in Debug mode

Use the function **Connect in debug mode** in the menu item **Debug ► Log in** or

the icon  in the RoCo toolbar to connect the RCL programming system to a running RCL program in debug mode and thus establish the online mode.

In this mode, breakpoints can be set or deleted in the RCL program.

15.9.3 Disconnecting in Debug mode

The control is disconnected via the function **Disconnect program** in the menu item **Debug ► Log off**.

It is returned to offline mode.

15.9.4 Starting Program in Debug Mode

Use the function **Start program in debug mode** in the menu item **Debug ► Log in** to connect the RCL programming system to the control and start the RCL program in the debug mode.

In this mode, breakpoints can be set or deleted in the RCL program.

15.9.5 Stopping Program in Debug Mode

The running program is stopped and the control is disconnected via the function **Stop program** in the menu item **Debug ► Log off**.

It is returned to offline mode.

15.9.6 Setting and Deleting Breakpoints

Set breakpoints to interrupt the running program at a specified location.

Set the cursor to the location in the program where the breakpoint should be set.

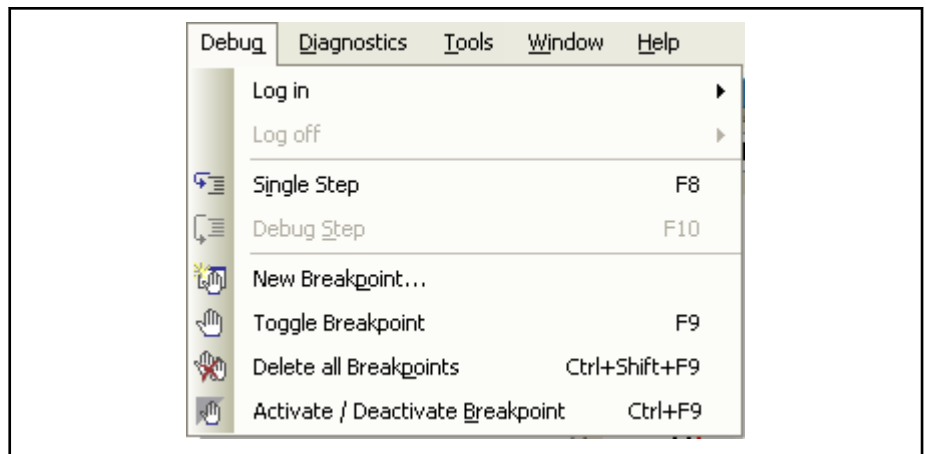


Fig.15-41: "Debug" menu

Click on the **New Breakpoint...** function in the **Debug** menu to set the breakpoint or to delete it again. Set and delete a breakpoint using the **Toggle Breakpoint** function or <F9>. Breakpoints can also be set and deleted by clicking on the indicator bar (grey margin on the left of the source text window).

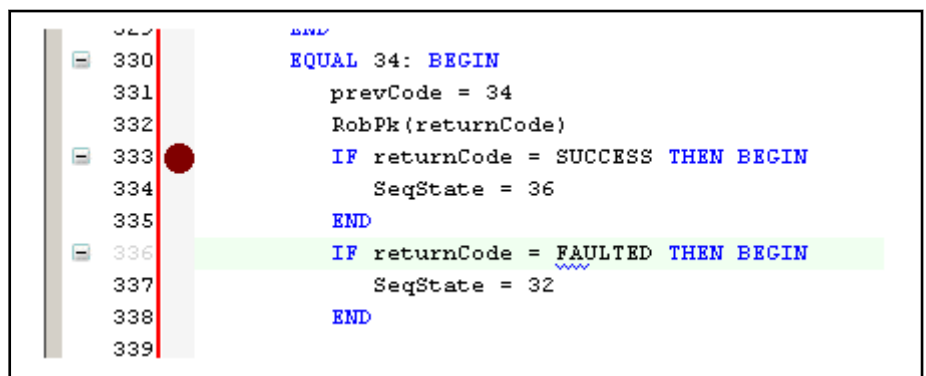


Fig.15-42: Set breakpoint

Delete all breakpoints using the **Delete all Breakpoints** function in the **Debug** menu or the shortcut <CTRL>+<Shift>+<F9>.

15.9.7 Debugging a Program

If the program was started in debug mode, tracing is interrupted at first breakpoint.

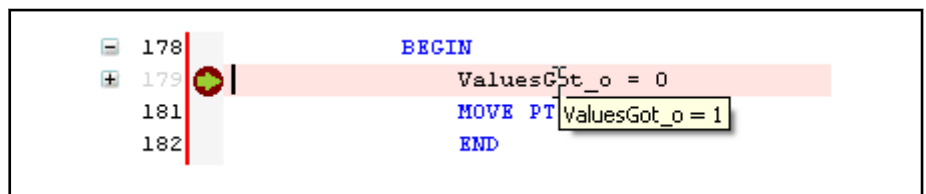


Fig.15-43: Program tracing stops at first breakpoint

Continue the program after a stop at a breakpoint with <F5>. Test the program in single steps with <F8>. Close the debugger with <Shift> + <F9>.

Set the cursor on a variable to show the current variable value in the tooltip.

15.9.8 Displaying Current Block Number

When the debug mode is started, the active block is graphically displayed in the RCL editor.

Kinematic User Interface

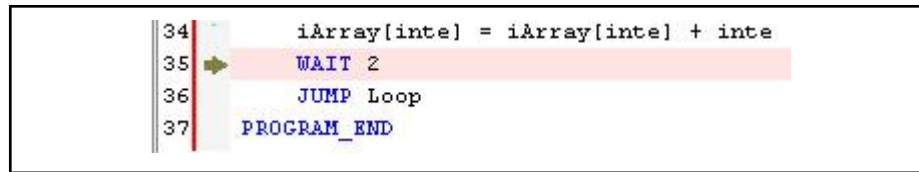


Fig. 15-44: Displaying the current block number

In addition to the graphic display, the current block number (SA ..) is displayed in the IndraWorks status bar.

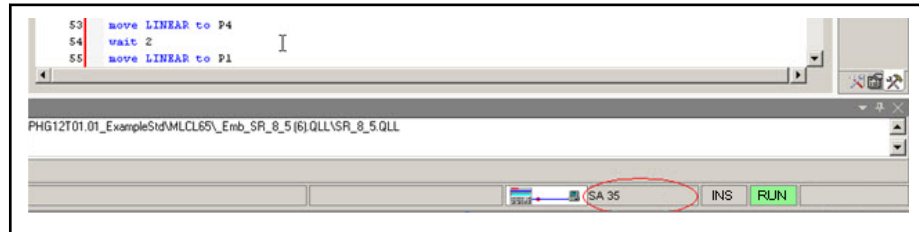


Fig. 15-45: Block processing (current block number)

Navigate to the currently executed line via the context menu **Go to Current Program Line**.

This is especially useful if the currently executed lines are located in the hidden section of the editor in case of a long RCL program.

15.9.9 Monitoring Variables

Variables can be transferred to the monitoring window in the **debug mode**. The variables can also be entered directly into the monitoring window.



The icon in the RoCo toolbar opens the monitoring window.

15.10 Settable Options for RoCo

15.10.1 General Information

Change the RoCo default settings via the IndraWorks menu item **Tools ► Options** under "XLC/MLC":

- Connection settings (see [chapter 15.10.2 "Connection settings" on page 233](#))
- Compiler (see [chapter 15.10.3 "Compiler" on page 233](#))
- Text editor (see [chapter 15.10.4 "Text Editor" on page 235](#))

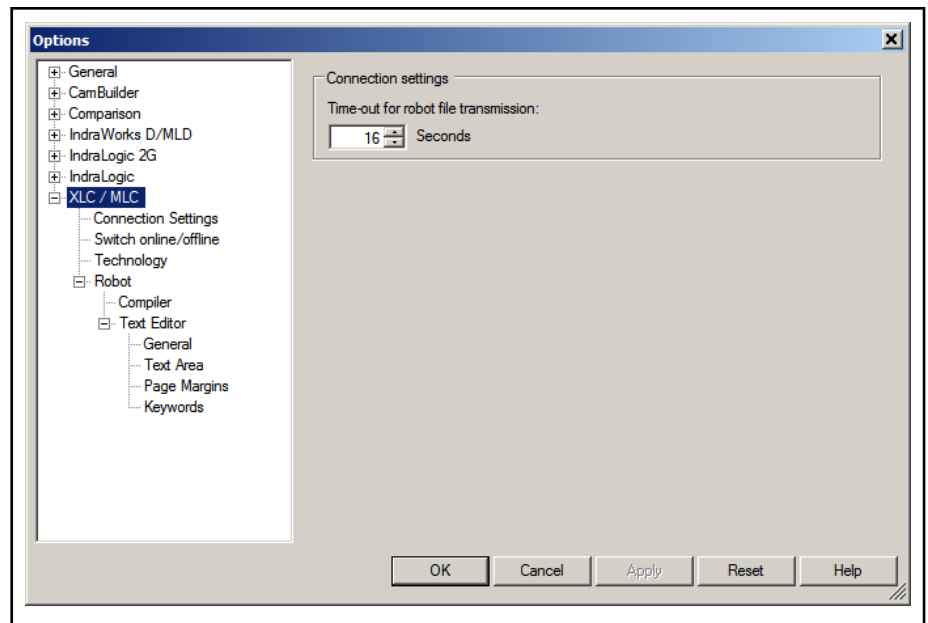


Fig. 15-46: RoCo "Options" dialogs

15.10.2 Connection settings

The time-out values for the RoCo during file transfer to the control can be changed using this dialog.

The currently set value is displayed in "seconds".

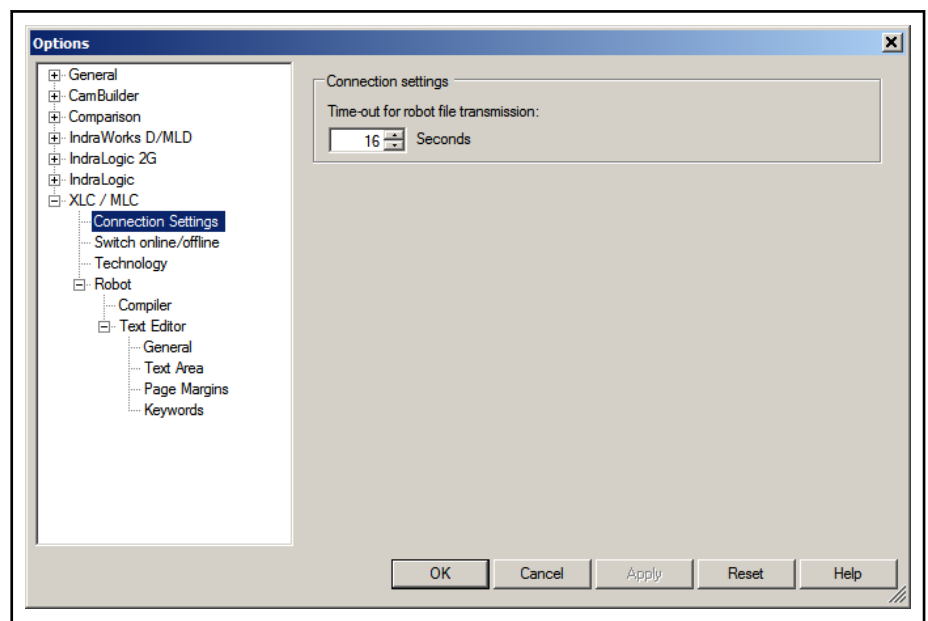


Fig. 15-47: Connection settings

15.10.3 Compiler

This "Options" dialog provides settings to add compiler instructions to a new RCL program, to enable the option "Copy to control after compiling (without confirmation)" and to disable the message "Overwrite point file" in debug mode.

Kinematic User Interface

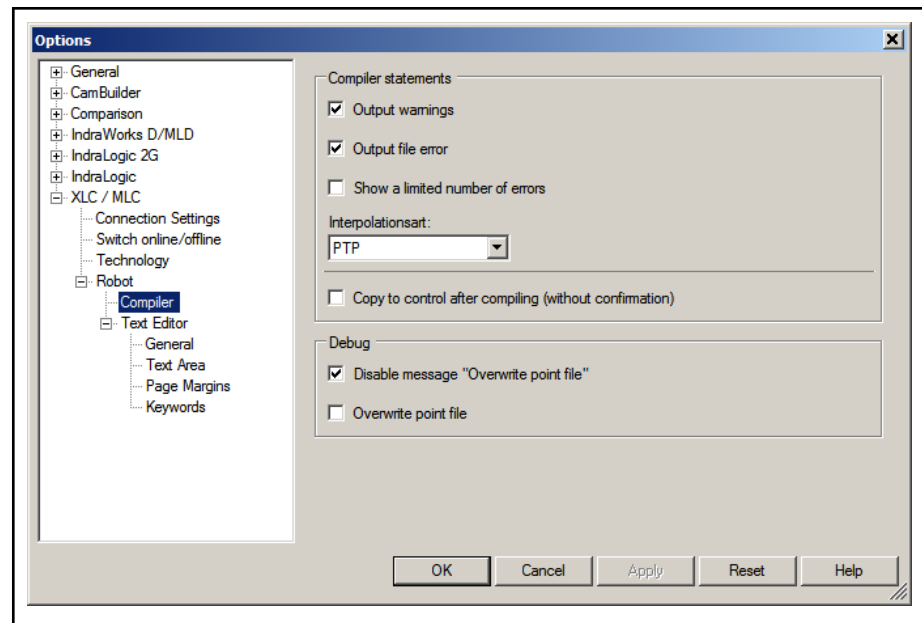


Fig. 15-48: RoCo compiler settings

Compiler instructions (example in the following):

1. Output warnings

If the checkbox is **not** selected, the "WARNING" compiler instruction is added when a new RCL program is created .

With this setting, possible warnings are not displayed after a faulty compilation.

2. Output file error

If the checkbox is **not** selected, the "File_ERROR" compiler instruction is added when a new RCL program is created.

This setting impedes the output of file errors at program runtime. That means that a program abort can be avoided and individual error messages can be implemented in the program.

3. Show a limited number of errors

If the checkbox is selected, the "WARNING" compiler instruction is added when a new RCL program is created . The maximum number of errors to be output can be determined by changing the value "100" in the RCL program.

This instruction can limit the number of errors to be output to a specified number after a faulty compilation.

4. Interpolation method

If a new RCL program is created, the selected interpolation type is added as compiler instruction.

This setting remains active until another interpolation type is selected by the ";;INT=" instruction.

5. Copy to control after compiling (without confirmation)

If this checkbox is selected, the RCL program and the files generated by the control are loaded to the control after a correct compilation. If this option is not selected, the program is only compiled.

6. Disable message "Overwrite point file"

Kinematic User Interface

If this checkbox is selected, it is not queried upon debug mode start whether the already existing point file is to be overwritten.

7. Overwrite point file

If this checkbox is selected and the checkbox "Disable message "Overwrite point file"" is selected, the point file is overwritten in the control without prompt.

```

1  ;;COMPILER = RCL3
2  ;;WARNING- ← Instruction "don't show warnings"
3  ;;FILE_ERROR- ← Instruction "don't show file errors"
4  ;;ERROR = 100 ← Instruction "limit displayed errors to 100"
5  ;;INT = PTP ← Interpolation type selection "PTP"
6  ;;KINEMATICS: { l= Kinematics1}
7  ;;Kinematics1.JC_NAMES = vaxis1, vaxis2
8  ;;Kinematics1.WC_NAMES = Ka,Kb

```

Fig. 15-49: Added compiler instructions

15.10.4 Text Editor

General Information

The following text editor properties can be adjusted in this "Options" dialog:

- Undo
- Auto completion
- Line break
- Tab size
- Indent width

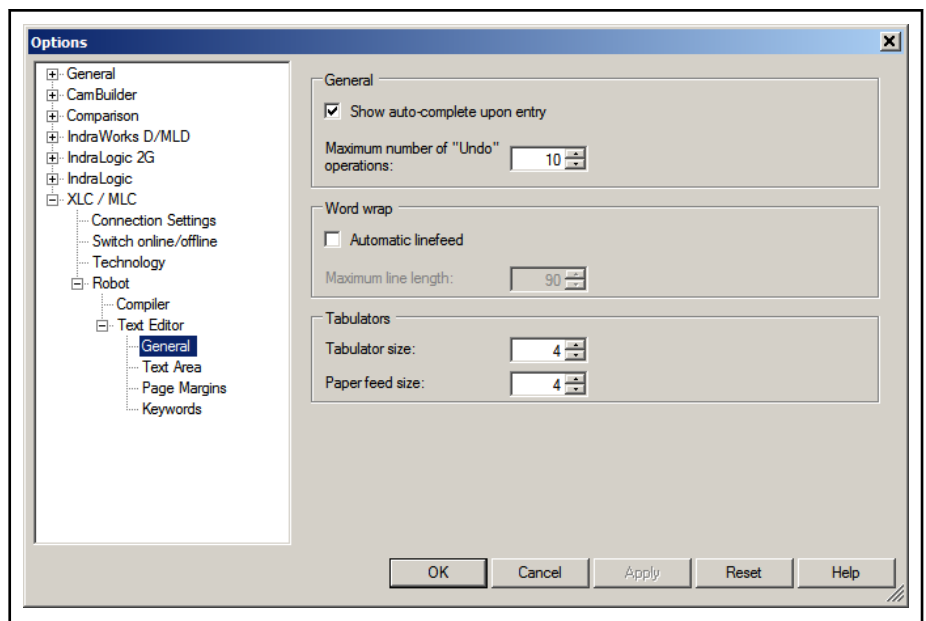


Fig. 15-50: Text editor settings

Setting options in the text editor "Options" dialog:

1. Show auto-complete upon entry

If this checkbox is selected, a window opens while writing a word. In this window, the word with the highest match is highlighted. Words in this list can be selected. The highlighted word is applied with "Return".

Kinematic User Interface

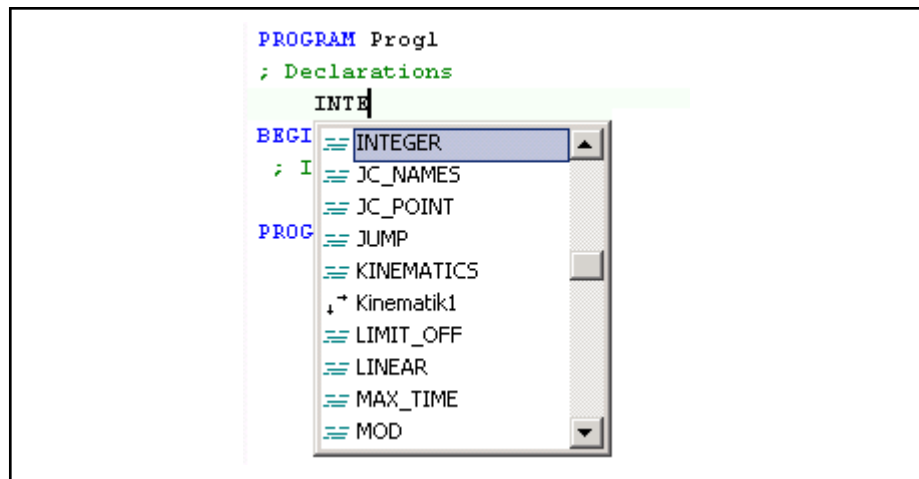


Fig. 15-51: Auto completion

2. **Maximum number of "Undo" operations**

The maximum number of undo steps can be set in the RoCo-Editor.

3. **Automatic linefeed**

If this checkbox is selected, a linefeed is executed automatically in the RoCo-Editor if the number of characters in the current line exceeds the value set in "Maximum line length".

4. **Tabulator size**

The size of the tabulator step width can be set in the "Tabulator size" selection box.

Example: In [fig. 15-51 "Auto completion" on page 236](#), the tabulator size is set to "4". That means that a tabulator step is the size of four blanks (see line "INTE").

5. **Indent width**

If "Return" is pressed, there is an automatic linefeed in the editor. This automatically generated linefeed is called "indent" or "indentation". The number of characters for the indentation can be set in the "Indent width" selection box.

Text Area

In this "Options" dialog, colors for highlighting as well as font type and font style in the RoCo-Editor can be set.

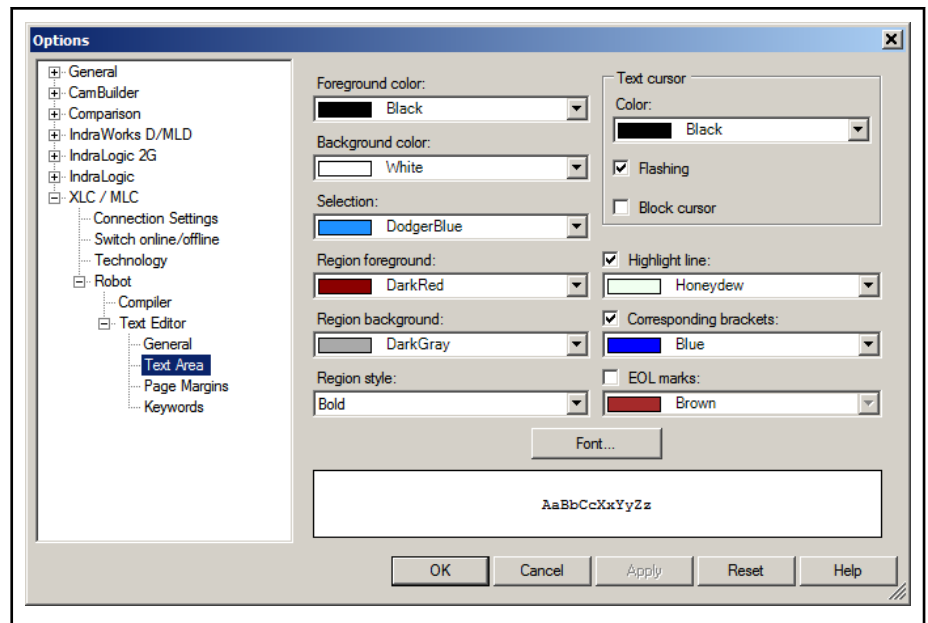


Fig. 15-52: Text area settings

Functions of the "Text area" "Options" dialog:

1. Foreground color, background color, font style of the editor

The foreground colors (text color) and the background colors of the Ro-Co-Editor can be selected with the combo boxes "Foreground color" and "Background color".



The colors for keywords, comments, etc. cannot be changed in this dialog. To change the above mentioned, open the "Keywords" (see [chapter "Keywords" on page 241](#)).

Press the **Font type** button and the Windows dialog to select font type, style and size is displayed.

2. Selection, Region foreground, Region background, Region style

Set the color for a highlighted area via "Selection". The minimized representation, e.g. between the keywords BEGIN and END or SUBROUTINE and SUB_END can be changed in color and in font style (bold, italic, etc.) via "Region foreground", "Region background" and "Region style".

Example:

Kinematic User Interface

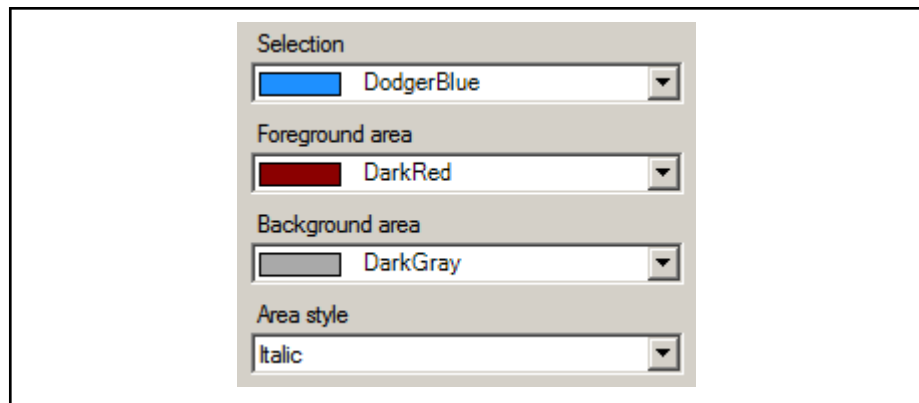


Fig. 15-53: Settings for selection, foreground, background and style

```

PROGRAM Prog1
; Declarations
DEF POINT: P0, P1, P2, P3, P4, P5, P6

BEGIN
; Program
KIN01.V_PTP = 0.4           ; Velocity for point to point movements
KIN01.A = 1000             ; Acceleration
KIN01.V = 1200             ; Velocity

MOVE TO P0                 ; Move to P0
KIN01.V = 400              ; Velocity
WAIT 3                     ; Wait 3 seconds
REPEAT 4 TIMES             ; Repeat 4 times
  MOVE TO P0
  MOVE TO P1
  MOVE TO P2
  MOVE TO P3
  MOVE VIA P4, P5, P6 TO P0 ; Move via P4, P5, P6 to P0
REPEAT_END

PROGRAM_END

```

Fig. 15-54: Display of the selected text area

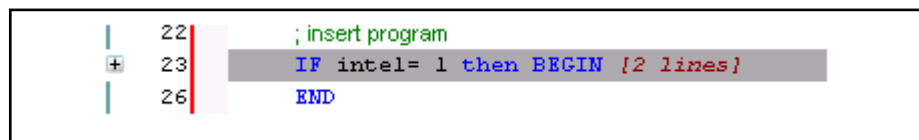


Fig. 15-55: Representation in color and text of the minimized area between BEGIN and END



If an area is minimized, this is indicated by a "-" symbol at the page margin. The area is minimized when clicking on the "-" symbol. This is indicated by a "+" symbol at the page margin. Minimization can be undone by clicking on the "+" symbol.

3. Cursor settings

The display of the text cursor in the RoCo-Editor can be changed. The following settings can be made:

- Cursor color
- Cursor flashing
- Block cursor

Example:

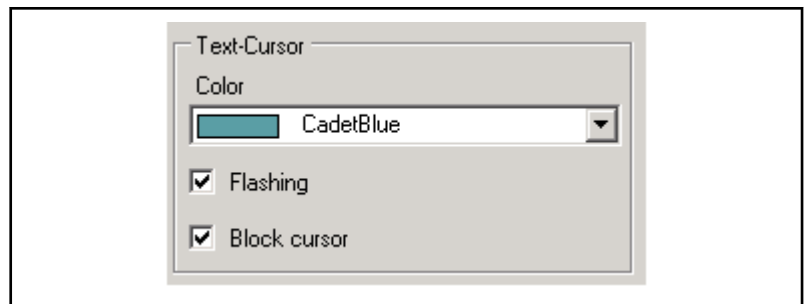


Fig. 15-56: Setting option for the text cursor

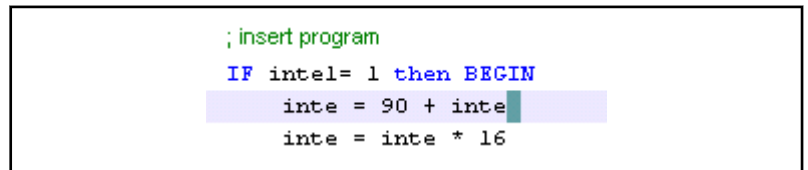


Fig. 15-57: Display of the text cursor in the RoCo-Editor

4. Highlight line

If the checkbox "Highlight line" is selected, the line in the cursor is highlighted with the selected color.

