

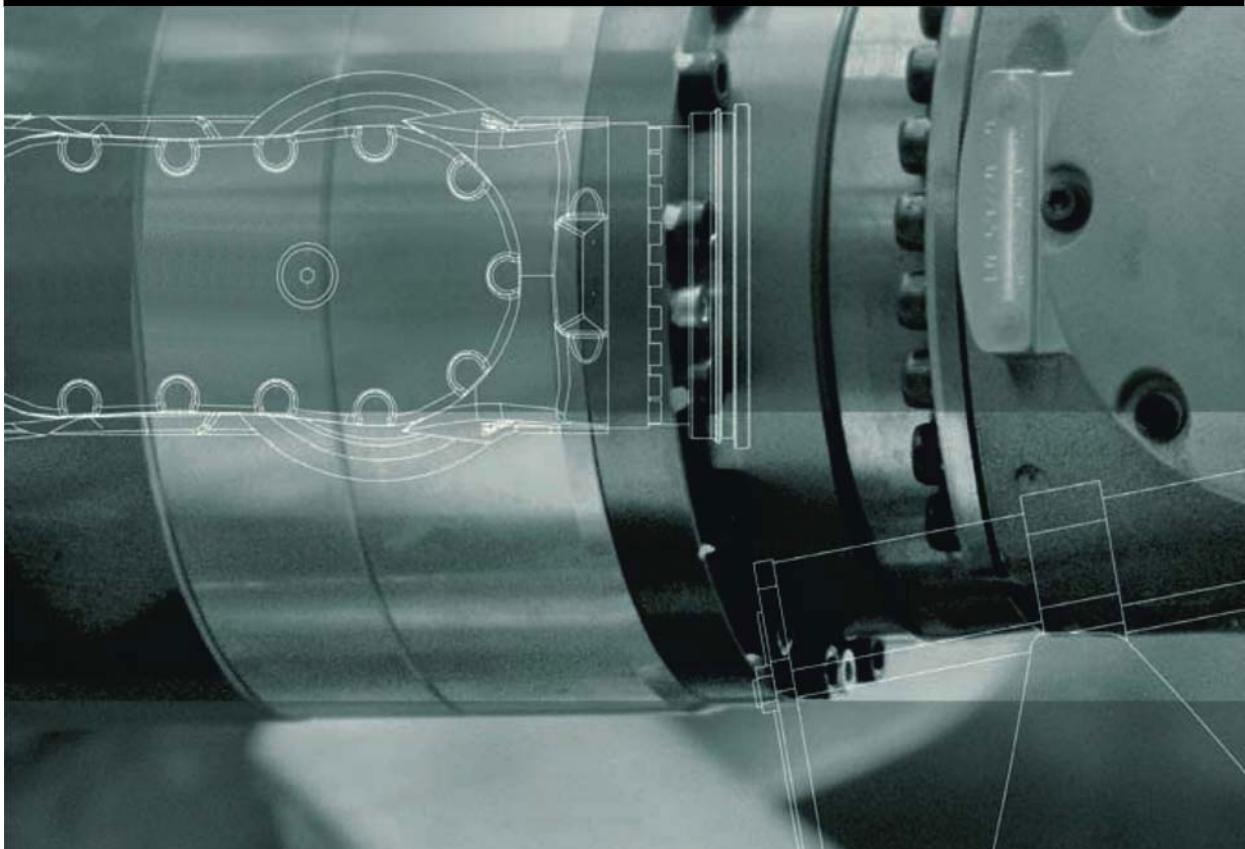
# KUKA

KUKA System Software

KUKA Roboter GmbH

## KUKA System Software 8.2

Operating and Programming Instructions for System Integrators



Issued: 02.11.2011

Version: KSS 8.2 SI V3 en

© Copyright 2011

KUKA Roboter GmbH  
Zugspitzstraße 140  
D-86165 Augsburg  
Germany

This documentation or excerpts therefrom may not be reproduced or disclosed to third parties without the express permission of KUKA Roboter GmbH.

Other functions not described in this documentation may be operable in the controller. The user has no claims to these functions, however, in the case of a replacement or service work.

We have checked the content of this documentation for conformity with the hardware and software described. Nevertheless, discrepancies cannot be precluded, for which reason we are not able to guarantee total conformity. The information in this documentation is checked on a regular basis, however, and necessary corrections will be incorporated in the subsequent edition.

Subject to technical alterations without an effect on the function.

Translation of the original documentation

KIM-PS5-DOC

Publication:	Pub KSS 8.2 SI en
Bookstructure:	KSS 8.2 SI V3.2
Version:	KSS 8.2 SI V3 en

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>13</b>
1.1	Target group .....	13
1.2	Industrial robot documentation .....	13
1.3	Representation of warnings and notes .....	13
1.4	Trademarks .....	14
<b>2</b>	<b>Product description .....</b>	<b>15</b>
2.1	Overview of the industrial robot .....	15
2.2	Overview of the software components .....	15
2.3	Overview of KUKA System Software (KSS) .....	15
<b>3</b>	<b>Safety .....</b>	<b>17</b>
3.1	General .....	17
3.1.1	Liability .....	17
3.1.2	Intended use of the industrial robot .....	17
3.1.3	EC declaration of conformity and declaration of incorporation .....	18
3.1.4	Terms used .....	18
3.2	Personnel .....	20
3.3	Workspace, safety zone and danger zone .....	21
3.4	Triggers for stop reactions .....	22
3.5	Safety functions .....	23
3.5.1	Overview of the safety functions .....	23
3.5.2	Safety controller .....	23
3.5.3	Mode selection .....	24
3.5.4	Operator safety .....	24
3.5.5	EMERGENCY STOP device .....	25
3.5.6	Logging off the higher-level safety controller .....	25
3.5.7	External EMERGENCY STOP device .....	25
3.5.8	Enabling device .....	26
3.5.9	External enabling device .....	26
3.5.10	External safe operational stop .....	26
3.5.11	External safety stop 1 and external safety stop 2 .....	26
3.5.12	Velocity monitoring in T1 .....	27
3.6	Additional protective equipment .....	27
3.6.1	Jog mode .....	27
3.6.2	Software limit switches .....	27
3.6.3	Mechanical end stops .....	27
3.6.4	Mechanical axis range limitation (optional) .....	27
3.6.5	Axis range monitoring (optional) .....	28
3.6.6	Release device (optional) .....	28
3.6.7	Labeling on the industrial robot .....	29
3.6.8	External safeguards .....	29
3.7	Overview of operating modes and safety functions .....	30
3.8	Safety measures .....	30
3.8.1	General safety measures .....	30
3.8.2	Transportation .....	31
3.8.3	Start-up and recommissioning .....	32

3.8.3.1	Start-up mode .....	33
3.8.4	Manual mode .....	34
3.8.5	Simulation .....	35
3.8.6	Automatic mode .....	35
3.8.7	Maintenance and repair .....	35
3.8.8	Decommissioning, storage and disposal .....	37
3.8.9	Safety measures for "single point of control" .....	37
3.9	Applied norms and regulations .....	39
<b>4</b>	<b>Operation .....</b>	<b>41</b>
4.1	KUKA smartPAD teach pendant .....	41
4.1.1	Front view .....	41
4.1.2	Rear view .....	43
4.1.3	Disconnecting and connecting the smartPAD .....	44
4.2	KUKA smartHMI user interface .....	45
4.2.1	Status bar .....	46
4.2.2	Keypad .....	47
4.2.3	Minimizing KUKA smartHMI .....	48
4.3	Switching on the robot controller and starting the KSS .....	48
4.4	Calling the main menu .....	48
4.5	Defining the start type for KSS .....	49
4.6	Exiting or restarting KSS .....	49
4.7	Switching the robot controller off .....	52
4.8	Setting the user interface language .....	52
4.9	Changing user group .....	52
4.10	Changing operating mode .....	53
4.11	Coordinate systems .....	54
4.12	Jogging the robot .....	55
4.12.1	"Jog options" window .....	56
4.12.1.1	"General" tab .....	57
4.12.1.2	"Keys" tab .....	57
4.12.1.3	"Mouse" tab .....	58
4.12.1.4	"KCP pos." tab .....	59
4.12.1.5	"Cur. tool/base" tab .....	59
4.12.2	Activating the jog mode .....	60
4.12.3	Setting the jog override (HOV) .....	60
4.12.4	Selecting the tool and base .....	60
4.12.5	Axis-specific jogging with the jog keys .....	61
4.12.6	Cartesian jogging with the jog keys .....	61
4.12.7	Configuring the Space Mouse .....	61
4.12.8	Defining the alignment of the Space Mouse .....	63
4.12.9	Cartesian jogging with the Space Mouse .....	64
4.12.10	Incremental jogging .....	65
4.13	Jogging external axes .....	66
4.14	Bypassing workspace monitoring .....	66
4.15	Monitor functions .....	67
4.15.1	Displaying the actual position .....	67
4.15.2	Displaying digital inputs/outputs .....	67
4.15.3	Displaying analog inputs/outputs .....	69

4.15.4	Displaying inputs/outputs for Automatic External .....	69
4.15.5	Displaying and modifying the value of a variable .....	70
4.15.6	Displaying the state of a variable .....	71
4.15.7	Displaying the variable overview and modifying variables .....	72
4.15.8	Displaying cyclical flags .....	73
4.15.9	Displaying flags .....	74
4.15.10	Displaying counters .....	75
4.15.11	Displaying timers .....	76
4.15.12	Displaying calibration data .....	77
4.15.13	Displaying information about the robot and robot controller .....	78
4.15.14	Displaying/editing robot data .....	78
4.16	Displaying the battery state .....	80
<b>5</b>	<b>Start-up and recommissioning .....</b>	<b>83</b>
5.1	Start-up wizard .....	83
5.2	Checking the machine data .....	83
5.3	Defining hardware options .....	84
5.4	Changing the safety ID of the PROFINET device .....	85
5.5	Jogging the robot without a higher-level safety controller .....	86
5.6	Checking the activation of the positionally accurate robot model .....	87
5.7	Activating palletizing mode .....	87
5.8	Copying machine data .....	88
5.9	Mastering .....	89
5.9.1	Mastering methods .....	90
5.9.2	Moving axes to the pre-mastering position .....	91
5.9.3	Mastering with the EMD .....	92
5.9.3.1	First mastering with the EMD .....	92
5.9.3.2	Teach offset .....	95
5.9.3.3	Master load with offset .....	96
5.9.4	Mastering with the dial gauge .....	97
5.9.5	Mastering external axes .....	98
5.9.6	Reference mastering .....	98
5.9.7	Manually unmastering axes .....	99
5.10	Modifying software limit switches .....	100
5.11	Calibration .....	102
5.11.1	Defining the tool direction .....	102
5.11.2	Tool calibration .....	102
5.11.2.1	TCP calibration: XYZ 4-point method .....	104
5.11.2.2	TCP calibration: XYZ Reference method .....	106
5.11.2.3	Defining the orientation: ABC World method .....	107
5.11.2.4	Defining the orientation: ABC 2-point method .....	107
5.11.2.5	Numeric input .....	109
5.11.3	Base calibration .....	109
5.11.3.1	3-point method .....	110
5.11.3.2	Indirect method .....	111
5.11.3.3	Numeric input .....	112
5.11.4	Fixed tool calibration .....	112
5.11.4.1	Calibrating an external TCP .....	113
5.11.4.2	Entering the external TCP numerically .....	114
5.11.4.3	Workpiece calibration: direct method .....	114

5.11.4.4 Workpiece calibration: indirect method .....	116
5.11.5 Renaming the tool/base .....	117
5.11.6 Linear unit .....	117
5.11.6.1 Checking whether the linear unit needs to be calibrated .....	117
5.11.6.2 Calibrating the linear unit .....	118
5.11.6.3 Entering the linear unit numerically .....	119
5.11.7 Calibrating an external kinematic system .....	120
5.11.7.1 Calibrating the root point .....	120
5.11.7.2 Entering the root point numerically .....	122
5.11.7.3 Workpiece base calibration .....	122
5.11.7.4 Entering the workpiece base numerically .....	124
5.11.7.5 Calibrating an external tool .....	124
5.11.7.6 Entering the external tool numerically .....	125
5.12 Load data .....	126
5.12.1 Checking loads with KUKA.Load .....	126
5.12.2 Calculating payloads with KUKA.LoadDataDetermination .....	126
5.12.3 Entering payload data .....	126
5.12.4 Entering supplementary load data .....	127
5.12.5 Online load data check (OLDC) .....	127
5.13 Maintenance handbook .....	129
5.13.1 Logging maintenance .....	130
5.13.2 Displaying a maintenance log .....	131
<b>6 Configuration .....</b>	<b>133</b>
6.1 Configuring the KUKA Line Interface (KLI) .....	133
6.1.1 Configuring the Windows interface (without PROFINET) .....	133
6.1.2 Configuring the Windows PROFINET interface and creating the Windows interface .....	134
6.1.3 Displaying ports of the Windows interface or enabling an additional port .....	136
6.1.4 Displaying or modifying filters .....	136
6.1.5 Displaying the subnet configuration of the robot controller .....	137
6.1.6 Error display in the Address and Subnet boxes .....	138
6.1.7 Configuring DNS .....	138
6.2 Reconfiguring the I/O driver .....	141
6.3 Checking the safety configuration of the robot controller .....	141
6.4 Configuring the variable overview .....	142
6.5 Changing the password .....	143
6.6 Configuring workspaces .....	143
6.6.1 Configuring Cartesian workspaces .....	144
6.6.2 Configuring axis-specific workspaces .....	146
6.6.3 Mode for workspaces .....	148
6.7 Warm-up .....	149
6.7.1 Configuring warm-up .....	149
6.7.2 Warm-up sequence .....	149
6.7.3 System variables for warm-up .....	150
6.8 Collision detection .....	151
6.8.1 Calculating the tolerance range and activating collision detection .....	153
6.8.2 Defining an offset for the tolerance range .....	154
6.8.3 Option window “Collision detection” .....	155
6.8.4 Torque monitoring .....	156

6.8.4.1 Determining values for torque monitoring .....	157
6.8.4.2 Programming torque monitoring .....	157
6.9 Defining calibration tolerances .....	158
6.10 Configuring Automatic External .....	158
6.10.1 Configuring CELL.SRC .....	159
6.10.2 Configuring Automatic External inputs/outputs .....	160
6.10.2.1 Automatic External inputs .....	162
6.10.2.2 Automatic External outputs .....	164
6.10.3 Transmitting error numbers to the higher-level controller .....	166
6.10.4 Signal diagrams .....	168
6.11 Torque mode .....	174
6.11.1 Overview of torque mode .....	174
6.11.1.1 Using torque mode .....	174
6.11.1.2 Robot program example: setting A1 to "soft" in both directions .....	176
6.11.2 Activating torque mode: SET_TORQUE_LIMITS() .....	177
6.11.3 Deactivating torque mode: RESET_TORQUE_LIMITS() .....	180
6.11.4 Interpreter specifics .....	180
6.11.5 Diagnostic variables for torque mode .....	181
6.11.5.1 \$TORQUE_AXIS_ACT .....	181
6.11.5.2 \$TORQUE_AXIS_MAX_0 .....	182
6.11.5.3 \$TORQUE_AXIS_MAX .....	182
6.11.5.4 \$TORQUE_AXIS_LIMITS .....	182
6.11.5.5 \$HOLDING_TORQUE .....	183
6.11.5.6 Comparison: \$TORQUE_AXIS_ACT and \$HOLDING_TORQUE .....	183
6.11.6 Other examples .....	184
6.11.6.1 Robot program: setting axis to "soft" in both directions .....	184
6.11.6.2 Robot program: avoiding damage in the event of collisions .....	184
6.11.6.3 Robot program: torque mode in the interrupt .....	186
6.11.6.4 Robot program: servo gun builds up pressure .....	187
6.11.6.5 Submit program: servo gun builds up pressure .....	188
6.12 Event planner .....	188
6.12.1 Configuring a data comparison .....	188
6.12.2 Configuring T1 and T2 Consistency, AUT and EXT Consistency .....	189
6.12.3 Configuring Logic Consistency .....	190
6.13 Brake test .....	191
6.13.1 Overview of the brake test .....	191
6.13.2 Programs for the brake test .....	192
6.13.3 Signal diagram of the brake test .....	193
6.13.4 Configuring input and output signals .....	194
6.13.5 Teaching positions for the brake test .....	195
6.13.6 Performing a manual brake test .....	196
6.13.7 Checking that the brake test is functioning correctly .....	197
<b>7 Program management .....</b>	<b>199</b>
7.1 Navigator file manager .....	199
7.1.1 Selecting filters .....	200
7.1.2 Creating a new folder .....	200
7.1.3 Creating a new program .....	201
7.1.4 Renaming a file .....	201
7.2 Selecting or opening a program .....	201

7.2.1	Selecting and deselecting a program .....	202
7.2.2	Opening a program .....	203
7.2.3	Toggling between the Navigator and the program .....	203
7.3	Structure of a KRL program .....	204
7.3.1	HOME position .....	205
7.4	Displaying/hiding program sections .....	205
7.4.1	Displaying/hiding the DEF line .....	205
7.4.2	Activating detail view .....	205
7.4.3	Activating/deactivating the line break function .....	206
7.4.4	Displaying Folds .....	206
7.5	Starting a program .....	207
7.5.1	Selecting the program run mode .....	207
7.5.2	Program run modes .....	208
7.5.3	Advance run .....	208
7.5.4	Setting the program override (POV) .....	209
7.5.5	Switching drives on/off .....	209
7.5.6	Robot interpreter status indicator .....	209
7.5.7	Starting a program forwards (manual) .....	210
7.5.8	Starting a program forwards (automatic) .....	210
7.5.9	Carrying out a block selection .....	211
7.5.10	Starting a program backwards .....	211
7.5.11	Resetting a program .....	211
7.5.12	Starting Automatic External mode .....	212
7.6	Editing a program .....	212
7.6.1	Inserting a comment or stamp .....	213
7.6.2	Deleting program lines .....	214
7.6.3	Creating Folds .....	214
7.6.4	Additional editing functions .....	215
7.7	Printing a program .....	216
7.8	Archiving and restoring data .....	216
7.8.1	Archiving overview .....	216
7.8.2	Archiving to a USB stick .....	217
7.8.3	Archiving on the network .....	218
7.8.4	Archiving the logbook .....	218
7.8.5	Restoring data .....	219
7.9	Project management .....	219
7.9.1	Pinning a project on the robot controller .....	219
7.9.2	Activating a project .....	220
7.9.3	<b>Project management window</b> .....	221
<b>8</b>	<b>Basic principles of motion programming .....</b>	<b>223</b>
8.1	Overview of motion types .....	223
8.2	Motion type PTP .....	223
8.3	Motion type LIN .....	223
8.4	Motion type CIRC .....	224
8.5	Approximate positioning .....	224
8.6	Orientation control LIN, CIRC .....	226
8.6.1	Combinations of \$ORI_TYPE and \$CIRC_TYPE .....	227
8.7	Motion type "Spline" .....	229

8.7.1	Velocity profile for spline motions .....	231
8.7.2	Block selection with spline motions .....	232
8.7.3	Modifications to spline blocks .....	233
8.7.4	Approximate positioning of spline motions .....	235
8.7.5	Replacing an approximated motion with a spline block .....	236
8.7.5.1	SLIN-SPL-SLIN transition .....	239
8.8	Orientation control SPLINE .....	239
8.8.1	SCIRC: reference system for the orientation control .....	241
8.8.2	SCIRC: orientation behavior .....	242
8.9	Status and Turn .....	244
8.9.1	Status .....	245
8.9.2	Turn .....	247
8.10	Singularities .....	248
<b>9</b>	<b>Programming for user group “User” (inline forms) .....</b>	<b>251</b>
9.1	Names in inline forms .....	251
9.2	Programming PTP, LIN and CIRC motions .....	251
9.2.1	Programming a PTP motion .....	251
9.2.2	Inline form “PTP” .....	252
9.2.3	Programming a LIN motion .....	252
9.2.4	Inline form “LIN” .....	253
9.2.5	Programming a CIRC motion .....	253
9.2.6	Inline form “CIRC” .....	254
9.2.7	Option window: <b>Frames</b> .....	<b>255</b>
9.2.8	Option window: <b>Motion parameters</b> (PTP) .....	255
9.2.9	Option window: <b>Motion parameters</b> (LIN, CIRC) .....	256
9.3	Spline motions .....	257
9.3.1	Programming tips for spline motions .....	257
9.3.2	Programming a SLIN motion (individual motion) .....	258
9.3.2.1	Inline form “SLIN” .....	258
9.3.2.2	Option window “Motion parameters” (SLIN) .....	259
9.3.3	Programming a SCIRC motion (individual motion) .....	260
9.3.3.1	Inline form “SCIRC” .....	260
9.3.3.2	Option window “Motion parameters” (SCIRC) .....	261
9.3.4	Programming a spline block .....	262
9.3.4.1	Inline form for spline block .....	263
9.3.4.2	Option window “Frames” (spline block) .....	263
9.3.4.3	Option window “Motion parameters” (spline block) .....	264
9.3.4.4	Programming an SPL or SLIN segment .....	265
9.3.4.5	Programming an SCIRC segment .....	265
9.3.4.6	Inline form for spline segment .....	266
9.3.4.7	Option window “Frames” (spline segment) .....	267
9.3.4.8	Option window “Motion parameters” (spline segment) .....	267
9.3.4.9	Programming triggers in the spline block .....	268
9.3.4.10	Inline form for spline trigger type “Set output” .....	269
9.3.4.11	Inline form for spline trigger type “Set pulse output” .....	270
9.3.4.12	Inline form for spline trigger type “Trigger assignment” .....	271
9.3.4.13	Inline form for spline trigger type “Trigger function call” .....	271
9.3.4.14	Limits for functions in the spline trigger .....	272
9.3.5	Copying spline inline forms .....	273

9.3.6	Converting spline inline forms from 8.1 .....	273
9.4	Modifying motion parameters .....	274
9.5	Modifying the coordinates of a taught point .....	274
9.6	Programming logic instructions .....	274
9.6.1	Inputs/outputs .....	274
9.6.2	Setting a digital output - OUT .....	275
9.6.3	Inline form "OUT" .....	275
9.6.4	Setting a pulse output - PULSE .....	276
9.6.5	Inline form "PULSE" .....	276
9.6.6	Setting an analog output - ANOUT .....	276
9.6.7	Inline form "ANOUT" (static) .....	277
9.6.8	Inline form "ANOUT" (dynamic) .....	277
9.6.9	Programming a wait time - WAIT .....	278
9.6.10	Inline form "WAIT" .....	278
9.6.11	Programming a signal-dependent wait function - WAITFOR .....	278
9.6.12	Inline form "WAITFOR" .....	279
9.6.13	Switching on the path - SYN OUT .....	280
9.6.14	Inline form "SYN OUT", option "START/END" .....	280
9.6.15	Inline form "SYN OUT", option "PATH" .....	283
9.6.16	Setting a pulse on the path - SYN PULSE .....	285
9.6.17	Inline form "SYN PULSE" .....	285
9.6.18	Modifying a logic instruction .....	286
<b>10</b>	<b>Programming for user group "Expert" (KRL syntax) .....</b>	<b>287</b>
10.1	Overview of KRL syntax .....	287
10.2	Symbols and fonts .....	288
10.3	Important KRL terms .....	289
10.3.1	SRC files and DAT files .....	289
10.3.2	Subprograms and functions .....	289
10.3.3	Naming conventions and keywords .....	290
10.3.4	Data types .....	291
10.3.5	Areas of validity .....	292
10.3.6	Constants .....	293
10.4	Variables and declarations .....	293
10.4.1	DECL .....	293
10.4.2	ENUM .....	295
10.4.3	STRUC .....	296
10.5	Motion programming: PTP, LIN, CIRC .....	297
10.5.1	PTP .....	297
10.5.2	PTP_REL .....	298
10.5.3	LIN .....	299
10.5.4	LIN_REL .....	300
10.5.5	CIRC .....	302
10.5.6	CIRC_REL .....	303
10.6	Motion programming: spline .....	304
10.6.1	Higher motion profile for spline motions .....	304
10.6.2	SPLINE ... ENDSPLINE .....	305
10.6.3	SLIN .....	306
10.6.4	SCIRC .....	307

10.6.5 SPL .....	309
10.6.6 TIME_BLOCK .....	309
10.6.7 System variables that affect the spline? .....	313
10.7 Program execution control .....	314
10.7.1 CONTINUE .....	314
10.7.2 EXIT .....	314
10.7.3 FOR ... TO ... ENDFOR .....	315
10.7.4 GOTO .....	315
10.7.5 HALT .....	316
10.7.6 IF ... THEN ... ENDIF .....	316
10.7.7 LOOP ... ENDLOOP .....	317
10.7.8 REPEAT ... UNTIL .....	317
10.7.9 SWITCH ... CASE ... ENDSWITCH .....	318
10.7.10 WAIT FOR .....	319
10.7.11 WAIT SEC .....	320
10.7.12 WHILE ... ENDWHILE .....	320
10.8 Inputs/outputs .....	321
10.8.1 ANIN .....	321
10.8.2 ANOUT .....	322
10.8.3 PULSE .....	323
10.8.4 SIGNAL .....	326
10.9 Subprograms and functions .....	327
10.9.1 RETURN .....	327
10.10 Interrupt programming .....	328
10.10.1 BRAKE .....	328
10.10.2 INTERRUPT ... DECL ... WHEN ... DO .....	329
10.10.3 INTERRUPT .....	330
10.10.4 RESUME .....	332
10.11 Path-related switching actions (=Trigger) .....	333
10.11.1 TRIGGER WHEN DISTANCE .....	333
10.11.2 TRIGGER WHEN PATH .....	336
10.11.3 TRIGGER WHEN PATH (for SPLINE) .....	339
10.11.3.1 Spline: trigger point in the case of approximate positioning .....	341
10.12 Communication .....	343
10.13 System functions .....	343
10.13.1 VARSTATE() .....	343
10.13.2 ROB_STOP() and ROB_STOP_RELEASE() .....	345
10.14 Editing string variables .....	346
10.14.1 String variable length in the declaration .....	346
10.14.2 String variable length after initialization .....	346
10.14.3 Deleting the contents of a string variable .....	347
10.14.4 Extending a string variable .....	347
10.14.5 Searching a string variable .....	348
10.14.6 Comparing the contents of string variables .....	349
10.14.7 Copying a string variable .....	349
<b>11 Submit interpreter .....</b>	<b>351</b>
11.1 Function of the Submit interpreter .....	351
11.2 Manually stopping or deselecting the Submit interpreter .....	352

11.3	Manually starting the Submit interpreter .....	352
11.4	Modifying the program SPS.SUB .....	353
11.5	Creating a new SUB program .....	353
11.6	Programming .....	354
<b>12</b>	<b>Diagnosis .....</b>	<b>357</b>
12.1	Logbook .....	357
12.1.1	Displaying the logbook .....	357
12.1.2	“Log” tab .....	357
12.1.3	“Filter” tab .....	358
12.1.4	Configuring the logbook .....	359
12.2	Displaying the caller stack .....	360
12.3	Displaying interrupts .....	361
12.4	Displaying diagnostic data about the kernel system .....	362
12.5	Compressing data for error analysis at KUKA .....	362
<b>13</b>	<b>Installation .....</b>	<b>363</b>
13.1	Overview of the software components .....	363
13.2	Installing Windows and KSS (from image) .....	363
13.3	Installing additional software .....	364
13.4	KSS update .....	365
13.4.1	Update from the KUKA USB stick .....	365
13.4.2	Update from the network .....	366
<b>14</b>	<b>Messages .....</b>	<b>369</b>
14.1	Automatic External error messages .....	369
14.2	Brake test messages .....	370
14.3	Error messages of the positionally accurate robot model .....	372
<b>15</b>	<b>KUKA Service .....</b>	<b>375</b>
15.1	Requesting support .....	375
15.2	KUKA Customer Support .....	375
	<b>Index .....</b>	<b>383</b>

# 1 Introduction

## 1.1 Target group

This documentation is aimed at users with the following knowledge and skills:

- Advanced knowledge of the robot controller system
- Advanced KRL programming skills



For optimal use of our products, we recommend that our customers take part in a course of training at KUKA College. Information about the training program can be found at [www.kuka.com](http://www.kuka.com) or can be obtained directly from our subsidiaries.

## 1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

- Documentation for the manipulator
- Documentation for the robot controller
- Operating and programming instructions for the KUKA System Software
- Documentation relating to options and accessories
- Parts catalog on storage medium

Each of these sets of instructions is a separate document.

## 1.3 Representation of warnings and notes

### Safety

These warnings are relevant to safety and **must** be observed.



**DANGER** These warnings mean that it is certain or highly probable that death or severe physical injury **will** occur, if no precautions are taken.



**WARNING** These warnings mean that death or severe physical injury **may** occur, if no precautions are taken.



**CAUTION** These warnings mean that minor physical injuries **may** occur, if no precautions are taken.



**NOTICE** These warnings mean that damage to property **may** occur, if no precautions are taken.



These warnings contain references to safety-relevant information or general safety measures. These warnings do not refer to individual hazards or individual precautionary measures.

### Hints

These hints serve to make your work easier or contain references to further information.



Tip to make your work easier or reference to further information.

## 1.4 Trademarks

**Windows** is a trademark of Microsoft Corporation.

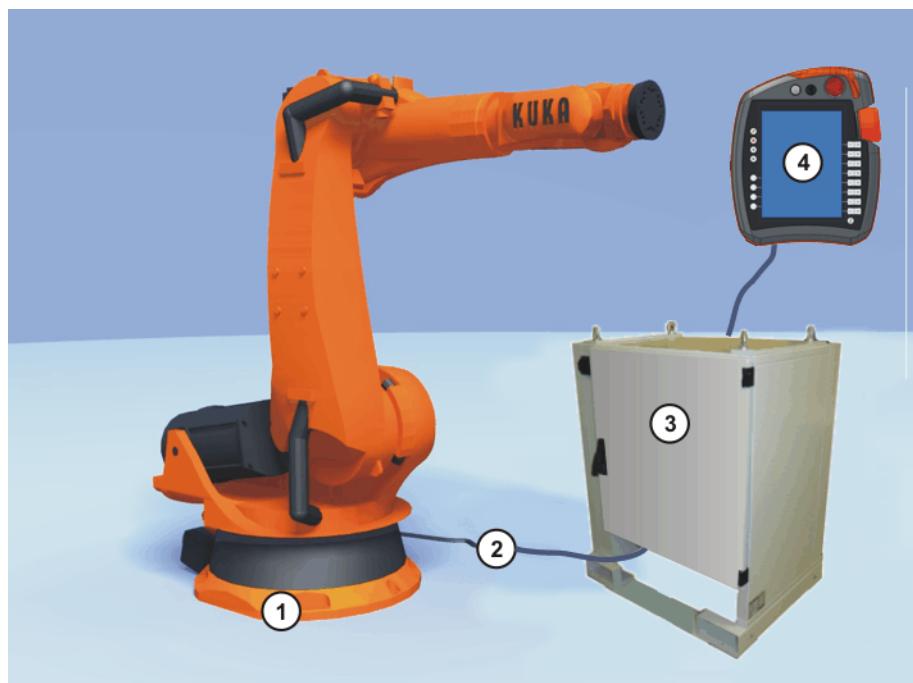
**WordPad** is a trademark of Microsoft Corporation.

## 2 Product description

### 2.1 Overview of the industrial robot

The industrial robot consists of the following components:

- Manipulator
- Robot controller
- Teach pendant
- Connecting cables
- Software
- Options, accessories



**Fig. 2-1: Example of an industrial robot**

- |                     |                    |
|---------------------|--------------------|
| 1 Manipulator       | 3 Robot controller |
| 2 Connecting cables | 4 Teach pendant    |

### 2.2 Overview of the software components

#### Overview

The following software components are used:

- KUKA System Software 8.2
- Windows XPe V3.0.0

### 2.3 Overview of KUKA System Software (KSS)

#### Description

The KUKA System Software (KSS) is responsible for all the basic operator control functions of the industrial robot.

- Path planning
- I/O management
- Data and file management
- etc.

Additional technology packages, containing application-specific instructions and configurations, can be installed.

**smartHMI**

The user interface of the KUKA System Software is called KUKA smartHMI (smart Human-Machine Interface).

Features:

- User administration
- Program editor
- KRL (KUKA Robot Language)
- Inline forms for programming
- Message display
- Configuration window
- etc.

(>>> 4.2 "KUKA smartHMI user interface" Page 45)



Depending on customer-specific settings, the user interface may vary from the standard interface.

## 3 Safety

### 3.1 General

#### 3.1.1 Liability

The device described in this document is either an industrial robot or a component thereof.

Components of the industrial robot:

- Manipulator
- Robot controller
- Teach pendant
- Connecting cables
- External axes (optional)  
e.g. linear unit, turn-tilt table, positioner
- Software
- Options, accessories

The industrial robot is built using state-of-the-art technology and in accordance with the recognized safety rules. Nevertheless, misuse of the industrial robot may constitute a risk to life and limb or cause damage to the industrial robot and to other material property.

The industrial robot may only be used in perfect technical condition in accordance with its intended use and only by safety-conscious persons who are fully aware of the risks involved in its operation. Use of the industrial robot is subject to compliance with this document and with the declaration of incorporation supplied together with the industrial robot. Any functional disorders affecting the safety of the industrial robot must be rectified immediately.

#### Safety information

Safety information cannot be held against KUKA Roboter GmbH. Even if all safety instructions are followed, this is not a guarantee that the industrial robot will not cause personal injuries or material damage.

No modifications may be carried out to the industrial robot without the authorization of KUKA Roboter GmbH. Additional components (tools, software, etc.), not supplied by KUKA Roboter GmbH, may be integrated into the industrial robot. The user is liable for any damage these components may cause to the industrial robot or to other material property.

In addition to the Safety chapter, this document contains further safety instructions. These must also be observed.

#### 3.1.2 Intended use of the industrial robot

The industrial robot is intended exclusively for the use designated in the "Purpose" chapter of the operating instructions or assembly instructions.



Further information is contained in the "Purpose" chapter of the operating instructions or assembly instructions of the industrial robot.

Using the industrial robot for any other or additional purpose is considered impermissible misuse. The manufacturer cannot be held liable for any damage resulting from such use. The risk lies entirely with the user.

Operating the industrial robot and its options within the limits of its intended use also involves observance of the operating and assembly instructions for

the individual components, with particular reference to the maintenance specifications.

<b>Misuse</b>	Any use or application deviating from the intended use is deemed to be impermissible misuse. This includes e.g.: <ul style="list-style-type: none"><li>■ Transportation of persons and animals</li><li>■ Use as a climbing aid</li><li>■ Operation outside the permissible operating parameters</li><li>■ Use in potentially explosive environments</li><li>■ Operation without additional safeguards</li><li>■ Outdoor operation</li></ul>
---------------	---

### 3.1.3 EC declaration of conformity and declaration of incorporation

This industrial robot constitutes partly completed machinery as defined by the EC Machinery Directive. The industrial robot may only be put into operation if the following preconditions are met:

- The industrial robot is integrated into a complete system.  
Or: The industrial robot, together with other machinery, constitutes a complete system.  
Or: All safety functions and safeguards required for operation in the complete machine as defined by the EC Machinery Directive have been added to the industrial robot.
- The complete system complies with the EC Machinery Directive. This has been confirmed by means of an assessment of conformity.

<b>Declaration of conformity</b>	The system integrator must issue a declaration of conformity for the complete system in accordance with the Machinery Directive. The declaration of conformity forms the basis for the CE mark for the system. The industrial robot must be operated in accordance with the applicable national laws, regulations and standards.
----------------------------------	--

The robot controller is CE certified under the EMC Directive and the Low Voltage Directive.

<b>Declaration of incorporation</b>	The industrial robot as partly completed machinery is supplied with a declaration of incorporation in accordance with Annex II B of the EC Machinery Directive 2006/42/EC. The assembly instructions and a list of essential requirements complied with in accordance with Annex I are integral parts of this declaration of incorporation.
-------------------------------------	---

The declaration of incorporation declares that the start-up of the partly completed machinery remains impermissible until the partly completed machinery has been incorporated into machinery, or has been assembled with other parts to form machinery, and this machinery complies with the terms of the EC Machinery Directive, and the EC declaration of conformity is present in accordance with Annex II A.

The declaration of incorporation, together with its annexes, remains with the system integrator as an integral part of the technical documentation of the complete machinery.

### 3.1.4 Terms used

STOP 0, STOP 1 and STOP 2 are the stop definitions according to EN 60204-1:2006.

Term	Description
Axis range	Range of each axis, in degrees or millimeters, within which it may move. The axis range must be defined for each axis.
Stopping distance	Stopping distance = reaction distance + braking distance The stopping distance is part of the danger zone.
Workspace	The manipulator is allowed to move within its workspace. The workspace is derived from the individual axis ranges.
Operator (User)	The user of the industrial robot can be the management, employer or delegated person responsible for use of the industrial robot.
Danger zone	The danger zone consists of the workspace and the stopping distances.
KCP	The KCP (KUKA Control Panel) teach pendant has all the operator control and display functions required for operating and programming the industrial robot.  The KCP variant for the KR C4 is called KUKA smartPAD. The general term "KCP", however, is generally used in this documentation.
Manipulator	The robot arm and the associated electrical installations
Safety zone	The safety zone is situated outside the danger zone.
Safe operational stop	The safe operational stop is a standstill monitoring function. It does not stop the robot motion, but monitors whether the robot axes are stationary. If these are moved during the safe operational stop, a safety stop STOP 0 is triggered.  The safe operational stop can also be triggered externally.  When a safe operational stop is triggered, the robot controller sets an output to the field bus. The output is set even if not all the axes were stationary at the time of triggering, thereby causing a safety stop STOP 0 to be triggered.
Safety STOP 0	A stop that is triggered and executed by the safety controller. The safety controller immediately switches off the drives and the power supply to the brakes.  <b>Note:</b> This stop is called safety STOP 0 in this document.
Safety STOP 1	A stop that is triggered and monitored by the safety controller. The braking process is performed by the non-safety-oriented part of the robot controller and monitored by the safety controller. As soon as the manipulator is at a standstill, the safety controller switches off the drives and the power supply to the brakes.  When a safety STOP 1 is triggered, the robot controller sets an output to the field bus.  The safety STOP 1 can also be triggered externally.  <b>Note:</b> This stop is called safety STOP 1 in this document.
Safety STOP 2	A stop that is triggered and monitored by the safety controller. The braking process is performed by the non-safety-oriented part of the robot controller and monitored by the safety controller. The drives remain activated and the brakes released. As soon as the manipulator is at a standstill, a safe operational stop is triggered.  When a safety STOP 2 is triggered, the robot controller sets an output to the field bus.  The safety STOP 2 can also be triggered externally.  <b>Note:</b> This stop is called safety STOP 2 in this document.

Term	Description
Stop category 0	The drives are deactivated immediately and the brakes are applied. The manipulator and any external axes (optional) perform path-oriented braking.  <b>Note:</b> This stop category is called STOP 0 in this document.
Stop category 1	The manipulator and any external axes (optional) perform path-maintaining braking. The drives are deactivated after 1 s and the brakes are applied.  <b>Note:</b> This stop category is called STOP 1 in this document.
Stop category 2	The drives are not deactivated and the brakes are not applied. The manipulator and any external axes (optional) are braked with a path-maintaining braking ramp.  <b>Note:</b> This stop category is called STOP 2 in this document.
System integrator (plant integrator)	System integrators are people who safely integrate the industrial robot into a complete system and commission it.
T1	Test mode, Manual Reduced Velocity (<= 250 mm/s)
T2	Test mode, Manual High Velocity (> 250 mm/s permissible)
External axis	Motion axis which is not part of the manipulator but which is controlled using the robot controller, e.g. KUKA linear unit, turn-tilt table, Posiflex.

### 3.2 Personnel

The following persons or groups of persons are defined for the industrial robot:

- User
- Personnel



All persons working with the industrial robot must have read and understood the industrial robot documentation, including the safety chapter.

#### User

The user must observe the labor laws and regulations. This includes e.g.:

- The user must comply with his monitoring obligations.
- The user must carry out instructions at defined intervals.

#### Personnel

Personnel must be instructed, before any work is commenced, in the type of work involved and what exactly it entails as well as any hazards which may exist. Instruction must be carried out regularly. Instruction is also required after particular incidents or technical modifications.

Personnel includes:

- System integrator
- Operators, subdivided into:
  - Start-up, maintenance and service personnel
  - Operating personnel
  - Cleaning personnel



Installation, exchange, adjustment, operation, maintenance and repair must be performed only as specified in the operating or assembly instructions for the relevant component of the industrial robot and only by personnel specially trained for this purpose.

#### System integrator

The industrial robot is safely integrated into a complete system by the system integrator.

The system integrator is responsible for the following tasks:

- Installing the industrial robot
- Connecting the industrial robot
- Performing risk assessment
- Implementing the required safety functions and safeguards
- Issuing the declaration of conformity
- Attaching the CE mark
- Creating the operating instructions for the complete system

#### **Operator**

The operator must meet the following preconditions:

- The operator must be trained for the work to be carried out.
- Work on the industrial robot must only be carried out by qualified personnel. These are people who, due to their specialist training, knowledge and experience, and their familiarization with the relevant standards, are able to assess the work to be carried out and detect any potential hazards.

#### **Example**

The tasks can be distributed as shown in the following table.

Tasks	Operator	Programmer	System integrator
Switch robot controller on/off	x	x	x
Start program	x	x	x
Select program	x	x	x
Select operating mode	x	x	x
Calibration (tool, base)		x	x
Master the manipulator		x	x
Configuration		x	x
Programming		x	x
Start-up			x
Maintenance			x
Repair			x
Decommissioning			x
Transportation			x



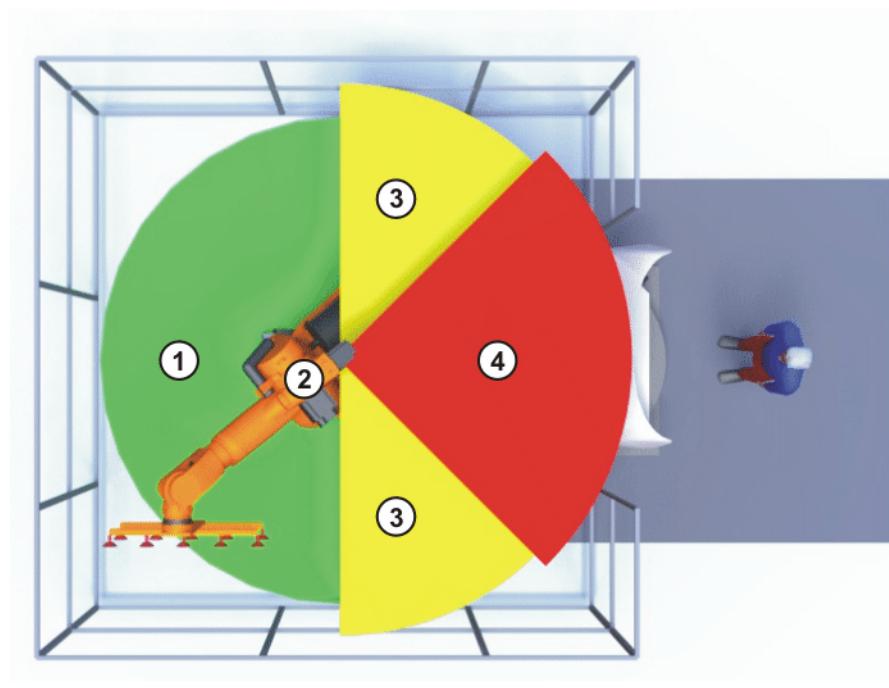
Work on the electrical and mechanical equipment of the industrial robot may only be carried out by specially trained personnel.

### **3.3 Workspace, safety zone and danger zone**

Workspaces are to be restricted to the necessary minimum size. A workspace must be safeguarded using appropriate safeguards.

The safeguards (e.g. safety gate) must be situated inside the safety zone. In the case of a stop, the manipulator and external axes (optional) are braked and come to a stop within the danger zone.

The danger zone consists of the workspace and the stopping distances of the manipulator and external axes (optional). It must be safeguarded by means of physical safeguards to prevent danger to persons or the risk of material damage.



**Fig. 3-1: Example of axis range A1**

- |   |             |   |                   |
|---|-------------|---|-------------------|
| 1 | Workspace   | 3 | Stopping distance |
| 2 | Manipulator | 4 | Safety zone       |

### 3.4 Triggers for stop reactions

Stop reactions of the industrial robot are triggered in response to operator actions or as a reaction to monitoring functions and error messages. The following tables show the different stop reactions according to the operating mode that has been set.

Trigger	T1, T2	AUT, AUT EXT
Start key released	STOP 2	-
STOP key pressed	STOP 2	
Drives OFF	STOP 1	
“Motion enable” input drops out	STOP 2	
Robot controller switched off (power failure)	STOP 0	
Internal error in non-safety-oriented part of the robot controller	STOP 0 or STOP 1 (dependent on the cause of the error)	
Operating mode changed during operation	Safety stop 2	
Safety gate opened (operator safety)	-	Safety stop 1
Enabling switch released	Safety stop 2	-
Enabling switch pressed fully down or error	Safety stop 1	-
E-STOP pressed	Safety stop 1	
Error in safety controller or periphery of the safety controller	Safety stop 0	

## 3.5 Safety functions

### 3.5.1 Overview of the safety functions

The following safety functions are present in the industrial robot:

- Mode selection
- Operator safety (= connection for the guard interlock)
- EMERGENCY STOP device
- Enabling device
- External safe operational stop
- External safety stop 1
- External safety stop 2
- Velocity monitoring in T1

The safety functions of the industrial robot have the following performance:

**Category 3** and **Performance Level d** in accordance with EN ISO 13849-1:2008. This corresponds to **SIL 2** and **HFT 1** in accordance with EN 62061.

This performance only applies under the following conditions, however:

- The EMERGENCY STOP button is pressed at least once every 6 months.

The following components are involved in the safety functions:

- Safety controller in the control PC
- KUKA Control Panel (KUKA smartPAD)
- Cabinet Control Unit (CCU)
- Resolver Digital Converter (RDC)
- KUKA Power Pack (KPP)
- KUKA Servo Pack (KSP)
- Safety Interface Board (SIB) (if used)

There are also interfaces to components outside the industrial robot and to other robot controllers.



**DANGER** In the absence of operational safety functions and safeguards, the industrial robot can cause personal injury or material damage. If safety functions or safeguards are dismantled or deactivated, the industrial robot may not be operated.



During system planning, the safety functions of the overall system must also be planned and designed. The industrial robot must be integrated into this safety system of the overall system.

### 3.5.2 Safety controller

The safety controller is a unit inside the control PC. It links safety-relevant signals and safety-relevant monitoring functions.

Safety controller tasks:

- Switching off the drives; applying the brakes
- Monitoring the braking ramp
- Standstill monitoring (after the stop)
- Velocity monitoring in T1
- Evaluation of safety-relevant signals
- Setting of safety-oriented outputs

### 3.5.3 Mode selection

The industrial robot can be operated in the following modes:

- Manual Reduced Velocity (T1)
- Manual High Velocity (T2)
- Automatic (AUT)
- Automatic External (AUT EXT)



Do not change the operating mode while a program is running. If the operating mode is changed during program execution, the industrial robot is stopped with a safety stop 2.

Operat-ing mode	Use	Velocities
T1	For test operation, pro-gramming and teach-ing	<ul style="list-style-type: none"> <li>■ Program verification: Programmed velocity, maximum 250 mm/s</li> <li>■ Jog mode: Jog velocity, maximum 250 mm/s</li> </ul>
T2	For test operation	<ul style="list-style-type: none"> <li>■ Program verification: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>
AUT	For industrial robots without higher-level controllers	<ul style="list-style-type: none"> <li>■ Program mode: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC	<ul style="list-style-type: none"> <li>■ Program mode: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>

### 3.5.4 Operator safety

The operator safety signal is used for interlocking physical safeguards, e.g. safety gates. Automatic operation is not possible without this signal. In the event of a loss of signal during automatic operation (e.g. safety gate is opened), the manipulator stops with a safety stop 1.

Operator safety is not active in the test modes T1 (Manual Reduced Velocity) and T2 (Manual High Velocity).



**WARNING** Following a loss of signal, automatic operation must not be resumed merely by closing the safeguard; it must first additionally be acknowledged. It is the responsibility of the system integrator to ensure this. This is to prevent automatic operation from being resumed inadvertently while there are still persons in the danger zone, e.g. due to the safety gate closing accidentally.

- The acknowledgement must be designed in such a way that an actual check of the danger zone can be carried out first. Acknowledgement functions that do not allow this (e.g. because they are automatically triggered by closure of the safeguard) are not permissible.
- Failure to observe this may result in death to persons, severe physical injuries or considerable damage to property.

### 3.5.5 EMERGENCY STOP device

The EMERGENCY STOP device for the industrial robot is the EMERGENCY STOP button on the KCP. The button must be pressed in the event of a hazardous situation or emergency.

Reactions of the industrial robot if the EMERGENCY STOP button is pressed:

- The manipulator and any external axes (optional) are stopped with a safety stop 1.

Before operation can be resumed, the EMERGENCY STOP button must be turned to release it.



**WARNING** Tools and other equipment connected to the manipulator must be integrated into the EMERGENCY STOP circuit on the system side if they could constitute a potential hazard. Failure to observe this precaution may result in death, severe physical injuries or considerable damage to property.

There must always be at least one external EMERGENCY STOP device installed. This ensures that an EMERGENCY STOP device is available even when the KCP is disconnected.

(>>> 3.5.7 "External EMERGENCY STOP device" Page 25)

### 3.5.6 Logging off the higher-level safety controller

If the robot controller is connected to a higher-level safety controller, switching off the robot controller inevitably terminates this connection.

- If an SIB interface is used, this triggers an EMERGENCY STOP for the overall system.
- If the PROFIsafe interface is used, the KUKA safety controller generates a signal that prevents the higher-level controller from triggering an EMERGENCY STOP for the overall system.



**WARNING** If the PROFIsafe interface is used: In his risk assessment, the system integrator must take into consideration whether the fact that switching off the robot controller does not trigger an EMERGENCY STOP of the overall system could constitute a hazard and, if so, how this hazard can be countered. Failure to take this into consideration may result in death to persons, severe physical injuries or considerable damage to property.



**WARNING** If a robot controller is switched off, the E-STOP button on the KCP is no longer functional. The user is responsible for ensuring that the KCP is either covered or removed from the system. This serves to prevent operational and non-operational EMERGENCY STOP facilities from becoming interchanged.

Failure to observe this precaution may result in death to persons, severe physical injuries or considerable damage to property.

### 3.5.7 External EMERGENCY STOP device

There must be EMERGENCY STOP devices available at every operator station that can initiate a robot motion or other potentially hazardous situation. The system integrator is responsible for ensuring this.

There must always be at least one external EMERGENCY STOP device installed. This ensures that an EMERGENCY STOP device is available even when the KCP is disconnected.

External EMERGENCY STOP devices are connected via the customer interface. External EMERGENCY STOP devices are not included in the scope of supply of the industrial robot.

### 3.5.8 Enabling device

The enabling devices of the industrial robot are the enabling switches on the KCP.

There are 3 enabling switches installed on the KCP. The enabling switches have 3 positions:

- Not pressed
- Center position
- Panic position

In the test modes, the manipulator can only be moved if one of the enabling switches is held in the central position.

- Releasing the enabling switch triggers a safety stop 2.
- Pressing the enabling switch down fully (panic position) triggers a safety stop 1.
- It is possible, for a short time, to hold 2 enabling switches in the center position simultaneously. This makes it possible to adjust grip from one enabling switch to another one. If 2 enabling switches are held simultaneously in the center position for a longer period of time, this triggers a safety stop after several seconds.

If an enabling switch malfunctions (jams), the industrial robot can be stopped using the following methods:

- Press the enabling switch down fully
- Actuate the EMERGENCY STOP system
- Release the Start key



**WARNING** The enabling switches must not be held down by adhesive tape or other means or manipulated in any other way.

Death, serious physical injuries or major damage to property may result.

### 3.5.9 External enabling device

External enabling devices are required if it is necessary for more than one person to be in the danger zone of the industrial robot. They are connected to the robot controller via interface X11 or PROFIsafe.

External enabling devices are not included in the scope of supply of the industrial robot.

### 3.5.10 External safe operational stop

The safe operational stop can be triggered via an input on the customer interface. The state is maintained as long as the external signal is FALSE. If the external signal is TRUE, the manipulator can be moved again. No acknowledgement is required.

### 3.5.11 External safety stop 1 and external safety stop 2

Safety stop 1 and safety stop 2 can be triggered via an input on the customer interface. The state is maintained as long as the external signal is FALSE. If

the external signal is TRUE, the manipulator can be moved again. No acknowledgement is required.

### 3.5.12 Velocity monitoring in T1

The velocity at the TCP is monitored in T1 mode. If, due to an error, the velocity exceeds 250 mm/s, a safety stop 0 is triggered.

## 3.6 Additional protective equipment

### 3.6.1 Jog mode

In the operating modes T1 (Manual Reduced Velocity) and T2 (Manual High Velocity), the robot controller can only execute programs in jog mode. This means that it is necessary to hold down an enabling switch and the Start key in order to execute a program.

- Releasing the enabling switch triggers a safety stop 2.
- Pressing the enabling switch down fully (panic position) triggers a safety stop 1.
- Releasing the Start key triggers a STOP 2.

### 3.6.2 Software limit switches

The axis ranges of all manipulator and positioner axes are limited by means of adjustable software limit switches. These software limit switches only serve as machine protection and must be adjusted in such a way that the manipulator/positioner cannot hit the mechanical end stops.

The software limit switches are set during commissioning of an industrial robot.



Further information is contained in the operating and programming instructions.

### 3.6.3 Mechanical end stops

The axis ranges of main axes A1 to A3 and wrist axis A5 of the manipulator are limited by means of mechanical end stops with buffers.

Additional mechanical end stops can be installed on the external axes.

#### **NOTICE**

If the manipulator or an external axis hits an obstruction or a buffer on the mechanical end stop or axis range limitation, this can result in material damage to the industrial robot. KUKA Roboter GmbH must be consulted before the industrial robot is put back into operation. ([>>> 15 "KUKA Service" Page 375](#))

The affected buffer must be replaced with a new one before operation of the industrial robot is resumed. If a manipulator (or external axis) collides with a buffer at more than 250 mm/s, the manipulator (or external axis) must be exchanged or recommissioning must be carried out by KUKA Roboter GmbH.

### 3.6.4 Mechanical axis range limitation (optional)

Some manipulators can be fitted with mechanical axis range limitation in axes A1 to A3. The adjustable axis range limitation systems restrict the working

range to the required minimum. This increases personal safety and protection of the system.

In the case of manipulators that are not designed to be fitted with mechanical axis range limitation, the workspace must be laid out in such a way that there is no danger to persons or material property, even in the absence of mechanical axis range limitation.

If this is not possible, the workspace must be limited by means of photoelectric barriers, photoelectric curtains or obstacles on the system side. There must be no shearing or crushing hazards at the loading and transfer areas.



This option is not available for all robot models. Information on specific robot models can be obtained from KUKA Roboter GmbH.

### 3.6.5 Axis range monitoring (optional)

Some manipulators can be fitted with dual-channel axis range monitoring systems in main axes A1 to A3. The positioner axes may be fitted with additional axis range monitoring systems. The safety zone for an axis can be adjusted and monitored using an axis range monitoring system. This increases personal safety and protection of the system.



This option is not available for all robot models. Information on specific robot models can be obtained from KUKA Roboter GmbH.

### 3.6.6 Release device (optional)

#### Description

The release device can be used to move the manipulator manually after an accident or malfunction. The release device can be used for the main axis drive motors and, depending on the robot variant, also for the wrist axis drive motors. It is only for use in exceptional circumstances and emergencies (e.g. for freeing people).



#### CAUTION

The motors reach temperatures during operation which can cause burns to the skin. Contact must be avoided. Appropriate safety precautions must be taken, e.g. protective gloves must be worn.

#### Procedure

1. Switch off the robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it on again.
2. Remove the protective cap from the motor.
3. Push the release device onto the corresponding motor and move the axis in the desired direction.

The directions are indicated with arrows on the motors. It is necessary to overcome the resistance of the mechanical motor brake and any other loads acting on the axis.



#### WARNING

Moving an axis with the release device can damage the motor brake. This can result in personal injury and material damage. After using the release device, the affected motor must be exchanged.

### 3.6.7 Labeling on the industrial robot

All plates, labels, symbols and marks constitute safety-relevant parts of the industrial robot. They must not be modified or removed.

Labeling on the industrial robot consists of:

- Identification plates
- Warning labels
- Safety symbols
- Designation labels
- Cable markings
- Rating plates



Further information is contained in the technical data of the operating instructions or assembly instructions of the components of the industrial robot.

### 3.6.8 External safeguards

The access of persons to the danger zone of the industrial robot must be prevented by means of safeguards. It is the responsibility of the system integrator to ensure this.

Physical safeguards must meet the following requirements:

- They meet the requirements of EN 953.
- They prevent access of persons to the danger zone and cannot be easily circumvented.
- They are sufficiently fastened and can withstand all forces that are likely to occur in the course of operation, whether from inside or outside the enclosure.
- They do not, themselves, represent a hazard or potential hazard.
- The prescribed minimum clearance from the danger zone is maintained.

Safety gates (maintenance gates) must meet the following requirements:

- They are reduced to an absolute minimum.
- The interlocks (e.g. safety gate switches) are linked to the operator safety input of the robot controller via safety gate switching devices or safety PLC.
- Switching devices, switches and the type of switching conform to the requirements of Performance Level d and category 3 according to EN ISO 13849-1.
- Depending on the risk situation: the safety gate is additionally safeguarded by means of a locking mechanism that only allows the gate to be opened if the manipulator is safely at a standstill.
- The button for acknowledging the safety gate is located outside the space limited by the safeguards.



Further information is contained in the corresponding standards and regulations. These also include EN 953.

#### Other safety equipment

Other safety equipment must be integrated into the system in accordance with the corresponding standards and regulations.

### 3.7 Overview of operating modes and safety functions

The following table indicates the operating modes in which the safety functions are active.

Safety functions	T1	T2	AUT	AUT EXT
Operator safety	-	-	active	active
EMERGENCY STOP device	active	active	active	active
Enabling device	active	active	-	-
Reduced velocity during program verification	active	-	-	-
Jog mode	active	active	-	-
Software limit switches	active	active	active	active

### 3.8 Safety measures

#### 3.8.1 General safety measures

The industrial robot may only be used in perfect technical condition in accordance with its intended use and only by safety-conscious persons. Operator errors can result in personal injury and damage to property.

It is important to be prepared for possible movements of the industrial robot even after the robot controller has been switched off and locked. Incorrect installation (e.g. overload) or mechanical defects (e.g. brake defect) can cause the manipulator or external axes to sag. If work is to be carried out on a switched-off industrial robot, the manipulator and external axes must first be moved into a position in which they are unable to move on their own, whether the payload is mounted or not. If this is not possible, the manipulator and external axes must be secured by appropriate means.



**DANGER** In the absence of operational safety functions and safeguards, the industrial robot can cause personal injury or material damage. If safety functions or safeguards are dismantled or deactivated, the industrial robot may not be operated.



**WARNING** Standing underneath the robot arm can cause death or serious physical injuries. For this reason, standing underneath the robot arm is prohibited!

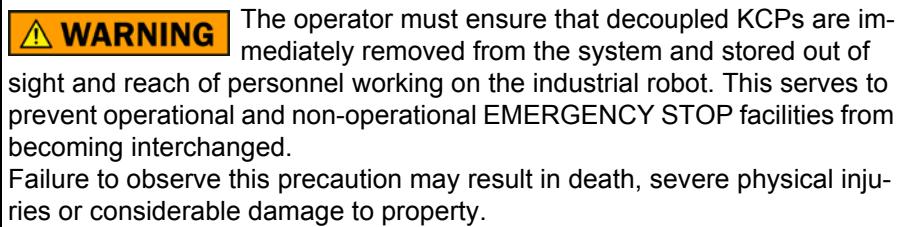


**CAUTION** The motors reach temperatures during operation which can cause burns to the skin. Contact must be avoided. Appropriate safety precautions must be taken, e.g. protective gloves must be worn.

#### KCP

The user must ensure that the industrial robot is only operated with the KCP by authorized persons.

If more than one KCP is used in the overall system, it must be ensured that each KCP is unambiguously assigned to the corresponding industrial robot. They must not be interchanged.



<b>External keyboard, external mouse</b>	<p>An external keyboard and/or external mouse may only be used if the following conditions are met:</p> <ul style="list-style-type: none"> <li>■ Start-up or maintenance work is being carried out.</li> <li>■ The drives are switched off.</li> <li>■ There are no persons in the danger zone.</li> </ul> <p>The KCP must not be used as long as an external keyboard and/or external mouse are connected.</p> <p>The external keyboard and/or external mouse must be removed as soon as the start-up or maintenance work is completed or the KCP is connected.</p>
<b>Faults</b>	<p>The following tasks must be carried out in the case of faults in the industrial robot:</p> <ul style="list-style-type: none"> <li>■ Switch off the robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it on again.</li> <li>■ Indicate the fault by means of a label with a corresponding warning (tag-out).</li> <li>■ Keep a record of the faults.</li> <li>■ Eliminate the fault and carry out a function test.</li> </ul>
<b>Modifications</b>	<p>After modifications to the industrial robot, checks must be carried out to ensure the required safety level. The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.</p> <p>New or modified programs must always be tested first in Manual Reduced Velocity mode (T1).</p> <p>After modifications to the industrial robot, existing programs must always be tested first in Manual Reduced Velocity mode (T1). This applies to all components of the industrial robot and includes modifications to the software and configuration settings.</p>

### 3.8.2 Transportation

<b>Manipulator</b>	The prescribed transport position of the manipulator must be observed. Transportation must be carried out in accordance with the operating instructions or assembly instructions of the manipulator.
<b>Robot controller</b>	<p>The robot controller must be transported and installed in an upright position. Avoid vibrations and impacts during transportation in order to prevent damage to the robot controller.</p> <p>Transportation must be carried out in accordance with the operating instructions or assembly instructions of the robot controller.</p>
<b>External axis (optional)</b>	The prescribed transport position of the external axis (e.g. KUKA linear unit, turn-tilt table, etc.) must be observed. Transportation must be carried out in accordance with the operating instructions or assembly instructions of the external axis.

### 3.8.3 Start-up and recommissioning

Before starting up systems and devices for the first time, a check must be carried out to ensure that the systems and devices are complete and operational, that they can be operated safely and that any damage is detected.

The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.



The passwords for logging onto the KUKA System Software as "Expert" and "Administrator" must be changed before start-up and must only be communicated to authorized personnel.



**DANGER** The robot controller is preconfigured for the specific industrial robot. If cables are interchanged, the manipulator and the external axes (optional) may receive incorrect data and can thus cause personal injury or material damage. If a system consists of more than one manipulator, always connect the connecting cables to the manipulators and their corresponding robot controllers.



If additional components (e.g. cables), which are not part of the scope of supply of KUKA Roboter GmbH, are integrated into the industrial robot, the user is responsible for ensuring that these components do not adversely affect or disable safety functions.



**NOTICE** If the internal cabinet temperature of the robot controller differs greatly from the ambient temperature, condensation can form, which may cause damage to the electrical components. Do not put the robot controller into operation until the internal temperature of the cabinet has adjusted to the ambient temperature.

#### Function test

The following tests must be carried out before start-up and recommissioning:

##### General test:

It must be ensured that:

- The industrial robot is correctly installed and fastened in accordance with the specifications in the documentation.
- There are no foreign bodies or loose parts on the industrial robot.
- All required safety equipment is correctly installed and operational.
- The power supply ratings of the industrial robot correspond to the local supply voltage and mains type.
- The ground conductor and the equipotential bonding cable are sufficiently rated and correctly connected.
- The connecting cables are correctly connected and the connectors are locked.

##### Test of the safety functions:

A function test must be carried out for the following safety functions to ensure that they are functioning correctly:

- Local EMERGENCY STOP device (= EMERGENCY STOP button on the KCP)
- External EMERGENCY STOP device (input and output)
- Enabling device (in the test modes)
- Operator safety
- All other safety-relevant inputs and outputs used
- Other external safety functions

### Test of reduced velocity control:

This test is to be carried out as follows:

1. Program a straight path with the maximum possible velocity.
2. Calculate the length of the path.
3. Execute the path in T1 mode with the override set to 100% and time the motion with a stopwatch.



**WARNING** It must be ensured that no persons are present within the danger zone during path execution. Death or severe physical injuries may result.

4. Calculate the velocity from the length of the path and the time measured for execution of the motion.

Control of reduced velocity is functioning correctly if the following results are achieved:

- The calculated velocity does not exceed 250 mm/s.
- The manipulator executes the path as programmed (i.e. in a straight line, without deviations).

#### Machine data

It must be ensured that the rating plate on the robot controller has the same machine data as those entered in the declaration of incorporation. The machine data on the rating plate of the manipulator and the external axes (optional) must be entered during start-up.



**DANGER** The industrial robot must not be moved if incorrect machine data are loaded. Death, severe physical injuries or considerable damage to property may otherwise result. The correct machine data must be loaded.

Following modifications to the machine data, the safety configuration must be checked.



Further information is contained in the Operating and Programming Instructions for System Integrators.

Following modifications to the machine data, control of the reduced velocity must be checked.

#### 3.8.3.1 Start-up mode

##### Description

The industrial robot can be set to Start-up mode via the smartHMI user interface. In this mode, the manipulator can be moved in T1 or CRR mode in the absence of the safety periphery. (CRR is an operating mode specifically for use with SafeOperation.)

- If an SIB interface is used:

Start-up mode is always possible if all input signals have the state "logic zero". If this is not the case, the robot controller prevents or terminates Start-up mode.

- If the PROFIsafe interface is used:

If a connection to a higher-level safety system exists or is established, the robot controller prevents or terminates Start-up mode.

##### Hazards

Possible hazards and risks involved in using Start-up mode:

- A person walks into the manipulator's danger zone.
- An unauthorized person moves the manipulator.

- In a hazardous situation, a disabled external EMERGENCY STOP device is actuated and the manipulator is not shut down.

Additional measures for avoiding risks in Start-up mode:

- Cover disabled EMERGENCY STOP devices or attach a warning sign indicating that the EMERGENCY STOP device is out of operation.
- If there is no safety fence, other measures must be taken to prevent persons from entering the manipulator's danger zone, e.g. use of warning tape.
- Use of Start-up mode must be minimized – and avoided where possible – by means of organizational measures.

## Use

Intended use of Start-up mode:

- Only service personnel who have received safety instruction may use Start-up mode.
- Start-up in T1 mode or CRR mode when the external safeguards have not yet been installed or put into operation. The danger zone must be delimited at least by means of warning tape.
- Fault localization (periphery fault).



Use of Start-up mode disables all external safeguards. The service personnel are responsible for ensuring that there is no-one in or near the danger zone of the manipulator as long as the safeguards are disabled. Failure to observe this may result in death to persons, physical injuries or damage to property.

## Misuse

Any use or application deviating from the designated use is deemed to be impermissible misuse. This includes, for example, use by any other personnel.

KUKA Roboter GmbH accepts no liability for damage or injury caused thereby. The risk lies entirely with the user.

### 3.8.4 Manual mode

Manual mode is the mode for setup work. Setup work is all the tasks that have to be carried out on the industrial robot to enable automatic operation. Setup work includes:

- Jog mode
- Teach
- Programming
- Program verification

The following must be taken into consideration in manual mode:

- If the drives are not required, they must be switched off to prevent the manipulator or the external axes (optional) from being moved unintentionally. New or modified programs must always be tested first in Manual Reduced Velocity mode (T1).
- The manipulator, tooling or external axes (optional) must never touch or project beyond the safety fence.
- Workpieces, tooling and other objects must not become jammed as a result of the industrial robot motion, nor must they lead to short-circuits or be liable to fall off.
- All setup work must be carried out, where possible, from outside the safeguarded area.

If the setup work has to be carried out inside the safeguarded area, the following must be taken into consideration:

**In Manual Reduced Velocity mode (T1):**

- If it can be avoided, there must be no other persons inside the safeguarded area.

If it is necessary for there to be several persons inside the safeguarded area, the following must be observed:

- Each person must have an enabling device.
- All persons must have an unimpeded view of the industrial robot.
- Eye-contact between all persons must be possible at all times.
- The operator must be so positioned that he can see into the danger area and get out of harm's way.

**In Manual High Velocity mode (T2):**

- This mode may only be used if the application requires a test at a velocity higher than Manual Reduced Velocity.
- Teaching and programming are not permissible in this operating mode.
- Before commencing the test, the operator must ensure that the enabling devices are operational.
- The operator must be positioned outside the danger zone.
- There must be no other persons inside the safeguarded area. It is the responsibility of the operator to ensure this.

### 3.8.5 Simulation

Simulation programs do not correspond exactly to reality. Robot programs created in simulation programs must be tested in the system in **Manual Reduced Velocity mode (T1)**. It may be necessary to modify the program.

### 3.8.6 Automatic mode

Automatic mode is only permissible in compliance with the following safety measures:

- All safety equipment and safeguards are present and operational.
- There are no persons in the system.
- The defined working procedures are adhered to.

If the manipulator or an external axis (optional) comes to a standstill for no apparent reason, the danger zone must not be entered until an EMERGENCY STOP has been triggered.

### 3.8.7 Maintenance and repair

After maintenance and repair work, checks must be carried out to ensure the required safety level. The valid national or regional work safety regulations must be observed for this check. The correct functioning of all safety circuits must also be tested.

The purpose of maintenance and repair work is to ensure that the system is kept operational or, in the event of a fault, to return the system to an operational state. Repair work includes troubleshooting in addition to the actual repair itself.

The following safety measures must be carried out when working on the industrial robot:

- Carry out work outside the danger zone. If work inside the danger zone is necessary, the user must define additional safety measures to ensure the safe protection of personnel.
- Switch off the industrial robot and secure it (e.g. with a padlock) to prevent it from being switched on again. If it is necessary to carry out work with the robot controller switched on, the user must define additional safety measures to ensure the safe protection of personnel.
- If it is necessary to carry out work with the robot controller switched on, this may only be done in operating mode T1.
- Label the system with a sign indicating that work is in progress. This sign must remain in place, even during temporary interruptions to the work.
- The EMERGENCY STOP systems must remain active. If safety functions or safeguards are deactivated during maintenance or repair work, they must be reactivated immediately after the work is completed.

**⚠ WARNING**

Before work is commenced on live parts of the robot system, the main switch must be turned off and secured against being switched on again. The system must then be checked to ensure that it is deenergized.