Example:

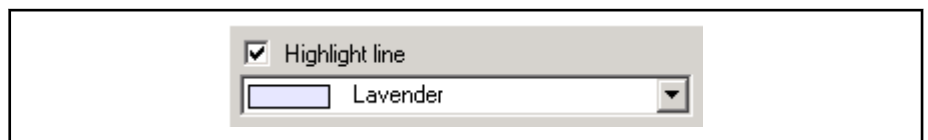


Fig. 15-58: Setting options for the current line

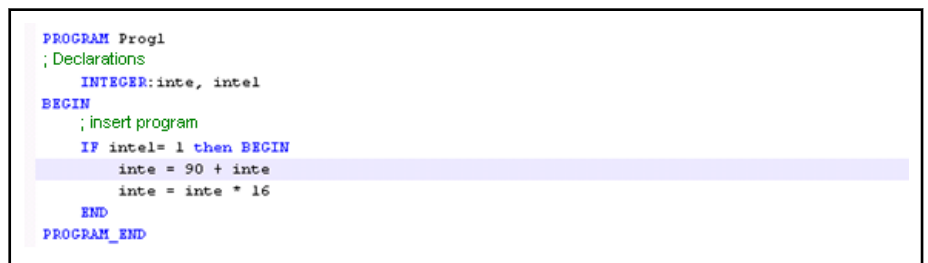


Fig. 15-59: Colored highlighting of the current line

5. Associated brackets

If the cursor is located directly behind an END, REPEAT_END etc., the respective beginning of the instruction block (BEGIN, REPEAT, etc.) is highlighted with a frame if the checkbox is selected. Additionally, the color of the frame can be selected.

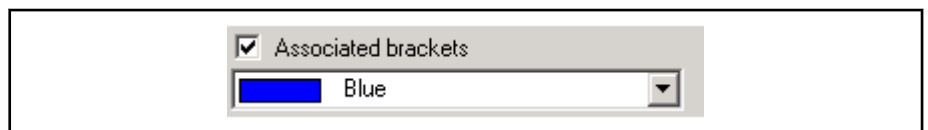


Fig. 15-60: Setting options for a program block

Highlighting the block beginning in the following figure is carried out according to the settings in [fig. 15-60 "Setting options for a program block" on page 239](#).

Kinematic User Interface

```

IF intel= 1 then BEGIN
    inte = 90 + inte
    inte = inte * 16
END

```

Fig. 15-61: colored highlighted of an instruction block

6. EOL marks

The display of the EOL (End Of Line) can be carried out here. If the option is enabled, it highlights a point at the end of the line. Additionally, the color of the point can be selected.

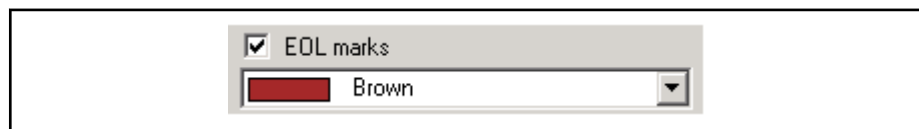


Fig. 15-62: Setting options for the EOL display

```

BEGIN.
; insert program.
IF intel= 1 then BEGIN.
    inte = 90 + inte.
    inte = inte * 16
END.

```

End Of Line marks

Fig. 15-63: EOL marks in the text

Page Margins

In this "Options" dialog, the page margins in the RoCo-Editor can be set.

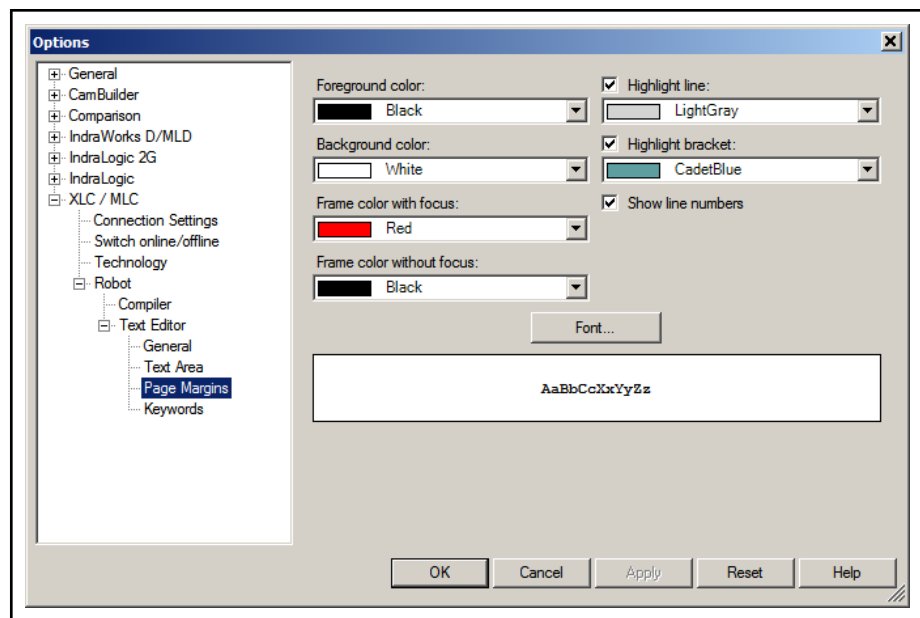


Fig. 15-64: Page margin settings

Functions of the "Page margins" "Options" dialog (example see below):

1. Foreground color, background color, font style of the page margin

The foreground colors (text color) and the background colors of the page margin (on the left) can be set using the combo boxes "Foreground color" and "Background color".

2. Frame color "in the focus", Frame color "without focus"

The vertical frame line on the right next to the line numbers indicates whether the window of the RoCo-Editor has the focus. This option can specify the line color for both states.

3. Highlight line

The color of the line number to be highlighted on which the current cursor is located can be set in this option.

4. Highlight brackets

If this option is selected, the area in the brackets ("(...)") is indicated with a square bracket ("[]") while editing at the page margin. Additionally, the color of the indicated bracket can be selected.

5. Show line numbers

The display of the line numbers at the left page margin can be switched on/off via this option.

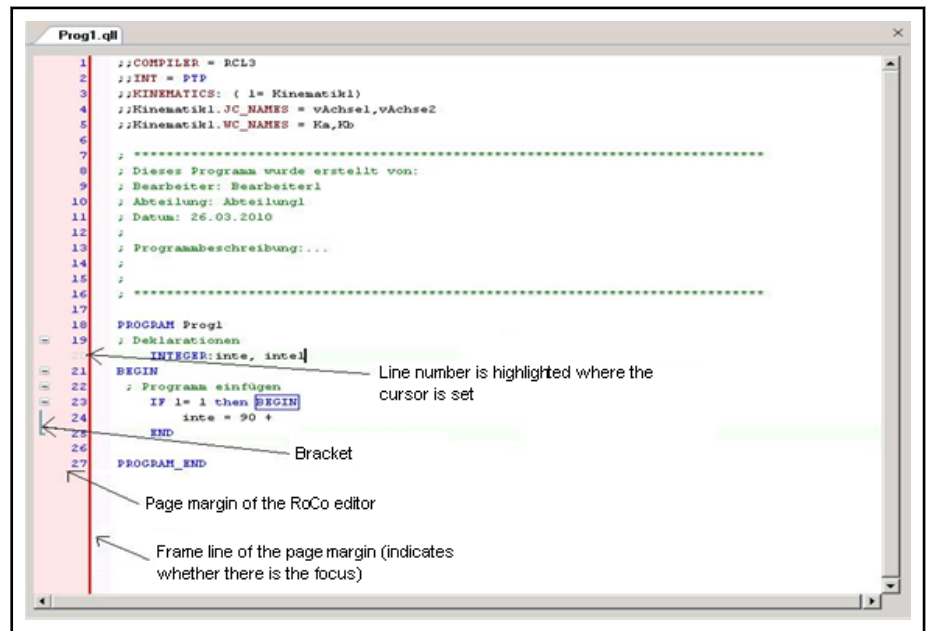
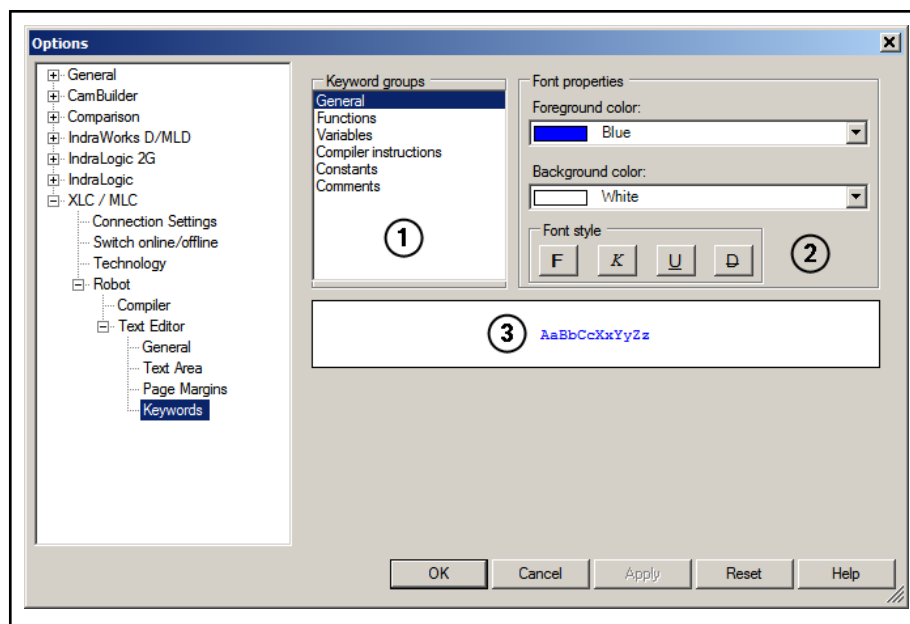


Fig. 15-65: Setting option for the page margin in the RoCo-Editor

Keywords

The colored display of the keywords can be changed in this "Options" dialog. The settings options include the colored representation (foreground color) the writing cut of the text (bold, italics, underlined, strikethrough) as well as the background color.

Kinematic User Interface



- ① Keyword groups
- ② Font Properties
- ③ Example display

Fig. 15-66: Keyword settings

Procedure for setting changes:

1. Marking the keyword group in the list ①
2. Changing the properties in ②
3. The example display indicates the representation with changed properties ③
4. Click on **Apply Comment** or **OK**..

16 Using the RoCo Multi-Kinematic Technology

16.1 General Information

- One MLC control supports up to 16 kinematics (robots).
- Different robot types can be controlled side by side in one MLC (one robot type, e.g. Cartesian, SCARA, Delta, etc., for each kinematics).
- One kinematics supports up to 16 axes (inclined belt axes).
- One kinematics supports up to 4 belt axes.
- Several robots can work synchronously on one belt (belt axes can be assigned to several kinematics).
- The operating mode can be selected individually for each kinematics (Programmed Motion, Manual Motion, Discrete/Synchronized Motion).
- Each kinematics has its own PLAN shared memory interface (Kind).

16.2 Creating Kinematics

16.2.1 General Information

In the parameter CA-0-2484, Kinematic configuration list, the kinematics are created via the menu items **Robot ▶ Add** when IndraWorks is offline.

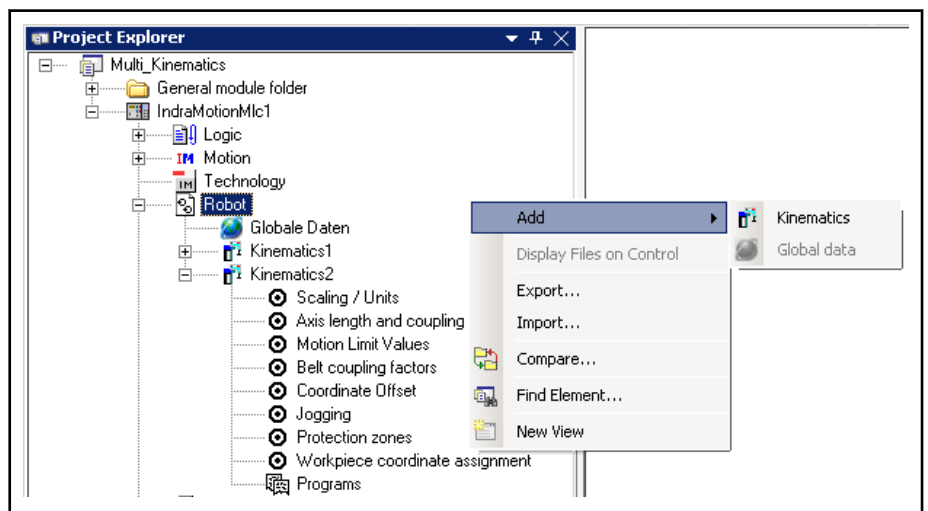


Fig. 16-1: Creating a kinematics

When creating a kinematics, a consecutive number is automatically assigned. Specify a unique symbolic identifier (device name) in the kinematic wizard. The wizard starts automatically when adding,

Using the RoCo Multi-Kinematic Technology

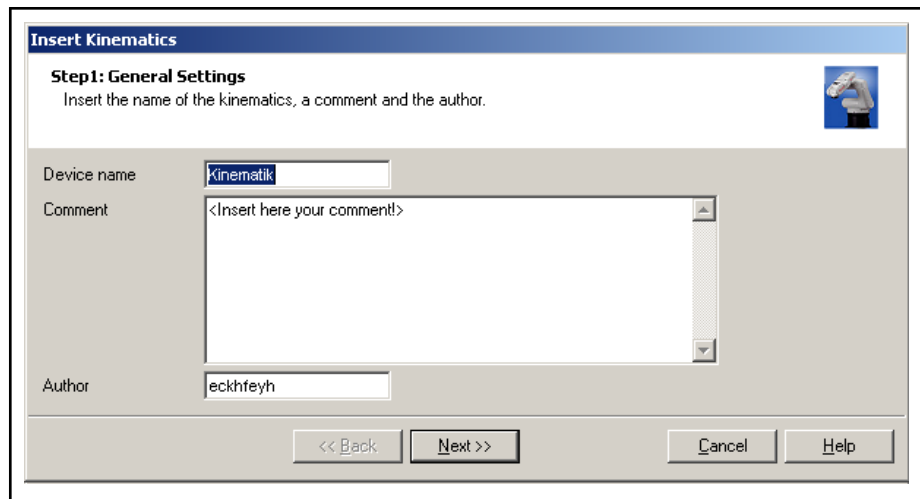


Fig. 16-2: Entering the kinematic name

The PLC program uses the kinematic name to access the kinematics irrespective of the assigned number. Furthermore, number and name of the kinematics are entered the RCL program head when an RCL program is created. This entry can ensure later on that only programs written for this kinematics can run on this kinematics.

16.2.2 Defining the Kinematic Type (Robot Type, K-0-0031)

Define the kinematic type via **Next**.

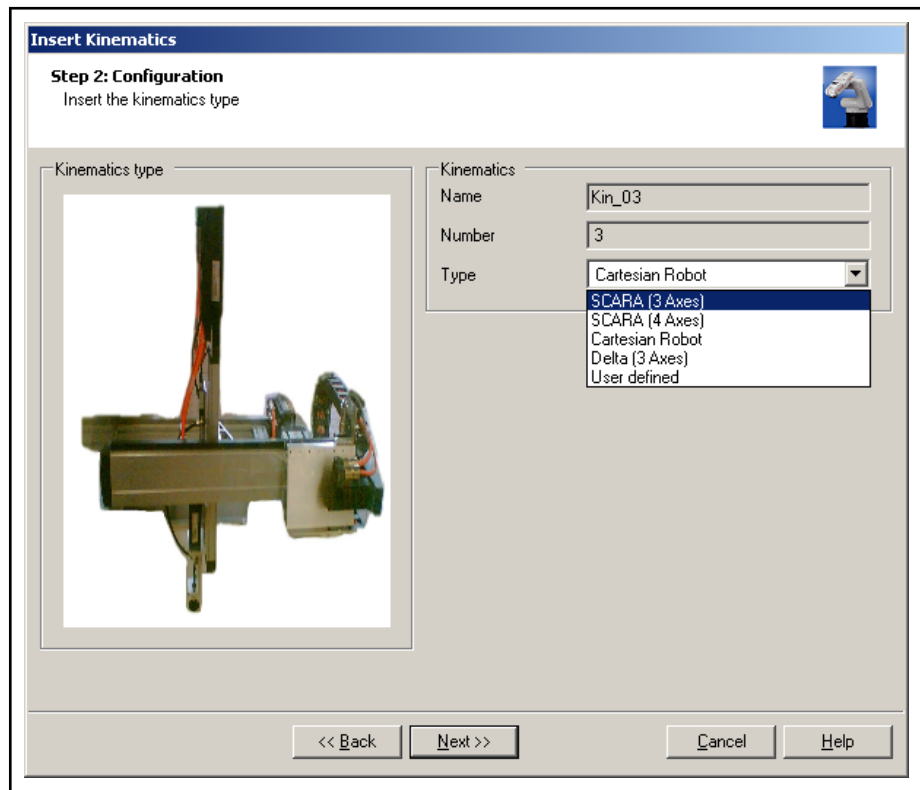


Fig. 16-3: Defining the kinematic type

All types (SCARA, Cartesian Robot, Delta, etc.) are always available here. Any robot types can be operated together in one MLC.

16.2.3 Defining the Axis Assignment (K-0-0008)

An MLC control can control up to 16 robots (kinematics) at a time with up to 16 axes each. Virtually, it can sum up to 256 axes. However, this theoretical number of axes is limited by the maximum number of axes of the control.

- On the L40.2, a maximum of 32 axes can be operated. This is a maximum of 16 kinematics with two axes each.
- On the L65, a maximum of 64 axes can be operated. This is a maximum of 16 kinematics with four axes each.

The axes available in the control are listed in the parameter C-0-0484, Axis configuration list. In the parameter K-0-0008, Axis configuration list, these axes can be assigned to their respective kinematics. The kinematic wizard provides only axes from C-0-0484, which are not yet assigned to a kinematics, as positioning and orientation axes. The wizard starts automatically when a kinematics is created.

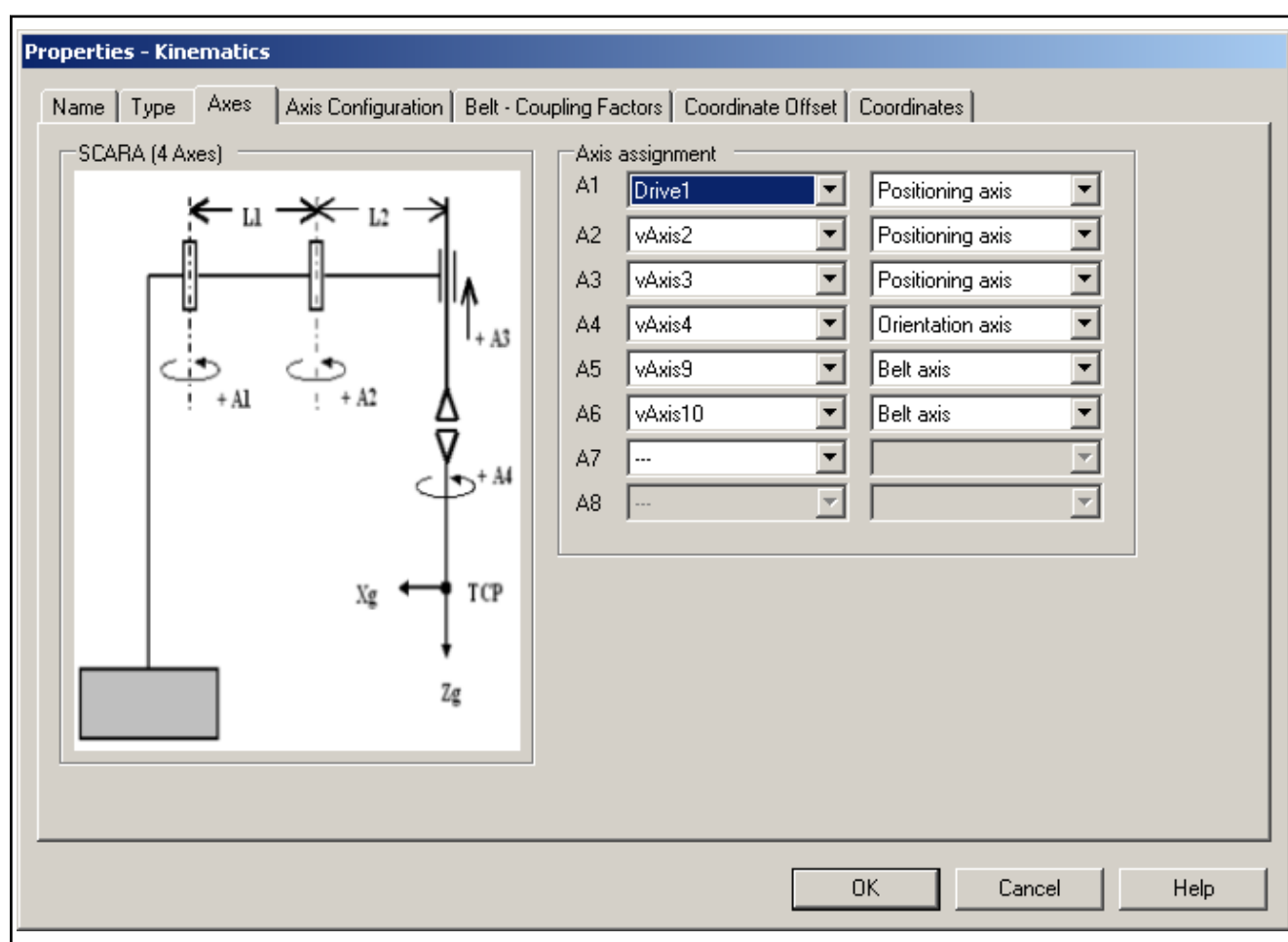


Fig. 16-4: Axis assignment for various kinematics (in this case, 4-axis Scara)

Using the RoCo Multi-Kinematic Technology

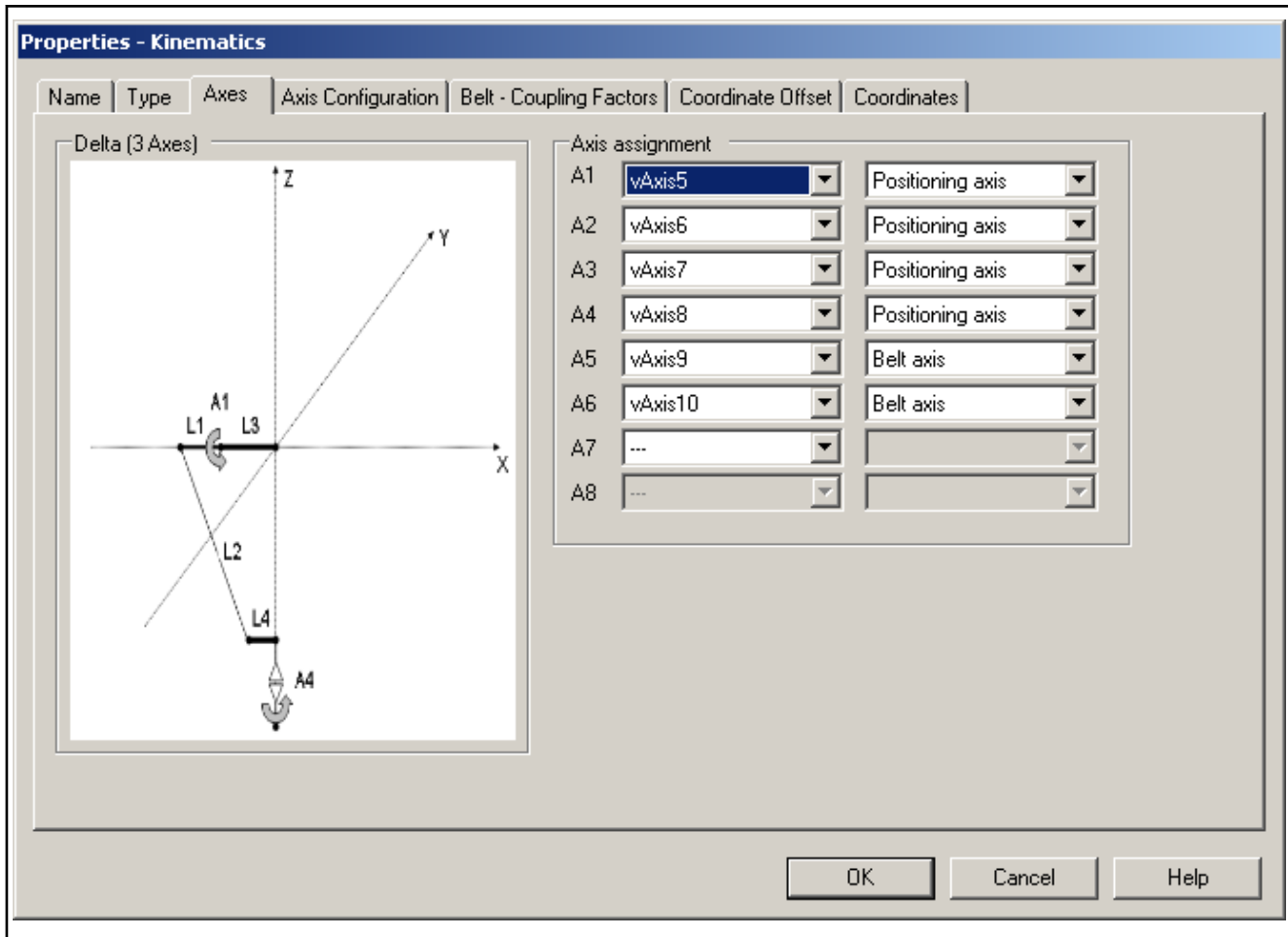


Fig. 16-5: Axis assignment for various kinematics (in this case, Delta pack)

This ensures that no axis (except for belt axes) can be assigned to more than one kinematics.

When switching from the parameterization mode (P2) to operating mode (BB or P4), the axis assignment in the K-0-0008 parameters is checked again and the error F0160040, Error in cross-check of K-parameters, is output if applicable. In case of error, the parameter K-0-0008 is found in K-0-0011, List of all invalid K-parameters.

Belt axes can be simultaneously assigned to several kinematics. This feature allows several different robot types to move synchronously and independently on one or more belts. Independently means that each kinematics manages its own belt counter.

Thus, the belt blending can be operated separately for each kinematics. The other kinematics do not have to be stopped. However, to detect the product position, all belt counters for the kinematics always have to be saved - unless it is already determined by which robot the product is picked up.

16.2.4 Deleting Kinematics

A kinematics can also be deleted from the command configuration of the IndraWorks project.

Using the RoCo Multi-Kinematic Technology

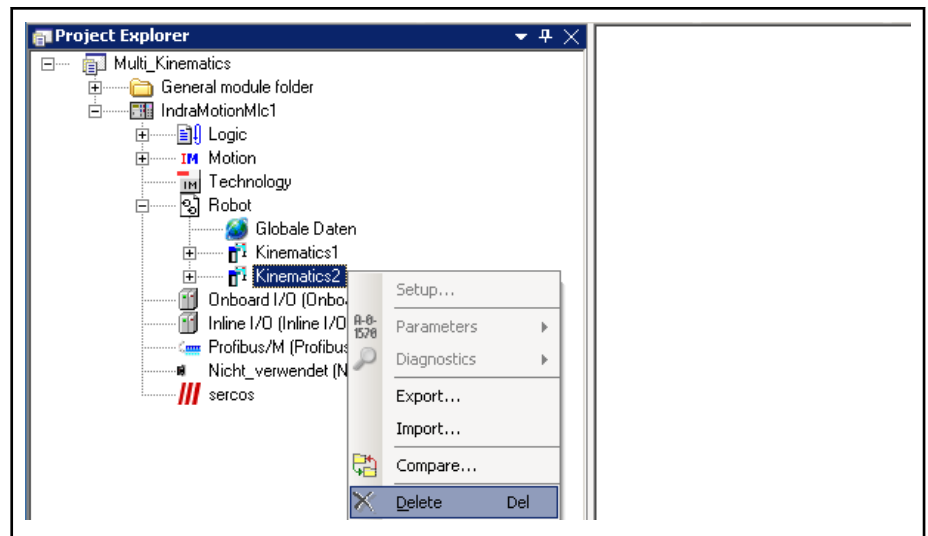


Fig. 16-6: Deleting a kinematics



In contrast to the axis configuration list in C-0-0484, the list in C-0-2484 may not contain any gaps.

Thus, the kinematics are always continuously addressed from 1..n (n = number of kinematics used). This simplifies the work for the firmware and the PLC program.

To delete a kinematics from the command configuration in IndraWorks, reload and assign the previously saved parameters to the remaining kinematics accordingly.

16.3 Coordinated Motion of the Kinematics

The axes of a robot can be moved

- "directly" from the PLC program via the PLCopen function blocks such as MOVE_LINEAR **the PLCopen function blocks are still in the development phase**
- by executing an RCL program (one RCL program for each kinematics).
- by setting the JOG bits in the respective KinData interface (manual motion).

Using the RoCo Multi-Kinematic Technology

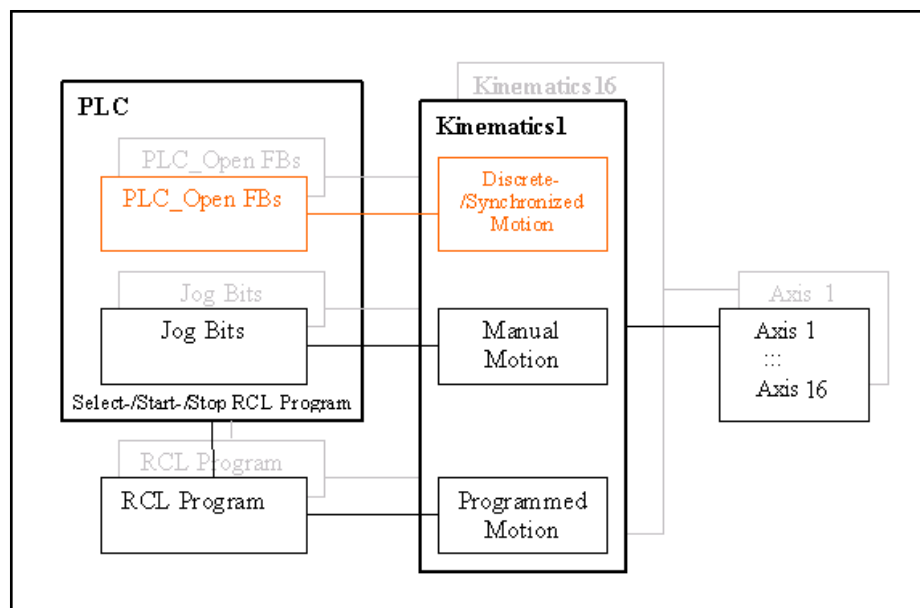


Fig. 16-7: Control options for axes in kinematics

All kinematics mostly run independently of each other, i.e. kinematics 1 can be in Manual Motion, kinematics 2 can be in Programmed Motion and kinematics 3 can be in Discrete Motion.

In other words: while one robot executes an RCL program and the motion of another is directly controlled by the PLC program with the PLCopen function blocks, a third robot can be jogged manually.

16.4 RCL Programs of the Kinematics



An RCL program can move exactly one kinematic. The RoCo is based on the following philosophy:

1. the RCL program is responsible for the robot's motion
2. the PLC program takes care of all other tasks (communication, coordination, system/process control...)

If a new RCL program is created using IndraWorks (**Programs** ► **Add**, etc.), head and body of the RCL program are automatically generated.

Using the RoCo Multi-Kinematic Technology

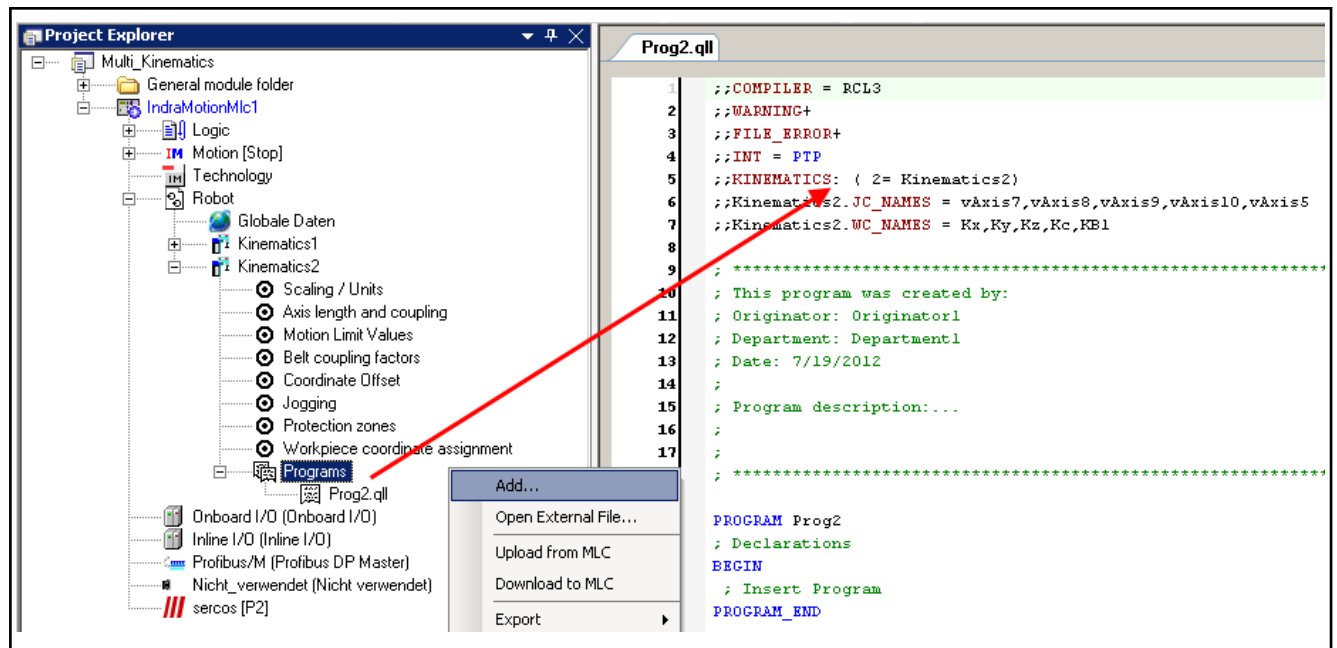


Fig.16-8: Creating a new RCL program

The entry KINEMATICS in the RCL compiler statement, e.g.

```
;;KINEMATICS: ( 2= Kinematic2)
```

defines the kinematics for which this program is written.

Subsequently, if it is tried to start an RCL program that was not written for this kinematics, the error F12C0072 is output and the kinematics state machine goes into the state KIS_Error. At first glance, this seems to be the correct response.

However, due to this response, an RCL program cannot run simultaneously on several robots. If there are several identical robots in a row, all of them can virtually run with the same RCL program. However, this can only be the case if a program is created for each kinematics and the source code is copied accordingly. Otherwise, it cannot be ensured that the RCL program can actually run on the kinematics.

In the MLC RoCo system, an RCL program can only be started once at a time. This is primarily the case due to the variable management. The variables are overwritten if the program starts more than once.

16.5 Point Tables of the Kinematics

16.5.1 General Information

One RCL program can have one point table.

From the perspective of block preparation, a point table is only another storage location for the point variables. An RCL program cannot be linked to a different (external) point table. Defines the names of the point variables. They are then permanently linked to one RCL program.

Using the RoCo Multi-Kinematic Technology

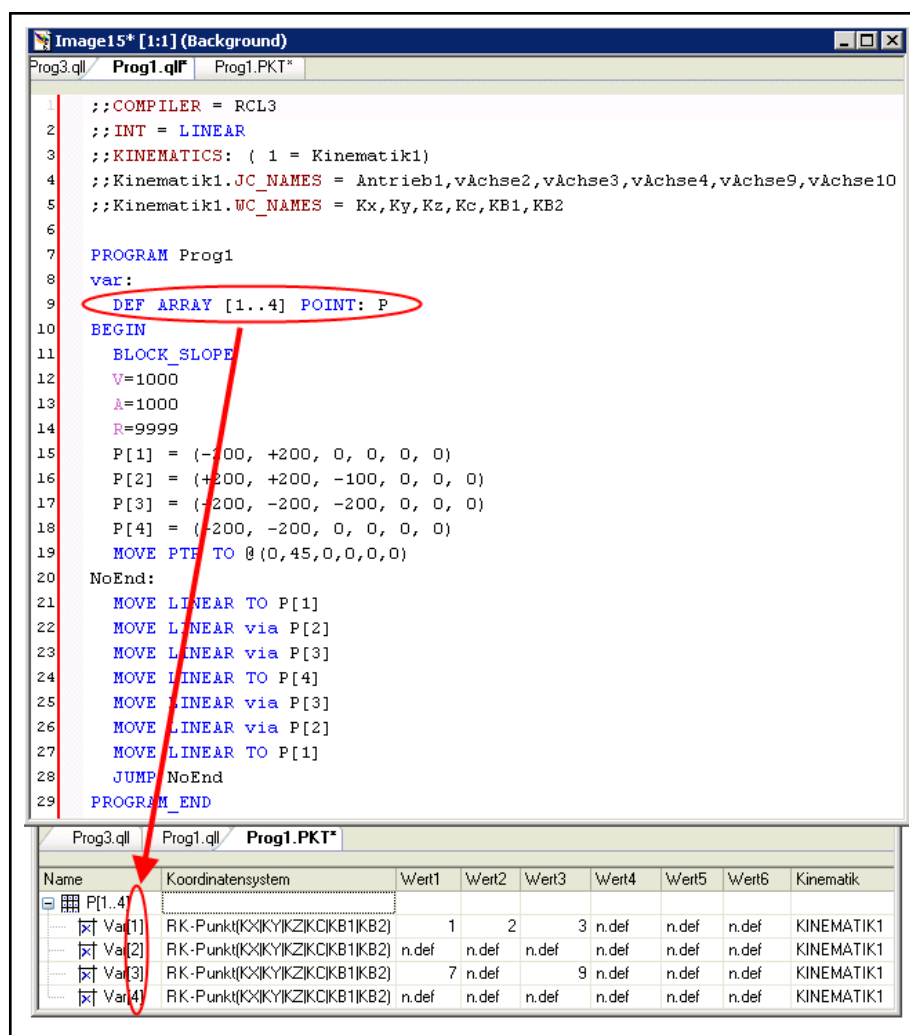


Fig.16-9: Assigning kinematics and point table

Definition of a point can be different in each kinematics (number of degrees of freedom = number of axes). The degrees of freedom for the points are set with the compiler statements JC_NAMES and WC_NAMES when creating an RCL program irrespective of the axis configuration in the parameter K-0-0008.

16.5.2 Point Tables, Assignment and Response

This results in the following strict assignment

- A point table is always permanently assigned to one RCL program.
An RCL program can have only one point table (not several!). The RCL compiler generates/modifies the point table and ensures that the IRD program can read or write the point variables from the point table.
- An RCL program cannot generate new points in the point table at run-time.
- One RCL program is assigned to one kinematics (see [RCL Programs of the Kinematics, page 248](#)).
- One kinematics can have several RCL programs.
- The RCL program name has to be unique in the entire MLC RoCo system, i.e. the program "Prog1.qll" has to be the only Prog1.qll - a second Prog1.qll may thus not exist in different kinematics.

Using the RoCo Multi-Kinematic Technology

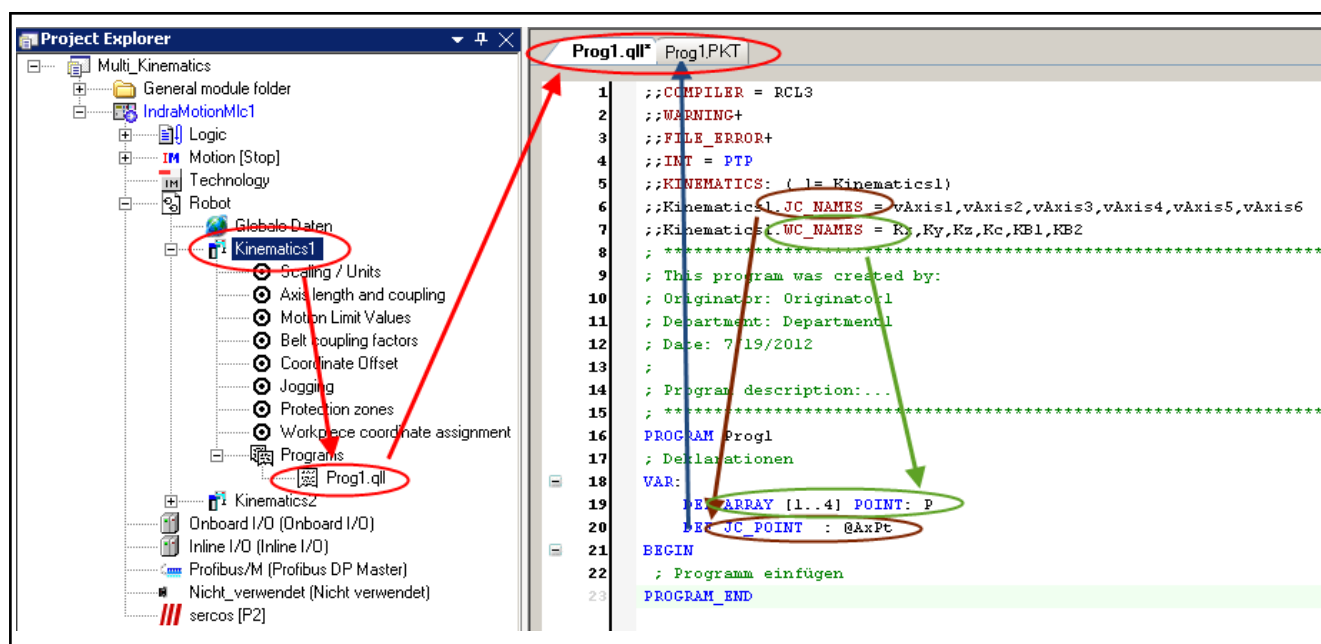


Fig. 16-10: Assigning kinematics, point table and coordinates

The "POINT" data type is defined by the compiler statement WC_NAMES (=structure with the elements Cx, Cy, Cz, etc., where each of the elements is of data type "REAL").

The compiler statement JC_NAMES defines the data type "JC_POINT" (= structure with the elements Drive1, vAxis2, etc., where each of the elements is of data type "REAL").

The number of elements in POINT and JC_POINT is always the same.

The RCL program "Prog1.qll" has exactly one point table, "prog1.pnt" with the name of the source program. The software engineer used the keyword DEF to define that these points are to be stored in the point table and not in the ird program. The program "Prog1" is permanently assigned to kinematics1. It exists only once in the entire MLC robot system.

16.6 PLC Shared Memory Interfaces (KinData) of the Kinematics

Each kinematics has its individual KinData interface to the PLC.

IndraWorks creates a variable of type "AXIS_GROUP_REF" for each kinematics under its name in the global variable list "GlobalMLCVariables" (as created for the axes).

This variable contains the GroupNo of the kinematics. Using this GroupNo, the KinData array can be indexed. This ensures that the PLC program always addresses the correct kinematics.

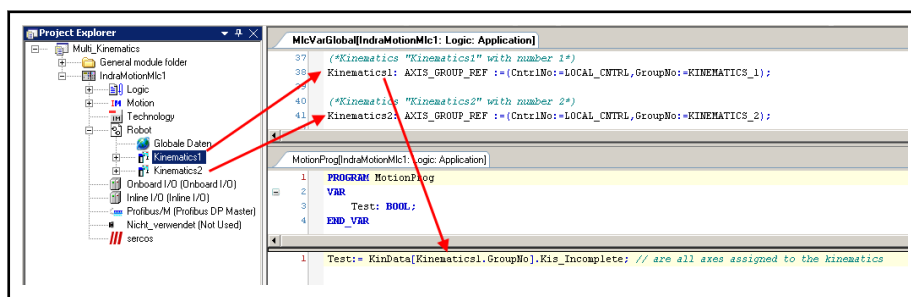


Fig. 16-11: Access to data in the shared memory

17 RobotControl Functionality Programmed in the PLC

The RobotControl functionality can often be programmed in the PLC as well as in the RCL program.

However, it is strongly recommended not to use both access options at the same time.

PLC libraries

- ML_Robot, function blocks, functions and data types to implement the RobotControl functionality.
- ML_Base, data types for ML_Robot and ML_PLCOpen of the systems IndraLogic XLC and IndraMotion MLC
- RIL_CommonTypes, data types for all DC systems.

The libraries are all described in

Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCOpen Libraries, DOK-XLCMLC-FUNLIB**V13-L1xx-EN-P, R911336360.

Function modules

- S-RAM module (CFL01.1-Y1), memory extension for IndraMotion MLC

The function modules are all described in

Rexroth IndraControl Lxx 13VRS Function Modules, DOK-CONTRL-FM*LXX**V13-APxx-EN-P, R911336283.

Parameters (XLC, MLC)

- Overview: Parameters (XLC, MLC)
- Overview: RCL parameters
- A-parameters (axis parameters)
- C-parameters (control parameters)
- K-parameters (kinematic parameters)

The parameters are described in

Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameters, DOK-XLCMLC-PARAM***V13-RExx-EN-P, R911336364.

Diagnostics (XLC, MLC)

- Overview: Diagnostics (XLC, MLC)
- F1 error, axis group error

The diagnostics are described in

Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnostics, DOK-XLCMLC-DIAG***V13-RExx-EN-P, R911336358.

...an example can be found in:

- First Steps - RobotControl Using MLC

Rexroth IndraMotion MLC 13VRS First Steps, DOK-MLC***-F*STEP**V13-COxx-EN-P, R911336293.

18 Transformation Rules

18.1 Transformation Rules, Overview

This chapter contains all transformation rules available for the programming language RCL (Robot Control Language).

- [Robot Type 00, Cartesian Robot, page 255,](#)
- [Robot Type 07, 2-Axis Delta with 1 Optional Manual Axis, page 259,](#)
- [Robot Type 09, 3-Axis Scara with 1 Optional Manual Axis, page 261,](#)
- [Robot Type 10, 4-Axis Scara with 1 Optional Manual Axis, page 264,](#)
- [Robot Type 12, 2-Axis Parallelogram, page 267,](#)
- [Robot Type 13, 2-Axis Parallelogram, page 269,](#)
- [Robot Type 18, Contact Kinematics, page 271,](#)
- [Robot Type 19, 2-Axis Parallelogram V1, page 272,](#)
- [Robot type 20, 3-Axis Delta with 1 Optional Manual Axis,, page 274,](#)
- [Robot Type 21, 2-Axis Parallelogram V2, page 280](#)
- [Robot Type 26, Robot with Offset Wrist with Six \(Seven\) Axes, page 282,](#)
- [Robot Type 27, Robot with Offset Wrist with Six Axes + Three Linear Axes, page 287,](#)
- [Robot Type 29, 2-Axis Delta with Rotary and Translatory Axis, page 294,](#)
- [Robot Type 30, Adabot, page 296,](#)
- [Robot Type 32, 2-Axis Delta with Asymmetric TCP, page 299,](#)
- [Robot Type 100, PLC, page 301](#)

18.2 Robot Type 00, Cartesian Robot

The transformation is switched off for robot type 0 (Cartesian robot). The axis coordinates (JC) are identical to the world coordinates (WC), i.e. in terms of the path motion, MOVE PTP is identical to MOVE LINEAR. All axes participating in the motion are started simultaneously and reach their target point simultaneously.

For MOVE LINEAR and CIRCULAR, the path velocity for the positioning axes in the RCL program is programmed with V and the velocities for the orientation axes are set so that all axes reach their target point at the same time.

For MOVE PTP motions, the velocity is determined based on the maximum axis velocities multiplied by the V_PTP programmed in RCL. The axis with the longest motion in terms of time determines the overall duration for the PTP motion. All other axes are adjusted to match this time.

Transformation Rules

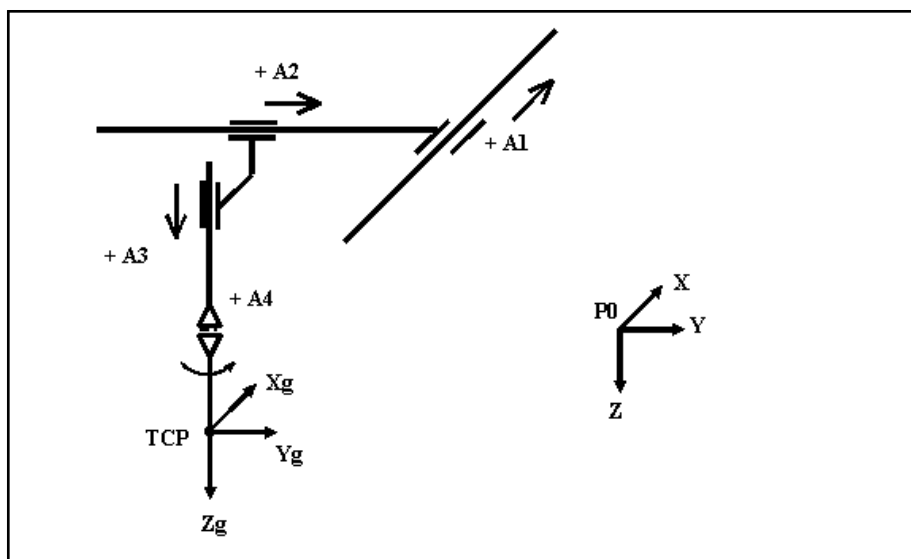


Fig. 18-1: Robot type 0, Cartesian robot, 3D perspective

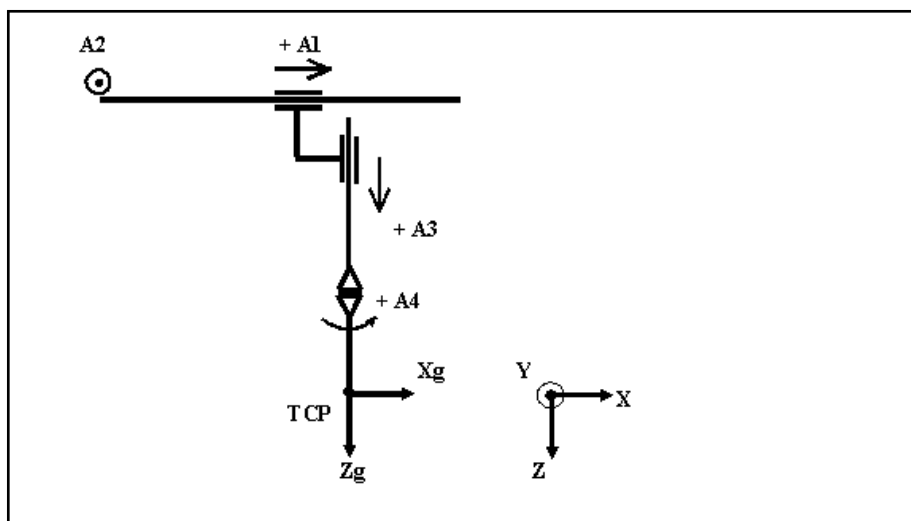


Fig. 18-2: Robot type 0, a typical X-Y-Z-C Cartesian system

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Translatory absolute	
A2:	Positioning axis (optional) Orientation axis (optional) Belt axis (optional)	Translatory absolute Rotary absolute/modulo Translatory absolute	Optional
A3:	Positioning axis (optional) Orientation axis (optional) Belt axis (optional)	Translatory absolute Rotary absolute/modulo Translatory absolute	Optional
A4:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A5:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional

Transformation Rules

A6:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A7:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A8:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A9:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A10:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A11:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A12:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A13:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A14:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A15:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional
A16:	Orientation axis (optional) Belt axis (optional)	Rotary absolute/modulo Translatory absolute	Optional

Fig. 18-3: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 0 (Cartesian robot)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1
- K-0-0040, Transformation lengths, axis configuration, axis lengths not used
- K-0-0041, Transformation factors, axis configuration, axis coupling not used
- K-0-0121 .. K-0-0136, Coordinate names,

Coordinates: in the Cartesian coordinate system the robot has exactly as many degrees of freedom as axes in K-0-0008 were assigned (incl. belt axes).

Axes have to be assigned to all degrees of freedom.

Example of a 3-axis Cartesian system with an orientation axis and two belt axes

Name	Coordinate	Remark
K-0-0121	Coordinate 1:	Cx

Transformation Rules

K-0-0122	Coordinate 2:	Cy
K-0-0123	Coordinate 3:	Cz
K-0-0124	Coordinate 4:	Cc
K-0-0125	Coordinate 5:	Cb1
K-0-0126	Coordinate 6	Cb2

Fig. 18-4: K-0-0121 .. K-0-0126, Coordinate names, coordinates



Tool coordinate system

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- exact adjustment of the axis angle
- correct direction of rotation of the axes

If one of the these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

The following views are available for robot type 0 (Cartesian):

- [fig. 18-1 "Robot type 0, Cartesian robot, 3D perspective "](#) on page 256 and
- [fig. 18-2 "Robot type 0, a typical X-Y-Z-C Cartesian system"](#) on page 256.
- The zero positions for all axes can be freely selected for the Cartesian robot and can also be outside the travel range limits.
- The travel direction for up to 3 positioning axes always forms a right-handed Cartesian coordinate system.
- The zero points and the travel directions of the orientation axes are arbitrary.