It is not sufficient, before commencing work on live parts, to execute an EMERGENCY STOP or a safety stop, or to switch off the drives, as this does not disconnect the robot system from the mains power supply in the case of the drives of the new generation. Parts remain energized. Death or severe physical injuries may result.

Faulty components must be replaced using new components with the same article numbers or equivalent components approved by KUKA Roboter GmbH for this purpose.

Cleaning and preventive maintenance work is to be carried out in accordance with the operating instructions.

**Robot controller**

Even when the robot controller is switched off, parts connected to peripheral devices may still carry voltage. The external power sources must therefore be switched off if work is to be carried out on the robot controller.

The ESD regulations must be adhered to when working on components in the robot controller.

Voltages in excess of 50 V (up to 780 V) can be present in various components for several minutes after the robot controller has been switched off! To prevent life-threatening injuries, no work may be carried out on the industrial robot in this time.

Water and dust must be prevented from entering the robot controller.

**Counterbalancing system**

Some robot variants are equipped with a hydropneumatic, spring or gas cylinder counterbalancing system.

The hydropneumatic and gas cylinder counterbalancing systems are pressure equipment and, as such, are subject to obligatory equipment monitoring. Depending on the robot variant, the counterbalancing systems correspond to category 0, II or III, fluid group 2, of the Pressure Equipment Directive.

The user must comply with the applicable national laws, regulations and standards pertaining to pressure equipment.

Inspection intervals in Germany in accordance with Industrial Safety Order, Sections 14 and 15. Inspection by the user before commissioning at the installation site.

The following safety measures must be carried out when working on the counterbalancing system:

- The manipulator assemblies supported by the counterbalancing systems must be secured.
- Work on the counterbalancing systems must only be carried out by qualified personnel.

## Hazardous substances

The following safety measures must be carried out when handling hazardous substances:

- Avoid prolonged and repeated intensive contact with the skin.
- Avoid breathing in oil spray or vapors.
- Clean skin and apply skin cream.



To ensure safe use of our products, we recommend that our customers regularly request up-to-date safety data sheets from the manufacturers of hazardous substances.

## 3.8.8 Decommissioning, storage and disposal

The industrial robot must be decommissioned, stored and disposed of in accordance with the applicable national laws, regulations and standards.

## 3.8.9 Safety measures for “single point of control”

### Overview

If certain components in the industrial robot are operated, safety measures must be taken to ensure complete implementation of the principle of “single point of control” (SPOC).

Components:

- Submit interpreter
- PLC
- OPC Server
- Remote control tools
- Tools for configuration of bus systems with online functionality
- KUKA.RobotSensorInterface
- External keyboard/mouse



The implementation of additional safety measures may be required. This must be clarified for each specific application; this is the responsibility of the system integrator, programmer or user of the system.

Since only the system integrator knows the safe states of actuators in the periphery of the robot controller, it is his task to set these actuators to a safe state, e.g. in the event of an EMERGENCY STOP.

### T1, T2

In the test modes, the components referred to above (with the exception of the external keyboard/mouse) may only access the industrial robot if the following signals have the following states:

Signal	State required for SPOC
\$USER_SAF	TRUE
\$SPOC_MOTION_ENABLE	TRUE

### Submit interpreter, PLC

If motions, (e.g. drives or grippers) are controlled with the Submit interpreter or the PLC via the I/O system, and if they are not safeguarded by other means, then this control will take effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

If variables that affect the robot motion (e.g. override) are modified with the Submit interpreter or the PLC, this takes effect even in T1 and T2 modes or while an EMERGENCY STOP is active.

Safety measures:

- In the test modes, the system variable \$OV\_PRO must not be written to by the Submit interpreter or the PLC.
- Do not modify safety-relevant signals and variables (e.g. operating mode, EMERGENCY STOP, safety gate contact) via the Submit interpreter or PLC.

If modifications are nonetheless required, all safety-relevant signals and variables must be linked in such a way that they cannot be set to a dangerous state by the Submit interpreter or PLC.

**OPC server,  
remote control  
tools**

These components can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- KUKA stipulates that these components are to be used exclusively for diagnosis and visualization.  
Programs, outputs or other parameters of the robot controller must not be modified using these components.
- If these components are used, outputs that could cause a hazard must be determined in a risk assessment. These outputs must be designed in such a way that they cannot be set without being enabled. This can be done using an external enabling device, for example.

**Tools for configu-  
ration of bus  
systems**

If these components have an online functionality, they can be used with write access to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

- WorkVisual from KUKA
- Tools from other manufacturers

Safety measures:

- In the test modes, programs, outputs or other parameters of the robot controller must not be modified using these components.

**External  
keyboard/mouse**

These components can be used to modify programs, outputs or other parameters of the robot controller, without this being noticed by any persons located inside the system.

Safety measures:

- Only use one operator console at each robot controller.
- If the KCP is being used for work inside the system, remove any keyboard and mouse from the robot controller beforehand.

### 3.9 Applied norms and regulations

Name	Definition	Edition
<b>2006/42/EC</b>	Machinery Directive: Directive 2006/42/EC of the European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC (recast)	2006
<b>2004/108/EC</b>	EMC Directive: Directive 2004/108/EC of the European Parliament and of the Council of 15 December 2004 on the approximation of the laws of the Member States relating to electromagnetic compatibility and repealing Directive 89/336/EEC	2004
<b>97/23/EC</b>	Pressure Equipment Directive: Directive 97/23/EC of the European Parliament and of the Council of 29 May 1997 on the approximation of the laws of the Member States concerning pressure equipment (Only applicable for robots with hydropneumatic counter-balancing system.)	1997
<b>EN ISO 13850</b>	Safety of machinery: Emergency stop - Principles for design	2008
<b>EN ISO 13849-1</b>	Safety of machinery: Safety-related parts of control systems - Part 1: General principles of design	2008
<b>EN ISO 13849-2</b>	Safety of machinery: Safety-related parts of control systems - Part 2: Validation	2008
<b>EN ISO 12100-1</b>	Safety of machinery: Basic concepts, general principles for design - Part 1: Basic terminology, methodology	2003
<b>EN ISO 12100-2</b>	Safety of machinery: Basic concepts, general principles for design - Part 2: Technical principles	2003
<b>EN ISO 10218-1</b>	Industrial robots: Safety	2008
<b>EN 614-1</b>	Safety of machinery: Ergonomic design principles - Part 1: Terms and general principles	2006
<b>EN 61000-6-2</b>	Electromagnetic compatibility (EMC): Part 6-2: Generic standards; Immunity for industrial environments	2005
<b>EN 61000-6-4</b>	Electromagnetic compatibility (EMC): Part 6-4: Generic standards; Emission standard for industrial environments	2007
<b>EN 60204-1</b>	Safety of machinery: Electrical equipment of machines - Part 1: General requirements	2006



## 4 Operation

### 4.1 KUKA smartPAD teach pendant

#### 4.1.1 Front view

##### Function

The smartPAD is the teach pendant for the industrial robot. The smartPAD has all the operator control and display functions required for operating and programming the industrial robot.

The smartPAD has a touch screen: the smartHMI can be operated with a finger or stylus. An external mouse or external keyboard is not necessary.



The general term “KCP” (KUKA Control Panel) is often used in this documentation for the smartPAD.

##### Overview

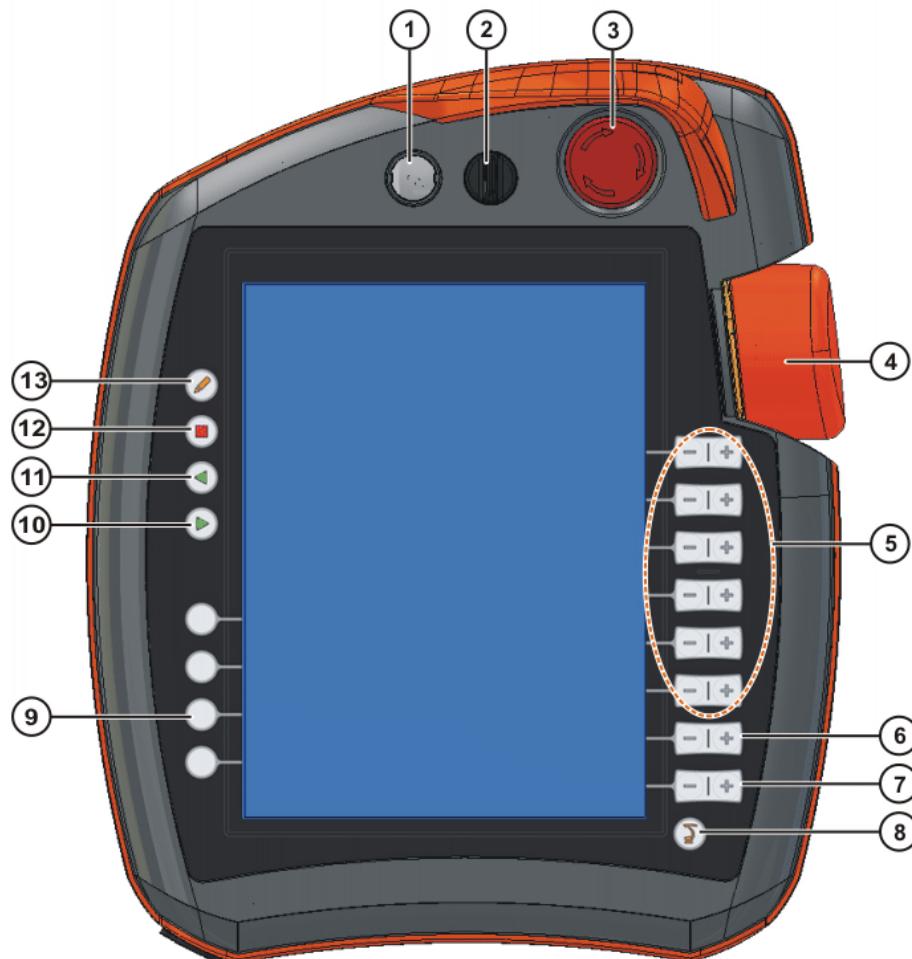


Fig. 4-1: KUKA smartPAD, front view

Item	Description
1	Button for disconnecting the smartPAD (>>> 4.1.3 "Disconnecting and connecting the smartPAD" Page 44)
2	Keypad for calling the connection manager. The switch can only be turned if the key is inserted. The connection manager is used to change the operating mode. (>>> 4.10 "Changing operating mode" Page 53)
3	EMERGENCY STOP button. Stops the robot in hazardous situations. The EMERGENCY STOP button locks itself in place when it is pressed.
4	Space Mouse. For moving the robot manually. (>>> 4.12 "Jogging the robot" Page 55)
5	Jog keys. For moving the robot manually. (>>> 4.12 "Jogging the robot" Page 55)
6	Key for setting the program override
7	Key for setting the jog override
8	Main menu key. Shows the menu items on the smartHMI. (>>> 4.4 "Calling the main menu" Page 48)
9	Status keys. The status keys are used primarily for setting parameters in technology packages. Their exact function depends on the technology packages installed.
10	Start key. The Start key is used to start a program.
11	Start backwards key. The Start backwards key is used to start a program backwards. The program is executed step by step.
12	STOP key. The STOP key is used to stop a program that is running.
13	Keyboard key Displays the keyboard. It is generally not necessary to press this key to display the keyboard, as the smartHMI detects when keyboard input is required and displays the keyboard automatically. (>>> 4.2.2 "Keypad" Page 47)

#### 4.1.2 Rear view

##### Overview



**Fig. 4-2: KUKA smartPAD, rear view**

- |   |                   |   |                      |
|---|-------------------|---|----------------------|
| 1 | Enabling switch   | 4 | USB connection       |
| 2 | Start key (green) | 5 | Enabling switch      |
| 3 | Enabling switch   | 6 | Identification plate |

##### Description

Element	Description
<b>Identification plate</b>	Identification plate
<b>Start key</b>	The Start key is used to start a program.
<b>Enabling switch</b>	<p>The enabling switch has 3 positions:</p> <ul style="list-style-type: none"> <li>■ Not pressed</li> <li>■ Center position</li> <li>■ Panic position</li> </ul> <p>The enabling switch must be held in the center position in operating modes T1 and T2 in order to be able to jog the manipulator.</p> <p>In the operating modes Automatic and Automatic External, the enabling switch has no function.</p>
<b>USB connection</b>	<p>The USB connection is used, for example, for archiving and restoring data.</p> <p>Only for FAT32-formatted USB sticks.</p>

### 4.1.3 Disconnecting and connecting the smartPAD

#### Description

The smartPAD can be disconnected while the robot controller is running.



If the smartPAD is disconnected, the system can no longer be switched off by means of the EMERGENCY STOP button on the smartPAD. For this reason, an external EMERGENCY STOP must be connected to the robot controller. The operator must ensure that disconnected smartPADs are immediately removed from the system and stored out of sight and reach of personnel working on the industrial robot. This serves to prevent operational and non-operational EMERGENCY STOP facilities from becoming interchanged. Failure to observe these precautions may result in death to persons, severe physical injuries or considerable damage to property.

#### Procedure

##### Disconnection:

1. Press the disconnect button on the smartPAD.

A message and a counter are displayed on the smartHMI. The counter runs for 30 s. During this time, the smartPAD can be disconnected from the robot controller.



If the smartPAD is disconnected without the counter running, this triggers an EMERGENCY STOP. The EMERGENCY STOP can only be canceled by plugging the smartPAD back in.

2. Disconnect the smartPAD from the robot controller.

If the counter expires without the smartPAD having been disconnected, this has no effect. The disconnect button can be pressed again at any time to display the counter again.

##### Connection:

- Connect the smartPAD to the robot controller.

A smartPAD can be connected at any time. Precondition: Same smartPAD variant as the disconnected device. The EMERGENCY STOP and enabling switches are operational again 30 s after connection. The smartHMI is automatically displayed again. (This may take longer than 30 s.)

The connected smartPAD assumes the current operating mode of the robot controller.



The current operating mode is not, in all cases, the same as that before the smartPAD was disconnected: if the robot controller is part of a RoboTeam, the operating mode may have been changed after disconnection, e.g. by the master.



The user connecting a smartPAD to the robot controller must subsequently stay with the smartPAD for at least 30 s, i.e. until the EMERGENCY STOP and enabling switches are operational once again. This prevents another user from trying to activate a non-operational EMERGENCY STOP in an emergency situation, for example. Failure to observe this may result in death to persons, physical injuries or damage to property.

## 4.2 KUKA smartHMI user interface

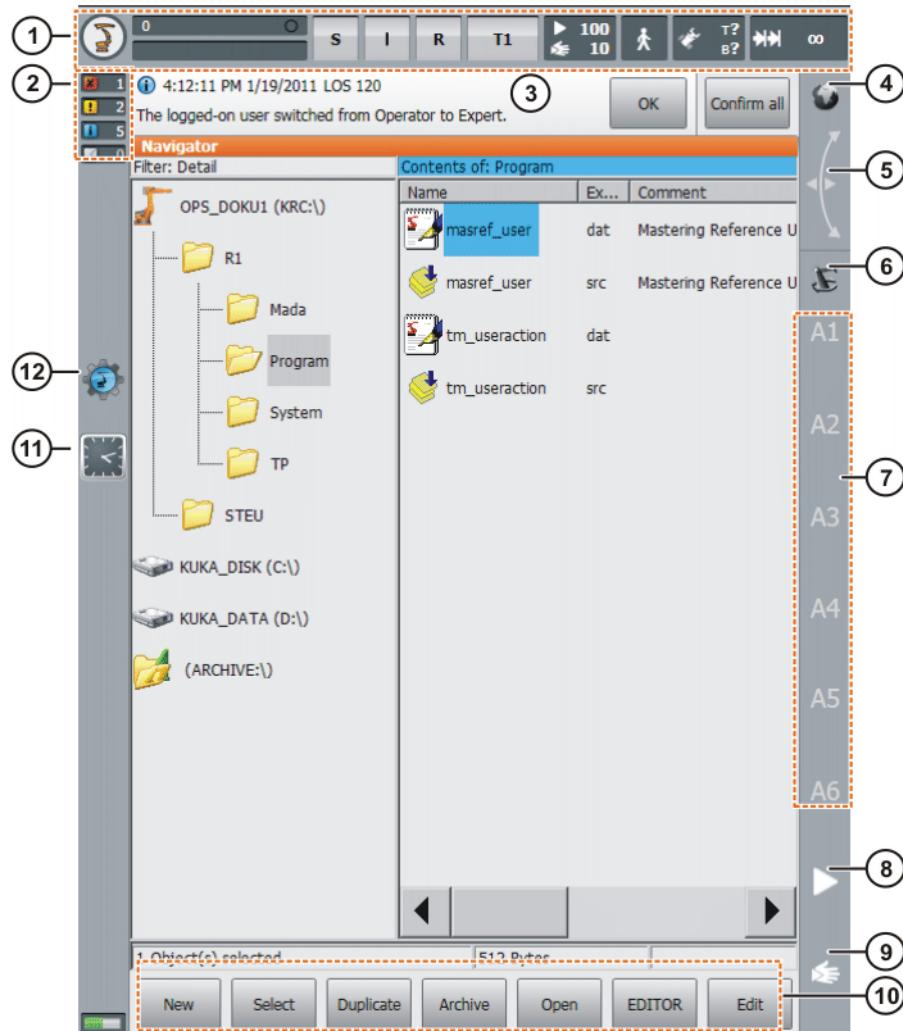


Fig. 4-3: KUKA smartHMI user interface

Item	Description
1	Status bar (=>> 4.2.1 "Status bar" Page 46)
2	Message counter The message counter indicates how many messages of each message type are active. Touching the message counter enlarges the display.
3	Message window By default, only the last message is displayed. Touching the message window expands it so that all active messages are displayed. An acknowledgeable message can be acknowledged with <b>OK</b> . All acknowledgeable messages can be acknowledged at once with <b>All OK</b> .
4	<b>Space Mouse</b> status indicator This indicator shows the current coordinate system for jogging with the Space Mouse. Touching the indicator displays all coordinate systems, allowing a different one to be selected.

Item	Description
5	<b>Space Mouse alignment</b> indicator Touching this indicator opens a window in which the current alignment of the Space Mouse is indicated and can be changed. (>>> 4.12.8 "Defining the alignment of the Space Mouse" Page 63)
6	<b>Jog keys</b> status indicator This indicator shows the current coordinate system for jogging with the jog keys. Touching the indicator displays all coordinate systems, allowing a different one to be selected.
7	Jog key labels If axis-specific jogging is selected, the axis numbers are displayed here (A1, A2, etc.). If Cartesian jogging is selected, the coordinate system axes are displayed here (X, Y, Z, A, B, C). Touching the label causes the selected kinematics group to be displayed.
8	Program override (>>> 7.5.4 "Setting the program override (POV)" Page 209)
9	Jog override (>>> 4.12.3 "Setting the jog override (HOV)" Page 60)
10	Button bar. The buttons change dynamically and always refer to the window that is currently active in the smartHMI. At the right-hand end is the <b>Edit</b> button. This can be used to call numerous commands relating to the Navigator.
11	Clock The clock displays the system time. Touching the clock displays the system time in digital format, together with the current date.
12	WorkVisual icon If no project can be opened, the icon has a small red X in the bottom left-hand corner. This is the case, for example, if the files belonging to the project are missing. In such a case, the system is only partially available, e.g. the safety configuration cannot be opened.

#### 4.2.1 Status bar

The status bar indicates the status of certain central settings of the industrial robot. In most cases, touching the display opens a window in which the settings can be modified.

##### Overview

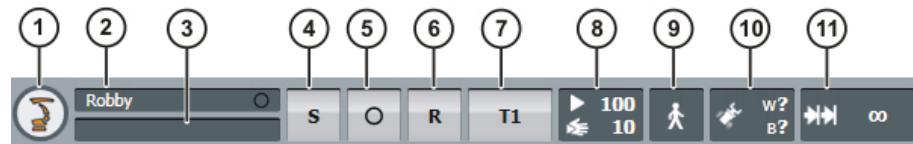


Fig. 4-4: KUKA smartHMI status bar

Item	Description
1	Main menu key. Shows the menu items on the smartHMI. (>>> 4.4 "Calling the main menu" Page 48)
2	Robot name. The robot name can be changed. (>>> 4.15.14 "Displaying/editing robot data" Page 78)
3	If a program has been selected, the name is displayed here.
4	<b>Submit interpreter</b> status indicator (>>> 11 "Submit interpreter" Page 351)
5	<b>Drives</b> status indicator. The drives can be switched on or off here. (>>> 7.5.5 "Switching drives on/off" Page 209)
6	<b>Robot interpreter</b> status indicator. Programs can be reset or canceled here. (>>> 7.5.6 "Robot interpreter status indicator" Page 209) (>>> 7.2.1 "Selecting and deselecting a program" Page 202) (>>> 7.5.11 "Resetting a program" Page 211)
7	Current operating mode (>>> 4.10 "Changing operating mode" Page 53)
8	<b>POV/HOV</b> status indicator. Indicates the current program override and the current jog override. (>>> 7.5.4 "Setting the program override (POV)" Page 209) (>>> 4.12.3 "Setting the jog override (HOV)" Page 60)
9	<b>Program run mode</b> status indicator. Indicates the current program run mode. (>>> 7.5.2 "Program run modes" Page 208)
10	<b>Tool/base</b> status indicator. Indicates the current tool and base. (>>> 4.12.4 "Selecting the tool and base" Page 60)
11	<b>Incremental jogging</b> status indicator. (>>> 4.12.10 "Incremental jogging" Page 65)

#### 4.2.2 Keypad

The smartPAD has a touch screen: the smartHMI can be operated with a finger or stylus.

There is a keypad on the smartHMI for entering letters and numbers. The smartHMI detects when the entry of letters or numbers is required and automatically displays the keypad.

The keypad only ever displays the characters that are required. If, for example, a box is edited in which only numbers can be entered, then only numbers are displayed and not letters.



**Fig. 4-5: Example keypad**

#### 4.2.3 Minimizing KUKA smartHMI

- |                     |   |
|---------------------|---|
| <b>Precondition</b> | <ul style="list-style-type: none"> <li>■ User group "Expert".</li> <li>■ Operating mode T1 or T2.</li> </ul>                |
| <b>Procedure</b>    | <ul style="list-style-type: none"> <li>■ Select <b>Start-up &gt; Service &gt; Minimize HMI</b> in the main menu.</li> </ul> |

### 4.3 Switching on the robot controller and starting the KSS

- |                  |   |
|------------------|---|
| <b>Procedure</b> | <ul style="list-style-type: none"> <li>■ Turn the main switch on the robot controller to ON.<br/>The operating system and the KSS start automatically.</li> </ul> |
|                  | <p>If the KSS does not start automatically, e.g. because the Startup function has been disabled, execute the file StartKRC.exe in the directory C:\KRC.</p>       |
|                  | <p>If the robot controller is logged onto the network, the start may take longer.</p>   |

### 4.4 Calling the main menu

- |                    |   |
|--------------------|---|
| <b>Procedure</b>   | <ul style="list-style-type: none"> <li>■ Press "Main menu" key on the KCP. The <b>Main menu</b> window is opened. The display is always the same as that which was in the window before it was last closed.</li> </ul>  |
| <b>Description</b> | <p>Properties of the <b>Main Menu</b> window:</p> <ul style="list-style-type: none"> <li>■ The main menu is displayed in the left-hand column.</li> <li>■ Touching a menu item that contains an arrow opens the corresponding submenu (e.g. <b>Configure</b>). Depending on how many nested submenus are open, the <b>Main Menu</b> column may no longer be visible, with only the submenus remaining visible.</li> <li>■ The arrow key in the top right-hand corner closes the most recently opened submenu.</li> <li>■ The Home key in the top right-hand corner closes all open submenus.</li> <li>■ The most recently selected menu items are displayed in the bottom section (maximum 6). This makes it possible to select these menu items again directly without first having to close other submenus that might be open.</li> <li>■ The white cross on the left-hand side closes the window.</li> </ul> |

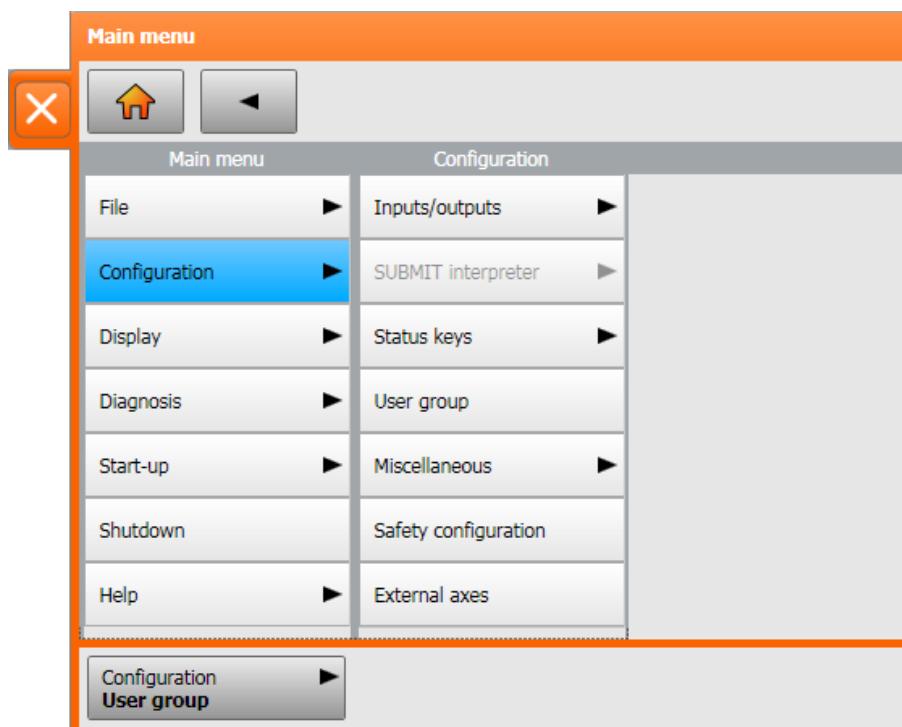


Fig. 4-6: Example: Configuration submenu is open.

## 4.5 Defining the start type for KSS

This function defines how the KSS starts after a power failure. A power failure and start are generally triggered by switching the main switch on the robot controller off and on.

By default, the KSS boots with Hibernate.

### Precondition

- Expert user group

### Procedure

1. In the main menu, select **Shutdown**. A window opens.
2. Select the start type: **Cold start** or **Hibernate**.  
(>>> "Start types" Page 52)
3. Close the window. The selected start type is applied.



Following installation or update of the KUKA System Software, the robot controller always performs an initial cold start.

## 4.6 Exiting or restarting KSS

### Precondition

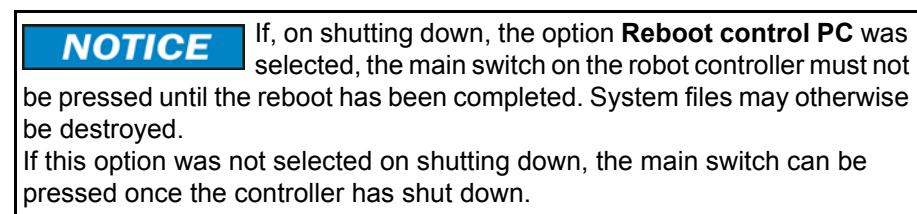
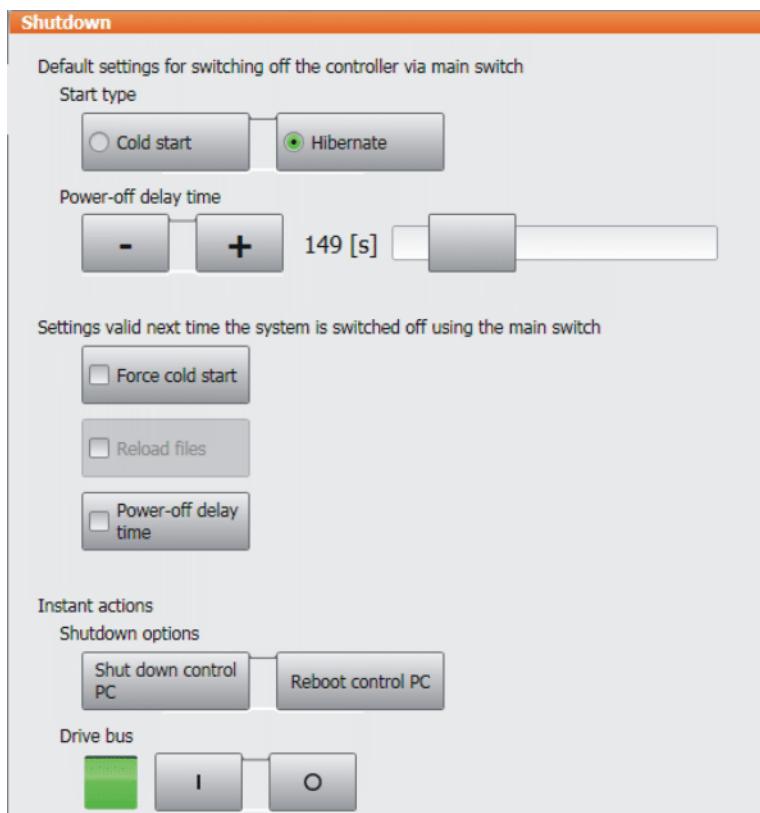
- For certain options: "Expert" user group

### Procedure

1. Select the menu item **Shutdown** in the main menu.
2. Select the desired options.
3. Press **Shut down control PC** or **Reboot control PC**.
4. Confirm the request for confirmation with **Yes**. The KSS is terminated and restarted in accordance with the selected option.

After the restart, the following message is displayed:

- **Cold start of controller**
- Or, if **Reload files** has been selected: **Initial cold start of controller**

**Description****Fig. 4-7: Shutdown window**

Option	Description
<b>Default settings for switching off the controller via main switch</b>	
<b>Cold start</b>	After a power failure, the robot controller starts with a cold start. (A power failure and start are generally triggered by switching the main switch on the robot controller off and on.) Can only be selected in the "Expert" user group. (>>> "Start types" Page 52)
<b>Hibernate</b>	After a power failure, the robot controller starts with Hibernate. Can only be selected in the "Expert" user group. (>>> "Start types" Page 52)
<b>Power-off delay time</b>	Wait time before the robot controller is shut down. This ensures that the system does not immediately shut down, for example, in the event of very brief power failures, but bridges the power failure. Can only be changed in the "Expert" user group.

Option	Description
<b>Settings valid next time the system is switched off using the main switch</b>	
These settings only apply to the next start.	
<b>Force cold start</b>	Activated: The next start is a cold start. Only available if <b>Hibernate</b> has been selected.
<b>Reload files</b>	Activated: The next start is an initial cold start.  This option must be selected in the following cases: <ul style="list-style-type: none"> <li>■ If XML files have been changed directly, i.e. the user has opened the file and modified it. (Any other changes to XML files, e.g. if the robot controller modifies them in the background, are irrelevant.)</li> <li>■ If hardware components are to be exchanged after shutdown.</li> </ul> Can only be selected in the "Expert" user group. Only available if <b>Cold start</b> or <b>Force cold start</b> has been selected.  Depending on the hardware, the initial cold start takes approx. 30 to 150 seconds longer than a normal cold start.
<b>Power-off delay time</b>	Activated: The wait time is adhered to the next time the system is shut down.  Deactivated: The wait time is ignored the next time the system is shut down.
<b>Instant actions</b>	
<b>Shut down control PC</b>	Only available in operating modes T1 and T2. The robot controller is shut down
<b>Reboot control PC</b>	Only available in operating modes T1 and T2. The robot controller is shut down and then restarted.
<b>Drive bus OFF / ON</b>	Only available in operating modes T1 and T2. The drive bus can be switched off or on.  <b>Drive bus status</b> indicator: <ul style="list-style-type: none"> <li>■ Green: Drive bus is on.</li> <li>■ Red: Drive bus is off.</li> <li>■ Gray: Status of the drive bus is unknown.</li> </ul>

Start types	Start type	Description
	Cold start	<p>After a cold start the robot controller displays the Navigator. No program is selected. The robot controller is reinitialized, e.g. all user outputs are set to FALSE.</p> <p><b>Note:</b> If XML files have been changed directly, i.e. the user has opened the file and modified it, these changes are taken into consideration in the case of a cold start with <b>Reload files</b>. This cold start is called an “initial cold start”.</p> <p>In the case of a cold start without <b>Reload files</b>, these changes are not taken into consideration.</p>
	Hibernate	<p>After a start with Hibernate, the previously selected robot program can be resumed. The state of the kernel system: programs, block pointer, variable contents and outputs, is completely restored.</p> <p>Additionally, all programs that were open parallel to the robot controller are reopened and have the same state that they had before the system was shut down. The last state of Windows is also restored.</p>

#### 4.7 Switching the robot controller off

##### Procedure

- Turn the main switch on the robot controller to OFF.  
The robot controller automatically backs up data.

**NOTICE** The main switch on the robot controller must not be operated if the KSS has been exited with the option **Reboot control PC** and the reboot has not yet been completed. System files may otherwise be destroyed.

#### 4.8 Setting the user interface language

##### Procedure

1. In the main menu, select **Configuration > Miscellaneous > Language**.
2. Select the desired language. Click **OK** to confirm.

#### 4.9 Changing user group

##### Procedure

1. Select **Configuration > User group** in the main menu. The current user group is displayed.
2. Press **Default** to switch to the default user group. (**Default** is not available if the default user group is already selected.)  
Press **Login...** to switch to a different user group. Select the desired user group.
3. If prompted: Enter password and confirm with **Log-on**.

##### Description

Different functions are available in the KSS, depending on the user group. The following user groups are available:

##### ■ **Operator**

User group for the operator. This is the default user group.

##### ■ **User**

User group for the operator. (By default, the user groups “Operator” and “User” are defined for the same target group.)

■ **Expert**

User group for the programmer. This user group is protected by means of a password.

■ **Safety recovery**

This user group can activate and configure the safety configuration of the robot. This user group is protected by means of a password.

■ **Safety maintenance**

This user group is only relevant if KUKA.SafeOperation or KUKA.SafeRangeMonitoring is used. The user group is protected by means of a password.

■ **Administrator**

The range of functions is the same as that for the user group "Expert". It is additionally possible, in this user group, to integrate plug-ins into the robot controller.

This user group is protected by means of a password.

The default password is "kuka". The password can be changed.

(>>> 6.5 "Changing the password" Page 143)

When the system is booted, the default user group is selected.

If the mode is switched to AUT or AUT EXT, the robot controller switches to the default user group for safety reasons. If a different user group is desired, this must be selected subsequently.

If no actions are carried out in the user interface within a certain period of time, the robot controller switches to the default user group for safety reasons. The default setting is 300 s.

## 4.10 Changing operating mode



Do not change the operating mode while a program is running. If the operating mode is changed during program execution, the industrial robot is stopped with a safety stop 2.

### Precondition

- The robot controller is not executing a program.
- Key for the switch for calling the connection manager

### Procedure

1. On the smartPAD, turn the switch for the connection manager. The connection manager is displayed.
2. Select the operating mode.
3. Return the switch for the connection manager to its original position.

The selected operating mode is displayed in the status bar of the smartPAD.

Operating mode	Use	Velocities
T1	For test operation, programming and teaching	<ul style="list-style-type: none"> <li>■ Program verification: Programmed velocity, maximum 250 mm/s</li> <li>■ Jog mode: Jog velocity, maximum 250 mm/s</li> </ul>
T2	For test operation	<ul style="list-style-type: none"> <li>■ Program verification: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>

Oper- ing mode	Use	Velocities
AUT	For industrial robots without higher-level controllers	<ul style="list-style-type: none"> <li>■ Program mode: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC	<ul style="list-style-type: none"> <li>■ Program mode: Programmed velocity</li> <li>■ Jog mode: Not possible</li> </ul>

## 4.11 Coordinate systems

### Overview

The following Cartesian coordinate systems are defined in the robot controller:

- WORLD
- ROBROOT
- BASE
- TOOL

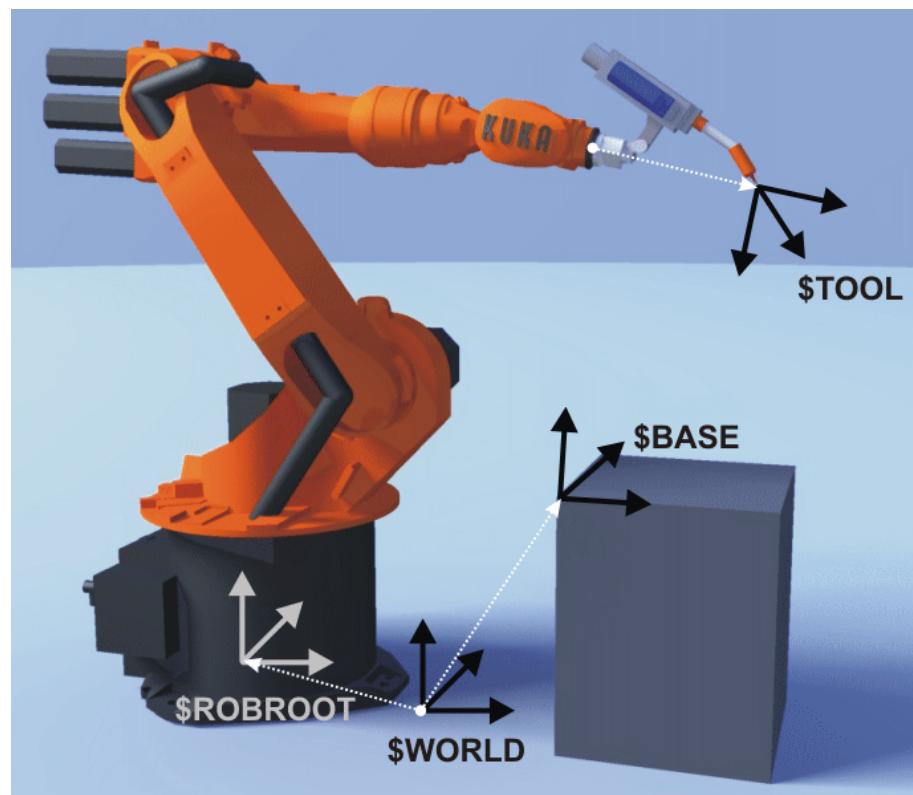


Fig. 4-8: Overview of coordinate systems

### Description

#### WORLD

The WORLD coordinate system is a permanently defined Cartesian coordinate system. It is the root coordinate system for the ROBROOT and BASE coordinate systems.

By default, the WORLD coordinate system is located at the robot base.

#### ROBROOT

The ROBROOT coordinate system is a Cartesian coordinate system, which is always located at the robot base. It defines the position of the robot relative to the WORLD coordinate system.

By default, the ROBROOT coordinate system is identical to the WORLD coordinate system. \$ROBROOT allows the definition of an offset of the robot relative to the WORLD coordinate system.

### **BASE**

The BASE coordinate system is a Cartesian coordinate system that defines the position of the workpiece. It is relative to the WORLD coordinate system.

By default, the BASE coordinate system is identical to the WORLD coordinate system. It is offset to the workpiece by the user.

(>>> 5.11.3 "Base calibration" Page 109)

### **TOOL**

The TOOL coordinate system is a Cartesian coordinate system which is located at the tool center point.

By default, the origin of the TOOL coordinate system is located at the flange center point. (In this case it is called the FLANGE coordinate system.) The TOOL coordinate system is offset to the tool center point by the user.

(>>> 5.11.2 "Tool calibration" Page 102)

### **Angles of rotation of the robot coordinate systems**

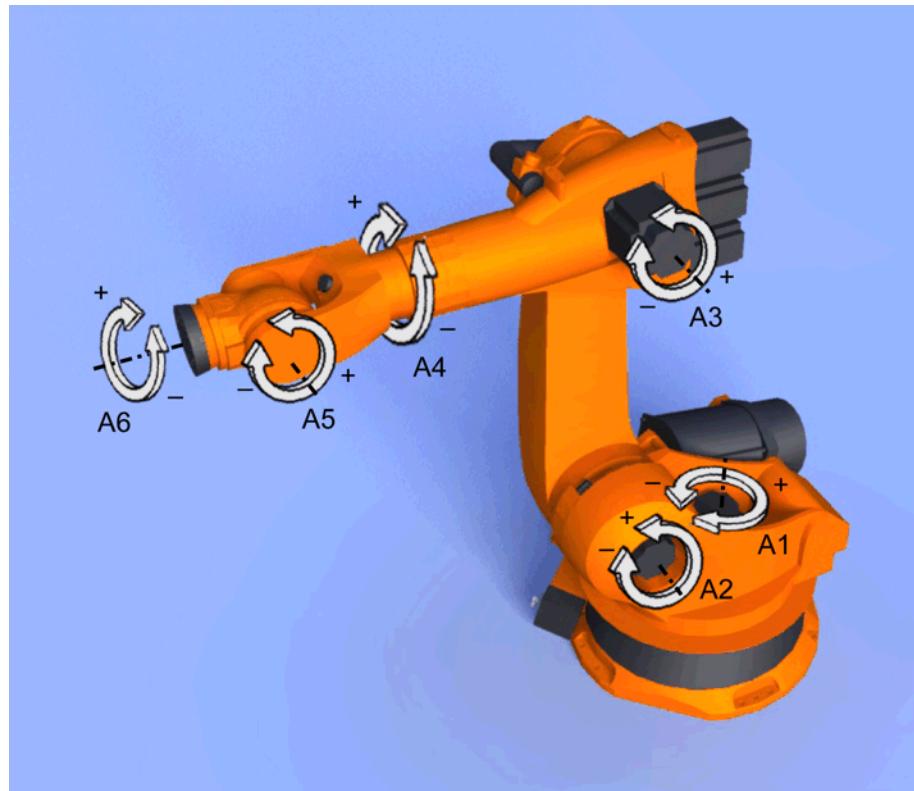
Angle	Rotation about axis
Angle A	Rotation about the Z axis
Angle B	Rotation about the Y axis
Angle C	Rotation about the X axis

## **4.12 Jogging the robot**

### **Description**

There are 2 ways of jogging the robot:

- **Cartesian jogging**  
The TCP is jogged in the positive or negative direction along the axes of a coordinate system.
- **Axis-specific jogging**  
Each axis can be moved individually in a positive and negative direction.



**Fig. 4-9: Axis-specific jogging**

There are 2 operator control elements that can be used for jogging the robot:

- Jog keys
- Space Mouse

#### Overview

	Cartesian jogging	Axis-specific jogging
Jog keys	(>>> 4.12.6 "Cartesian jogging with the jog keys" Page 61)	(>>> 4.12.5 "Axis-specific jogging with the jog keys" Page 61)
Space Mouse	(>>> 4.12.9 "Cartesian jogging with the Space Mouse" Page 64)	Axis-specific jogging with the Space Mouse is possible, but is not described here.

#### 4.12.1 “Jog options” window

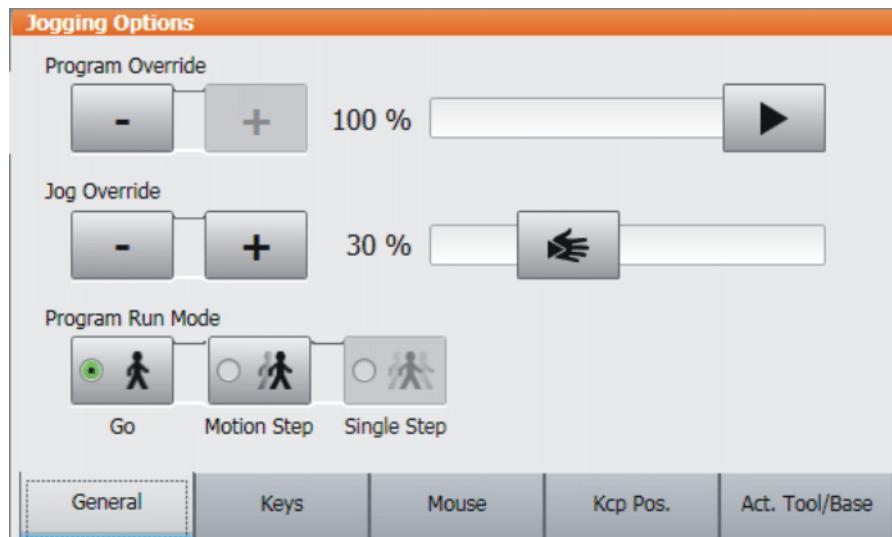
**Description** All parameters for jogging the robot can be set in the **Jogging Options** window.

**Procedure** Open the **Jogging Options** window:

1. Open a status indicator on the smartHMI, e.g. the **POV** status indicator.  
(Not possible for the **Submit interpreter**, **Drives** and **Robot interpreter** status indicators.)  
A window opens.
2. Press **Options**. The **Jogging Options** window is opened.

For most parameters, it is not necessary to open the **Jogging Options** window. They can be set directly via the smartHMI status indicators.

#### 4.12.1.1 “General” tab

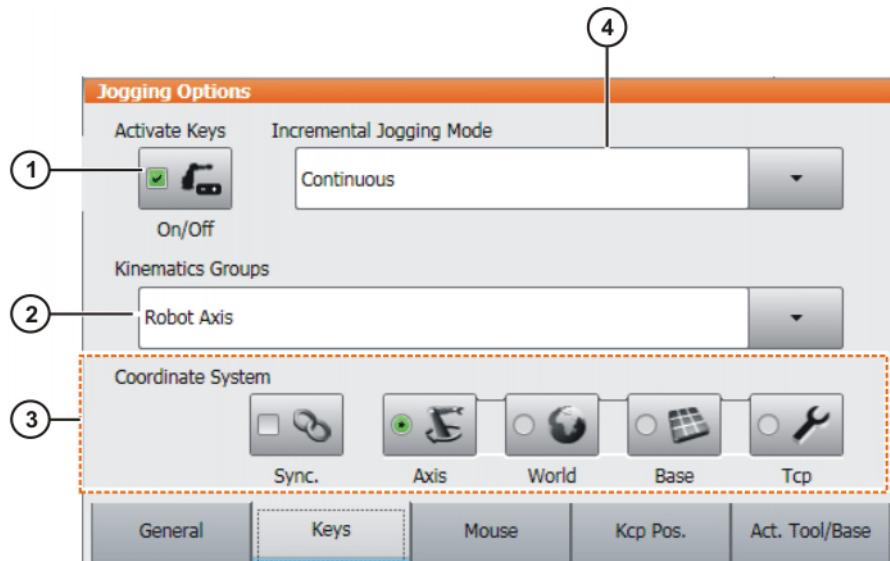


**Fig. 4-10: General tab**

#### Description

Item	Description
1	Set program override (>>> 7.5.4 "Setting the program override (POV)" Page 209)
2	Set jog override (>>> 4.12.3 "Setting the jog override (HOV)" Page 60)
3	Select the program run mode (>>> 7.5.2 "Program run modes" Page 208)

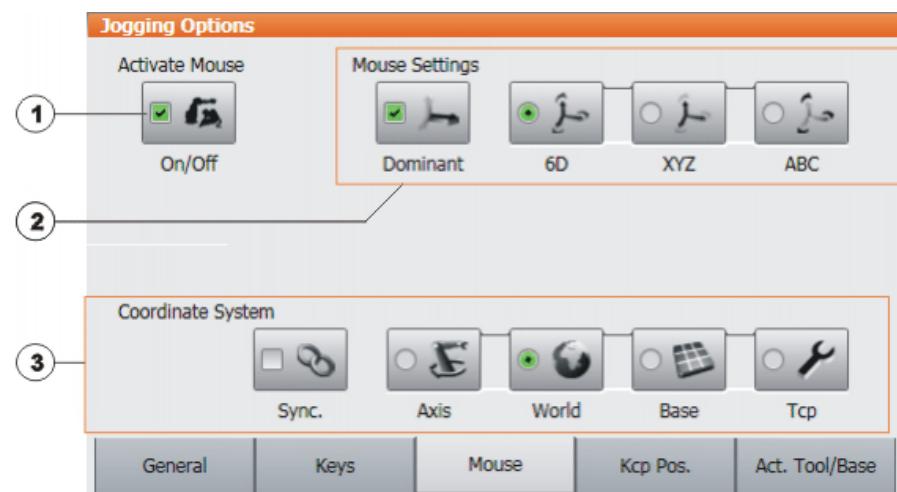
#### 4.12.1.2 “Keys” tab



**Fig. 4-11: Keys tab**

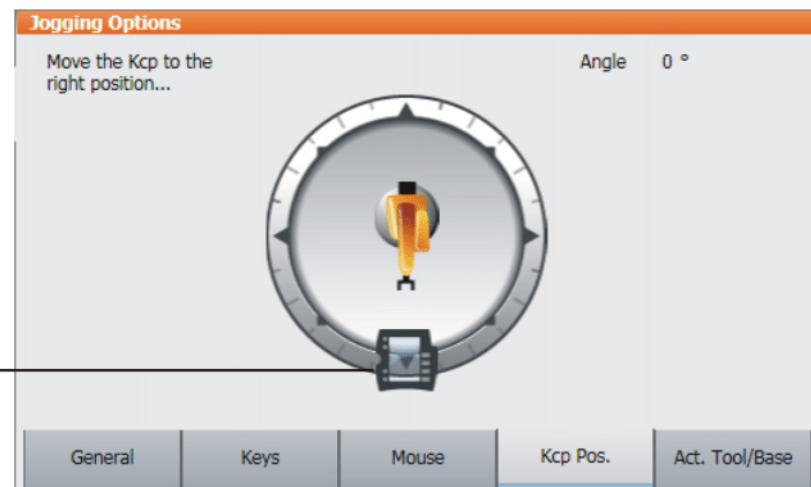
**Description**

Item	Description
1	Activate jog mode "Jog keys" (>>> 4.12.2 "Activating the jog mode" Page 60)
2	Select a kinematics group. The kinematics group defines the axes to which the jog keys refer. Default: <b>Robot axes</b> (= A1 to A6) Depending on the system configuration, other kinematics groups may be available. (>>> 4.13 "Jogging external axes" Page 66)
3	Select the coordinate system for jogging with the jog keys
4	Incremental jogging (>>> 4.12.10 "Incremental jogging" Page 65)

**4.12.1.3 "Mouse" tab****Fig. 4-12: Mouse tab****Description**

Item	Description
1	Activate jog mode "Space Mouse" (>>> 4.12.2 "Activating the jog mode" Page 60)
2	Configure the Space Mouse (>>> 4.12.7 "Configuring the Space Mouse" Page 61)
3	Select the coordinate system for jogging with the Space Mouse

#### 4.12.1.4 "KCP pos." tab

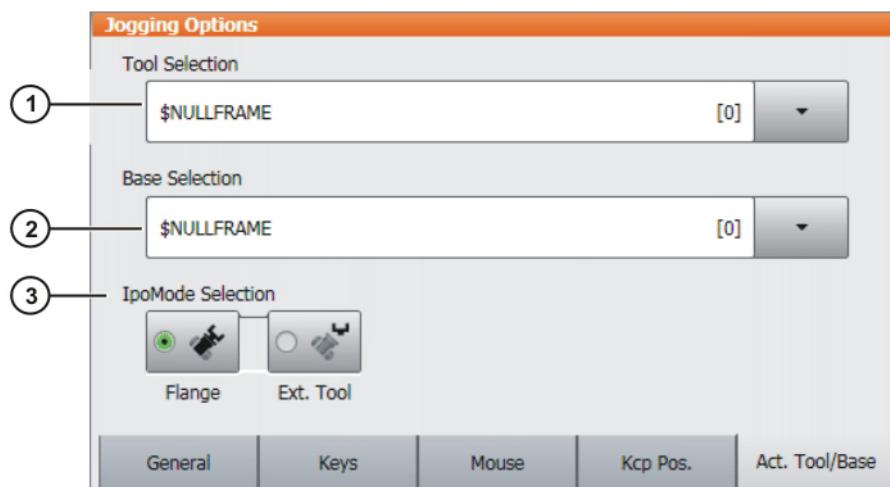


**Fig. 4-13: Kcp Pos. tab**

#### Description

Item	Description
1	(>>> 4.12.8 "Defining the alignment of the Space Mouse" Page 63)

#### 4.12.1.5 "Cur. tool/base" tab



**Fig. 4-14: Act. Tool/Base tab**

#### Description

Item	Description
1	The current tool is displayed here. A different tool can be selected. (>>> 4.12.4 "Selecting the tool and base" Page 60) The display <b>Unknown [?]</b> means that no tool has yet been calibrated.

Item	Description
2	<p>The current base is displayed here. A different base can be selected.</p> <p>(&gt;&gt;&gt; 4.12.4 "Selecting the tool and base" Page 60)</p> <p>The display <b>Unknown [?]</b> means that no base has yet been calibrated.</p>
3	<p>Select the interpolation mode:</p> <ul style="list-style-type: none"> <li>■ <b>Flange</b>: The tool is mounted on the mounting flange.</li> <li>■ <b>ext. Tool</b>: The tool is a fixed tool.</li> </ul>

#### 4.12.2 Activating the jog mode

<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. Open the <b>Jogging Options</b> window. (&gt;&gt;&gt; 4.12.1 "Jog options" window" Page 56)</li> <li>2. To activate the jog mode "Jog keys": On the <b>Keys</b> tab, activate the <b>Activate Keys</b> check box. To activate the jog mode "Space Mouse": On the <b>Mouse</b> tab, activate the <b>Activate Mouse</b> check box.</li> </ol>
<b>Description</b>	Both jog modes "Jog keys" and "Space Mouse" can be activated simultaneously. If the robot is jogged using the keys, the Space Mouse is disabled until the robot comes to a standstill. If the Space Mouse is actuated, the keys are disabled.

#### 4.12.3 Setting the jog override (HOV)

<b>Description</b>	Jog override is the velocity of the robot during jogging. It is specified as a percentage and refers to the maximum possible jog velocity. This is 250 mm/s.
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. Touch the <b>POV/HOV</b> status indicator. The <b>Overrides</b> window is opened.</li> <li>2. Set the desired jog override. It can be set using either the plus/minus keys or by means of the slide controller. <ul style="list-style-type: none"> <li>■ Plus/minus keys: The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%</li> <li>■ Slide controller: The override can be adjusted in 1% steps.</li> </ul> </li> <li>3. Touch the <b>POV/HOV</b> status indicator again. (Or touch the area outside the window.) The window closes and the selected override value is applied.</li> </ol>



The **Jog options** window can be opened via **Options** in the **Overrides** window.

<b>Alternative procedure</b>	Alternatively, the override can be set using the plus/minus key on the right-hand side of the KCP.  The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%.
------------------------------	---

#### 4.12.4 Selecting the tool and base

<b>Description</b>	A maximum of 16 TOOL and 32 BASE coordinate systems can be saved in the robot controller. One tool (TOOL coordinate system) and one base (BASE coordinate system) must be selected for Cartesian jogging.
--------------------	---

- Procedure**
1. Touch the **Tool/base** status indicator. The **Act. Tool/Base** window is opened.
  2. Select the desired tool and base.
  3. The window closes and the selection is applied.

#### 4.12.5 Axis-specific jogging with the jog keys

- Precondition**
- The jog mode “Jog keys” is active.
  - Operating mode T1
- Procedure**
1. Select **Axes** as the coordinate system for the jog keys.
  2. Set jog override.
  3. Hold down the enabling switch.  
Axes A1 to A6 are displayed next to the jog keys.
  4. Press the Plus or Minus jog key to move an axis in the positive or negative direction.

 The position of the robot during jogging can be displayed: select **Display > Actual position** in the main menu.

#### 4.12.6 Cartesian jogging with the jog keys

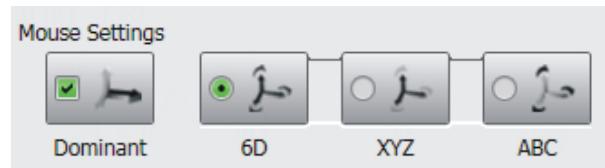
- Precondition**
- The jog mode “Jog keys” is active.
  - Operating mode T1
  - Tool and base have been selected.  
(>>> 4.12.4 "Selecting the tool and base" Page 60)
- Procedure**
1. Select **World, Base** or **Tool** as the coordinate system for the jog keys.
  2. Set jog override.
  3. Hold down the enabling switch.  
The following designations are displayed next to the jog keys:
    - **X, Y, Z**: for the linear motions along the axes of the selected coordinate system
    - **A, B, C**: for the rotational motions about the axes of the selected coordinate system
  4. Press the Plus or Minus jog key to move the robot in the positive or negative direction.

 The position of the robot during jogging can be displayed: select **Display > Actual position** in the main menu.

#### 4.12.7 Configuring the Space Mouse

- Procedure**
1. Open the **Jogging Options** window and select the **Mouse** tab.  
(>>> 4.12.1 "Jog options" window" Page 56)
  2. **Mouse Settings** group:
    - **Dominant** check box:  
Activate or deactivate dominant mode as desired.
    - **6D/XYZ/ABC** option box:  
Select whether the TCP is to be moved using translational motions, rotational motions, or both.

3. Close the **Jogging Options** window.

**Description**

**Fig. 4-15: Mouse settings**

**Dominant** check box:

Depending on the dominant mode, the Space Mouse can be used to move just one axis or several axes simultaneously.

Check box	Description
Active	Dominant mode is activated. Only the coordinate axis with the greatest deflection of the Space Mouse is moved.
Inactive	Dominant mode is deactivated. Depending on the axis selection, either 3 or 6 axes can be moved simultaneously.

Option	Description
<b>6D</b>	The robot can be moved by pulling, pushing, rotating or tilting the Space Mouse.  The following motions are possible with Cartesian jogging: <ul style="list-style-type: none"><li>■ Translational motions in the X, Y and Z directions</li><li>■ Rotational motions about the X, Y and Z axes</li></ul>
<b>XYZ</b>	The robot can only be moved by pulling or pushing the Space Mouse.  The following motions are possible with Cartesian jogging: <ul style="list-style-type: none"><li>■ Translational motions in the X, Y and Z directions</li></ul>
<b>ABC</b>	The robot can only be moved by rotating or tilting the Space Mouse.  The following motions are possible with Cartesian jogging: <ul style="list-style-type: none"><li>■ Rotational motions about the X, Y and Z axes</li></ul>

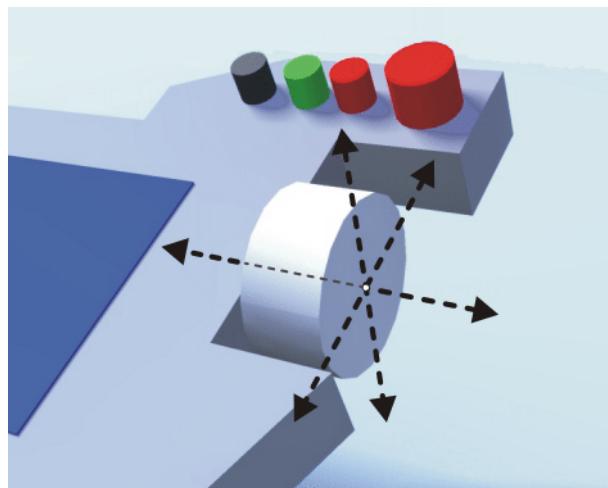


Fig. 4-16: Pushing and pulling the Space Mouse

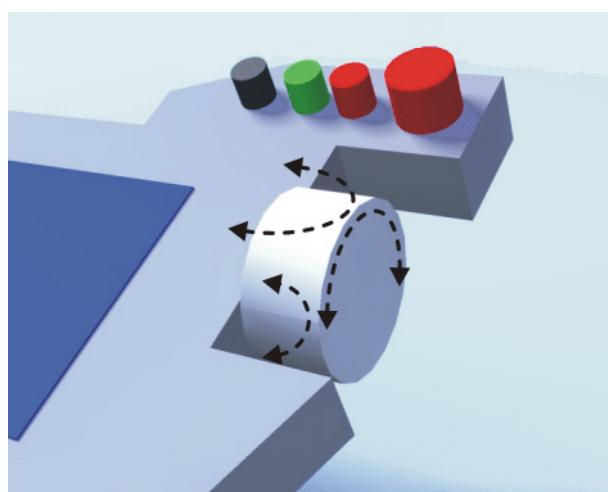


Fig. 4-17: Rotating and tilting the Space Mouse

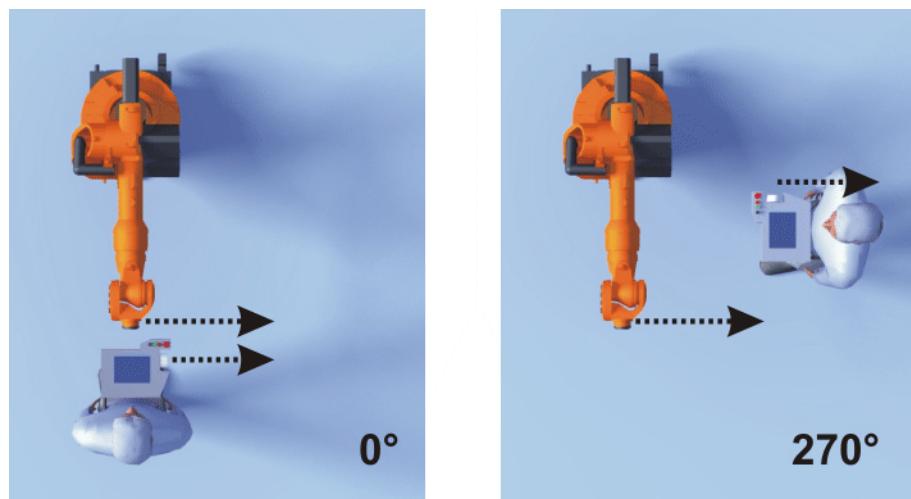
#### 4.12.8 Defining the alignment of the Space Mouse

##### Description

The functioning of the Space Mouse can be adapted to the location of the user so that the motion direction of the TCP corresponds to the deflection of the Space Mouse.

The location of the user is specified in degrees. The reference point for the specification in degrees is the junction box on the base frame. The position of the robot arm or axes is irrelevant.

Default setting: 0°. This corresponds to a user standing opposite the junction box.



**Fig. 4-18: Space Mouse: 0° and 270°**

**Precondition**

- Operating mode T1

**Procedure**

1. Open the **Jog options** window and select the **Kcp pos.** tab.



**Fig. 4-19: Defining the alignment of the Space Mouse**

2. Drag the KCP to the position corresponding to the location of the user (in 45° steps).
3. Close the **Jog options** window.



Switching to Automatic External mode automatically resets the alignment of the Space Mouse to 0°.

#### 4.12.9 Cartesian jogging with the Space Mouse

**Precondition**

- The jog mode "Space Mouse" is active.
- Operating mode T1
- Tool and base have been selected.  
(>>> 4.12.4 "Selecting the tool and base" Page 60)
- The Space Mouse is configured.  
(>>> 4.12.7 "Configuring the Space Mouse" Page 61)

- The alignment of the Space Mouse has been defined.  
(>>> 4.12.8 "Defining the alignment of the Space Mouse" Page 63)

- Procedure**
1. Select **World**, **Base** or **Tool** as the coordinate system for the Space Mouse.
  2. Set jog override.
  3. Hold down the enabling switch.
  4. Move the robot in the desired direction using the Space Mouse.



The position of the robot during jogging can be displayed: select **Display > Actual position** in the main menu.

#### 4.12.10 Incremental jogging

- Description** Incremental jogging makes it possible to move the robot a defined distance, e.g. 10 mm or 3°. The robot then stops by itself.

Incremental jogging can be activated for jogging with the jog keys. Incremental jogging is not possible in the case of jogging with the Space Mouse.

Areas of application:

- Positioning of equidistant points
- Moving a defined distance away from a position, e.g. in the event of a fault
- Mastering with the dial gauge

The following options are available:

Setting	Description
<b>Continuous</b>	Incremental jogging is deactivated.
<b>100 mm / 10°</b>	1 increment = 100 mm or 10°
<b>10 mm / 3°</b>	1 increment = 10 mm or 3°
<b>1 mm / 1°</b>	1 increment = 1 mm or 1°
<b>0.1 mm / 0.005°</b>	1 increment = 0.1 mm or 0.005°

Increments in mm:

- Valid for Cartesian jogging in the X, Y or Z direction.

Increments in degrees:

- Valid for Cartesian jogging in the A, B or C direction.
- Valid for axis-specific jogging.

- Precondition**
- The jog mode "Jog keys" is active.
  - Operating mode T1

- Procedure**
1. Select the size of the increment in the status bar.
  2. Jog the robot using the jog keys. Jogging can be Cartesian or axis-specific.

Once the set increment has been reached, the robot stops.



If the robot motion is interrupted, e.g. by releasing the enabling switch, the interrupted increment is not resumed with the next motion; a new increment is started instead.

## 4.13 Jogging external axes

External axes cannot be moved using the Space Mouse. If "Space Mouse" mode is selected, only the robot can be jogged with the Space Mouse. The external axes, on the other hand, must be jogged using the jog keys.

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ The jog mode "Jog keys" is active.</li> <li>■ Operating mode T1</li> </ul>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. Select the desired kinematics group, e.g. <b>External axes</b>, on the <b>Keys</b> tab in the <b>Jog options</b> window. The type and number of kinematics groups available depend on the system configuration.</li> <li>2. Set jog override.</li> <li>3. Hold down the enabling switch. The axes of the selected kinematics group are displayed next to the jog keys.</li> <li>4. Press the Plus or Minus jog key to move an axis in the positive or negative direction.</li> </ol>
<b>Description</b>	Depending on the system configuration, the following kinematics groups may be available.

Kinematics group	Description
<b>Robot Axes</b>	The robot axes can be moved using the jog keys. The external axes cannot be jogged.
<b>External Axes</b>	All configured external axes (e.g. external axes E1 to E5) can be moved using the jog keys.
<b>NAME / External Kinematics Group<math>n</math></b>	<p>The axes of an external kinematics group can be moved using the jog keys.</p> <p>The name is taken from the system variable <math>\\$ETn\_NAME</math> (<math>n</math> = number of the external kinematic system). If <math>\\$ETn\_NAME</math> is empty, the default name <b>External Kinematics Group<math>n</math></b> is displayed.</p>
[User-defined kinematics group]	<p>The axes of a user-defined kinematics group can be moved using the jog keys.</p> <p>The name corresponds to the name of the user-defined kinematics group.</p>

## 4.14 Bypassing workspace monitoring

<b>Description</b>	<p>Workspaces can be configured for a robot. Workspaces serve to protect the system.</p> <p>There are 2 types of workspace:</p> <ul style="list-style-type: none"> <li>■ The workspace is an exclusion zone. The robot may only move outside the workspace.</li> <li>■ Only the workspace is a permitted zone. The robot may not move outside the workspace.</li> </ul> <p>Exactly what reactions occur when the robot violates a workspace depends on the configuration. (<a href="#">&gt;&gt;&gt; 6.6 "Configuring workspaces" Page 143</a>)</p> <p>One possible reaction, for example, is that the robot stops and an error message is generated. The workspace monitoring must be bypassed in such a case. The robot can then move back out of the prohibited workspace.</p>
--------------------	--

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ User group “Expert”</li> <li>■ Operating mode T1</li> </ul>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Configuration &gt; Miscellaneous &gt; Workspace monitoring &gt; Override</b>.</li> <li>2. Move the robot manually out of the prohibited workspace. Once the robot has left the prohibited workspace, the workspace monitoring is automatically active again.</li> </ol>

## 4.15 Monitor functions

### 4.15.1 Displaying the actual position

<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Display &gt; Actual position</b>. The Cartesian actual position is displayed.</li> <li>2. To display the axis-specific actual position, press <b>Axis-specific</b>.</li> <li>3. To display the Cartesian actual position again, press <b>Cartesian</b>.</li> </ol>
------------------	--

#### Description      **Actual position, Cartesian:**

The current position (X, Y, Z) and orientation (A, B, C) of the TCP are displayed. Status and Turn are also displayed.

#### **Actual position, axis-specific:**

The current position of axes A1 to A6 is indicated. If external axes are being used, the position of the external axes is also displayed.

The actual position can also be displayed while the robot is moving.

Robot Position (Axis Specific)			
Axis	Pos. [deg, mm]	Motor [deg]	
A1	0.00	0.00	
A2	-90.00	22530.00	
A3	90.00	-24228.95	
A4	0.00	0.00	
A5	0.00	0.00	
A6	0.00	0.00	
E1	0.00	0.00	

**Fig. 4-20: Actual position, axis-specific**

### 4.15.2 Displaying digital inputs/outputs

<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Display &gt; Inputs/outputs &gt; Digital I/O</b>.</li> <li>2. To display a specific input/output: <ul style="list-style-type: none"> <li>■ Click on the <b>Go to</b> button. The <b>Go to:</b> box is displayed.</li> <li>■ Enter the number and confirm with the Enter key.</li> </ul> </li> </ol>
------------------	---

The display jumps to the input/output with this number.

**Description**


Digital I/O			
No.	Value	State	Name
139	○		Eingang
140	○	SYS	Eingang
141	○		Eingang
142	●	SIM	Eingang
143	○		Eingang
144	○		Finnano

Inputs   Outputs

-100   +100   Go to   Value   Name

**Fig. 4-21: Digital inputs**


Digital I/O			
No.	Value	State	Name
991	○		Ausgang
992	○		Ausgang
993	●	SYS	Ausgang
994	○	SYS	Ausgang
995	○	SYS	Ausgang
996	○	SYS	Ausgang

Inputs   Outputs

-100   +100   Go to   Value   Name

**Fig. 4-22: Digital outputs**

Item	Description
1	Input/output number
2	Value of the input/output. The icon is red if the input or output is TRUE.
3	SIM entry: The input/output is simulated. SYS entry: The value of the input/output is saved in a system variable. This input/output is write-protected.
4	Name of the input/output

The following buttons are available:

Button	Description
-100	Toggles back 100 inputs or outputs in the display.
+100	Toggles forward 100 inputs or outputs in the display.
Go to	The number of the input or output being searched for can be entered.

Button	Description
<b>Value</b>	Toggles the selected input/output between TRUE and FALSE. Precondition: The enabling switch is pressed.  This button is not available in AUT EXT mode, and is only available for inputs if simulation is activated.
<b>Name</b>	The name of the selected input or output can be changed.

#### 4.15.3 Displaying analog inputs/outputs

##### Procedure

1. In the main menu, select **Display > Inputs/outputs > Analog I/O**.
2. To display a specific input/output:
  - Click on the **Go to** button. The **Go to:** box is displayed.
  - Enter the number and confirm with the Enter key.
 The display jumps to the input/output with this number.

The following buttons are available:

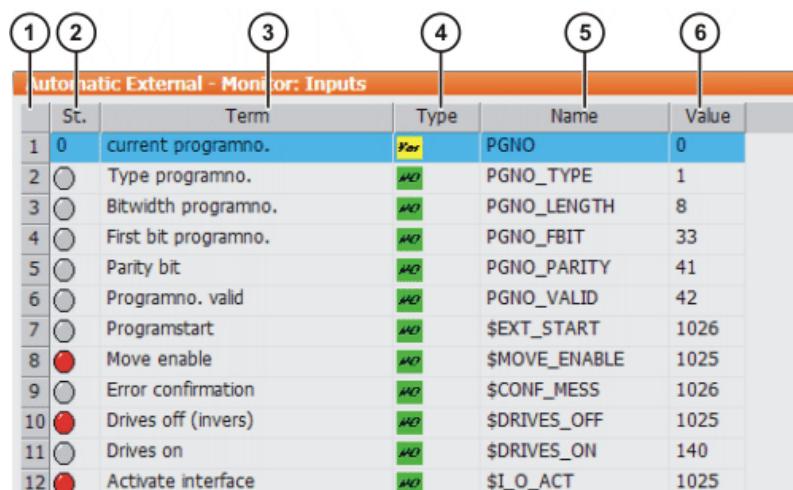
Button	Description
<b>Go to</b>	The number of the input or output being searched for can be entered.
<b>Voltage</b>	A voltage can be entered for the selected output. <ul style="list-style-type: none"> <li>■ <b>-10 ... 10 V</b></li> </ul> This button is only available for outputs.
<b>Name</b>	The name of the selected input or output can be changed.

#### 4.15.4 Displaying inputs/outputs for Automatic External

##### Procedure

- In the main menu, select **Display > Inputs/outputs > Automatic External**.

##### Description



St.	Term	Type	Name	Value
1	0 current programno.	PGNO	PGNO	0
2	Type programno.	PGNO_TYPE	PGNO_TYPE	1
3	Bitwidth programno.	PGNO_LENGTH	PGNO_LENGTH	8
4	First bit programno.	PGNO_FBIT	PGNO_FBIT	33
5	Parity bit	PGNO_PARITY	PGNO_PARITY	41
6	Programno. valid	PGNO_VALID	PGNO_VALID	42
7	Programstart	\$EXT_START	\$EXT_START	1026
8	Move enable	\$MOVE_ENABLE	\$MOVE_ENABLE	1025
9	Error confirmation	\$CONF_MESS	\$CONF_MESS	1026
10	Drives off (invers)	\$DRIVES_OFF	\$DRIVES_OFF	1025
11	Drives on	\$DRIVES_ON	\$DRIVES_ON	140
12	Activate interface	\$I_O_ACT	\$I_O_ACT	1025

Fig. 4-23: Automatic External inputs (detail view)

Automatic External - Monitor: Outputs					
	St.	Term	Type	Name	Value
1	●	Control ready	AO	\$RC_RDY1	137
2	●	Alarm stop active	AO	\$ALARM_STOP	1013
3	●	User safety switch closed	AO	\$USER_SAF	1011
4	●	Drives ready	AO	\$PERI_RDY	1012
5	●	Robot calibrated	AO	\$ROB_CAL	1001
6	○	Interface active	AO	\$I_O_ACTCONF	140
7	○	Error collection	AO	\$STOPMESS	1010
8	●	Internal emergency stop	AO	IntEstop	1002

Fig. 4-24: Automatic External outputs (detail view)

Item	Description
1	Number
2	Status <ul style="list-style-type: none"> <li>■ Gray: inactive (FALSE)</li> <li>■ Red: active (TRUE)</li> </ul>
3	Long text name of the input/output
4	Type <ul style="list-style-type: none"> <li>■ Green: input/output</li> <li>■ Yellow: variable or system variable (\$...)</li> </ul>
5	Name of the signal or variable
6	Input/output number or channel number

Columns 4, 5 and 6 are only displayed if **Details** has been pressed.

The following buttons are available:

Button	Description
<b>Config.</b>	Switches to the configuration of the Automatic External interface. ( <a href="#">&gt;&gt;&gt; 6.10.2 "Configuring Automatic External inputs/outputs" Page 160</a> )
<b>Inputs/Outputs</b>	Toggles between the windows for inputs and outputs.
<b>Details/Normal</b>	Toggles between the <b>Details</b> and <b>Normal</b> views.

#### 4.15.5 Displaying and modifying the value of a variable

##### Precondition

To modify a variable:

- “Expert” user group

##### Procedure

1. In the main menu, select **Display > Variable > Single**.  
The **Variable Overview - Single** window is opened.
2. Enter the name of the variable in the **Name** box and confirm with the **Enter** key.
3. If a program has been selected, it is automatically entered in the **Module** box.  
If a variable from a different program is to be displayed, enter the program as follows:

*/R1/Program name*

Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.

**i** In the case of system variables, no program needs to be specified in the **Module** box.

4. The current value of the variable is displayed in the **Current value** box. If nothing is displayed, no value has yet been assigned to the variable.  
To modify the variable:
5. Enter the desired value in the **New value** box.
6. Press the **Set value** button. The new value is displayed in the **Current value** box.

#### Description

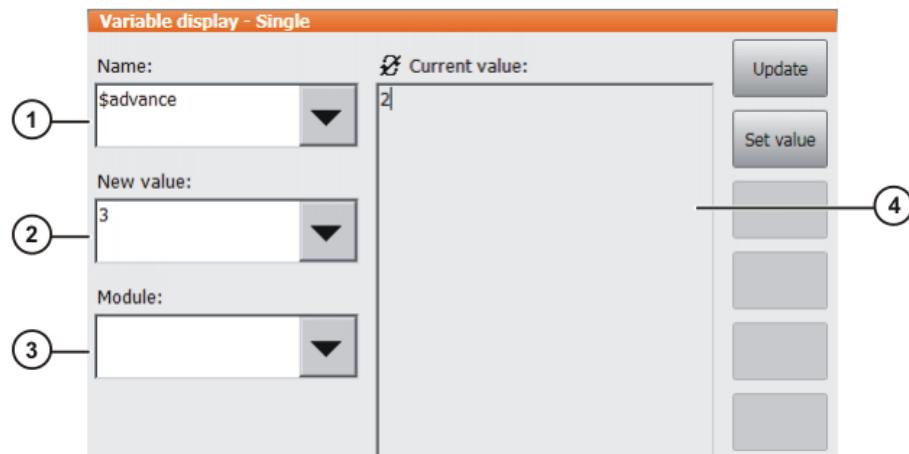


Fig. 4-25: Variable Overview - Single window

Item	Description
1	Name of the variable to be modified.
2	New value to be assigned to the variable.
3	Program in which the search for the variable is to be carried out. In the case of system variables, the <b>Module</b> box is irrelevant.
4	This box has two states: <ul style="list-style-type: none"> <li>■  : The displayed value is not refreshed automatically.</li> <li>■  : The displayed value is refreshed automatically.</li> </ul> Switching between the states: <ul style="list-style-type: none"> <li>■ Press <b>Refresh</b>.</li> </ul>

#### 4.15.6 Displaying the state of a variable

##### Description

Variables can have the following states:

- UNKNOWN: The variable is unknown.
- DECLARED: The variable is declared.
- INITIALIZED: The variable is initialized.

##### Procedure

1. In the main menu, select **Display > Variable > Single**.  
The **Variable Overview - Single** window is opened.
2. In the **Name** box, enter: =VARSTATE("name")  
*name* = name of the variable whose state is to be displayed.
3. If a program has been selected, it is automatically entered in the **Module** box.

If a variable from a different program is to be displayed, enter the program as follows:

/R1/Program name

Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.



In the case of system variables, no program needs to be specified in the **Module** box.

4. Press **Refresh**.

The current state of the variable is displayed in the **Current value** box.

#### 4.15.7 Displaying the variable overview and modifying variables

In the variable overview, variables are displayed in groups. The variables can be modified.

The number of groups and which variables they contain are defined in the configuration.

(>>> 6.4 "Configuring the variable overview" Page 142)



Variables can only be displayed and modified in the user group "User" if these functions have been enabled in the configuration.

##### Procedure

1. In the main menu, select **Display > Variable > Overview > Display**.  
The **Variable overview - Display** window is opened.
2. Select the desired group.
3. Select the cell to be modified. Carry out modification using the buttons.
4. Press **OK** to save the change and close the window.

##### Description

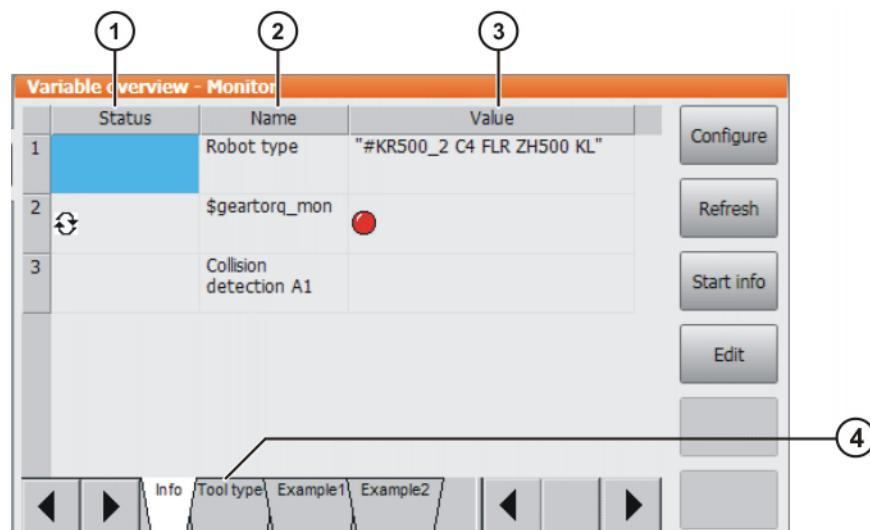


Fig. 4-26: Variable overview - Monitor window

Item	Description
1	Arrow symbol  : If the value of the variable changes, the display is automatically refreshed. No arrow symbol: The display is not automatically refreshed.
2	Descriptive name

Item	Description
3	Value of the variable. In the case of inputs/outputs, the state is indicated: <ul style="list-style-type: none"> <li>■ <b>Gray:</b> inactive (FALSE)</li> <li>■ <b>Red:</b> active (TRUE)</li> </ul>
4	There is one tab per group.

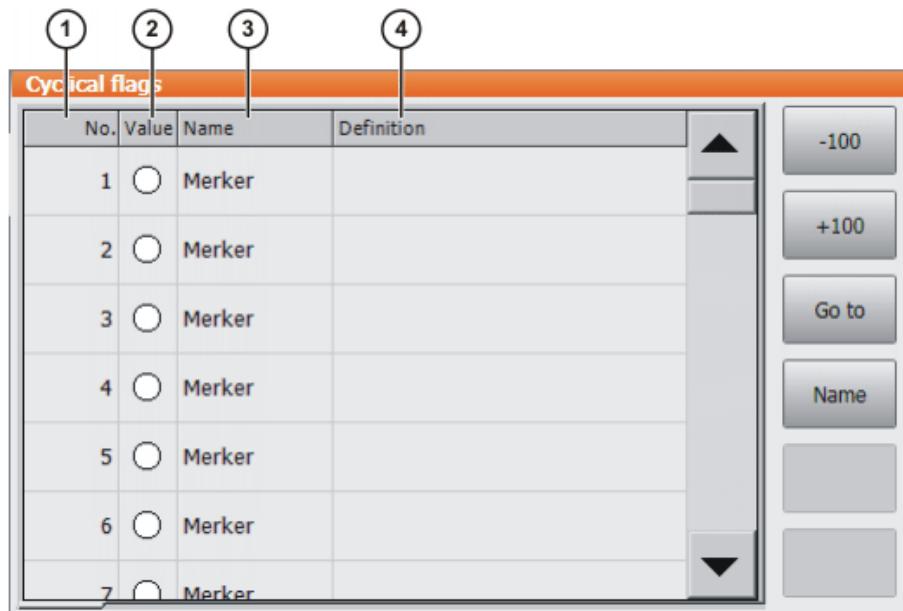
The following buttons are available:

Button	Description
<b>Config.</b>	Switches to the configuration of the variable overview.  (>>> 6.4 "Configuring the variable overview" Page 142)  This button is not available in the user group "User".
<b>Refresh all</b>	Refreshes the display.
<b>Cancel info</b>	Deactivates the automatic refreshing function.
<b>Start info</b>	Activates the automatic refreshing function.  A maximum of 12 variables per group can be refreshed automatically.
<b>Edit</b>	Switches the current cell to edit mode so that the name or value can be modified. In the <b>Value</b> column, this button changes the state of inputs/outputs (TRUE/FALSE).  This button is only available in the user group "User" if it has been enabled in the configuration.  <b>Note:</b> The values of write-protected variables cannot be changed.

#### 4.15.8 Displaying cyclical flags

##### Procedure

1. In the main menu, select **Display > Variable > Cyclical flags**. The **Cyclical flags** window is opened.
2. To display a specific flag:
  - Click on the **Go to** button. The **Go to:** box is displayed.
  - Enter the number and confirm with the Enter key.
 The display jumps to the flag with this number.

**Description****Fig. 4-27: Cyclical flags**

Item	Description
1	Number of the flag
2	Value of the flag. The icon is red if a flag is set.
3	Name of the flag
4	The conditions linked to the setting of a cyclical flag are indicated here.

The following buttons are available:

Button	Description
<b>-100</b>	Toggles back 100 flags in the display.
<b>+100</b>	Toggles forward 100 flags in the display.
<b>Go to</b>	The number of the flag being searched for can be entered.
<b>Name</b>	The name of the selected flag can be modified.

**4.15.9 Displaying flags****Procedure**

1. In the main menu, select **Display > Variable > Flags**. The **Flags** window is opened.
2. To display a specific flag:
  - Click on the **Go to** button. The **Go to:** box is displayed.
  - Enter the number and confirm with the Enter key.
 The display jumps to the flag with this number.

## Description

No.	Value	Name
1		Flag
2		Flag
3		Flag
4		Flag
5		Flag
6		Flag
7		Flag

Fig. 4-28: Flags

Item	Description
1	Number of the flag
2	Value of the flag. The icon is red if a flag is set.
3	Name of the flag

The following buttons are available:

Button	Description
<b>-100</b>	Toggles back 100 flags in the display.
<b>+100</b>	Toggles forward 100 flags in the display.
<b>Go to</b>	The number of the flag being searched for can be entered.
<b>Value</b>	Toggles the selected flag between TRUE and FALSE. Precondition: The enabling switch is pressed. This button is not available in AUT EXT mode.
<b>Name</b>	The name of the selected flag can be modified.

### 4.15.10 Displaying counters

#### Procedure

- In the main menu, select **Display > Variable > Counter**. The **Counter** window is opened.
  - To display a specific counter:
    - Click on the **Go to** button. The **Go to:** box is displayed.
    - Enter the number and confirm with the Enter key.
- The display jumps to the counter with this number.

**Description**

No.	Value	Name
1	0	Zaehler
2	0	Zaehler
3	0	Zaehler
4	8	my_counter
5	0	Zaehler
6	0	Zaehler
7	0	Zaehler

**Fig. 4-29: Counter**

Item	Description
1	Counter number
4	Value of the counter.
5	Name of counter

The following buttons are available:

Button	Description
<b>Go to</b>	The number of the counter being searched for can be entered.
<b>Value</b>	A value can be entered for the selected counter.
<b>Name</b>	The name of the selected counter can be modified.

**4.15.11 Displaying timers****Procedure**

1. In the main menu, select **Display > Variable > Timer**. The **Timer** window is opened.
2. To display a specific timer:
  - Click on the **Go to** button. The **Go to:** box is displayed.
  - Enter the number and confirm with the Enter key.
 The display jumps to the timer with this number.

**Description**

No.	Status	T	Value [m...]	Name
1	■		0	Timer
2	■		0	Timer
3	■	✓	59376	Timer
4	■		0	Timer
5	■		0	Timer
6	■		0	Timer
7	■		0	Timer

1
2
3
4
5

Go to
Status
Value
Name

**Fig. 4-30: Timer**

Item	Description
1	Number of the timer
2	Status of the timer <ul style="list-style-type: none"> <li>■ If the timer is activated, this is indicated in green.</li> <li>■ If the timer is deactivated, this is indicated in red.</li> </ul>
3	State of the timer <ul style="list-style-type: none"> <li>■ If the value of the timer is &gt; 0, the timer flag is set (red check mark).</li> <li>■ If the value of the timer is ≤ 0, no timer flag is set.</li> </ul>
4	Value of the timer (unit: ms)
5	Name of timer

The following buttons are available:

Button	Description
<b>Go to</b>	The number of the timer being searched for can be entered.
<b>State</b>	Toggles the selected timer between TRUE and FALSE. Precondition: The enabling switch is pressed.
<b>Value</b>	A value can be entered for the selected timer.
<b>Name</b>	The name of the selected timer can be modified.

**4.15.12 Displaying calibration data****Procedure**

- In the main menu, select **Start-up > Calibrate > Calibration points** and the desired menu item:
  - **Tool type**
  - **Base type**
  - **External axis**
- Enter the number of the tool, base or external kinematic system.  
The calibration method and the calibration data are displayed.

#### 4.15.13 Displaying information about the robot and robot controller

**Procedure**

- Select **Help > Info** in the main menu.

**Description**

The information is required, for example, when requesting help from KUKA Customer Support.

The tabs contain the following information:

Tab	Description
<b>Info</b>	<ul style="list-style-type: none"><li>■ Robot controller type</li><li>■ Robot controller version</li><li>■ User interface version</li><li>■ Kernel system version</li></ul>
<b>Robot</b>	<ul style="list-style-type: none"><li>■ Robot name</li><li>■ Robot type and configuration</li><li>■ Service life</li></ul> <p>The operating hours meter is running as long as the drives are switched on. Alternatively, the operating hours can also be displayed via the variable \$ROB-RUNTIME.</p> <ul style="list-style-type: none"><li>■ Number of axes</li><li>■ List of external axes</li><li>■ Machine data version</li></ul>
<b>System</b>	<ul style="list-style-type: none"><li>■ Control PC name</li><li>■ Operating system versions</li><li>■ Storage capacities</li></ul>
<b>Options</b>	Additionally installed options and technology packages
<b>Comments</b>	Additional comments
<b>Modules</b>	Names and versions of important system files  The <b>Save</b> button exports the contents of the <b>Modules</b> tab to the file C:\KRC\ROBOTER\LOG\OCXVER.TXT.

#### 4.15.14 Displaying/editing robot data

**Precondition**

- T1 or T2 operating mode
- No program is selected.

**Procedure**

- Select **Start-up > Robot data** in the main menu.

**Description**

Robot data	
Serial number	<input type="text" value="0"/> 1
Runtime [Hours]	<input type="text" value="0.00"/> 2
Machine data	<input type="text" value="#KR2210_2 S C4 FLR ZH210"/> 3
Robot name	<input type="text" value="0"/> 4
Network archive path	<input type="text"/> 5
Archive name	<input type="text" value="0"/> 6
<input type="checkbox"/> Incorporate robot name into archive name. 7	

**Fig. 4-31: Robot data window**

Item	Description
1	Serial number
2	Operating hours. The operating hours meter is running as long as the drives are switched on. Alternatively, the operating hours can also be displayed via the variable \$ROBRUNTIME.
3	Machine data name
4	Robot name. The robot name can be changed.
5	Robot controller data can be archived to a network path. (>>> 7.8.3 "Archiving on the network" Page 218) The archive path is defined here.
6	This box is only displayed if the check box <b>Incorporate robot name into archive name.</b> is not activated. A name for the archive file can be defined here.
7	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> The robot name is used as the name for the archive file. If no robot name is defined, the name <i>archive</i> is used.</li> <li>■ <b>Check box not active:</b> A separate name can be defined for the archive file.</li> </ul>

The following buttons are available in the user group "Expert":

Button	Description
<b>Transfer PID&gt;&gt;RDC</b>	Only relevant for positionally accurate robots: the XML file with the data for the positionally accurate robot can be transferred manually to the RDC.  Pressing this button displays the directory structure. The directory containing the file with the current serial number is selected here. The file can be selected and transferred to the RDC.
<b>Transfer MAM&gt;&gt;RDC</b>	Only relevant for robots with fixed mastering marks: the MAM file with the robot-specific mastering offset data can be transferred manually to the RDC.  Pressing this button displays the directory structure. The directory containing the file with the current serial number is selected here. The file can be selected and transferred to the RDC.

Button	Description
<b>Transfer CAL&gt;&gt;RDC</b>	The CAL file with the EMD mastering data can be transferred manually to the RDC.  Pressing this button displays the directory structure. The directory containing the file with the current serial number is selected here. The file can be selected and transferred to the RDC.
<b>Save RDC data</b>	The data on the RDC can be backed up temporarily in the directory C:\KRC\Roboter\RDC by pressing this button.  <b>Note:</b> The directory is deleted when the robot controller is rebooted or data are archived. If the RDC data are to be retained permanently, they must be backed up elsewhere.

#### 4.16 Displaying the battery state

<b>Description</b>	If the robot controller is switched off at the main switch, or if the power fails, the robot controller is backed up by a battery and is shut down in a controlled manner (without loss of data). The battery charge can be displayed for the user. The user can also transfer it to the PLC.  The battery charge is displayed by means of the system variable \$ACCU_STATE.  The state can only be displayed and not modified.  The charging current characteristic is monitored every time the robot controller is booted. An additional battery test is carried out cyclically. The state indicated by \$ACCU_STATE is derived from the information regarding the charging current and the battery test.														
<b>States</b>	Possible states for \$ACCU_STATE:  <table border="1"> <tr> <td></td> <td>The user must configure the info to the PLC himself.</td> </tr> </table> <table border="1"> <tr> <td></td> <td>Information about exchanging the battery can be found in the operating instructions for the robot controller.</td> </tr> </table> <table border="1"> <tr> <td>#CHARGE_OK</td> </tr> <tr> <td><b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test.</td> </tr> <tr> <td><b>Action required by the user:</b> Do not exchange the battery.</td> </tr> <tr> <td><b>Info to PLC:</b> Supply voltage disconnection OK.</td> </tr> <tr> <td><b>Message:</b> No message.</td> </tr> </table> <table border="1"> <tr> <td>#CHARGE_OK_LOW</td> </tr> <tr> <td><b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test. The battery is not fully charged, however, after the maximum charging time.</td> </tr> <tr> <td><b>Action required by the user:</b> Exchange the battery.</td> </tr> <tr> <td><b>Info to PLC:</b> Supply voltage disconnection OK.</td> </tr> <tr> <td><b>Message:</b> Battery warning - full charge not possible</td> </tr> </table>		The user must configure the info to the PLC himself.		Information about exchanging the battery can be found in the operating instructions for the robot controller.	#CHARGE_OK	<b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test.	<b>Action required by the user:</b> Do not exchange the battery.	<b>Info to PLC:</b> Supply voltage disconnection OK.	<b>Message:</b> No message.	#CHARGE_OK_LOW	<b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test. The battery is not fully charged, however, after the maximum charging time.	<b>Action required by the user:</b> Exchange the battery.	<b>Info to PLC:</b> Supply voltage disconnection OK.	<b>Message:</b> Battery warning - full charge not possible
	The user must configure the info to the PLC himself.														
	Information about exchanging the battery can be found in the operating instructions for the robot controller.														
#CHARGE_OK															
<b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test.															
<b>Action required by the user:</b> Do not exchange the battery.															
<b>Info to PLC:</b> Supply voltage disconnection OK.															
<b>Message:</b> No message.															
#CHARGE_OK_LOW															
<b>Meaning:</b> The charging current dropped as required after booting and/or the battery tested positive in the battery test. The battery is not fully charged, however, after the maximum charging time.															
<b>Action required by the user:</b> Exchange the battery.															
<b>Info to PLC:</b> Supply voltage disconnection OK.															
<b>Message:</b> Battery warning - full charge not possible															

<b>#CHARGE_UNKNOWN</b>
<b>Meaning:</b> The battery is being charged. Or the battery has not yet been checked since the controller was booted. Or the charging current has not yet dropped sufficiently.
<b>Action required by the user:</b> Do not exchange the battery.
<b>Info to PLC:</b> Supply voltage disconnection can cause errors in Hibernate mode.
<b>Message:</b> No message
<b>#CHARGE_TEST_NOK</b>
<b>Meaning:</b> The result of the battery test was negative.
<b>Action required by the user:</b> Exchange the battery.
<b>Info to PLC:</b> Supply voltage disconnection can cause errors in Hibernate mode.
<b>Message:</b> Battery defective - load test failed
<b>#CHARGE_NOK</b>
<b>Meaning:</b> No battery test possible. The battery is not fully charged after the maximum charging time.
<b>Action required by the user:</b> Exchange the battery.
<b>Info to PLC:</b> Supply voltage disconnection can cause errors in the case of a warm start.
<b>Message:</b> Battery defective - reliable backup cannot be assured
<b>#CHARGE_OFF</b>
<b>Meaning:</b> There is no battery present or the battery is defective.
<b>Action required by the user:</b> Exchange the battery.
<b>Info to PLC:</b> Supply voltage disconnection can cause errors in the case of a warm start.
<b>Message:</b> Battery defective - backup not possible



## 5 Start-up and recommissioning

### 5.1 Start-up wizard

**Description** Start-up can be carried out using the Start-up wizard. This guides the user through the basic start-up steps.

**Precondition**

- No program is selected.
- Operating mode T1

**Procedure**

- Select **Start-up > Start-up wizard** in the main menu.

### 5.2 Checking the machine data

**Description** The correct machine data must be loaded. This must be checked by comparing the loaded machine data with the machine data on the rating plate.

If machine data are reloaded, the version of the machine data must correspond exactly to the KSS version. This is ensured if the machine data supplied together with the KSS release are used.

**DANGER** The industrial robot must not be moved if incorrect machine data are loaded. Death, severe physical injuries or considerable damage to property may otherwise result. The correct machine data must be loaded.

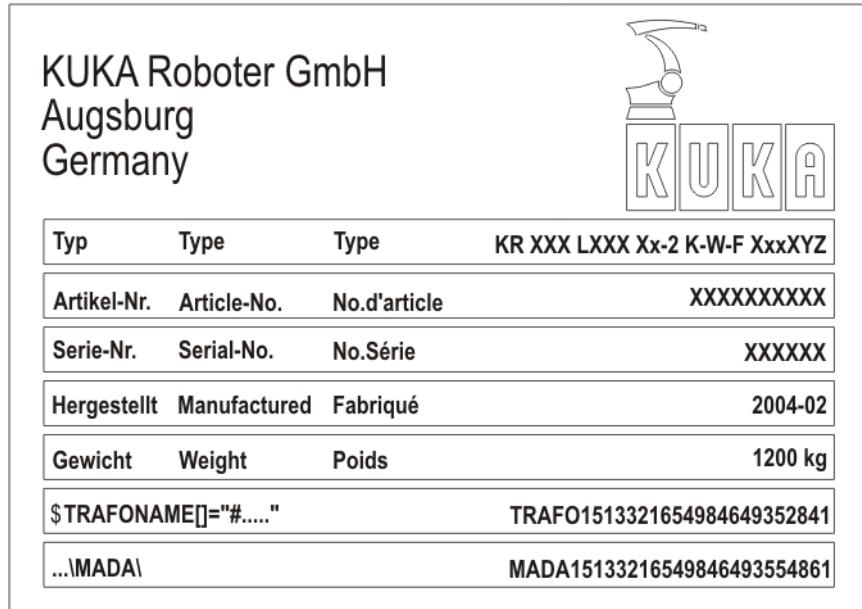


Fig. 5-1: Rating plate

**Precondition**

- T1 or T2 operating mode
- No program is selected.

**Procedure**

1. In the main menu, select **Start-up > Robot data**.  
The **Robot Data** window is opened.
2. Compare the following entries:
  - In the **Robot data** window: the entry in the **Machine data** box
  - On the rating plate on the base of the robot: the entry in the line **\$TRAFONAME()="# ....."**



The file path of the machine data on the CD is specified on the rating plate in the line ...\\MADA\\.

### 5.3 Defining hardware options

- Precondition** ■ User group “Safety recovery”
- Procedure**
1. Select **Configuration > Safety configuration** in the main menu.
  2. Press **Hardware options**.
  3. Modify hardware options and press **Save**.

**Description**

Hardware options		
Customer interface	SIB	
Input signal for peripheral contactor (US2)	not used	
Operator safety acknowledgment	external unit	
<b>Back</b>	<b>Save</b>	<b>Revert changes</b>

**Fig. 5-2: Hardware options**

Parameter	Description
Customer interface	Select here which interface is used: ■ <b>ProfiSafe</b> ■ <b>SIB</b> ■ <b>SIB, SIB extended</b> ■ <b>SIB with operating mode output</b> ■ <b>SIB with operating mode output, SIB extended</b>  This option is available with System Software version 8.2.4 or higher.

Parameter	Description
Peripheral contactor circuit (US2)	<p>Main contactor 2 can be used as a peripheral contactor, i.e. as a switching element for the power supply to peripheral devices.</p> <p><b>Deactivated:</b> Peripheral contactor is not used. (Default)</p> <p><b>By external PLC:</b> The peripheral contactor is switched by an external PLC via input US2.</p> <p><b>By KRC:</b> The peripheral contactor is switched in accordance with the motion enable. If motion enable is present, the contactor is energized.</p>
Operator safety acknowledgement	<p>If the Operator Safety signal is lost and reset in Automatic mode, it must be acknowledged before operation can be continued.</p> <p><b>By acknowledgement button:</b> Acknowledgement is given e.g. by an acknowledgement button (situated outside the cell). Acknowledgement is communicated to the safety controller. The safety controller re-enables automatic operation only after acknowledgement.</p> <p><b>External unit:</b> Acknowledgement is given by the system PLC.</p>

## 5.4 Changing the safety ID of the PROFINET device

**Description** If multiple KUKA robot controllers are operated with a single PROFIsafe master PLC, each PROFINET device must have a unique safety ID. The default ID is always 7.

**Precondition**

- User group “Safety recovery”



If one of the options KUKA.SafeOperation or KUKA.SafeRangeMonitoring is installed on the robot controller, different user groups may apply. Information can be found in the documentation for these options.

**Procedure**

1. Select **Configuration > Safety configuration** in the main menu.
2. Press **Device management**.
3. In the column **New safety ID**, press the ID to be modified and change the ID.
4. Press **Apply safety IDs**.
5. A request for confirmation is displayed, asking if the change should be saved. Confirm the request with **Yes**.
6. A message is displayed, indicating that the change has been saved. Confirm the message with **OK**.



This procedure can only be used to save changes to the safety ID. If other unsaved changes have been made elsewhere in the safety configuration, these are not saved here.

If an attempt is now made to close the safety configuration, a query is generated asking whether you wish to reject the changes or cancel the action. To save the changes, proceed as follows:

1. Cancel the action.
2. In the safety configuration, press **Save**. (If the **Save** button is not available, first go back a level by pressing **Back**.)
3. A request for confirmation is displayed, asking if all the changes should be saved. Confirm the request with **Yes**.
4. A message is displayed, indicating that the change has been saved. Confirm the message with **OK**.

All changes in the safety configuration are saved.

## 5.5 Jogging the robot without a higher-level safety controller

### Description

To jog the robot without a higher-level safety controller, Start-up mode must first be activated. The robot can then be jogged in T1 mode. If SafeOperation is used, the robot can also be jogged in CRR.

If the RoboTeam option is used, it is only possible to activate Start-up mode and jog the robot using the local smartPAD.



External safeguards are disabled in Start-up mode. Observe the safety instructions relating to Start-up mode.  
(>>> 3.8.3.1 "Start-up mode" Page 33)

The robot controller automatically deactivates Start-up mode in the following cases:

- If no operator action has been carried out within 30 min of activation.
- If the smartPAD is switched to passive mode or disconnected from the robot controller.
- If the PROFIsafe interface is used: when a connection to a higher-level safety controller is established.
- If an SIB interface is used: when some input signals no longer have the state "logic zero".

In Start-up mode, the system switches to the following simulated input image:

- The external EMERGENCY STOP is not active.
- The safety gate is open.
- No safety stop 1 has been requested.
- No safety stop 2 has been requested.
- No safe operational stop has been requested.
- Only for VKR C4: E2 is closed.

If SafeOperation or SafeRangeMonitoring is used, Start-up mode also influences other signals.



Information about the effects of Start-up mode in conjunction with SafeOperation or SafeRangeMonitoring can be found in the documentation **SafeOperation** and **SafeRangeMonitoring**.

### Precondition

- If the PROFIsafe interface is used: No connection to a higher-level safety controller

- If an SIB interface is used: all input signals have the state “logic zero”.
- Operating mode T1 or CRR
- In the case of VKR C4: no E2/E7 signals are activated via a USB stick or retrofit interface.
- In the case of RoboTeam: the local smartPAD is used.

**Procedure**

- In the main menu, select **Start-up > Service > Start-up mode**.

Menu	Description
 Start-up mode	Start-up mode is active. Touching the menu item deactivates the mode.
 Start-up mode	Start-up mode is not active. Touching the menu item activates the mode.

**5.6 Checking the activation of the positionally accurate robot model****Description**

If a positionally accurate robot is used, it must be checked that the positionally accurate robot model is activated.

In the case of positionally accurate robots, position deviations resulting from workpiece tolerances and elastic effects of the individual robots are compensated for. The positionally accurate robot positions the programmed TCP anywhere in the Cartesian workspace within the tolerance limits. The model parameters of the positionally accurate robot are determined at a calibration station and permanently saved on the robot (RDC).



The positionally accurate robot model is only valid for the robot as delivered.

Following conversion or retrofitting of the robot, e.g. with an arm extension or a new wrist, the robot must be recalibrated.

**Functions**

A positionally accurate robot has the following functions:

- Increased positioning accuracy, approximately by the factor 10
- Increased path accuracy



A precondition for the increased positioning and path accuracy is the correct input of the load data into the robot controller.

- Simplified transfer of programs if the robot is exchanged (no reteaching)
- Simplified transfer of programs after offline programming with WorkVisual (no reteaching)

**Procedure**

1. In the main menu, select **Help > Info**.
2. Check on the **Robot** tab that the positionally accurate robot model is activated. (= specification **Positionally accurate robot**).

**5.7 Activating palletizing mode****Description**

Only relevant for palletizing robots with 6 axes!

In the case of palletizing robots with 6 axes, palletizing mode is deactivated by default and must be activated. When palletizing mode is active, A4 is locked at 0° and the mounting flange is parallel to the floor.

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ The robot is mastered.</li><li>■ There is no load on the robot; i.e. there is no tool, workpiece or supplementary load mounted.</li></ul>
<b>Procedure</b>	<ul style="list-style-type: none"><li>■ Activate palletizing mode in the program as follows: <pre>\$PAL_MODE = TRUE</pre></li></ul>
<b>Alternative procedure</b>	<ol style="list-style-type: none"><li>1. Set \$PAL_MODE to TRUE via the variable correction function.</li><li>2. The following message is displayed: <i>Palletizing mode: Move axis A4 [direction] into position.</i> Move A4 in the direction specified in the message [+/-].</li><li>3. Once A4 has reached its position, the following message is displayed: <i>Palletizing mode: Move axis A5 [direction] into position.</i> Move A5 in the direction specified in the message [+/-].</li></ol>
<b>Restrictions</b>	<ul style="list-style-type: none"><li>■ After every cold restart of the robot controller, \$PAL_MODE is automatically set to FALSE.</li></ul> <p><b>i</b> Recommendation: Integrate \$PAL_MODE = TRUE into the initialization section of all programs for the palletizing robot.</p> <ul style="list-style-type: none"><li>■ In the case of robots with palletizing mode active, payload determination with KUKA.LoadDataDetermination is not possible.</li></ul> <p><b>⚠ WARNING</b> In the case of robots with palletizing mode active, payload determination with KUKA.LoadDataDetermination must not be carried out. Physical injuries or damage to property may result.</p> <ul style="list-style-type: none"><li>■ If palletizing mode is active, the robot cannot be mastered. If mastering is nonetheless required, proceed as follows:<ol style="list-style-type: none"><li>a. Remove all loads from the robot.</li><li>b. Set \$PAL_MODE to FALSE via the variable correction function.</li><li>c. Master the robot.</li><li>d. Set \$PAL_MODE to TRUE. (Not necessary if \$PAL_MODE = TRUE is in the initialization section of all programs for the palletizing robot.)</li><li>e. Move the robot to the palletizing position.</li><li>f. Re-attach all loads to the robot.</li></ol></li></ul>

## 5.8 Copying machine data

<b>Description</b>	This function can be used to copy machine data to the robot controller, e.g. if a robot has been exchanged.
	<p><b>⚠ WARNING</b> If machine data are reloaded, the version of the machine data must correspond exactly to the KUKA System Software (KSS) version. This is ensured if only machine data supplied together with the KSS release used (e.g. on a USB stick) are loaded. The robot must not be moved if incorrect machine data are loaded. Personal injuries or damage to property may result in this case.</p>

**WARNING**

Following modifications to the machine data, all safety measures specified in the “Safety” chapter of the operating and programming instructions must be carried out. This includes at least the following safety measures:

1. The safety configuration must be updated.
2. Reduced velocity control must be tested.
3. All programs must be tested in Manual Reduced Velocity mode (T1).

Failure to observe this may result in death to persons, severe physical injuries or damage to property.

**Precondition**

- The System Software release is available on the local drive or network.
- No program is selected.
- T1 or T2 operating mode
- “Expert” user group

**Procedure**

1. In the main menu, select **Start-up > Copy machine data**. A directory structure is displayed.
2. Navigate to the System Software release.
3. Navigate to the desired machine data within the release and select the directory.
4. Press **Copy**. The machine data are copied. The message *Reconfiguration in progress ...* is displayed.

The copying operation is successfully completed when this message is no longer displayed and the message *Copying of machine data completed* is displayed.

The following icons are displayed in the directory structure:

Icon	Description
	Red check mark All directories containing valid data that can be copied are labeled with a red check mark.
	Green arrow If a directory is labeled with a red check mark, the icon changes to a green arrow. Furthermore, the complete path of the selected directory is displayed underneath the directory structure.  This directory can now be copied using the <b>Copy</b> button.

The following buttons are available:

Button	Description
<b>Refresh</b>	Refreshes the directory structure.
<b>Copy</b>	Copies the selected directory.  Precondition: the selected directory is labeled with the green arrow.

**5.9 Mastering****Overview**

Every robot must be mastered. Only if the robot has been mastered can it move to programmed positions and be moved using Cartesian coordinates. During mastering, the mechanical position and the electronic position of the robot are aligned. For this purpose, the robot is moved to a defined mechanical

position, the mastering position. The encoder value for each axis is then saved.

The mastering position is similar, but not identical, for all robots. The exact positions may even vary between individual robots of a single robot type.



**Fig. 5-3: Mastering position – approximate position**

A robot must be mastered in the following cases:

Case	Comments
During commissioning	- - -
After maintenance work during which the robot loses its mastering, e.g. exchange of motor or RDC	(>>> 5.9.6 "Reference mastering" Page 98)
When the robot has been moved without the robot controller (e.g. with the release device)	- - -
After exchanging a gear unit	Before carrying out a new mastering procedure, the old mastering data must first be deleted! Mastering data are deleted by manually unmastering the axes.
After an impact with an end stop at more than 250 mm/s	(>>> 5.9.7 "Manually unmastering axes" Page 99)
After a collision	

### 5.9.1 Mastering methods

#### Overview

A robot can be mastered in the following ways:

- With the EMD (Electronic Mastering Device)  
(>>> 5.9.3 "Mastering with the EMD" Page 92)

- With the dial gauge  
(>>> 5.9.4 "Mastering with the dial gauge" Page 97)

The axes must be moved to the pre-mastering position before every mastering operation.



EMD mastering is recommended.

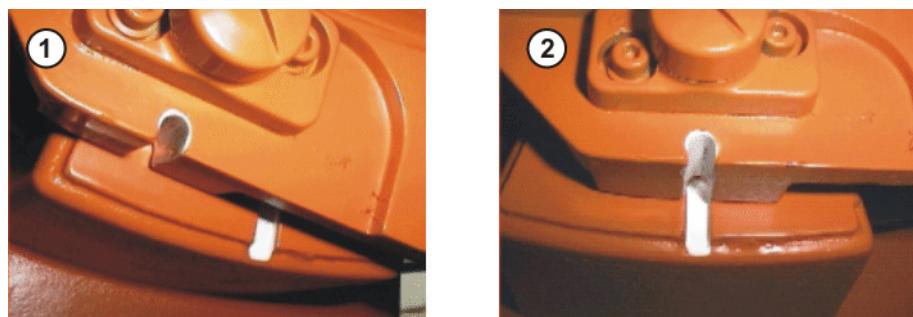
Another method is "reference mastering". It is only used for mastering the robot after certain maintenance tasks.

(>>> 5.9.6 "Reference mastering" Page 98)

### 5.9.2 Moving axes to the pre-mastering position

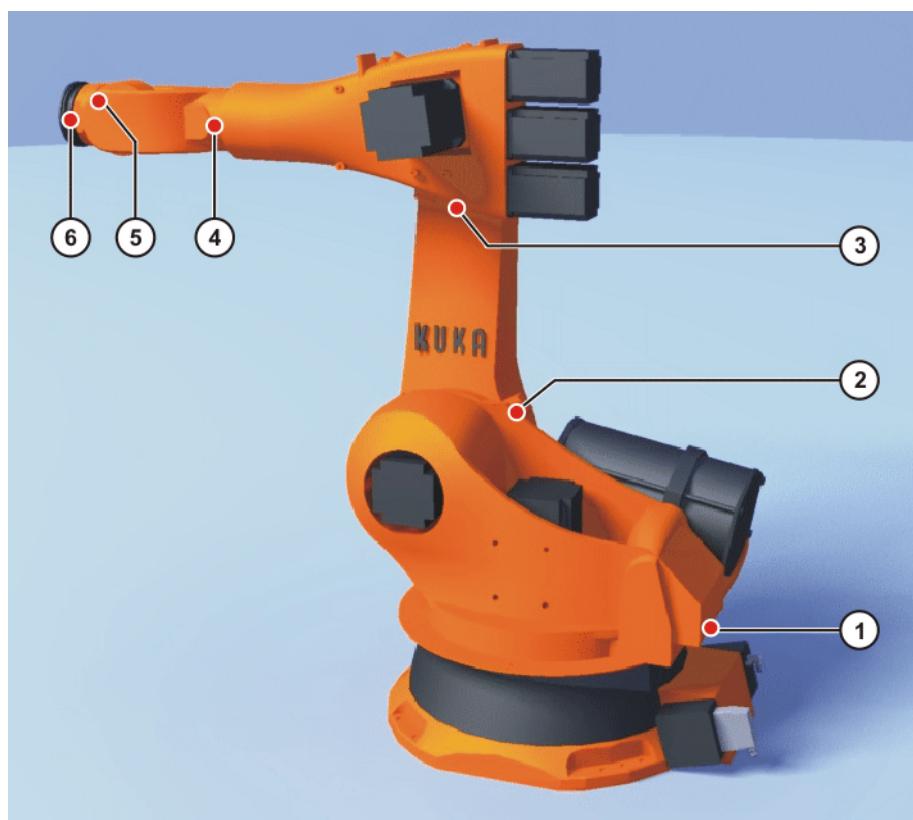
#### Description

Each axis is moved so that the mastering marks line up.



**Fig. 5-4: Moving an axis to the pre-mastering position**

The mastering marks are situated in the following positions on the robot:



**Fig. 5-5: Mastering marks on the robot**



Depending on the specific robot model, the positions of the mastering marks may deviate slightly from those illustrated.

#### Precondition

- The jog mode “Jog keys” is active.
- Operating mode T1

#### Procedure

1. Select **Axes** as the coordinate system for the jog keys.
2. Hold down the enabling switch.  
Axes A1 to A6 are displayed next to the jog keys.
3. Press the Plus or Minus jog key to move an axis in the positive or negative direction.
4. Move each axis, starting from A1 and working upwards, so that the mastering marks line up.

**NOTICE**

If A4 and A6 are moved to the pre-mastering position, ensure that the energy supply system – if present – is in its correct position and not rotated through 360°.

### 5.9.3 Mastering with the EMD

#### Overview

In EMD mastering, the axis is automatically moved by the robot controller to the mastering position. Mastering is carried out first without and then with a load. It is possible to save mastering data for different loads.

EMD mastering consists of the following steps:

Step	Description
1	<p><b>First mastering</b>            (&gt;&gt;&gt; 5.9.3.1 "First mastering with the EMD" Page 92)</p> <p>First mastering is carried out without a load.</p>
2	<p><b>Teach offset</b>            (&gt;&gt;&gt; 5.9.3.2 "Teach offset" Page 95)</p> <p>“Teach offset” is carried out with a load. The difference from the first mastering is saved.</p>
3	<p>If required: <b>Master load with offset</b>            (&gt;&gt;&gt; 5.9.3.3 "Master load with offset" Page 96)</p> <p>“Load mastering with offset” is carried out with a load for which an offset has already been taught.</p> <p>Area of application:</p> <ul style="list-style-type: none"> <li>■ Checking first mastering</li> <li>■ Restoring first mastering if it has been lost (e.g. following exchange of motor or collision). Since an offset that has been taught is retained, even if mastering is lost, the robot controller can calculate the first mastering.</li> </ul>

#### 5.9.3.1 First mastering with the EMD

#### Precondition

- There is no load on the robot; i.e. there is no tool, workpiece or supplementary load mounted.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

## Procedure

**NOTICE**

The EMD must always be screwed onto the gauge cartridge without the signal cable attached. Only then may the signal cable be attached to the EMD. Otherwise, the signal cable could be damaged.

Similarly, when removing the EMD, the signal cable must always be removed from the EMD first. Only then may the EMD be removed from the gauge cartridge.

After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. Select **Start-up > Master > EMD > With load correction > First mastering** in the main menu.

A window opens. All axes to be mastered are displayed. The axis with the lowest number is highlighted.

2. Remove the cover from connection X32.



**Fig. 5-6: Removing cover from X32**

3. Connect the signal cable to X32.



**Fig. 5-7: Connecting signal cable to X32**

4. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. (Turned around, the EMD can be used as a screwdriver.)



**Fig. 5-8: Removing protective cap from gauge cartridge**

5. Screw the EMD onto the gauge cartridge.



**Fig. 5-9: Screwing EMD onto gauge cartridge**

6. Attach the signal cable to the EMD, aligning the red dot on the connector with the groove in the EMD.



**Fig. 5-10: Attaching signal cable to EMD**

7. Press **Master**.
8. Press an enabling switch and the Start key.

When the EMD has passed through the reference notch, the mastering position is calculated. The robot stops automatically. The values are saved. The axis is no longer displayed in the window.

9. Remove the signal cable from the EMD. Then remove the EMD from the gauge cartridge and replace the protective cap.
10. Repeat steps 4 to 9 for all axes to be mastered.
11. Close the window.
12. Remove signal cable from connection X32.

### 5.9.3.2 Teach offset

#### Description

“Teach offset” is carried out with a load. The difference from the first mastering is saved.

If the robot is operated with different loads, “Teach offset” must be carried out for every load. In the case of grippers used for picking up heavy workpieces, “Teach offset” must be carried out for the gripper both with and without the workpiece.

#### Precondition

- Same ambient conditions (temperature, etc.) as for first mastering.
- The load is mounted on the robot.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

#### Procedure

**NOTICE**

The EMD must always be screwed onto the gauge cartridge without the signal cable attached. Only then may the signal cable be attached to the EMD. Otherwise, the signal cable could be damaged.

Similarly, when removing the EMD, the signal cable must always be removed from the EMD first. Only then may the EMD be removed from the gauge cartridge.

After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. Select **Start-up > Master > EMD > With load correction > Teach offset** in the main menu.
2. Enter tool number. Confirm with **Tool OK**.

A window opens. All axes for which the tool has not yet been taught are displayed. The axis with the lowest number is highlighted.

3. Remove the cover from connection X32 and connect the signal cable.
4. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. (Turned around, the EMD can be used as a screwdriver.)
5. Screw the EMD onto the gauge cartridge.
6. Attach the signal cable to the EMD, aligning the red dot on the connector with the groove in the EMD.
7. Press **Learn**.
8. Press an enabling switch and the Start key.

When the EMD has passed through the reference notch, the mastering position is calculated. The robot stops automatically. A window opens. The deviation of this axis from the first mastering is indicated in degrees and increments.

9. Click **OK** to confirm. The axis is no longer displayed in the window.
10. Remove the signal cable from the EMD. Then remove the EMD from the gauge cartridge and replace the protective cap.
11. Repeat steps 4 to 10 for all axes to be mastered.

12. Close the window.
13. Remove signal cable from connection X32.

### 5.9.3.3 Master load with offset

#### Description

Area of application:

- Checking first mastering
- Restoring first mastering if it has been lost (e.g. following exchange of motor or collision). Since an offset that has been taught is retained, even if mastering is lost, the robot controller can calculate the first mastering.



An axis can only be checked if all axes with lower numbers have been mastered.

#### Precondition

- Same ambient conditions (temperature, etc.) as for first mastering.
- A load for which "Teach offset" has been carried out is mounted on the robot.
- All axes are in the pre-mastering position.
- No program is selected.
- Operating mode T1

#### Procedure

##### NOTICE

The EMD must always be screwed onto the gauge cartridge without the signal cable attached. Only then may the signal cable be attached to the EMD. Otherwise, the signal cable could be damaged.

Similarly, when removing the EMD, the signal cable must always be removed from the EMD first. Only then may the EMD be removed from the gauge cartridge.

After mastering, remove the signal cable from connection X32. Failure to do so may result in interference signals or damage.

1. In the main menu, select **Start-up > Master > EMD > With load correction > Master load > With offset**.
2. Enter tool number. Confirm with **Tool OK**.  
A window opens. All axes for which an offset has been taught with this tool are displayed. The axis with the lowest number is highlighted.
3. Remove the cover from connection X32 and connect the signal cable.
4. Remove the protective cap of the gauge cartridge on the axis highlighted in the window. (Turned around, the EMD can be used as a screwdriver.)
5. Screw the EMD onto the gauge cartridge.
6. Attach the signal cable to the EMD, aligning the red dot on the connector with the groove in the EMD.
7. Press **Check**.
8. Hold down an enabling switch and press the Start key.  
When the EMD has passed through the reference notch, the mastering position is calculated. The robot stops automatically. The difference from "Teach offset" is displayed.
9. If required, press **Save** to save the values. The old mastering values are deleted.  
To restore a lost first mastering, always save the values.



Axes A4, A5 and A6 are mechanically coupled. This means:  
If the values for A4 are deleted, the values for A5 and A6 are also deleted.

If the values for A5 are deleted, the values for A6 are also deleted.

10. Remove the signal cable from the EMD. Then remove the EMD from the gauge cartridge and replace the protective cap.
11. Repeat steps 4 to 10 for all axes to be mastered.
12. Close the window.
13. Remove signal cable from connection X32.

#### **5.9.4 Mastering with the dial gauge**

**Description**

In dial mastering, the axis is moved manually by the user to the mastering position. Mastering is always carried out with a load. It is not possible to save mastering data for different loads.



**Fig. 5-11: Dial gauge**

**Precondition**

- The load is mounted on the robot.
- All axes are in the pre-mastering position.
- The jog mode “Jog keys” is active and the coordinate system **Axis** has been selected.
- No program is selected.
- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Master > Dial**.  
A window opens. All axes that have not been mastered are displayed. The axis that must be mastered first is selected.
2. Remove the protective cap from the gauge cartridge on this axis and mount the dial gauge on the gauge cartridge.  
Using the Allen key, loosen the screws on the neck of the dial gauge. Turn the dial so that it can be viewed easily. Push the pin of the dial gauge in as far as the stop.  
Using the Allen key, tighten the screws on the neck of the dial gauge.
3. Reduce jog override to 1%.
4. Jog axis from “+” to “-”. At the lowest position of the reference notch, recognizable by the change in direction of the pointer, set the dial gauge to 0.  
If the axis inadvertently overshoots the lowest position, jog the axis backwards and forwards until the lowest position is reached. It is immaterial whether the axis is moved from “+” to “-” or from “-” to “+”.
5. Move the axis back to the pre-mastering position.
6. Move the axis from “+” to “-” until the pointer is about 5-10 scale divisions before zero.
7. Switch to incremental jogging.

8. Move the axis from “+” to “-” until zero is reached.

**i** If the axis overshoots zero, repeat steps 5 to 8.

9. Press **Master**. The axis that has been mastered is removed from the window.
10. Remove the dial gauge from the gauge cartridge and replace the protective cap.
11. Switch back from incremental jogging to the normal jog mode.
12. Repeat steps 2 to 11 for all axes to be mastered.
13. Close the window.

### 5.9.5 Mastering external axes

#### Description

- KUKA external axes can be mastered using either the EMD or the dial gauge.
- Non-KUKA external axes can be mastered using the dial gauge. If mastering with the EMD is desired, the external axis must be fitted with gauge cartridges.

#### Procedure

- The procedure for mastering external axes is the same as that for mastering robot axes. Alongside the robot axes, the configured external axes now also appear in the axis selection window.

#### First mastering.

Master the following axes with the electronic measure tool EMD.  
Mastering must be made without load:

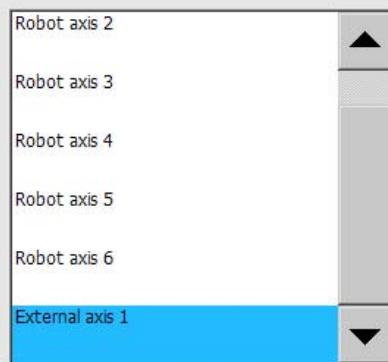


Fig. 5-12: Selection list of axes to be mastered

**i** Mastering in the case of industrial robots with more than 2 external axes: if the system contains more than 8 axes, it may be necessary to connect the signal cable of the EMD to the second RDC.

### 5.9.6 Reference mastering



Reference mastering must not be used when commissioning the robot.

<b>Description</b>	<p>Reference mastering is suitable if maintenance work is due on a correctly mastered robot and it is to be expected that the robot will lose its mastering. Examples:</p> <ul style="list-style-type: none"> <li>■ Exchange of RDC</li> <li>■ Exchange of motor</li> </ul> <p>The robot is moved to the \$MAMES position before the maintenance work is commenced. Afterwards, the axis values of this system variable are reassigned to the robot by means of reference mastering. The state of the robot is then the same as before the loss of mastering. Taught offsets are retained. No EMD or dial gauge is required.</p> <p>In the case of reference mastering, it is irrelevant whether or not there is a load mounted on the robot. Reference mastering can also be used for external axes.</p>
<b>Preparation</b>	<ul style="list-style-type: none"> <li>■ Move the robot to the \$MAMES position before commencing the maintenance work. To do so, program a point PTP \$MAMES and move the robot to it. This is only possible in the user group "Expert"!</li> </ul> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p> <b>WARNING</b> The robot must not move to the default HOME position instead of to \$MAMES. \$MAMES may be, but is not always, identical to the default HOME position. Only in the \$MAMES position will the robot be correctly mastered by means of reference mastering. If the robot is reference mastered at any position other than \$MAMES, this may result in physical injury and material damage.</p> </div>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ No program is selected.</li> <li>■ Operating mode T1</li> <li>■ The position of the robot was not changed during the maintenance work.</li> <li>■ If the RDC has been exchanged: the robot data have been transferred from the hard drive to the RDC (this can only be done in the user group "Expert"!)</li> </ul> <p>(&gt;&gt;&gt; 4.15.14 "Displaying/editing robot data" Page 78)</p>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Start-up &gt; Master &gt; Reference</b>. The option window <b>Reference mastering</b> is opened. All axes that have not been mastered are displayed. The axis that must be mastered first is selected.</li> <li>2. Press <b>Master</b>. The selected axis is mastered and removed from the option window.</li> <li>3. Repeat step 2 for all axes to be mastered.</li> </ol>
<b>5.9.7 Manually unmastering axes</b>	
<b>Description</b>	<p>The mastering values of the individual axes can be deleted. The axes do not move during unmastering.</p> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p> Axes A4, A5 and A6 are mechanically coupled. This means: If the values for A4 are deleted, the values for A5 and A6 are also deleted. If the values for A5 are deleted, the values for A6 are also deleted.</p> </div> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p><b>NOTICE</b> The software limit switches of an unmastered robot are deactivated. The robot can hit the end stop buffers, thus damaging the robot and making it necessary to exchange the buffers. An unmastered robot must not be jogged, if at all avoidable. If it must be jogged, the jog override must be reduced as far as possible.</p> </div>

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ No program is selected.</li> <li>■ Operating mode T1</li> </ul>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Start-up &gt; Master &gt; Unmaster</b>. A window opens.</li> <li>2. Select the axis to be unmastered.</li> <li>3. Press <b>Unmaster</b>. The mastering data of the axis are deleted.</li> <li>4. Repeat steps 2 and 3 for all axes to be unmastered.</li> <li>5. Close the window.</li> </ol>

## 5.10 Modifying software limit switches

There are 2 ways of modifying the software limit switches:

- Enter the desired values manually.
- Or automatically adapt the limit switches to one or more programs.

The robot controller determines the minimum and maximum axis positions occurring in the program. These values can then be set as software limit switches.

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ “Expert” user group</li> <li>■ T1, T2 or AUT mode</li> </ul>
---------------------	---

### Procedure

#### Modifying software limit switches manually:

1. In the main menu, select **Start-up > Service > Software limit switch**. The **Software limit switch** window is opened.
2. Modify the limit switches as required in the columns **Negative** and **Positive**.
3. Save the changes with **Save**.

#### Adapting software limit switches to a program:

1. In the main menu, select **Start-up > Service > Software limit switch**. The **Software limit switch** window is opened.
2. Click on **Auto detection**. The following message is displayed: **Auto detection is running**.
3. Start the program to which the limit switches are to be adapted. Execute the program completely and then cancel it.  
The maximum and minimum position reached by each axis is displayed in the **Software limit switch** window.
4. Repeat step 3 for all programs to which the limit switches are to be adapted.  
The maximum and minimum position reached by each axis in all executed programs is displayed in the **Software limit switch** window.
5. Once all desired programs have been executed, press **End** in the **Software limit switch** window.
6. Press **Save** to save the determined values as software limit switches.
7. If required, modify the automatically determined values manually.



Recommendation: Reduce the determined minimum values by 5°. Increase the determined maximum values by 5°.

This margin prevents the axes from reaching the limit switches during program execution and thus triggering a stop.

8. Save the changes with **Save**.

<b>Description</b>	<b>Software limit switch</b> window:
--------------------	--------------------------------------

Software limit switch			
Axis	Negative	Current position	Positive
A1 [°]	-185.00	0.00	185.00
A2 [°]	-146.00	0.00	0.00
A3 [°]	-119.00	0.00	155.00
A4 [°]	-350.00	0.00	350.00
A5 [°]	-125.00	0.00	125.00
A6 [°]	-350.00	0.00	350.00

Auto detection  
End  
Save

**Fig. 5-13: Before automatic determination**

Item	Description
1	Current negative limit switch
2	Current position of the axis
3	Current positive limit switch

Software limit switch			
Axis	Minimum	Current position	Maximum
A1 [°]	0.00	0.00	0.00
A2 [°]	0.00	0.00	0.00
A3 [°]	0.00	0.00	0.00
A4 [°]	0.00	0.00	0.00
A5 [°]	0.00	0.00	0.00
A6 [°]	0.00	0.00	0.00

Auto detection  
End  
Save

**Fig. 5-14: During automatic determination**

Item	Description
4	Minimum position of the axis since the start of determination
5	Maximum position of the axis since the start of determination

**Buttons**

The following buttons are available (only in the “Expert” user group):

Button	Description
<b>Auto detection</b>	Starts the automatic determination: The robot controller writes the minimum and maximum positions adopted by the axes from now on to the columns <b>Minimum</b> and <b>Maximum</b> in the <b>Software limit switch</b> window.
<b>End</b>	Ends the automatic determination. Transfers the calculated minimum/maximum positions to the columns <b>Negative</b> and <b>Positive</b> , but does not yet save them.
<b>Save</b>	Saves the values in the columns <b>Negative</b> and <b>Positive</b> as software limit switches.

## 5.11 Calibration

### 5.11.1 Defining the tool direction

- Description** By default, the X axis is defined in the system as the tool direction. The tool direction can be changed using the system variable \$TOOL\_DIRECTION.
- The change relates only to spline motions. For LIN and CIRC motions, the tool direction is the X axis and cannot be changed.
  - The change applies to all tools. It is not possible to define different tool directions for different tools.



**WARNING** The tool direction must be defined before calibration and before program creation. It cannot be modified subsequently. Failure to observe this may result in unexpected changes to the motion characteristics of the robot. Death to persons, severe physical injuries or considerable damage to property may result.

**Precondition**

- Expert user group

**Procedure**

- Set the system variable \$TOOL\_DIRECTION to the desired value in the file \$CUSTOM.DAT, located in the directory KRC\Steu\MaDa.  
Possible values: #X (default); #Y; #Z

It is not possible to modify \$TOOL\_DIRECTION by means of the variable correction function or by writing to the variable from the program.

### 5.11.2 Tool calibration

- Description** During tool calibration, the user assigns a Cartesian coordinate system (TOOL coordinate system) to the tool mounted on the mounting flange.

The TOOL coordinate system has its origin at a user-defined point. This is called the TCP (Tool Center Point). The TCP is generally situated at the working point of the tool.



In the case of a fixed tool, the type of calibration described here must not be used. A separate type of calibration must be used for fixed tools. ([>>> 5.11.4 "Fixed tool calibration" Page 112](#))

Advantages of tool calibration:

- The tool can be moved in a straight line in the tool direction.
- The tool can be rotated about the TCP without changing the position of the TCP.

- In program mode: The programmed velocity is maintained at the TCP along the path.

A maximum of 16 TOOL coordinate systems can be saved. Variable: TOOL\_DATA[1...16].

The following data are saved:

- X, Y, Z:  
Origin of the TOOL coordinate system relative to the FLANGE coordinate system
- A, B, C:  
Orientation of the TOOL coordinate system relative to the FLANGE coordinate system

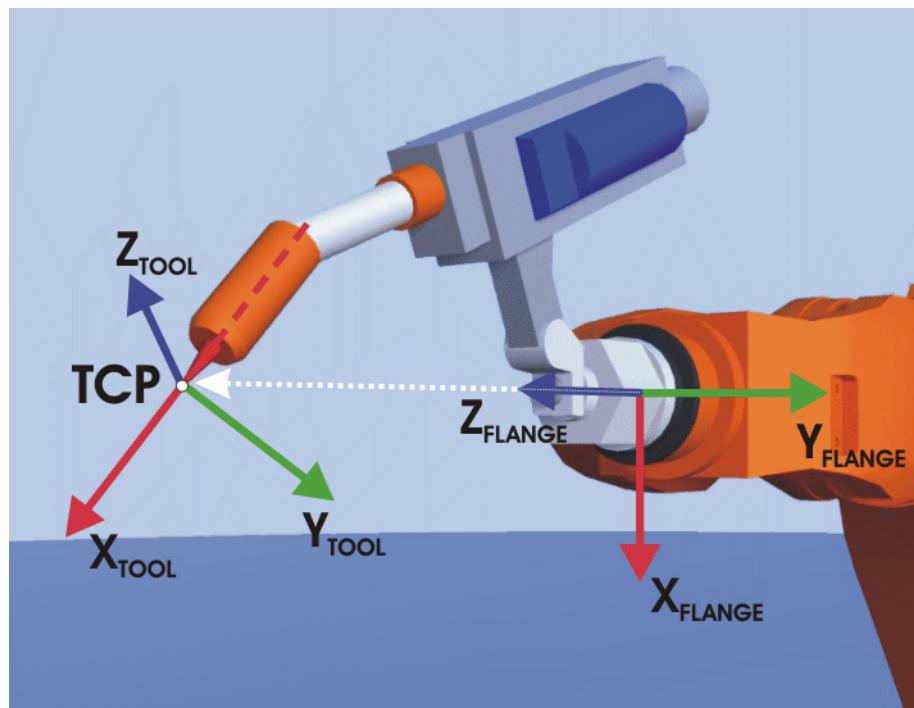


Fig. 5-15: TCP calibration principle

## Overview

Tool calibration consists of 2 steps:

Step	Description
1	<b>Definition of the origin of the TOOL coordinate system</b> The following methods are available: <ul style="list-style-type: none"><li>■ XYZ 4-point<ul style="list-style-type: none"><li>(&gt;&gt;&gt; 5.11.2.1 "TCP calibration: XYZ 4-point method" Page 104)</li></ul></li><li>■ XYZ Reference<ul style="list-style-type: none"><li>(&gt;&gt;&gt; 5.11.2.2 "TCP calibration: XYZ Reference method" Page 106)</li></ul></li></ul>
2	<b>Definition of the orientation of the TOOL coordinate system</b> The following methods are available: <ul style="list-style-type: none"><li>■ ABC 2-point<ul style="list-style-type: none"><li>(&gt;&gt;&gt; 5.11.2.4 "Defining the orientation: ABC 2-point method" Page 107)</li></ul></li><li>■ ABC World<ul style="list-style-type: none"><li>(&gt;&gt;&gt; 5.11.2.3 "Defining the orientation: ABC World method" Page 107)</li></ul></li></ul>

If the calibration data are already known, they can be entered directly.  
(>>> 5.11.2.5 "Numeric input" Page 109)

### 5.11.2.1 TCP calibration: XYZ 4-point method



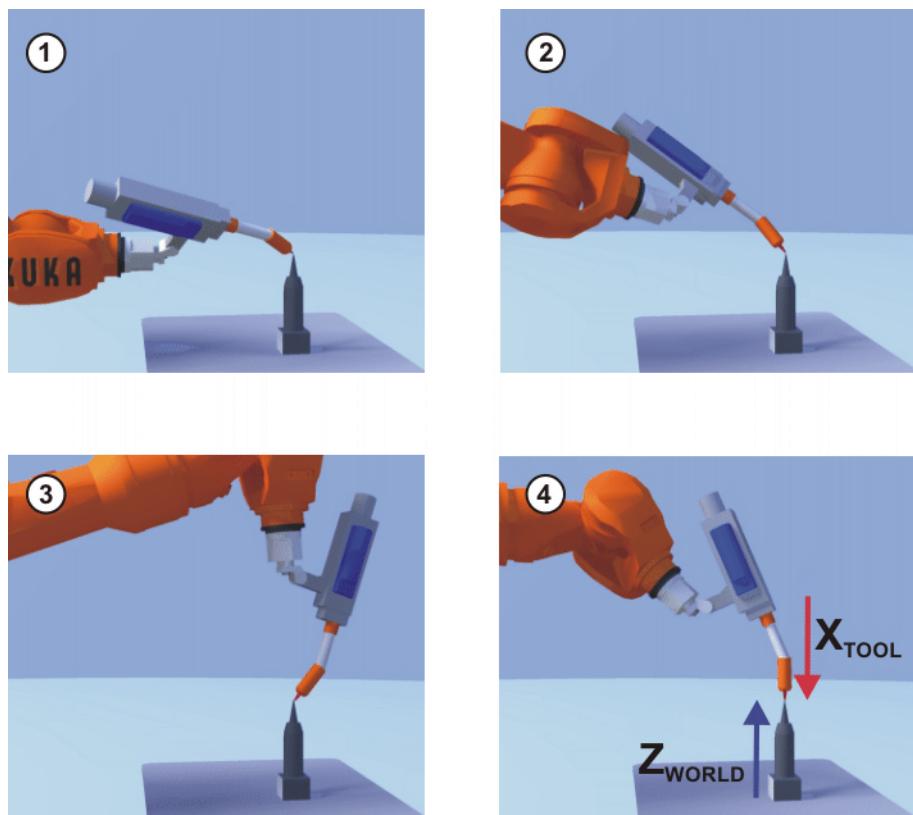
The XYZ 4-point method cannot be used for palletizing robots.

#### Description

The TCP of the tool to be calibrated is moved to a reference point from 4 different directions. The reference point can be freely selected. The robot controller calculates the TCP from the different flange positions.



The 4 flange positions at the reference point must be sufficiently different from one another.



**Fig. 5-16: XYZ 4-Point method**

#### Precondition

- The tool to be calibrated is mounted on the mounting flange.
- Operating mode T1

#### Procedure

1. In the main menu, select **Start-up > Calibrate > Tool > XYZ 4-point**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Next**.
3. Move the TCP to a reference point. Press **Calibrate**. Answer the request for confirmation with **Yes**.
4. Move the TCP to the reference point from a different direction. Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Repeat step 4 twice.
6. Enter the payload data. (This step can be skipped if the payload data are entered separately instead.)  
 (>>>> 5.12.3 "Entering payload data" Page 126)
7. Confirm with **Next**.
8. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
9. Either: press **Save** and then close the window via the **Close** icon.  
 Or: press **ABC 2-point** or **ABC World**. The previous data are automatically saved and a window is opened in which the orientation of the TOOL coordinate system can be defined.  
 (>>>> 5.11.2.4 "Defining the orientation: ABC 2-point method" Page 107)  
 (>>>> 5.11.2.3 "Defining the orientation: ABC World method" Page 107)

### 5.11.2.2 TCP calibration: XYZ Reference method

#### Description

In the case of the XYZ Reference method, a new tool is calibrated with a tool that has already been calibrated. The robot controller compares the flange positions and calculates the TCP of the new tool.

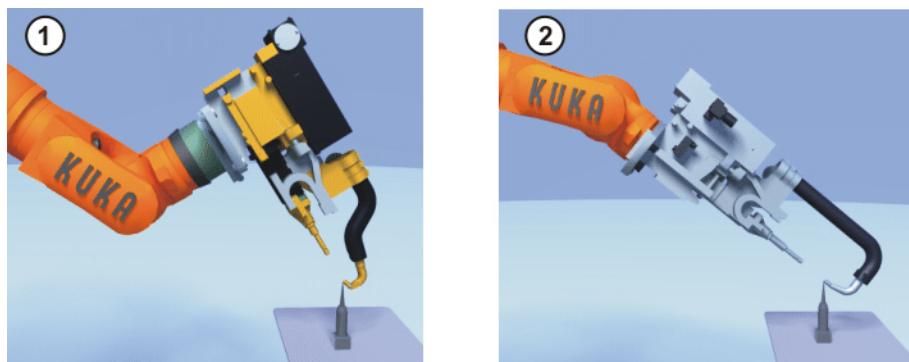


Fig. 5-17: XYZ Reference method

#### Precondition

- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1

#### Preparation

Calculate the TCP data of the calibrated tool:

1. In the main menu, select **Start-up > Calibrate > Tool > XYZ Reference**.
2. Enter the number of the calibrated tool.
3. The tool data are displayed. Note the X, Y and Z values.
4. Close the window.