18.3 Robot Type 07, 2-Axis Delta with 1 Optional Manual Axis

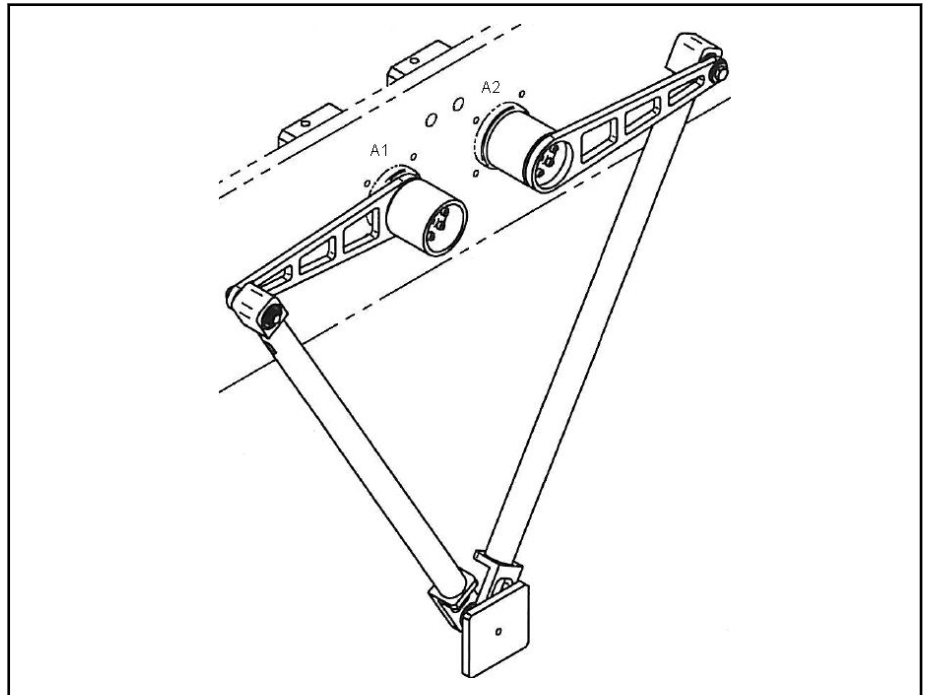


Fig. 18-5: Robot type 07, 2-axis delta

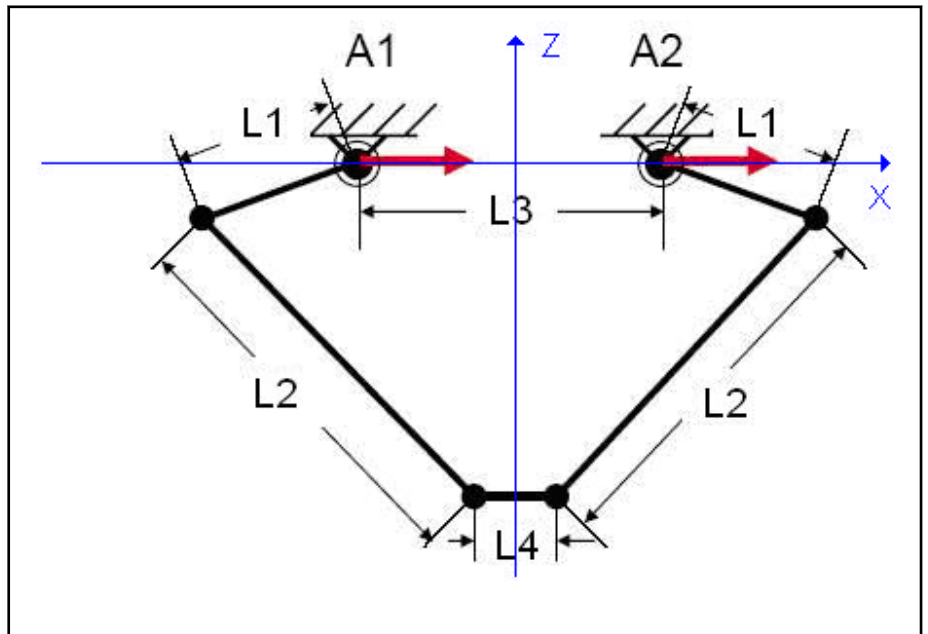


Fig. 18-6: Robot type 07, 2-axis delta (X-Z view)

Parameter settings

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

Transformation Rules

A4:	Belt axis (optional)	Translatory, modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-7: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, transformation scheme, Type = 7 (2-axis delta, bipod)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1

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

Fig. 18-8: K-0-0040, Transformation lengths, axis configuration, axis lengths

K-0-0041, Transformation factors, axis configuration, axis coupling, is not used

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cz)	
Coordinate 3:	Name of the third coordinate (e.g. Cc)	Optional
Coordinate 4:	Name of the fourth coordinate (e.g. Kbelt1)	Optional
Coordinate 5:	Name of the fifth coordinate (e.g. Kbelt2)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Kbelt3)	Optional
Coordinate 7:	Name of the seventh coordinate (e.g. Kbelt4)	Optional

Fig. 18-9: K-0-0121 .. K-0-0136, Coordinate names, coordinates

**Tool coordinate system**

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

Transformation Rules

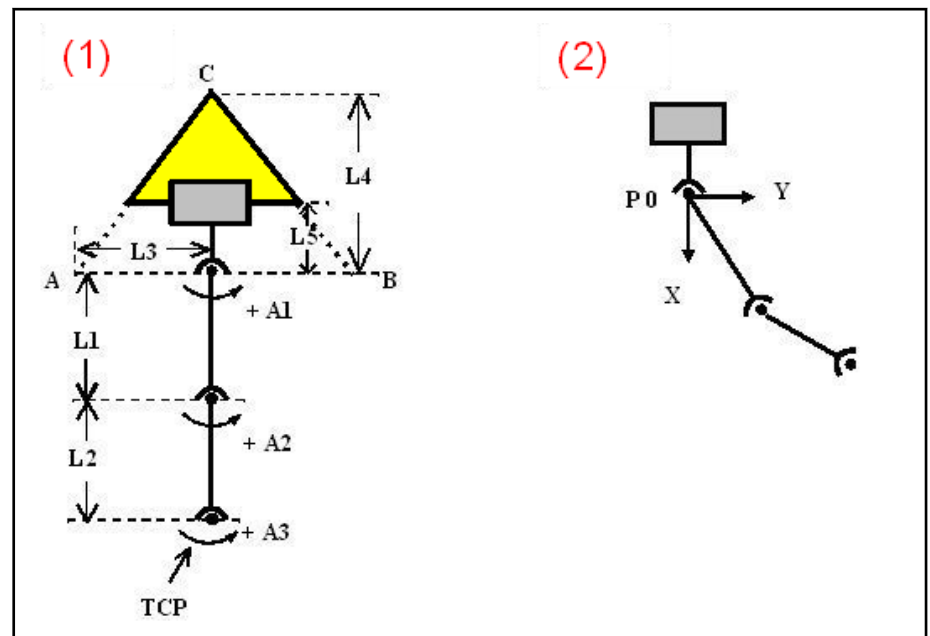
This can result in incorrect or unexpected motions.

fig. 18-5 "Robot type 07, 2-axis delta" on page 259 shows a bipod robot in 3D with the axis positions $A1=180^\circ$ and $A2 = 0^\circ$.

In zero position, the axes 1 and 2 (positioning axes) are horizontal towards the x-axis (see red arrow in fig. 18-6 "Robot type 07, 2-axis delta (X-Z view)" on page 259).

- The zero point for the third axis (orientation axis) is in the positive direction on the x-axis.
- The arm length L_1 is measured from the rotary center point to the center point of the rod retainer.
- The rod length L_2 is measured from the reference points on both sides.
- L_3 is the distance of the rotary center point of both arms.
- L_4 is the length of the mold 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 $A2$.

18.4 Robot Type 09, 3-Axis Scara with 1 Optional Manual Axis



(1) Top view with zero positions for all axes; yellow: prohibited area
(2) Zero point reference of all axes
Fig. 18-10: Robot type 09, 3-axis Scara with 1 optional manual axis

Parameter settings

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

Transformation Rules

A4:	Belt axis (optional)	Translatory, modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-11: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 09 (3-axis Scara)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1

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

Fig. 18-12: 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 3	Factor	Optional

Fig. 18-13: K-0-0041, Transformation factors, axis configuration, axis coupling

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	
Coordinate 3:	Name of the third coordinate (e.g. Cc)	Optional
Coordinate 4:	Name of the fourth coordinate (e.g. Cb1)	Optional
Coordinate 5:	Name of the fifth coordinate (e.g. Cb2)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Cb3)	Optional
Coordinate 7:	Name of the seventh (e.g. Cb4)	Optional

Fig. 18-14: K-0-0121 .. K-0-0128, Coordinate names, coordinates

**Tool coordinate system**

The transformation supports the following orientation offsets that have to be configured in the file TOOLS.DAT.

- O1 orientation offset of the TCP related to the tool (rotation along the z-axis)
- O2 0
- O3 0

Transformation Rules

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Exact determination of the coupling factors (if present)
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed. This can result in incorrect or unexpected motions.

fig. 18-10 "Robot type 09, 3-axis Scara with 1 optional manual axis" on page 261: (1) displays the zero positions of all axes in a top view.

- If the robot is in the position shown in the figures, all axes in the machine coordinates have to display zero.

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

- The figures also show arm lengths L_1 , L_2 and the positive rotary directions of the axes.
- A prohibited area can be defined with the lengths L_3 , L_4 , L_5 where the tool center point (TCP) may not enter.

This prevents collisions when the travel ranges of axes 1 and 2 are so large that the robot could run into its own support.

In ①, this prohibited area is shown in yellow.

- If the length $L_3=0$, range monitoring is not enabled.
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and L_1 , L_2 may not be equal to zero.
- The transformation can consider a mechanical coupling of axes 1 and 2 with axis 3.

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

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



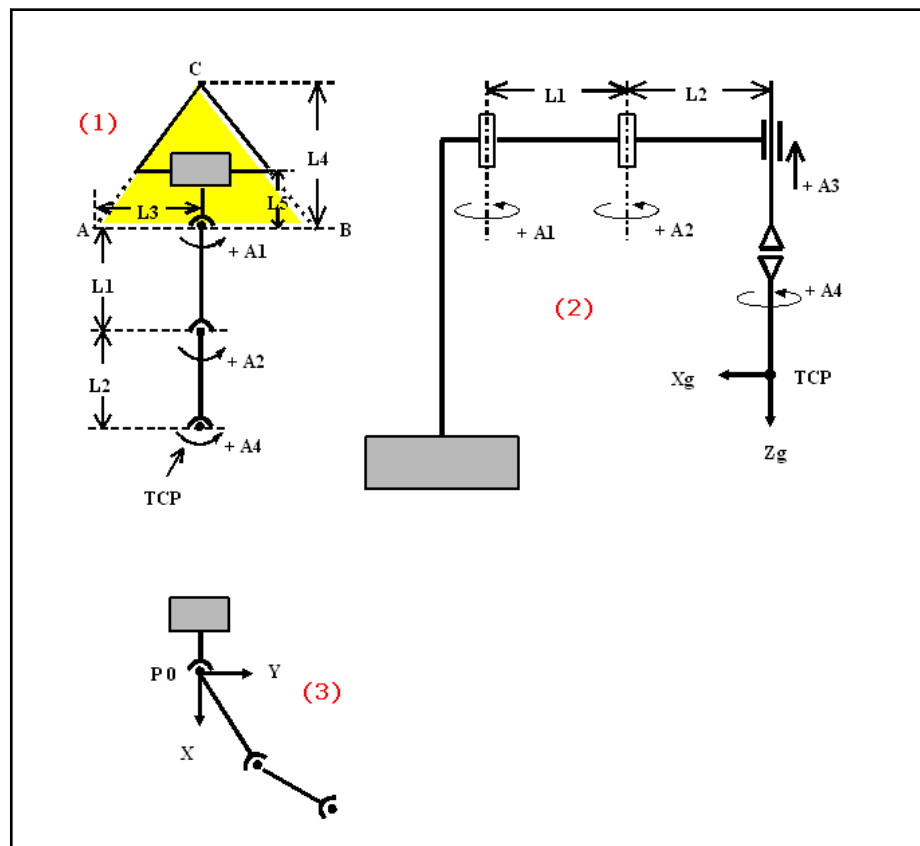
The maximum travel range of the axes 1 and 2 is +/-179 degrees. The travel range for the third axis is unlimited.

The respective world coordinate system is identified in fig. 18-10 "Robot type 09, 3-axis Scara with 1 optional manual axis" on page 261 (2) by the two main axes (x, y).

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

Transformation Rules

18.5 Robot Type 10, 4-Axis Scara with 1 Optional Manual Axis



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

Fig. 18-15: Robot type 10, 4-axis Scara with 1 optional manual axis

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Positioning axis (optional)	translatory	Optional
A4:	Orientation axis (optional)	rotary	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-16: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 10 (2-axis Scara)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1

Transformation Rules

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

Fig. 18-17: K-0-0040, Transformation lengths, axis configuration, axis lengths

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

Fig. 18-18: K-0-0041, Transformation factors, axis configuration, axis coupling

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	
Coordinate 3:	Name of the third coordinate (e.g. Cz)	Optional
Coordinate 4:	Name of the fourth coordinate (e.g. Cc)	Optional
Coordinate 5:	Name of the fifth coordinate (e.g. Cb1)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Cb2)	Optional
Coordinate 7:	Name of the seventh coordinate (e.g. Cb3)	Optional
Coordinate 8:	Name of the eighth coordinate (e.g. Cb4)	Optional

Fig. 18-19: K-0-0121 .. K-0-0128, Coordinate names, coordinates



Tool coordinate system

The transformation supports the following orientation offsets that have to be configured in the file TOOLS.DAT.

- O1 orientation offset of the TCP related to the tool (rotation along the z-axis)
- O2 0
- O3 0

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Exact determination of the coupling factors (if present)
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

Transformation Rules

This can result in incorrect or unexpected motions.

fig. 18-15 "Robot type 10, 4-axis Scara with 1 optional manual axis" on page 264: (1) displays the zero positions of all axes in top view and (2) in side view.

- If the robot is in the position shown in the figures, all axes in the machine coordinates have to display zero.
This can be ensured by setting the actual values of the reference point accordingly.
- In the zero position, axes 1, 2 and 4 are extended in forward direction.
- The zero point for the third axis can be freely chosen.
- The figures also show the arm lengths L_1 , L_2 and the positive directions of rotations of the axes.
- A prohibited area can be defined with the lengths L_3 , L_4 , L_5 where the tool center point (TCP) may not enter.
- This prevents collisions when the travel ranges of axes 1 and 2 are so large that the robot could run into its own support.
In ①, this prohibited area is shown in yellow.
- If the length $L_3=0$, range monitoring is not enabled.
- The lengths L_1 , L_2 , L_3 , L_4 , L_5 always have to be entered as positive number and L_1 , L_2 may not be equal to zero.
- The transformation can consider a mechanical coupling of the axes 1 and 2 with axis 4.
Such a coupling exists if axis 4 moves along when axes 1 and 2 travel and the measuring system for the fourth axis does not detect this motion. This is generally the case if the motor for the fourth axis is not directly on the axis, but instead on its support.
Set this coupling using the parameter K-0-0041 (see above).
- The transformation also consider a mechanical coupling of the fourth axis with the third axis.
Such a coupling exists if the third world coordinate (z-axis) changes when axis 4 travels and the measuring system for the third axis does not detect 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 world 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 world coordinate system is identified by the 3 main axes (x, y, z).

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

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

18.6 Robot Type 12, 2-Axis Parallelogram

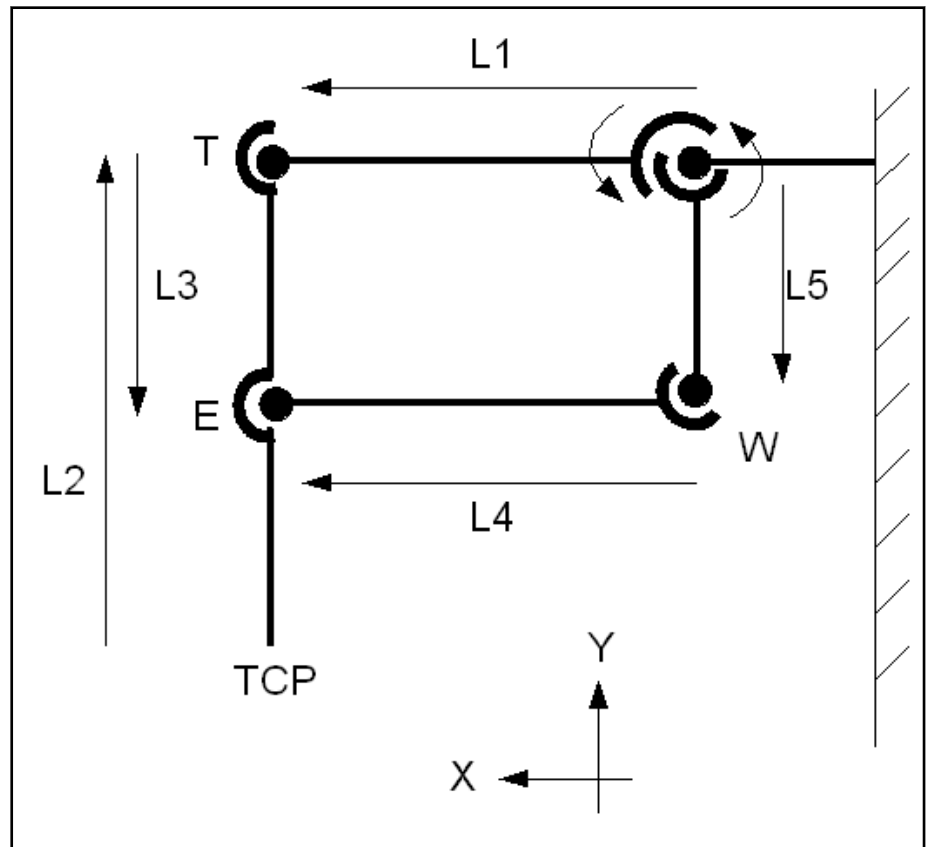


Fig. 18-20: Robot type 12, 2-axis parallelogram

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees
A3:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-21: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 12 (2-axis parallelogram)
- K-0-0032, Referencing all axes = 0

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

Fig. 18-22: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation Rules

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	

Fig. 18-23: K-0-0121 .. K-0-0128, Coordinate names, coordinates

**Tool coordinate system**

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct determination of signs of individual lengths
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-20 "Robot type 12, 2-axis parallelogram" on page 267 shows

- The schematic kinematic representation for axes in zero position.
- The figures also show the arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes.
- 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 have to be unequal to zero.
- The respective world coordinate system is identified by the two main axes (x, y).
- The world coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

18.7 Robot Type 13, 2-Axis Parallelogram

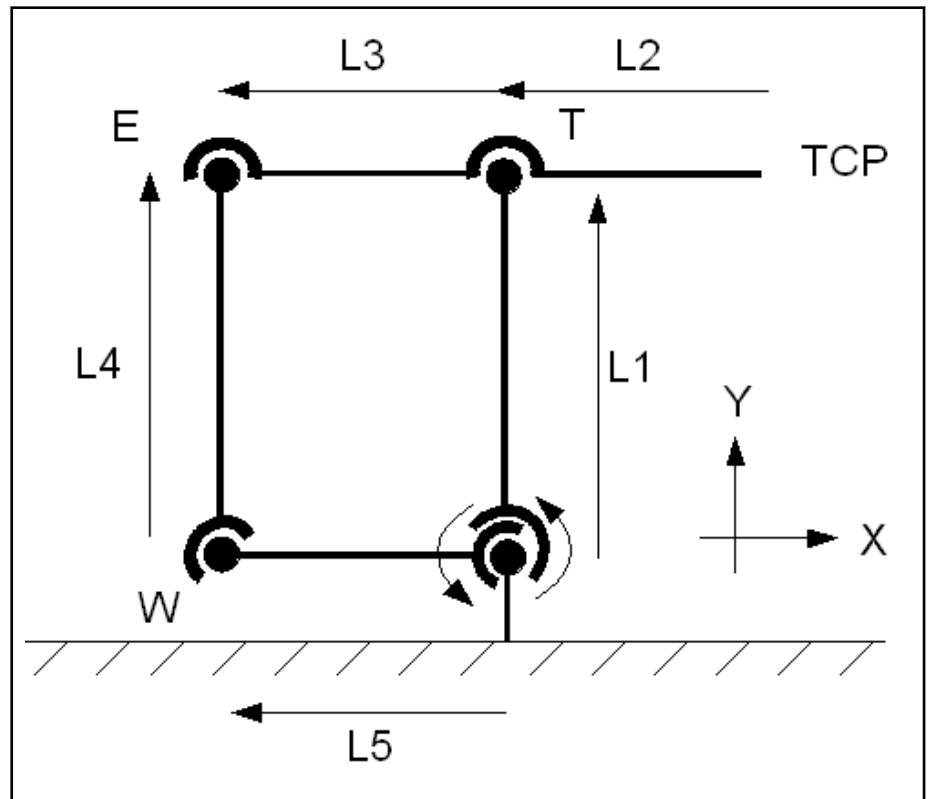


Fig. 18-24: Robot type 13, 2-axis parallelogram

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees
A3:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-25: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 13 (2-axis parallelogram)
- K-0-0032, Referencing all axes = 0

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

Fig. 18-26: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation Rules

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	

Fig. 18-27: K-0-0121 .. K-0-0128, Coordinate names, coordinates

**Tool coordinate system**

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct determination of signs of individual lengths
- Correct direction of rotation of the axes

If one of the these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-24 "Robot type 13, 2-axis parallelogram" on page 269 shows

- The schematic kinematic representation for axes in zero position.
- The figures also show the arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes.
- 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 world coordinate system is identified by the two main axes (x, y).
- The world coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

18.8 Robot Type 18, Contact Kinematics

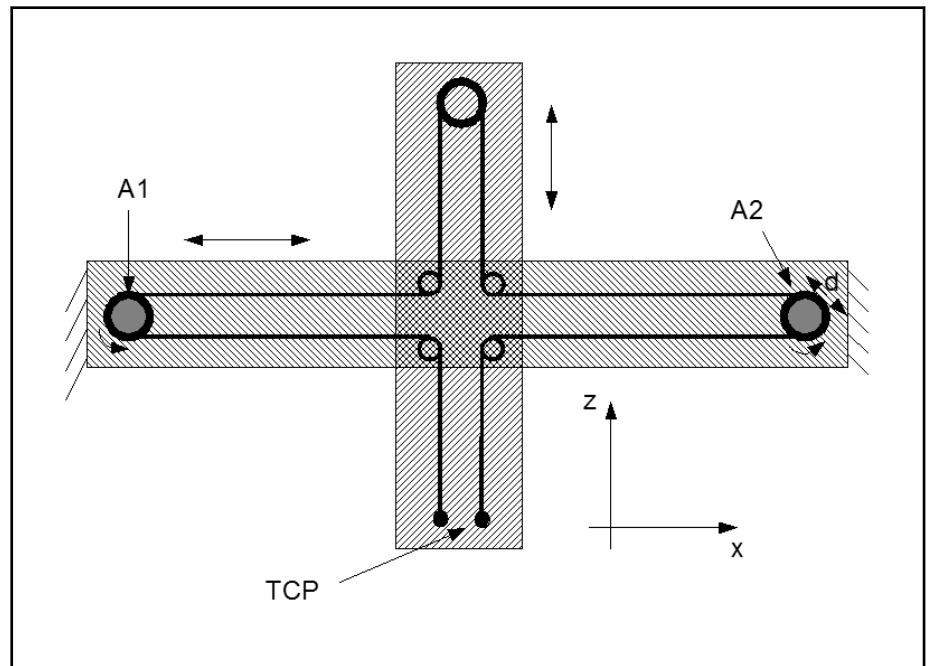


Fig. 18-28: Robot type 18, contact kinematics

Parameter settings

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

Fig. 18-29: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, type = 18 (Cartesian kinematics).
- K-0-0032, Referencing all axes = 0.
- K-0-0040, Transformation lengths, axis configuration, axis lengths are not used.

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cz)	

Fig. 18-30: K-0-0121 .. K-0-0128, Coordinate names, coordinates



Tool coordinate system

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- exact measurement of diameter d of driving axes

Transformation Rules

- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-28 "Robot type 18, contact kinematics" on page 271 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.
- The set travel range limits of the axes may not exceed the mechanical travel ranges.
- The respective world coordinate system is identified by the two main axes (x, z).
- The gear of both axes has to be set so that the radius ($d/2$) is entered in mm on the load side (S-0-0121 = 1, S-0-0122 = $d/2$).

18.9 Robot Type 19, 2-Axis Parallelogram V1

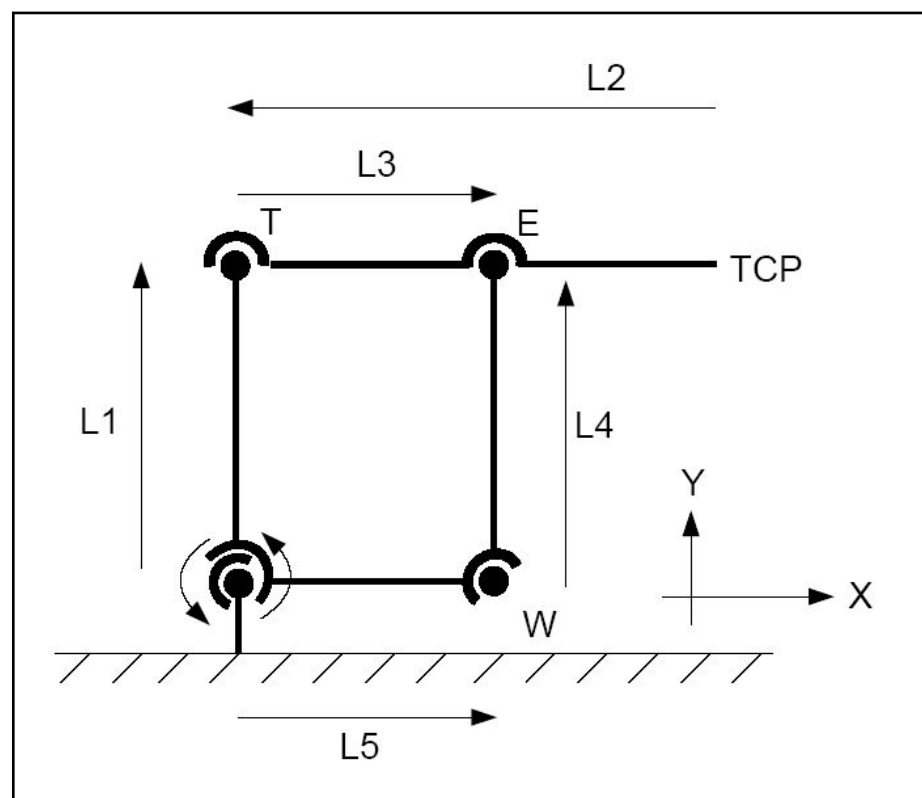


Fig.18-31: Robot type 19, 2-axis parallelogram V1

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees
A3:	Positioning axis	Translatory absolute	Optional

Transformation Rules

A4:	Orientation axis	Rotary mod./abs.	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-32: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 19 (2-axis parallelogram)
- K-0-0032, Referencing all axes = 0

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

Fig. 18-33: K-0-0040, Transformation lengths, axis configuration, axis lengths

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	
Coordinate 3:	Name of the third coordinate (e.g. Cz)	
Coordinate 4:	Name of the fourth coordinate (e.g. Cb)	
Coordinate 5:	Name of the fifth coordinate (e.g. Cbelt)	

Fig. 18-34: K-0-0121 .. K-0-0128, Coordinate names, coordinates



Tool coordinate system

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct determination of signs of individual lengths
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-31 "Robot type 19, 2-axis parallelogram V1" on page 272 shows

- The schematic kinematic representation for axes in zero position.