#### Procedure

1. In the main menu, select **Start-up > Calibrate > Tool > XYZ Reference**.
2. Assign a number and a name for the new tool. Confirm with **Next**.
3. Enter the TCP data of the calibrated tool. Confirm with **Next**.
4. Move the TCP to a reference point. Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Move the tool away and remove it. Mount the new tool.
6. Move the TCP of the new tool to the reference point. Press **Calibrate**. Answer the request for confirmation with **Yes**.
7. Enter the payload data. (This step can be skipped if the payload data are entered separately instead.)  
(>>> 5.12.3 "Entering payload data" Page 126)
8. Confirm with **Next**.
9. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
10. Either: press **Save** and then close the window via the **Close** icon.  
Or: press **ABC 2-point** or **ABC World**. The previous data are automatically saved and a window is opened in which the orientation of the TOOL coordinate system can be defined.  
(>>> 5.11.2.4 "Defining the orientation: ABC 2-point method" Page 107)  
(>>> 5.11.2.3 "Defining the orientation: ABC World method" Page 107)

### 5.11.2.3 Defining the orientation: ABC World method

<b>Description</b>	The axes of the TOOL coordinate system are aligned parallel to the axes of the WORLD coordinate system. This communicates the orientation of the TOOL coordinate system to the robot controller.  There are 2 variants of this method:
	<ul style="list-style-type: none"> <li>■ <b>5D</b>: Only the tool direction is communicated to the robot controller. By default, the tool direction is the X axis. The directions of the other axes are defined by the system and cannot be detected easily by the user. Area of application: e.g. MIG/MAG welding, laser cutting or waterjet cutting</li> <li>■ <b>6D</b>: The directions of all 3 axes are communicated to the robot controller. Area of application: e.g. for weld guns, grippers or adhesive nozzles</li> </ul>
<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ The tool to be calibrated is mounted on the mounting flange.</li> <li>■ The TCP of the tool has already been measured.</li> <li>■ Operating mode T1</li> </ul>
	<p> The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly.  <span style="color: orange;">(&gt;&gt;&gt; 5.11.1 "Defining the tool direction" Page 102)</span></p>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; Tool &gt; ABC World</b>.</li> <li>2. Enter the number of the tool. Confirm with <b>Next</b>.</li> <li>3. Select a variant in the box <b>5D/6D</b>. Confirm with <b>Next</b>.</li> <li>4. If <b>5D</b> is selected: Align <math>+X_{TOOL}</math> parallel to <math>-Z_{WORLD}</math>. (<math>+X_{TOOL}</math> = tool direction) If <b>6D</b> is selected: Align the axes of the TOOL coordinate system as follows. <ul style="list-style-type: none"> <li>■ <math>+X_{TOOL}</math> parallel to <math>-Z_{WORLD}</math>. (<math>+X_{TOOL}</math> = tool direction)</li> <li>■ <math>+Y_{TOOL}</math> parallel to <math>+Y_{WORLD}</math></li> <li>■ <math>+Z_{TOOL}</math> parallel to <math>+X_{WORLD}</math></li> </ul> </li> <li>5. Press <b>Calibrate</b>. Answer the request for confirmation with <b>Yes</b>. <p> The following two steps are eliminated if the procedure is not called via the main menu, but by means of the <b>ABC World</b> button after TCP calibration.</p> </li> <li>6. Enter the payload data. (This step can be skipped if the payload data are entered separately instead.)  <span style="color: orange;">(&gt;&gt;&gt; 5.12.3 "Entering payload data" Page 126)</span> </li> <li>7. Confirm with <b>Next</b>.</li> <li>8. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press <b>Meas. points</b>. Then return to the previous view by pressing <b>Back</b>.</li> <li>9. Press <b>Save</b>.</li> </ol>

### 5.11.2.4 Defining the orientation: ABC 2-point method

<b>Description</b>	The axes of the TOOL coordinate system are communicated to the robot controller by moving to a point on the X axis and a point in the XY plane.  This method is used if it is necessary to define the axis directions with particular precision.
--------------------	--

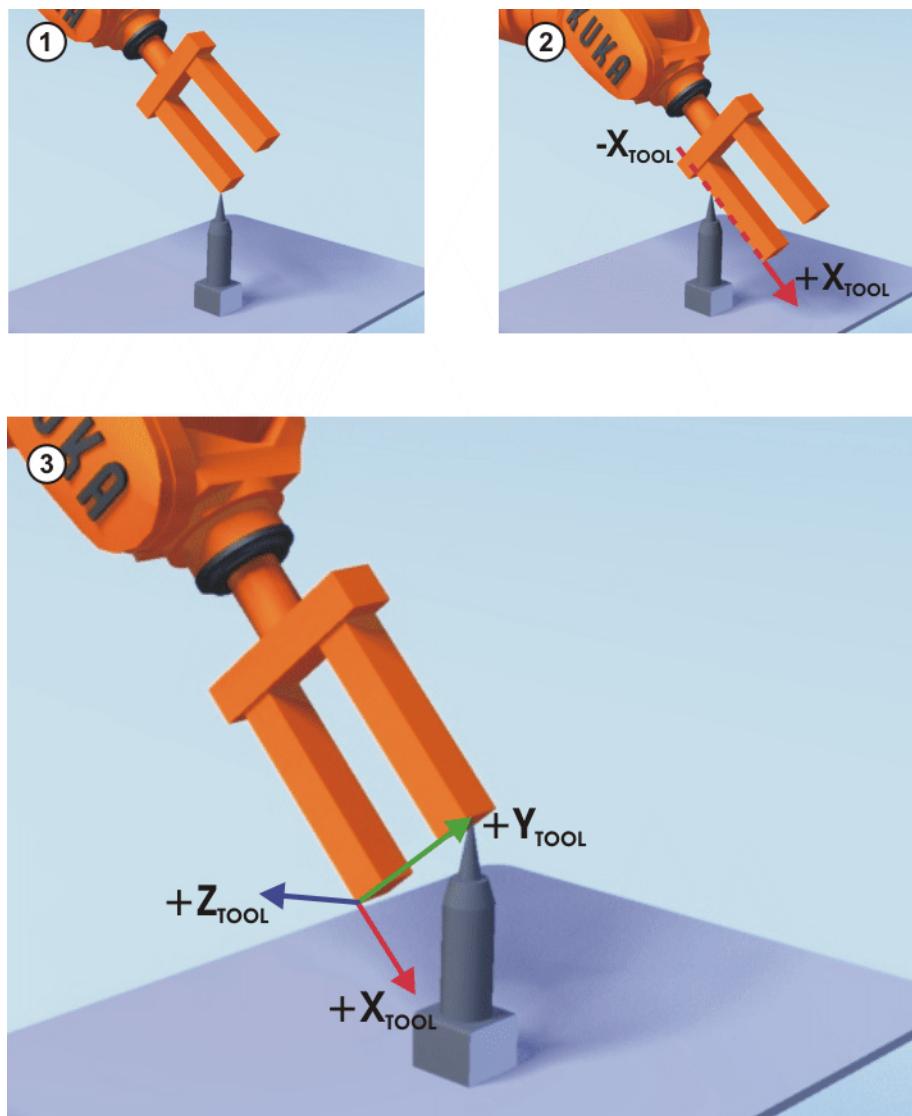


Fig. 5-18: ABC 2-Point method

#### Precondition

- The tool to be calibrated is mounted on the mounting flange.
- The TCP of the tool has already been measured.
- Operating mode T1



The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly.  
(>>> 5.11.1 "Defining the tool direction" Page 102)

#### Procedure

1. In the main menu, select **Start-up > Calibrate > Tool > ABC 2-point**.
2. Enter the number of the mounted tool. Confirm with **Next**.
3. Move the TCP to any reference point. Press **Calibrate**. Answer the request for confirmation with **Yes**.
4. Move the tool so that the reference point on the X axis has a negative X value (i.e. move against the tool direction). Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Move the tool so that the reference point in the XY plane has a negative Y value. Press **Calibrate**. Answer the request for confirmation with **Yes**.



The following two steps are eliminated if the procedure is not called via the main menu, but by means of the **ABC 2-point** button after TCP calibration.

6. Enter the payload data. (This step can be skipped if the payload data are entered separately instead.)  
(>>> 5.12.3 "Entering payload data" Page 126)
7. Confirm with **Next**.
8. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
9. Press **Save**.

#### 5.11.2.5 Numeric input

**Description** The tool data can be entered manually.

Possible sources of data:

- CAD
- Externally calibrated tool
- Tool manufacturer specifications



In the case of palletizing robots with 4 axes, e.g. KR 180 PA, the tool data must be entered numerically. The XYZ and ABC methods cannot be used as reorientation of these robots is highly restricted.

**Precondition**

- The following values are known:
  - X, Y and Z relative to the FLANGE coordinate system
  - A, B and C relative to the FLANGE coordinate system
- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Tool > Numeric input**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Next**.
3. Enter the tool data. Confirm with **Next**.
4. Enter the payload data. (This step can be skipped if the payload data are entered separately instead.)  
(>>> 5.12.3 "Entering payload data" Page 126)
5. If online load data verification is available (this depends on the robot type): configure as required.  
(>>> 5.12.5 "Online load data check (OLDC)" Page 127)
6. Confirm with **Next**.
7. Press **Save**.

#### 5.11.3 Base calibration

**Description** During base calibration, the user assigns a Cartesian coordinate system (BASE coordinate system) to a work surface or the workpiece. The BASE coordinate system has its origin at a user-defined point.



If the workpiece is mounted on the mounting flange, the type of calibration described here must not be used. A separate type of calibration must be used for workpieces mounted on the mounting flange.

(>>> 5.11.4 "Fixed tool calibration" Page 112)

Advantages of base calibration:

- The TCP can be jogged along the edges of the work surface or workpiece.
- Points can be taught relative to the base. If it is necessary to offset the base, e.g. because the work surface has been offset, the points move with it and do not need to be retaught.

A maximum of 32 BASE coordinate systems can be saved. Variable: BASE\_DATA[1...32].

## Overview

There are 2 ways of calibrating a base:

- 3-point method ([>>> 5.11.3.1 "3-point method" Page 110](#))
- Indirect method ([>>> 5.11.3.2 "Indirect method" Page 111](#))

If the calibration data are already known, they can be entered directly. ([>>> 5.11.3.3 "Numeric input" Page 112](#))

### 5.11.3.1 3-point method

#### Description

The robot moves to the origin and 2 further points of the new base. These 3 points define the new base.

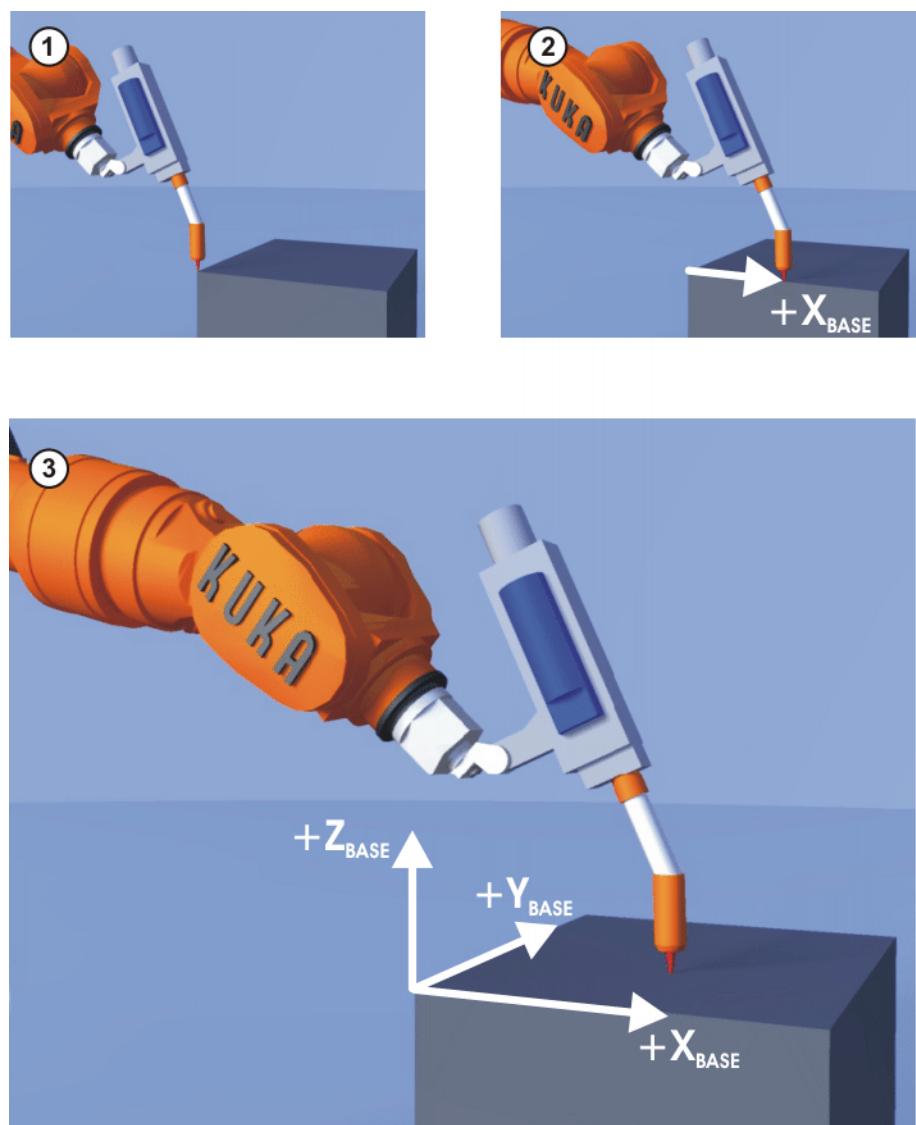


Fig. 5-19: 3-point method

#### Precondition

- A previously calibrated tool is mounted on the mounting flange.

- Operating mode T1

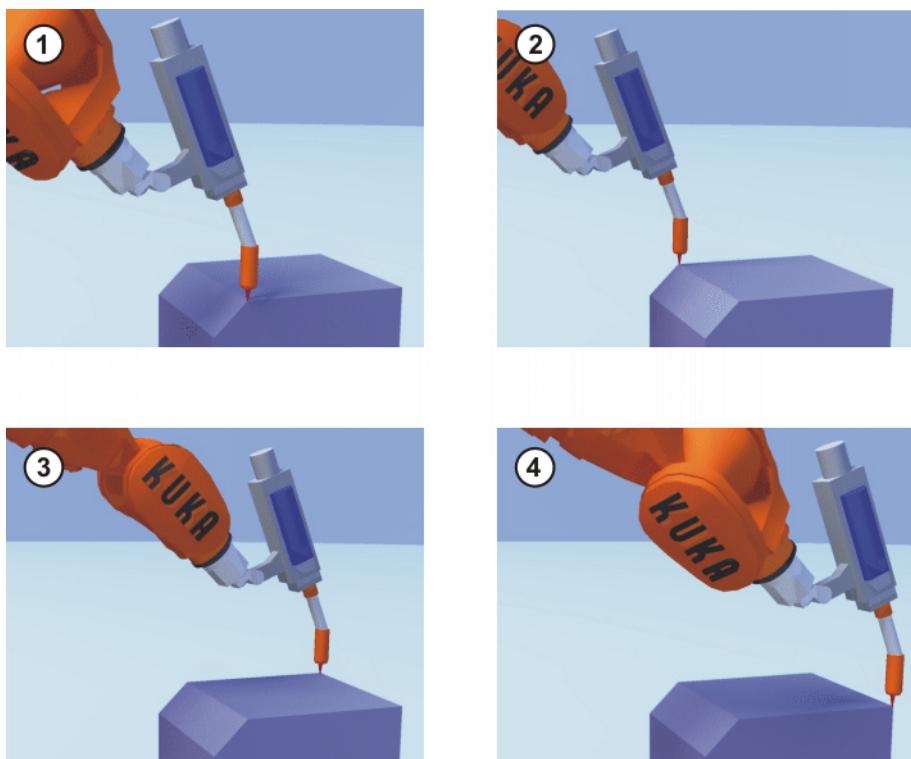
**Procedure**

1. In the main menu, select **Start-up > Calibrate > Base > ABC 3-point**.
2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter the number of the mounted tool. Confirm with **Next**.
4. Move the TCP to the origin of the new base. Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Move the TCP to a point on the positive X axis of the new base. Press **Calibrate**. Answer the request for confirmation with **Yes**.
6. Move the TCP to a point in the XY plane with a positive Y value. Press **Calibrate**. Answer the request for confirmation with **Yes**.
7. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
8. Press **Save**.

**5.11.3.2 Indirect method****Description**

The indirect method is used if it is not possible to move to the origin of the base, e.g. because it is inside a workpiece or outside the workspace of the robot.

The TCP is moved to 4 points in the base, the coordinates of which must be known. The robot controller calculates the base from these points.



**Fig. 5-20: Indirect method**

**Precondition**

- A calibrated tool is mounted on the mounting flange.
- The coordinates of 4 points in the new base are known, e.g. from CAD data. The 4 points are accessible to the TCP.
- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Base > Indirect**.

2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter the number of the mounted tool. Confirm with **Next**.
4. Enter the coordinates of a known point in the new base and move the TCP to this point. Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Repeat step 4 three times.
6. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
7. Press **Save**.

### 5.11.3.3 Numeric input

#### Precondition

- The following numerical values are known, e.g. from CAD data:  
Distance between the origin of the base and the origin of the WORLD coordinate system  
Rotation of the base axes relative to the WORLD coordinate system
- Operating mode T1

#### Procedure

1. In the main menu, select **Start-up > Calibrate > Base > Numeric input**.
2. Assign a number and a name for the base. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Press **Save**.

### 5.11.4 Fixed tool calibration

#### Overview

Calibration of a fixed tool consists of 2 steps:

Step	Description
1	<p><b>Calibration of the TCP of the fixed tool</b></p> <p>The TCP of a fixed tool is called an external TCP.  <a href="#">(&gt;&gt;&gt; 5.11.4.1 "Calibrating an external TCP" Page 113)</a></p> <p>If the calibration data are already known, they can be entered directly.  <a href="#">(&gt;&gt;&gt; 5.11.4.2 "Entering the external TCP numerically" Page 114)</a></p>
2	<p><b>Calibration of the workpiece</b></p> <p>The following methods are available:</p> <ul style="list-style-type: none"> <li>■ Direct method  <a href="#">(&gt;&gt;&gt; 5.11.4.3 "Workpiece calibration: direct method" Page 114)</a></li> <li>■ Indirect method  <a href="#">(&gt;&gt;&gt; 5.11.4.4 "Workpiece calibration: indirect method" Page 116)</a></li> </ul>

The robot controller saves the external TCP as the BASE coordinate system and the workpiece as the TOOL coordinate system. A maximum of 32 BASE coordinate systems and 16 TOOL coordinate systems can be saved.

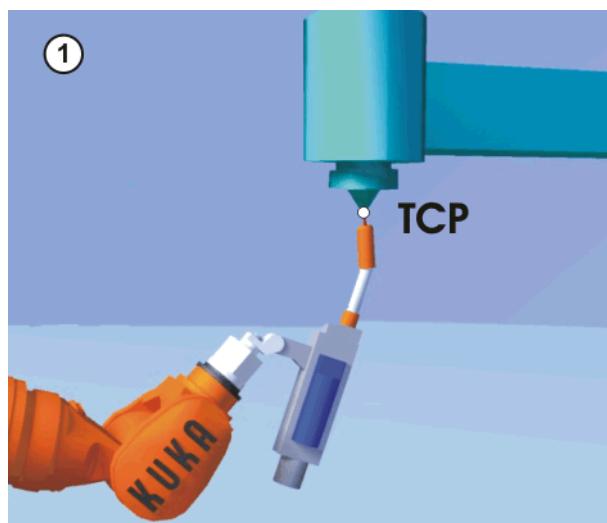
#### 5.11.4.1 Calibrating an external TCP

##### Description

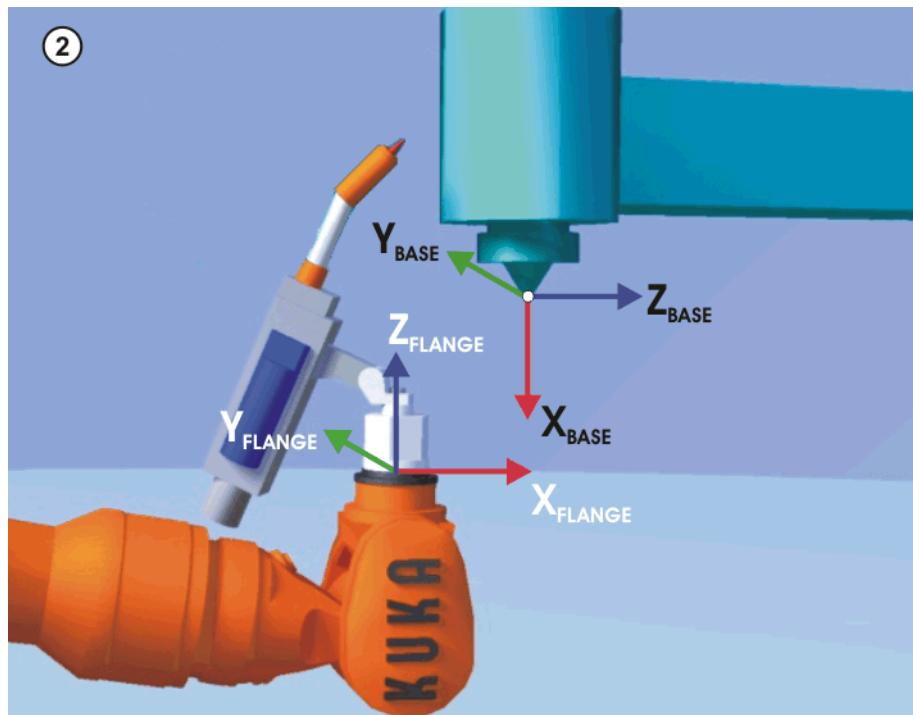
First of all, the TCP of the fixed tool is communicated to the robot controller. This is done by moving a calibrated tool to it.

Then, the orientation of the coordinate system of the fixed tool is communicated to the robot controller. For this purpose, the coordinate system of the calibrated tool is aligned parallel to the new coordinate system. There are 2 variants:

- **5D**: Only the tool direction of the fixed tool is communicated to the robot controller. By default, the tool direction is the X axis. The orientation of the other axes is defined by the system and cannot be detected easily by the user.
- **6D**: The orientation of all 3 axes is communicated to the robot controller.



**Fig. 5-21: Moving to the external TCP**



**Fig. 5-22: Aligning the coordinate systems parallel to one another**

##### Precondition

- A previously calibrated tool is mounted on the mounting flange.

- Operating mode T1



The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly.  
 (>>> 5.11.1 "Defining the tool direction" Page 102)

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Fixed tool > Tool**.
2. Assign a number and a name for the fixed tool. Confirm with **Next**.
3. Enter the number of the calibrated tool. Confirm with **Next**.
4. Select a variant in the box **5D/6D**. Confirm with **Next**.
5. Move the TCP of the calibrated tool to the TCP of the fixed tool. Press **Calibrate**. Answer the request for confirmation with **Yes**.
6. If **5D** is selected:  
 Align  $+X_{BASE}$  parallel to  $-Z_{FLANGE}$ .  
 (i.e. align the mounting flange perpendicular to the tool direction of the fixed tool.)  
 If **6D** is selected:  
 Align the mounting flange so that its axes are parallel to the axes of the fixed tool:
  - $+X_{BASE}$  parallel to  $-Z_{FLANGE}$   
 (i.e. align the mounting flange perpendicular to the tool direction.)
  - $+Y_{BASE}$  parallel to  $+Y_{FLANGE}$
  - $+Z_{BASE}$  parallel to  $+X_{FLANGE}$
7. Press **Calibrate**. Answer the request for confirmation with **Yes**.
8. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
9. Press **Save**.

**5.11.4.2 Entering the external TCP numerically****Precondition**

- The following numerical values are known, e.g. from CAD data:
  - Distance between the TCP of the fixed tool and the origin of the WORLD coordinate system (X, Y, Z)
  - Rotation of the axes of the fixed tool relative to the WORLD coordinate system (A, B, C)
- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Fixed tool > Numeric input**.
2. Assign a number and a name for the fixed tool. Confirm with **Next**.
3. Enter data. Confirm with **Next**.
4. Press **Save**.

**5.11.4.3 Workpiece calibration: direct method****Description**

The origin and 2 further points of the workpiece are communicated to the robot controller. These 3 points uniquely define the workpiece.

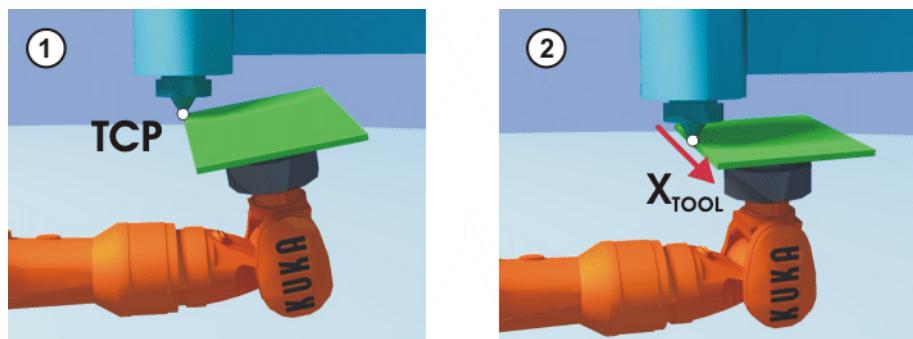


Fig. 5-23

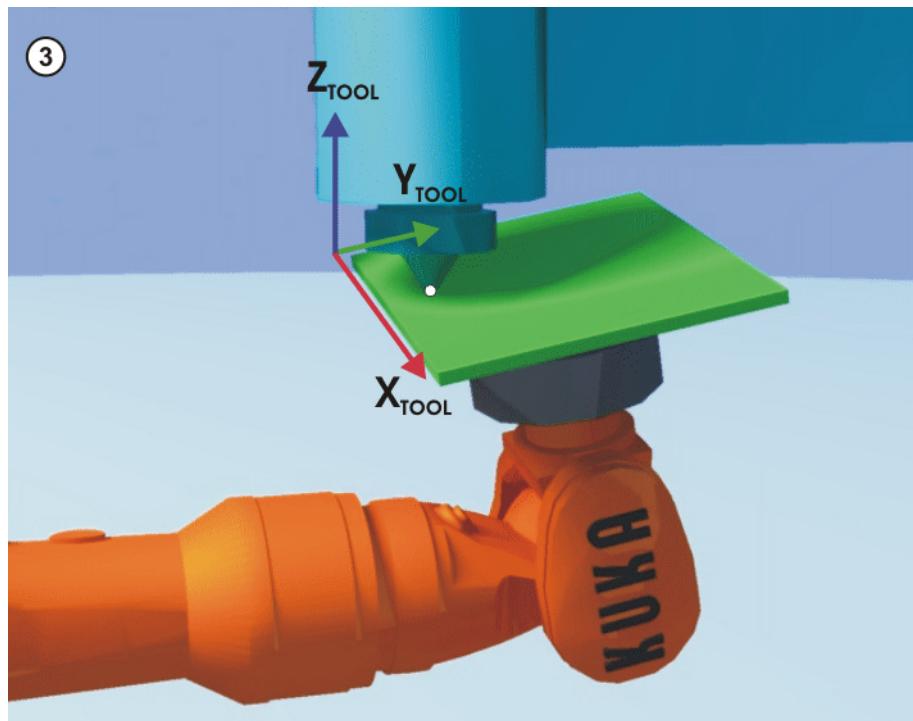


Fig. 5-24: Workpiece calibration: direct method

**Precondition**

- The workpiece is mounted on the mounting flange.
- A previously calibrated fixed tool is mounted.
- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Fixed tool > Workpiece > Direct calibration**.
2. Assign a number and a name for the workpiece. Confirm with **Next**.
3. Enter the number of the fixed tool. Confirm with **Next**.
4. Move the origin of the workpiece coordinate system to the TCP of the fixed tool. Press **Calibrate**. Answer the request for confirmation with **Yes**.
5. Move a point on the positive X axis of the workpiece coordinate system to the TCP of the fixed tool. Press **Calibrate**. Answer the request for confirmation with **Yes**.
6. Move a point with a positive Y value in the XY plane of the workpiece coordinate system to the TCP of the fixed tool. Press **Calibrate**. Answer the request for confirmation with **Yes**.
7. Enter the load data of the workpiece. (This step can be skipped if the load data are entered separately instead.)  
(>>> 5.12.3 "Entering payload data" Page 126)

8. Confirm with **Next**.
9. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate system). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.
10. Press **Save**.

#### 5.11.4.4 Workpiece calibration: indirect method

<b>Description</b>	The robot controller calculates the workpiece on the basis of 4 points whose coordinates must be known. The robot does not move to the origin of the workpiece.
--------------------	---

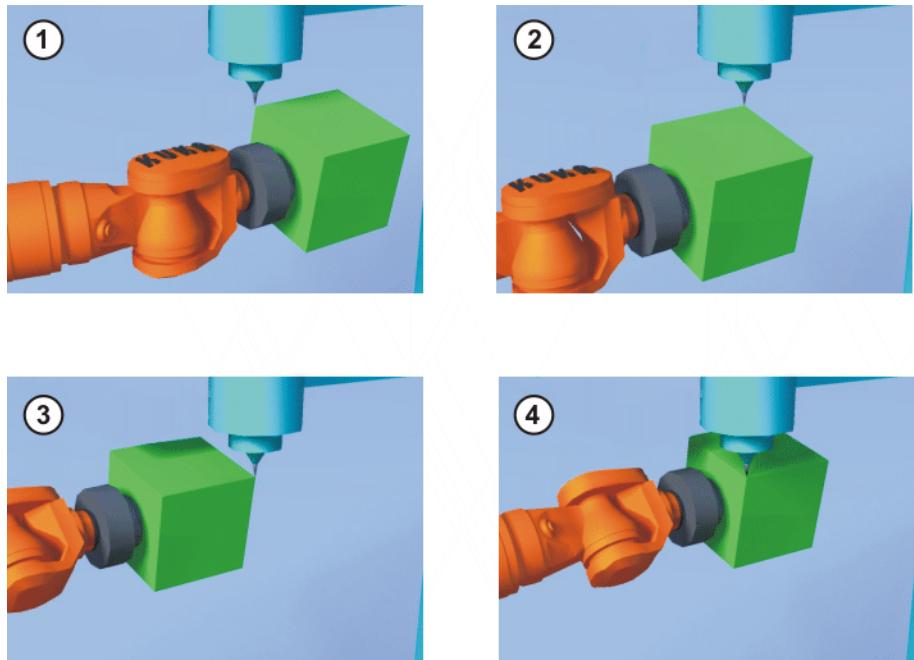


Fig. 5-25: Workpiece calibration: indirect method

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ A previously calibrated fixed tool is mounted.</li><li>■ The workpiece to be calibrated is mounted on the mounting flange.</li><li>■ The coordinates of 4 points of the new workpiece are known, e.g. from CAD data. The 4 points are accessible to the TCP.</li><li>■ Operating mode T1</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; Fixed tool &gt; Workpiece &gt; Indirect calibration</b>.</li><li>2. Assign a number and a name for the workpiece. Confirm with <b>Next</b>.</li><li>3. Enter the number of the fixed tool. Confirm with <b>Next</b>.</li><li>4. Enter the coordinates of a known point on the workpiece and move this point to the TCP of the fixed tool. Press <b>Calibrate</b>. Answer the request for confirmation with <b>Yes</b>.</li><li>5. Repeat step 4 three times.</li><li>6. Enter the load data of the workpiece. (This step can be skipped if the load data are entered separately instead.) (&gt;&gt;&gt; 5.12.3 "Entering payload data" Page 126)</li><li>7. Confirm with <b>Next</b>.</li><li>8. If required, coordinates and orientation of the calibrated points can be displayed in increments and degrees (relative to the FLANGE coordinate sys-</li></ol>

tem). For this, press **Meas. points**. Then return to the previous view by pressing **Back**.

9. Press **Save**.

### 5.11.5 Renaming the tool/base

**Precondition**

- Operating mode T1

**Procedure**

1. In the main menu, select **Start-up > Calibrate > Tool or Base > Change name**.
2. Select the tool or base and press **Name**.
3. Enter the new name and confirm with **Save**.

### 5.11.6 Linear unit

The KUKA linear unit is a self-contained, one-axis linear unit mounted on the floor or ceiling. It is used for linear traversing of the robot and is controlled by the robot controller as an external axis.

The linear unit is a ROBROOT kinematic system. When the linear unit is moved, the position of the robot in the WORLD coordinate system changes. The current position of the robot in the WORLD coordinate system is defined by the vector **\$ROBROOT\_C**.

**\$ROBROOT\_C** consists of:

- **\$ERSYSROOT** (static component)

Root point of the linear unit relative to **\$WORLD**. The root point is situated by default at the zero position of the linear unit and is not dependent on **\$MAMES**.

- **#ERSYS** (dynamic component)

Current position of the robot on the linear unit relative to the **\$ERSYS-ROOT**

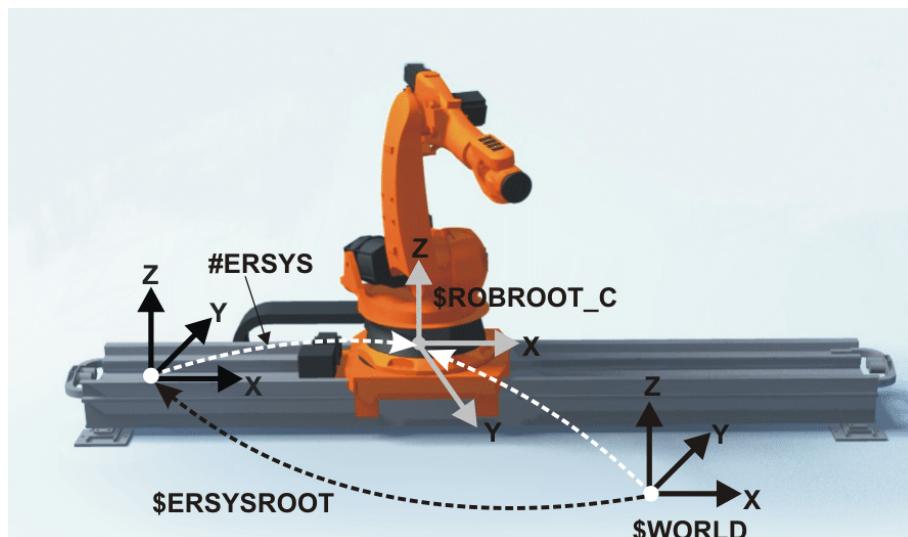


Fig. 5-26: ROBROOT kinematic system – linear unit

#### 5.11.6.1 Checking whether the linear unit needs to be calibrated

**Description**

The robot is standing on the flange of the linear unit. Ideally, the ROBROOT coordinate system of the robot should be identical to the FLANGE coordinate system of the linear unit. In reality, there are often slight discrepancies which mean that positions cannot be moved to correctly. Calibration allows mathe-

matical correction of these discrepancies. (Rotations about the direction of motion of the linear unit cannot be corrected. They do not, however, cause errors when moving to positions.)

If there are no discrepancies, the linear unit does not need to be calibrated. The following procedure can be used to determine whether calibration is required.

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ The machine data of the linear unit have been configured and loaded into the robot controller.</li><li>■ A previously calibrated tool is mounted on the mounting flange.</li><li>■ No program is open or selected.</li><li>■ Operating mode T1</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. Align the TCP against a freely selected point and observe it.</li><li>2. Execute a Cartesian (not axis-specific!) motion with the linear unit.<ul style="list-style-type: none"><li>■ If the TCP stops: the linear unit does not require calibration.</li><li>■ If the TCP moves: the linear unit does require calibration.</li></ul></li></ol>

If the calibration data are already known (e.g. from CAD), they can be entered directly. ([>>> 5.11.6.3 "Entering the linear unit numerically" Page 119](#))

### 5.11.6.2 Calibrating the linear unit

<b>Description</b>	During calibration, the TCP of a tool that has already been calibrated is moved to a reference point 3 times. <ul style="list-style-type: none"><li>■ The reference point can be freely selected.</li><li>■ The position of the robot on the linear unit from which the reference point is approached must be different all 3 times. The 3 positions must be far enough apart.</li></ul>
--------------------	--

The correction values determined by the calibration are factored into the system variable \$ETx\_TFLA3.

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ The machine data of the linear unit have been configured and loaded into the robot controller.</li><li>■ A previously calibrated tool is mounted on the mounting flange.</li><li>■ No program is open or selected.</li><li>■ Operating mode T1</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; External kinematic system &gt; Linear unit</b>. The robot controller detects the linear unit automatically and displays the following data:<ul style="list-style-type: none"><li>■ <b>Ext. kinematic system no.:</b> number of the external kinematic system (1 ... 6) (\$EX_KIN)</li><li>■ <b>Axis:</b> number of the external axis (1 ... 6) (\$ETx_AX)</li><li>■ <b>Name of the external kinematic system</b> (\$ETx_NAME) (If the robot controller is unable to determine these values, e.g. because the linear unit has not yet been configured, calibration cannot be continued.)</li></ul></li><li>2. Move the linear unit with the jog key “+”.</li><li>3. Specify whether the linear unit is moving to “+” or “-”. Confirm with <b>Next</b>.</li><li>4. Move the TCP to the reference point.</li><li>5. Press <b>Calibrate</b>.</li></ol>

6. Repeat steps 4 and 5 twice, but move the linear unit first each time in order to address the reference point from different positions.
7. Press **Save**. The calibration data are saved.
8. The system asks whether the positions that have already been taught are to be corrected.
  - If no positions have been taught prior to the calibration, it makes no difference whether the question is answered with **Yes** or **No**.
  - If positions have been taught prior to the calibration:  
Answering **Yes** will cause positions with base 0 to be corrected automatically. Other positions will not be corrected!  
Answering **No** will cause no positions to be corrected.

**NOTICE**

After calibration of a linear unit, the following safety measures must be carried out:

1. Check the software limit switches of the linear unit and adapt them if required.
2. Test programs in T1.

Damage to property may otherwise result.

#### 5.11.6.3 Entering the linear unit numerically

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ The machine data of the linear unit have been configured and loaded into the robot controller.</li> <li>■ No program is open or selected.</li> <li>■ The following numerical values are known, e.g. from CAD data:           <ul style="list-style-type: none"> <li>■ Distance between the robot base flange and the origin of the ERSYS-ROOT coordinate system (X, Y, Z)</li> <li>■ Orientation of the robot base flange relative to the ERSYSROOT coordinate system (A, B, C)</li> </ul> </li> <li>■ Operating mode T1</li> </ul>
<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; External kinematic system &gt; Linear unit (numeric)</b>. The robot controller detects the linear unit automatically and displays the following data:           <ul style="list-style-type: none"> <li>■ <b>Ext. kinematic system no.:</b> number of the external kinematic system (1 ... 6)</li> <li>■ <b>Axis:</b> number of the external axis (1 ... 6)</li> <li>■ <b>Name of the kinematic system</b></li> </ul> <p>(If the robot controller is unable to determine these values, e.g. because the linear unit has not yet been configured, calibration cannot be continued.)</p> </li> <li>2. Move the linear unit with the jog key “+”.</li> <li>3. Specify whether the linear unit is moving to “+” or “-”. Confirm with <b>Next</b>.</li> <li>4. Enter data. Confirm with <b>Next</b>.</li> <li>5. Press <b>Save</b>. The calibration data are saved.</li> <li>6. The system asks whether the positions that have already been taught are to be corrected.           <ul style="list-style-type: none"> <li>■ If no positions have been taught prior to the calibration, it makes no difference whether the question is answered with <b>Yes</b> or <b>No</b>.</li> <li>■ If positions have been taught prior to the calibration: Answering <b>Yes</b> will cause positions with base 0 to be corrected automatically. Other positions will not be corrected!</li> </ul> </li> </ol>

Answering **No** will cause no positions to be corrected.

**NOTICE**

After calibration of a linear unit, the following safety measures must be carried out:

1. Check the software limit switches of the linear unit and adapt them if required.
2. Test programs in T1.

Damage to property may otherwise result.

### 5.11.7 Calibrating an external kinematic system

**Description**

Calibration of the external kinematic system is necessary to enable the motion of the axes of the kinematic system to be synchronized and mathematically coupled with the robot axes. An external kinematic system can be a turn-tilt table or positioner, for example.



For linear units, the type of calibration described here must not be used. A separate type of calibration must be used for linear units.  
(>>> 5.11.6 "Linear unit" Page 117)

**Overview**

Calibration of an external kinematic system consists of 2 steps:

Step	Description
1	<p>Calibrate the root point of the external kinematic system. (&gt;&gt;&gt; 5.11.7.1 "Calibrating the root point" Page 120)</p> <p>If the calibration data are already known, they can be entered directly. (&gt;&gt;&gt; 5.11.7.2 "Entering the root point numerically" Page 122)</p>
2	<p>If there is a workpiece on the external kinematic system: calibrate the base of the workpiece. (&gt;&gt;&gt; 5.11.7.3 "Workpiece base calibration" Page 122)</p> <p>If the calibration data are already known, they can be entered directly. (&gt;&gt;&gt; 5.11.7.4 "Entering the workpiece base numerically" Page 124)</p> <p>If there is a tool mounted on the external kinematic system: calibrate the external tool. (&gt;&gt;&gt; 5.11.7.5 "Calibrating an external tool" Page 124)</p> <p>If the calibration data are already known, they can be entered directly. (&gt;&gt;&gt; 5.11.7.6 "Entering the external tool numerically" Page 125)</p>

#### 5.11.7.1 Calibrating the root point

**Description**

In order to be able to move the robot with a mathematical coupling to a kinematic system, the robot must know the precise location of the kinematic system. This location is determined by means of root point calibration.

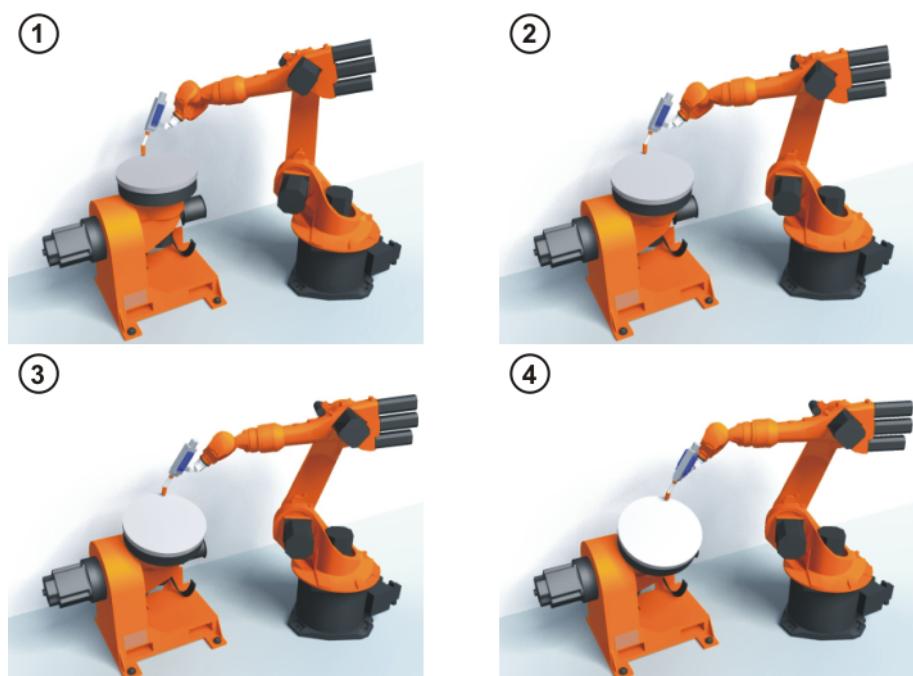
The TCP of a tool that has already been calibrated is moved to a reference point on the kinematic system 4 times. The position of the reference point must be different each time. This is achieved by moving the axes of the kinematic

system. The robot controller uses the different positions of the reference point to calculate the root point of the kinematic system.

In the case of external kinematic systems from KUKA, the reference point is configured in the system variable \$ET<sub>x</sub>\_TPINFL in the machine data. This contains the position of the reference point relative to the FLANGE coordinate system of the kinematic system. ( $x$  = number of the kinematic system.) The reference point is also marked on the kinematic system. During calibration, this reference point must be addressed.

In the case of non-KUKA external kinematic systems, the reference point must be configured in the machine data.

The robot controller saves the coordinates of the root point as the BASE coordinate system.



**Fig. 5-27: Root point calibration principle**

#### Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
- The number of the external kinematic system is known.
- A previously calibrated tool is mounted on the mounting flange.
- If \$ET<sub>x</sub>\_TPINFL is to be modified: user group “Expert”
- Operating mode T1

#### Procedure

1. In the main menu, select **Start-up > Calibrate > External kinematic system > Root point**.
2. Select the number of the BASE coordinate system the root point is to be saved as. Confirm with **Next**.
3. Enter the number of the external kinematic system.
4. Assign a name for the external kinematic system. Confirm with **Next**.
5. Enter the number of the reference tool. Confirm with **Next**.
6. The value of \$ET<sub>x</sub>\_TPINFL is displayed.
  - If the value is not correct: the value can be modified here in the user group “Expert”.
  - If the value is correct: confirm with **Next**.
7. Move the TCP to the reference point.

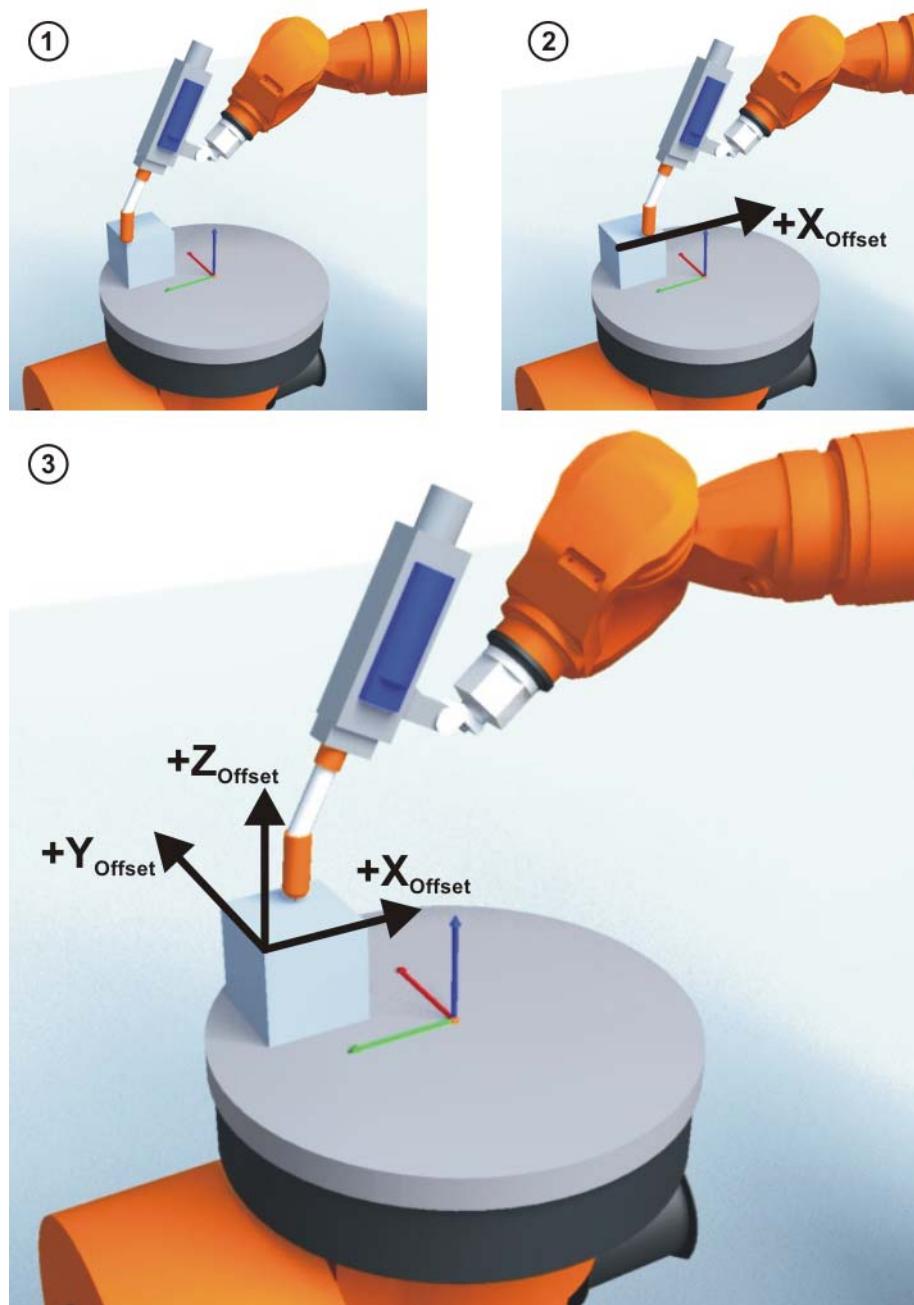
8. Press **Calibrate**. Confirm with **Next**.
9. Repeat steps 7 and 8 three times. Each time, move the kinematic system first so that the reference point is approached from different positions.
10. Press **Save**.

#### 5.11.7.2 Entering the root point numerically

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ The following numerical values are known, e.g. from CAD data:<ul style="list-style-type: none"><li>■ Distance between the origin of the ROOT coordinate system and the origin of the WORLD coordinate system (X, Y, Z)</li><li>■ Orientation of the ROOT coordinate system relative to the WORLD coordinate system (A, B, C)</li></ul></li><li>■ The number of the external kinematic system is known.</li><li>■ Operating mode T1</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; External kinematic system &gt; Root point (numeric)</b>.</li><li>2. Select the number of the BASE coordinate system the root point is to be saved as. Confirm with <b>Next</b>.</li><li>3. Enter the number of the external kinematic system.</li><li>4. Assign a name for the external kinematic system. Confirm with <b>Next</b>. (The name is automatically also assigned to the BASE coordinate system.)</li><li>5. Enter the data of the ROOT coordinate system. Confirm with <b>Next</b>.</li><li>6. Press <b>Save</b>.</li></ol>

#### 5.11.7.3 Workpiece base calibration

<b>Description</b>	<p>During this calibration, the user assigns a BASE coordinate system to a work-piece located on the kinematic system. This BASE coordinate system is relative to the FLANGE coordinate system of the kinematic system. The base is thus a moving base that moves in the same way as the kinematic system.</p> <p>It is not strictly necessary to calibrate a base. If none is calibrated, the FLANGE coordinate system of the kinematic system is taken as the base.</p> <p>During calibration, the TCP of a calibrated tool is moved to the origin and 2 other points of the desired base. These 3 points define the base. Only one base can be calibrated per kinematic system.</p>
--------------------	--



**Fig. 5-28: Base calibration principle**

#### Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
- A previously calibrated tool is mounted on the mounting flange.
- The root point of the external kinematic system has been calibrated.
- The number of the external kinematic system is known.
- Operating mode T1

#### Procedure

1. In the main menu, select **Start-up > Calibrate > External kinematic system > Offset**.
2. Enter the number of the BASE coordinate system the root point was saved as. The name of the BASE coordinate system is displayed.  
Confirm with **Next**.
3. Enter the number of the external kinematic system. The name of the external kinematic system is displayed.  
Confirm with **Next**.

4. Enter the number of the reference tool. Confirm with **Next**.
5. Move the TCP to the origin of the workpiece base. Press **Calibrate** and confirm with **Next**.
6. Move the TCP to a point on the positive X axis of the workpiece base. Press **Calibrate** and confirm with **Next**.
7. Move the TCP to a point in the XY plane with a positive Y value. Press **Calibrate** and confirm with **Next**.
8. Press **Save**.

#### 5.11.7.4 Entering the workpiece base numerically

##### Precondition

- The following numerical values are known, e.g. from CAD data:
  - Distance between the origin of the workpiece base and the origin of the FLANGE coordinate system of the kinematic system (X, Y, Z)
  - Rotation of the axes of the workpiece base relative to the FLANGE coordinate system of the kinematic system (A, B, C)
- The root point of the external kinematic system has been calibrated.
- The number of the external kinematic system is known.
- Operating mode T1

##### Procedure

1. In the main menu, select **Start-up > Calibrate > External kinematic system > Offset (numeric)**.
2. Enter the number of the BASE coordinate system the root point was saved as. The name of the BASE coordinate system is displayed.  
Confirm with **Next**.
3. Enter the number of the external kinematic system. The name of the external kinematic system is displayed.  
Confirm with **Next**.
4. Enter data. Confirm with **Next**.
5. Press **Save**.

#### 5.11.7.5 Calibrating an external tool

##### Description

During calibration of the external tool, the user assigns a coordinate system to the tool mounted on the kinematic system. This coordinate system has its origin in the TCP of the external tool and is relative to the FLANGE coordinate system of the kinematic system.

First of all, the user communicates to the robot controller the TCP of the tool mounted on the kinematic system. This is done by moving a calibrated tool to the TCP.

Then, the orientation of the coordinate system of the tool is communicated to the robot controller. For this purpose, the user aligns the coordinate system of the calibrated tool parallel to the new coordinate system. There are 2 variants:

- **5D:** The user communicates the tool direction to the robot controller. By default, the tool direction is the X axis. The orientation of the other axes is defined by the system and cannot be influenced by the user.  
The system always defines the orientation of the other axes in the same way. If the tool subsequently has to be calibrated again, e.g. after a crash, it is therefore sufficient to define the tool direction again. Rotation about the tool direction need not be taken into consideration.
- **6D:** The user communicates the direction of all 3 axes to the robot controller.



If **6D** is selected: it is advisable to document the alignment of all axes. If the tool subsequently has to be calibrated again, e.g. after a crash, the axes must be aligned the same way as the first time in order to be able to continue moving to existing points correctly.

The robot controller saves the coordinates of the external tool as the BASE coordinate system.

#### Precondition

- The machine data of the kinematic system have been configured and loaded into the robot controller.
- A previously calibrated tool is mounted on the mounting flange.
- The root point of the external kinematic system has been calibrated.
- The number of the external kinematic system is known.
- Operating mode T1



The following procedure applies if the tool direction is the default tool direction (= X axis). If the tool direction has been changed to Y or Z, the procedure must also be changed accordingly.

(>>> 5.11.1 "Defining the tool direction" Page 102)

#### Procedure

1. In the main menu, select **Start-up > Calibrate > Fixed tool > External kinematic offset**.
2. Enter the number of the BASE coordinate system the root point was saved as. The name of the BASE coordinate system is displayed.  
Confirm with **Next**.
3. Enter the number of the external kinematic system. The name of the external kinematic system is displayed.  
Confirm with **Next**.
4. Enter the number of the reference tool. Confirm with **Next**.
5. Select a variant in the box **5D/6D**. Confirm with **Next**.
6. Move the TCP of the calibrated tool to the TCP of the external tool. Press **Calibrate** and confirm with **Next**.
7. If **5D** is selected:  
Align +X<sub>BASE</sub> parallel to -Z<sub>FLANGE</sub>  
(i.e. align the mounting flange perpendicular to the tool direction of the external tool.)  
If **6D** is selected:  
Align the mounting flange so that its axes are parallel to the axes of the external tool:
  - +X<sub>BASE</sub> parallel to -Z<sub>FLANGE</sub>  
(i.e. align the mounting flange perpendicular to the tool direction of the external tool.)
  - +Y<sub>BASE</sub> parallel to +Y<sub>FLANGE</sub>
  - +Z<sub>BASE</sub> parallel to +X<sub>FLANGE</sub>
8. Press **Calibrate** and confirm with **Next**.
9. Press **Save**.

#### 5.11.7.6 Entering the external tool numerically

#### Precondition

- The following numerical values are known, e.g. from CAD data:
  - Distance between the TCP of the external tool and the origin of the FLANGE coordinate system of the kinematic system (X, Y, Z)

- Rotation of the axes of the external tool relative to the FLANGE coordinate system of the kinematic system (A, B, C)
- Operating mode T1

<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>Start-up &gt; Calibrate &gt; Fixed tool &gt; Numeric input</b>.</li><li>2. Assign a number and a name for the external tool. Confirm with <b>Next</b>.</li><li>3. Enter data. Confirm with <b>Next</b>.</li><li>4. Press <b>Save</b>.</li></ol>
------------------	--

## 5.12 Load data

The load data are factored into the calculation of the paths and accelerations and help to optimize the cycle times. The load data must be entered in the robot controller.

<b>Sources</b>	Load data can be obtained from the following sources: <ul style="list-style-type: none"><li>■ Software option KUKA.LoadDataDetermination (only for payloads on the flange)</li><li>■ Manufacturer information</li><li>■ Manual calculation</li><li>■ CAD programs</li></ul>
----------------	---

### 5.12.1 Checking loads with KUKA.Load

All load data (payload and supplementary loads) must be checked with the KUKA.Load software. Exception: If the payload is checked with KUKA.LoadDataDetermination, it is not necessary to check it with KUKA.Load.

A sign-off sheet can be generated for the loads with KUKA.Load. KUKA.Load can be downloaded free of charge, complete with the documentation, from the KUKA website [www.kuka.com](http://www.kuka.com).



More information is contained in the **KUKA.Load** documentation.

### 5.12.2 Calculating payloads with KUKA.LoadDataDetermination

<b>Description</b>	KUKA.LoadDataDetermination can be used to calculate payloads exactly and transfer them to the robot controller.
<b>Precondition</b>	<ul style="list-style-type: none"><li>■ T1 or T2 operating mode</li><li>■ No program is selected.</li></ul>
<b>Procedure</b>	<ul style="list-style-type: none"><li>■ In the main menu, select <b>Start-up &gt; Service &gt; Load data determination</b>.</li></ul>



More information is contained in the **KUKA.LoadDataDetermination** documentation.

### 5.12.3 Entering payload data

<b>Description</b>	The payload data must be entered in the robot controller and assigned to the correct tool.
--------------------	--

Exception: If the payload data have already been transferred to the robot controller by KUKA.LoadDataDetermination, no manual entry is required.

- Precondition**
- The payload data have been checked with KUKA.Load or KUKA.LoadDataDetermination and the robot is suitable for these payloads.
- Procedure**
1. In the main menu, select **Start-up > Calibrate > Tool > Payload data**.
  2. Enter the number of the tool in the box **Tool no.**. Confirm with **Next**.
  3. Enter the payload data:
    - Box **M**: Mass
    - Boxes **X**, **Y**, **Z**: Position of the center of gravity relative to the flange
    - Boxes **A**, **B**, **C**: Orientation of the principal inertia axes relative to the flange
    - Boxes **JX**, **JY**, **JZ**: Mass moments of inertia  
(JX is the inertia about the X axis of the coordinate system that is rotated relative to the flange by A, B and C. JY and JZ are the analogous inertia about the Y and Z axes.)
  - Or, if the default values for this robot type are to be used: press **Default**.
  4. If online load data verification is available (this depends on the robot type): configure as required.  
(>>> 5.12.5 "Online load data check (OLDC)" Page 127)
  5. Confirm with **Next**.
  6. Press **Save**.

#### 5.12.4 Entering supplementary load data

- Description** The supplementary load data must be entered in the robot controller.  
Reference systems of the X, Y and Z values for each supplementary load:

Load	Reference system
Supplementary load A1	ROBROOT coordinate system A1 = 0°
Supplementary load A2	ROBROOT coordinate system A2 = -90°
Supplementary load A3	FLANGE coordinate system A4 = 0°, A5 = 0°, A6 = 0°

- Precondition**
- The supplementary loads have been verified with KUKA.Load and are suitable for this robot type.
- Procedure**
1. In the main menu, select **Setup > Measure > Supplementary load data**.
  2. Enter the number of the axis on which the supplementary load is to be mounted. Confirm with **Continue**.
  3. Enter the load data. Confirm with **Continue**.
  4. Press **Save**.

#### 5.12.5 Online load data check (OLDC)

- Description** For many robot types, the robot controller monitors whether or not there is an overload or underload during operation. This monitoring is called "Online load data check" (OLDC).  
If the OLDC detects an underload, for example, the robot controller reacts, e.g. by displaying a message. The reactions can be configured.

The results of the check can be polled using the system variable \$LDC\_RESULT. ([">>>> "\\$LDC\\_RESULT" Page 129](#))

OLDC is available for those robot types for which KUKA.LoadDataDetermination can also be used. Whether or not OLDC is available for the current robot type can be checked by means of \$LDC\_LOADED (TRUE = yes).

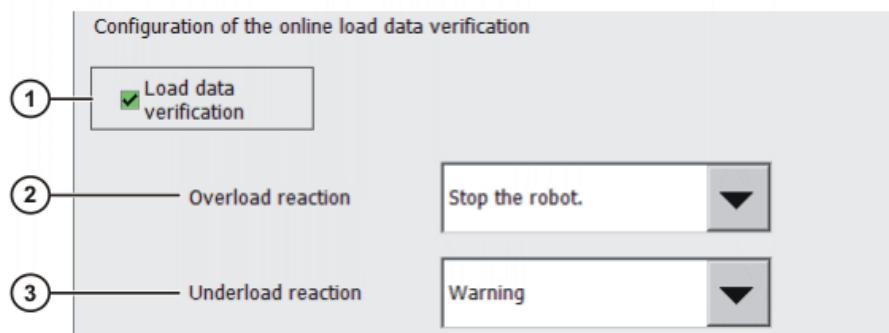
Overload	There is an overload if the actual load is greater than the configured load.
Underload	There is an underload if the actual load is less than the configured load.

## Configuration

OLDC can be configured as follows:

- During manual entry of the tool data  
([">>>> 5.11.2.5 "Numeric input" Page 109](#))
- During separate entry of the payload data  
([">>>> 5.12.3 "Entering payload data" Page 126](#))

The following boxes are displayed in the same window in which the payload data are also entered:



**Fig. 5-29: Online load data check**

Item	Description
1	<b>TRUE:</b> OLDC is activated for the tool displayed in the same window. The defined reactions are carried out in the case of an overload or underload.  <b>FALSE:</b> OLDC is deactivated for the tool displayed in the same window. There is no reaction in the case of an overload or underload.
2	The overload reaction can be defined here. <ul style="list-style-type: none"> <li>■ <b>None:</b> No reaction.</li> <li>■ <b>Warning:</b> The robot controller generates the following status message: <b>Check of robot load (Tool (Tool number)) calculated overload.</b></li> <li>■ <b>Stop robot:</b> The robot controller generates an acknowledgement message with the same content as that generated under <b>Warning</b>. The robot stops with a STOP 2.</li> </ul>
3	The underload reaction can be defined here. The possible reactions are analogous to those for an overload.

The reactions can be modified in the KRL program using the system variable \$LDC\_CONFIG. ([">>>> "\\$LDC\\_CONFIG" Page 129](#))

## NULFRAME

OLDC cannot be configured for motions to which the tool **NULFRAME** has been assigned. The reactions are defined for this case and cannot be influenced by the user.

- Overload reaction: **Stop robot**  
The following acknowledgement message is generated: **Overload calculated when checking robot load (no tool defined) and the set load data.** The robot stops with a STOP 2.
- Underload reaction: **Warning**  
The following status message is generated: **Underload calculated when checking robot load (no tool defined) and the set load data.**

**\$LDC\_CONFIG**

`$LDC_CONFIG[Index] = {UNDERLOAD Reaction, OVERLOAD Reaction}`

Element	Description
<i>Index</i>	Type: INT Tool number ■ 1 ... 32
<i>Reaction</i>	Type: CHAR ■ #NONE (= <b>None</b> ) ■ #WARNONLY (= <b>Warning</b> ) ■ #STOPROBOT (= <b>Stop robot</b> )

**Example:**

```

1 ...
2 $LDC_CONFIG[1]={UNDERLOAD #NONE, OVERLOAD #NONE}
3 ...
4 $LDC_CONFIG[1]={UNDERLOAD #WARNONLY, OVERLOAD #WARNONLY}
5 ...

```

Line	Description
2	The reaction for both underload and overload is set to <b>None</b> .
4	The reaction is set to <b>Warning</b> . If there is an underload or overload, the corresponding status messages will be displayed in the message window.

**\$LDC\_RESULT**

`$LDC_RESULT[Index] = Result`

Element	Description
<i>Index</i>	Type: INT Tool number ■ 1 ... 32
<i>Result</i>	Type: CHAR ■ #OK: The payload is OK. (Neither overload nor underload.) ■ #OVERLOAD: There is an overload. ■ #UNDERLOAD: There is an underload. ■ #CHECKING ■ #NONE: There are currently no results, e.g. because the tool has been changed.

**5.13 Maintenance handbook**

The **Maintenance handbook** functionality is available in the KUKA System Software. The maintenance handbook enables logging of the maintenance work. The logged maintenance work can be displayed in an overview.

The robot controller uses messages to indicate when maintenance is due:

- A message is generated one month before the maintenance work is due. This message can be acknowledged.
- At the end of the month, the robot controller generates a message indicating that the maintenance is now due. This message cannot be acknowledged. Additionally, LED4 on the Controller System Panel flashes (= first LED from the left in the bottom row).  
Only when the corresponding maintenance work has been logged does the robot controller reset the message and the LED stops flashing.

The due dates are determined by the maintenance intervals specified in the KUKA maintenance agreements. The intervals are counted from the initial start-up of the robot controller. The operating hours of the robot are counted.

### 5.13.1 Logging maintenance

<b>Description</b>	It is not possible to log multiple maintenance activities of the same kind on one day.
--------------------	--



Once saved, changes can no longer be made.

#### Precondition

- Expert user group

#### Procedure

1. In the main menu, select **Start-up > Service > Maintenance handbook**. The **Maintenance handbook** window is opened.
2. Select the **Maintenance input** tab and enter the maintenance details. Entries must be made in all boxes.
3. Press **Save**. A request for confirmation is displayed.
4. If all entries are correct, answer the request for confirmation with **Yes**.

The entries are now saved. Switching to the **Maintenance overview** tab causes the maintenance to be displayed there.

Maintenance handbook

Maintenance type

Person/Company

Order number

Comment

Save

Maintenance input      Maintenance overview

**Fig. 5-30: Maintenance input**

Item	Description
1	Select which type of maintenance has been carried out.
2	Enter who performed the maintenance.
3	For maintenance carried out and logged by KUKA employees: enter the order number. For other maintenance: enter any number.
4	Enter a comment.

- Maintenance types** By default, the following maintenance types can be selected:
- **Basic inspection**
  - **In-line wrist maintenance**
  - **Main axis maintenance**
  - **Gear backlash measurement**
  - **Minor electrical maintenance**
  - **Major electrical maintenance**
  - **Data backup with spare hard drive**
  - **Repair**

These maintenance types correspond to those in the KUKA maintenance agreements. Depending on the options used (e.g. linear axis or technology packages), other maintenance types may be available for selection.

### 5.13.2 Displaying a maintenance log

- Description** The logged maintenance work can be displayed in an overview. If the KUKA System Software is updated (e.g. from KSS 8.2.3 to KSS 8.2.4), this overview is retained.

When archiving is carried out, the maintenance logs are also archived. If, when the data are restored, other maintenance work has been logged on the robot controller in the meantime, these logs are not overwritten; instead, the restored logs are added to the overview.

**Procedure**

1. In the main menu, select **Start-up > Service > Maintenance handbook**.  
The **Maintenance handbook** window is opened.
2. Select the **Maintenance overview** tab.

Basic inspection				
Date	Person/Company	Operating hou	Order no.	Comment
10/7/2011	KUKA	0	123456	my test description
3/7/2011	KUKA Germany	0	1122778	Part xyz changed by reason of a defect
Minor electrical maintenance				
Date	Person/Company	Operating hou	Order no.	Comment
3/7/2011	KUKA Robotics	0	11223344	Batteries changed

**Fig. 5-31: Maintenance overview**

## 6 Configuration

### 6.1 Configuring the KUKA Line Interface (KLI)

The KLI is the Ethernet interface of the robot controller for external communication. It is a physical interface and can contain multiple virtual interfaces.

In order for external PCs to be able to connect to the robot controller via a network, the KLI must be configured accordingly. This is a precondition, for example, for being able to transfer WorkVisual projects to the robot controller via the network.

**NOTICE**

The following address ranges are used by default by the robot controller for internal purposes. IP addresses from this range must not therefore be assigned by the user.

- 192.168.0.0 ... 192.168.0.255
- 172.16.0.0 ... 172.16.255.255
- 172.17.0.0 ... 172.17.255.255

#### 6.1.1 Configuring the Windows interface (without PROFINET)

##### Description

The Windows interface is a virtual interface of the KLI.

From Build 48 onwards, KSS 8.2 is supplied with a preconfigured static IP address for the Windows interface:

- IP address: 172.31.1.147
- Subnet mask: 255.255.0.0

These values can be modified by the user. If required, it is also possible to switch to dynamic address assignment. Similarly, it is also possible to switch back from a dynamic address to a static address.

##### Precondition

- PROFINET is not used.
- If the address type **Dynamic IP address** is to be selected: there is a DHCP server present in the network.
- No program is selected.
- T1 or T2 operating mode
- “Expert” user group

##### Procedure



The Address and Subnet boxes may be displayed with a red frame. This means that there is an error.

(>>> 6.1.6 "Error display in the Address and Subnet boxes"

Page 138)

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window is opened. The active Windows interface is displayed. (Default: “virtual5”)
2. Select the desired type in the **Address type** box: **Dynamic IP address** or **Fixed IP address**

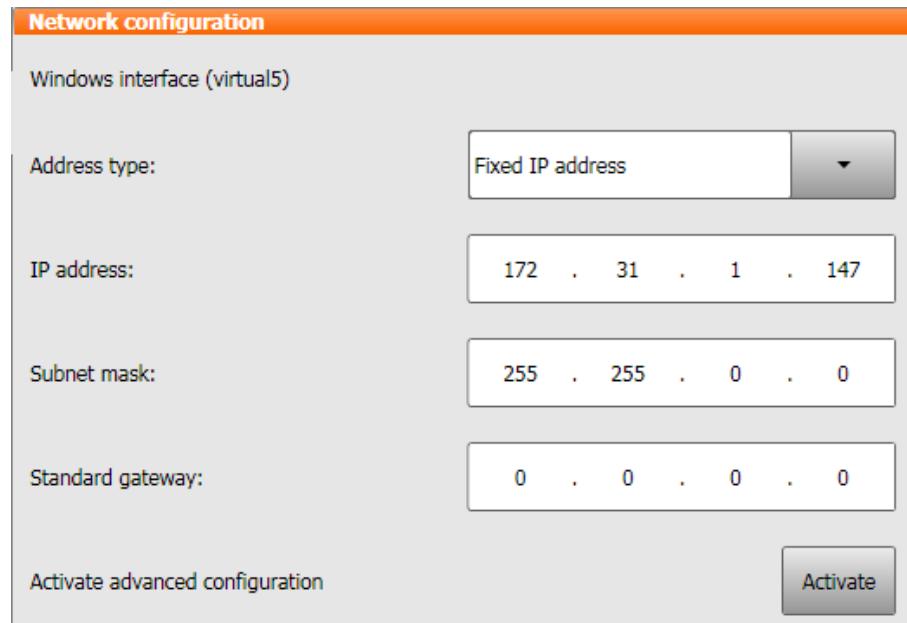


The other types in this box (e.g. **Real-time IP address**) must not be selected.

3. Only for **Dynamic IP address**:

The following boxes are deactivated, as the values are automatically assigned by the DHCP server: **IP address**, **Subnet mask**, **Standard gateway**

4. Only for **Fixed IP address**: Fill out the following boxes:
  - **IP address**: enter the IP address of the robot controller.
  - **Subnet mask**: the selected subnet mask must match the IP network.
  - **Standard gateway** (if required): specifies the IP address that can be used to leave the network.  
“0.0.0.0” is admissible and is ignored by the system.
5. Press **Save**.
6. Reboot the robot controller so that the change takes effect.



**Fig. 6-1: Example: Fixed IP address**

### 6.1.2 Configuring the Windows PROFINET interface and creating the Windows interface

**Description** PROFINET automatically occupies the virtual interface “virtual5”. This must be adapted to PROFINET.  
“Virtual5” can also be used as a Windows interface, but the static IP address of PROFINET must then be used for Windows access.  
If a different IP configuration is to be used for the Windows interface, an additional virtual interface, e.g. “virtual6” must be added and configured for this purpose.

**Precondition**

- PROFINET is used.
- PROFINET device naming has been completed.

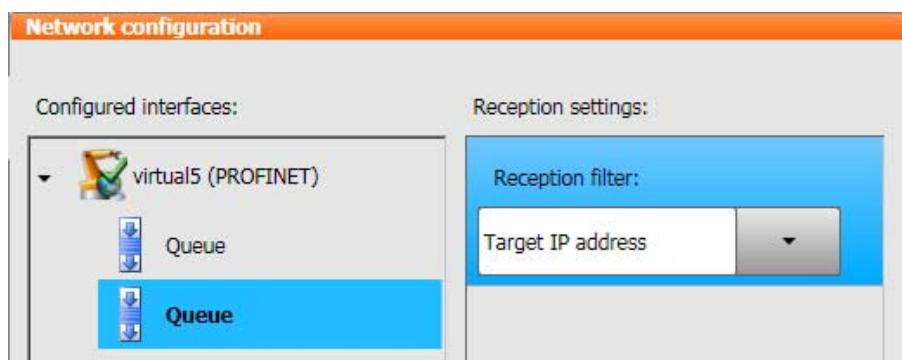
**Information about device naming**  
KR C4 PROFINET.

- No program is selected.
- T1 or T2 operating mode
- “Expert” user group

**Procedure**

**Information about device naming**  
The Address and Subnet boxes may be displayed with a red frame.  
This means that there is an error.  
(>>> 6.1.6 "Error display in the Address and Subnet boxes"  
Page 138)

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window is opened. The “virtual5” interface is displayed.
2. If it has not already been selected: select the type **Fixed IP address** in the box **Address type**.
3. Fill out the boxes **IP address** and **Subnet mask**. Enter the address and mask that are also assigned by the PROFINET PLC.
4. Press **Activate**. The window for advanced network configuration opens.
5. Select the **Interfaces** tab.
6. Select the entry “virtual5” in the **Configured interfaces** area and enter “PROFINET” in the **Interface name** box.
7. Several “Queue” entries are displayed under the entry “virtual5”. Select the bottom one.
8. The **Reception filter** box indicates **Accept all**. Change the entry to **Target IP address**.

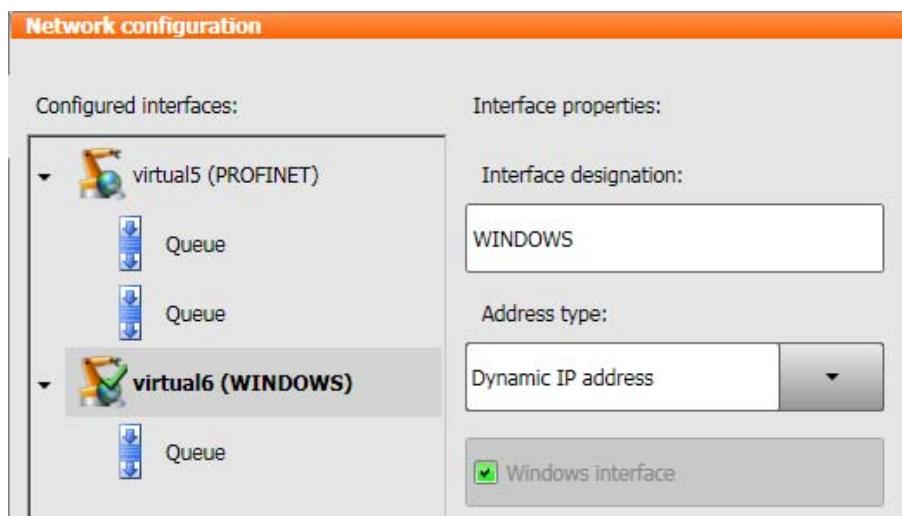


**Fig. 6-2**

9. Select the entry “virtual5” and press the **Add interface** button. The entry “virtual6” is automatically created.
10. Select the entry “virtual6” and enter “WINDOWS” in the **Interface name** box.
11. Select the type **Dynamic IP address** in the **Address type** box.

**Fixed IP address** can also be selected here if a static IP address is desired. The static address must be in a different address range, however, than the address of “virtual5”.

12. Set the check mark in the check box **Windows interface**.



**Fig. 6-3: Interface “virtual6”**

13. Several the “Queue” entry under the entry “virtual6”.
14. If not already the case, change the **Reception filter** box to **Accept all**.
15. Press **Save**.
16. Close the **Network configuration** window using the Close icon.
17. Reboot the robot controller. For this, select **Shutdown** in the main menu and select the option **Reload files**.



The check mark in the check box **Windows interface** cannot be removed. Deselection is only possible by defining a different interface as the Windows interface.

#### 6.1.3 Displaying ports of the Windows interface or enabling an additional port



It is not generally necessary to enable additional ports. If this is nevertheless to be done, KUKA Roboter GmbH must be contacted beforehand.

**NOTICE**

The ports enabled by KUKA by default must not be removed. Doing so may result in the loss of functions of the robot controller.

**Precondition**

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

**Procedure**

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window is opened.
2. Press **Activate**. The window for advanced network configuration opens.
3. Select the **NAT** tab. A list of all the enabled ports of the Windows interface is displayed in the **Available ports** area.
4. Only if a port is to be enabled:
  - a. Press **Add port**. A new port with the number “0” is added to the list.
  - b. Complete the boxes **Port number** and **Permitted protocols**.
  - c. Press **Save**.A maximum total of 40 ports can be enabled.
5. Close the **Network configuration** window using the Close icon.
6. Only if modifications have been made:  
Reboot the robot controller so that the changes take effect. To do so, if PROFINET is used, select **Shutdown** in the main menu and select the option **Reload files**.

#### 6.1.4 Displaying or modifying filters



It is not generally necessary to modify the filters. If this is nevertheless to be done, KUKA Roboter GmbH must be contacted beforehand.

**NOTICE**

The filters set by KUKA by default must not be modified or removed. Doing so may result in the loss of functions of the robot controller.

**Precondition**

- User group “Expert”
- Operating mode T1 or T2.

- No program is selected.

**Procedure**

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window is opened.
2. Press **Activate**. The window for advanced network configuration opens.
3. Select the **Internal subnets** tab. The filters of the KUKA Line Interface and their properties are displayed here.
4. Only if modifications are required: Carry them out and press **Save**.
5. Close the **Network configuration** window using the Close icon.
6. Only if modifications have been made:  
Reboot the robot controller so that the changes take effect. To do so, if PROFINET is used, select **Shutdown** in the main menu and select the option **Reload files**.

**6.1.5 Displaying the subnet configuration of the robot controller****Description**

The subnet configuration of the robot controller can be displayed. This makes it possible to compare it with the subnets of the customer network.

**NOTICE**

The subnet configuration of the robot controller may be modified only in consultation with KUKA Roboter GmbH. Modifications carried out without consultation may result in the loss of functions of the robot controller.

**Precondition**

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

**Procedure**

1. In the main menu, select **Start-up > Network configuration**. The **Network configuration** window is opened.
2. Press **Activate**. The window for advanced network configuration opens.
3. Select the **Internal subnets** tab. The subnet configuration of the robot controller is displayed.  
Only the network address is displayed. The range containing the address of the device is indicated by an “x”.



The Address and Subnet boxes may be displayed with a red frame. This means that there is an error.  
(>>> 6.1.6 "Error display in the Address and Subnet boxes"

Page 138



**Fig. 6-4: Example: Internal subnets**

### 6.1.6 Error display in the Address and Subnet boxes

#### Description

The Address and Subnet boxes may be displayed with a red frame. This indicates an error and may have the following causes:

Cause	Possible remedy
<p>The entry does not conform to the IP system.</p> <p>Example: The number ranges of the IP address and subnet do not match.</p>	<p>Check the box with the red frame and correct it.</p> <p>The red frame disappears.</p>
<p>This address is already used in one of the internal subnets.</p> <p>In this case, both the address and the corresponding box on the <b>Internal subnets</b> tab are displayed with a red frame.</p>	<ul style="list-style-type: none"> <li>■ Change the actual address.</li> <li>■ Or change the address of the internal subnet.</li> </ul> <p>The red frame disappears.</p>

#### NOTICE

The subnet configuration of the robot controller may be modified only in consultation with KUKA Roboter GmbH. Modifications carried out without consultation may result in the loss of functions of the robot controller.

#### Example

#### Example of an entry that is not system-compliant:

In the binary display, subnet masks may only contain closed groups of leading ones.

- The subnet mask "255.255.208.0" is entered.
  - The box is now displayed with a red frame.
- Reason: The binary representation of "208" corresponds to "11010000" and is thus not a valid entry.
- Possible remedy: enter 255.255.240.0.
- The binary representation of "240" corresponds to "11110000" and is thus a valid entry.

### 6.1.7 Configuring DNS

#### Description

If the robot controller is connected to the IT network, name resolution of the devices can be carried out via the DNS server address. (In other words, the DNS server receives a query with a device name and responds with the corresponding IP address.)

In KSS 8.2, this configuration must be carried out manually. This is not dependent on whether the address type **Dynamic IP address** or **Fixed IP address** has been selected.

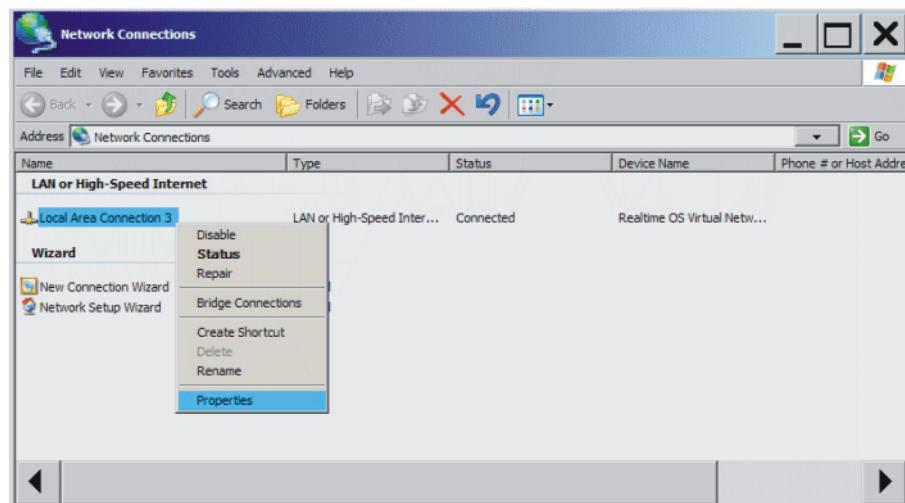
#### Precondition

- User group "Expert".
- Operating mode T1 or T2.

#### Procedure

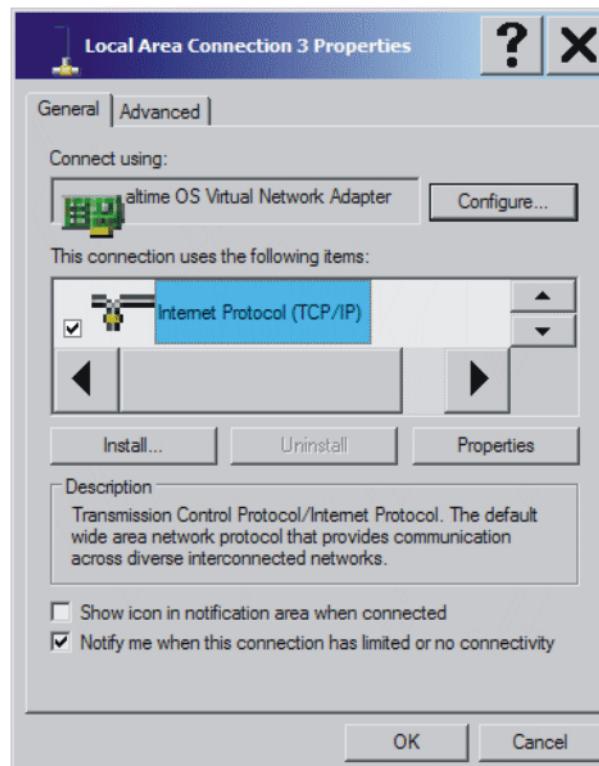
1. Select **Start-up > Service > Minimize HMI** in the main menu.
2. Open the Windows Start menu and select **Network Connections**.

The **Network Connections** window is opened and the LAN connection is displayed.



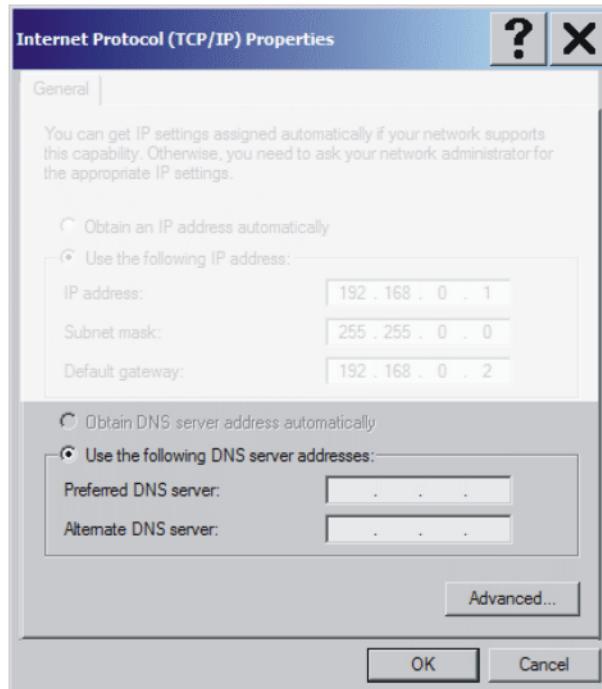
**Fig. 6-5: “Network Connections” window**

3. Right-click on the LAN connection and select **Properties**.  
The **Local Area Connection properties** window is opened.



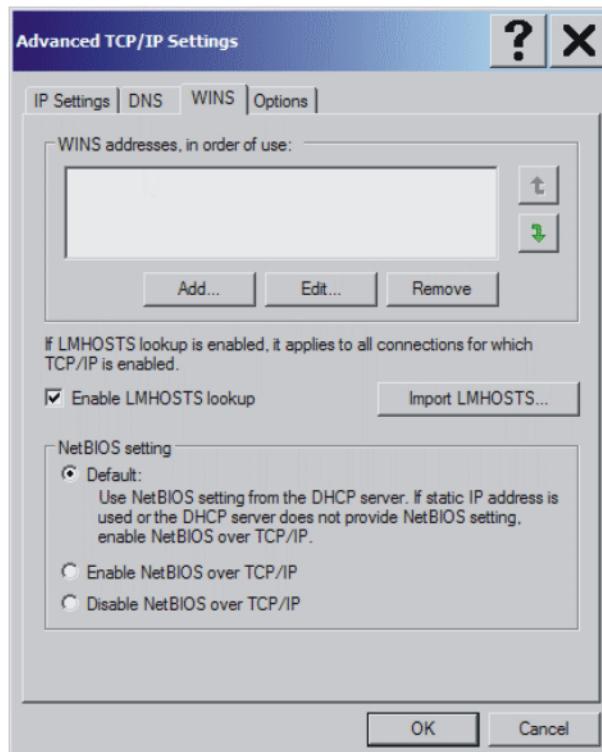
**Fig. 6-6: “Local Area Connection Properties” window**

4. Select **Internet Protocol (TCP/IP)** and select the **Properties** button.  
The **Internet Protocol (TCP/IP) Properties** window is opened.



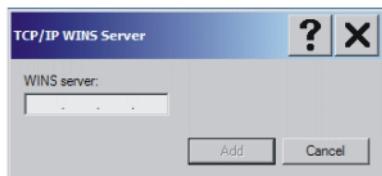
**Fig. 6-7: “Internet Protocol Properties” window, example**

5. Activate the radio button **Use the following DNS server addresses**.  
Enter the preferred DNS server address. An alternative DNS server address can also be entered.
6. Press the **Advanced...** button.  
The **Advanced TCP/IP Settings** window opens.
7. Select the **WINS** tab.



**Fig. 6-8: “Advanced TCP/IP Settings” window**

8. Press the **Add...** button.  
The **TCP/IP WINS Server** window opens.



**Fig. 6-9: "TCP/IP WINS Server" window**

9. Enter the WINS server address. The WINS server address must match the DNS server address entered in step 4.
10. Press the **Add** button.  
The WINS server address is applied and displayed in the **Advanced TCP/IP Settings** window.
11. Close the **Advanced TCP/IP Settings** window with **OK**.
12. Close the **Internet Protocol (TCP/IP) Properties** window with **OK**.
13. Close the **Local Area Connection properties** window with **OK**.

## 6.2 Reconfiguring the I/O driver

### Precondition

- User group "Expert".
- Operating mode T1 or T2.

### Procedure

- ⚠ WARNING** All outputs are reset!
- In the main menu, select **Configuration > Inputs/outputs > Reconfigure I/O driver**.

## 6.3 Checking the safety configuration of the robot controller

### Description

The safety configuration of the robot controller must be checked in the following cases:

- After activation of a WorkVisual project on the robot controller
- Generally after changes to the machine data (independent of WorkVisual).

**⚠ WARNING** If the safety configuration is not checked and updated where necessary, it may contain incorrect data. Death to persons, severe physical injuries or considerable damage to property may result.

### Precondition

- User group Expert or higher

**i** If one of the options KUKA.SafeOperation or KUKA.SafeRangeMonitoring is installed on the robot controller, different user groups may apply. Information can be found in the documentation for these options.

### Procedure

1. Select the menu sequence **Configuration > Safety configuration**.
2. The safety configuration checks whether there are any relevant deviations between the data in the robot controller and those in the safety controller.
3. The following situations can now occur:
  - a. If there are no deviations, the **Safety configuration** window is opened. No message is displayed. No further action is necessary.
  - b. If there are deviations regarding the machine data, a dialog message is displayed. This indicates which machine data in the robot controller deviate from those in the safety controller.

The message asks whether the safety controller should be updated.  
Confirm the request with **Yes**.

The system asks whether the deviations should now be accepted.  
Confirm the request with **Yes**.

- c. The safety configuration also checks whether there are any other deviations (other than in the machine data) between the robot controller and the safety controller.
- If so, the **Troubleshooting wizard** window is opened. A description of the problem and a list of possible causes is displayed. The user can select the applicable cause. The wizard then suggests a solution.

## 6.4 Configuring the variable overview

This is where the variables to be displayed in the variable overview and the number of groups are defined. A maximum of 10 groups is possible. A maximum of 25 variables per group is possible. System variables and user-defined variables can be displayed.

### Precondition

- Expert user group

### Procedure

1. In the main menu, select **Display > Variable > Overview > Configuration**.  
The **Variable overview - Configuration** window is opened.
2. Make the desired settings. To edit a cell, select it.
3. Press **OK** to save the configuration and close the window.

### Description

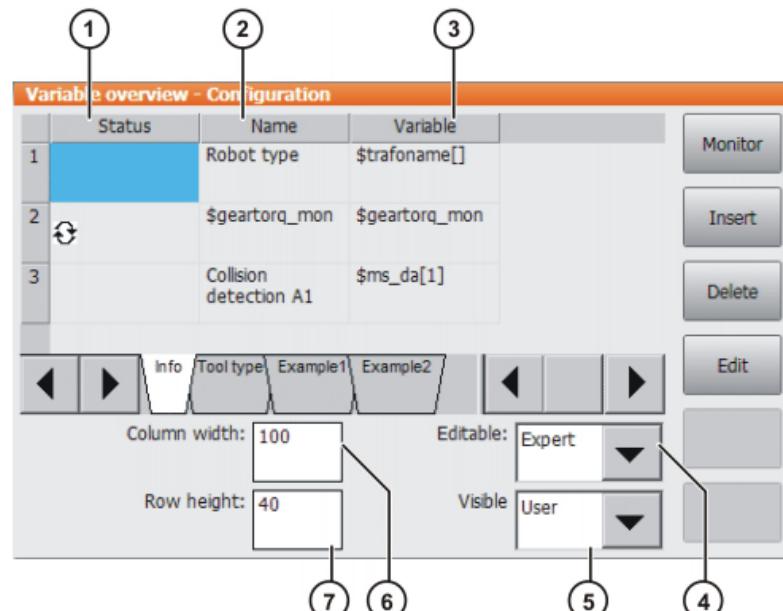


Fig. 6-10: Variable overview - Configuration

Item	Description
1	Arrow symbol : If the value of the variable changes, the display is automatically updated. No arrow symbol: The display is not automatically updated.
2	Descriptive name

Item	Description
3	Path and name of the variable  <b>Note:</b> For system variables, the name is sufficient. Other variables must be specified as follows:  /R1/Program name/Variable name  Do not specify a folder between /R1/ and the program name. Do not add a file extension to the file name.
4	Lowest user group in which the variable overview can be modified.
5	Lowest user group in which the variable overview can be displayed.
6	Column width in mm. Enter the desired value via the keypad and confirm it with the Enter key.
7	Row height in mm. Enter the desired value via the keypad and confirm it with the Enter key.

The following buttons are available:

Button	Description
<b>Display</b>	Switches to the variable overview.  (>>> 4.15.7 "Displaying the variable overview and modifying variables" Page 72)
<b>Insert</b>	Displays additional buttons: <ul style="list-style-type: none"><li>■ <b>Row above:</b> Inserts a new row above the one currently selected.</li><li>■ <b>Row below:</b> Inserts a new row below the one currently selected.</li><li>■ <b>Group before:</b> Inserts a new group to the left of the one currently selected.</li><li>■ <b>Group after:</b> Inserts a new group to the right of the one currently selected.</li></ul>
<b>Delete</b>	Displays additional buttons: <ul style="list-style-type: none"><li>■ <b>Row:</b> The selected row is deleted.</li><li>■ <b>Group:</b> The current group is deleted.</li></ul>

## 6.5 Changing the password

### Procedure

1. Select **Configuration > User group** in the main menu. The current user group is displayed.
2. Press **Login....**
3. Select the user group for which the password is to be changed.
4. Press **Password ....**
5. Enter the old password. Enter the new password twice.  
For security reasons, the entries are displayed encrypted. Upper and lower case are taken into consideration.
6. Press **OK**. The new password is valid immediately.

## 6.6 Configuring workspaces

Workspaces can be configured for a robot. These serve to protect the system.

A maximum of 8 Cartesian (=cubic) and 8 axis-specific workspaces can be configured at any one time. The workspaces can overlap.

(>>> 6.6.1 "Configuring Cartesian workspaces" Page 144)

(>>> 6.6.2 "Configuring axis-specific workspaces" Page 146)

There are 2 types of workspace:

- Non-permitted spaces. The robot may only move outside such a space.
- Permitted spaces. The robot must not move outside such a space.

Exactly what reactions occur when the robot violates a workspace depends on the configuration.

### 6.6.1 Configuring Cartesian workspaces



In the case of Cartesian workspaces, only the position of the TCP is monitored. It is not possible to monitor whether other parts of the robot violate the workspace.

#### Description

The following parameters define the position and size of a Cartesian workspace:

- Origin of the workspace relative to the WORLD coordinate system
- Dimensions of the workspace, starting from the origin

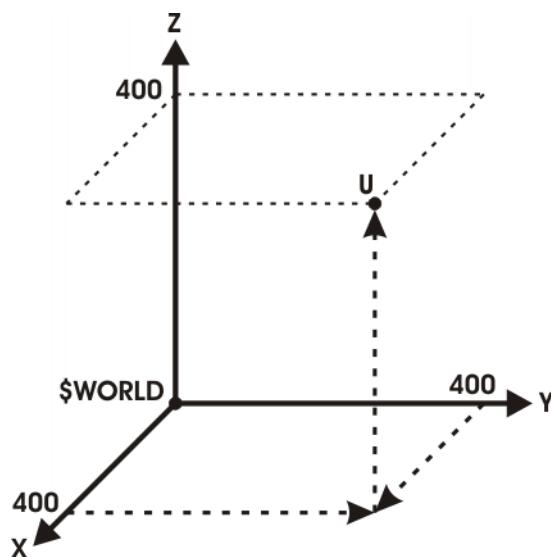
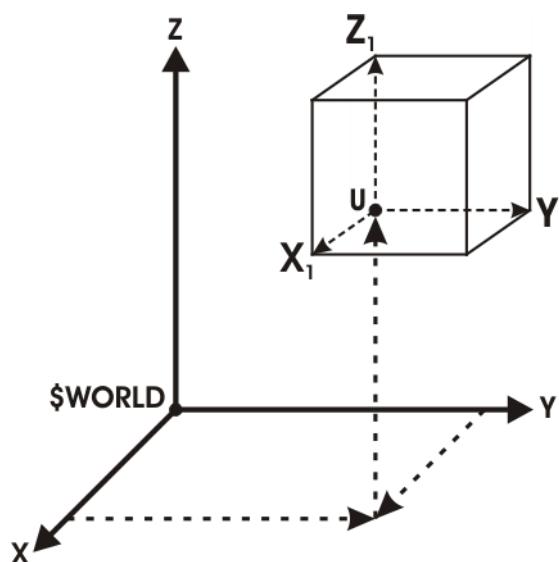


Fig. 6-11: Cartesian workspace, origin U



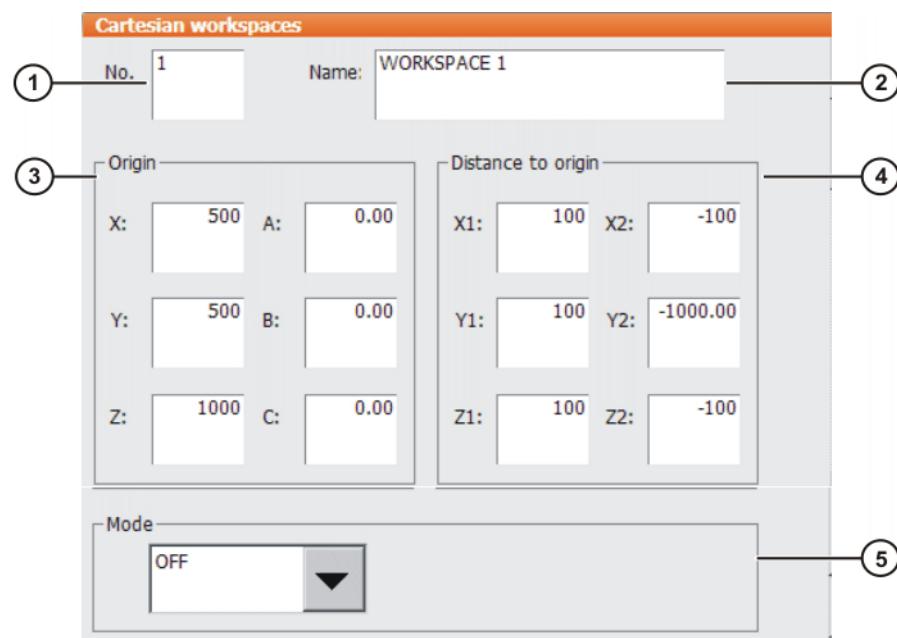
**Fig. 6-12: Cartesian workspace, dimensions**

**Precondition**

- User group "Expert".
- Operating mode T1 or T2.

**Procedure**

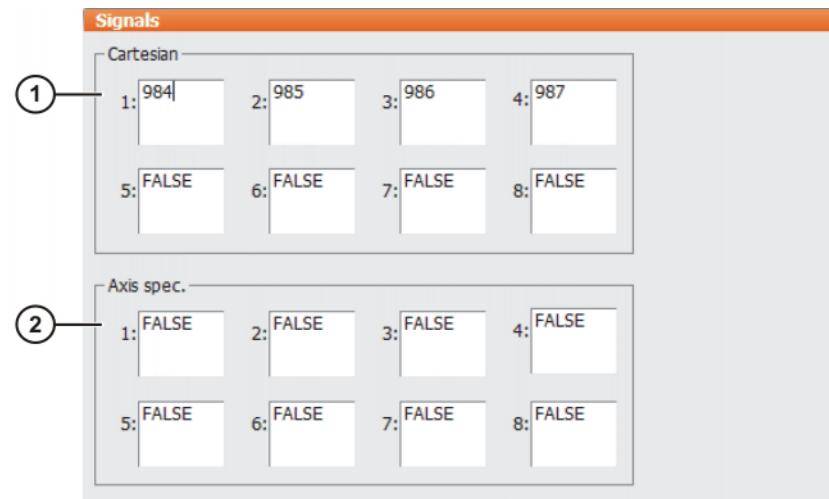
1. In the main menu, select **Configuration > Miscellaneous > Workspace monitoring > Configuration**.  
The **Cartesian workspaces** window is opened.
2. Enter values and press **Save**.
3. Press **Signal**. The **Signals** window is opened.
4. In the **Cartesian** group: next to the number of the workspace, enter the output that is to be set if the workspace is violated.
5. Press **Save**.
6. Close the window.



**Fig. 6-13: Configuring a Cartesian workspace**

Item	Description
1	Number of the workspace (max. 8)
2	Designation of the workspace

Item	Description
3	Origin and orientation of the workspace relative to the WORLD coordinate system
4	Dimensions of the workspace in mm
5	Mode ( <a href="#">6.6.3 "Mode for workspaces" Page 148</a> )



**Fig. 6-14: Workspace signals**

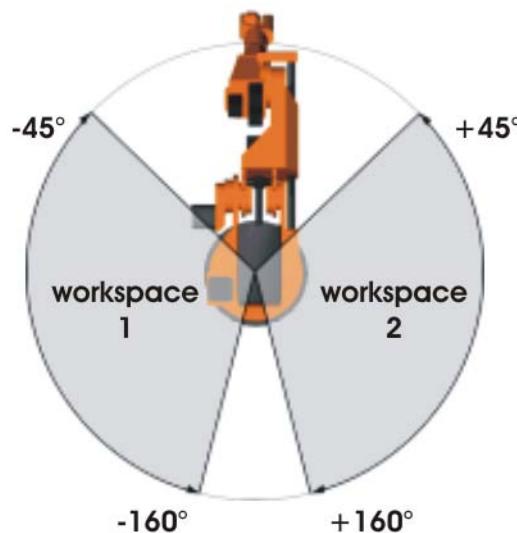
Item	Description
1	Outputs for the monitoring of the Cartesian workspaces
2	Outputs for the monitoring of the axis-specific workspaces

If no output is to be set when the workspace is violated, the value FALSE must be entered.

### 6.6.2 Configuring axis-specific workspaces

#### Description

Axis-specific workspaces can be used to restrict further the areas defined by the software limit switches in order to protect the robot, tool or workpiece.



**Fig. 6-15: Example of axis-specific workspaces for A1**

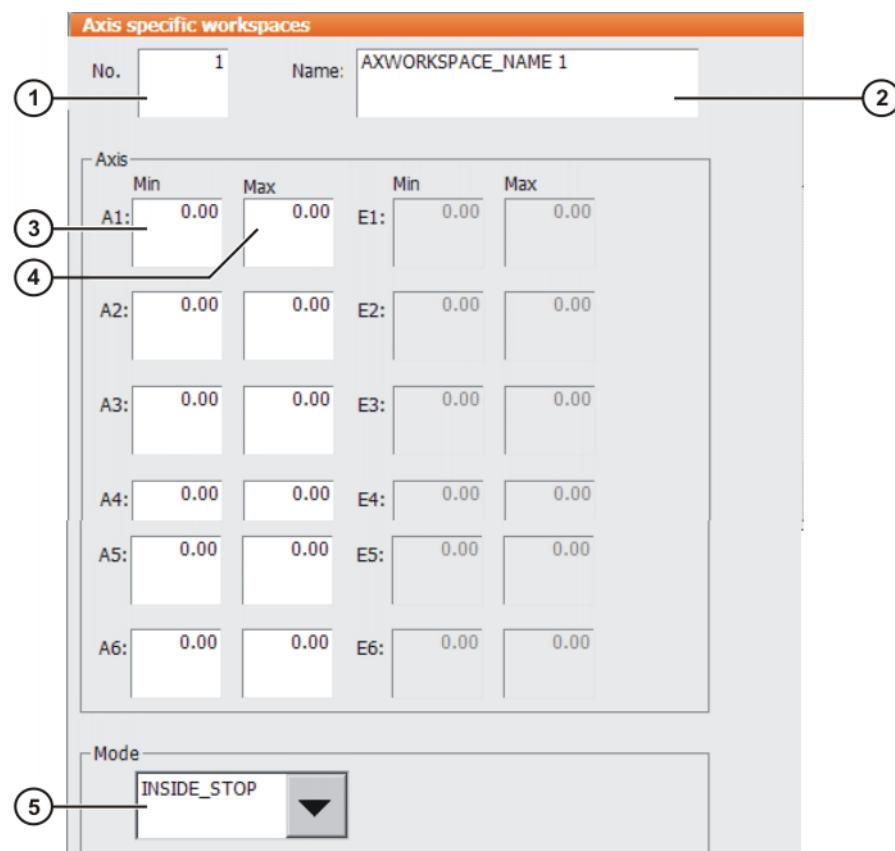
#### Precondition

- User group "Expert".

- Operating mode T1 or T2.

#### Procedure

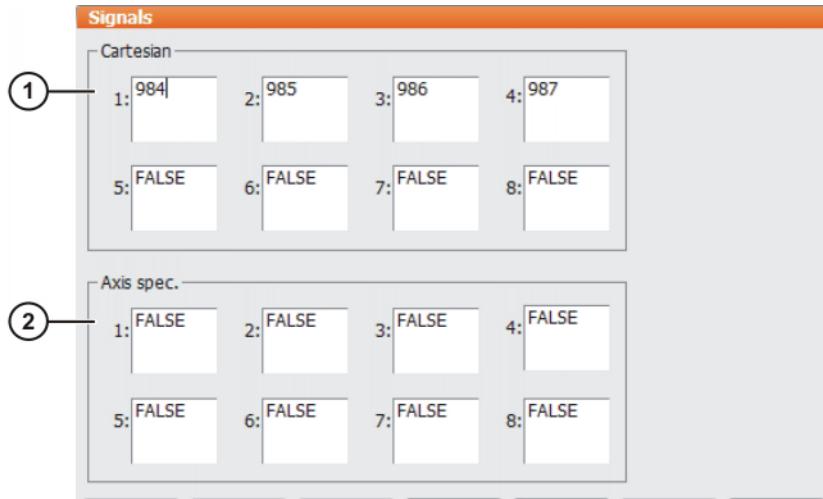
1. In the main menu, select **Configuration > Miscellaneous > Workspace monitoring > Configuration**.  
The **Cartesian workspaces** window is opened.
2. Press **Axis-spec.** to switch to the window **Axis-specific workspaces**.
3. Enter values and press **Save**.
4. Press **Signal**. The **Signals** window is opened.
5. In the **Axis-specific** group: next to the number of the workspace, enter the output that is to be set if the workspace is violated.
6. Press **Save**.
7. Close the window.



**Fig. 6-16: Configuring an axis-specific workspace**

Item	Description
1	Number of the workspace (max. 8)
2	Designation of the workspace
3	Lower limit for axis angle
4	Upper limit for axis angle
5	Mode (=> 6.6.3 "Mode for workspaces" Page 148)

If the value 0 is entered for an axis under Item 3 and Item 4, the axis is not monitored, irrespective of the mode.



**Fig. 6-17: Workspace signals**

Item	Description
1	Outputs for the monitoring of the Cartesian workspaces
2	Outputs for the monitoring of the axis-specific workspaces

If no output is to be set when the workspace is violated, the value FALSE must be entered.

### 6.6.3 Mode for workspaces

Mode	Description
#OFF	Workspace monitoring is deactivated.
#INSIDE	<ul style="list-style-type: none"> <li>■ Cartesian workspace: The defined output is set if the TCP or flange is located inside the workspace.</li> <li>■ Axis-specific workspace: The defined output is set if the axis is located inside the workspace.</li> </ul>
#OUTSIDE	<ul style="list-style-type: none"> <li>■ Cartesian workspace: The defined output is set if the TCP or flange is located outside the workspace.</li> <li>■ Axis-specific workspace: The defined output is set if the axis is located outside the workspace.</li> </ul>

Mode	Description
#INSIDE_STOP	<ul style="list-style-type: none"> <li>■ Cartesian workspace: The defined output is set if the TCP, flange or wrist root point is located inside the workspace. (Wrist root point = center point of axis A5)</li> <li>■ Axis-specific workspace: The defined output is set if the axis is located inside the workspace.</li> </ul> <p>The robot is also stopped and messages are displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed.</p> <p>(&gt;&gt;&gt; 4.14 "Bypassing workspace monitoring" Page 66)</p>
#OUTSIDE_STOP	<ul style="list-style-type: none"> <li>■ Cartesian workspace: The defined output is set if the TCP or flange is located outside the workspace.</li> <li>■ Axis-specific workspace: The defined output is set if the axis is located outside the workspace.</li> </ul> <p>The robot is also stopped and messages are displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed.</p> <p>(&gt;&gt;&gt; 4.14 "Bypassing workspace monitoring" Page 66)</p>

## 6.7 Warm-up

**Description** If a robot is started in low ambient temperatures, this results in increased friction in the gear unit. This can cause the motor current of an axis (or of more than one axis) to reach its maximum value. This stops the robot and the robot controller generates the error message *Regulator limit exceeded <axis number>*. To avoid this, the motor current can be monitored during the warm-up phase. If a defined value is reached, the robot controller reduces the motion velocity. This, in turn, reduces the motor current.



The monitoring refers to PTP motions and PTP-CP approximate positioning blocks. CP motions (including Spline) are not monitored and their velocity is not reduced.

### 6.7.1 Configuring warm-up

**Precondition** ■ Expert user group

**Procedure**

1. Open the file R1\Mada\\$machine.dat.
2. Set the corresponding system variables to the desired values.  
(>>> 6.7.3 "System variables for warm-up" Page 150)
3. Close the file. Respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

### 6.7.2 Warm-up sequence

**Precondition**

- \$WARMUP\_RED\_VEL =TRUE
- Operating mode AUT or AUT EXT
- The robot is considered to be cold. This applies in the following cases:
  - Cold start
  - Or \$COOLDOWN\_TIME has expired.

- Or \$WARMUP\_RED\_VEL has been set from FALSE to TRUE.

### Example

Sequence on the basis of the following example values in \$machine.dat:

```
BOOL $WARMUP_RED_VEL = TRUE
REAL $WARMUP_TIME = 30.0
REAL $COOLDOWN_TIME = 120.0
INT $WARMUP_CURR_LIMIT = 95
INT $WARMUP_MIN_FAC = 60
REAL $WARMUP_SLEW_RATE = 5.0
```

1. The cold robot starts. The motor currents are monitored for 30 minutes (\$WARMUP\_TIME).
2. If the motor current of an axis exceeds 95% (\$WARMUP\_CURR\_LIMIT) of the maximum permissible motor current, the monitoring is triggered. The robot controller then generates the message *Warm-up active* and reduces the internal override. The robot slows down and the motor current drops. The program override on the KCP remains unchanged! The internal override is reduced to a maximum of 60% (\$WARMUP\_MIN\_FAC) of the programmed override. There is no way of influencing how quickly the internal override is reduced.
3. Once the monitoring is no longer triggered, the robot controller increases the internal override again. This is generally the case before the minimum \$WARMUP\_MIN\_FAC has been reached! The robot accelerates again. Once a second, the robot controller edges back up towards the programmed override. \$WARMUP\_SLEW\_RATE determines the rate of increase. In the example, the internal override is increased by 5% per second.
4. It is possible that the robot is still not warm enough and that the motor current thus exceeds the maximum \$WARMUP\_CURR\_LIMIT once again. The robot controller reacts (within \$WARMUP\_TIME) the same way as the first time.
5. If the robot is warm enough for the robot controller to increase the internal override all the way back up to the programmed override, the robot controller deactivates the message *Warm-up active*.
6. After 30 minutes (\$WARMUP\_TIME), the robot is deemed to be warmed up and the motor currents are no longer monitored.

### LOG file

The following events are logged in the file "Warmup.LOG" (path "KRC:\Robot\er\Log\"):

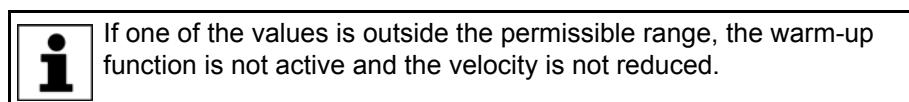
Entry	Meaning
<b>Monitoring active</b>	The motor currents are monitored.
<b>Monitoring inactive</b>	The motor currents are not monitored.
<b>Controlling active</b>	The velocity is reduced.
<b>Controlling inactive</b>	The velocity corresponds to the programmed override once again.

Example:

```
...
Date: 21.08.08 Time: 14:46:57 State: Monitoring active
Date: 21.08.08 Time: 14:54:06 State: Controlling active
Date: 21.08.08 Time: 14:54:07 State: Controlling inactive
Date: 21.08.08 Time: 18:23:43 State: Monitoring inactive
...
```

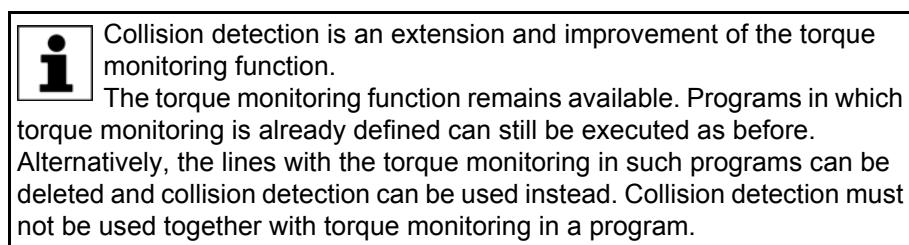
### 6.7.3 System variables for warm-up

The system variables for the warm-up can only be modified in the file \$Machine.dat (path KRC:\R1\MADA).



System variable	Description
\$WARMUP_RED_VEL	<ul style="list-style-type: none"> <li>■ <b>TRUE:</b> Warm-up functionality is activated.</li> <li>■ <b>FALSE:</b> Warm-up functionality is deactivated (default).</li> </ul> <p>If \$WARMUP_RED_VEL is set from FALSE to TRUE, this sets the runtime of the robot to zero. The robot is deemed to be cold, irrespective of how long it was previously under servo control.</p>
\$WARMUP_TIME	<p>Time during which the motor currents are monitored by the warm-up function.</p> <p>If the cold robot is started, a runtime value is incremented. If the robot is not under servo control, the value is decremented. If the runtime is greater than \$WARMUP_TIME, the robot is deemed to be warmed up and the motor currents are no longer monitored.</p> <ul style="list-style-type: none"> <li>■ &gt; 0.0 min</li> </ul>
\$COOLDOWN_TIME	<p>If the warm robot is not under servo control, a standstill value is incremented. If the robot is under servo control, the value is decremented. If the standstill time is greater than \$COOLDOWN_TIME, the robot is deemed to be cold and the motor currents are monitored. (Precondition: \$WARMUP_RED_VEL = TRUE.)</p> <p>If the controller of a warm robot is shut down and restarted with a warm restart, the time the controller was switched off is counted as standstill time.</p> <ul style="list-style-type: none"> <li>■ &gt; 0.0 min</li> </ul>
\$WARMUP_CURR_LIMIT	<p>Maximum permissible motor current during warm-up (relative to the regular maximum permissible motor current)</p> <p>Regular maximum permissible motor current = <math>(\\$CURREN\_LIM * \\$CURREN\_MAX) / 100</math></p> <ul style="list-style-type: none"> <li>■ 0 ... 100%</li> </ul>
\$WARMUP_MIN_FAC	<p>Minimum for the override reduction due to the warm-up function</p> <p>The internal override is reduced at most to the factor of the programmed override defined here.</p> <ul style="list-style-type: none"> <li>■ 0 ... 100%</li> </ul>
\$WARMUP_SLEW_RATE	<p>Rate of increase for the increase in velocity</p> <p>Once the monitoring is no longer triggered, the robot controller increases the internal override again. The rate of increase is defined here.</p> <ul style="list-style-type: none"> <li>■ &gt; 0.0%/s</li> </ul>

## 6.8 Collision detection



<b>Description</b>	<p>If the robot collides with an object, the robot controller increases the axis torques in order to overcome the resistance. This can result in damage to the robot, tool or other objects.</p> <p>Collision detection reduces the risk of such damage. It monitors the axis torques. If these deviate from a specified tolerance range, the following reactions are triggered:</p> <ul style="list-style-type: none"><li>■ The robot stops with a STOP 1.</li><li>■ The robot controller calls the program tm_useraction. This is located in the Program folder and contains the HALT statement. Alternatively, the user can program other reactions in the program tm_useraction.</li></ul> <p>The robot controller automatically calculates the tolerance range. (Exception: no values are calculated in T1 mode.) A program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. The user can define an offset via the user interface for the tolerance range calculated by the robot controller.</p> <p>If the robot is not operated for a longer period (e.g. over the weekend), the motors, gear units, etc., cool down. Different axis torques are required in the first few runs after such a break than in the case of a robot that is already at operating temperature. The robot controller automatically adapts the collision detection to the changed temperature.</p>
<b>Precondition</b>	<ul style="list-style-type: none"><li>■ In order to be able to use the collision detection function, acceleration adaptation must be activated. Acceleration adaptation is activated when system variable \$ADAP_ACC is not equal to #NONE. (This is the default setting.) The system variable can be found in the file C:\KRC\Roboter\KRCR1\MaDa\\$ROBCOR.DAT.</li><li>■ The tolerance range is only calculated for motion blocks that have been executed completely.</li><li>■ To activate collision detection for a motion, the parameter <b>Collision detection</b> must be set to TRUE during programming. This can be seen from the addition CD in the program code: <pre>PTP P2 Vel= 100 % PDAT1 Tool[1] Base[1] CD</pre></li></ul>
<b>Restrictions</b>	<ul style="list-style-type: none"><li>■ Collision detection is not possible for HOME positions and other global positions.</li><li>■ Collision detection is not possible for external axes.</li><li>■ Collision detection is not possible during backward motion.</li><li>■ Collision detection is not possible in T1 mode.</li><li>■ High axis torques arise when the stationary robot starts to move. For this reason, the axis torques are not monitored in the starting phase (approx. 700 ms).</li><li>■ The collision detection function reacts much less sensitively for the first 2 or 3 program executions after the program override value has been modified. Thereafter, the robot controller has adapted the tolerance range to the new program override.</li></ul>
<b>System variables</b>	



The parameter **Collision detection** is only available if the motion is programmed via an inline form.  
Information about collision detection for motions programmed without inline forms can be found in the documentation "KUKA.ExpertTech".

System variable	Description
\$TORQ_DIFF	The values of \$TORQ_DIFF (torque) and \$TORQ_DIFF2 (impact) are automatically calculated during program execution. These values are compared with the values from the previous program execution or with the default values. The highest value is saved. The values are always calculated, even when collision detection is deactivated.
\$TORQ_DIFF2	If collision detection is active, the system compares the values of \$TORQ_DIFF and \$TORQ_DIFF2 with the saved values during the motion.
\$TORQMON_DEF[1] ... \$TORQMON_DEF[6]	Values for the tolerance range in program mode (per axis)* File C:\KRC\Roboter\KRC\MaDa\\$CUSTOM.DAT
\$TORQMON_COM_DEF[1] ... \$TORQMON_COM_DEF[6]	Values for the tolerance range in jog mode (per axis)* File C:\KRC\Roboter\KRC\MaDa\\$CUSTOM.DAT
\$COLL_ENABLE	Signal declaration. This output is set if the value of one of the \$TORQMON_DEF[...] variables is less than 200. File: KRC:\STEU\Mada\\$machine.dat
\$COLL_ALARM	Signal declaration. This output is set if message 117 "Collision Detection axis <axis number>" is generated. The output remains set as long as \$STOPMESS is active. File: KRC:\STEU\Mada\\$machine.dat
\$TORQMON_TIME	Response time for the collision detection. Unit: milliseconds. Default value: 0.0 File: C:\KRC\Roboter\KRC\Steu\Mada\\$CUSTOM.DAT.

\*The width of the tolerance range is equal to the maximum torque [Nm] multiplied by the value in \$TORQMON\_.... . The default value is 200. Unit: percent.

### 6.8.1 Calculating the tolerance range and activating collision detection

- Precondition**
- The acceleration adaptation is switched on.
  - The load data have been entered correctly.
  - In the program, the parameter **Collision detection** is set to TRUE for all motions that are to be monitored.
  - If required: the desired collision response has been programmed in the program tm\_useraction.

- Procedure**
1. In the main menu, select **Configuration > Miscellaneous > Collision detection**.  
(>>> 6.8.3 "Option window "Collision detection"" Page 155)
  2. The box **KCP** must contain the entry **MonOff**. If this is not the case, press **Deactivate**.
  3. Start the program and execute it several times. After 2 or 3 program executions, the robot controller has calculated a practicable tolerance range.
  4. Press **Activate**. The box **KCP** in the **Collision detection** window now contains the entry **MonOn**.
- Save the configuration by pressing **Close**.

If required, the user can define an offset for the tolerance range.  
(>>> 6.8.2 "Defining an offset for the tolerance range" Page 154)



The tolerance range must be recalculated in the following cases:

- The velocity has been modified.
- Points have been changed, added or removed.

### 6.8.2 Defining an offset for the tolerance range

#### Description

An offset for the torque and for the impact can be defined for the tolerance range. The lower the offset, the more sensitive the reaction of the collision detection. The higher the offset, the less sensitive the reaction of the collision detection.

**Torque:** The torque is effective if the robot meets a continuous resistance. Examples:

- The robot collides with a wall and pushes against the wall.
- The robot collides with a container. The robot pushes against the container and moves it.

**Impact:** The impact is effective if the robot meets a brief resistance. Example:

- The robot collides with a panel which is sent flying by the impact.



If the collision detection reacts too sensitively, do not immediately increase the offset. Instead, recalculate the tolerance range first and test whether the collision detection now reacts as desired.

(>>> 6.8.1 "Calculating the tolerance range and activating collision detection" Page 153)

#### Procedure

1. Select program.
2. In the main menu, select **Configuration > Miscellaneous > Collision detection**.  
(>>> 6.8.3 "Option window "Collision detection"" Page 155)
3. The offset for a motion can be modified while a program is running:  
If the desired motion is displayed in the **Collision detection** window, press an arrow key in the **Torque** or **Impact** box. The window remains focused on the motion and the offset can be modified.  
Alternatively, a block selection to the desired motion can be carried out.
4. Save the change by pressing **Save**.
5. Save the configuration by pressing **Close**.
6. Set the original operating mode and program run mode.

### 6.8.3 Option window “Collision detection”

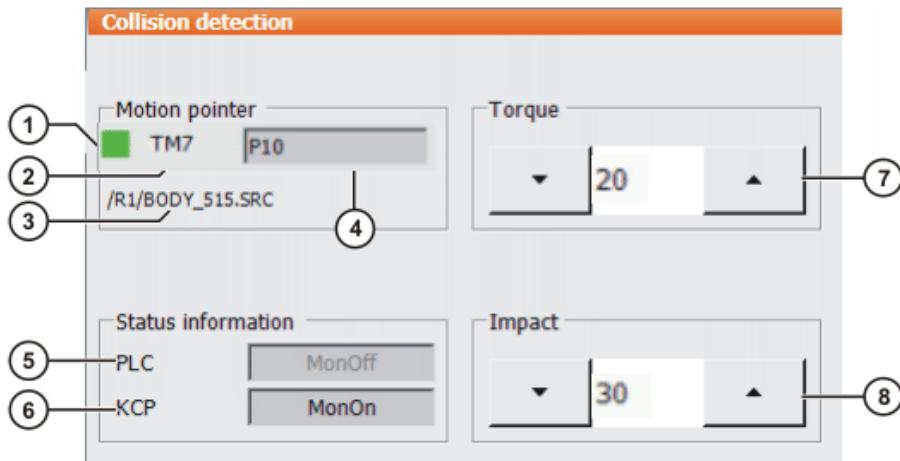


Fig. 6-18: Option window “Collision detection”



The values in the option window **Collision detection** do not always refer to the current motion. Deviations are particularly possible in the case of points which are close together and approximated motions.

Item	Description
1	Indicates the status of the current motion: <ul style="list-style-type: none"> <li>■ Red: the current motion is not monitored.</li> <li>■ Green: the current motion is monitored.</li> <li>■ Orange: an arrow key has been pressed in the <b>Torque</b> or <b>Impact</b> box. The window remains focused on the motion and the offset can be modified. The change can then be applied by pressing <b>Save</b>.</li> <li>■ Pixelated: a program must generally be executed 2 or 3 times before the robot controller has calculated a practicable tolerance range. This display is pixelated during this learning phase.</li> </ul>
2	Number of the TMx variable  The robot controller creates a TMx variable for each motion block in which the parameter <b>Collision detection</b> is set to TRUE. TMx contains all the values for the tolerance range of this motion block. If 2 motion blocks refer to the same point Px, the robot controller creates 2 TMx variables.
3	Path and name of the selected program
4	Point name
5	This box is only active in “Automatic External” mode. It appears gray in all other modes.  <b>MonOn:</b> collision detection has been activated by the PLC.  If collision detection is activated by the PLC, the PLC sends the input signal <b>sTQM_SPSACTIVE</b> to the robot controller. The robot controller responds with the output signal <b>sTQM_SPSSTATUS</b> . The signals are defined in the file \$config.dat.  <b>Note:</b> Collision detection is only active in Automatic External mode if both the <b>PLC</b> box and the <b>KCP</b> box show the entry <b>MonOn</b> .

Item	Description
6	<b>MonOn:</b> collision detection has been activated from the KCP. <b>Note:</b> Collision detection is only active in Automatic External mode if both the <b>PLC</b> box and the <b>KCP</b> box show the entry <b>MonOn</b> .
7	Offset for the torque. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 20. If an arrow key is pressed, the window remains focused on the motion and the offset can be modified. The change can then be applied by pressing <b>Save</b> . ■ <b>N.A.:</b> the option <b>Collision detection</b> in the inline form is set to FALSE for this motion.
8	Offset for the impact. The lower the offset, the more sensitive the reaction of the collision detection. Default value: 30. If an arrow key is pressed, the window remains focused on the motion and the offset can be modified. The change can then be applied by pressing <b>Save</b> . ■ <b>N.A.:</b> the option <b>Collision detection</b> in the inline form is set to FALSE for this motion.

Button	Description
<b>Activate</b>	Activates collision detection. This button is not displayed if the torque or impact has been changed, but the changes have not yet been saved.
<b>Deactivate</b>	Deactivates collision detection. This button is not displayed if the torque or impact has been changed, but the changes have not yet been saved.
<b>Save</b>	Saves changes to the torque and/or impact.
<b>Cancel</b>	Rejects changes to the torque and/or impact.

#### 6.8.4 Torque monitoring



Collision detection is an extension and improvement of the torque monitoring function.

The torque monitoring function remains available. Programs in which torque monitoring is already defined can still be executed as before. Alternatively, the lines with the torque monitoring in such programs can be deleted and collision detection can be used instead. Collision detection must not be used together with torque monitoring in a program.

##### Description

Differences between torque monitoring and collision detection:

- The tolerance range is not automatically calculated by the robot controller, but must be defined by the user.
- The tolerance range only refers to the torque. No values can be defined for the impact.
- The robot controller cannot automatically adapt the tolerance range to changed temperatures.
- If a collision is detected, the robot stops with a STOP 1. It is not possible to call a user-defined program.

**Overview**

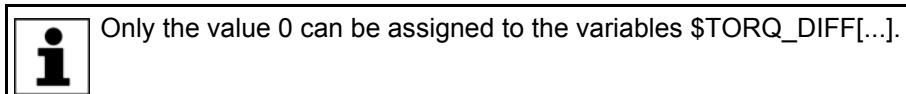
Step	Description
1	Determine suitable values for torque monitoring. (>>> 6.8.4.1 "Determining values for torque monitoring" Page 157)
2	Program torque monitoring. (>>> 6.8.4.2 "Programming torque monitoring" Page 157)

**6.8.4.1 Determining values for torque monitoring**

**Description** The maximum torque deviation that has occurred can be determined as a percentage by means of the system variable \$TORQ\_DIFF[...].

**Procedure**

1. In the main menu, select **Display > Variable > Single**.
2. Set the value of the variable \$TORQ\_DIFF [...] to 0.
3. Execute the motion block and read the variable again. The value corresponds to the maximum torque deviation.
4. Set the variable for the monitoring of the axis to this value plus a safety margin of 5 - 10%.

**6.8.4.2 Programming torque monitoring**

**Precondition**

- In order to be able to use the collision detection function, acceleration adaptation must be activated. Acceleration adaptation is activated when system variable \$ADAP\_ACC is **not equal to #NONE**. (This is the default setting.) The system variable can be found in the file C:\KRC\Robot-er\KRC\R1\MaDa\\$ROBCOR.DAT.
- A program is selected.

**Procedure**

1. Position the cursor in the line before the motion for which the torque monitoring is to be programmed.
2. Select the menu sequence **Commands > Motion parameters > Torque monitoring**. An inline form is opened.

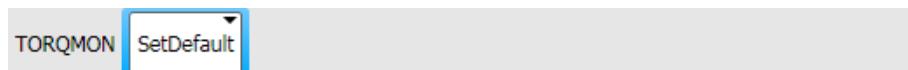


Fig. 6-19

3. In the **TORQMON** box, select the entry **SetLimits**.

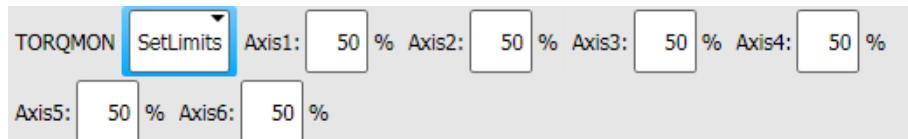


Fig. 6-20

4. For each axis, enter the amount by which the command torque may deviate from the actual torque.
5. Press **Cmd OK**.
6. If a response time for the torque monitoring is to be defined:  
Set the variable \$TORQMON\_TIME to the desired value. Unit: milliseconds. Default value: 0.



The values are automatically reset to the default value 200 in the following cases:

- Reset
- Block selection
- Program deselection

## 6.9 Defining calibration tolerances



Only modify the default values in exceptional cases. Otherwise, increased error messages and inaccuracy may result.

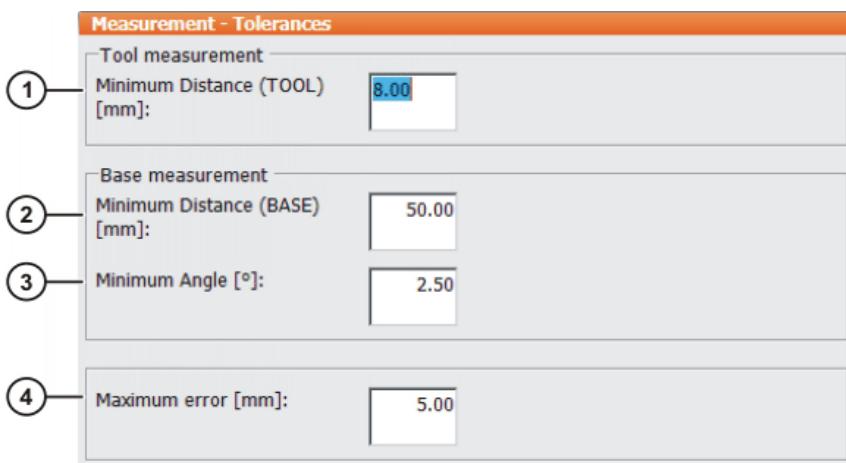
### Precondition

- Expert user group

### Procedure

- In the main menu, select **Start-up > Calibrate > Tolerances**.

### Description



**Fig. 6-21: Default error tolerances**

Item	Description
1	The minimum distance for tool calibration. ■ <b>0 ... 200 mm</b>
2	The minimum distance for base calibration. ■ <b>0 ... 200 mm</b>
3	The minimum angle between the straight lines through the 3 calibration points in base calibration. ■ <b>0 ... 360°</b>
4	Maximum error in calculation. ■ <b>0 ... 200 mm</b>

The following buttons are available:

Button	Description
<b>Default</b>	Restores the default settings. The data must then be saved by pressing <b>OK</b> .

## 6.10 Configuring Automatic External

### Description

If robot processes are to be controlled centrally by a higher-level controller (e.g. a PLC), this is carried out using the Automatic External interface.

The higher-level controller transmits the signals for the robot processes (e.g. motion enable, fault acknowledgement, program start, etc.) to the robot controller via the Automatic External interface. The robot controller transmits information about operating states and fault states to the higher-level controller.

## Overview

To enable use of the Automatic External interface, the following configurations must be carried out:

Step	Description
1	Configuration of the CELL.SRC program. (>>> 6.10.1 "Configuring CELL.SRC" Page 159)
2	Configuration of the inputs/outputs of the Automatic External interface. (>>> 6.10.2 "Configuring Automatic External inputs/outputs" Page 160)
3	Only if error numbers are to be transmitted to the higher-level controller: configuration of the P00.DAT file. (>>> 6.10.3 "Transmitting error numbers to the higher-level controller" Page 166)

### 6.10.1 Configuring CELL.SRC

**Description** In Automatic External mode, programs are called using the program CELL.SRC.

#### Program

```

1 DEF CELL ( )
...
6   INIT
7   BASISTECHINI
8   CHECK HOME
9   PTP HOME Vel= 100 % DEFAULT
10  AUTOEXTINI
11  LOOP
12    P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
13    SWITCH PGNO ; Select with Programnumber
14
15    CASE 1
16      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17      ;EXAMPLE1 ( ) ; Call User-Program
18
19    CASE 2
20      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
21      ;EXAMPLE2 ( ) ; Call User-Program
22
23    CASE 3
24      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
25      ;EXAMPLE3 ( ) ; Call User-Program
26
27    DEFAULT
28      P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
29      ENDSWITCH
30    ENDLOOP
31  END

```

Line	Description
12	The robot controller calls the program number from the higher-level controller.
15	CASE branch for program number = 1
16	Receipt of program number 1 is communicated to the higher-level controller.
17	The user-defined program EXAMPLE1 is called.

Line	Description
27	DEFAULT = the program number is invalid.
28	Error treatment in the case of an invalid program number

**Precondition**

- Expert user group

**Procedure**

1. Open the program CELL.SRC in the Navigator. (This program is located in the folder "R1".)
2. In the section CASE 1, replace the name EXAMPLE1 with the name of the program that is to be called via program number 1. Delete the semicolon in front of the name.

```
...
15      CASE 1
16      P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17      MY_PROGRAM ( ) ; Call User-Program
...
```

3. For all further programs, proceed as described in step 2. If required, add additional CASE branches.
4. Close the program CELL.SRC. Respond to the request for confirmation asking whether the changes should be saved with **Yes**.

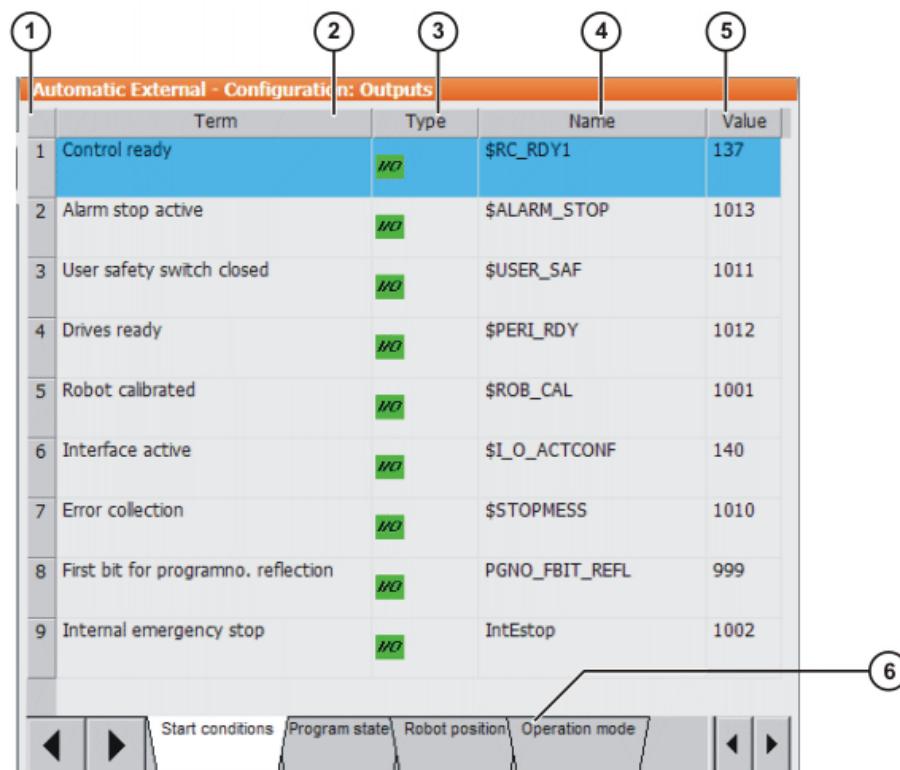
#### 6.10.2 Configuring Automatic External inputs/outputs

**Procedure**

1. In the main menu, select **Configuration > Inputs/outputs > Automatic External**.
2. In the **Value** column, select the cell to be edited and press **Edit**.
3. Enter the desired value and save it by pressing **OK**.
4. Repeat steps 2 and 3 for all values to be edited.
5. Close the window. The changes are saved.

**Description**


Automatic External - Configuration: Inputs				
	Term	Type	Name	Value
1	Type programno.	Var	PGNO_TYPE	1
2	programno. reflection	Var	REFLECT_PROG_NR	0
3	Bitwidth programno.	Var	PGNO_LENGTH	8
4	First bit programno.	IO	PGNO_FBIT	33
5	Parity bit	IO	PGNO_PARITY	41
6	Programno. valid	IO	PGNO_VALID	42
7	Programstart	IO	\$EXT_START	1026
8	Move enable	IO	\$MOVE_ENABLE	1025
9	Error confirmation	IO	\$CONF_MESS	1026
10	Drives off (invers)	IO	\$DRIVES_OFF	1025
11	Drives on	IO	\$DRIVES_ON	140
12	Activate interface	IO	\$I_O_ACT	1025

**Fig. 6-22: Configuring Automatic External inputs**


Automatic External - Configuration: Outputs				
	Term	Type	Name	Value
1	Control ready	IO	\$RC_RDY1	137
2	Alarm stop active	IO	\$ALARM_STOP	1013
3	User safety switch closed	IO	\$USER_SAF	1011
4	Drives ready	IO	\$PERI_RDY	1012
5	Robot calibrated	IO	\$ROB_CAL	1001
6	Interface active	IO	\$I_O_ACTCONF	140
7	Error collection	IO	\$STOPMESS	1010
8	First bit for programno. reflection	IO	PGNO_FBIT_REFL	999
9	Internal emergency stop	IO	IntEstop	1002

**Fig. 6-23: Configuring Automatic External outputs**

Item	Description
1	Number
2	Long text name of the input/output

Item	Description
3	Type ■ <b>Green:</b> Input/output ■ <b>Yellow:</b> variable or system variable (\$...)
4	Name of the signal or variable
5	Input/output number or channel number
6	The outputs are thematically assigned to tabs.

(>>> 6.10.2.1 "Automatic External inputs" Page 162)

(>>> 6.10.2.2 "Automatic External outputs" Page 164)

### 6.10.2.1 Automatic External inputs

#### PGNO\_TYPE

Type: variable

This variable defines the format in which the program number sent by the higher-level controller is read.

Value	Description	Example
1	Read as binary number.  The program number is transmitted by the higher-level controller as a binary coded integer.	0 0 1 0 0 1 1 1 => PGNO = 39
2	Read as BCD value.  The program number is transmitted by the higher-level controller as a binary coded decimal.	0 0 1 0 0 1 1 1 => PGNO = 27
3	Read as "1 of n".  The program number is transmitted by the higher-level controller or the periphery as a "1 of n" coded value.	0 0 0 0 0 0 0 1 => PGNO = 1 0 0 0 0 1 0 0 0 => PGNO = 4

\* When using this transmission format, the values of PGNO\_REQ, PGNO\_PARITY and PGNO\_VALID are not evaluated and are thus of no significance.

#### REFLECT\_PROG\_NR

Type: variable

This variable defines whether the program number is to be mirrored to an output range. The output of the signal starts with the output defined using PGNO\_FBIT\_REFL. (>>> 6.10.2.2 "Automatic External outputs" Page 164)

Value	Description
0	Function deactivated
1	Function activated

#### PGNO\_LENGTH

Type: variable

This variable determines the number of bits in the program number sent by the higher-level controller. Range of values: 1 ... 16.

Example: PGNO\_LENGTH = 4 => the external program number is 4 bits long.

If PGNO\_TYPE has the value 2, only 4, 8, 12 and 16 are permissible values for the number of bits.

#### PGNO\_FBIT

Input representing the first bit of the program number. Range of values: 1 ... 4096.

Example: PGNO\_FBIT = 5 => the external program number begins with the input \$IN[5].

#### **PGNO\_PARITY**

Input to which the parity bit is transferred from the higher-level controller.

Input	Function
Negative value	Odd parity
0	No evaluation
Positive value	Even parity

If PGNO\_TYPE has the value 3, PGNO\_PARITY is not evaluated.

#### **PGNO\_VALID**

Input to which the command to read the program number is transferred from the higher-level controller.

Input	Function
Negative value	Number is transferred at the falling edge of the signal.
0	Number is transferred at the rising edge of the signal on the EXT_START line.
Positive value	Number is transferred at the rising edge of the signal.

If PGNO\_TYPE has the value 3, PGNO\_VALID is not evaluated.

#### **\$EXT\_START**

If the I/O interface is active, this input can be set to start or continue a program.



Only the rising edge of the signal is evaluated.



**WARNING** There is no BCO run in Automatic External mode. This means that the robot moves to the first programmed position after the start at the programmed (not reduced) velocity and does not stop there.