Transformation Rules

- The figures also show the arm lengths L_1 , L_2 , L_3 , L_4 , L_5 and the positive directions of rotation of the axes.
- 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 world coordinate system is identified by the two main axes (x, y).
- The world coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

18.10 Robot type 20, 3-Axis Delta with 1 Optional Manual Axis,

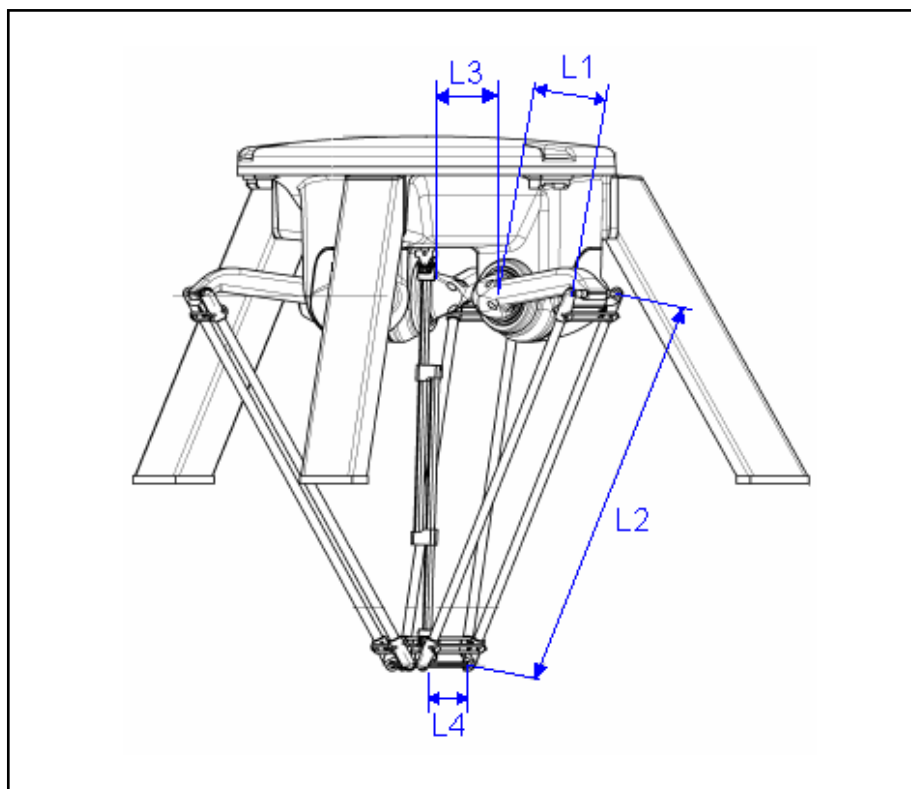


Fig.18-35: Robot type 20, 3-axis Delta with 1 optional manual axis

Transformation Rules

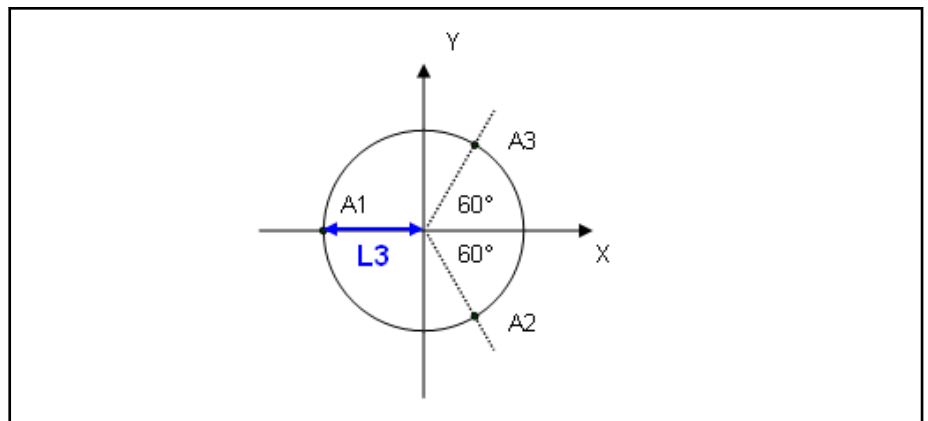


Fig. 18-36: Radius L3 of the motor base and the coordinate system (X-Y view)
The three axes A1 to A3 are all displaced by 120° from each other.

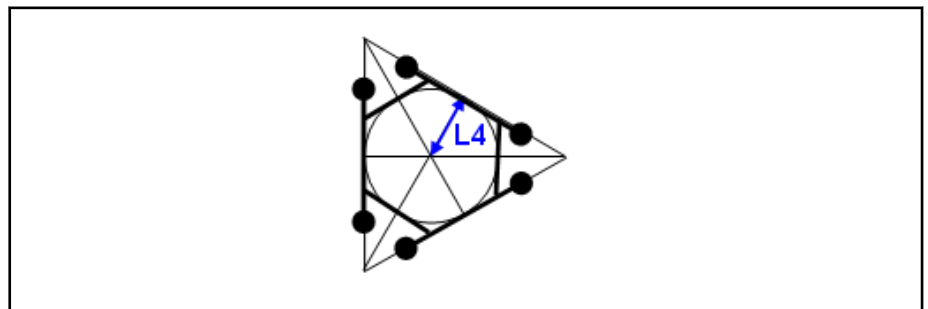
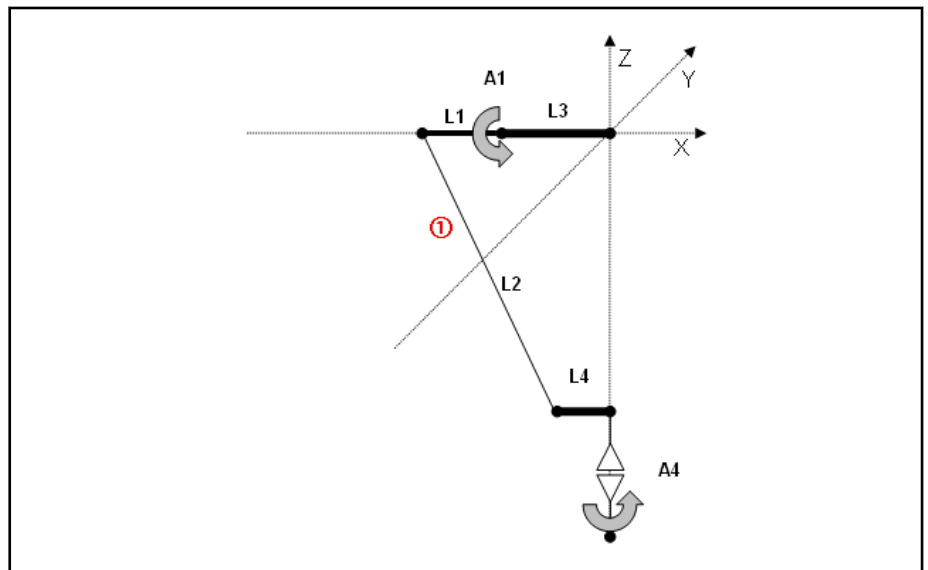


Fig. 18-37: Radius L4 of the tool plate



① Connection rod

Fig. 18-38: Representation of the zero position of the first axis

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

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 90 degree
A2:	Positioning axis	Rotary absolute	+/- 90 degree

Transformation Rules

A3:	Positioning axis	Rotary absolute	+/- 90 degree
A4:	Orientation axis (optional)	Rotary abs./modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-39: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 20 (3-axis Delta)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm at the motor	mm or in (see K-0-0054)	>0
L2:	Length of rod to mold plate	mm or in (see K-0-0054)	>0
L3:	Radius of the circle defined by the three rotation points of the arms	mm or in (see K-0-0054)	>0
L4:	Radius of the base plate	mm or in (see K-0-0054)	>0
L5:	Has to be 0		=0
L6:	Has to be 0		=0

Fig. 18-40: K-0-0040, Transformation lengths, axis configuration, axis lengths



If L5 and L6 $\neq$ 0, translatorily scaled axes are used. Refer to [Translatorily-scaled positioning axes on page 277](#).

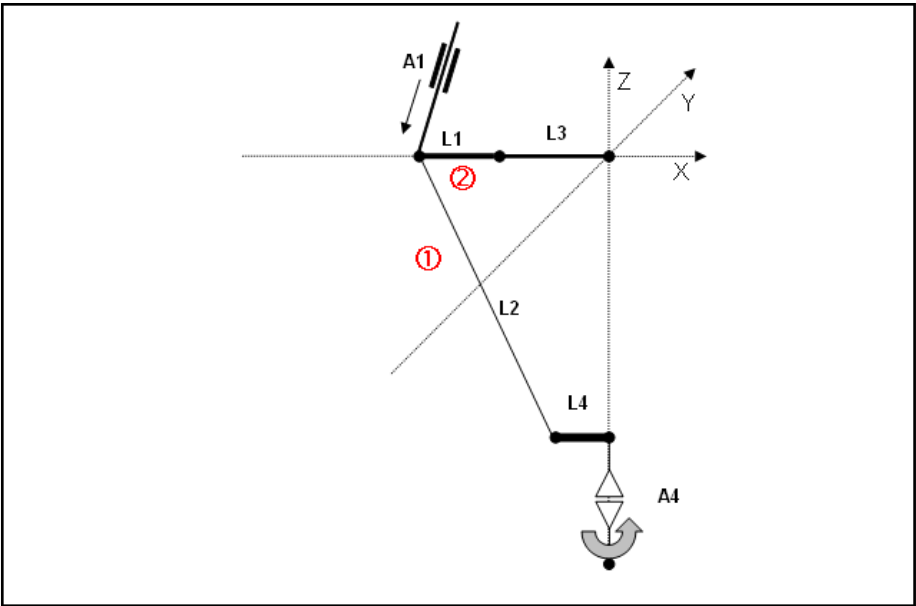
K-0-0041, Transformation factors, axis configuration, axis coupling, is not used

Name	Coordinate	Remark
K-0-0121	Coordinate 1:	Cx
K-0-0122	Coordinate 2:	Cy
K-0-0123	Coordinate 3:	Cz
K-0-0124	Coordinate 4:	Cc
K-0-0125	Coordinate 5:	Cb1
K-0-0126	Coordinate 6:	Cb2
K-0-0127	Coordinate 7:	Cb3
K-0-0128	Coordinate 8:	Cb4

Fig. 18-41: K-0-0121 .. K-0-0128, Coordinate names, coordinates

	 Tool coordinate system The transformation does not support any orientation offsets configured in the file TOOLS.DAT.
Axis directions of rotation and zero positions	The basic requirements to perform the coordinate transformation correctly are: • Exact setting of the axis zero points • Exact measurement of the arm lengths or center offsets • Correct direction of rotation of the axes If one of the these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed. This can result in incorrect or unexpected motions. fig. 18-35 "Robot type 20, 3-axis Delta with 1 optional manual axis" on page 274 shows a Delta robot at zero position of all axes, i.e. if the robot is in the position shown in the figure, the joint coordinates for all of axes have to be zero. This can be ensured by setting the actual values of the reference point accordingly. In the zero position, the axes 1, 2 and 3 (positioning axes) are horizontal. The zero point for the fourth axis (orientation axis) is in the positive direction of the x-axis. The arm length L_1 is measured from the rotary center point to the center point of the rod retainer. The rod length L_2 is measured from the reference points on both sides. L_3 is the radius of the perimeter formed by the three motor spindles (or gears) measured at the rotary center points of the arms (see fig. 18-36 "Radius L3 of the motor base and the coordinate system (X-Y view)" on page 275). L_4 is the radius of the incircle of the equilateral triangle on the tool plate. The equilateral triangle is defined by the reference points of the rods. (see fig. 18-37 "Radius L4 of the tool plate" on page 275). The direction of rotation of the positioning axes A1 to A3 is positive if the arms move down into z-direction. The orientation axis A4 (rotation along the z-axis) rotates positively counterclockwise. The Cartesian zero point for the Delta robot is at the center point of the perimeter defined by the three motor spindles (see also fig. 18-36 "Radius L3 of the motor base and the coordinate system (X-Y view)" on page 275).
Translatorily-scaled positioning axes	The transformation can also be operated with translatorily-scaled positioning axes.

Transformation Rules



- ① Connection rod
- ② Arm

Fig. 18-42: Representation of the zero position for translatorily-scaled positioning axes

When the arms are in a horizontal position, the positions of the axes A1 to A3 have to be zero. The positive motion of the positioning axes pushes the corresponding arm down.

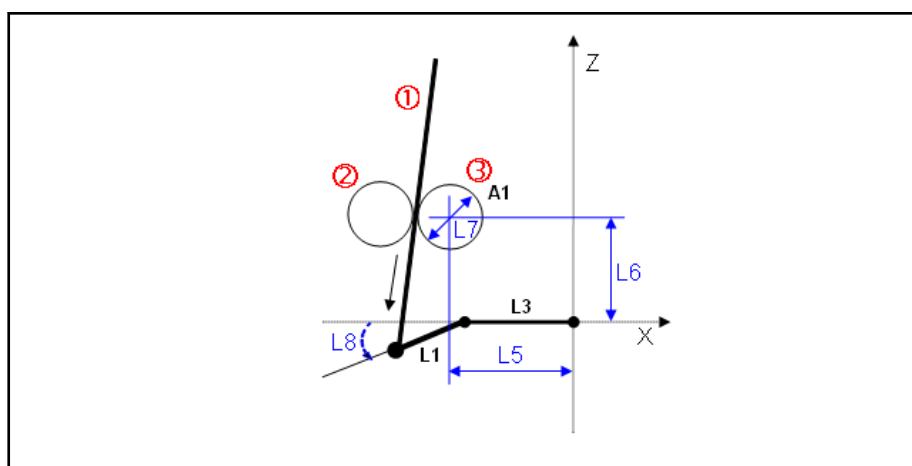
Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Translatory absolute	
A2:	Positioning axis	Translatory absolute	
A3:	Positioning axis	Translatory absolute	

Fig. 18-43: K-0-0008, Axis configuration list, axis assignment

The positioning axes A1 to A3 can also be translatory.

- K-0-0040, Transformation lengths, axis configuration, axis lengths; in addition to the K-0-0040 entries L1 to L4 described above, L5 to L8 have to be set.

Transformation Rules



- ① Rack
- ② Pressure roller
- ③ Drive roller

Fig. 18-44: Further settings for translatorily-scaled positioning axes

Transformation length	Axis configuration	Measuring unit	Axis length
L5:	Horizontal distance from the rack drive roller to the origin of the coordinate system	mm or in (see K-0-0054)	>0
L6:	Vertical distance from the drive roller to the x-y plane	mm or in (see K-0-0054)	>0
L7:	Diameter of the drive roller	mm or in (see K-0-0054)	>0
L8:	Phase correction	Degrees	-45 ... +45

Fig. 18-45: K-0-0040, Transformation lengths, axis configuration, axis lengths



If L5 and L6 ≠ 0, translatorily-scaled axes are used.

Transformation Rules

18.11 Robot Type 21, 2-Axis Parallelogram V2

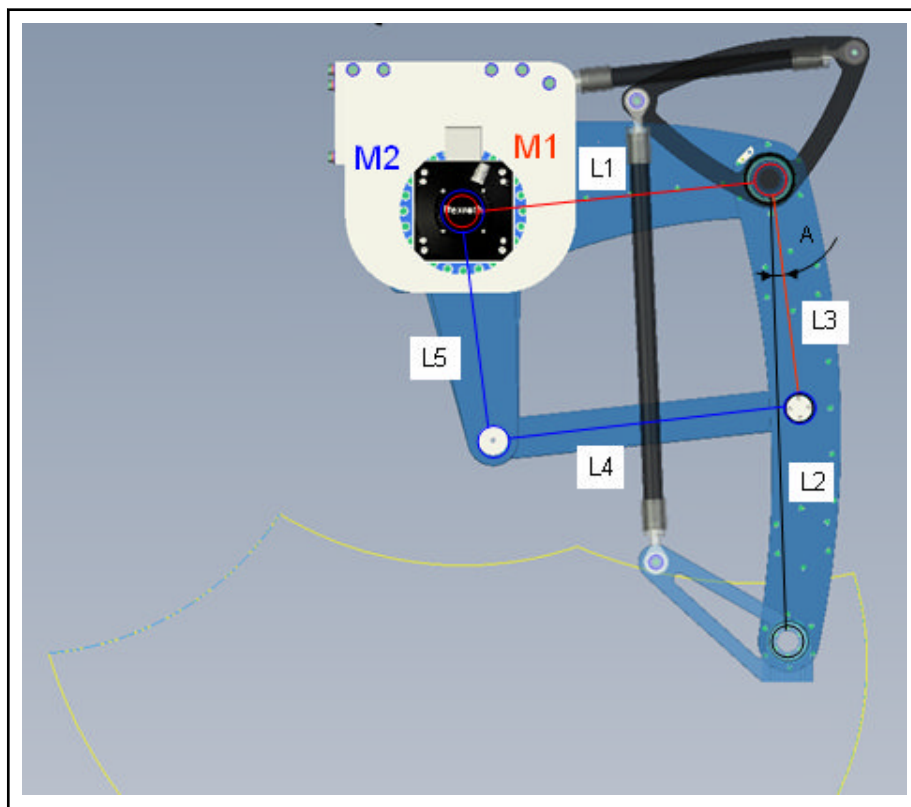


Fig.18-46: Robot type 21, 2-axis parallelogram V2

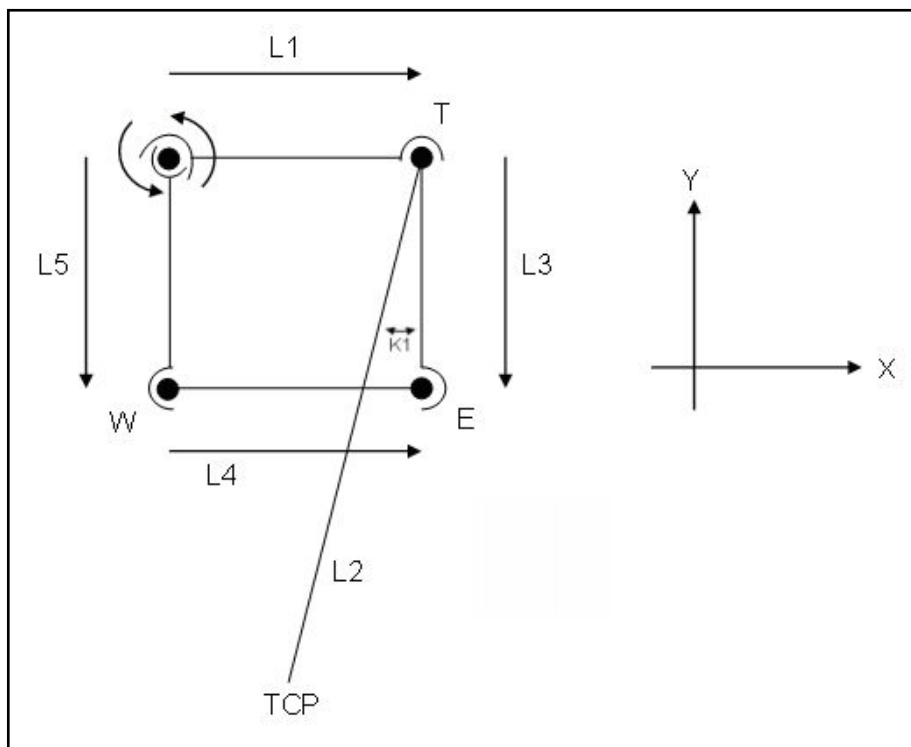


Fig.18-47: Robot type 21, 2-axis parallelogram V2, in the zero position of the axes

Parameter settings

Transformation Rules

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 89 degrees
A2:	Positioning axis	Rotary absolute	+/- 89 degrees
A3:	Positioning axis	Translatory absolute	Optional
A4:	Orientation axis	Rotary abs./mod.	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-48: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, Type = 21 (2-axis parallelogram)
- K-0-0032, Referencing all axes = 0

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

Fig. 18-49: K-0-0040, Transformation lengths, axis configuration, axis lengths

Transformation factor	Configuration	Measuring unit	Remark
K1:	Angle between L ₂ and L ₃	Degrees	

Fig. 18-50: K-0-0041, Transformation factors

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	
Coordinate 3:	Name of the third coordinate (e.g. Cz)	
Coordinate 4:	Name of the fourth coordinate (e.g. Cb)	
Coordinate 5:	Name of the fifth coordinate (e.g. Cbelt)	

Fig. 18-51: K-0-0121 .. K-0-0128, Coordinate names, coordinates



Tool coordinate system

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points

Transformation Rules

- Exact measurement of the arm lengths or center offsets
- Correct determination of signs of individual lengths
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-47 "Robot type 21, 2-axis parallelogram V2, in the zero position of the axes" on page 280 shows the schematic kinematic representation for axes in zero position.

- The figures also show the arm lengths L_1 , L_2 , L_3 , L_4 , L_5 . The positive direction of rotations of the axes are counterclockwise.
- 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 world coordinate system is identified by the two main axes (x, y).
- The world coordinate zero point is located in the intersection of the two length specifications L_1 and L_5 .

18.12 Robot Type 26, Robot with Offset Wrist with Six (Seven) Axes

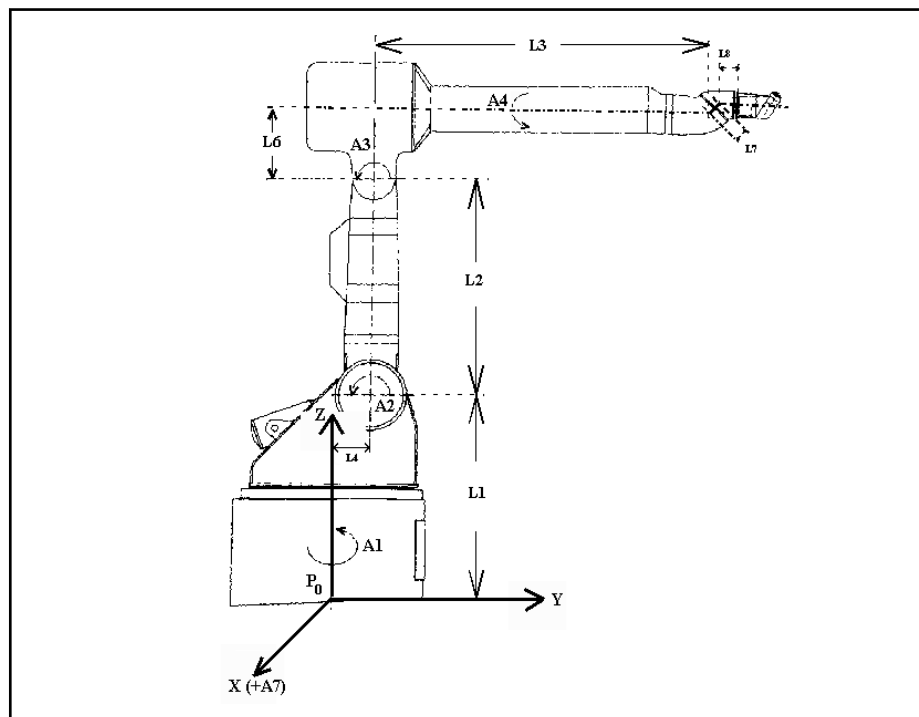


Fig. 18-52: Robot type 26, robot with offset wrist with 6 (7) axes, zero position

Transformation Rules

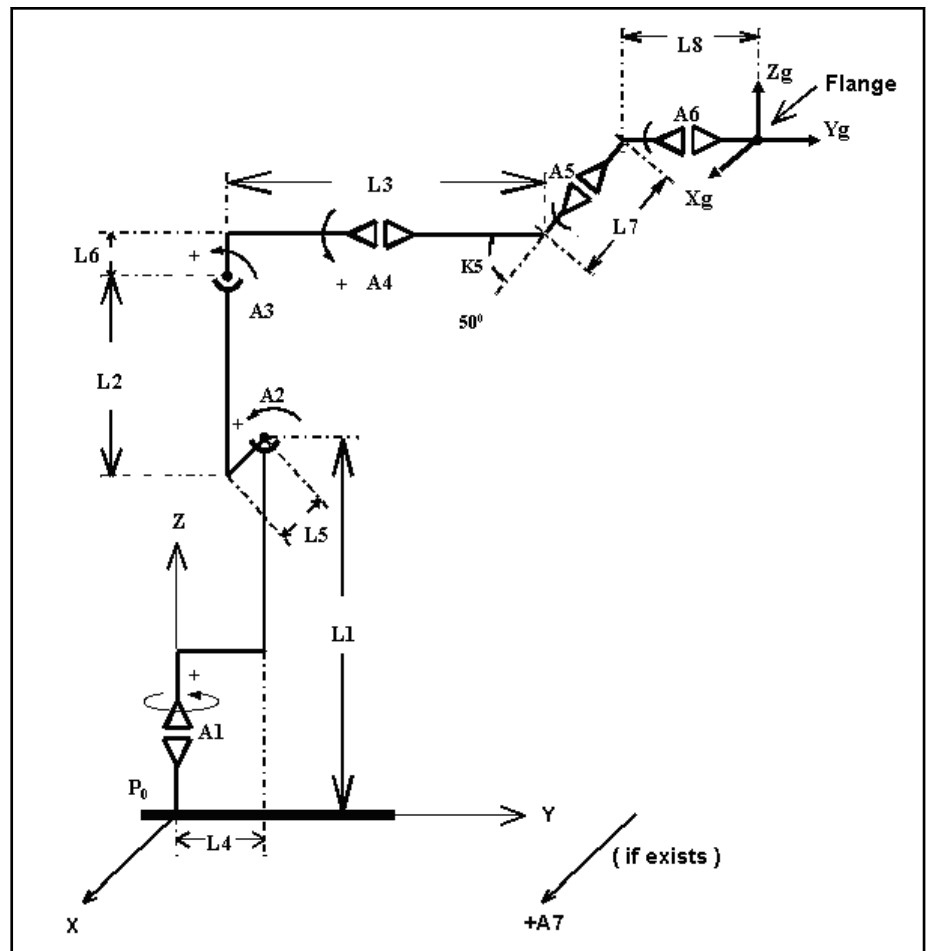


Fig. 18-53: Robot type 26, robot with offset wrist with 6 (7) axes, zero position, schematic

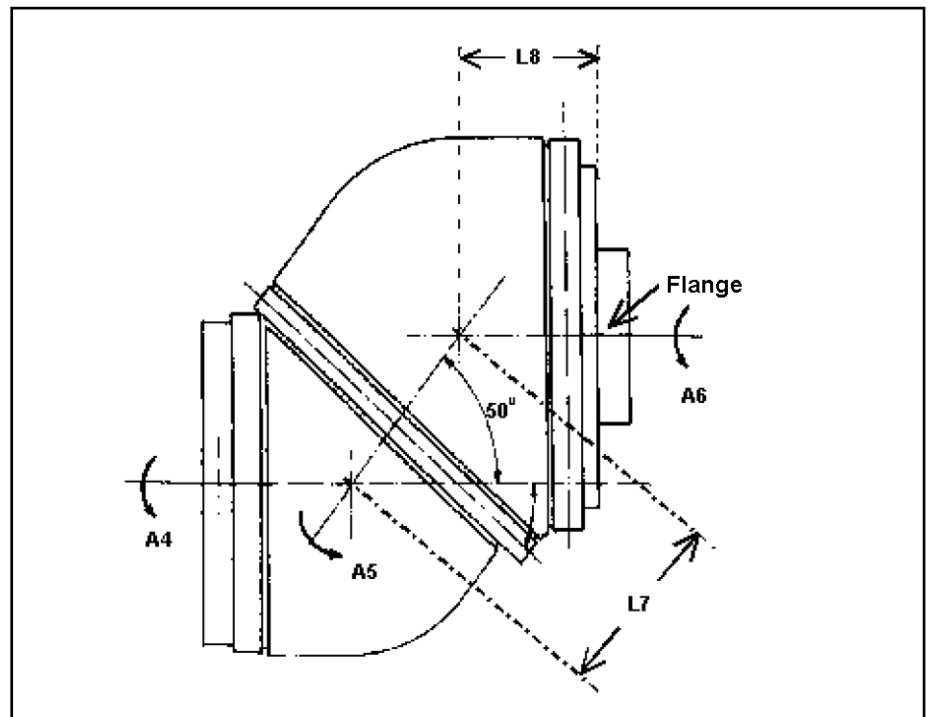


Fig. 18-54: Robot type 26, robot with offset wrist with 6 (7) axes, wrist

Transformation Rules

- Zero position** The basic requirements to perform the coordinate transformation correctly are:
- Exact setting of the axis zero points
 - Exact determination of the coupling factors (if present)
 - correct directions of rotation of the axes

⚠ CAUTION

If one of the these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed. This can cause incorrect and unexpected robot motions.

The respective world coordinate system is identified by the three main axes (x, y, z) in [fig. 18-53 "Robot type 26, robot with offset wrist with 6 \(7\) axes, zero position, schematic" on page 283](#) The zero point is at the base point P0 of the robot.



When using this transformation with PROGR_SLOPE, the path for spatial blending is not calculated by the control. The user has to program the tangential transitions at block switching.

Settings in the axis configuration list, axis assignment (K-0-0008)

The following settings have to be made in the kinematics and in the axis configuration:

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Positioning axis	Rotary absolute	+/- 90 degree
A4:	Orientation axis	Rotary absolute	Unlimited
A5:	Orientation axis	Rotary absolute	Unlimited
A6:	Orientation axis	Rotary absolute	Unlimited
A7:	Orientation axis	Translatory absolute	parallel to the belt (optional)

Fig. 18-55: K-0-0008, Axis configuration list, axis assignment

26 has to be set as transformation scheme.

Transformation scheme, type (K-0-0031)

Referencing all axes (K-0-0032)

Select 1 as value for this parameter.

IPOS; update at coordinate change (K-0-0033)

Select 1 as value for this parameter.

Transformation lengths, axis configuration, axis lengths (K-0-0040)

Find all data in [fig. 18-53 "Robot type 26, robot with offset wrist with 6 \(7\) axes, zero position, schematic" on page 283](#):

Transformation length	Meaning	Measuring unit	Restriction
L1:	Arm length L_1	mm or in (see K-0-0054)	≥ 0
L2:	Arm length L_2	mm or in (see K-0-0054)	> 0
L3:	Arm length L_3	mm or in (see K-0-0054)	> 0

Transformation Rules

L4:	Eccentricity L_4	mm or in (see K-0-0054)	Without restriction
L5:	Eccentricity L_5	mm or in (see K-0-0054)	Without restriction
L6:	Eccentricity L_6	mm or in (see K-0-0054)	Without restriction
L7:	Arm length L_7	mm or in (see K-0-0054)	≥ 0
L8:	Arm length L_8	mm or in (see K-0-0054)	≥ 0

Fig. 18-56: K-0-0040, Transformation lengths, axis configuration, axis lengths

The eccentricities L_4 , L_5 , L_6 can be positive or negative:

- $L_4 > 0$: Eccentricity in (+y) direction
- $L_4 < 0$: Eccentricity in (-y) direction
- $L_5 > 0$: Eccentricity in (+x) direction
- $L_5 < 0$: Eccentricity in (-x) direction
- $L_6 > 0$: Eccentricity in (+z) direction
- $L_6 < 0$: Eccentricity in (-z) direction

Transformation factors, axis configuration, axis coupling (K-0-0041)

Coupling factor	Meaning	Restriction
K1:	C_{45} mechanical coupling between the axes 4/5	Without restriction
K2:	C_{46} mechanical coupling between the axes 4/6	Without restriction
K3:	C_{56} mechanical coupling between the axes 5/6	Without restriction
K4:	C_{23} mechanical coupling between the axes 2/3	Without restriction
K5:	Bending angle of the fifth axis (50°)	Without restriction
K6:	Max. sum of the manual axes (1080°)	Without restriction
K7:	Abort accuracy for iteration	≥ 0
K8:	Max. number of iteration steps + selection of the orientation definition + activation of the tensor interpolation	≥ 0

Fig. 18-57: K-0-0041, Transformation factors, axis configuration, axis coupling

Coupling factors C1 to C4

If mechanical axis couplings exist, the respective coupling factors have to be determined. They are defined as follows:

$$C_{23} = \Delta G_3 / \Delta A_2$$

$$C_{45} = \Delta G_5 / \Delta A_4$$

$$C_{46} = \Delta J_6 / \Delta A_4$$

$$C_{56} = \Delta G_6 / \Delta A_5$$

A_i is used for the axis angle and J_i for the joint angle detected by the measuring system. That means that the coupling factor C_{ij} indicates the effect on the joint j when axis i changes without notifying the measuring system j about this change.

Coupling factor C5

Angle by which the robot hand (50°) is bended, [fig. 18-53 "Robot type 26, robot with offset wrist with 6 \(7\) axes, zero position, schematic" on page 283](#)

Transformation Rules

- Coupling factor C6** Max. sum of the three manual axes (1080°) If the value is exceeded, there is a motion abort with the message "manual axis rotation exceeded"
- Coupling factor C7** Abort accuracy in degrees to iterate the inverse position kinematics. If the value is smaller than 1 (e.g. 0.01), the same abort accuracy is applied to the position (1, 2, 3) and orientation axes (4, 5, 6). If the value is higher than 1, different abort criteria for position and orientation axes can be applied: The decimal values specifies the abort accuracy. The abort accuracy of the orientation axes results from the decimal value multiplied by the integer value (e.g. 20.01 => position accuracy = 0.01° orientation accuracy = 20 x 0.01° = 0.2°)
- Coupling factor C8** The first and the second place of C8 (indicates the maximum number of iterations allowed (99 max., 8 recommended). If no convergence has been reached after the set number of iterations, the motion is aborted and the message "Iteration divergent" is output. The third and fourth place of C8 (WC orientation type) specifies the definition of the orientation angle (world coordinates 4, 5, 6). If the fifth place of C8 is 1, the tensor interpolation is enabled.

Example:

- K8 = 8
 - Maximum of eight iterations
 - WC orientation type 0
 - No tensor interpolation
- K8 = 312
 - maximum of twelve iterations
 - WC orientation type 3
 - No tensor interpolation
- C8 = 10807
 - Maximum of seven iterations
 - WC orientation type 8
 - With tensor interpolation

The following definition applies for the WC orientation type = 0, 4:

- WC[4] = rotation along the spatially fixed z-axis
- WC[5] = rotation along the resulting y'-axis
- WC[6] = rotation along the resulting x''-axis

The following definition applies for the WC orientation type = 1, 5:

- WC[4] = rotation along the spatially fixed z-axis
- WC[6] = rotation along the resulting x'-axis
- WC[5] = rotation along the resulting z''-axis

The following definition applies for the WC orientation type = 2, 6:

- WC[4] = rotation along the spatially fixed z-axis
- WC[5] = rotation along the resulting x'-axis
- WC[6] = rotation along the resulting y''-axis

The following definition applies for the WC orientation type = 3, 7:

- WC[4] = rotation along the spatially fixed y-axis
- WC[5] = rotation along the resulting x'-axis
- WC[6] = rotation along the resulting z''-axis

Transformation Rules

The following definition (Euler angle) applies for the WC orientation type = 8:

- WC[4] = rotation along the spatially fixed x-axis
- WC[5] = rotation along the resulting y'-axis
- WC[6] = rotation along the resulting z"-axis

The world coordinates 4, 5 and 6 are not restricted for the WC orientation types 0, 1, 2 and 3. The types 4, 5, 6 and 7 are subject to the following restrictions:

- $-90^\circ < Wc[4] < 90^\circ$
- $-180^\circ < Wc[4] < 180^\circ$
- $-180^\circ < Wc[6] < 180^\circ$

The following limitations apply for the "classic" Euler angles (type 8):

- $-180^\circ < Wc[4] < 180^\circ$
- $0^\circ < Wc[5] < 180^\circ$
- $-180^\circ < Wc[6] < 180^\circ$

The tensor interpolation is enabled via a 5-digit C_8 . The spatial orientations (WC[4], WC[5], WC[6]) at linear motions are interpolated via a rotary sensor in automatic mode. For the tensor interpolation, one of the restricted WC orientation types has to be selected via the coupling factor C8 (i.e. WC orientation type).

18.13 Robot Type 27, Robot with Offset Wrist with Six Axes + Three Linear Axes

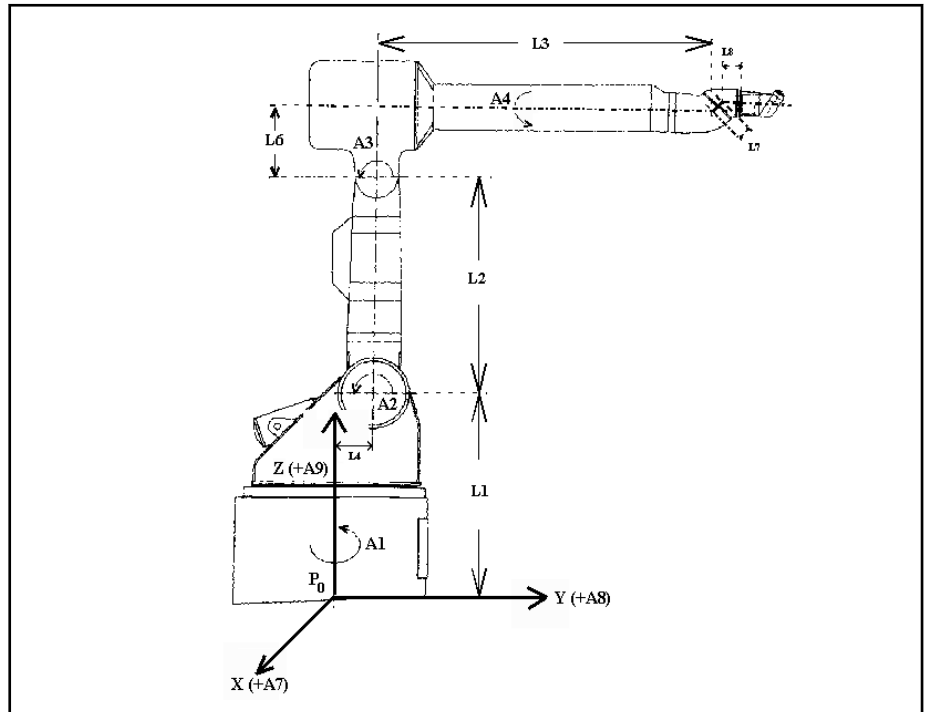


Fig. 18-58: Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position

Transformation Rules

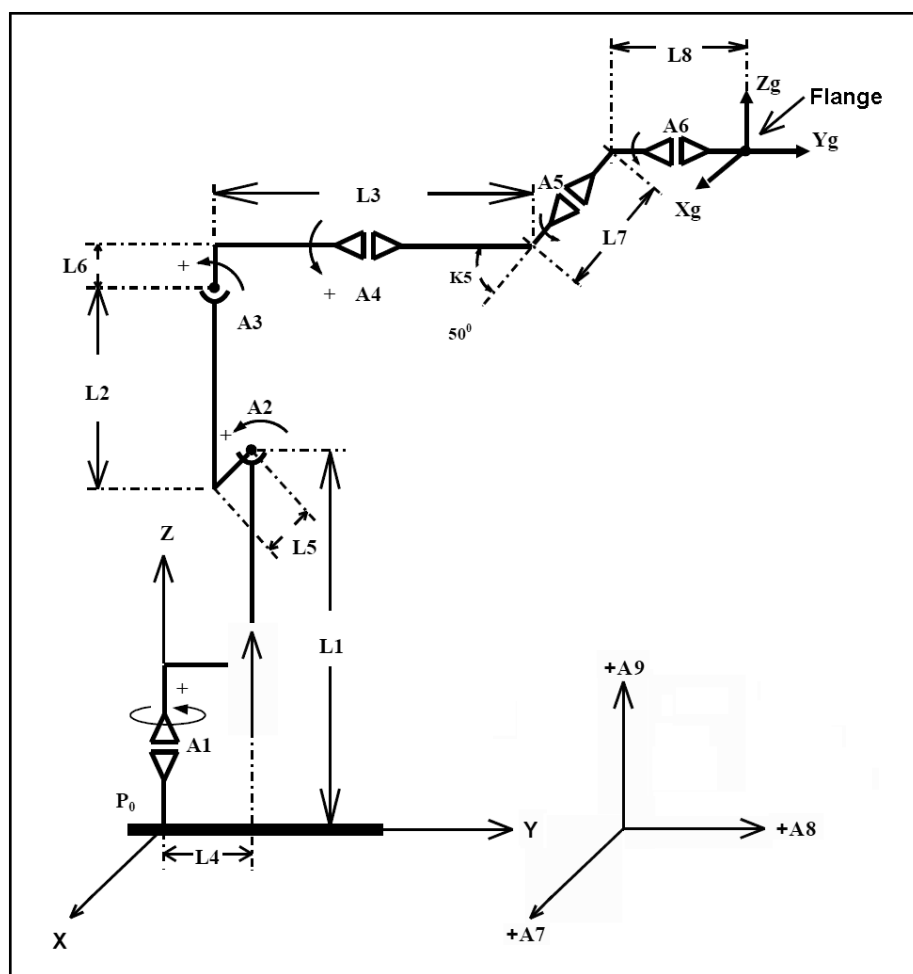


Fig. 18-59: Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position, schematic

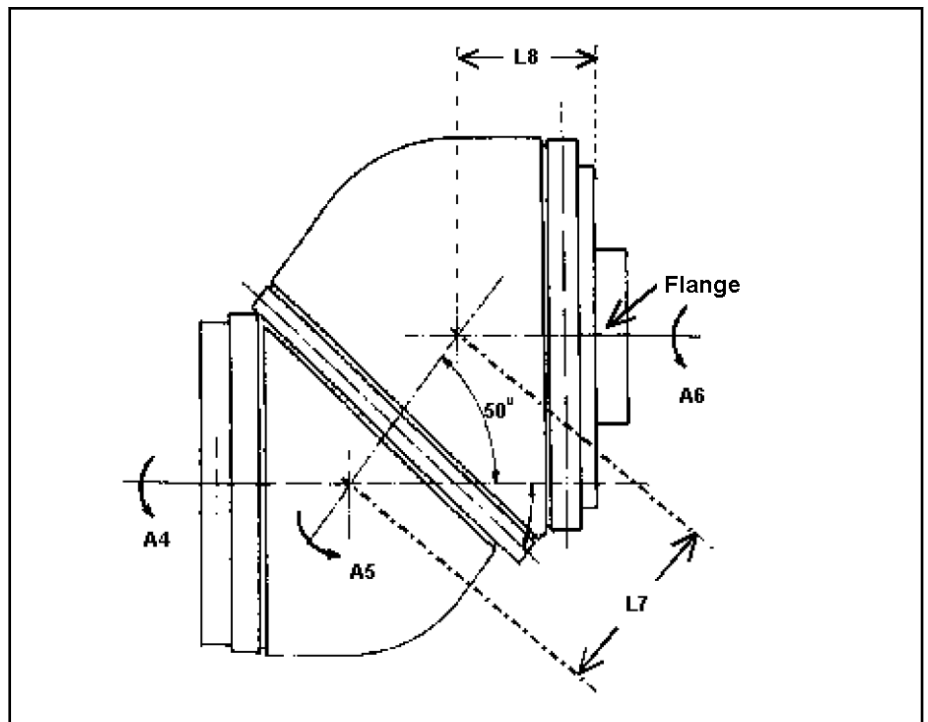


Fig. 18-60: Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position, wrist

Zero position

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact determination of the coupling factors (if present)
- correct directions of rotation of the axes

⚠ CAUTION

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed. This can cause incorrect and unexpected robot motions.

The respective world coordinate system is identified by the three main axes (x, y, z) in fig. 18-59 "Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position, schematic" on page 288. The zero point is at the base point P0 of the robot.



When using this transformation with PROGR_SLOPE, the path for spatial blending is not calculated by the control. The user has to program the tangential transitions at block switching.

Settings in the axis configuration list, axis assignment (K-0-0008)

The following settings have to be made in the kinematics and in the axis configuration:

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 179 degrees
A2:	Positioning axis	Rotary absolute	+/- 179 degrees
A3:	Positioning axis	Rotary absolute	+/- 90 degree

Transformation Rules

A4:	Orientation axis	Rotary absolute	Unlimited
A5:	Orientation axis	Rotary absolute	Unlimited
A6:	Orientation axis	Rotary absolute	Unlimited
A7:	Orientation axis	Translatory absolute	Parallel to the x-axis
A8:	Orientation axis	Translatory absolute	parallel to the y-axis
A9:	Orientation axis	Translatory absolute	Parallel to the z-axis

Fig. 18-61: K-0-0008, Axis configuration list, axis assignment

27 has to be set as transformation scheme.

Transformation scheme, type
(K-0-0031)

Referencing all axes (K-0-0032)

IPOS; update at coordinate
change (K-0-0033)

Transformation lengths, axis configuration,
axis lengths (K-0-0040)

Select 1 as value for this parameter.

Select 1 as value for this parameter.

Find all data in [fig. 18-59 "Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position, schematic" on page 288:](#)

Transformation length	Meaning	Measuring unit	Restriction
L1:	Arm length L_1	mm or in (see K-0-0054)	≥ 0
L2:	Arm length L_2	mm or in (see K-0-0054)	> 0
L3:	Arm length L_3	mm or in (see K-0-0054)	> 0
L4:	Eccentricity L_4	mm or in (see K-0-0054)	Without restriction
L5:	Eccentricity L_5	mm or in (see K-0-0054)	Without restriction
L6:	Eccentricity L_6	mm or in (see K-0-0054)	Without restriction
L7:	Arm length L_7	mm or in (see K-0-0054)	≥ 0
L8:	Arm length L_8	mm or in (see K-0-0054)	≥ 0

Fig. 18-62: K-0-0040, Transformation lengths, axis configuration, axis lengths

The eccentricities L_4 , L_5 , L_6 can be positive or negative:

- $L_4 > 0$: Eccentricity in (+y) direction
- $L_4 < 0$: Eccentricity in (-y) direction
- $L_5 > 0$: Eccentricity in (+x) direction
- $L_5 < 0$: Eccentricity in (-x) direction
- $L_6 > 0$: Eccentricity in (+z) direction
- $L_6 < 0$: Eccentricity in (-z) direction

Transformation factors, axis configuration,
axis coupling
(K-0-0041)

Transformation Rules

Coupling factor	Meaning	Restriction
K1:	C ₄₅ mechanical coupling between the axes 4/5	Without restriction
K2:	C ₄₆ mechanical coupling between the axes 4/6	Without restriction
K3:	C ₅₆ mechanical coupling between the axes 5/6	Without restriction
K4:	C ₂₃ mechanical coupling between the axes 2/3	Without restriction
K5:	Bending angle of the fifth axis (50°)	Without restriction
K6:	Variant of the additional linear axes	Without restriction
K7:	Abort accuracy for iteration	>=0
K8:	Max. number of iteration steps + selection of the orientation definition + activation of the tensor interpolation	>=0

Fig. 18-63: K-0-0041, Transformation factors, axis configuration, axis coupling

Coupling factors C1 to C4

If mechanical axis couplings exist, the respective coupling factors have to be determined. They are defined as follows:

$$C_{23} = \Delta G_3 / \Delta A_2$$

$$C_{45} = \Delta G_5 / \Delta A_4$$

$$C_{46} = \Delta J_6 / \Delta A_4$$

$$C_{56} = \Delta G_6 / \Delta A_5$$

A_i is used for the axis angle and J_j for the joint angle detected by the measuring system. That means that the coupling factor C_{ij} indicates the effect on the joint j when axis i changes without notifying the measuring system j about this change.

Coupling factor C5

Angle by which the robot hand (50°) is bended, [fig. 18-59 "Robot type 27, robot with offset wrist with 6 axes + 3 linear axes, zero position, schematic" on page 288](#)

Coupling factor C6

Variants of additional linear axes

Entry in C6	Number of axes	Additional linear axes
0	6	Without additional linear axis
-11	7	Additional linear axis in x-direction
-12	7	Additional linear axis in y-direction
-13	7	Additional linear axis in z-direction
-21	8	Additional linear axes in x- and y-direction
-22	8	Additional linear axes in x- and z-direction
-23	8	Additional linear axes in y- and z-direction
10	9	Additional linear axis in x-direction; y- and z-direction are virtual axes
11	9	Additional linear axis in x-direction, y- and z-direction are virtual axes
12	9	Additional linear axis in y-direction, x- and z-direction are virtual axes
13	9	Additional linear axis in z-direction, x- and y-direction are virtual axes
21	9	Additional linear axes in x- and y-direction, z-direction is a virtual axis

Transformation Rules

22	9	Additional linear axes in x- and z-direction, y-direction is a virtual axis
23	9	Additional linear axes in y- and z-direction, x-direction is a virtual axis
31	9	Additional linear axes in x-, y- and z-direction
Others	6	No additional axis available

Fig. 18-64: Variants of additional linear axes

Virtual axes have to be formally created. If points are specified, they have to be carried along as coordinates with a value of 0.

Coupling factor C7 Abort accuracy in degrees to iterate the inverse position kinematics. If the value is smaller than 1 (e.g. 0.01), the same abort accuracy is applied to the position (1, 2, 3) and orientation axes (4, 5, 6). If the value is higher than 1, different abort criteria for position and orientation axes can be applied: The decimal values specifies the abort accuracy. The abort accuracy of the orientation axes results from the decimal value multiplied by the integer value (e.g. 20.01 => position accuracy = 0.01° orientation accuracy = 20 x 0.01° = 0.2°)

Coupling factor C8 The first and the second place of C8 (indicates the maximum number of iterations allowed (99 max., 8 recommended). If no convergence has been reached after the set number of iterations, the motion is aborted and the message "Iteration divergent" is output. The third and fourth place of C8 (WC orientation type) specifies the definition of the orientation angle (world coordinates 4, 5, 6). If the fifth place of C8 is 1, the tensor interpolation is enabled.

Example:

- K8 = 8
 - Maximum of eight iterations
 - WC orientation type 0
 - No tensor interpolation
- K8 = 312
 - maximum of twelve iterations
 - WC orientation type 3
 - No tensor interpolation
- C8 = 10807
 - Maximum of seven iterations
 - WC orientation type 8
 - With tensor interpolation

The following definition applies for the WC orientation type = 0, 4:

- WC[4] = rotation along the spatially fixed z-axis
- WC[5] = rotation along the resulting y'-axis
- WC[6] = rotation along the resulting x''-axis

The following definition applies for the WC orientation type = 1, 5:

- WC[4] = rotation along the spatially fixed z-axis
- WC[6] = rotation along the resulting x'-axis
- WC[5] = rotation along the resulting z''-axis

The following definition applies for the WC orientation type = 2, 6:

- WC[4] = rotation along the spatially fixed z-axis
- WC[5] = rotation along the resulting x'-axis

Transformation Rules

- WC[6] = rotation along the resulting y''-axis

The following definition applies for the WC orientation type = 3, 7:

- WC[4] = rotation along the spatially fixed y-axis
- WC[5] = rotation along the resulting x'-axis
- WC[6] = rotation along the resulting z''-axis

The following definition (Euler angle) applies for the WC orientation type = 8:

- WC[4] = rotation along the spatially fixed x-axis
- WC[5] = rotation along the resulting y'-axis
- WC[6] = rotation along the resulting z''-axis

The world coordinates 4, 5 and 6 are not restricted for the WC orientation types 0, 1, 2 and 3. The types 4, 5, 6 and 7 are subject to the following restrictions:

- $-90^\circ < Wc[4] < 90^\circ$
- $-180^\circ < Wc[4] < 180^\circ$
- $-180^\circ < Wc[6] < 180^\circ$

The following limitations apply for the "classic" Euler angles (type 8):

- $-180^\circ < Wc[4] < 180^\circ$
- $0^\circ < Wc[5] < 180^\circ$
- $-180^\circ < Wc[6] < 180^\circ$

The tensor interpolation is enabled via a 5-digit C₈. The spatial orientations (WC[4], WC[5], WC[6]) at linear motions are interpolated via a rotary sensor in automatic mode. For the tensor interpolation, one of the restricted WC orientation types has to be selected via the coupling factor C8 (i.e. WC orientation type).