#### **\$MOVE\_ENABLE**

This input is used by the higher-level controller to check the robot drives.

Signal	Function
TRUE	Jogging and program execution are possible.
FALSE	All drives are stopped and all active commands inhibited.



If the drives have been switched off by the higher-level controller, the message "GENERAL MOTION ENABLE" is displayed. It is only possible to move the robot again once this message has been reset and another external start signal has been given.



During commissioning, the variable \$MOVE\_ENABLE is often configured with the value \$IN[1025]. If a different input is not subsequently configured, no external start is possible.

#### **\$CHCK\_MOVENA**

Type: variable

If the variable \$CHCK\_MOVENA has the value FALSE, \$MOVE\_ENABLE can be bypassed. The value of the variable can only be changed in the file C:\KRC\ROBOTER\KRC\STEU\Mada\\$OPTION.DAT.

Signal	Function
TRUE	MOVE_ENABLE monitoring is activated.
FALSE	MOVE_ENABLE monitoring is deactivated.



In order to be able to use MOVE\_ENABLE monitoring, \$MOVE\_ENABLE must have been configured with the input \$IN[1025]. Otherwise, \$CHCK\_MOVENA has no effect.

#### \$CONF\_MESS

Setting this input enables the higher-level controller to acknowledge error messages automatically as soon as the cause of the error has been eliminated.



Only the rising edge of the signal is evaluated.

#### \$DRIVES\_OFF

If there is a low-level pulse of at least 20 ms duration at this input, the higher-level controller switches off the robot drives.

#### \$DRIVES\_ON

If there is a high-level pulse of at least 20 ms duration at this input, the higher-level controller switches on the robot drives.

#### \$I\_O\_ACT

If this input is TRUE, the Automatic External interface is active. Default setting: \$IN[1025].

#### 6.10.2.2 Automatic External outputs

##### \$RC\_RDY1

Ready for program start.

##### \$ALARM\_STOP

This output is reset in the following EMERGENCY STOP situations:

- The EMERGENCY STOP button on the KCP is pressed.
- External E-STOP



In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs \$ALARM\_STOP and Int. E-Stop:

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- \$ALARM\_STOP is FALSE, Int. E-Stop is TRUE: external EMERGENCY STOP.

##### \$USER\_SAF

This output is reset if the safety fence monitoring switch is opened (AUTO mode) or an enabling switch is released (T1 or T2 mode).

##### \$PERI\_RDY

By setting this output, the robot controller communicates to the higher-level controller the fact that the robot drives are switched on.

##### \$ROB\_CAL

The signal is FALSE as soon as a robot axis has been unmastered

##### \$I\_O\_ACTCONF

This output is TRUE if Automatic External mode is selected and the input \$I\_O\_ACT is TRUE.

##### \$STOPMESS

This output is set by the robot controller in order to communicate to the higher-level controller any message occurring which requires the robot to be stopped. (Examples: EMERGENCY STOP, Driving condition or Operator safety)

##### PGNO\_FBIT\_REF L

Output representing the first bit of the program number. Precondition: The variable REFLECT\_PROG\_NR has the value 1. ([>>> 6.10.2.1 "Automatic External inputs" Page 162](#))

The size of the output area depends on the number of bits defining the program number (PGNO\_LENGTH).

If a program selected by the PLC is deselected by the user, the output area starting with PGNO\_FBIT\_REFI is set to FALSE. In this way, the PLC can prevent a program from being restarted manually.

PGNO\_FBIT\_REFI is also set to FALSE if the interpreter is situated in the CELL program.

#### Int. E-Stop

This output is set to FALSE if the EMERGENCY STOP button on the KCP is pressed.



In the case of an EMERGENCY STOP, the nature of the EMERGENCY STOP can be recognized from the states of the outputs **\$ALARM\_STOP** and **Int. E-Stop**:

- Both outputs are FALSE: the EMERGENCY STOP was triggered on the KCP.
- **\$ALARM\_STOP** is FALSE, **Int. E-Stop** is TRUE: external EMERGENCY STOP.

#### \$PRO\_ACT

This output is always set if a process is active at robot level. The process is therefore active as long as a program or an interrupt is being processed. Program processing is set to the inactive state at the end of the program only after all pulse outputs and all triggers have been processed.

In the event of an error stop, a distinction must be made between the following possibilities:

- If interrupts have been activated but not processed at the time of the error stop, the process is regarded as inactive (\$PRO\_ACT=FALSE).
- If interrupts have been activated and processed at the time of the error stop, the process is regarded as active (\$PRO\_ACT=TRUE) until the interrupt program is completed or a STOP occurs in it (\$PRO\_ACT=FALSE).
- If interrupts have been activated and a STOP occurs in the program, the process is regarded as inactive (\$PRO\_ACT=FALSE). If, after this, an interrupt condition is met, the process is regarded as active (\$PRO\_ACT=TRUE) until the interrupt program is completed or a STOP occurs in it (\$PRO\_ACT=FALSE).

#### PGNO\_REQ

A change of signal at this output requests the higher-level controller to send a program number.

If PGNO\_TYPE has the value 3, PGNO\_REQ is not evaluated.

#### APPL\_RUN

By setting this output, the robot controller communicates to the higher-level controller the fact that a program is currently being executed.

#### \$PRO\_MOVE

Means that a synchronous axis is moving, including in jog mode. The signal is thus the inverse of \$ROB\_STOPPED.

#### \$IN\_HOME

This output communicates to the higher-level controller whether or not the robot is in its HOME position.

#### \$ON\_PATH

This output remains set as long as the robot stays on its programmed path. The output ON\_PATH is set after the BCO run. This output remains set until the robot leaves the path, the program is reset or block selection is carried out. The ON\_PATH signal has no tolerance window, however; as soon as the robot leaves the path the signal is reset.

#### \$NEAR\_POSRET

This signal allows the higher-level controller to determine whether or not the robot is situated within a sphere about the position saved in \$POS\_RET. The higher-level controller can use this information to decide whether or not the program may be restarted.

The user can define the radius of the sphere in the file \$CUSTOM.DAT using the system variable \$NEARPATHTOL.

<b>\$ROB_STOPPED</b>	The signal is set when the robot is at a standstill. In the event of a WAIT statement, this output is set during the wait.  The signal is thus the inverse of \$PRO_MOVE.
<b>\$T1, \$T2, \$AUT, \$EXT</b>	These outputs are set when the corresponding operating mode is selected.

### 6.10.3 Transmitting error numbers to the higher-level controller

Error numbers of the robot controller in the range 1 to 255 can be transmitted to the higher-level controller. To transmit the error numbers, the file P00.DAT, in the directory C:\KRC\ROBOTER\KRC\R1\TP, must be configured as follows:

```

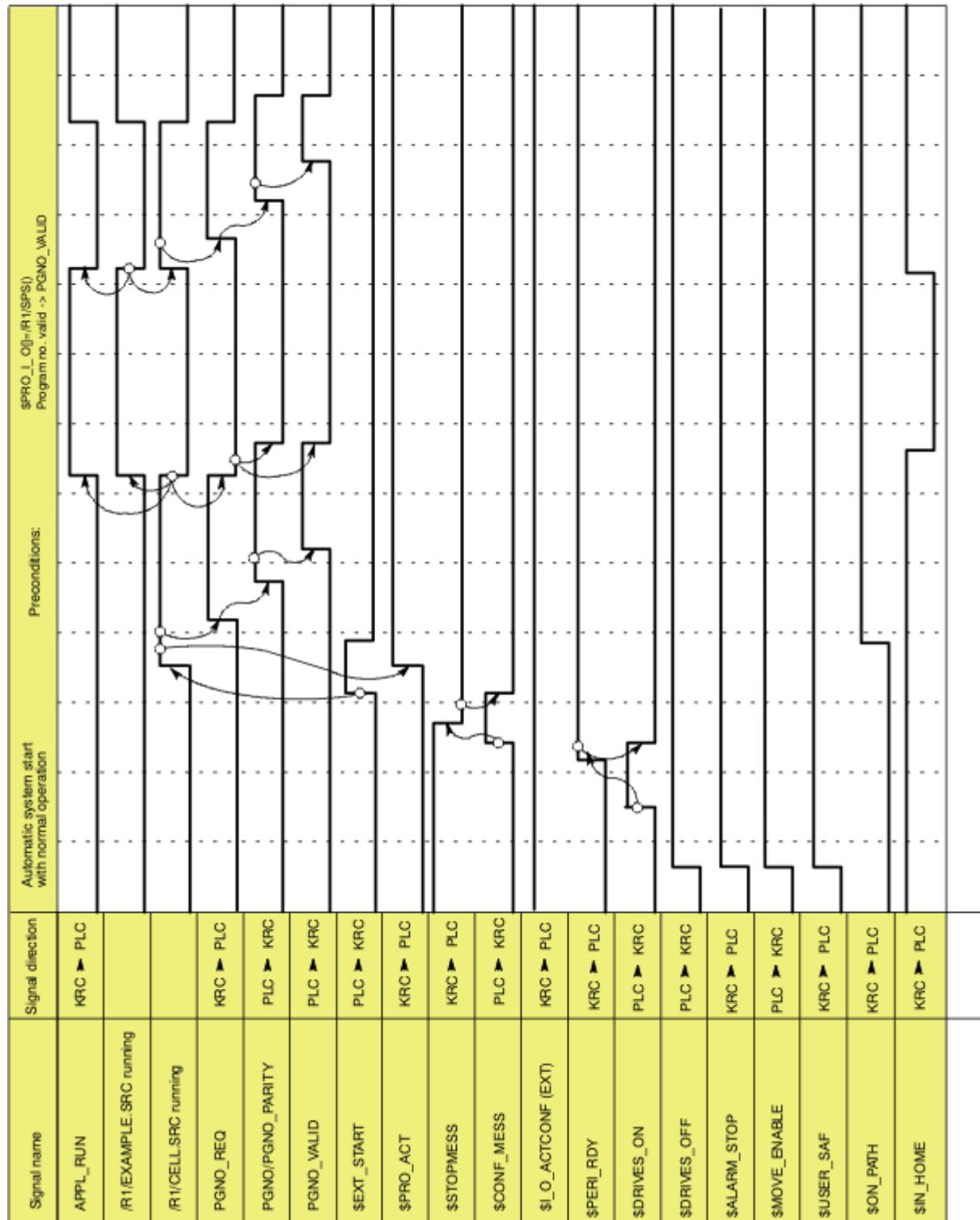
1  DEFDAT  P00
2
3  BOOL PLC_ENABLE=TRUE ; Enable error-code transmission to plc
4  INT I
5  INT F_NO=1
6  INT MAXERR_C=1 ; maximum messages for $STOPMESS
7  INT MAXERR_A=1 ; maximum messages for APPLICATION
8  DECL STOPMESS MLD
9  SIGNAL ERR $OUT[25]  TO $OUT[32]
10 BOOL FOUND
11
12 STRUC PRESET INT OUT,CHAR PKG[3],INT ERR
13 DECL PRESET P[255]
...
26 P[1]={OUT 2,PKG[] "P00",ERR 10}
...
30 P[128]={OUT 128,PKG[] "CTL",ERR 1}
...
35 STRUC ERR_MESS CHAR P[3],INT E
36 DECL ERR_MESS ERR_FILE[64]
37 ERR_FILE[1]={P[] "XXX",E 0}
...
96 ERR_FILE[64]={P[] "XXX",E 0}
97 ENDDAT

```

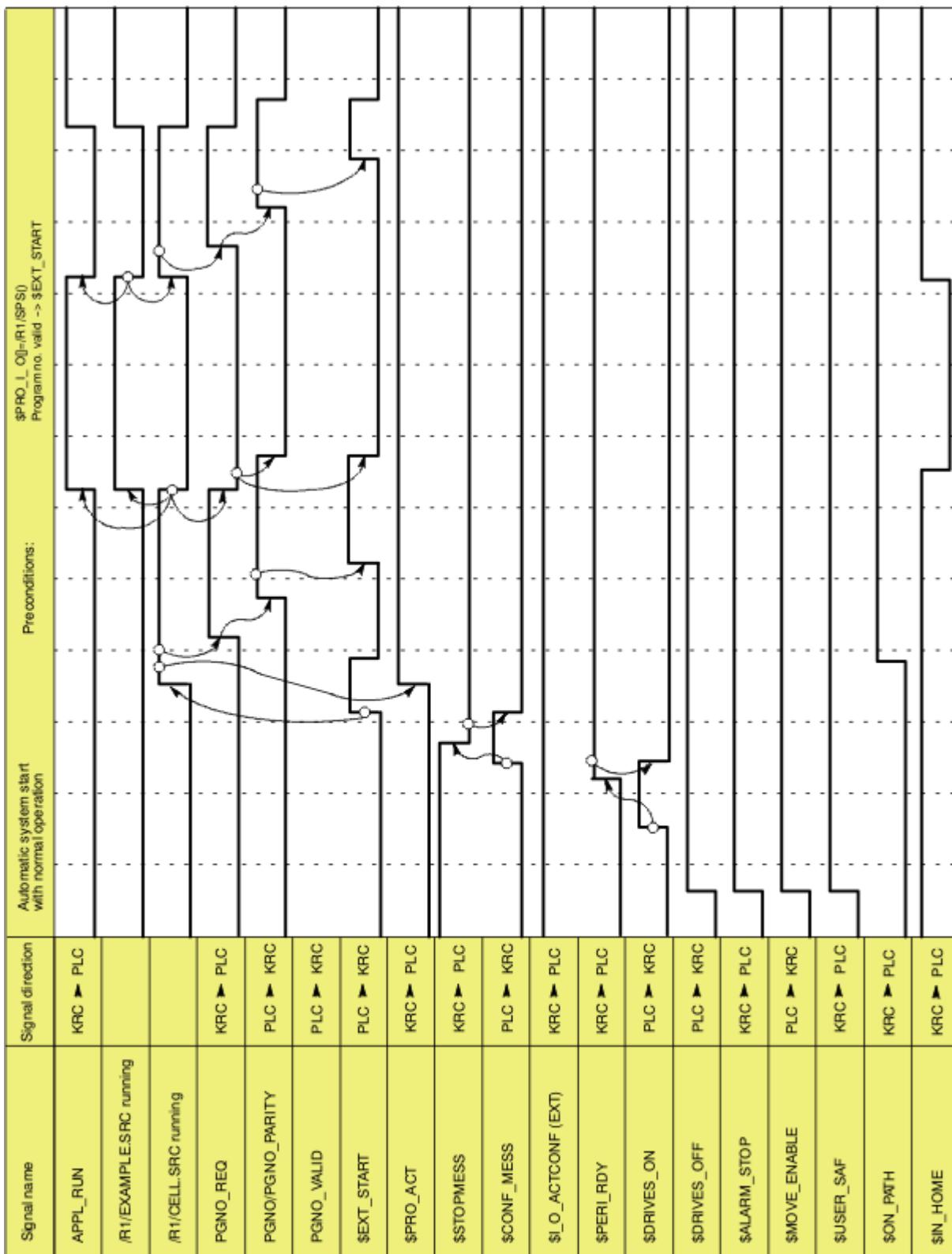
Line	Description
3	PLC_ENABLE must be TRUE.
6	Enter the number of controller errors for the transmission of which parameters are to be defined.
7	Enter the number of application errors for the transmission of which parameters are to be defined.
9	Specify which robot controller outputs the higher-level controller should use to read the error numbers. There must be 8 outputs.
13	In the following section, enter the parameters of the errors.  P[1] ... P[127]: range for application errors  P[128] ... P[255]: range for controller errors
26	Example of parameters for application errors: <ul style="list-style-type: none"> <li>■ OUT 2 = error number 2</li> <li>■ PKG[] "P00" = technology package</li> <li>■ ERR 10 = error number in the selected technology package</li> </ul>

Line	Description
30	Example of parameters for controller errors: <ul style="list-style-type: none"><li>■ OUT 128 = error number 128</li><li>■ PKG[] "CTL" = technology package</li><li>■ ERR 1 = error number in the selected technology package</li></ul>
37 ... 96	The last 64 errors that have occurred are stored in the ERR_FILE memory.

#### 6.10.4 Signal diagrams



**Fig. 6-24: Automatic system start and normal operation with program number acknowledgement by means of PGNO\_VALID**



**Fig. 6-25: Automatic system start and normal operation with program number acknowledgement by means of \$EXT\_START**

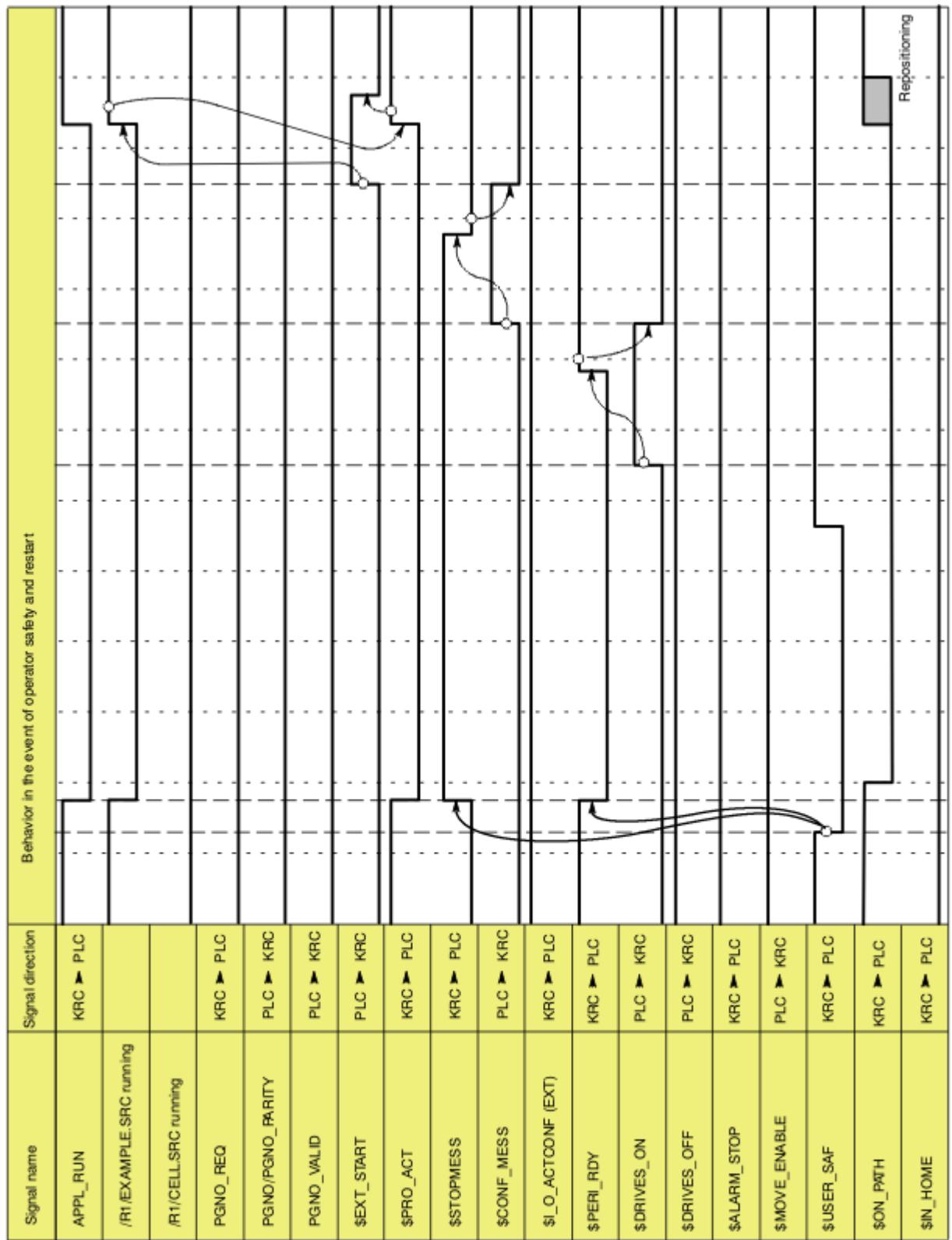
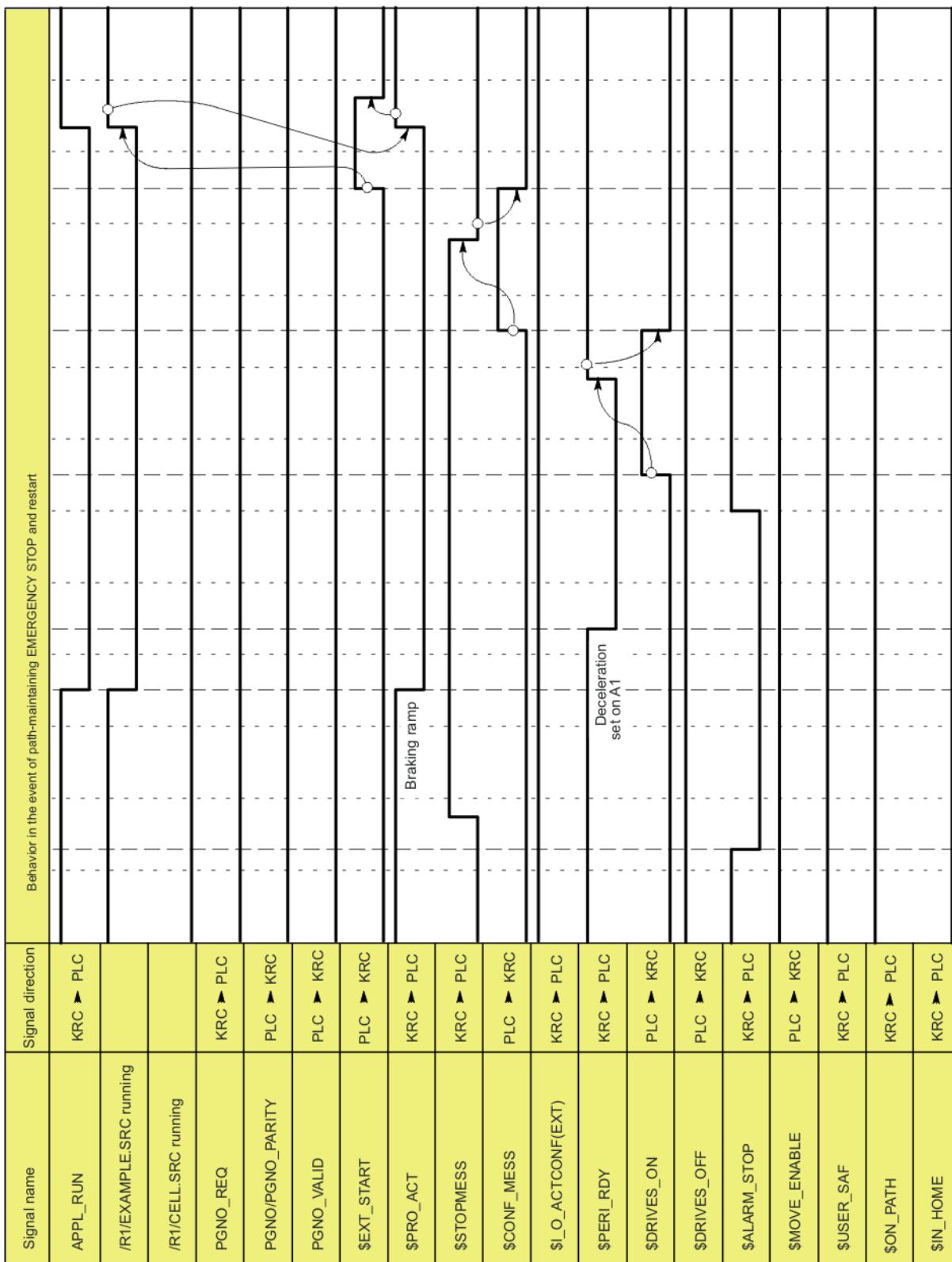


Fig. 6-26: Restart after dynamic braking (operator safety and restart)



**Fig. 6-27: Restart after path-maintaining EMERGENCY STOP**

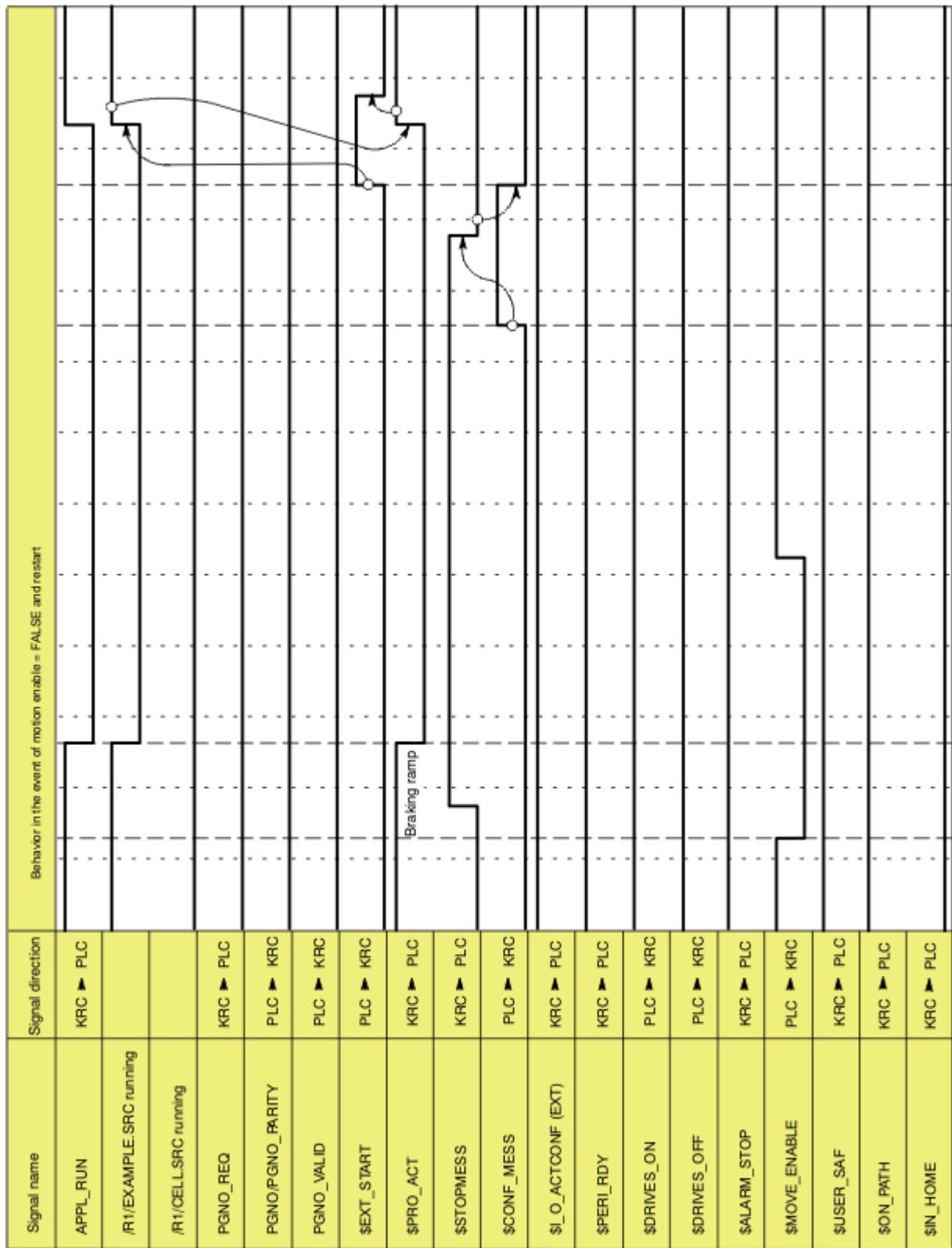


Fig. 6-28: Restart after motion enable

Signal name	Signal direction	Behavior in the event of user STOP and restart
APPL_RUN	KRC ► PLC	
/R1/EXAMPLE.SRC running		
PGNO_REQ	KRC ► PLC	
PGNO/PGNO_PARITY	PLC ► KRC	
PGNO_VALID	PLC ► KRC	
\$EXT_START	PLC ► KRC	
\$PRO_ACT	KRC ► PLC Programmed user STOP	
\$STOPMESS	KRC ► PLC	
\$CONF_MESS	PLC ► KRC	
\$I_O_ACTCONF (EXT)	KRC ► PLC	
\$PERI_RDY	KRC ► PLC	
\$DRIVES_ON	PLC ► KRC	
\$DRIVES_OFF	PLC ► KRC	
\$ALARM_STOP	KRC ► PLC	
\$MOVE_ENABLE	PLC ► KRC	
\$USER_SAF	KRC ► PLC	
\$ON_PATH	KRC ► PLC	
\$IN_HOME	KRC ► PLC	

Fig. 6-29: Restart after user STOP

## 6.11 Torque mode

### 6.11.1 Overview of torque mode

**Description** The “torque mode” function consists of the sub-functionalities “torque limitation” and “deactivation of monitoring functions”.

#### Torque limitation:

The torques, i.e. the motor current, can be limited for individual axes or multiple axes. Torque limitation enables the following applications:

- The axis can push or pull with a defined torque against a resistance.

Example:

application of a defined pressure on the workpiece by an electric motor-driven spot welding gun.

- The axis can be set to “soft”. It can then be moved by application of an external force. It can be pushed away, for example.

Examples:

The robot must grip a workpiece in a press that is then ejected by the press. In order for the robot to be able to yield and absorb the ejector stroke, the affected axis is set to “soft”.

The robot must set a workpiece down at a point from which it can be pulled into exactly the right position by means of clamps. The robot must be compliant for this.

#### Deactivation of monitoring functions:

The torque limitation generally results in a relatively large deviation between the command position and the actual position. Certain monitoring functions are triggered by this deviation, although this is undesirable with torque limitation. These regular monitoring functions can thus be deactivated.

#### Restriction

The following restriction must be taken into consideration if axes are to absorb ejector motions:

A diagonal ejector motion cannot generally be absorbed by switching a single axis to “soft”. Remedy:

- In the case of slightly diagonal ejector motions, a possible remedy is to install the robot with a slight inclination.
- Or contact KUKA Roboter GmbH.



Inclined installation of the robot is only permissible up to a certain angle of inclination. Further information is contained in the robot operating or assembly instructions.

#### 6.11.1.1 Using torque mode

Torque mode is only possible in program mode, not in manual mode.

**WARNING**

By default, the robot controller is configured so that only limits that exceed the holding torque of the axis (\$HOLDING\_TORQUE) can be set. It is nonetheless possible that the axis with limited torque is no longer able to achieve the necessary torque for braking, holding or moving the axis. This can be the case, for example, if the default configuration of the robot controller has been changed or incorrect load data are used.

Incorrectly set values can result in unexpected behavior of the robot controller, e.g. motion in a different direction or with different acceleration.

For this reason:

- Only ever limit the torque in small steps, gradually approaching the required limit.
- Do not limit the torque further than necessary.

Failure to observe this may result in death to persons, severe physical injuries or considerable damage to property.



If an application requires torque limits that no longer exceed the holding torque of the axis, KUKA Roboter GmbH must be contacted.

**Procedure**

1. Set the torque limits for the desired axis and/or deactivate the regular monitoring functions.

(>>> 6.11.2 "Activating torque mode: SET\_TORQUE\_LIMITS()" Page 177)

If the regular monitoring functions are deactivated, other monitoring functions specially adapted to torque mode are automatically activated.

(>>> "Monitoring functions" Page 176)

2. If the axis is to be set to "soft": move the axis so that the torque limit becomes active. At the end of the motion, the brakes of this axis remain open.

Alternatively, a "motion" to the current position can be executed. The robot does not move, but the brakes are released.

3. Optionally: generate a signal indicating that the axis is stationary (e.g. signal to an injection molding machine).
4. Perform the desired action, e.g. move to workpiece and build up pressure or push the axis away.
5. Optionally: wait for a signal to end torque mode.
6. Deactivate torque mode again.

(>>> 6.11.3 "Deactivating torque mode: RESET\_TORQUE\_LIMITS()" Page 180)

The torque limits are canceled and the regular monitoring functions are reactivated. Furthermore, the command position is adjusted to the actual position.

**Activated/deactivated**

Torque mode is considered to be activated in the following case:

- If the upper torque limit is less than or equal to the upper value of the \$TORQUE\_AXIS\_MAX interval.  
(>>> 6.11.5.3 "\$TORQUE\_AXIS\_MAX" Page 182)
- And/or: If the lower torque limit is greater than or equal to the lower value of the \$TORQUE\_AXIS\_MAX interval.
- And/or: If the regular monitoring functions are deactivated.

Torque mode is considered to be deactivated in the following case:

- If no limits are set or if the limits are invalid. A limit is invalid if it is outside the \$TORQUE\_AXIS\_MAX interval.

- And: If the regular monitoring functions are deactivated.

**Automatic deactivation** Torque mode is automatically deactivated in the following cases:

- End of program
- Program reset
- Program deselection
- Block selection (but no deactivation if the target of the block selection is in an interrupt program)
- RESUME (but no deactivation if the RESUME statement returns to an interrupt program)
- Manual mode is activated. Planning is carried out from the current actual position and motion is resumed with full torque.

**Monitoring functions**

Torque mode generally results in a relatively large deviation between the command position and the actual position. Certain monitoring functions are triggered by this deviation, although this is undesirable in torque mode. These regular monitoring functions can thus be deactivated using SET\_TORQUE\_LIMITS().

If the regular monitoring functions are deactivated, other monitoring functions specially adapted to torque mode are automatically activated for the actual velocity and following error. If required, user-defined values can be set for these special monitoring functions using SET\_TORQUE\_LIMITS().

The following messages belong to the regular monitoring functions. They are no longer displayed if the regular monitoring functions are deactivated.

Monitoring	Message no. / message
Following error monitoring	26024: <b>Ackn. Max. following error exceeded (Drive).</b>
Standstill monitoring	1100: <b>Stopped (Axis number)</b>
Positioning monitoring	1105: <b>Positioning monitoring (Axis number)</b>
Monitoring whether motor blocked	26009: <b>Motor blocked (Drive).</b>

It is also possible, however, to retain the regular monitoring functions in torque mode. This may be useful, for example, if torque mode is used to avoid damage in the case of collisions.

(>>> 6.11.6.2 "Robot program: avoiding damage in the event of collisions"  
Page 184)

### 6.11.1.2 Robot program example: setting A1 to “soft” in both directions

**Description**

This simple example illustrates the basic principle of torque mode.

In this example, A1 is to be set to “soft” in both directions. For this purpose, both the positive and negative current limits are set to 0 Nm. This allows A1 to be moved by application of an external force.

**Program**

```

...
1 PTP {A1 10}
2 SET_TORQUE_LIMITS(1, {lower 0, upper 0, monitor #off})
3 PTP {A1 11}
...
4 RESET_TORQUE_LIMITS(1)
5 PTP {A1 -20}
...
```

Line	Description
2	Negative and positive current limit of A1 are set to 0; the regular monitoring functions are deactivated.  (The actual velocity and the following error are now monitored with special monitoring functions.)
3	A "motion" is executed to activate torque limitation. (Since both current limits are set to 0, the robot will not actually move.)  A1 can now be moved by application of an external force.
4	Deactivate torque mode again for A1.  Torque limitation is canceled and the regular monitoring functions are reactivated. Furthermore, the command position is automatically adjusted to the actual position.
5	The robot moves to the next position.  (The motion from line 4 is not belatedly executed now, as the command-actual adjustment has been carried out in line 5.)



It is only for A1 that the holding torque is 0 Nm. The limits can therefore not generally be set to 0 for setting an axis to "soft".

A more detailed example that can also be applied to other axes can be found here: ([>>> 6.11.6.1 "Robot program: setting axis to "soft" in both directions" Page 184](#))

### 6.11.2 Activating torque mode: SET\_TORQUE\_LIMITS()

#### Description

This function can be used to perform the following actions for a specific axis:

- Limit the torques in the positive and/or negative direction.
- Deactivate the regular monitoring functions that would be triggered by a higher following error.
- If the regular monitoring functions are deactivated: modify the values for the special monitoring functions.

#### Function

SET\_TORQUE\_LIMITS (axis: in, values: in)

Element	Description
axis	Type: INT  Axis to which the statement applies
values	Type: TorqLimitParam  Values set for the axis

#### TorqLimitParam

STRUC TorqLimitParam REAL lower, upper, SW\_ONOFF monitor,  
REAL max\_vel, max\_lag

Element	Description
lower	Lower torque limit  Unit: Nm (for linear axes: N)  Default value: -1E10 (i.e. unlimited)
upper	Upper torque limit  Unit: Nm (for linear axes: N)  Default value: 1E10 (i.e. unlimited)

Element	Description
monitor	<ul style="list-style-type: none"> <li>■ <b>#ON</b> (Default): activates the regular monitoring functions.</li> <li>■ <b>#OFF</b>: deactivates the regular monitoring functions. Instead, the monitoring functions <code>max_vel</code> and <code>max_lag</code> are activated.</li> </ul>
<code>max_vel</code>	<p>Maximum permissible actual velocity in torque mode (only relevant if the regular monitoring functions are deactivated)</p> <p>Only a positive value may be programmed.</p> <p>Unit: Degrees (for linear axes: mm)</p> <p>Default value (valid for all operating modes): T1 jog velocity * internal safety factor</p> <p>In T1, the maximum velocity with which jogging can be carried out is the default value, even if a higher value is programmed.</p> <p><b>Note:</b> Only set a higher value than the default value if absolutely necessary.</p>
<code>max_lag</code>	<p>Maximum permissible following error in torque mode (only relevant if the regular monitoring functions are deactivated)</p> <p>Only a positive value may be programmed.</p> <p>Unit: Degrees (for linear axes: mm)</p> <p>Default value: 5 degrees (for linear axes: 100 mm)</p> <p><b>Note:</b> Only set a higher value than the default value if absolutely necessary.</p>

**lower/upper**

When must the upper torque be limited and when must the lower torque be limited?

General: The direction in which the following error is building up must always be limited.

Example: The robot is to be moved up against an obstacle and then stop there. The torque that is thus built up is to be limited.

- If the obstacle appears in the positive direction, `upper` must be set.
- If the obstacle appears in the negative direction, `lower` must be set.

**Characteristics**

- `SET_TORQUE_LIMITS()` can be used in robot programs and in submit programs.
- **Advance run stop:** In the robot program, the statement triggers an advance run stop.
- *Values* may remain partially non-initialized. The non-initialized components mean that the existing values are to remain unchanged.
- If both limits are set, `upper` must be  $\geq$  `lower`.
- If one limit (or both) is already set and the other limit is then set, and if the new limit would result in an empty interval, the new limit value becomes the value for both limits. Example:
  - Already set: {`lower 1, upper 2`}
  - Newly set: {`lower 3`}
  - This results in: {`lower 3, upper 3`}
- It is permissible to set a positive `lower` or a negative `upper` limit.
- The limits set must be greater than the current holding torque `$HOLDING_TORQUE`. If they are set differently, the robot controller generates an error message that must be acknowledged by the user.



If an application requires torque limits that no longer exceed the holding torque of the axis, KUKA Roboter GmbH must be contacted.

- lower must be less than or equal to the upper value of the \$TORQUE\_AXIS\_MAX\_0 interval.
- upper must be greater than or equal to the lower value of the \$TORQUE\_AXIS\_MAX\_0 interval.
- If the limits are set differently, the robot controller generates an error message that must be acknowledged by the user.

## Examples

### Example 1:

For A1, the permissible torque range is limited to the interval 800 ... 1,400 Nm.

```
SET_TORQUE_LIMITS(1, {lower 800, upper 1400} )
```

### Example 2:

For A3, the upper torque limit is set to 1,200 Nm.

```
SET_TORQUE_LIMITS(3, {upper 1200} )
```

### Example 3:

For A1, the regular monitoring functions are (re)activated.

```
SET_TORQUE_LIMITS(1, {monitor #on} )
```

### Example 4:

For A1, the permissible torque range is limited to the interval -1,000 ... 1,000 Nm. Furthermore, the regular monitoring functions are deactivated and the special monitoring functions are set to user-defined values.

```
SET_TORQUE_LIMITS(1, {lower -1000, upper 1000, monitor #off, max_vel 10, max_lag 20} )
```

### Example 5:

For A1, the permissible torque range is set to -1E10 ... 1E10, i.e. the range is unlimited. The regular monitoring functions are (re)activated.

```
SET_TORQUE_LIMITS(1, {lower -1E10, upper 1E10, monitor #on} )
```

This all corresponds to RESET\_TORQUE\_LIMITS(1), with the difference that in example 5, the command position is not to adapted to the actual position.

### Example 6:

For A1, the lower torque limit is set to a calculated value.

The value has been calculated with the function myCalc() and transferred with the variable myLimits. (In the concrete application, the user must write his own function for this.)

In order for the other components to be non-initialized, the value is pre-initialized with a partially initialized aggregate.

```
DECL TorqLimitParam myParams
...
myParams = {lower 0}
myParams.lower = myCalc()
SET_TORQUE_LIMITS(1, myParams)
```

### Example 7:

In this case, the limits are also set to a value that has been calculated with a function. (In the concrete application, the user must write his own function for this.)

The return value of the function is transferred directly, however.

```
DEFFCT TorqLimitParam myCalcLimits()
DECL TorqLimitParam myLimits
...
RETURN myLimits
ENDFCT
...
SET_TORQUE_LIMITS(1, myCalcLimits())
```

### 6.11.3 Deactivating torque mode: RESET\_TORQUE\_LIMITS()

#### Description

This function has the following effect on the selected axis:

- It cancels the limitation of the torques insofar as they were limited.
- It reactivates the regular monitoring functions insofar as they were deactivated.
- It adapts the command position to the actual position.

#### Function

RESET\_TORQUE\_LIMITS (axis: in)

Element	Description
axis	Type: INT Axis to which the statement applies

#### Characteristics

- The statement can be used in robot programs and in submit programs.
- **Advance run stop:** In the robot program, the statement triggers an advance run stop. This cannot be masked with CONTINUE!

#### Alternative

If no command/actual value adjustment is required, torque mode can also be deactivated with SET\_TORQUE\_LIMITS instead of RESET\_TORQUE\_LIMITS:

```
SET_TORQUE_LIMITS(1, {lower -1E10, upper 1E10, monitor #on} )
```

- Advantage: Can be used during a motion ("on the fly").
- Disadvantage: If the torque limitation has resulted in a relatively large following error, the robot accelerates very fast. This can trigger monitoring functions and stop the program.



Deactivation by means of SET\_TORQUE\_LIMITS is not suitable in most cases.

### 6.11.4 Interpreter specifics

#### Description

- SET\_TORQUE\_LIMITS() and RESET\_TORQUE\_LIMITS() can be used in robot programs and in submit programs.
- The statements are interpreter-specific, i.e. they only work in the interpreter in which they have been used.
- SET\_TORQUE\_LIMITS() first takes effect when the axis is moved for the interpreter that generates the statement. Example:
  - a. Torque mode is activated in the robot program for an external axis.
  - b. The external axis is moved by a submit program. Torque mode has no effect.
  - c. The external axis is moved by a robot program. Torque mode takes effect.
- If torque mode is already active, SET\_TORQUE\_LIMITS() takes effect immediately.

- SET\_TORQUE\_LIMITS() works immediately if a motion is active. For this reason, the torque limits can be set at any time in robot programs, both inside and outside an interrupt, and the monitoring functions can be activated and deactivated.

It is also possible to use torque mode inside an interrupt program only. (If RESET\_TORQUE\_LIMITS() is used, it may subsequently be necessary to return to the interrupt position with PTP \$AXIS\_RET.)

(>>> 6.11.6.3 "Robot program: torque mode in the interrupt" Page 186)

- When a torque-driven axis "changes owner", the command position is adjusted to the actual position.

"Change of owner" means: an interpreter has moved the axis in torque mode (and thus "owns" it). While torque mode is active, the axis is moved by a different interpreter.

The main application here is: jogging after a program has been interrupted in torque mode.

## Example

The following example shows when SET\_TORQUE\_LIMITS() is effective, depending on whether torque mode is already active or not.

Initial situation (default): the monitoring functions are activated.

```
1 SET_TORQUE_LIMITS(1, {monitor #off})
2 HALT
3 PTP_REL {A1 10}
4 HALT
5 SET_TORQUE_LIMITS(1, {monitor #on})
6 HALT
7 PTP_REL {A1 15}
```

Line	Description
1	The monitoring functions for A1 are deactivated.
2	Here the monitoring functions are still activated.
3	The axis is moved. From here on, the statement SET_TORQUE_LIMITS is effective.
4	The monitoring functions are deactivated.
5	The monitoring functions are activated.
6	Here the monitoring functions are already activated. Because torque limitation was already active, the statement took effect immediately and not just after the next motion of this axis.

## 6.11.5 Diagnostic variables for torque mode

All these variables and constants are write-protected.

Their value is not dependent on the interpreter.

### 6.11.5.1 \$TORQUE\_AXIS\_ACT

**Variable** \$TORQUE\_AXIS\_ACT[*axis number*]

Data type: REAL

**Description** Current motor torque for axis [*axis number*]

Unit: Nm (for linear axes: N)

The value is only relevant if the brakes are released. If the brakes are applied, it is virtually zero. (The state of the brakes can be displayed by means of the system variable \$BRAKE\_SIG. The value of \$BRAKE\_SIG is a bit array: bit 0 corresponds to A1, bit 6 corresponds to E1.)

**Advance run stop:** In the robot program, the variable triggers an advance run stop.

(>>> 6.11.5.6 "Comparison: \$TORQUE\_AXIS\_ACT and \$HOLDING\_TORQUE" Page 183)

#### 6.11.5.2 \$TORQUE\_AXIS\_MAX\_0

**Constant** \$TORQUE\_AXIS\_MAX\_0[*axis number*]

Data type: REAL

**Description** Maximum permanent motor torque for axis [*axis number*] at velocity 0

The value specifies an interval: from -*value* to +*value*.

Unit: Nm (for linear axes: N)

**Advance run stop:** does not trigger an advance run stop.



*lower* must be less than or equal to the upper value of the \$TORQUE\_AXIS\_MAX\_0 interval.

*upper* must be greater than or equal to the lower value of the \$TORQUE\_AXIS\_MAX\_0 interval.

If the limits are set differently, the robot controller generates an error message that must be acknowledged by the user.

#### 6.11.5.3 \$TORQUE\_AXIS\_MAX

**Constant** \$TORQUE\_AXIS\_MAX[*axis number*]

Data type: REAL

**Description** Absolute maximum motor torque for axis [*axis number*]

The value specifies an interval: from -*value* to +*value*.

Unit: Nm (for linear axes: N)

**Advance run stop:** does not trigger an advance run stop.

#### 6.11.5.4 \$TORQUE\_AXIS\_LIMITS

**Variable** \$TORQUE\_AXIS\_LIMITS[*axis number*]

Data type: TorqLimitParam

**Description** Currently active motor torque limitation for axis [*axis number*]

Unit: Nm (for linear axes: N)

The variable is primarily intended for diagnosis via the variable correction function or variable overview.

##### Characteristics:

- If there are currently no limits active, *upper* and *lower* remain non-initialized.
- The component *monitor* is always initialized unless the axis does not exist.

This is relevant, for example, in the case of 4-axis and 5-axis robots: if the entire array is displayed, the non-existent axes can be easily identified.

Non-existent external axes are simply not displayed when the entire array is displayed.

- `Max_vel` and `max_lag` are non-initialized if `monitor = #ON`, as the regular monitoring functions are active in this case.  
If `monitor = #OFF`, the values of `max_vel` and `max_lag` are displayed, irrespective of whether they have been set explicitly in the current program or whether the default values are being used.



The fact that certain components may remain non-initialized simplifies diagnosis for the user.

If the variable is accessed via KRL, however, the robot controller may regard the access as "invalid". Recommendation: Check the state of the variable with `VARSTATE()` prior to access.

**Advance run stop:** In the robot program, the variable triggers an advance run stop.

#### 6.11.5.5 \$HOLDING\_TORQUE

**Variable**            `$HOLDING_TORQUE[axis number]`

Data type: REAL

**Description**          Holding torque for the robot axis [axis number]

Unit: Nm

The holding torque refers to the current actual position of the axis and the current load.

(For external axes, the value 0 N is always returned.)

**Advance run stop:** In the robot program, the variable triggers an advance run stop.

(>>> 6.11.5.6 "Comparison: \$TORQUE\_AXIS\_ACT and \$HOLDING\_TORQUE " Page 183)



If the upper and lower torque limits are set to `$HOLDING_TORQUE` for all axes, the robot must remain stationary when the brakes are released.

If this is not the case, i.e. if the robot drifts, the load is not correctly configured.

#### 6.11.5.6 Comparison: \$TORQUE\_AXIS\_ACT and \$HOLDING\_TORQUE

In the case of a robot that is stationary with the brakes released, `$TORQUE_AXIS_ACT` is not equal to `$HOLDING_TORQUE`, although this might be assumed.

Characteristics of `$HOLDING_TORQUE`:

- A calculated value that does not take friction into consideration. Control effects have no influence on the value.
- Is only dependent on the current position of the axis. Thus remains unchanged if the position remains constant.

Characteristics of `$TORQUE_AXIS_ACT`:

- Calculated from the actual currents. Friction and control effects thus affect the value.
- Can change if the position remains constant.

The discrepancy between `$HOLDING_TORQUE` and `$TORQUE_AXIS_ACT` is less than or equal to the current friction if the robot remains stationary for at least 1 second. A precondition is that the load data are correct.

## 6.11.6 Other examples

### 6.11.6.1 Robot program: setting axis to “soft” in both directions

<b>Description</b>	The robot must grip a workpiece in a press that is then ejected by the press. In order for the robot to be able to yield and absorb the ejector stroke, the axis is set to “soft”.  For this, the torque limits must be set to a very small interval around the holding torque. (It is only for A1 that the holding torque = 0 Nm. The limits can therefore not generally be set to 0 for setting an axis to “soft”.)  Assumption for this example: a rotation about the axis [ <i>ideal_axis</i> ] move the gripper almost exactly in the ejector direction.
<b>Program</b>	<pre> 1 DECL TorqLimitParam myLimits 2 DECL INT ideal_axis ... 3 myLimits.monitor = #off 4 myLimits.lower = \$holding_torque[ideal_axis] - 10 5 myLimits.upper = \$holding_torque[ideal_axis] + 10 6 SET_TORQUE_LIMITS(ideal_axis, myLimits) 7 PTP \$AXIS_ACT 8 OUT_SIGNAL_SOFT = TRUE 9 WAIT FOR IN_SIGNAL_EJECTED 10 RESET_TORQUE_LIMITS(i) 11 OUT_SIGNAL_SOFT = FALSE 12 WAIT FOR IN_SIGNAL_NEXTMOVE ... </pre>

Line	Description
3 ...6	For the axis [ <i>ideal_axis</i> ], the regular monitoring functions are deactivated and the torques are limited to a very small interval around the holding torque.
7	Execute a “motion” to the current position to activate reduction of the current limits. The axis [ <i>ideal_axis</i> ] can now be moved by the ejector of the press.
8	Signal to the press controller that the axis is ready for the ejection.
9	Wait for signal from press controller that the workpiece has been ejected.
10	Cancels torque limitation and reactivates the regular monitoring functions. Furthermore, adjusts the command position the actual position.
11	Signal to the press controller that the axis is no longer ready for an ejection.
12	If required:  Wait for a signal that motion away from the position is allowed.

### 6.11.6.2 Robot program: avoiding damage in the event of collisions

<b>Description</b>	Torque limitation can be used to avoid damage in the event of collisions. Compared with the <b>Collision detection</b> function that can be activated in the inline forms of the motion instructions, this has the following advantage and disadvantage:
	<ul style="list-style-type: none"> <li>■ Advantage: It is assured that the robot only presses against the obstacle with a defined, limited force.</li> <li>■ Disadvantage: The robot becomes sluggish. High accelerations are no longer possible.</li> </ul>

**Program**

The robot fetches workpieces from a box. During the motion to points P7, P8 and P9, the possibility cannot be ruled out of the robot with the workpiece colliding with the box. It must be ensured that the robot does not press so hard that damage can result. For this, the forces are limited before the critical points.

The regular monitoring functions are deactivated. This is not because they would otherwise be triggered unnecessarily; on the contrary, they are not strict enough for this example. Instead, one of the special monitoring functions is set to a very low value. (Depending on the specific application, it may also be useful to use the regular monitoring functions.)

```

...
1 DECL TorqLimitParam myParams
...
2 FOR i = 1 to 6
3   myParams.lower = $holding_torque[i] - 500
4   myParams.upper = $holding_torque[i] + 500
5   myParams.monitor = #off
6   myParams.max_lag = 0.1
7   SET_TORQUE_LIMITS(i, myParams)
8 ENDFOR
9 $acc.cp = my_low_acceleration
10 $vel.cp = my_low_velocity
11 LIN P7
12 LIN P8
13 LIN P9
14 FOR i = 1 to 6
15   myParams.lower = -1E10
16   myParams.upper = 1E10
17   myParams.monitor = #on
18   SET_TORQUE_LIMITS(i, myParams)
19 ENDFOR
20 $acc.cp = my_high_acceleration
21 $vel.cp = my_high_velocity
22 LIN P10
...

```

Line	Description
2 ... 7	The torques for A1 ... A6 are limited.
3, 4	The limits are set to a relatively small interval centered on the holding torque.
5, 6	The regular monitoring functions are deactivated. Max_lag = 0.1 has the effect of triggering a stop in the case of a following error as low as 0.1°.
9, 10	Acceleration and velocity are reduced so that the robot approaches the critical point slowly.
11 ...13	Points at which a collision could occur. If a collision occurs, the monitoring function max_lag is triggered and the system operator can intervene.
After the critical section:	
14 ...19	Torque mode is deactivated. SET_TORQUE_LIMITS can be used here: the robot only reaches this point if it has passed the critical points without collision. In this case, no following error has built up and command/actual value adjustment is not required.
20, 21	Acceleration and velocity are reset to the previous higher values.
22	Uncritical point

### 6.11.6.3 Robot program: torque mode in the interrupt

<b>Description</b>	This demonstrates that torque mode can be used completely inside an interrupt program. The example is not primarily a practical one, but is intended to show that this use is generally possible.
<b>Program</b>	The robot is to determine the position of a workpiece whose possible location is known, but not its exact position. First, a proximity sensor signals to the robot controller when the robot is in the vicinity of the workpiece. On the basis of this sensor signal, an interrupt program is then called.  The actual search is carried out in the interrupt program: A1 moves towards the workpiece. Torque mode is activated for A1 beforehand. The robot comes to a standstill where it makes contact with the workpiece due to the reduced torques: the workpiece has been "found". This position can now be saved as the position of the workpiece.
	<pre> ... 1 LIN P1 2 INTERRUPT DECL 1 WHEN sensor_signal==true DO search() 3 INTERRUPT ON 1 4 LIN P2 5 INTERRUPT OFF 1 ... ----- 6 DEF search() 7 ... 8 BRAKE 9 SET_TORQUE_LIMITS(1,{lower 1000, upper 1000, monitor #off}) 10 PTP_REL {A1 10} 11 RESET_TORQ_LIMITS(1) 12 piece_found = \$POS_ACT_MES 13 PTP \$AXIS_RET 14 END </pre>

Line	Description
1 ... 4	On the way to P2, the sensor is to signal when the robot is in the vicinity of the workpiece.
2	When the sensor signal is received, the subprogram search() is called.
<b>In the subprogram:</b>	
8	The current motion is stopped as soon as search() is executed.
9	Sets torque limits for A1, deactivates regular monitoring functions.
10	A1 moves. The robot comes to a standstill where it makes contact with the workpiece due to the reduced torques.  Once the robot has reached its command position {A1 10}, the next program line is executed. (The fact that the actual position deviates from the command position has no effect, as the regular monitoring functions have been deactivated.)
11	Cancels torque limitation and reactivates the regular monitoring functions. Furthermore, adjusts the command position the actual position.
12	The current position of the robot now indicates the position of the workpiece. This is saved here in a variable.
13	Returns to the position at which the robot left the path in the main program.

#### 6.11.6.4 Robot program: servo gun builds up pressure

**Description** This program shows how torque mode can be activated by means of a trigger. (Comparable programs are used in the background in the KUKA.ServoGun technology packages. They thus do not need to be programmed by the user.)

**Program** **Main program:**

```

1 DEF SPOT()
2 DECL BOOL error_occurred
...
3 Interrupt DECL 1 WHEN $stopmess DO resume_subprog()
4 Interrupt ON 1
5 REPEAT
6 error_occurred = false
7 SPOT_MOVE()
8 UNTIL error_occurred == false
...

```

Line	Description
3	If an error occurs, resume_subprog() is to be called.
7	The weld program SPOT_MOVE() is called.
5 ... 8	If an error has occurred (i.e. if error_occurred == true), SPOT_MOVE() is repeated.

**Weld program:**

```

1 DEF SPOT_MOVE()
...
2 TorqLimWeld = {lower -1000, upper 1000 , monitor #off}
3 i = 6+EG_EXTAX_ACTIVE
...
4 LIN P_APPROX C_DIS
5 $VEL_EXTAX[EG_EXTAX_ACTIVE]=EG_MAX_CONST_VEL[EG_EXTAX_ACTIVE]
6 LIN P_APPROX C_DIS
7 TRIGGER WHEN DISTANCE=0 DELAY=50 DO SET_TORQUE_LIMITS(i,
TorqLimWeld) PRIO = -1
8 LIN P_PART C_DIS
9 TRIGGER WHEN DISTANCE=0 DELAY=50 DO START_TIMER_SPOT() PRIO=82
10 LIN P_PRESSURE C_DIS
11 LIN P_WELD
12 WAIT FOR EG_TRIGGER_END
13 RESET_TORQUE_LIMITS(i)
14 Interrupt OFF 1
15 LIN P_PART C_DIS
16 END

```

Line	Description
7	Shortly before the gun touches the workpiece, the torques are reduced.
9 ... 11	Build up pressure and then weld.
12	The weld timer signals the end of welding.
13	Cancels torque limitation and reactivates the regular monitoring functions. Furthermore, adjusts the command position the actual position.

**Interrupt program in the event of an error:**

```

1 DEF resume_subprog()
2 BRAKE
3 Suppress_repositioning()
4 HALT
5 error_occurred = true
6 RESUME
7 END

```

Line	Description
3	Normally, in the case of a restart after HALT, the robot is repositioned to the position at which the interrupt was triggered. (This is because of \$STOPMESS.) Suppress_repositioning() prevents this repositioning.  Suppress_repositioning() may be useful, depending on the application, but not necessarily.
5	Set error_occurred to TRUE so that the REPEAT loop in the main program is repeated.
6	RESUME deactivates torque mode. Jump back to line 8 in the main program.

#### 6.11.6.5 Submit program: servo gun builds up pressure

##### Precondition

- E1 is already asynchronous with ASYPTP {E1 10}.
- Or: \$ASYNC\_MODE is configured in such a way (bit 0 = 1) that the axis is implicitly set to synchronous mode in the case of ASYPTP in the submit program.

##### Program

```

...
1 IF $PRO_STATE1==#P_FREE
2   SET_TORQUE_LIMITS(7,{upper 1000, monitor #off })
3   ASYPTP {E1 10}
...
4   RESET_TORQUE_LIMITS(7)
5   ASYPTP {E1 -10}
6 ENDIF
...

```

Line	Description
1	Ensure that no robot program is selected.
2	Limit the positive torque and deactivate the regular monitoring functions.
3	Motion towards end point {E1 10} behind the workpiece. The pressure on the workpiece builds up.
4	Cancels torque limitation and reactivates the regular monitoring functions. Furthermore, adjusts the command position the actual position.  The interpreter waits in RESET_TORQUE_LIMITS(7) until the asynchronous motion has been completed. Only then does it perform the command/actual value adjustment. It is therefore not necessary to program WAIT FOR \$ASYNC_STATE == #IDLE before "RESET...".
5	Reopen the gun.

#### 6.12 Event planner

This function can be used for time-related or action-related control of the data comparison between the kernel system and the hard drive. During data comparison, the kernel system data are written to the hard drive.

##### 6.12.1 Configuring a data comparison

##### Precondition

- Expert user group

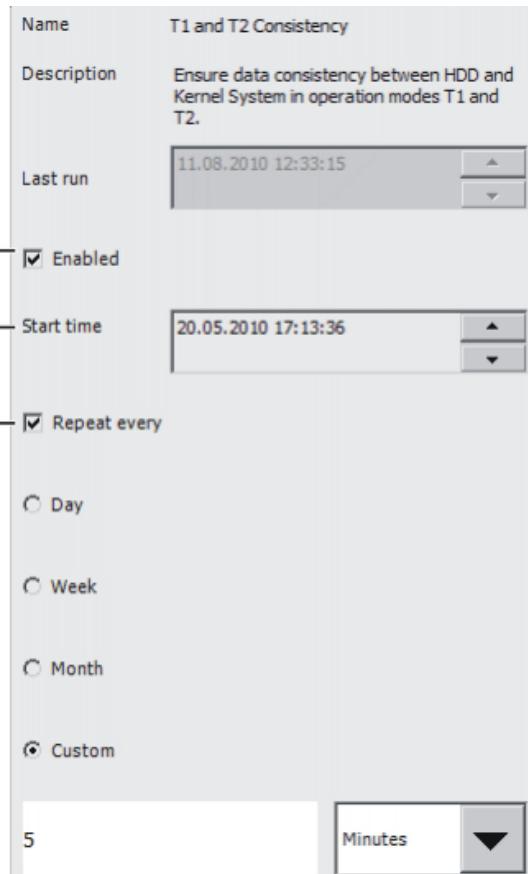
- |                  |  |
|------------------|--|
| <b>Procedure</b> | <ol style="list-style-type: none"> <li>1. In the main menu, select <b>Configuration &gt; Miscellaneous &gt; Event planner</b>.</li> <li>2. Open the tree structure in the left-hand part of the window.</li> <li>3. Select the desired action:           <ul style="list-style-type: none"> <li>■ <b>T1 and T2 consistency</b>: Compare data in operating modes T1 and T2 at regular intervals.</li> <li>■ <b>AUT and EXT consistency</b>: Compare data in operating mode Automatic External at regular intervals.</li> <li>■ <b>Logic consistency</b>: Compare data after a change of operating mode or after online optimizing.<br/>Online optimizing is the modification of the parameters of a program during operation.</li> </ul> </li> <li>4. Make the desired settings in the right-hand part of the window.<br/>           (&gt;&gt;&gt;&gt; 6.12.2 "Configuring T1 and T2 Consistency, AUT and EXT Consistency" Page 189)<br/>           (&gt;&gt;&gt;&gt; 6.12.3 "Configuring Logic Consistency" Page 190)         </li> <li>5. Press <b>Save</b>.</li> </ol> |
|------------------|--|

## 6.12.2 Configuring T1 and T2 Consistency, AUT and EXT Consistency

The following settings are possible here:

- Definition of the date and time that a comparison will first be made between the data in the kernel system and those on the hard drive.
- Definition of the interval at which this operation is to be repeated.

### Description



**Fig. 6-30: Configuring T1 and T2 Consistency**

Item	Description
1	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data comparison is activated.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>
2	Enter date and time in the format indicated.
3	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> interval is activated.</li> <li>■ <b>Check box not active:</b> interval is deactivated.</li> </ul>

**NOTICE** If the intervals selected are too small, this can result in damage to the hard drive. An interval of several minutes is recommended.

### 6.12.3 Configuring Logic Consistency

#### Description

The following settings are possible here:

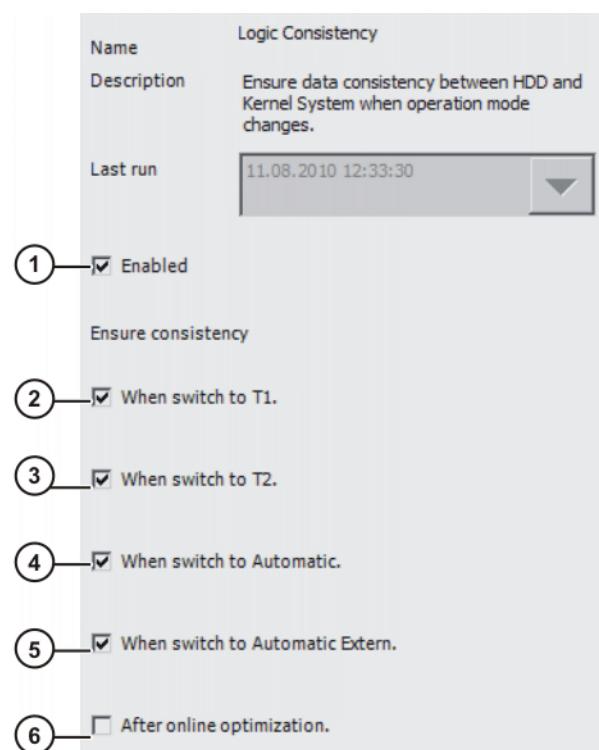


Fig. 6-31: Configuring Logic Consistency

Item	Description
1	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data comparison is activated.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>
2	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data are compared when operating mode is switched to T1.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>
3	<ul style="list-style-type: none"> <li>■ Check box activated: data are compared when operating mode is switched to T2.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>
4	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data are compared when operating mode is switched to Automatic.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>

Item	Description
5	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data are compared when operating mode is switched to Automatic External.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>
6	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> data are compared after online optimizing.</li> <li>■ <b>Check box not active:</b> data comparison is deactivated.</li> </ul>

## 6.13 Brake test

### 6.13.1 Overview of the brake test

**Description** Each robot axis has a holding brake integrated into the motor. The brake test checks every axis at low speed and at the current temperature to see if the braking torque exceeds a certain minimum value. The absolute value of the braking torque is not calculated. The minimum braking torque for the individual axes is stored in the machine data.



The braking torque in the machine data may vary from the information in the data sheet of the brake manufacturer.

#### Performance of the brake test

A precondition for the brake test is that the robot is at operating temperature. This is the case after approx. 1 h in normal operation.

The brake test is carried out using the program BrakeTestReq.SRC. It can be carried out in the following ways:

- Automatically

Integrate BrakeTestReq.SRC into the application program in such a way that it is cyclically called as a subprogram. If a brake test is requested, the robot detects this and performs the brake test immediately.

- Manually

Start the program BrakeTestReq.SRC manually.

#### Brake test request

The following events cause a brake test to be requested.

- Input \$BRAKETEST\_REQ\_EX is set externally, e.g. by a PLC (external request)
- Brake test cycle time has elapsed (internal request)
- Robot controller is rebooted (internal request)
- Manual start of the program BrakeTestReq.SRC (external request)
- Function test of the brake test (internal request)
- In the main menu by means of **Configuration > Inputs/outputs > Reconfigure I/O driver** (external request)

#### Functional principle of the brake test

The brake test checks all brakes one after the other.

1. The robot accelerates to a defined velocity. (The velocity cannot be influenced by the user.)
2. Once the robot has reached the velocity, the brake is applied and the result for this braking operation is displayed in the message window.
3. If a brake has been identified as being defective, the brake test can be repeated for confirmation or the robot can be moved to the parking position.

If a brake has reached the wear limit, the robot controller indicates this by means of a message. A worn brake will soon be identified as defective. Until then, the robot can be moved without restrictions.



If a brake has been identified as being defective, the drives remain under servo-control for 2 hours following the start of the brake test (= monitoring time). The robot controller then switches the drives off.

#### Cycle time

The cycle time is 46 h. It is deemed to have elapsed when the drives have been under servo-control for a total of 46 h. The robot controller then requests a brake test and generates the following message: *Brake test required*. The robot can be moved for another 2 hours. It then stops and the robot controller generates the following acknowledgement message: *Test cycle for brake test request exceeded*. Once the message has been acknowledged, the robot can be moved for another 2 hours.

#### Overview

Step	Description
1	Assign input and output signals for the brake test.  (>>> 6.13.4 "Configuring input and output signals" Page 194)
2	Teach positions for the brake test.  The parking position must be taught. The start position and end position can be taught.  (>>> 6.13.5 "Teaching positions for the brake test" Page 195)
3	If the brake test is to be carried out automatically:  Integrate BrakeTestReq.SRC into the application program in such a way that it is cyclically called as a subprogram.
4	If the brake test is to be carried out manually:  Start the program BrakeTestReq.SRC manually.  (>>> 6.13.6 "Performing a manual brake test" Page 196)

#### 6.13.2 Programs for the brake test

The programs are located in the directory C:\KRC\ROBOT-ER\KRC\R1\TP\BrakeTest.

Program	Description
BrakeTestReq.SRC	This program performs the brake test.  It can be performed in the following ways: <ul style="list-style-type: none"> <li>■ Integrate the program into the application program in such a way that it is cyclically called as a subprogram. If a brake test is requested, the robot detects this and performs the brake test immediately.</li> <li>■ Execute the program manually.</li> <li>■ Test the function of the brake test. The robot controller executes BrakeTestReq.SRC with special parameterization.</li> </ul>
BrakeTestPark.SRC	The parking position of the robot must be taught in this program.  The robot can be moved to the parking position if a brake has been identified as being defective. Alternatively, the brake test can be repeated for confirmation.

Program	Description
BrakeTestStart.SRC	The start position of the brake test can be taught in this program. The robot starts the brake test from this position. If the start position is not taught, the robot performs the brake test at the actual position.
BrakeTestBack.SRC	The end position of the brake test can be taught in this program. The robot moves to this position after the brake test. If the end position is not taught, the robot remains at the actual position after the brake test.
BrakeTestSelfTest.SRC	The program checks whether the brake test has correctly detected a defective brake. For this purpose, the robot controller executes BrakeTestReq.SRC with special parameterization.

### 6.13.3 Signal diagram of the brake test

#### Example 1

The signal diagram for the brake test applies in the following case:

- No brake has reached the wear limit.
- No brake is defective.

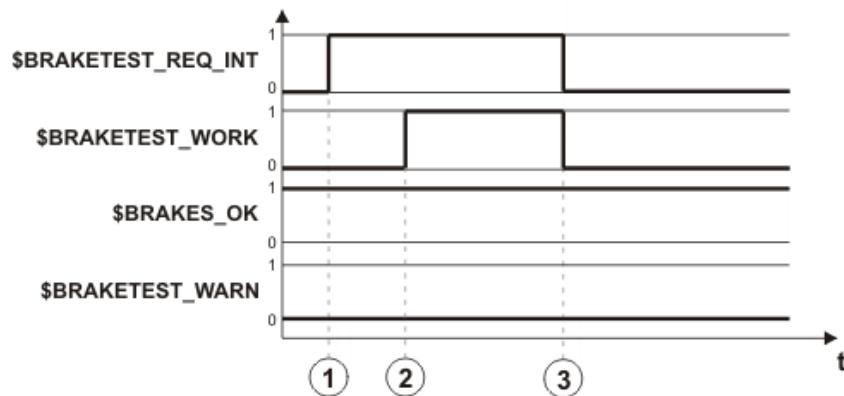


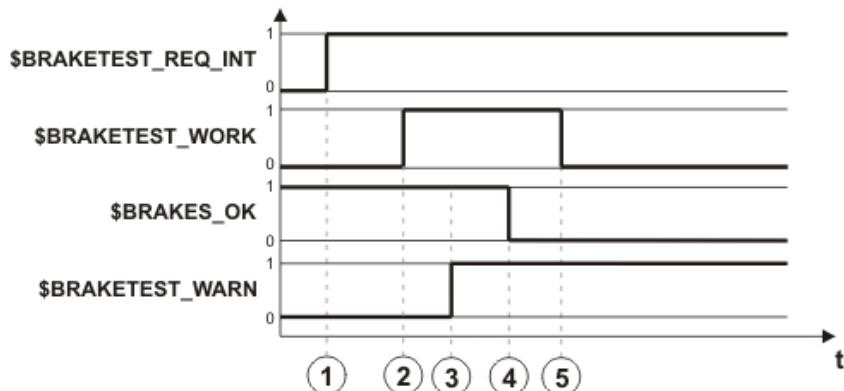
Fig. 6-32: Signal diagram: brakes OK

Item	Description
1	The brake test is requested.
2	Automatic call of the program BrakeTestReq.SRC Start of the brake test
3	The brake test is completed.

#### Example 2

The signal diagram for the brake test applies in the following case:

- Brake A2 is worn.
- Brake A4 is defective.



**Fig. 6-33: Signal diagram: brakes not OK**

Item	Description
1	The brake test is requested. (\$BRAKETEST_REQ_INT is not set to FALSE again until a brake test is carried out with a positive result.)
2	Automatic call of the program BrakeTestReq.SRC Start of the brake test
3	Brake A2 is tested: brake is worn.
4	Brake A4 is tested: brake is defective.
5	The robot has been moved to the parking position or the program has been canceled.

#### 6.13.4 Configuring input and output signals

##### Description

All signals for the brake test are declared in the file \$machine.dat in the directory C:\KRC\ROBOTER\KRC\STEU\MADA.



These signals are not redundant in design and can supply incorrect information. Do not use these signals for safety-relevant applications.

By default, the input signal is routed to \$IN[1026], i.e. there is no active request for a brake test.

The output signals are preset to FALSE. There is no compelling need to assign outputs to them. It is only necessary to assign outputs if there is a need to be able to read the signals (e.g. via the variable correction function or program execution.)

##### Precondition

- Expert user group

##### Procedure

1. Open the file \$machine.dat.
2. Assign inputs and outputs.
3. Save and close the file.

##### \$machine.dat

Extract from the file \$machine.dat (with default settings, without comments):

```
...
SIGNAL $BRAKETEST_REQ_EX $IN[1026]
SIGNAL $BRAKETEST_MONTIME FALSE
SIGNAL $BRAKETEST_REQ_INT FALSE
SIGNAL $BRAKETEST_WORK FALSE
SIGNAL $BRAKES_OK FALSE
SIGNAL $BRAKETEST_WARN FALSE
...
```

## Signals

Signal	Description
\$BRAKETEST_REQ_EX	<p>Input</p> <ul style="list-style-type: none"> <li>■ <b>TRUE</b> = brake test is being requested externally (e.g. by PLC). The robot controller confirms the signal with \$BRAKETEST_REQ_INT = TRUE and generates message 27004.</li> <li>■ <b>FALSE</b> = brake test is not being requested externally.</li> </ul>
\$BRAKETEST_MONTIME	<p>Output</p> <ul style="list-style-type: none"> <li>■ <b>TRUE</b> = robot was stopped due to elapsed monitoring time. Acknowledgement message 27002 is generated.</li> <li>■ <b>FALSE</b> = acknowledgement message 27002 is not active. (Not generated, or has been acknowledged.)</li> </ul>
\$BRAKETEST_REQ_INT	<p>Output</p> <ul style="list-style-type: none"> <li>■ <b>TRUE</b> = message 27004 is active. The signal is not set to FALSE again until a brake test is carried out with a positive result, i.e. with message 27012.</li> <li>■ <b>FALSE</b> = brake test is not requested (either internally or externally).</li> </ul>
\$BRAKETEST_WORK	<p>Output</p> <ul style="list-style-type: none"> <li>■ <b>TRUE</b> = brake test is currently being performed.</li> <li>■ <b>FALSE</b> = brake test is not being performed. If no defective brakes have been detected, message 27012 is generated.</li> </ul> <p>Edge <b>TRUE --&gt; FALSE</b>:</p> <ul style="list-style-type: none"> <li>■ Test was successfully completed. No brake is defective. Message 27012 is generated.</li> <li>■ Or at least 1 defective brake was detected and the robot has moved to the parking position.</li> <li>■ Or the program was canceled during execution of the brake test.</li> </ul>
\$BRAKES_OK	<p>Output</p> <ul style="list-style-type: none"> <li>■ Edge <b>FALSE --&gt; TRUE</b>: Output was set to FALSE by the previous brake test. The brake test was carried out again and no defective brake was detected.</li> <li>■ Edge <b>TRUE --&gt; FALSE</b>: A brake has just been detected as defective. Message 27007 is generated.</li> </ul>
\$BRAKETEST_WARN	<p>Output</p> <ul style="list-style-type: none"> <li>■ Edge <b>FALSE --&gt; TRUE</b>: At least 1 brake has been detected as having reached the wear limit. Message 27001 is generated at the same time.</li> <li>■ Edge <b>TRUE --&gt; FALSE</b>: Output was set to TRUE by the previous brake test. The brake test was carried out again and no worn brake was detected.</li> </ul>

### 6.13.5 Teaching positions for the brake test

**Description** The parking position must be taught.

The start position and end position can be taught.

- If the start position is not taught, the robot performs the brake test at the actual position.
- If the end position is not taught, the robot remains at the actual position after the brake test.

#### Parking position

If a brake is identified as being defective, the robot can be moved to the parking position. The parking position must be selected in a position where no persons are endangered if the robot sags because of the defective brake.

The parking position can correspond to the transport position, for example.



Further information about the transport position is contained in the robot operating or assembly instructions.

#### Precondition

- All output signals are assigned to outputs.
- User group “Expert”
- Operating mode T1

#### Procedure

1. Open the program BrakeTestStart.SRC in the directory R1\TP\BrakeTest.
2. Teach the motions to the start position of the brake test.
  - The motions must be taught in such a way that the robot cannot cause a collision on the way to the start position.
  - In the start position, every robot axis must have an available motion range of  $\pm 10^\circ$ .
3. Save and close the program.
4. Open the program BrakeTestBack.SRC in the directory R1\TP\BrakeTest.
5. Teach the motions from the start position to the end position of the brake test.



The start and end position of the brake test can be identical.

6. Save and close the program.
7. Open the program BrakeTestPark.SRC in the directory R1\TP\BrakeTest.
8. Program the motions from the end position to the parking position of the robot.
9. Save and close the program.

#### 6.13.6 Performing a manual brake test



If a brake is defective and the drives are deactivated, the robot may sag. For this reason, no stop may be triggered during the motion to the parking position. The monitoring functions that can trigger a stop in this range (e.g. monitoring spaces) must be deactivated beforehand. No safety functions may be executed that would trigger a stop (e.g. E-STOP, opening the safety gate, change of operating mode, etc.).

#### Precondition

- No persons or objects are present within the motion range of the robot.
- In the start position, every robot axis has an available motion range of  $\pm 10^\circ$ . (Or, if no start position has been taught, in the actual position.)
- The parking position has been taught in the program BrakeTestPark.SRC.
- “Expert” user group

- Program run mode GO
- AUT mode
- The robot is at operating temperature (= after approx. 1 h in normal operation).

**Procedure**

1. Select the program BrakeTestReq.SRC in the directory R1\TP\BrakeTest and press the Start key.
2. The following message is displayed: *acknowledge to manually perform brake test*. Acknowledge the message.
3. Press the Start key. The message *Programmed path reached (BCO)* is displayed.
4. Press the Start key. The brakes are tested, starting with A1.



**DANGER** Program override is automatically set to 100%. The robot moves at high velocity. Make sure that the robot cannot collide and that no persons are in the motion range of the robot.

## 5. Possible results:

- If a brake is OK, this is indicated by the following message: *Brake X OK*. If all brakes are OK, this is indicated after the brake test by the following message: *Brake test successful*. (It is possible that one or more brakes may have reached the wear limit. This is also indicated by a message.)  
Deselect the program BrakeTestReq.SRC.
- If a brake is defective, this is indicated by the following message: *Insufficient holding torque of brake X*. Once all brakes have been tested, either press **Repeat** to repeat the brake test for checking purposes or press **Park pos.** to move the robot to the parking position.



If a brake has been identified as being defective, the drives remain under servo-control for 2 hours following the start of the brake test (= monitoring time). The robot controller then switches the drives off.

**6.13.7 Checking that the brake test is functioning correctly****Description**

It is possible to check whether the brake test has correctly detected a defective brake: the program BrakeTestSelfTest.SRC simulates a fault in the brakes and triggers a brake test. If the brake test detects the simulated fault, it is functioning correctly.



**DANGER** If the function of the brake test is being tested, the meaning of the messages is different from that in a normal brake test. The following applies to the function test:

- Message *Insufficient holding torque of brake 3*:  
The brake test has correctly detected the simulated fault. The brake test is functioning correctly.
- Any other message, or no message, means:  
The brake test has not detected the simulated fault. The brake test is not functioning correctly. The robot must no longer be moved. KUKA Roboter GmbH must be contacted.

**Precondition**

- No persons or objects are present within the motion range of the robot.
- In the start position, every robot axis has an available motion range of  $\pm 10^\circ$ . (Or, if no start position has been taught, in the actual position.)
- The parking position has been taught in the program BrakeTestPark.SRC.

- “Expert” user group
- Program run mode GO
- AUT mode
- The robot is at operating temperature (= after approx. 1 h in normal operation).

**Procedure**

1. Select the program BrakeTestSelfTest.SRC in the directory C:\KRC\ROBOTER\KRC\R1\TP\SAFEROBOT and press the Start key.
2. The following message is displayed: *Performing self-test for brake test - please acknowledge*. Confirm the message by pressing **Ackn..**.
3. Press the Start key.



Program override is automatically set to 100%. The robot moves at high velocity. Make sure that the robot cannot collide and that no persons are in the motion range of the robot.

4. Result of the function test:

- Message *Insufficient holding torque of brake 3*: The brake test has correctly detected the simulated fault. The brake test is functioning correctly. Deselect the program BrakeTestSelfTest.SRC.  
Perform a manual brake test. This ensures that the simulated fault does not remain active.
- Any other message, or no message, means: The brake test has not detected the simulated fault. The brake test is not functioning correctly.



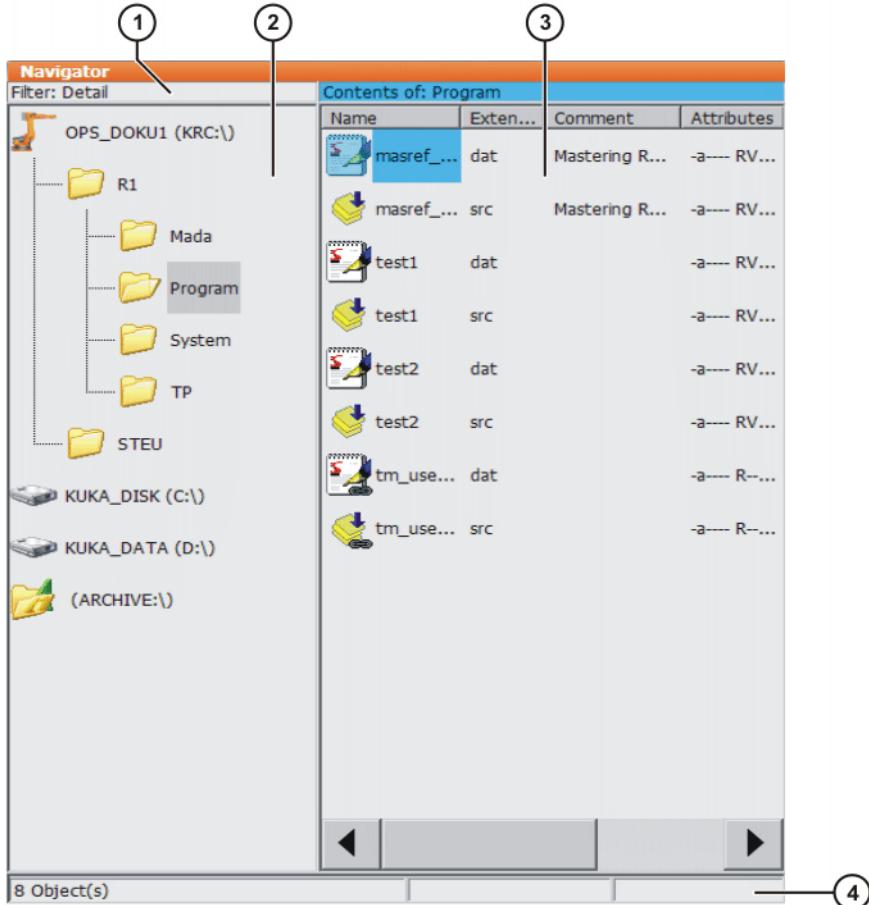
If the function test establishes that the brake test is not functioning correctly:

- The robot must no longer be moved.
- KUKA Roboter GmbH must be contacted.

## 7 Program management

### 7.1 Navigator file manager

#### Overview



**Fig. 7-1: Navigator**

- |                       |              |
|-----------------------|--------------|
| 1 Header              | 3 File list  |
| 2 Directory structure | 4 Status bar |

#### Description

In the Navigator, the user manages programs and system-specific files.

#### Header

- Left-hand area: the selected filter is displayed.  
(>>> 7.1.1 "Selecting filters" Page 200)
- Right-hand area: the directory or drive selected in the directory structure is displayed.

#### Directory structure

Overview of directories and drives. Exactly which directories and drives are displayed depends on the user group and configuration.

#### File list

The contents of the directory or drive selected in the directory structure are displayed. The manner in which programs are displayed depends on the selected filter.

The file list has the following columns:

Column	Description
Name	Directory or file name
Extension	File extension This column is not displayed in the user group "User".
Comment	Comment
Attributes	Attributes of the operating system and kernel system This column is not displayed in the user group "User".
Size	File size in kilobytes This column is not displayed in the user group "User".
#	Number of changes made to the file
Changed	Date and time of the last change
Created	Date and time of file creation This column is not displayed in the user group "User".

### Status bar

The status bar can display the following information:

- Selected objects
- Action in progress
- User dialogs
- User entry prompts
- Requests for confirmation

#### 7.1.1 Selecting filters

**Description**

This function is not available in the user group "User".

The filter defines how programs are displayed in the file list. The following filters are available:

- **Detail**  
Programs are displayed as SRC and DAT files. (Default setting)
- **Modules**  
Programs are displayed as modules.

**Precondition**

- Expert user group

**Procedure**

1. Select the menu sequence **Edit > Filter**.
2. Select the desired filter in the left-hand section of the Navigator.
3. Click **OK** to confirm.

#### 7.1.2 Creating a new folder

**Precondition**

- The Navigator is displayed.

**Procedure**

1. In the directory structure, select the folder in which the new folder is to be created, e.g. the folder **R1**.

Not all folders allow the creation of new folders within them. In the user groups "Operator" and "User", new folders can only be created in the folder **R1**.

2. Press **New**.
3. Enter a name for the folder and confirm it with **OK**.

### 7.1.3 Creating a new program

- Precondition** ■ The Navigator is displayed.
- Procedure**
1. In the directory structure, select the folder in which the program is to be created, e.g. the folder **Program**. (Not all folders allow the creation of programs within them.)
  2. Press **New**.
  3. Only in the user group "Expert":  
The **Template selection** window is opened. Select the desired template and confirm with **OK**.
  4. Enter a name for the program and confirm it with **OK**.



It is not possible to select a template in the user group "User". By default, a program of type "Module" is created.

### 7.1.4 Renaming a file

- Precondition** ■ The Navigator is displayed.
- Procedure**
1. In the directory structure, select the folder in which the file is located.
  2. Select the file in the file list.
  3. Select **Edit > Rename**.
  4. Overwrite the file name with the new name and confirm with **OK**.

## 7.2 Selecting or opening a program

- Overview** A program can be selected or opened. Instead of the Navigator, an editor is then displayed with the program.

(>>> 7.2.1 "Selecting and deselecting a program" Page 202)

(>>> 7.2.2 "Opening a program" Page 203)

It is possible to toggle backwards and forwards between the program display and the Navigator.

(>>> 7.2.3 "Toggling between the Navigator and the program" Page 203)

### Differences

#### Program is selected:

- The block pointer is displayed.
- The program can be started.
- The program can be edited to a certain extent.  
Selected programs are particularly suitable for editing in the user group "User".  
Example: KRL instructions covering several lines (e.g. LOOP ... END-LOOP) are not permissible.
- When the program is deselected, modifications are accepted without a request for confirmation. If impermissible modifications are programmed, an error message is displayed.

#### Program is opened:

- The program cannot be started.
- The program can be edited.  
Opened programs are particularly suitable for editing in the user group "Expert".

- A request for confirmation is generated when the program is closed. Modifications can be accepted or rejected.

### 7.2.1 Selecting and deselecting a program



If a selected program is edited in the user group "Expert", the cursor must then be removed from the edited line and positioned in any other line!

Only in this way is it certain that the editing will be applied when the program is deselected again.

#### Precondition

- T1, T2 or AUT mode

#### Procedure

1. Select the program in the Navigator and press **Select**.

The program is displayed in the editor. It is irrelevant whether a module, an SRC file or a DAT file is selected. It is always the SRC file that is displayed in the editor.

2. Start or edit the program.

3. Deselect the program again:

Select **Edit > Cancel program**.

Or: In the status bar, touch the **Robot interpreter** status indicator. A window opens. Select **Cancel program**.



When the program is deselected, modifications are accepted without a request for confirmation!

If the program is running, it must be stopped before it can be deselected.

#### Description

If a program is selected, this is indicated by the **Robot interpreter** status indicator.

(>>> 7.5.6 "Robot interpreter status indicator" Page 209)

The screenshot shows the KUKA Editor interface. At the top, there is a toolbar with several icons. Below the toolbar, the title bar says "Editor". The main area of the editor contains the following program code:

```

1   INI
2
3   PTP HOME Vel= 100 % DEFAULT
4   loop
5   PTP P1 Vel=100 % PDAT1 Tool[1] Base[0]
6   PTP P2 Vel=100 % PDAT2 Tool[1] Base[0]
7   endloop

```

The line "4 loop" is highlighted with a red selection bar underneath it. The status bar at the bottom of the editor window displays the path "KRC:\R1\PROGRAM\MINILOOP.SRC" and the coordinates "Ln 4, Col 4".

**Fig. 7-2: Program is selected**

- 1 Block pointer
- 2 Cursor
- 3 Program path and file name
- 4 Position of the cursor in the program
- 5 The icon indicates that the program is selected.

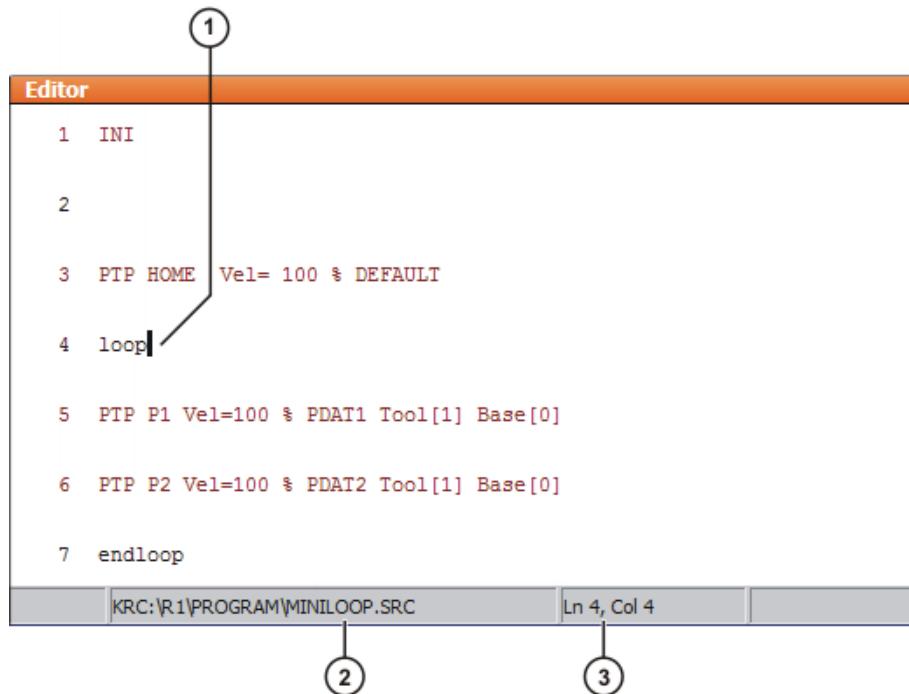
### 7.2.2 Opening a program

**Precondition** ■ T1, T2 or AUT mode

A program can be opened in AUT EXT mode, but not edited.

- Procedure**
1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
  - If a module has been selected, the SRC file is displayed in the editor. If an SRC file or DAT file has been selected, the corresponding file is displayed in the editor.
  2. Edit the program.
  3. Close the program.
  4. To accept the changes, answer the request for confirmation with **Yes**.

**Description**



**Fig. 7-3: Program is open**

- 1 Cursor
- 2 Program path and file name
- 3 Position of the cursor in the program

### 7.2.3 Toggling between the Navigator and the program

- Description** If a program is selected or open, it is possible to display the Navigator again without having to deselect or close the program. The user can then return to the program.

**Procedure****Program is selected:**

- Toggling from the program to the Navigator: select the menu sequence **Edit > Navigator**.
- Toggling from the Navigator to the program: press **PROGRAM**.

**Program is open:**

- Toggling from the program to the Navigator: select the menu sequence **Edit > Navigator**.
- Toggling from the Navigator to the program: press **EDITOR**.



Programs that are running or have been interrupted must first be stopped before the menu sequences and buttons referred to above are available.

### 7.3 Structure of a KRL program

```

1 DEF my_program( )
2 INI
3
4 PTP HOME Vel= 100 % DEFAULT
...
8 LIN point_5 CONT Vel= 2 m/s CPDAT1 Tool[3] Base[4]
...
14 PTP point_1 CONT Vel= 100 % PDAT1 Tool[3] Base[4]
...
20 PTP HOME Vel= 100 % DEFAULT
21
22 END

```

Line	Description
1	The DEF line indicates the name of the program. If the program is a function, the DEF line begins with "DEFFCT" and contains additional information. The DEF line can be displayed or hidden.  (>>> 7.4.1 "Displaying/hiding the DEF line" Page 205)
2	The INI line contains initializations for internal variables and parameters.
4	HOME position  (>>> 7.3.1 "HOME position" Page 205)
8	LIN motion  (>>> 9.2.3 "Programming a LIN motion" Page 252)
14	PTP motion  (>>> 9.2.1 "Programming a PTP motion" Page 251)
20	HOME position
22	The END line is the last line in any program. If the program is a function, the wording of the END line is "ENDFCT". The END line must not be deleted!

The first motion instruction in a KRL program must define an unambiguous starting position. The HOME position, which is stored by default in the robot controller, ensures that this is the case.

If the first motion instruction is not the default HOME position, or if this position has been changed, one of the following statements must be used:

- Complete PTP instruction of type POS or E6POS
- Complete PTP instruction of type AXIS or E6AXIS

“Complete” means that all components of the end point must be specified.



If the HOME position is modified, this affects all programs in which it is used. Physical injuries or damage to property may result.

In programs that are used exclusively as subprograms, different statements can be used as the first motion instruction.

### 7.3.1 HOME position

The HOME position is not program-specific. It is generally used as the first and last position in the program as it is uniquely defined and uncritical.

The HOME position is stored by default with the following values in the robot controller:

Axis	A1	A2	A3	A4	A5	A6
Position	0°	- 90°	+ 90°	0°	0°	0°

Additional HOME positions can be taught. A HOME position must meet the following conditions:

- Good starting position for program execution
- Good standstill position. For example, the stationary robot must not be an obstacle.



If the HOME position is modified, this affects all programs in which it is used. Physical injuries or damage to property may result.

## 7.4 Displaying/hiding program sections

### 7.4.1 Displaying/hiding the DEF line

**Description** By default, the DEF line is hidden. Declarations can only be made in a program if the DEF line is visible.

The DEF line is displayed and hidden separately for opened and selected programs. If detail view (ASCII mode) is activated, the DEF line is visible and does not need to be activated separately.

**Precondition**

- User group “Expert”
- Program is selected or open.

**Procedure**

- Select the menu sequence **Edit > View > DEF line**.

Check mark activated in menu: DEF line is displayed.

Check mark not activated in menu: DEF line is hidden.

### 7.4.2 Activating detail view

**Description** Detail view (ASCII mode) is deactivated by default to keep the program transparent. If detail view is activated, hidden program lines, such as the FOLD and ENDFOLD lines and the DEF line, are displayed.

Detail view is activated and deactivated separately for opened and selected programs.

**Precondition**

- Expert user group

- |                  |  |
|------------------|--|
| <b>Procedure</b> | <ul style="list-style-type: none"><li>■ Select the menu sequence <b>Edit &gt; View &gt; Detail view (ASCII)</b>.<br/>Check mark activated in menu: detail view is activated.<br/>Check mark not activated in menu: detail view is deactivated.</li></ul> |
|------------------|--|

#### 7.4.3 Activating/deactivating the line break function

- |                    |  |
|--------------------|--|
| <b>Description</b> | If a line is wider than the program window, the line is broken by default. The part of the line after the break has no line number and is marked with a black, L-shaped arrow. The line break function can be deactivated. |
|--------------------|--|

```
8 EXT IBGN {IBGN_COMMAND :IN,BOOL :IN,REAL :IN,REAL  
↓ :IN,BOOL :IN,E6POS :OUT }
```

**Fig. 7-4: Line break**

The line break function is activated and deactivated separately for opened and selected programs.

- |                     |   |
|---------------------|---|
| <b>Precondition</b> | <ul style="list-style-type: none"><li>■ User group “Expert”</li><li>■ Program is selected or open.</li></ul>  |
| <b>Procedure</b>    | <ul style="list-style-type: none"><li>■ Select the menu sequence <b>Edit &gt; View &gt; Line break</b>.<br/>Check mark activated in menu: line break function is activated.<br/>Check mark not activated in menu: line break function is deactivated.</li></ul> |

#### 7.4.4 Displaying Folds

- |                    |   |
|--------------------|---|
| <b>Description</b> | Folds are used to hide sections of the program. In this way, Folds make programs more transparent. The hidden program sections are processed during program execution in exactly the same way as normal program sections.   |
|                    | <ul style="list-style-type: none"><li>■ In the user group “User”, Folds are always closed. In other words, the contents of the Folds are not visible and cannot be edited.</li><li>■ In the user group “Expert”, Folds are closed by default. They can be opened and edited. New Folds can be created.<br/>(&gt;&gt;&gt; 7.6.3 "Creating Folds" Page 214)</li></ul> |

If a program is deselected, all Folds are automatically closed.

```
2  
  
3 PTP HOME Vel= 100 % DEFAULT  
  
4
```

**Fig. 7-5: Example of a closed Fold**

```

2

3 PTP HOME Vel= 100 % DEFAULT

4 $BWDSTART = FALSE

5 PDAT_ACT=PDEFAULT

6 FDAT_ACT=FHOME

7 BAS (#PTP_PARAMS,100 )

8 $H_POS=XHOME

9 PTP XHOME

10

```

**Fig. 7-6: Example of an open Fold**

Color coding of Folds:

Color	Description
Dark red	Closed fold
Light red	Opened fold
Dark blue	Closed sub-Fold
Light blue	Opened sub-Fold
Green	Contents of the Fold

- Precondition**
- User group “Expert”
  - Program is selected or open.
- Procedure**
1. Select the line containing the Fold.
  2. Press **Open/close fold**. The Fold then opens.
  3. To close the fold, press **Open/close fold** again.
- Alternatively, use the menu sequence **Edit > FOLD > Open all FOLDs** or **Close all FOLDs** to open or close all the Folds in a program at once.

## 7.5 Starting a program

### 7.5.1 Selecting the program run mode

- Procedure**
1. Touch the **Program run mode** status indicator. The **Program run mode** window is opened.
  2. Select the desired program run mode.  
(>>> 7.5.2 "Program run modes" Page 208)
- The window closes and the selected program run mode is applied.

### 7.5.2 Program run modes

Program run mode	Description
<b>Go</b> #GO	The program is executed through to the end without stopping.
<b>Motion</b> #MSTEP	The program is executed with a stop after each motion block. The Start key must be pressed again for each motion block.
<b>Single Step</b> #ISTEP	The program is executed with a stop after each program line. Program lines that cannot be seen and blank lines are also taken into consideration. The Start key must be pressed again for each line.  <b>Single Step</b> is only available to the user group "Expert".
Backward #BSTEP	This program run mode is automatically selected if the Start backwards key is pressed.



In **Motion Step** and **Single Step** modes, the program is executed without an advance run.

The following additional program run modes are available for systems integrators.

These program run modes can only be selected via the variable correction function. System variable for the program run mode: \$PRO\_MODE.

Program run mode	Description
Program Step #PSTEP	The program is executed step by step without an advance run. Subprograms are executed completely.
Continuous Step #CSTEP	Approximate positioning points are executed with advance processing, i.e. they are approximated.  Exact positioning points are executed without an advance run and with a stop after the motion instruction.

### 7.5.3 Advance run

The advance run is the maximum number of motion blocks that the robot controller calculates and plans in advance during program execution. The actual number is dependent on the capacity of the computer.

The advance run refers to the current position of the block pointer. It is set via the system variable \$ADVANCE:

- Default value: 3
- Maximum value: 5

The advance run is required, for example, in order to be able to calculate approximate positioning motions. If \$ADVANCE = 0 is set, approximate positioning is not possible.

Certain statements trigger an advance run stop. These include statements that influence the periphery, e.g. OUT statements.

### 7.5.4 Setting the program override (POV)

<b>Description</b>	Program override is the velocity of the robot during program execution. The program override is specified as a percentage of the programmed velocity.
--------------------	---

 In T1 mode, the maximum velocity is 250 mm/s, irrespective of the value that is set.

<b>Procedure</b>	<ol style="list-style-type: none"> <li>1. Touch the <b>POV/HOV</b> status indicator. The <b>Overrides</b> window is opened.</li> <li>2. Set the desired program override. It can be set using either the plus/minus keys or by means of the slide controller.           <ul style="list-style-type: none"> <li>■ Plus/minus keys: The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%</li> <li>■ Slide controller: The override can be adjusted in 1% steps.</li> </ul> </li> <li>3. Touch the <b>POV/HOV</b> status indicator again. (Or touch the area outside the window.)</li> </ol> <p>The window closes and the selected override value is applied.</p>
------------------	--

 The **Jog options** window can be opened via **Options** in the **Overrides** window.

<b>Alternative procedure</b>	Alternatively, the override can be set using the plus/minus key on the right-hand side of the KCP.
------------------------------	--

The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%.

### 7.5.5 Switching drives on/off

The status of the drives is indicated in the status bar. The drives can also be switched on or off here.

Icon	Color	Description
	Green	Drives ready.
	Red	Drives not ready.

### 7.5.6 Robot interpreter status indicator

Icon	Color	Description
	Gray	No program is selected.
	Yellow	The block pointer is situated on the first line of the selected program.
	Green	The program is selected and is being executed.

Icon	Color	Description
	Red	The selected and started program has been stopped.
	Black	The block pointer is situated at the end of the selected program.

### 7.5.7 Starting a program forwards (manual)

#### Precondition

- Program is selected.
- Operating mode T1 or T2.

#### Procedure

1. Select the program run mode.
2. Hold the enabling switch down and wait until the status bar indicates "Drives ready":



Fig. 7-7

3. Carry out a BCO run: Press Start key and hold it down until the message "Programmed path reached (BCO)" is displayed in the message window. The robot stops.



**WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Press Start key and hold it down.

The program is executed with or without stops, depending on the program run mode.

To stop a program that has been started manually, release the Start key.

### 7.5.8 Starting a program forwards (automatic)

#### Precondition

- A program is selected.
- Operating mode Automatic (not Automatic External)

#### Procedure

1. Select the program run mode **Go**.
2. Switch on the drives.
3. Carry out a BCO run:

Press Start key and hold it down until the message "Programmed path reached (BCO)" is displayed in the message window. The robot stops.



**WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Press the Start key. The program is executed.

To stop a program that has been started in Automatic mode, press the STOP key.

### 7.5.9 Carrying out a block selection

**Description** A program can be started at any point by means of a block selection.

**Precondition**

- Program is selected.
- Operating mode T1 or T2.

**Procedure**

1. Select the program run mode.
2. Select the motion block at which the program is to be started.
3. Press **Block selection**. The block pointer indicates the motion block.
4. Hold the enabling switch down and wait until the status bar indicates "Drives ready":



5. Carry out a BCO run: Press Start key and hold it down until the message "Programmed path reached (BCO)" is displayed in the message window. The robot stops.



**WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

6. The program can now be started manually or automatically. It is not necessary to carry out a BCO run again.

### 7.5.10 Starting a program backwards

**Description** In the case of backward motion, the robot stops at every point. Approximate positioning is not possible.

**Precondition**

- Program is selected.
- Operating mode T1 or T2.

**Procedure**

1. Hold the enabling switch down and wait until the status bar indicates "Drives ready":



2. Carry out a BCO run: Press Start key and hold it down until the message "Programmed path reached (BCO)" is displayed in the message window. The robot stops.



**WARNING** A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

3. Press Start backwards key.
4. Press Start backwards key again for each motion block.

### 7.5.11 Resetting a program

**Description** In order to restart an interrupted program from the beginning, it must be reset. This returns the program to the initial state.

**Precondition**

- Program is selected.

**Procedure**

- Select the menu sequence **Edit > Reset program**.

**Alternative procedure**

- In the status bar, touch the **Robot interpreter** status indicator. A window opens.  
Select **Reset program**.

**7.5.12 Starting Automatic External mode****WARNING**

There is no BCO run in Automatic External mode. This means that the robot moves to the first programmed position after the start at the programmed (not reduced) velocity and does not stop there.

**Precondition**

- Operating mode T1 or T2
- Inputs/outputs for Automatic External and the program CELL.SRC are configured.

**Procedure**

1. Select the program CELL.SRC in the Navigator. (This program is located in the folder "R1".)
2. Set program override to 100%. (This is the recommended setting. A different value can be set if required.)
3. Carry out a BCO run:  
Hold down the enabling switch. Then press the Start key and hold it down until the message "Programmed path reached (BCO)" is displayed in the message window.

**WARNING**

A BCO run is always executed as a PTP motion from the actual position to the target position. Observe the motion to avoid collisions. The velocity is automatically reduced during the BCO run.

4. Select "Automatic External" mode.
5. Start the program from a higher-level controller (PLC).

To stop a program that has been started in Automatic mode, press the STOP key.

**7.6 Editing a program****Overview**

- A running program cannot be edited.
- Programs cannot be edited in AUT EXT mode.



If a selected program is edited in the user group "Expert", the cursor must then be removed from the edited line and positioned in any other line!

Only in this way is it certain that the editing will be applied when the program is deselected again.

Action	Possible in user group ...?
Insert comment or stamp	<b>User:</b> Yes  <b>Expert:</b> Yes
Delete lines	<b>User:</b> Yes  <b>Expert:</b> Yes

Action	Possible in user group ...?
Create folds	<b>User:</b> No  <b>Expert:</b> Yes
Copy	<b>User:</b> No  <b>Expert:</b> Yes
Paste	<b>User:</b> No  <b>Expert:</b> Yes
Insert blank lines (press the Enter key)	<b>User:</b> No  <b>Expert:</b> Yes
Cut	<b>User:</b> No  <b>Expert:</b> Yes
Find	<b>User:</b> Yes  <b>Expert:</b> Yes  Possible for all user groups in an open program, even in AUT EXT mode.
Replace	<b>User:</b> No  <b>Expert:</b> Yes (program is open, not selected)
Programming with inline forms	<b>User:</b> Yes  <b>Expert:</b> Yes
KRL programming	<b>User:</b> Possible to a certain extent. KRL instructions covering several lines (e.g. LOOP ... ENDLOOP) are not permissible.  <b>Expert:</b> Yes

### 7.6.1 Inserting a comment or stamp

- Precondition**
- Program is selected or open.
  - T1, T2 or AUT mode
- Procedure**
1. Select the line after which the comment or stamp is to be inserted.
  2. Select the menu sequence **Commands > Comment > Normal or Stamp**.
  3. Enter the desired data. If a comment or stamp has already been entered previously, the inline form still contains the same entries.

- In the case of a comment, the box can be cleared using **New text** ready for entry of a new text.
- In the case of a stamp, the system time can also be updated using **New time** and the **NAME** box can be cleared using **New name**.

4. Save with **Cmd Ok**.

**Description**  
Comment

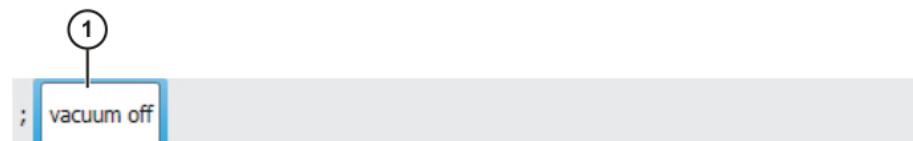


Fig. 7-8: Inline form “Comment”

Item	Description
1	Any text

**Description**  
Stamp

A stamp is a comment that is extended to include the system date and time and the user ID.



Fig. 7-9: Inline form “Stamp”

Item	Description
1	System date (cannot be edited)
2	System time
3	Name or ID of the user
4	Any text

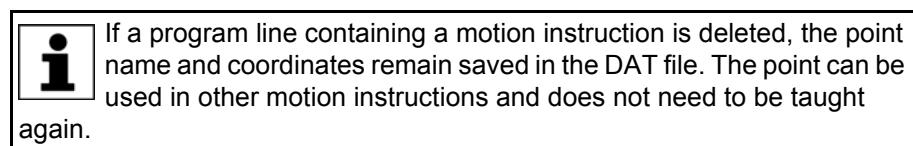
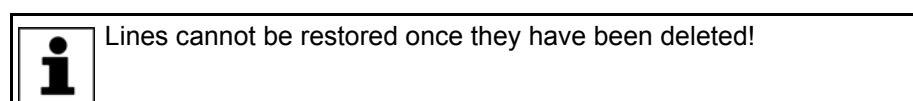
### 7.6.2 Deleting program lines

#### Precondition

- Program is selected or open.
- T1, T2 or AUT mode

#### Procedure

1. Select the line to be deleted. (The line need not have a colored background. It is sufficient for the cursor to be in the line.)  
If several consecutive lines are to be deleted: drag a finger or stylus across the desired area. (The area must now have a colored background.)
2. Select the menu sequence **Edit > Delete**.
3. Confirm the request for confirmation with **Yes**.



### 7.6.3 Creating Folds

#### Syntax

`; FOLD Name`

**Statements**

```
;ENDFOLD <Name>
```

The ENDFOLD lines can be assigned more easily if the name of the Fold is entered here as well. Folds can be nested.

**Precondition**

- User group “Expert”
- Program is selected or open.

**Procedure**

1. Enter Fold in program. A double semicolon prevents the Fold from closing when edited.

```
4
5  ;;FOLD outputs
6  $OUT[1] = TRUE
7  $OUT[2] = TRUE
8  ;;ENDFOLD outputs
9
```

**Fig. 7-10: Creating a sample Fold, step 1**

2. Delete the second semicolon.

```
4
5  ;FOLD outputs
6  $OUT[1] = TRUE
7  $OUT[2] = TRUE
8  ;ENDFOLD outputs
9
```

**Fig. 7-11: Creating a sample Fold, step 2**

3. Position the cursor in a line outside the Fold. The Fold closes.

```
4
5  outputs
6
```

**Fig. 7-12: Creating a sample Fold, step 3**

#### 7.6.4 Additional editing functions

The following additional program editing functions can be called using **Edit**:

##### **Copy**

Precondition:

- Program is selected or open.
- Expert user group
- T1, T2 or AUT mode

##### **Paste**

Precondition:

- Program is selected or open.
- Expert user group
- T1, T2 or AUT mode

##### **Cut**

Precondition:

- Program is selected or open.
- Expert user group
- T1, T2 or AUT mode

### Find

Precondition:

- Program is selected or open.

### Replace

Precondition:

- Program has been opened.
- Expert user group

## 7.7 Printing a program

### Procedure

1. Select the program in the Navigator. Multiple program selection is also possible.
2. Select the menu sequence **Edit > Print**.

## 7.8 Archiving and restoring data

### 7.8.1 Archiving overview

#### Target locations

Archiving can be performed to the following target destinations:

- USB stick in KCP or robot controller
- Network

#### Menu items

The following menu items are available:

("\*.\*" means all files and subdirectories.)

Menu item	Archives the directories/files
All	<ul style="list-style-type: none"><li>■ KRC:\*.*</li><li>■ C:\KRC\Roboter\Config\User\*.*</li><li>■ C:\KRC\Roboter\Config\System\Common\Mada\*.*</li><li>■ C:\KRC\Roboter\Init\*.*</li><li>■ C:\KRC\Roboter\Ir_Spec\*.*</li><li>■ C:\KRC\Roboter\Template\*.*</li><li>■ C:\KRC\Roboter\Rdc\*.*</li><li>■ C:\KRC\User\*.*</li><li>■ C:\KRC\Roboter\log\Mastery.log</li><li>■ Some additional log data</li></ul>
Applications	<ul style="list-style-type: none"><li>■ KRC:\R1\Program\*.*</li><li>■ KRC:\R1\System\*.*</li><li>■ KRC:\R1\cell\*.*</li><li>■ KRC:\Steu\\$config\*.*</li></ul>

Menu item	Archives the directories/files
<b>System data</b>	<ul style="list-style-type: none"> <li>■ KRC:\R1\Mada\*.*</li> <li>■ KRC:\R1\System\*.*</li> <li>■ KRC:\R1\TP\*.*</li> <li>■ KRC:\Steu\Mada\*.*</li> <li>■ C:\KRC\Roboter\Config\User\*.*</li> <li>■ C:\KRC\Roboter\Config\System\Common\Mada\*.*</li> <li>■ C:\KRC\Roboter\Init\*.*</li> <li>■ C:\KRC\Roboter\Ir_Spec\*.*</li> <li>■ C:\KRC\Roboter\Template\*.*</li> <li>■ C:\KRC\Roboter\Rdc\*.*</li> <li>■ C:\KRC\User\*.*</li> </ul>
<b>Log data</b>	<ul style="list-style-type: none"> <li>■ C:\KRC\Roboter\log\*.*</li> </ul> <p>Except: Poslog.xlsx and files with the extension DMP</p> <ul style="list-style-type: none"> <li>■ Some additional log data</li> </ul>
<b>KrcDiag</b>	<p>If it is necessary for an error to be analyzed by KUKA Roboter GmbH, this menu item can be used to compress the data for sending to KUKA.</p> <p>In addition to the menu sequence <b>File &gt; Archive</b>, there are other methods available for compressing these data.</p> <p>(&gt;&gt;&gt; 12.5 "Compressing data for error analysis at KUKA" Page 362)</p>

If archiving is carried out using the menu item **All** and there is an existing archive present, this will be overwritten.

If archiving is carried out using a menu item other than **All** or **KrcDiag** and an archive is already available, the robot controller compares its robot name with that in the archive. If the names are different, a request for confirmation is generated.

If archiving is carried out repeatedly via **KrcDiag**, a maximum of 10 archives can be created. Further archives will overwrite the oldest existing archive.

The logbook can also be activated. (>>> 7.8.4 "Archiving the logbook" Page 218)

## 7.8.2 Archiving to a USB stick

### Description

This procedure generates a ZIP file on the stick. By default, this file has the same name as the robot. A different name can be defined for the file, however, under **Robot data**.

(>>> 4.15.14 "Displaying/editing robot data" Page 78)

The archive is displayed in the ARCHIVE:\ directory in the Navigator. Archiving is also carried out automatically to D:\ as well as to the stick. The file INTERN.ZIP is generated here.

Special case **KrcDiag**:

This menu item generates the folder **KRCdiag** on the stick. This contains a ZIP file. The ZIP file is also automatically archived in C:\KUKA\KRCdiag.

### NOTICE

Use only a KUKA.USB data stick: if a different USB stick is used, data may be lost or changed.

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ A KUKA.USB data stick is connected. The stick can be connected to the KCP or robot controller.</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>File &gt; Archive &gt; USB (KCP) or USB (cabinet)</b> and then the desired menu item.</li><li>2. Confirm the request for confirmation with <b>Yes</b>. The archive is created. Once the archiving is completed, this is indicated in the message window. <b>Special case KrcDiag:</b> If archiving is carried out using this menu item, a separate window indicates when archiving has been completed. The window is then automatically hidden again.</li><li>3. The stick can be removed when the LED on the stick is no longer lit.</li></ol>

### 7.8.3 Archiving on the network

<b>Description</b>	This procedure generates a ZIP file on the network path. By default, this file has the same name as the robot. A different name can be defined for the file, however, under <b>Robot data</b> .  The network path to which archiving is to be carried out must be configured under <b>Robot data</b> .  (>>> 4.15.14 "Displaying/editing robot data" Page 78)
	The archive is displayed in the ARCHIVE:\ directory in the Navigator. Archiving is also carried out automatically to D:\ as well as to the network path. The file INTERN.ZIP is generated here.  <b>Special case KrcDiag:</b> This menu item generates the folder <b>KRCdiag</b> on the network path. This contains a ZIP file. The ZIP file is also automatically archived in C:\KUKA\KRCdiag.

<b>Precondition</b>	<ul style="list-style-type: none"><li>■ The network path to which the data are to be archived is configured.</li></ul>
<b>Procedure</b>	<ol style="list-style-type: none"><li>1. In the main menu, select <b>File &gt; Archive &gt; Network</b> and then the desired menu item.</li><li>2. Confirm the request for confirmation with <b>Yes</b>. The archive is created. Once the archiving is completed, this is indicated in the message window. <b>Special case KrcDiag:</b> If archiving is carried out using this menu item, a separate window indicates when archiving has been completed. The window is then automatically hidden again.</li></ol>

### 7.8.4 Archiving the logbook

<b>Description</b>	The file "Logbuch.txt" is generated as an archive in the directory C:\KRC\ROBOTER\LOG.
<b>Procedure</b>	<ul style="list-style-type: none"><li>■ In the main menu, select <b>File &gt; Archive &gt; Logbook</b>.</li></ul> <p>The archive is created. Once the archiving is completed, this is indicated in the message window.</p>

## 7.8.5 Restoring data

### Description

 <b>WARNING</b>	<p>Only KSS 8.2 archives may be loaded into KSS 8.2. If other archives are loaded, the following may occur:</p> <ul style="list-style-type: none"> <li>■ Error messages</li> <li>■ Robot controller is not operable.</li> <li>■ Personal injury and damage to property.</li> </ul>
--	--

The following menu items are available for restoring data:

- **All**
- **Applications**
- **System data**

If the archived files are not the same version as the files present in the system, an error message is generated during restoration.

Similarly, if the version of the archived technology packages does not match the installed version, an error message is generated.

### Precondition

- If data are to be restored from the USB stick: A KUKA.USB data stick with the archive is connected.
- The stick can be connected to the KCP or robot controller.

### Procedure

1. In the main menu, select **File > Restore** and then the desired subitems.
2. Confirm the request for confirmation with **Yes**. Archived files are restored to the robot controller. A message indicates completion of the restoration process.
3. When data have been restored from the USB stick: remove the stick when the LED on the stick is no longer lit.
4. Reboot the robot controller.

## 7.9 Project management

### 7.9.1 Pinning a project on the robot controller

#### Description

Projects that are present on the robot controller can be pinned. A project can be pinned directly on the robot controller or in WorkVisual.

Pinned projects cannot be changed, activated or deleted. They can be copied or unpinned, however. A project can thus be pinned e.g. to prevent it from being accidentally deleted.



Information about how projects can be pinned via WorkVisual can be found in the **WorkVisual** documentation.

#### Precondition

- Expert user group

#### Procedure

1. Select the menu sequence **File > Project management**.
2. Select the desired project and press the **Pin** button. The project is pinned and labeled with a pin symbol in the project list.  
(>>> 7.9.3 "Project management window" Page 221)
3. The project can be unpinned again by pressing the **Unpin** button.

## 7.9.2 Activating a project

### Precondition

- User group Expert or higher

Restriction: If the activation would cause changes in the area **Safety-relevant communication parameters**, the user group "Safety recovery" or higher must be selected.

- If the operating mode AUT or AUT EXT is selected: The project can only be activated if this affects only KRL programs. If the project contains settings that would cause other changes, it cannot be activated.



If one of the options KUKA.SafeOperation or KUKA.SafeRangeMonitoring is installed on the robot controller, different user groups may apply. Information can be found in the documentation for these options.

### Procedure

1. Select the menu sequence **File > Project management**. The **Project management** window is opened.  
(>>> 7.9.3 "Project management window" Page 221)
2. Select the desired project and activate it using the **Activate** button.
3. The KUKA smartHMI displays the request for confirmation *Do you want to activate the project [...]?*. In addition, a message is displayed as to whether the activation would overwrite a project, and if so, which.  
If no relevant project will be overwritten: Confirm with **Yes** within 30 minutes.
4. An overview is displayed of the changes which will be made in comparison to the project that is still active on the robot controller. The check box **Details** can be used to display details about the changes.



**WARNING** If changes are listed in the overview under the heading **Safety-relevant communication parameters**, this means that the behavior of the Emergency Stop and "Operator safety" signal may have changed compared with the previous project.  
After activation of the project, the Emergency Stop and the "Operator safety" signal must be checked for safe functioning. If the project is activated on several robot controllers, this check must be carried out for every robot controller. Failure to carry out this check may result in death to persons, severe physical injuries or considerable damage to property.

5. The overview displays the request for confirmation *Do you want to continue?*. Confirm with **Yes**. The project is activated on the robot controller.



**WARNING** After activation of a project on the robot controller, the safety configuration must be checked there! If this is not done, the robot will possibly be operated with incorrect data. Death to persons, severe physical injuries or considerable damage to property may result.

(>>> 6.3 "Checking the safety configuration of the robot controller" Page 141)



**WARNING** If the activation of a project fails, an error message is displayed. In this case, one of the following measures must be carried out:

- Either: Activate a project again (the same one or a different one).
- Or: Reboot the robot controller with a cold restart.



In the case of a KSS/VSS update (e.g. from 8.2.3 to 8.2.4), the initial project and base project are overwritten by copies of the active project.

### 7.9.3 Project management window

#### Overview

In addition to the regular projects, the **Project management** window contains the following special projects:

Project	Description
Initial project	The initial project is always present. It cannot be changed by the user. It contains the initial state of the robot controller as shipped.
Base project	<p>The user can save the active project as the base project. This functionality is generally used to save a functional, tried-and-tested project state.</p> <p>The base project cannot be activated, but copied. The base project can no longer be changed by the user. It can, however, be overwritten by saving a new base project (after a request for confirmation).</p> <p>If a project is activated which does not contain all the configuration files, the missing information is inserted from the base project. This is the case e.g. if a project is activated from an earlier version of WorkVisual. (The configuration files include machine data files, safety configuration files and many others.)</p>



In the case of a KSS/VSS update (e.g. from 8.2.3 to 8.2.4), the initial project and base project are overwritten by copies of the active project.

#### Description

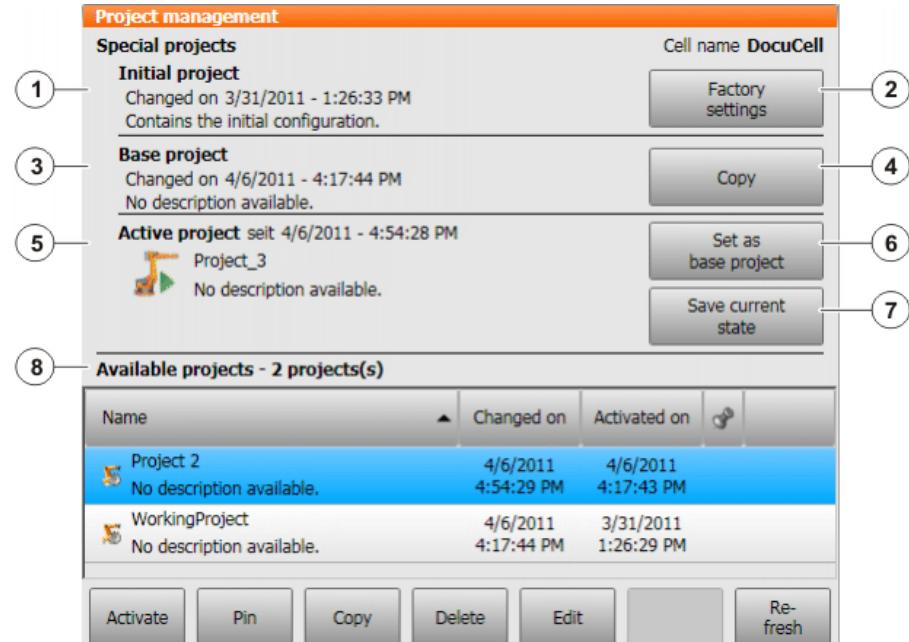


Fig. 7-13: “Projects management” window

Item	Description
1	The initial project is displayed.
2	Restores the factory settings of the robot controller. Only available to the user group “Expert” or higher.
3	The base project is displayed.

Item	Description
4	Creates a copy of the base project. Only available to the user group "Expert" or higher.
5	The active project is displayed.
6	Saves the active project as the base project. The active project remains active. Only available to the user group "Expert" or higher.
7	Creates a pinned copy of the active project. Only available to the user group "Expert" or higher.
8	List of projects. The active project is not displayed here.

With all copying operations, a window opens in which a name and a description can be entered for the copy.

## Buttons

The following buttons are available:

Button	Description
<b>Activate</b>	Activates the selected project.  If the selected project is pinned: Creates a copy of the selected project. (A pinned project cannot be activated itself, only a copy of it.) The user can then decide whether to activate this copy or whether the current project should remain active.  Only available to the user group "Expert" or higher.
<b>Pin</b>	(>>> 7.9.1 "Pinning a project on the robot controller" Page 219)  Only available if an unpinned project is selected. Only available to the user group "Expert" or higher.
<b>Unpin</b>	Unpins the project.  Only available if a pinned project is selected. Only available to the user group "Expert" or higher.
<b>Copy</b>	Copies the selected project.  Only available to the user group "Expert" or higher.
<b>Delete</b>	Deletes the selected project.  Only available if a non-activated, unpinned project is selected. Only available to the user group "Expert" or higher.
<b>Edit</b>	Opens a window in which the name and/or description of the selected project can be changed.  Only available if an unpinned project is selected. Only available to the user group "Expert" or higher.
<b>Refresh</b>	Refreshes the project list. This enables e.g. projects to be displayed which have been transferred to the robot controller since the display was opened.

## 8 Basic principles of motion programming

### 8.1 Overview of motion types

The following motion types can be programmed:

- Point-to-point motions (PTP)  
(>>> 8.2 "Motion type PTP" Page 223)
- Linear motions (LIN)  
(>>> 8.3 "Motion type LIN" Page 223)
- Circular motions (CIRC)  
(>>> 8.4 "Motion type CIRC" Page 224)
- Spline motions  
(>>> 8.7 "Motion type "Spline"" Page 229)

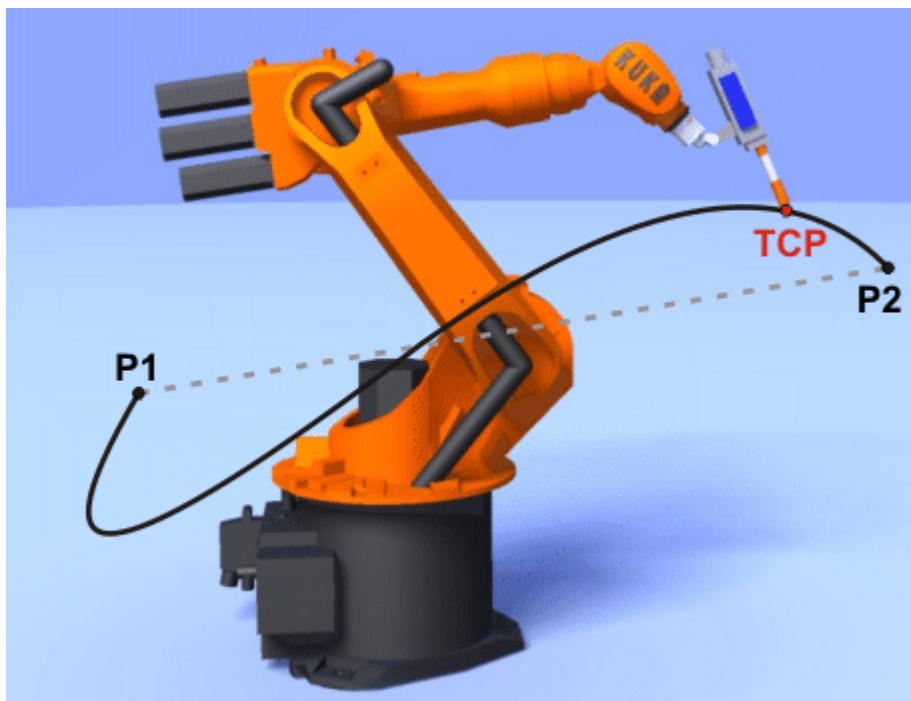
LIN, CIRC and spline motions are also known as CP ("Continuous Path") motions.

The start point of a motion is always the end point of the previous motion.

### 8.2 Motion type PTP

The robot guides the TCP along the fastest path to the end point. The fastest path is generally not the shortest path and is thus not a straight line. As the motions of the robot axes are rotational, curved paths can be executed faster than straight paths.

The exact path of the motion cannot be predicted.



**Fig. 8-1: PTP motion**

### 8.3 Motion type LIN

The robot guides the TCP at a defined velocity along a straight path to the end point.

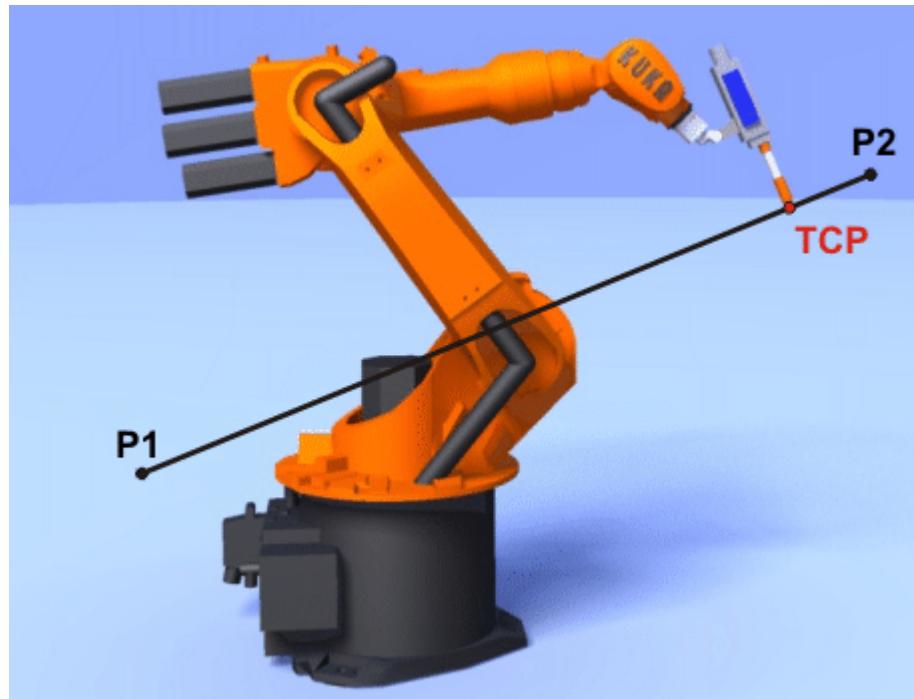


Fig. 8-2: LIN motion

#### 8.4 Motion type CIRC

The robot guides the TCP at a defined velocity along a circular path to the end point. The circular path is defined by a start point, auxiliary point and end point.

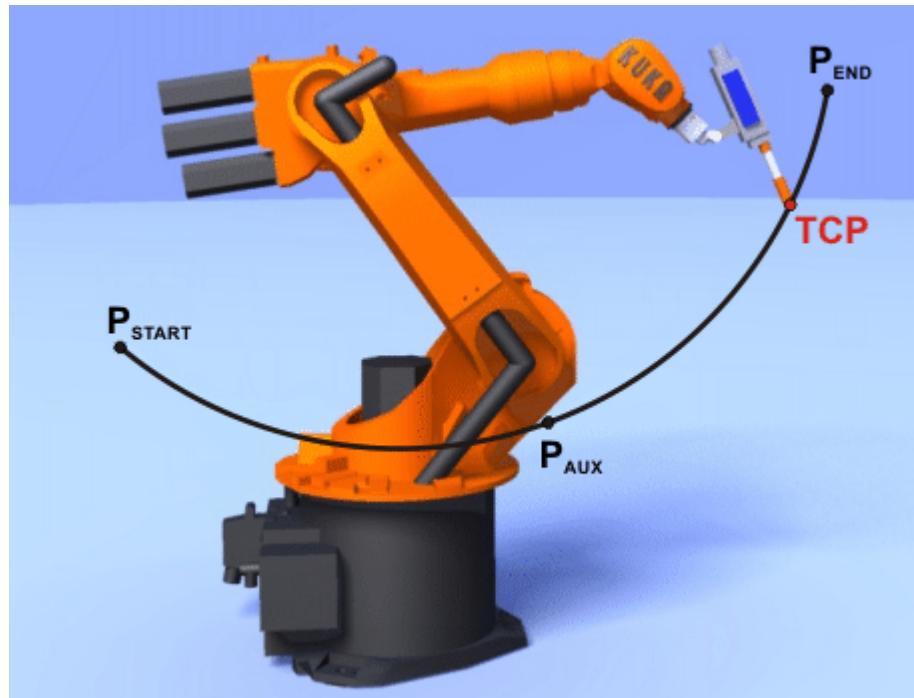


Fig. 8-3: CIRC motion

#### 8.5 Approximate positioning

Approximate positioning means that the motion does not stop exactly at the programmed point. Approximate positioning is an option that can be selected during motion programming.

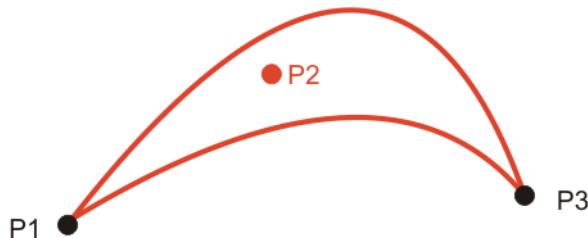


Approximate positioning is not possible if the motion instruction is followed by an instruction that triggers an advance run stop.

### PTP motion

The TCP leaves the path that would lead directly to the end point and moves along a faster path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The path of an approximated PTP motion cannot be predicted. It is also not possible to predict which side of the approximated point the path will run.

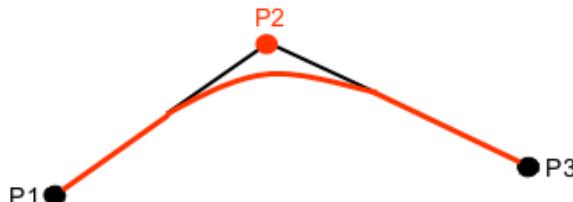


**Fig. 8-4: PTP motion, P2 is approximated**

### LIN motion

The TCP leaves the path that would lead directly to the end point and moves along a shorter path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The path in the approximate positioning range is **not** an arc.



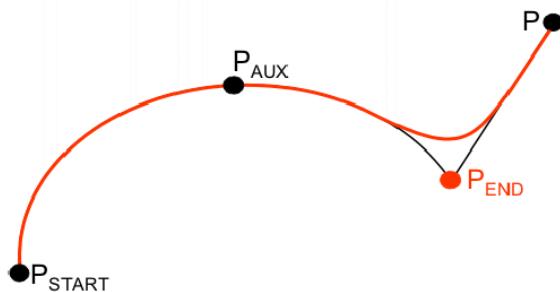
**Fig. 8-5: LIN motion, P2 is approximated**

### CIRC motion

The TCP leaves the path that would lead directly to the end point and moves along a shorter path. During programming of the motion, the maximum distance from the end point at which the TCP may deviate from its original path is defined.

The motion always stops exactly at the auxiliary point.

The path in the approximate positioning range is **not** an arc.



**Fig. 8-6: CIRC motion,  $P_{END}$  is approximated**

## 8.6 Orientation control LIN, CIRC

### Description

The orientation of the TCP can be different at the start point and end point of a motion. There are several different types of transition from the start orientation to the end orientation. A type must be selected when a CP motion is programmed.

The orientation control for LIN and CIRC motions is defined as follows:

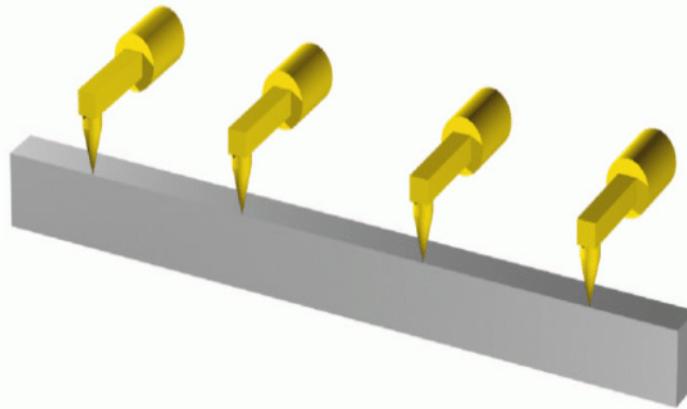
- In the option window **Motion parameter**  
(>>> 9.2.9 "Option window: Motion parameters (LIN, CIRC)" Page 256)
- Or via the system variable \$ORI\_TYPE

### LIN motion

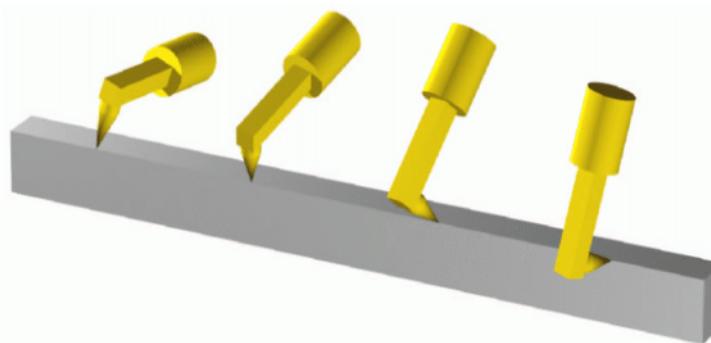
Orientation control	Description
<ul style="list-style-type: none"> <li>■ Option window: <b>Constant orientation</b></li> <li>■ \$ORI_TYPE = #CONSTANT</li> </ul>	<p>The orientation of the TCP remains constant during the motion.</p> <p>The programmed orientation is disregarded for the end point and that of the start point is retained.</p>
<ul style="list-style-type: none"> <li>■ Option window: <b>Standard</b></li> <li>■ \$ORI_TYPE = #VAR</li> </ul>	<p>The orientation of the TCP changes continuously during the motion.</p> <p><b>Note:</b> If, with <b>Standard</b>, the robot passes through a wrist axis singularity, use <b>Wrist PTP</b> instead.</p>
<ul style="list-style-type: none"> <li>■ Option window: <b>Wrist PTP</b></li> <li>■ \$ORI_TYPE = #JOINT</li> </ul>	<p>The orientation of the TCP changes continuously during the motion. This is done by linear transformation (axis-specific motion) of the wrist axis angles.</p> <p><b>Note:</b> Use <b>Wrist PTP</b> if, with <b>Standard</b>, the robot passes through a wrist axis singularity.</p> <p>The orientation of the TCP changes continuously during the motion, but not uniformly. <b>Wrist PTP</b> is thus not suitable if a specific orientation must be maintained exactly, e.g. in the case of laser welding.</p>



If a wrist axis singularity occurs with **Standard** and the desired orientation cannot be maintained exactly enough with **Wrist PTP**, the following remedy is recommended:  
Re-teach start and/or end point. Select orientations that prevent a wrist axis singularity from occurring and allow the path to be executed with **Standard**.



**Fig. 8-7: Orientation control - Constant**



**Fig. 8-8: Standard or Wrist PTP**

#### CIRC motion

During CIRC motions, the robot controller only takes the programmed orientation of the end point into consideration. The programmed orientation of the auxiliary point is disregarded.

The same orientation control options are available for selection for CIRC motions as for LIN motions.

It is also possible to define for CIRC motions whether the orientation control is to be base-related or path-related. This is defined via the system variable \$CIRC\_TYPE.

Orientation control	Description
\$CIRC_TYPE = #BASE	Base-related orientation control during the circular motion
\$CIRC_TYPE = #PATH	Path-related orientation control during the circular motion



\$CIRC\_TYPE is meaningless if \$ORI\_TYPE = #JOINT.