Transformation Rules

18.14 Robot Type 29, 2-Axis Delta with Rotary and Translatory Axes

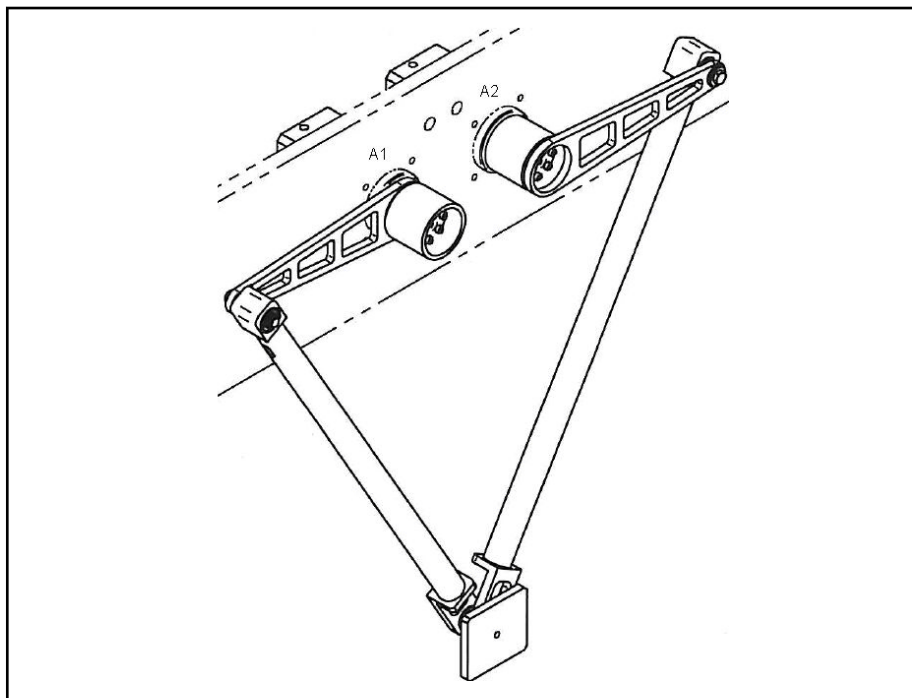


Fig. 18-65: Robot type 29, 2-axis delta

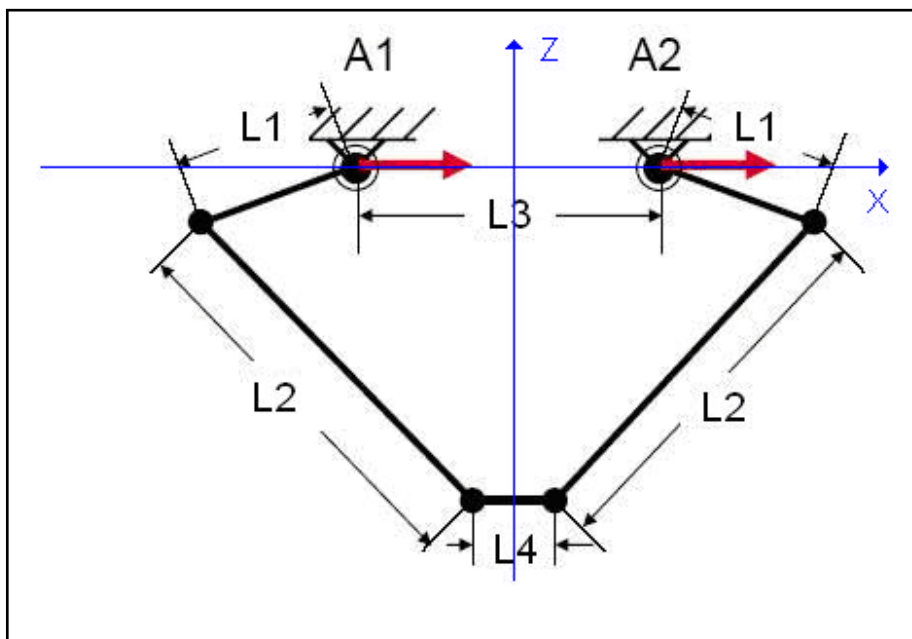


Fig. 18-66: Robot type 29, 2-axis delta (X-Z view)

Parameter settings

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

Transformation Rules

A3:	Positioning axis	Translatory absolute	Optional
A4:	Orientation axis	Rotary abs./modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-67: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, transformation scheme, Type = 29 (2-axis delta, bipod)
- K-0-0032, Referencing all axes = 0
- K-0-0033, IPOS, update at coordinate change = 0
- K-0-0036, WC axis monitoring = 1

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

Fig. 18-68: K-0-0040, Transformation lengths, axis configuration, axis lengths

K-0-0041, Transformation factors, axis configuration, axis coupling, is not used

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cz)	
Coordinate 3:	Name of third coordinate (e.g. Cy)	
Coordinate 4:	Name of the fourth coordinate (e.g. Cc)	
Coordinate 5:	Name of the fifth coordinate (e.g. Kbelt1)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Kbelt2)	Optional
Coordinate 7:	Name of the seventh coordinate (e.g. Kbelt3)	Optional
Coordinate 8:	Name of the eighth coordinate (e.g. Kbelt4)	Optional

Fig. 18-69: K-0-0121 .. K-0-0128, Coordinate names, coordinates



Tool coordinate system

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets

Transformation Rules

- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

fig. 18-65 "Robot type 29, 2-axis delta" on page 294 shows a bipod robot in 3D with the axis positions $A1=180^\circ$ and $A2 = 0^\circ$.

In zero position, the axes 1 and 2 (positioning axes) are horizontal towards the x-axis (see red arrow in fig. 18-66 "Robot type 29, 2-axis delta (X-Z view)" on page 294).

- The arm length L_1 is measured from the rotary center point to the center point of the rod retainer.
- The rod length L_2 is measured from the reference points on both sides.
- L_3 is the distance of the rotary center point of both arms.
- L_4 is the length of the mold 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 $A2$.

18.15 Robot Type 30, Adabot

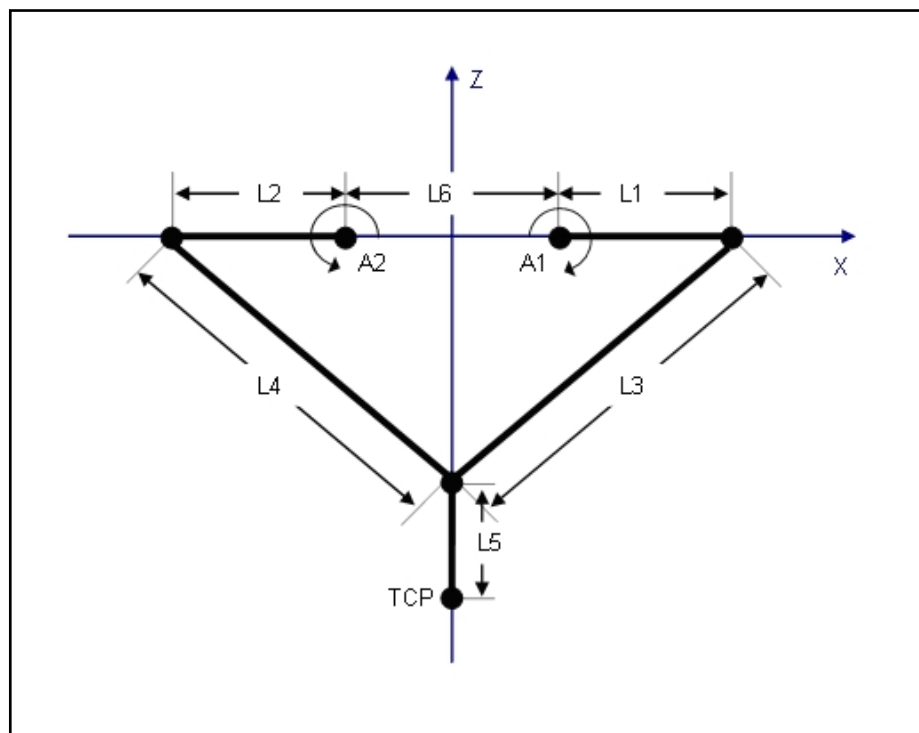


Fig. 18-70: Robot type 30, Adabot (X-Z view, axis zero positioning)

Transformation Rules

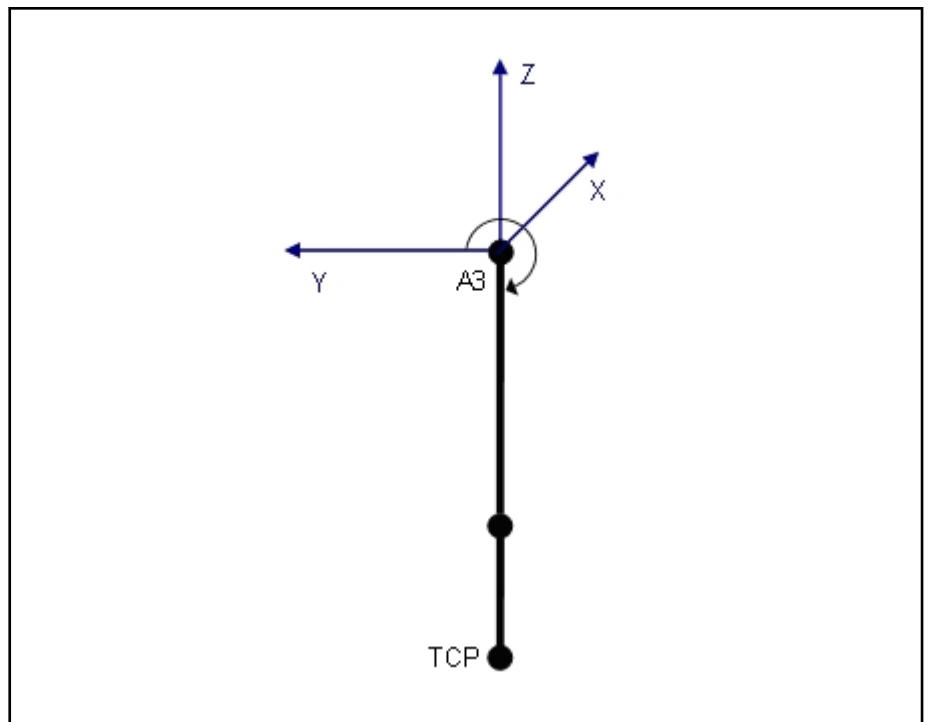


Fig. 18-71: Robot type 30, Adabot (X-Z view, axis zero positioning)

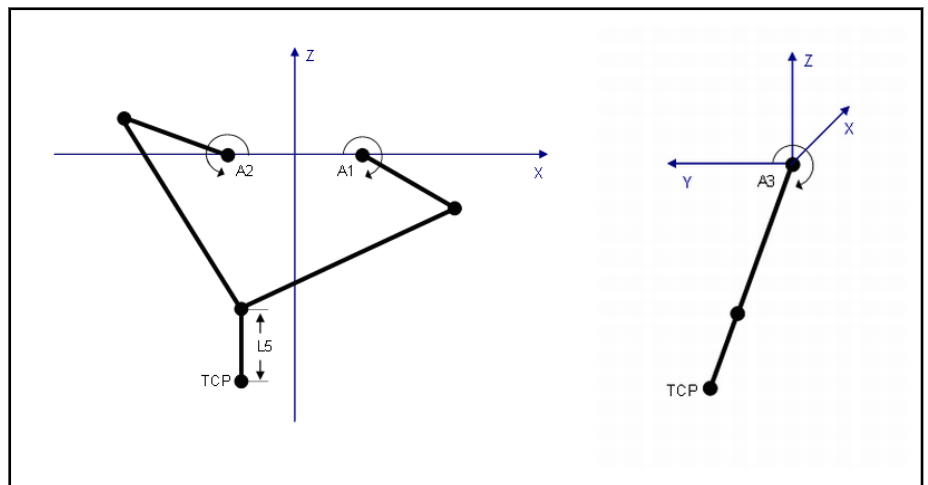


Fig. 18-72: Robot type 30, Adabot (behavior of rod L5)

Parameter settings

Axis name	Axis type	Scaling	Remark
A1:	Positioning axis	Rotary absolute	+/- 90 degree
A2:	Positioning axis	Rotary absolute	+/- 90 degree
A3:	Positioning axis	Rotary absolute	+/- 90 degree
A4:	Orientation axis	Rotary abs./modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional

Transformation Rules

A7:	Belt axis (optional)	Translatory, modulo	Optional
A8:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-73: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, type = 30 (Adabot).
- K-0-0032, Referencing all axes = 0

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

Fig. 18-74: K-0-0040, Transformation lengths, axis configuration, axis lengths

K-0-0041, Transformation factors, axis configuration, axis coupling, is not used

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cy)	
Coordinate 3:	Name of the third coordinate (e.g. Cz)	
Coordinate 4:	Name of the fourth coordinate (e.g. Cc)	Optional
Coordinate 5:	Name of the fifth coordinate (e.g. Kbelt1)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Kbelt2)	Optional
Coordinate 7:	Name of the seventh coordinate (e.g. Kbelt3)	Optional
Coordinate 8:	Name of the eighth coordinate (e.g. Kbelt4)	Optional

Fig. 18-75: K-0-0121 .. K-0-0136, Coordinate names, coordinates

**Tool coordinate system**

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

Transformation Rules

fig. 18-70 "Robot type 30, Adabot (X-Z view, axis zero positioning)" on page 296 and fig. 18-71 "Robot type 30, Adabot (X-Z view, axis zero positioning)" on page 297 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. 18-72 "Robot type 30, Adabot (behavior of rod L_5)" on page 297).

The Cartesian zero point of the Adabot kinematics is in the center of both arm rotation points A1 and A2 (see fig. 18-70 "Robot type 30, Adabot (X-Z view, axis zero positioning)" on page 296).

18.16 Robot Type 32, 2-Axis Delta with Asymmetric TCP

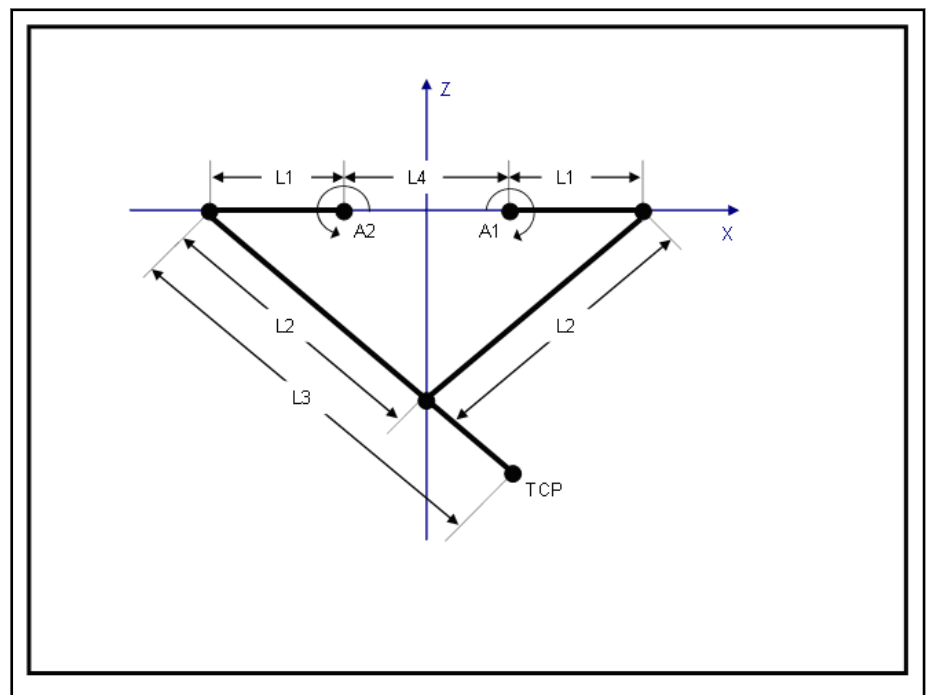


Fig. 18-76: Robot type 32, 2-axis delta with asymmetric TCP (X-Z view, axis zero positioning)

Parameter settings

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

Transformation Rules

A4:	Belt axis (optional)	Translatory, modulo	Optional
A5:	Belt axis (optional)	Translatory, modulo	Optional
A6:	Belt axis (optional)	Translatory, modulo	Optional
A7:	Belt axis (optional)	Translatory, modulo	Optional

Fig. 18-77: K-0-0008, Axis configuration list, axis assignment

- K-0-0031, Transformation scheme, type = 32
- K-0-0032, Referencing all axes = 0

Transformation length	Axis configuration	Measuring unit	Axis length
L1:	Length of the arm 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

Fig. 18-78: K-0-0040, Transformation lengths, axis configuration, axis lengths

K-0-0041, Transformation factors, axis configuration, axis coupling, is not used

Name	Coordinate	Remark
Coordinate 1:	Name of the first coordinate (e.g. Cx)	
Coordinate 2:	Name of the second coordinate (e.g. Cz)	
Coordinate 3:	Name of the third coordinate (e.g. Cc)	Optional
Coordinate 4:	Name of the fourth coordinate (e.g. Kbelt1)	Optional
Coordinate 5:	Name of the fifth coordinate (e.g. Kbelt2)	Optional
Coordinate 6:	Name of the sixth coordinate (e.g. Kbelt3)	Optional
Coordinate 7:	Name of the seventh coordinate (e.g. Kbelt4)	Optional

Fig. 18-79: K-0-0121 .. K-0-0136, Coordinate names, coordinates

**Tool coordinate system**

The transformation does not support any orientation offsets configured in the file TOOLS.DAT.

Axis directions of rotation and zero positions

The basic requirements to perform the coordinate transformation correctly are:

- Exact setting of the axis zero points
- Exact measurement of the arm lengths or center offsets
- Correct direction of rotation of the axes

If one of these requirements is not completely fulfilled, a precise linear motion cannot be guaranteed.

This can result in incorrect or unexpected motions.

Transformation Rules

fig. 18-76 "Robot type 32, 2-axis delta with asymmetric TCP (X-Z view, axis zero positioning)" on page 299 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 .
- The Cartesian zero point is in the center of the two arm rotation points A1 and A2.

18.17 Robot Type 100, PLC

Function It often occurs in the application that the transformations provided by the system do not fit the kinematics. Therefore, the system provides the user-defined transformation type with the number 1100. The conversion rules from world to axis coordinates and vice versa are implemented in the PLC for this type.

To define such a transformation, the interface MC_KIN_REF has to be implemented by a function block.

Subsequently, the transformation has to be registered at the system. This is done using the function block MC_SetKinTransform.

The function pointers have to be transferred to the individual functions of the interface MC_KIN_REF as follows to the function block via the structure ML_KIN_AXES_CONFIG_DATA / ML_KIN_AXIS_CONFIG.

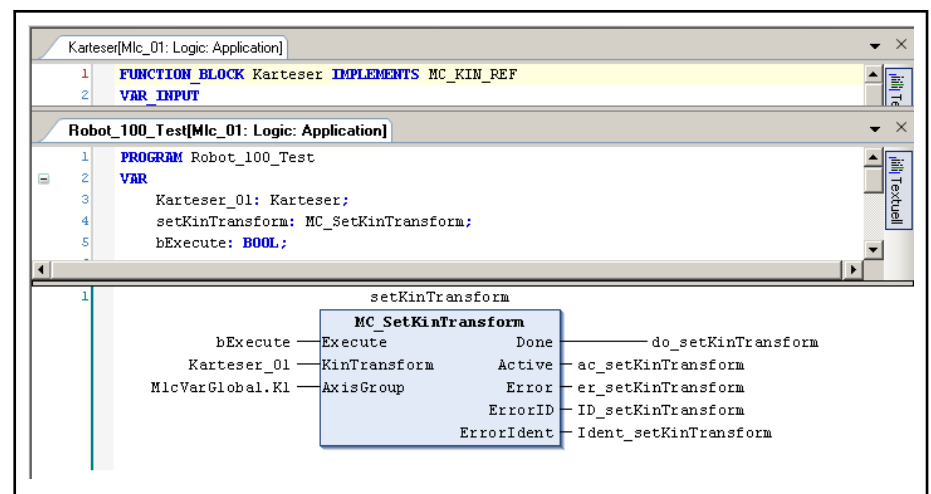


Fig. 18-80: Selecting a transformation

After a transformation has been registered at the system, the transformation is used by the system.

Details The required function block, the interface and the structures are part of the library ML_Robot_xx.library.

The description of this library is part of the documentation:

Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen Libraries, DOK-XLCMLC-FUNLIB**V13-L1xx-EN-P, R911336360.

Parameter settings

- K-0-0031, Transformation scheme, type = 1100
- K-0-0040, Transformation, lengths of the mechanics
- K-0-0041, Transformation, factors

19 RCL Keywords

In the following, all the language symbols currently reserved for the RCL are listed. The language symbols listed may not be used in an RCL program as variables, file names or subroutine names.

Keyword	Remark
@	Joint coordinate points. See JC_POINT in Composed RCL Data Types , page 61..
@MPOS	Reserved
@POS	Current position in joint coordinates, see also Reading the actual position POS, @POS , page 67.
A	Path acceleration, see also A , page 131.
ABS	Function provides the value of a REAL or INTEGER number, ABS Function , page 97.
AFACTOR	Path acceleration factor, see also AFACTOR , page 132.
AFIX	AFIX , page 128.
ALSO	Reserved
AND	For logical AND operation bit by bit, see also Logic Expressions , page 91.
APPROX	See MOVE_REL APPROX... , page 120.
ARRAY	Working with Array Variables (Arrays) , page 69.
ASC_INT	Data conversion from CHAR to INTEGER, ASC_INT Function , page 99.
ASSIGN	File access, ASSIGN Function , page 98.
ATAN	ATAN Function , page 95.
BEGIN	Start of program code, see also Statement Part , page 46.
BELT	Belt Synchronization , page 135., Communication Channels to PLC, General , page 70.
BINARY	Overview , page 59.
BLOCK_SLOPE	SATZ_SLOPE Block Transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , page 137.
BRN_FILE	Reserved
BREAK	Reserved
CASE	Case decision, CASE Statement , page 88.
CASE_END	
CHAR	RCL Data Types , page 59.
CHR	Data conversion, calculates a CHAR from INTEGER MOD 256, CHR Function , page 97.
CIRCULAR	Interpolation Method , page 122.
CLOSE	See also ASSIGN Function , page 98.

RCL Keywords

Keyword	Remark
CLS	Reserved
COMPILER	Mode of the Compiler Operation, page 41.
CONDITION	Reserved
CONST	RCL Constant Declaration , page 57.
CONT_SLOPE	Block Transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , page 137.
CONTROL	Reserved
COS	COS Function, page 95.
DEBUGINFO	DEBUGINFO+, DEBUGINFO-, compiler statement, see also Compiler Statements, page 42.
DEF	Point variable in point table, RCL Variable Declaration, page 63.
DEFAULT	CASE Statement, page 88.
DFACTOR	Deceleration factor, see also DFACTOR, page 132.
ELSE	IF-THEN-(ELSE) Statement, page 85.
END	Program Declaration, page 46.
END_OF_FILE	EOF Function, page 98.
EQUAL	CASE Statement, page 88.
ERROR	Maximum waiting time, page 79.
EVERY	Reserved
EXACT	See MOVE_REL EXACT..., page 121.
EXCLUSIVE	Reserved
EXCLUSIVE_END	Reserved
EXTERNAL	Main Program Call in Main Program, page 47.
FILE	File work, see FILE, page 61.
HALT	HALT Statement, page 81.
IF	IF-THEN-(ELSE) Statement, page 85.
INCLUDE	Including Files, page 43.
INPUT	Interface to PLC, PLC outputs - RoCo inputs, RCL Variable Declaration, page 63..
INT	Global interpolation method , page 124.
INT_ASC	Function converts an integer value to an array of CHAR, INT_ASC Function, page 98.
INTEGER	RCL Data Types, page 59.
J	Path jerk, see also A, page 133.
JC (..)	Coordinate transformation, conversion of world coordinates to joint coordinates, JC Standard Function, page 101.

RCL Keywords

Keyword	Remark
JC_NAMES	Compiler statements, Axis Names (JC Name Definition) , page 43 .
JC_POINT	Point Variables , page 65 .
JUMP	JUMP Statement , page 83 .
KINEMATICS	Compiler statement, Kinematic Definition , page 42 .
LIMIT_MAX	Workspace limitations, LIMIT_MIN , LIMIT_MAX , LIMIT_OFF Functions , page 101 .
LIMIT_MIN	
LIMIT_OFF	
LINEAR	Interpolation Method , page 122 .
MAX_TIME	Maximum waiting time , page 79 , see also WAIT .
MCP	Reserved
MOD	Modulo division, Arithmetic Expressions , page 89 .
MOVE	Motion command, MOVE , page 118 .
MOVE TO	Motion command, MOVE TO ... , page 119 .
MOVE VIA	Motion command, MOVE VIA ... (blending) , page 119 .
MOVE VIA..TO	Motion command, MOVE VIA ... TO ... , page 119 .
MOVE_REL	Motion command, MOVE_REL , page 120 .
MOVE_REL APPROX	Motion command, MOVE_REL APPROX... , page 120 .
MOVE_REL EXACT...	Motion command, MOVE_REL EXACT... , page 121 .
MOVE_REL APPROX.. EXACT	Motion command, MOVE_REL APPROX ... EXACT ... , page 121 .
NOT	Bit-wise negation, Logic Expressions , page 91 .
OR	Bit by bit OR operation, Logic Expressions , page 91 .
ORD	Data conversion, INTEGER value of a variable of type CHAR, ORD Function , page 97 .
OUTPUT	Interface to PLC, RoCo outputs, PLC inputs, RCL Variable Declaration , page 63 .
PARALLEL	Reserved
PARALLEL_END	Reserved
PAUSE	PAUSE Statement , page 81 .
PERMANENT	Reserved
PLC	Reserved
PLC_PROCESS	Reserved
PLC_TIME	Reserved
POINT	Point Variables , page 65 .
POS	Current position in world coordinates, Reading the actual position POS , @POS , page 67 .
POSE	Reserved

RCL Keywords

Keyword	Remark
PRIOR	Reserved
PROCESS_KIND	Reserved
PROGR_SLOPE	Block Transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , page 137.
PROGRAM	Program Declaration, page 46.
PROGRAM_END	
PTP	Interpolation Method , page 122.
PUBLIC	Global Variables, page 64.
R	R, page 142.
R_PTP	R_PTP, page 142.
READ	See also ASSIGN Function , page 98., FILE , page 61.
READ_BEGIN	See also ASSIGN Function , page 98.
REAL	Overview , page 59.
RECORD	Record Types , RECORDs , page 62.
RECORD_END	
REF_PNT	Reserved
REPEAT	n: number of repetitions REPEAT Statement , page 82.
REPEAT n TIMES	
REPEAT_END	
RETURN	Subroutine Declaration , page 50.
RHO_FCT	Reserved
SEMAPHORE	Reserved
SIN	SIN Function , page 95.
SIZEOF	SIZEOF Function , page 100.
SPC_FCT	RCL Special Functions , Listed as Table, page 104.
SQRT	SQRT Root Function , page 96.
START	Reserved
STOP	Reserved
SUB_END	Subroutine Declaration , page 50.
SUBROUTINE	
SYNC	SYNC , page 135,
SYNCHRON	SYNCHRON , page 135,
SYNCHRON_END	SYNCHRON_END , page 135,
TEXT	Overview , page 59.
THEN	IF-THEN-(ELSE) Statement , page 85.

RCL Keywords

Keyword	Remark
TIMES	See REPEAT Statement , page 82.
TO	See MOVE TO ... , page 119.
TOOL	Reserved
TRUNC	Data conversion that uses truncation to transform the argument from the REAL type into an INTEGER type value, TRUNC Function , page 97.
TTY	Reserved
TYPE	User-Defined RCL Types , page 61.
UNTIL	See WAIT UNTIL , Querying channels/signals , page 74.
V	V , page 127.
V_PTP	V_PTP , page 127.
VALUE	Subroutine call, call by value, Subroutine Sequence , page 52.
VAR	Beginning of the variable definition, see Simple RCL Data Types , page 60.
VERSION	Reserved
VFACTOR	VFACTOR , page 129.
VFIX	VFIX , page 128.
VIA	See MOVE VIA ...(blending) , page 119.
WAIT	WAIT Statement , page 77.
WAIT UNTIL	
WARNING	Reserved
WC (..)	Coordinate transformation, conversion of joint coordinates to world coordinates, WC Standard Function , page 101.
WC_CALCULATION	Reserved
WC_FRAME	Reserved
WC_NAMES	Compiler statement, World Coordinate Names (WC Name Definition) , page 42.
WC_SYSTEM	Reserved
WC_UR	See WC_UR , page 57.
WITH	Statement-related velocity specification (WITH) , page 128.
WRITE	See also ASSIGN Function , page 98., FILE , page 61.
WRITE_BEGIN	See also ASSIGN Function , page 98.
WRITE_END	See also ASSIGN Function , page 98.

Fig. 19-1: Keywords for RCL programs

RCL compiler statements RCL compiler statements begin with ";;", followed by the actual control statement.

RCL Keywords

Control statement	Remark
CONTROL	Reserved
COMPILER	General Information, page 42.
DEBUGINFO	Debug Information, page 44.
FILE_ERROR	Reserved
INCLUDE	Including Files, page 43.
INT	Global interpolation method , page 124.
JC_NAMES	Axis Names (JC Name Definition), page 43.
KINEMATICS	Kinematic Definition, page 42.
WARNING	Reserved
WC_NAMES	World Coordinate Names (WC Name Definition), page 42.

Fig. 19-2: Control statements for the RCL compiler

RCL standard functions

Standard functions	Remark
ABS	Function provides the value of a REAL or INTEGER number, ABS Function, page 97.
AND	For logical AND operation bit by bit, see also Logic Expressions, page 91.
ASC_INT	Data conversion from CHAR to INTEGER, ASC_INT Function, page 99.
ASSIGN	File access, ASSIGN Function, page 98.
ATAN	ATAN Function, page 95.
BNR_FILE	Reserved
CHR	Data conversion, calculates a CHAR from INTEGER MOD 256, CHR Function, page 97.
CONDITION	Reserved
COS	COS Function, page 95.
END_OF_FILE	Query whether the file end has been reached when reading a dat file, EOF Function, page 98.
INT_ASC	Function converts an integer value to an array of CHAR, INT_ASC Function, page 98.
JC	Coordinate transformation, conversion of world coordinates to joint coordinates, JC Standard Function, page 101.
MOD	Modulo division, Arithmetic Expressions, page 89.
NOT	For logical negation bit by bit, see also Logic Expressions, page 91.
OR	For logical OR operation bit by bit, see also Logic Expressions, page 91.
ORD	Data conversion, INTEGER value of a variable of type CHAR, ORD Function, page 97.
PLC_PROCESS	Reserved

RCL Keywords

Standard functions	Remark
PLC_TIME	Reserved
ROUND	Converts REAL into INTEGER. Rounding is to the closest integer. ROUND Function, page 97.
SIN	SIN Function, page 95.
SIZEOF	Determination of the memory space required for any sort of RCL variables or types, SIZEOF Function, page 100.
SQRT	SQRT Root Function, page 96.
TRUNC	Uses truncation to convert REAL into INTEGER. TRUNC Function, page 97.
WC	Coordinate transformation, conversion of joint coordinates to world coordinates, WC Standard Function, page 101.
WC_CALCULATION	Reserved

Fig. 19-3: RCL standard functions

RCL Special Functions

SPC_FCT	Brief functional description
1	IOL , page 106 , (accurate position switching of digital outputs on the path)
2	PPO , page 107 , (accurate position switching of process data on the path)
21	belt_kind , page 108 , (belt type)
23	DateTime , page 109 , (system date and time)
24	SysClk , page 109 , (system counter, real-time counter)
28	set_belt , page 110 , (set belt counter reset value) See SYNC
53	belt_area , page 110 , (belt range with belt type 4)
59	getBlendMode , page 114 , (query blending mode)
60	setBlendMode , page 113 , (set blending mode)

Fig. 19-4: RCL Special Functions

RCL standard constants

Standard constants are currently in limited use.

Standard constants	Remark
CLS	Reserved
VERSION	Reserved
WC_UR	See WC_UR , page 57.

Fig. 19-5: RCL standard constants

RCL standard variables

Standard variables	Remark
@	Joint coordinate points See JC_POINT in Composed RCL Data Types, page 61..
@MPOS	Reserved

RCL Keywords

Standard variables	Remark
@POS	Current position in joint coordinates Reading the actual position POS, @POS, page 67.
A	A, page 131.
AFACTOR	AFACTOR, page 132.
AFIX	AFIX, page 128.
DFACTOR	DFACTOR, page 132
J	Path jerk, see also A, page 133.
LIMIT_MAX	Workspace limitations, LIMIT_MIN, LIMIT_MAX, LIMIT_OFF Functions, page 101.
LIMIT_MIN	
LIMIT_OFF	
MCP	Reserved
PLC	Reserved
POS	Current position in world coordinates Reading the actual position POS, @POS, page 67.
R	R, page 142.
R_PTP	R_PTP, page 142.
TTY	Reserved
V	V, page 127.
V_PTP	V_PTP, page 127.
VFACTOR	VFACTOR, page 129.
VFIX	VFIX, page 128.
WC_SYSTEM	Reserved

Fig.19-6: RCL standard variables

RCL standard data types

Information on the RCL standard data types can be found under [Overview, page 59.](#)

Standard data types	Remark
Simple data types	
BINARY	0, 1
CHAR	The ASCII (American Standard Code for Information Interchange) character encoding (7 bits) is used to represent letters and symbols used on a standard US keyboard. The ASCII character encoding corresponds to the first 128 characters (0-127) of the ANSI character set.
INTEGER	INTEGER type variables (long integers) are saved as 32 bit numbers (4 bytes) with signs in the range from -2,147,483,648 to 2,147,483,647.

RCL Keywords

Standard data types	Remark
REAL	REAL data type variables (long integers) are saved as 32 bit numbers (4 bytes) -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values.
Structured data types	
ARRAY	Working with Array Variables (Arrays), page 69.
BELT	Reserved
FILE	Composed RCL Data Types, page 61.
JC_POINT	Point Variables, page 65.
POINT	Point Variables, page 65.
RECORD RECORD_END	Record Types, RECORDs, page 62.
TEXT	Working with Text Variables, page 68.
WC_FRAME	Reserved

Fig. 19-7: Standard data types

20 Service and Support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Helpdesk & Hotline** under:

Phone:	+49 9352 40 5060
Fax:	+49 9352 18 4941
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances resulting in the malfunction
- Type plate name of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your email address)

Index

Symbols

@.....	61
@POS.....	67

A

A.....	131
A-0-0028, Travel range limit switch.....	157
A-0-0029, Position polarities.....	157
A-0-0030, Position limit value, positive.....	158
A-0-0031, Position limit value, negative.....	158
A-0-0032, Velocity limit value, positive.....	159
A-0-0033, Velocity limit value, negative.....	159
A-0-0034, Acceleration limit value, bipolar.....	159
A-0-0036, Jerk limit value, bipolar.....	159
A-0-0038, Torque/force limit value, bipolar.....	160
A-0-0048, Velocity polarities.....	160
A-0-0049, Torque-/force polarities.....	161
A-0-0080, Jogging, short distance.....	161
A-0-0081, Jogging, long distance.....	162
A-0-0082, Jogging, low velocity.....	163
A-0-0083, Jogging, high velocity.....	163
A-0-0084, Jogging, acceleration.....	164
A-0-0085, Jogging, deceleration.....	165
A-0-0086, Jogging, acceleration jerk.....	165
A-0-0087, Jogging, deceleration jerk.....	165
A-0-0224, Rapid stop deceleration.....	166
A-0-0225, Rapid stop jerk.....	166
A-0-0230, Blending distance PTP.....	166
A-0-0240, Feedrate override.....	167
A-0-0241, Feedrate override control word.....	167
A-0-0242, Position control, synchronization acceleration.....	167
A-0-0243, Position control, synchronization velocity.....	168
Abbreviations.....	17
About this documentation.....	11
Information representation	16
Validity of the documentation	11
ABS.....	97
Acceleration.....	130
Acceleration, effective.....	132
Acceleration factor.....	131
Acceleration specification.....	130
Global	130
Statement-related	131
Actual position reading.....	67
Additional information.....	127
Acceleration	130
Acceleration factor	131
Deceleration factor	132
Effective acceleration	132
Effective deceleration	133
Effective velocity	129
Global acceleration specification (global specification)	130

A

...Additional information	
Global jerk specification (global specifica- tion)	133
Global velocity specification	128
Jerk	133
Motion optimization for orientation axes	127
Override	128
Statement-related acceleration specifica- tion (WITH)	131
Statement-related jerk specification (WITH)	134
Statement-related velocity specification (WITH)	128
Velocity	127
Velocity factor	129
Additional Information	
Disable override	128
AFACTOR.....	132
AFIX.....	128
ALIAS name.....	61
AND.....	92
Arithmetic expressions.....	89
ARRAY.....	63, 69
Array variables.....	69
ASC_INT.....	99
ASSIGN.....	98
ATAN.....	95

B

Basic coordinate system (BCS).....	20
Basic coordinate system relative to the world coordinate system.....	25
BEGIN.....	46
BELT.....	135
Belt synchronization.....	135
BINARY.....	60
Blending mode.....	151
RCL programming	113
Blending when changing the interpolation method.....	147, 149
BLOCK_SLOPE.....	137
Block transitions (slope types).....	137

C

C-0-0123, Maximum number of kinematics.....	169
C-0-2480, Configured kinematic number.....	169
C-0-2482, List of non-linked axes.....	170
C-0-2483, List of configured kinematics.....	170
C-0-2484, Kinematic configuration list.....	170
Call by VALUE.....	47
CASE.....	88
Channel declaration.....	73
CHAR.....	60

Index

C

CHR.....	97
CIRCULAR.....	122
CLOSE.....	61
Comparative operations.....	91
Compiler - Mode of operation.....	41
Compiler statements.....	42
CONST.....	57
CONT_SLOPE.....	138
Coordinates.....	19
Coordinate systems.....	19
COS.....	95

D

Debugger	
Connect in debug mode	230
Debug program	231
Disconnect in debug mode	230
Display current block number	231
Monitor variables	232
Sett and delete breakpoints	230
Start program in debug mode	230
Stop program in debug mode	230
DEBUGINFO.....	44
Deceleration, effective.....	133
Deceleration factor.....	132
DEF.....	63
DEFAULT.....	88
Define coordinate systems.....	25
DFACTOR.....	132
Direct motion statements.....	117
Additional information	127
Interpolation method	122
Motion commands	118
Disable override.....	128

E

ELSE.....	85
END_OF_FILE.....	98
EOF.....	98
EQUAL.....	88
err file.....	42
Extension for include files.....	44
EXTERNAL.....	47
EXTERNAL, with parameter transfer.....	49

F

FILE.....	61

G

Global variables.....	64

H

HALT.....	49, 81

I

IF.....	85, 91
INCLUDE.....	43
Information representation	
Names and abbreviations	17
Safety instructions	16
Symbols used	17
INPUT.....	63
INT.....	124
INT_ASC.....	98
INTEGER.....	60
Intermediate point.....	122
Interpolation method.....	122
CIRCULAR	122
Global	124
LINEAR	122
PTP	123
Statement-related	123

J

J.....	133
JC.....	67, 101
JC_NAMES.....	43
JC_POINT.....	61, 66
Jerk.....	133
jerk specification.....	133
Jerk specification	
Global	133
Statement-related	134
Jog	
Axis coordinates	153
Joint coordinates	153
Jogging	
World coordinates	155, 156
Jog in different coordinate systems.....	153
Jog mode.....	153
Joint coordinate system (JCS).....	20
JUMP.....	83
Jump target.....	83

K

K-0-0001, Logic kinematic number.....	171
K-0-0002, Kinematic name.....	171
K-0-0003, Maximum path speed.....	171
K-0-0004, Reserved.....	172
K-0-0005, Software limit switches, positive, WC.....	172
K-0-0006, Software limit switches, negative, WC.....	172
K-0-0007, Axis number of kinematics.....	172
K-0-0008, Axis configuration list.....	172
K-0-0008 Define axis configuration list.....	245
K-0-0009, Osci: list of all K-parameters that can be oscilloscoped.....	174
K-0-0010, List of all K-parameters.....	174
K-0-0011, List of all invalid K-parameters.....	175

K

K-0-0012, Checksum of all K-parameters.....	175
K-0-0013, IDN list of checksum K-parameters..	175
K-0-0014, Path acceleration, change time.....	175
K-0-0015, Path deceleration, change time.....	178
K-0-0016, Project identification number.....	180
K-0-0019, Kinematic type.....	181
K-0-0020, Kinematic diagnostic message.....	181
K-0-0021, Kinematic status.....	181
K-0-0022, Extended kinematic status.....	183
K-0-0023, Kinematics diagnostic number.....	186
K-0-0031, Transformation scheme.....	186
K-0-0031 Define transformation scheme.....	244
K-0-0032, Referencing axes, command configuration.....	187
K-0-0033, IPOS, update at coordinate change.	187
K-0-0036, WC axis monitoring.....	187
K-0-0040, Transformation, lengths of the mechanics.....	188
K-0-0041, Transformation, factors.....	188
K-0-0042, WC system offset.....	188
K-0-0043, Assignment list of workpiece coordinates.....	189
K-0-0045, Tool selection.....	190
K-0-0050, Distance units WC.....	190
K-0-0051, Velocity units WC.....	191
K-0-0052, Acceleration units WC.....	191
K-0-0053, Jerk units WC.....	192
K-0-0054, Measuring unit for kinematic data....	193
K-0-0070, Belt number of kinematics.....	193
K-0-0071, Coupling factors, belt 1.....	193
K-0-0072, Coupling factors, belt 2.....	194
K-0-0073, Coupling factors, belt 3.....	194
K-0-0074, Coupling factors, belt 4.....	195
K-0-0080, Jogging, distance short in WC system.....	195
K-0-0081, Jogging, distance long in WC system.....	196
K-0-0082, Jogging, velocity low in WC system.	197
K-0-0083, Jogging, velocity high in WC system	197
K-0-0084, Jogging, acceleration in WC system	198
K-0-0085, Jogging, deceleration in WC system	199
K-0-0086, Jogging, acceleration jerk in WC system.....	199
K-0-0087, Jogging, deceleration jerk in WC system.....	200
K-0-0099, Kinematic configuration.....	201
K-0-0100, Current position, Cartesian original coordinates.....	201
K-0-0102, Current path velocity.....	201
K-0-0103, Current path acceleration.....	201
K-0-0104, Current path jerk.....	201
K-0-0105, Actual position, Cartesian workpiece coordinates.....	202
K-0-0106, Coordinate values of tool.....	202
K-0-0107, Coordinate values of workpiece.....	202
K-0-0121, Coordinate name C1.....	202

K

K-0-0121 to K-0-0136, coordinate names.....	202
K-0-0155, Safe zones on/off.....	202
K-0-0156, Safe zone 1 min. point.....	203
K-0-0156/K-0-0157 Safezone1 min./max. Point till K-0-0218/K-0-0219 Safezone32 min./max. Point.....	203
K-0-0157, Safe zone 1 max. point.....	203
K-0-0220, File transfer name.....	205
K-0-0221, File name to be compiled.....	205
K-0-0240, Feedrate override.....	206
K-0-0241, Feedrate override control word.....	206
K-0-0242, Acceleration override.....	206
K-0-0243, Deceleration override.....	206
K-0-0250, Continuous slope lead time.....	207
K-0-1060, Command: Load file from CF.....	207
K-0-1061, Command: Save file to CF.....	207
K-0-1062, Command: Compile file to MLC.....	207
K-0-2200, Programmed position 1 (active).....	207
K-0-2201, Programmed position 2 (active).....	207
K-0-2202, Type of actively programmed position.....	208
K-0-2203, Interpolation type.....	208
K-0-2204, Currently active QLL file name.....	208
K-0-2205, Currently active line number.....	208
K-0-2300, PLC-Bit input (RCL).....	208
K-0-2301, PLC-Integer input (RCL).....	208
K-0-2302, PLC-Real input (RCL).....	208
KINEMATICS.....	42
Kinematic user interface.....	209

L

Late blending.....	149
Late commanding.....	141
Late commanding	141
LIMIT_MAX.....	101
LIMIT_MIN.....	101
LIMIT_OFF.....	101
LINEAR.....	122
Linear blending for BLOCK_SLOPE.....	149
Linear blending for PROGR_SLOPE and CONT_SLOPE.....	144
Logic expressions.....	91
Look-ahead across several blocks at PROGR_SLOPE and CONT_SLOPE.....	141

M

Main program	
Declaration part	45
Statement part	46
Structure	44
MAX_TIME.....	79
MLC RobotControl.....	0
MOD.....	90
Motion commands.....	118

Index

M

Motion-influencing RCL statements and parameters.....	125
Motion-influencing statements.....	134
Belt synchronization	135
Block transitions (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)	137
Look-ahead across several blocks at PROGR_SLOPE and CONT_SLOPE	141
Motion optimization for orientation axes.....	127
MOVE.....	118
MOVE_REL	120
MOVE_REL APPROX.....	120
MOVE_REL APPROX ... EXACT	121
MOVE_REL EXACT.....	121
MOVE TO	119
MOVE VIA	119
MOVE VIA ... TO	119

N

NOT.....	92

O

OPEN.....	61
Operations.....	89
Arithmetic expressions	89
Comparative operations	91
Logic expressions	91
Sequence of operations	92
Operators	
Arithmetic	89
Logic	91
OR.....	92
ORD.....	97
OUTPUT.....	63
Override.....	128
Overview	
Motion-influencing RCL statements and parameters	125
Travel range limits	39

P

PAUSE.....	81
pkt file.....	41
PLC communication channels.....	70
POINT.....	61, 66
Points and point file pkt.....	66
Point variables.....	65
Complete value assignment	66
Declaration	65
Value assignment for individual components	66
POS.....	67
Position accurate switching	
Belt type	109
Position-accurate switching.....	105

P

...Position-accurate switching	
Belt synchronous operating range of belt type 4	110
Belt type	108
digital outputs on the path	106
Set belt counter	110
Switching of process data on the path	107
SysClk	109
PROGR_SLOPE.....	138
PROGRAM.....	46
PROGRAM_END.....	46
Program for tracing	
Connect for tracing	229
Disconnect for tracing	229
Start program for tracing	230
Program tracing	
Stop program during tracing	230
PTP.....	123
PTP blending for BLOCK_SLOPE.....	147
PTP blending for PROGR_SLOPE and CONT_SLOPE.....	143

Q

qll file.....	41

R

R.....	142
R_PTP.....	142
RCL	
Program structure	41
RCL constants.....	57
Constant declarations	57
Predefined	57
RCL coordinate systems	
Basic coordinate system (BCS)	20
Joint coordinate system (JCS)	20
Tool coordinate system (TCS)	24
Workpiece coordinate system (WpCS)	23
World coordinate system (WCS)	22
RCL data types.....	59, 100
Composed	61
Simple	60
User-defined	61
RCL functions.....	95
ABS function	97
ASC_INT function	99
ASC function	98
ASSIGN function	98
ATAN function	95
CHR function	97
COS function	95
EOF function	98
JC function	101
LIMIT_MAX function	101
LIMIT_MIN function	101

R

...RCL functions	
LIMIT_OFF function	101
ORD function	97
ROUND function	97
SIN function	95
SIZEOF function	100
SQRT root function	96
TRUNC function	97
WC function	101
RCL keywords.....	303
RCL library functions.....	30
RCL motion statements.....	117
Direct	117
Motion-influencing	134
RCL parameters.....	157
Axis-specific	157
C-parameters of all kinematics	169
Parameters of a kinematics	171
RCL program call	
External programs	47
Main program	47
Subroutine	50
Subroutine sequence	52
RCL program control.....	77
CASE statement	88
HALT statement	81
IF-THEN--ELSE) statement	85
JUMP statement	83
PAUSE statement	81
REPEAT statement	82
WAIT statement	77
RCL programming	
Blending mode	113
Position-accurate switching	105
RCL program structure.....	41
Compiler	41
Compiler statements	42
Main program call	47
Main program structure	44
Program declaration	46
Subroutine declaration	50
Subroutine sequence	52
RCL special functions.....	103
Call	103
Position-accurate switching	105
RCL variable declaration.....	63
RCL variables.....	59
Array variables	69
PLC communication channels	70
Points and point file "pkt"	66
RCL data types	59
RCL global variables	64
RCL point variables	65
RCL variable declaration	63
Text variables	68
READ.....	61

R

REAL.....	60
RECORD.....	62
RECORD_END.....	62
Record types.....	62
REPEAT.....	69, 82
REPEAT_END.....	54, 82
REPEAT n TIMES.....	54, 82
RETURN.....	50
RobotControl	
Compiler	228
Context menu "Global Data"	212
Debugger	230
Folder "Robot" – Dialog: Scaling/Units	216
Folder "Robot" - Dialog for axis lengths and coupling	216
Folder Robot – Dialog: Coordinate Offset ..	219
Folder Robot – Dialog: Coupling Factors	
Belt	218
Folder Robot – Dialog: Jogging	219
Folder Robot – Dialog: Motion Limit Val- ues	217
Folder Robot – Dialog: Workpiece coordi- nate Assignment	223
Folder Robot – Kinematic dialogs	216
General information	209
Kinematics - context menu	213
Kinematics – context menu: Diagnostics ..	214
Kinematics – context menu: Setup	214
Menu structure: Kinematics	210
Options, compiler	233
Options, connection settings	233
Options, text editor	235
Program for tracing	229
RCL editor	223
Robot – Dialog: Safe zones folder	221
Settable options for RoCo	232
Robot Control	
RCL editor, create new RCL program	224
RCL editor, Work with editor	224
RoCo editor, General Information.....	235
RoCo editor, keywords.....	241
RoCo editor, page margins.....	240
RoCo-Editor, page margins – frame color "in the focus", frame color "without focus".....	241
RoCo-Editor, page margins - highlight brack- ets.....	241
RoCo-Editor, page margins - highlight line.....	241
RoCo-Editor, page margins - show line num- bers.....	241
RoCo editor, text area.....	236
RoCo-Editor, text area - associated brackets...	239
RoCo-Editor, text area - cursor settings.....	238
RoCo-Editor, text area - EOL marks.....	240
RoCo-Editor, text area - foreground color, background color and font type of the editor.....	237

Index

R

RoCo-Editor, text area - foreground color, background color and font type of the page margin.....	240
RoCo-Editor, text area - highlight line.....	239
RoCo-Editor, text area – Selection, Region foreground, Region background, Region style..	237
RoCo-Editor, text editor – Automatic linefeed...	236
RoCo-Editor, text editor – indent width.....	236
RoCo editor, text editor – Maximum number of "Undo" operations.....	236
RoCo editor, text editor – Show auto-complete upon entry.....	235
RoCo-Editor, text editor – Tabulator size.....	236
RoCo multi-kinematics.....	243
Coordinated motion	247
Define axis assignment	245
Define kinematic type	244
Delete kinematics	246
Overview	243
PLC shared memory interfaces (KinData) .	251
Point tables	249
Point tables, assignment and response	250
RCL programs	248
RoCo programming environment.....	209
Room multi-kinematics	
Create kinematics	243
ROUND.....	97

S

Sequence of operations.....	92
Setup mode.....	153
SIN.....	95
SIZEOF.....	100
Source file.....	41
Spatial blending.....	142
Overview	142
Spatial blending for BLOCK_SLOPE.....	147
Spatial blending for PROGR_SLOPE and CONT_SLOPE.....	143
SPC_FCT	
List as table	104
SPC_FCT: 1	
Position-accurate switching of digital outputs on the path	106
SPC_FCT: 2	
Position-accurate switching of process data on the path	107
SPC_FCT: 21	
Belt type	108
SPC_FCT: 23	
DateTime	109
SPC_FCT: 24	
SysClk	109
SPC_FCT: 28	
Set belt counter	110

S

SPC_FCT: 53	
Belt synchronized workspace with belt type 4	110
SPC_FCT: 59	
Query blending mode	114
SPC_FCT: 60	
Set blending mode	113
SQRT.....	96
SUB_END.....	50
SUBROUTINE.....	50
Support	
See service hotline	313
sym file.....	42
SYNC.....	135
SYNCHRON.....	135
SYNCHRON_END.....	135

T

Target point names.....	122
TEXT.....	61
Text variables.....	68
THEN.....	85
Tool coordinate system.....	36
Tool coordinate system	36
Tool coordinate system (TCS).....	24
Tool management.....	213
Transformation rule	
Overview	255
Robot type 00, Cartesian robot	255
Robot type 09, 3-axis Scara with 1 optional manual axis	261
Robot type 10, 4-axis Scara with 1 optional manual axis	264
Robot type 100, PLC	301
Robot type 12, 2-axis parallelogram	267
Robot type 13, 2-axis parallelogram	269
Robot type 18, Contact kinematics	271
Robot type 19, 2-axis parallelogram V1	272
Robot type 2, 3-axis delta with 1 optional manual axis	259
Robot type 20, 3-axis Delta with 1 optional manual axis	274
Robot type 21, 2-axis parallelogram V2	280
Robot type 29, 2-axis delta with rotary and translatory axis	294
Robot type 30, Adabot	296
Robot type 32, 2-axis delta with asymmetric TCP	299
Robot with offset wrist with six (seven) axes	282
Robot with offset wrist with six axes + three linear axes	287
Travel range limits.....	39
TRUNC.....	97
TYPE.....	61

U

UNTIL.....	91
Use coordinate systems.....	25

V

V.....	127
V_PTP.....	127
VALUE.....	52
Value assignments.....	89
VAR.....	60
Velocity.....	127
Velocity, effective.....	129
Velocity factor	
VFACTOR	129
Velocity specification	
Global	128
Statement-related	128
VFACTOR.....	129
VFIX.....	128
VFIX_PTP.....	128

W

WAIT.....	77

W

WC.....	67, 101
WC_FRAME.....	57
WC_NAMES.....	42
WC_SYSTEM.....	57
WC_UR.....	57
WITH.....	128
Workpiece coordinate editor.....	29
Workpiece coordinate file.....	29
Workpiece coordinates.....	212
Workpiece coordinate system (WpCS).....	23
Workpiece coordinate system relative to the world coordinate system.....	27
World coordinate system	
Basic coordinate system	25
Workpiece coordinate system	27
World coordinate system (WCS).....	22
WRITE.....	61

Notes

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911336297

DOK-MLC***-RCL*PRO*V13-AP01-EN-P