(>> 8.6.1 "Combinations of \$ORI\_TYPE and \$CIRC\_TYPE" Page 227)

#### 8.6.1 Combinations of \$ORI\_TYPE and \$CIRC\_TYPE

**\$ORI\_TYPE = #CONSTANT, \$CIRC\_TYPE = #PATH:**

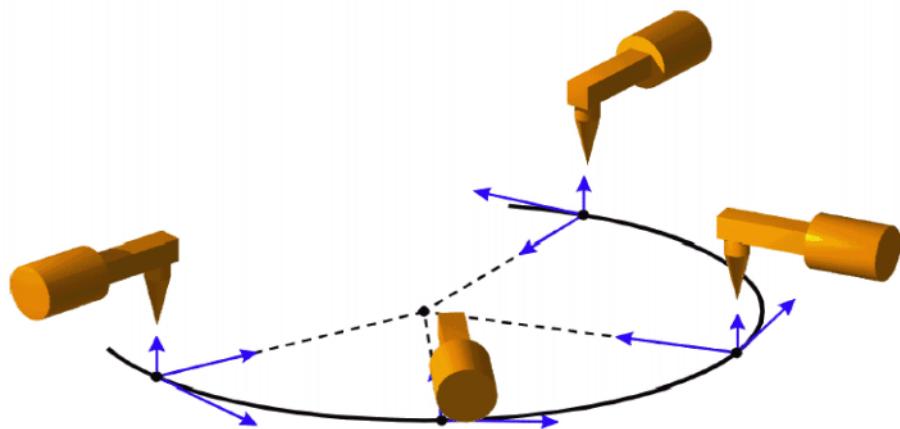


Fig. 8-9: Constant orientation, path-related

`$ORI_TYPE = #VAR, $CIRC_TYPE = #PATH:`

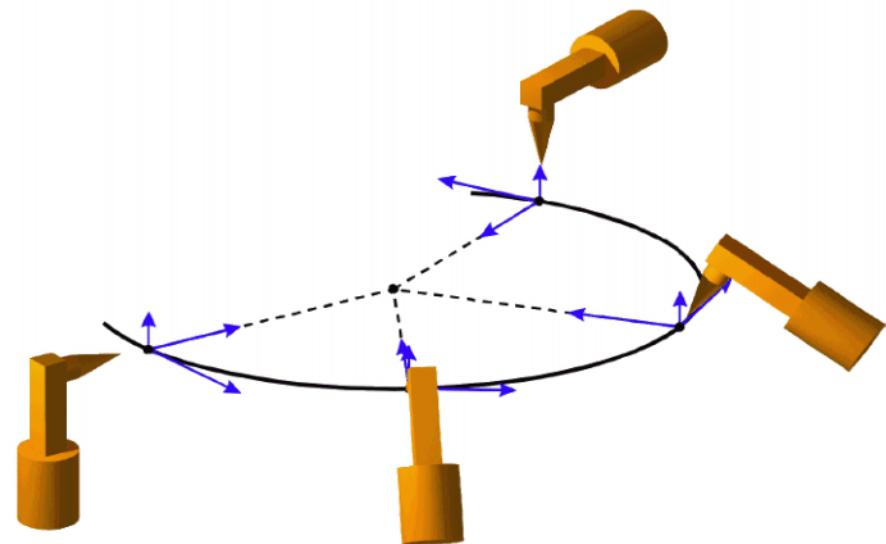


Fig. 8-10: Variable orientation, path-related

`$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #BASE:`

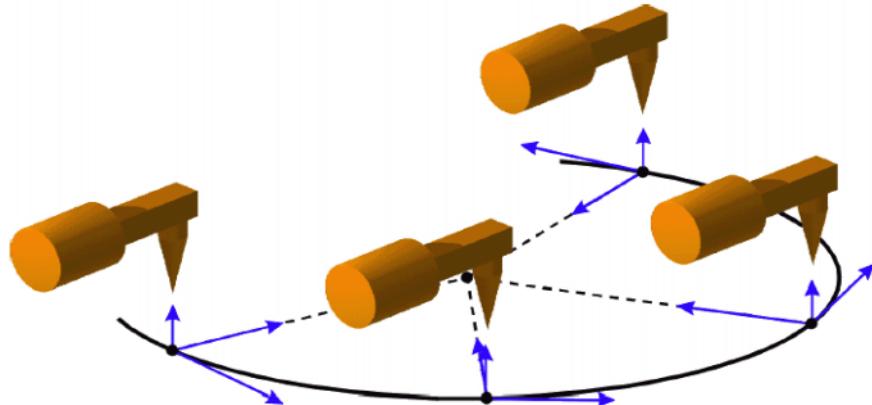
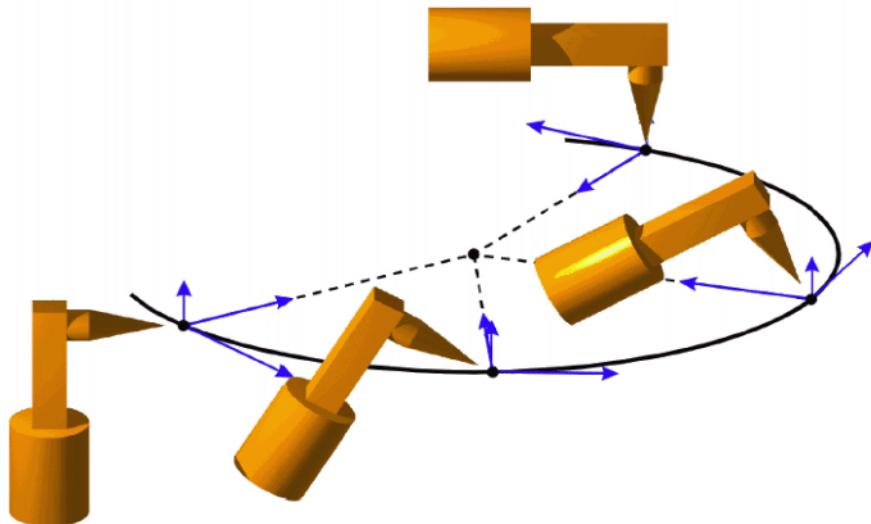


Fig. 8-11: Constant orientation, base-related

**\$ORI\_TYPE = #VAR, \$CIRC\_TYPE = #BASE:**



**Fig. 8-12: Variable orientation, base-related**

### 8.7 Motion type “Spline”

“Spline” is a Cartesian motion type that is suitable for particularly complex, curved paths. Such paths can generally also be generated using approximated LIN and CIRC motions, but Spline nonetheless has advantages.

#### Disadvantages of approximated LIN and CIRC motions:

- The path is defined by means of approximated points that are not located on the path. The approximate positioning ranges are difficult to predict. Generating the desired path is complicated and time-consuming.
- In many cases, the velocity may be reduced in a manner that is difficult to predict, e.g. in the approximate positioning ranges and near points that are situated close together.
- The path changes if approximate positioning is not possible, e.g. for time reasons.
- The path changes in accordance with the override setting, velocity or acceleration.

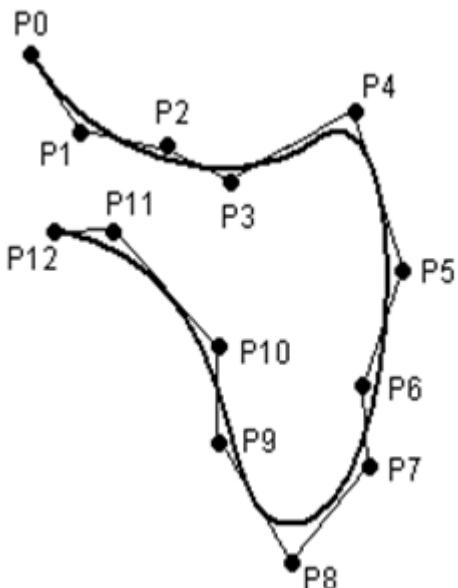


Fig. 8-13: Curved path with LIN

#### Advantages of Spline:

- The path is defined by means of points that are located on the path. The desired path can be generated easily.
- The programmed velocity is maintained. There are few cases in which the velocity is reduced.  
(>>> 8.7.1 "Velocity profile for spline motions" Page 231)
- The path always remains the same, irrespective of the override setting, velocity or acceleration.
- Circles and tight radii are executed with great precision.

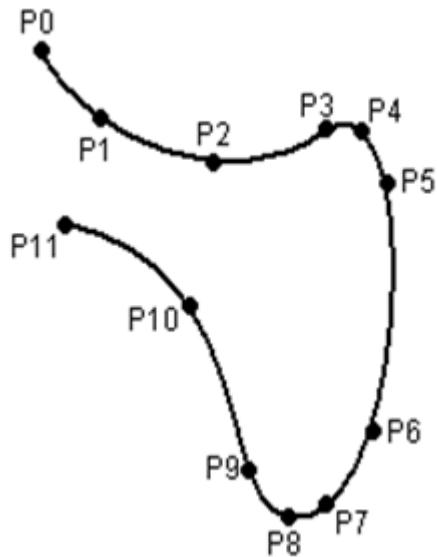


Fig. 8-14: Curved path with spline block

A spline motion can consist of several individual motions: spline segments. These are taught separately. The segments are grouped together to form the overall motion in a so-called spline block. A spline block is planned and executed by the robot controller as a single motion block.

Furthermore, individual SLIN and SCIRC motions are possible (without spline block).

Further characteristics of all spline motions:

- If all points are situated on a plane, then the path is also situated in this plane.
- If all points are situated on a straight line, then the path is also a straight line.

### 8.7.1 Velocity profile for spline motions

The path always remains the same, irrespective of the override setting, velocity or acceleration. Only dynamic effects can cause deviations at different velocities.

The programmed acceleration is valid not only for the direction along the path, but also perpendicular to the path. The same applies to the jerk limitation. Effects include the following:

- In the case of circles, the centrifugal acceleration is taken into consideration. The velocity that can be achieved thus also depends on the programmed acceleration and the radius of the circle.
- In the case of curves, the maximum permissible velocity is derived from the radius of the curve, the acceleration and the jerk limitation.

#### Reduction of the velocity

In the case of spline motions, the velocity may, under certain circumstances, fall below the programmed velocity. This occurs particularly in the case of:

- Tight corners
- Major reorientation
- Large motions of the external axes



If the points are close together, the velocity is not reduced.

#### Reduction of the velocity to 0

This is the case for:

- Successive points with the same Cartesian coordinates.
- Successive SLIN and/or SCIRC segments. Cause: inconstant velocity direction.

In the case of SLIN-SCIRC transitions, the velocity is also reduced to 0 if the straight line is a tangent of the circle, as the circle, unlike the straight line, is curved.

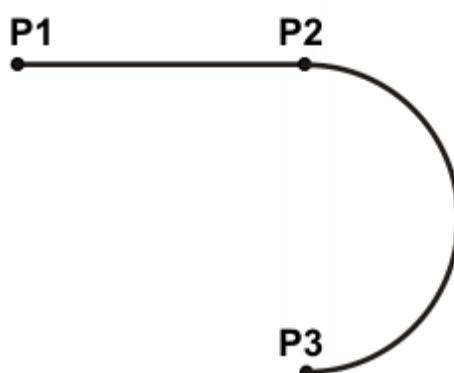
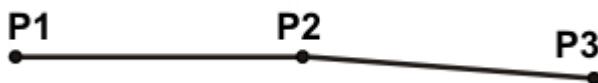


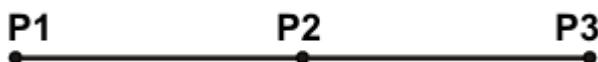
Fig. 8-15: Exact positioning at P2



**Fig. 8-16: Exact positioning at P2**

Exceptions:

- In the case of successive SLIN segments that result in a straight line and in which the orientations change uniformly, the velocity is not reduced.



**Fig. 8-17: P2 is executed without exact positioning.**

- In the case of a SCIRC-SCIRC transition, the velocity is not reduced if both circles have the same center point and the same radius and if the orientations change uniformly. (This is difficult to teach, so calculate and program points.)



Circles with the same center point and the same radius are sometimes programmed to obtain circles  $\geq 360^\circ$ . A simpler method is to program a circular angle.

### 8.7.2 Block selection with spline motions

#### Spline block

A spline block is planned and executed by the robot controller as a single motion block. Block selection to the spline segments is nonetheless possible. The BCO run is executed as a LIN motion. This is indicated by means of a message that must be acknowledged.

If the second segment in the spline block is an SPL segment, a modified path is executed in the following cases:

- Block selection to the first segment in the spline block
- Block selection to the spline block
- Block selection to a line before the spline block if this does not contain a motion instruction and if there is no motion instruction before the spline block

If the Start key is pressed after the BCO run, the modified path is indicated by means of a message that must be acknowledged.

#### Example:

```

1 PTP P0
2 SPLINE
3 SPL P1
4 SPL P2
5 SPL P3
6 SPL P4
7 SCIRC P5, P6
8 SPL P7
9 SLIN P8
10 ENDSPLINE

```

Line	Description
2	Start of the spline block

Line	Description
3 ... 9	Spline segments
10	End of the spline block

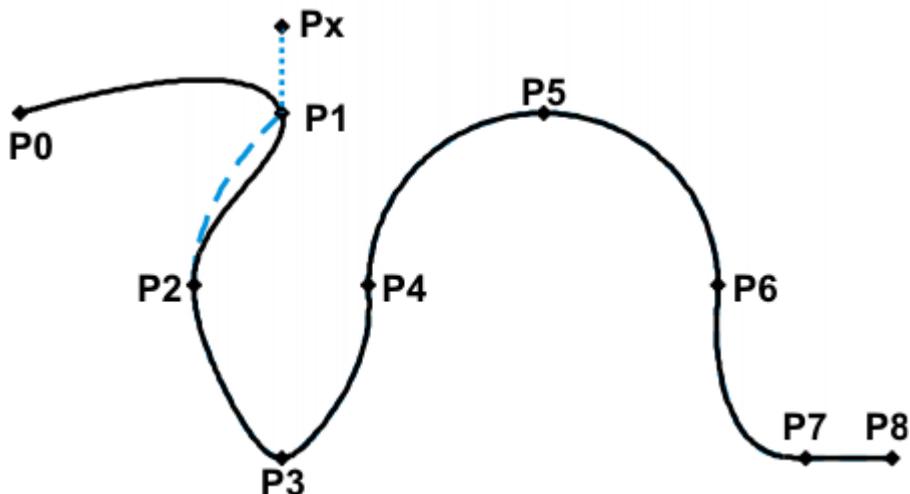


Fig. 8-18: Example: modified path in the case of block selection to P1

#### SCIRC

In the case of block selection to a SCIRC instruction for which a circular angle has been programmed, the motion is executed to the end point including the circular angle, provided that the robot controller knows the start point. If this is not possible, the motion is executed to the programmed end point. In this case, a message is generated, indicating that the circular angle is not being taken into consideration.

Position/type of SCIRC instruction	End point for block selection
SCIRC segment is 1st segment in the spline block	Circular angle is not taken into consideration
Other SCIRC segments in the spline block	Circular angle is taken into consideration
Individual SCIRC motions	Circular angle is not taken into consideration

#### 8.7.3 Modifications to spline blocks

##### Description

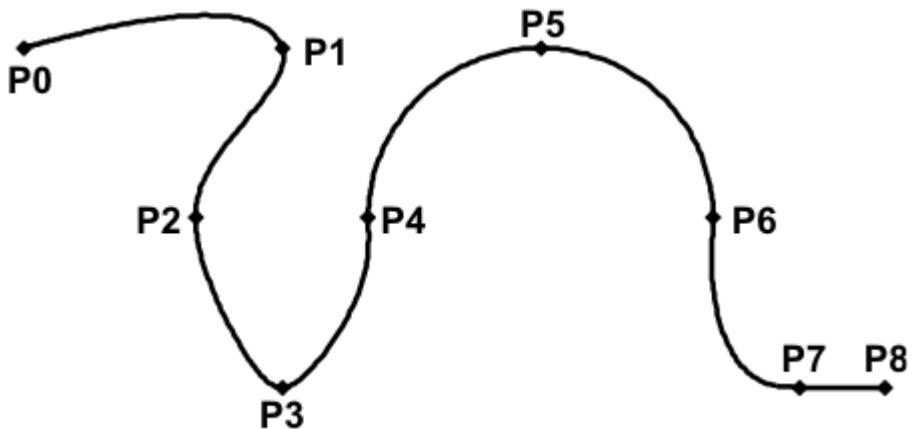
- Modification of the position of the point:  
If a point within a spline block is offset, the path is modified, at most, in the 2 segments before this point and the 2 segments after it.  
Small point offsets generally result in small modifications to the path. If, however, very long segments are followed by very short segments or vice versa, small modifications can have a very great effect, as the tangents and curves can change very greatly in such cases.
- Modification of the segment type:  
If an SPL segment is changed into an SLIN segment or vice versa, the path changes in the previous segment and the next segment.

**Example 1**

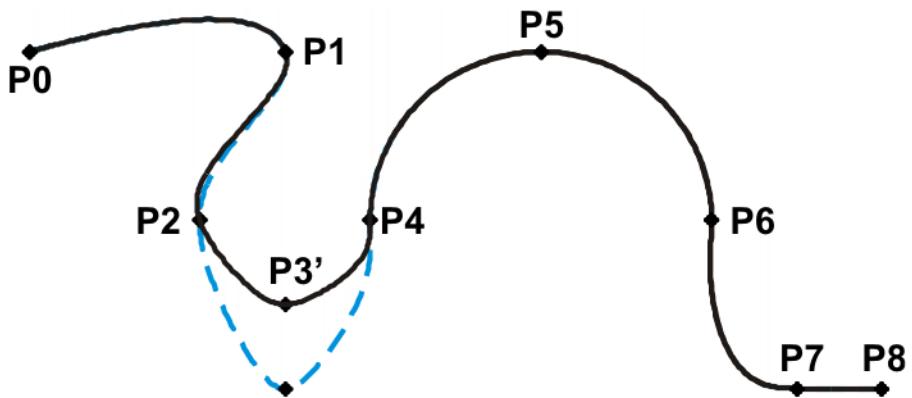
```

PTP P0
SPLINE
SPL P1
SPL P2
SPL P3
SPL P4
SCIRC P5, P6
SPL P7
SLIN P8
ENDSPLINE

```

**Fig. 8-19: Original path**

P3 is offset. This causes the path to change in segments P1 - P2, P2 - P3 and P3 - P4. Segment P4 - P5 is not changed in this case, as it belongs to an SCIRC and a circular path is thus defined.

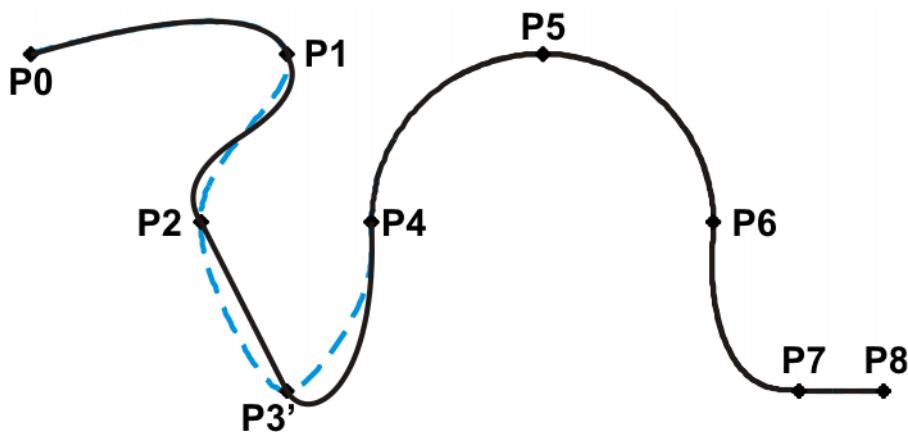
**Fig. 8-20: Point has been offset**

In the original path, the segment type of P2 - P3 is changed from SPL to SLIN. The path changes in segments P1 - P2, P2 - P3 and P3 - P4.

```

PTP P0
SPLINE
SPL P1
SPL P2
SLIN P3
SPL P4
SCIRC P5, P6
SPL P7
SLIN P8
ENDSPLINE

```



**Fig. 8-21: Segment type has been changed**

### Example 2

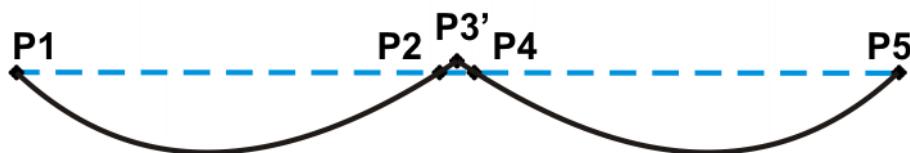
```
...
SPLINE
SPL {X 100, Y 0, ...}
SPL {X 102, Y 0}
SPL {X 104, Y 0}
SPL {X 204, Y 0}
ENDSPLINE
```



**Fig. 8-22: Original path**

P3 is offset. This causes the path to change in all the segments illustrated. Since P2 - P3 and P3 - P4 are very short segments and P1 - P2 and P4 - P5 are long segments, the slight offset causes the path to change greatly.

```
...
SPLINE
SPL {X 100, Y 0, ...}
SPL {X 102, Y 1}
SPL {X 104, Y 0}
SPL {X 204, Y 0}
ENDSPLINE
```



**Fig. 8-23: Point has been offset**

Remedy:

- Distribute the points more evenly
- Program straight lines (except very short ones) as SLIN segments

### 8.7.4 Approximate positioning of spline motions

Approximate positioning between spline motions (individual SLIN and SCIRC motions and spline blocks) is possible.

Approximate positioning between spline motions and LIN, CIRC or PTP is not possible.

#### Approximation not possible due to time or advance run stop:

If approximation is not possible for reasons of time or due to an advance run stop, the robot waits at the start of the approximate positioning arc.

- In the case of time reasons: the robot moves again as soon as it has been possible to plan the next block.
- In the case of an advance run stop: the end of the current block is reached at the start of the approximate positioning arc. This means that the advance run stop is canceled and the robot controller can plan the next block. Robot motion is resumed.

In both cases, the robot now moves along the approximate positioning arc. Approximate positioning is thus technically possible; it is merely delayed.

This response differs from that for LIN, CIRC or PTP motions. If approximate positioning is not possible for the reasons specified, the motion is executed to the end point with exact positioning.

#### No approximate positioning in MSTEP and ISTEP:

In the program run modes MSTEP and ISTEP, the robot stops exactly at the end point, even in the case of approximated motions.

In the case of approximate positioning from one spline block to another spline block, the result of this exact positioning is that the path is different in the last segment of the first block and in the first segment of the second block from the path in program run mode GO.

In all other segments of both spline blocks, the path is identical in MSTEP, ISTEP and GO.

### 8.7.5 Replacing an approximated motion with a spline block

#### Description

In order to replace conventional approximated motions with spline blocks, the program must be modified as follows:

- Replace LIN - LIN with SLIN - SPL - SLIN.
- Replace LIN - CIRC with SLIN - SPL - SCIRC.

Recommendation: Allow the SPL to project a certain way into the original circle. The SCIRC thus starts later than the original CIRC.

In approximated motions, the corner point is programmed. In the spline block, the points at the start and end of the approximation are programmed instead.



The approximate positioning arc of the approximated motions varies according to the override. For this reason, when reproducing an approximated motion, it must be ensured that it is executed with the desired override.

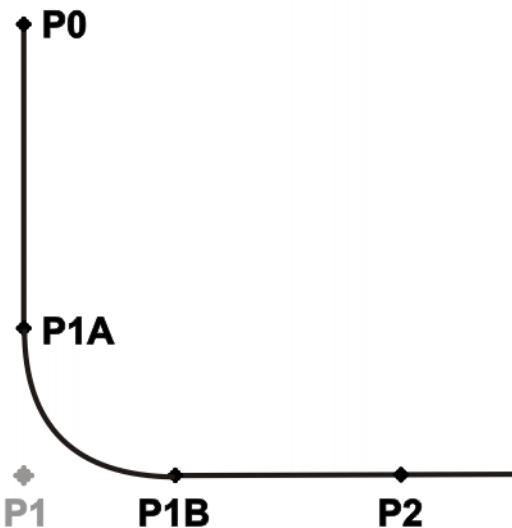
The following approximated motion is to be reproduced:

```
LIN P1 C_DIS  
LIN P2
```

Spline motion:

```
SPLINE  
SLIN P1A  
SPL P1B  
SLIN P2  
ENDSPLINE
```

P1A = start of approximation, P1B = end of approximation



**Fig. 8-24: Approximated motion - spline motion**

Ways of determining P1A and P1B:

- Execute the approximated path and save the positions at the desired point by means of Trigger.
- Calculate the points in the program with KRL.
- The start of the approximation can be determined from the approximate positioning criterion. Example: If C\_DIS is specified as the approximate positioning criterion, the distance from the start of the approximation to the corner point corresponds to the value of \$APO.CDIS.

The end of the approximation is dependent on the programmed velocity.

The SPL path does not correspond exactly to the approximate positioning arc, even if P1A and P1B are exactly at the start/end of the approximation. In order to recreate the exact approximate positioning arc, additional points must be inserted into the spline. Generally, one point is sufficient.

### Example

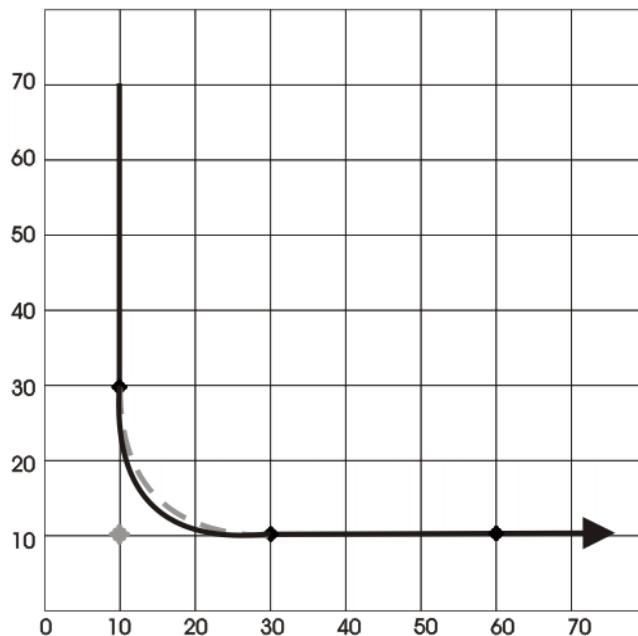
The following approximated motion is to be reproduced:

```
$APO.CDIS=20
$VEL.CP=0.5
LIN {Z 10} C_DIS
LIN {Y 60}
```

Spline motion:

```
SPLINE WITH $VEL.CP=0.5
SLIN {Z 30}
SPL {Y 30, Z 10}
SLIN {Y 60}
ENDSPLINE
```

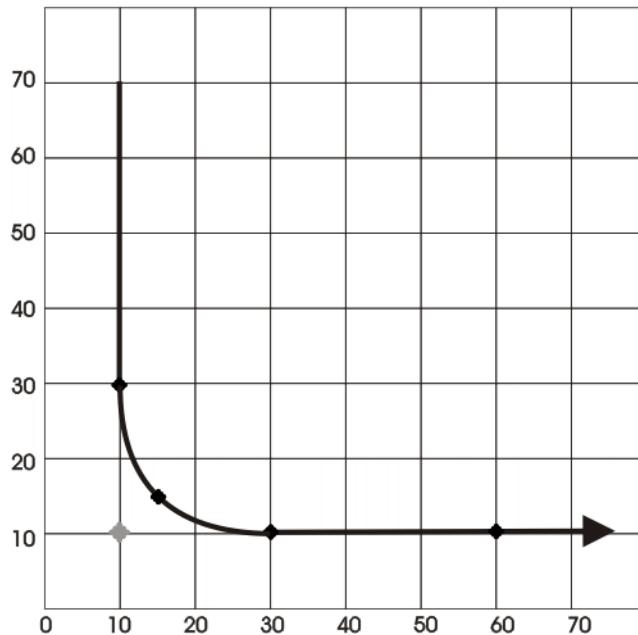
The start of the approximate positioning arc has been calculated from the approximate positioning criterion.



**Fig. 8-25: Example: Approximated motion - spline motion 1**

The SPL path does not yet correspond exactly to the approximate positioning arc. For this reason, an additional SPL segment is inserted into the spline.

```
SPLINE WITH $VEL.CP=0.5
SLIN {Z 30}
SPL {Y 15, Z 15}
SPL {Y 30, Z 10}
SLIN {Y 60}
ENDSPLINE
```

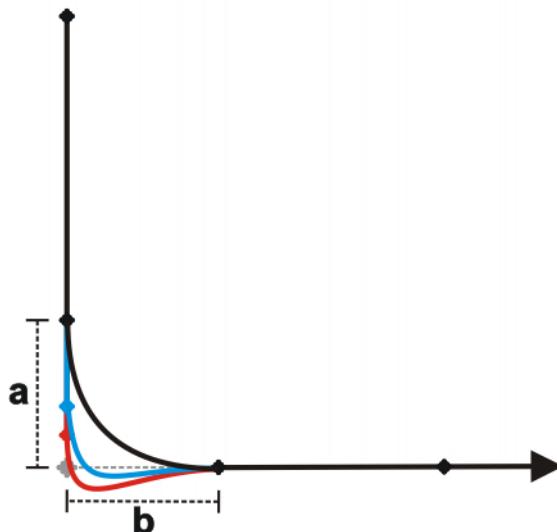


**Fig. 8-26: Example: Approximated motion - spline motion 2**

With the additional point, the path now corresponds to the approximate positioning arc.

### 8.7.5.1 SLIN-SPL-SLIN transition

In the case of a SLIN-SPL-SLIN segment sequence, it is usually desirable for the SPL segment to be located within the smaller angle between the two straight lines. Depending on the start and end point of the SPL segment, the path may also move outside this area.



**Fig. 8-27: SLIN-SPL-SLIN**

The path remains inside if the following conditions are met:

- The extensions of the two SLIN segments intersect.
  - $2/3 \leq a/b \leq 3/2$
- a = distance from start point of the SPL segment to intersection of the SLIN segments  
 b = distance from intersection of the SLIN segments to end point of the SPL segment

## 8.8 Orientation control SPLINE

### Description

The orientation of the TCP can be different at the start point and end point of a motion. When a CP motion is programmed, it is necessary to select how to deal with the different orientations.

The orientation control for SLIN and SCIRC motions is defined as follows:

- If programming with KRL syntax: by means of the system variable \$ORI\_TYPE
- If programming with inline forms: in the option window **Motion parameters**

Orientation control	Description
<ul style="list-style-type: none"> <li>■ Option window: <b>Constant orientation</b></li> <li>■ <math>\\$ORI\_TYPE = \#CONSTANT</math></li> </ul>	<p>The orientation of the TCP remains constant during the motion.</p> <p>The orientation of the start point is retained. The programmed orientation of the end point is not taken into consideration.</p>
<ul style="list-style-type: none"> <li>■ Option window: <b>Standard</b></li> <li>■ <math>\\$ORI\_TYPE = \#VAR</math></li> </ul>	<p>The orientation of the TCP changes continuously during the motion. At the end point, the TCP has the programmed orientation.</p>
<ul style="list-style-type: none"> <li>■ Option window: <b>Ignore orientation</b></li> <li>■ <math>\\$ORI\_TYPE = \#IGNORE</math></li> </ul>	<p>This option is only available for spline segments (not for the spline block or for individual spline motions).</p> <p>It is used if no specific orientation is required at a point.</p> <p>(&gt;&gt;&gt; "#IGNORE" Page 240)</p>

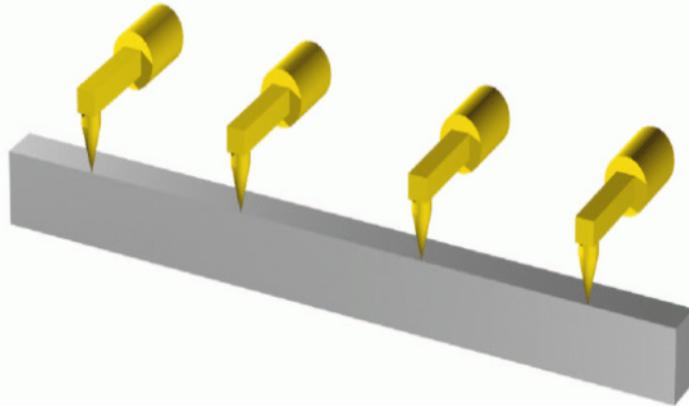


Fig. 8-28: Orientation control - Constant

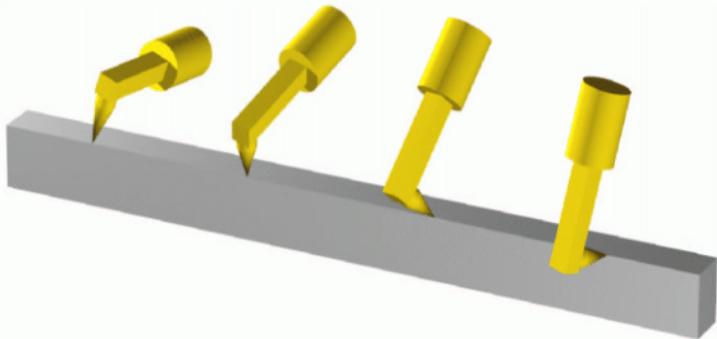


Fig. 8-29: Standard

#### #IGNORE

$\$ORI\_TYPE = \#IGNORE$  is used if no specific orientation is required at a point. If this option is selected, the taught or programmed orientation of the point is ignored. Instead, the robot controller calculates the optimal orientation for this point on the basis of the orientations of the surrounding points.

Example:

```

SPLINE
  SPL XP1
  SPL XP2
  SPL XP3 WITH $ORI_TYPE=#IGNORE
  SPL XP4 WITH $ORI_TYPE=#IGNORE
  SPL XP5
  SPL XP6
ENDSPLINE

```

The taught or programmed orientation of XP3 and XP4 is ignored.

Characteristics of \$ORI\_TYPE = #IGNORE:

- In the program run modes MSTEP and ISTEP, the robot stops with the orientations calculated by the robot controller.
- In the case of a block selection to a point with #IGNORE, the robot adopts the orientation calculated by the robot controller.

\$ORI\_TYPE = #IGNORE is not allowed for the following segments:

- The first segment in a spline block
- The last segment in a spline block
- SCIRC segments with \$CIRC\_TYPE=#PATH
- Segments followed by a SCIRC segment with \$CIRC\_TYPE=#PATH
- Segments followed by a segment with \$ORI\_TYPE=#CONSTANT
- In the case of successive segments with identical Cartesian end points, #IGNORE is not allowed for the first and last segments.

## SCIRC

It is also possible to define for SCIRC motions whether the orientation control is to be space-related or path-related.

(>>> 8.8.1 "SCIRC: reference system for the orientation control" Page 241)

It is also possible to define for SCIRC motions whether, and to what extent, the orientation of the auxiliary point is to be taken into consideration. The orientation behavior at the end point can also be defined.

(>>> 8.8.2 "SCIRC: orientation behavior" Page 242)

### 8.8.1 SCIRC: reference system for the orientation control

It is possible to define for SCIRC motions whether the orientation control is to be space-related or path-related. This can be defined as follows:

- If programming with inline forms: in the option window **Motion parameter**
- If programming with KRL syntax: by means of the system variable \$CIRC\_TYPE

Orientation control	Description
<ul style="list-style-type: none"> <li>■ Option window: <b>Base-related</b></li> <li>■ \$CIRC_TYPE = #BASE</li> </ul>	Base-related orientation control during the circular motion
<ul style="list-style-type: none"> <li>■ Option window: <b>Path-related</b></li> <li>■ \$CIRC_TYPE = #PATH</li> </ul>	Path-related orientation control during the circular motion

\$CIRC\_TYPE = #PATH is not allowed for the following motions:

- SCIRC segments for which \$ORI\_TYPE = #IGNORE
- SCIRC motions preceded by a spline segment for which \$ORI\_TYPE = #IGNORE

(>>> 8.6.1 "Combinations of \$ORI\_TYPE and \$CIRC\_TYPE" Page 227)

## 8.8.2 SCIRC: orientation behavior

<b>Description</b>	<p>During SCIRC motions, the robot controller can take the programmed orientation of the auxiliary point into consideration. \$CIRC_MODE can be used to define whether and to what extent it is taken into consideration.</p> <p>In the case of SCIRC statements with circular angles, \$CIRC_MODE can also be used to define whether the end point is to have the programmed orientation or whether the orientation is to be scaled according to the circular angle.</p> <p>\$CIRC_MODE can only be written to by means of a SCIRC statement. \$CIRC_MODE cannot be read.</p>						
<b>Syntax</b>	<p>For auxiliary points:</p> <pre>\$CIRC_MODE.AUX_PT.ORI = BehaviorAUX</pre> <p>For end points:</p> <pre>\$CIRC_MODE.TARGET_PT.ORI = BehaviorEND</pre>						
<b>Explanation of the syntax</b>	<table border="1"> <thead> <tr> <th>Element</th><th>Description</th></tr> </thead> <tbody> <tr> <td><i>BehaviorAUX</i></td><td> <p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation is accepted in the auxiliary point.</li> <li>■ <b>#IGNORE</b>: The transition from the start orientation to the end orientation is carried out over the shortest possible distance. The programmed orientation of the auxiliary point is disregarded.</li> <li>■ <b>#CONSIDER</b> (default): The transition from the start orientation to the end orientation passes through the programmed orientation of the auxiliary point, i.e. the orientation of the auxiliary point is accepted at some point during the transition, but not necessarily at the auxiliary point.</li> </ul> </td></tr> <tr> <td><i>BehaviorEND</i></td><td> <p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation of the end point is accepted at the actual end point. (Only possibility for SCIRC without specification of circular angle. If #EXTRAPOLATE is set, #INTERPOLATE is nonetheless executed.)</li> <li>■ <b>#EXTRAPOLATE</b>: The programmed orientation is accepted at the programmed end point. The orientation at the actual end point is scaled according to the circular angle. (Default for SCIRC with specification of circular angle.)</li> </ul> </td></tr> </tbody> </table>	Element	Description	<i>BehaviorAUX</i>	<p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation is accepted in the auxiliary point.</li> <li>■ <b>#IGNORE</b>: The transition from the start orientation to the end orientation is carried out over the shortest possible distance. The programmed orientation of the auxiliary point is disregarded.</li> <li>■ <b>#CONSIDER</b> (default): The transition from the start orientation to the end orientation passes through the programmed orientation of the auxiliary point, i.e. the orientation of the auxiliary point is accepted at some point during the transition, but not necessarily at the auxiliary point.</li> </ul>	<i>BehaviorEND</i>	<p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation of the end point is accepted at the actual end point. (Only possibility for SCIRC without specification of circular angle. If #EXTRAPOLATE is set, #INTERPOLATE is nonetheless executed.)</li> <li>■ <b>#EXTRAPOLATE</b>: The programmed orientation is accepted at the programmed end point. The orientation at the actual end point is scaled according to the circular angle. (Default for SCIRC with specification of circular angle.)</li> </ul>
Element	Description						
<i>BehaviorAUX</i>	<p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation is accepted in the auxiliary point.</li> <li>■ <b>#IGNORE</b>: The transition from the start orientation to the end orientation is carried out over the shortest possible distance. The programmed orientation of the auxiliary point is disregarded.</li> <li>■ <b>#CONSIDER</b> (default): The transition from the start orientation to the end orientation passes through the programmed orientation of the auxiliary point, i.e. the orientation of the auxiliary point is accepted at some point during the transition, but not necessarily at the auxiliary point.</li> </ul>						
<i>BehaviorEND</i>	<p>Data type: ENUM</p> <ul style="list-style-type: none"> <li>■ <b>#INTERPOLATE</b>: The programmed orientation of the end point is accepted at the actual end point. (Only possibility for SCIRC without specification of circular angle. If #EXTRAPOLATE is set, #INTERPOLATE is nonetheless executed.)</li> <li>■ <b>#EXTRAPOLATE</b>: The programmed orientation is accepted at the programmed end point. The orientation at the actual end point is scaled according to the circular angle. (Default for SCIRC with specification of circular angle.)</li> </ul>						
<b>Restrictions</b>	<ul style="list-style-type: none"> <li>■ If \$ORI_TYPE = #IGNORE for a SCIRC segment, \$CIRC_MODE is not evaluated.</li> <li>■ If a SCIRC segment is preceded by a SCIRC or SLIN segment with \$ORI_TYPE = #IGNORE, #CONSIDER cannot be used in this SCIRC segment.</li> </ul> <p>For SCIRC with circular angle:</p> <ul style="list-style-type: none"> <li>■ #INTERPOLATE must not be set for the auxiliary point.</li> <li>■ If \$ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.</li> <li>■ If it is preceded by a spline segment with \$ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.</li> </ul>						

**Example:**  
**Auxiliary point**

The TCP executes an arc with a Cartesian angle of 192°.

- The orientation at the start point is 0°.
- The orientation at the auxiliary point is 98°.
- The orientation at the end point is 197°.

The re-orientation is thus 197° if the auxiliary point is taken into consideration.

If the orientation at the auxiliary point is ignored, the end orientation can also be achieved by means of a re-orientation of  $360^\circ - 197^\circ = 163^\circ$ .

■ **#INTERPOLATE:**

The programmed orientation of 98° is accepted in the auxiliary point. The re-orientation is thus 197°.

■ **#IGNORE:**

The programmed orientation of the auxiliary point is disregarded. The shorter re-orientation through 163° is used.

■ **#CONSIDER:**

The distance traveled includes the orientation of the auxiliary point, in this case the re-orientation through 197°, i.e. the 98° are accepted at some point during the transition, but not necessarily at the auxiliary point.

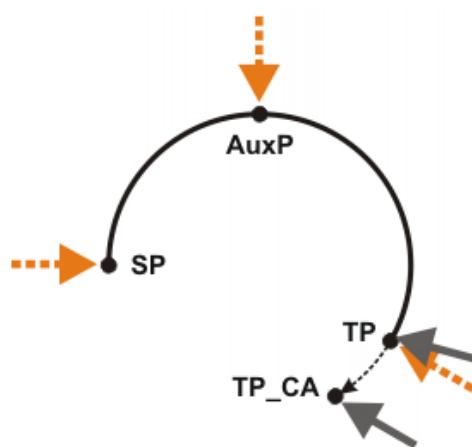
**Example:**  
**End point**

The example schematically illustrates the behavior of #INTERPOLATE and #EXTRAPOLATE.

- The pale, dotted arrows show the programmed orientation.
- The dark arrows show the actual orientation where this differs from the programmed orientation.

**#INTERPOLATE:**

At **TP**, which is situated before **TP\_CA**, the programmed orientation has not yet been reached. The programmed orientation is accepted at **TP\_CA**.



**Fig. 8-30: #INTERPOLATE**

<b>SP</b>	Start point
<b>AuxP</b>	Auxiliary point
<b>TP</b>	Programmed end point
<b>TP_CA</b>	Actual end point. Determined by the circular angle.

**#EXTRAPOLATE:**

The programmed orientation is accepted at **TP**. For **TP\_CA**, this orientation is scaled in accordance with the circular angle.

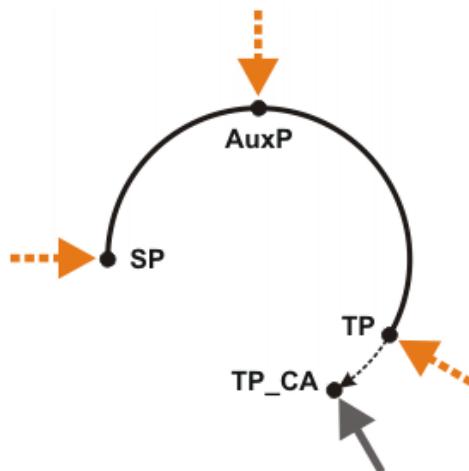


Fig. 8-31: #EXTRAPOLATE

## 8.9 Status and Turn

### Overview

The position (X, Y, Z) and orientation (A, B, C) values of the TCP are not sufficient to define the robot position unambiguously, as different axis positions are possible for the same TCP. Status and Turn serve to define an unambiguous position that can be achieved with different axis positions.

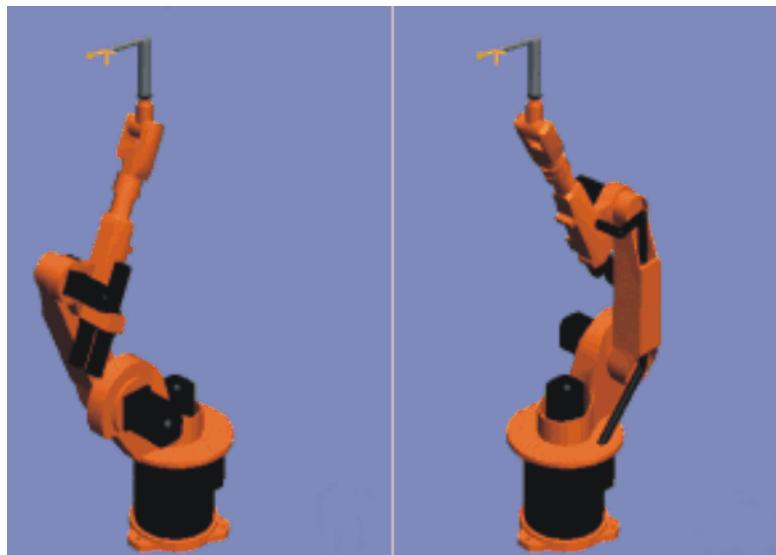


Fig. 8-32: Example: Same TCP position, different axis position

Status (S) and Turn (T) are integral parts of the data types POS and E6POS:

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

### KRL program

The robot controller only takes the programmed Status and Turn values into consideration for PTP motions. They are ignored for CP motions.

The first motion instruction in a KRL program must therefore be one of the following instructions so that an unambiguous starting position is defined for the robot:

- A complete PTP instruction of type POS or E6POS
- Or a complete PTP instruction of type AXIS or E6AXIS

“Complete” means that all components of the end point must be specified. The default HOME position is always a complete PTP instruction.

Status and Turn can be omitted in the subsequent instructions:

- The robot controller retains the previous Status value.
- The Turn value is determined by the path in CP motions. In the case of PTP motions, the robot controller selects the Turn value that results in the shortest possible path.

### 8.9.1 Status

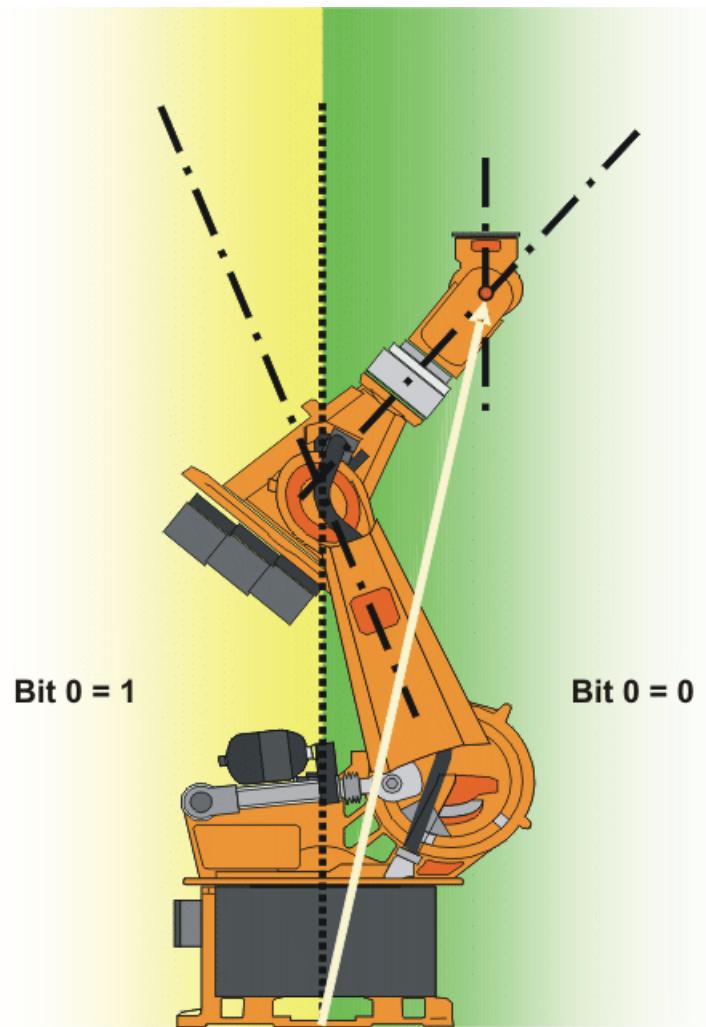
The Status specification prevents ambiguous axis positions.

#### Bit 0

Bit 0 specifies the position of the intersection of the wrist axes (A4, A5, A6).

Position	Value
Overhead area  If the x-value of the intersection of the wrist axes, relative to the A1 coordinate system, is negative, the robot is in the overhead area.	Bit 0 = 1
Basic area  If the x-value of the intersection of the wrist axes, relative to the A1 coordinate system, is positive, the robot is in the basic area.	Bit 0 = 0

The A1 coordinate system is identical to the \$ROBROOT coordinate system if axis 1 is at 0°. For values not equal to 0°, it moves with axis 1.



**Fig. 8-33: Example: The intersection of the wrist axes (red dot) is in the basic area.**

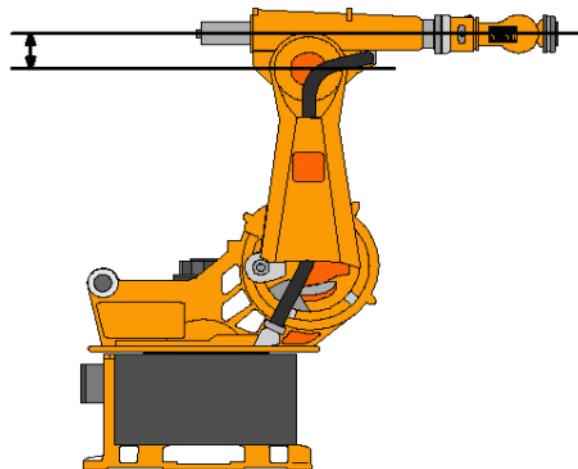
#### Bit 1

Bit 1 specifies the position of axis 3. The angle at which the value of bit 1 changes depends on the robot type.

For robots whose axes 3 and 4 intersect, the following applies:

Position	Value
A3 $\geq 0^\circ$	Bit 1 = 1
A3 < 0°	Bit 1 = 0

For robots with an offset between axis 3 and axis 4, the angle at which the value of bit 1 changes depends on the size of this offset.



**Fig. 8-34: Offset between A3 and A4 – example: KR 30**

#### Bit 2

Bit 2 specifies the position of axis 5.

Position	Value
A5 > 0	Bit 2 = 1
A5 ≤ 0	Bit 2 = 0

#### Bit 3

Bit 3 is not used and is always 0.

#### Bit 4

Bit 4 specifies whether or not the point was taught using an absolutely accurate robot.

Depending on the value of the bit, the point can be executed by both absolutely accurate robots and non-absolutely-accurate robots. Bit 4 is for information purposes only and has no influence on how the robot calculates the point. This means, therefore, that when a robot is programmed offline, bit 4 can be ignored.

Description	Value
The point was not taught with an absolutely accurate robot.	Bit 4 = 0
The point was taught with an absolutely accurate robot.	Bit 4 = 1

### 8.9.2 Turn

#### Description

The Turn specification makes it possible to move axes through angles greater than  $+180^\circ$  or less than  $-180^\circ$  without the need for special motion strategies (e.g. auxiliary points). With rotational axes, the individual bits determine the sign before the axis value in the following way:

Bit = 0: angle  $\geq 0^\circ$

Bit = 1: angle  $< 0^\circ$

Value	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	$A6 \geq 0^\circ$	$A5 \geq 0^\circ$	$A4 \geq 0^\circ$	$A3 \geq 0^\circ$	$A2 \geq 0^\circ$	$A1 \geq 0^\circ$
1	$A6 < 0^\circ$	$A5 < 0^\circ$	$A4 < 0^\circ$	$A3 < 0^\circ$	$A2 < 0^\circ$	$A1 < 0^\circ$

#### Example

```
DECL POS XP1 = {X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 19}
```

T 19 corresponds to T 'B010011'. This means:

Axis	Angle
A1	negative
A2	negative
A3	positive
A4	positive
A5	negative
A6	positive

## 8.10 Singularities

KUKA robots with 6 degrees of freedom have 3 different singularity positions.

- Overhead singularity
- Extended position singularity
- Wrist axis singularity

A singularity position is characterized by the fact that unambiguous reverse transformation (conversion of Cartesian coordinates to axis-specific values) is not possible, even though Status and Turn are specified. In this case, or if very slight Cartesian changes cause very large changes to the axis angles, one speaks of singularity positions.

### Overhead

In the overhead singularity, the wrist root point (intersection of axes A4, A5 and A6) is located vertically above axis 1.

The position of axis A1 cannot be determined unambiguously by means of reverse transformation and can thus take any value.

If the end point of a PTP motion is situated in this overhead singularity position, the robot controller may react as follows by means of the system variable \$SINGUL\_POS[1]:

- **0:** The angle for axis A1 is defined as 0 degrees (default setting).
- **1:** The angle for axis A1 remains the same from the start point to the end point.

### Extended position

In the extended position singularity, the wrist root point (intersection of axes A4, A5 and A6) is located in the extension of axes A2 and A3 of the robot.

The robot is at the limit of its work envelope.

Although reverse transformation does provide unambiguous axis angles, low Cartesian velocities result in high axis velocities for axes A2 and A3.

If the end point of a PTP motion is situated in this extended position singularity, the robot controller may react as follows by means of the system variable \$SINGUL\_POS[2]:

- **0:** The angle for axis A2 is defined as 0 degrees (default setting).
- **1:** The angle for axis A2 remains the same from the start point to the end point.

### Wrist axes

In the wrist axis singularity position, the axes A4 and A6 are parallel to one another and axis A5 is within the range  $\pm 0.01812^\circ$ .

The position of the two axes cannot be determined unambiguously by reverse transformation. There is an infinite number of possible axis positions for axes A4 and A6 with identical axis angle sums.

If the end point of a PTP motion is situated in this wrist axis singularity, the robot controller may react as follows by means of the system variable \$SINGUL\_POS[3]:

- **0:** The angle for axis A4 is defined as 0 degrees (default setting).
- **1:** The angle for axis A4 remains the same from the start point to the end point.



In the case of SCARA robots, only the extended position singularity can arise. In this case, the robot starts to move extremely fast.



## 9 Programming for user group "User" (inline forms)

Inline forms are available in the KSS for frequently used instructions. They simplify programming.



Instructions can also be programmed without inline forms. Information is contained in the description of the KRL syntax.

(>>> 10 "Programming for user group "Expert" (KRL syntax)"

Page 287)

### **NOTICE**

In the case of programs with the following axis motions or positions, the film of lubricant on the gear units of the axes may break down:

- Motions <3°
- Oscillating motions
- Areas of gear units permanently facing upwards

It must be ensured that the gear units have a sufficient supply of oil. For this, in the case of oscillating motions or short motions (<3°), programming must be carried out in such a way that the affected axes regularly move more than 40° (e.g. once per cycle).

In the case of areas of gear units permanently facing upwards, sufficient oil supply must be achieved by programming re-orientations of the in-line wrist. In this way, the oil can reach all areas of the gear units by means of gravity. Required frequency of re-orientations:

- With low loads (gear unit temperature <+35 °C): daily
- With medium loads (gear unit temperature +35 °C to 55 °C): hourly
- With heavy loads (gear unit temperature >+55 °C): every 10 minutes

Failure to observe this precaution may result in damage to the gear units.

### 9.1 Names in inline forms

Names for data sets can be entered in inline forms. These include, for example, point names, names for motion data sets, etc.

The following restrictions apply to names:

- Maximum length 23 characters
- No special characters are permissible, with the exception of \$.
- The first character must not be a number.

The restrictions do not apply to output names.

Other restrictions may apply in the case of inline forms in technology packages.

### 9.2 Programming PTP, LIN and CIRC motions

#### 9.2.1 Programming a PTP motion

### **NOTICE**

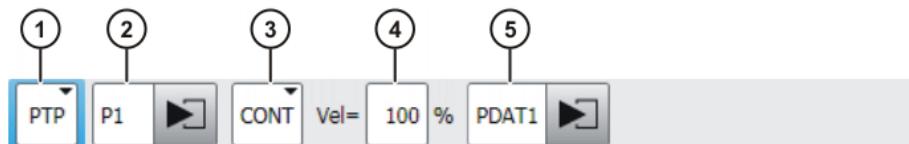
When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

#### Precondition

- A program is selected.
- Operating mode T1

**Procedure**

1. Move the TCP to the position that is to be taught as the end point.
2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands > Motion > PTP**.
4. Set the parameters in the inline form.  
(>>> 9.2.2 "Inline form "PTP"" Page 252)
5. Save instruction with **Cmd Ok**.

**9.2.2 Inline form "PTP"****Fig. 9-1: Inline form for PTP motions**

Item	Description
1	Motion type <b>PTP</b>
2	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 251) Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window: Frames" Page 255)
3	■ <b>CONT</b> : end point is approximated. ■ <b>[Empty box]</b> : the motion stops exactly at the end point.
4	Velocity ■ <b>1 ... 100%</b>
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.8 "Option window: Motion parameters (PTP)" Page 255)

**9.2.3 Programming a LIN motion****NOTICE**

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

**Precondition**

- A program is selected.
- Operating mode T1

**Procedure**

1. Move the TCP to the position that is to be taught as the end point.
2. Position the cursor in the line after which the motion instruction is to be inserted.
3. Select the menu sequence **Commands > Motion > LIN**.

4. Set the parameters in the inline form.  
(>>> 9.2.4 "Inline form "LIN"" Page 253)
5. Save instruction with **Cmd Ok**.

#### 9.2.4 Inline form "LIN"

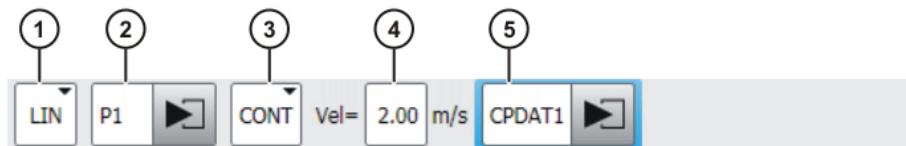


Fig. 9-2: Inline form for LIN motions

Item	Description
1	Motion type <b>LIN</b>
2	Name of the end point The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 251) Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window: Frames" Page 255)
3	<ul style="list-style-type: none"> <li>■ <b>CONT</b>: end point is approximated.</li> <li>■ <b>[Empty box]</b>: the motion stops exactly at the end point.</li> </ul>
4	<ul style="list-style-type: none"> <li>■ <b>Velocity</b></li> <li>■ <b>0.001 ... 2 m/s</b></li> </ul>
5	Name for the motion data set The system automatically generates a name. The name can be overwritten. Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.9 "Option window: Motion parameters (LIN, CIRC)" Page 256)

#### 9.2.5 Programming a CIRC motion

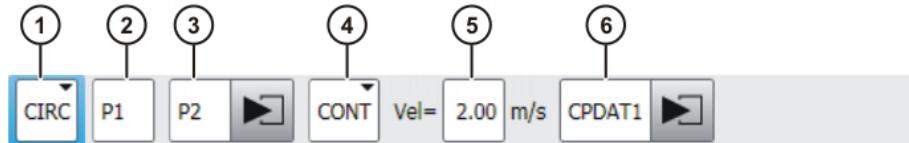
**NOTICE**

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

- Precondition**
- A program is selected.
  - Operating mode T1
- Procedure**
1. Move the TCP to the position that is to be taught as the auxiliary point.
  2. Position the cursor in the line after which the motion instruction is to be inserted.
  3. Select the menu sequence **Commands > Motion > CIRC**.
  4. Set the parameters in the inline form.  
(>>> 9.2.6 "Inline form "CIRC"" Page 254)
  5. Press **Teach Aux**.

6. Move the TCP to the position that is to be taught as the end point.
7. Save instruction with **Cmd Ok**.

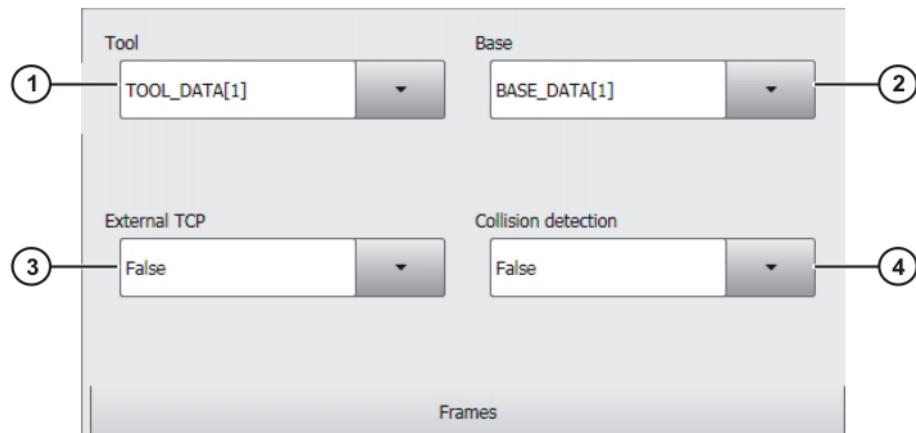
### 9.2.6 Inline form "CIRC"



**Fig. 9-3: Inline form for CIRC motions**

Item	Description
1	Motion type <b>CIRC</b>
2	Name of the auxiliary point The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 251)
3	Name of the end point The system automatically generates a name. The name can be overwritten. Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window: Frames" Page 255)
4	<ul style="list-style-type: none"> <li>■ <b>CONT</b>: end point is approximated.</li> <li>■ <b>[Empty box]</b>: the motion stops exactly at the end point.</li> </ul>
5	Velocity <ul style="list-style-type: none"> <li>■ <b>0.001 ... 2 m/s</b></li> </ul>
6	Name for the motion data set The system automatically generates a name. The name can be overwritten. Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.9 "Option window: Motion parameters (LIN, CIRC)" Page 256)

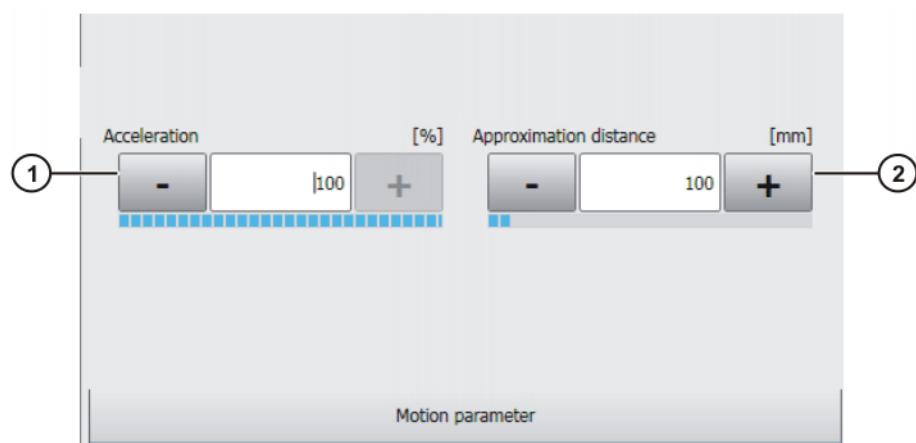
### 9.2.7 Option window: Frames



**Fig. 9-4: Option window: Frames**

Item	Description
1	Tool selection. If <b>True</b> in the box <b>External TCP</b> : workpiece selection. Range of values: [1] ... [16]
2	Base selection. If <b>True</b> in the box <b>External TCP</b> : fixed tool selection. Range of values: [1] ... [32]
3	Interpolation mode <ul style="list-style-type: none"> <li>■ <b>False:</b> The tool is mounted on the mounting flange.</li> <li>■ <b>True:</b> The tool is a fixed tool.</li> </ul>
4	<ul style="list-style-type: none"> <li>■ <b>True:</b> For this motion, the robot controller calculates the axis torques. These are required for collision detection.</li> <li>■ <b>False:</b> For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.</li> </ul>

### 9.2.8 Option window: Motion parameters (PTP)



**Fig. 9-5: Option window: Motion parameters (PTP)**

Item	Description
1	<p>Acceleration</p> <p>Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode.</p> <ul style="list-style-type: none"> <li>■ 1 ... 100 %</li> </ul>
2	<p>This box is only displayed if <b>CONT</b> was selected in the inline form.</p> <p>Furthest distance before the end point at which approximate positioning can begin.</p> <p>Maximum distance 100%: half the distance between the start point and the end point relative to the contour of the PTP motion without approximate positioning</p> <ul style="list-style-type: none"> <li>■ 1 ... 100 %</li> </ul>

### 9.2.9 Option window: Motion parameters (LIN, CIRC)

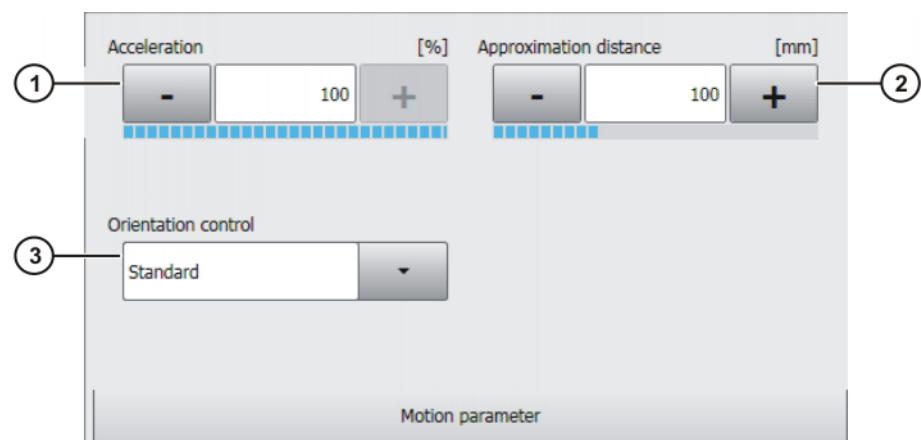


Fig. 9-6: Option window: Motion parameters (LIN, CIRC)

Item	Description
1	<p>Acceleration</p> <p>Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode.</p>
2	<p>Furthest distance before the end point at which approximate positioning can begin</p> <p>The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used.</p> <p>This box is only displayed if <b>CONT</b> was selected in the inline form.</p>
3	<p>Orientation control selection.</p> <ul style="list-style-type: none"> <li>■ Standard</li> <li>■ Wrist PTP</li> <li>■ Constant orientation control</li> </ul> <p>(&gt;&gt;&gt; 8.6 "Orientation control LIN, CIRC" Page 226)</p>

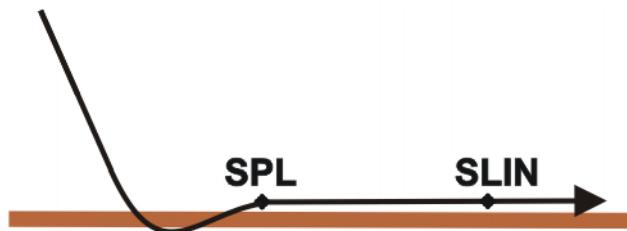
## 9.3 Spline motions

### 9.3.1 Programming tips for spline motions

- A spline block should cover only 1 process (e.g. 1 adhesive seam). More than one process in a spline block leads to a loss of structural clarity within the program and makes changes more difficult.
  - Use SLIN and SCIRC segments in cases where the workpiece necessitates straight lines and arcs. (Exception: use SPL segments for very short straight lines.) Otherwise, use SPL segments, particularly if the points are close together.
  - Procedure for defining the path:
    - a. First teach or calculate a few characteristic points. Example: points at which the curve changes direction.
    - b. Test the path. At points where the accuracy is still insufficient, add more SPL points.
  - Avoid successive SLIN and/or SCIRC segments, as this often reduces the velocity to 0.
- Program SPL segments between SLIN and SCIRC segments. The length of the SPL segments must be at least  $> 0.5$  mm. Depending on the actual path, significantly larger SPL segments may be required.
- Avoid successive points with identical Cartesian coordinates, as this reduces the velocity to 0.
  - The parameters (tool, base, velocity, etc.) assigned to the spline block have the same effect as assignments before the spline block. The assignment to the spline block has the advantage, however, that the correct parameters are read in the case of a block selection.
  - Use the option **Ignore Orientation** if no specific orientation is required at a point. The robot controller calculates the optimal orientation for this point on the basis of the orientations of the surrounding points. This way, even large changes in orientation between two points are optimally distributed over the points in between.
  - Jerk limitation can be programmed. The jerk is the change in acceleration.

Procedure:

- a. Use the default values initially.
  - b. If vibrations occur at tight corners: reduce values.  
If the velocity drops or the desired velocity cannot be reached: increase values or increase acceleration.
- If the robot executes points on a work surface, a collision with the work surface is possible when the first point is addressed.



**Fig. 9-7: Collision with work surface**

In order to avoid a collision, observe the recommendations for the SLIN-SPL-SLIN transition.

(>>> 8.7.5.1 "SLIN-SPL-SLIN transition" Page 239)

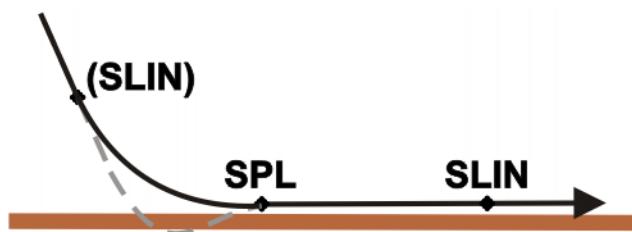


Fig. 9-8: Avoiding a collision with the work surface

### 9.3.2 Programming a SLIN motion (individual motion)

**NOTICE**

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

**Precondition**

- A program is selected.
- Operating mode T1

**Procedure**

1. Move the TCP to the end point.
2. Position the cursor in the line after which the motion is to be inserted.  
(But not within a spline block. This opens another inline form.)  
(>>> 9.3.4.6 "Inline form for spline segment" Page 266)
3. Select **Commands > Motion > SLIN**.
4. Set the parameters in the inline form.  
(>>> 9.3.2.1 "Inline form "SLIN"" Page 258)
5. Press **Cmd OK**.

#### 9.3.2.1 Inline form "SLIN"

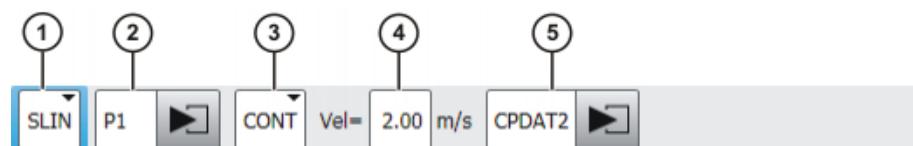
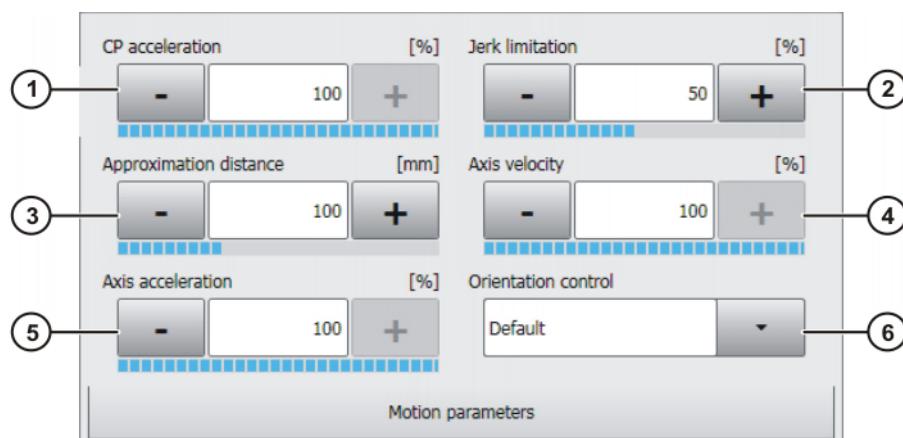


Fig. 9-9: Inline form "SLIN" (individual motion)

Item	Description
1	Motion type <b>SLIN</b>
2	Point name for end point. The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 251) Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window: Frames" Page 255)
3	<ul style="list-style-type: none"> <li>■ <b>CONT</b>: end point is approximated.</li> <li>■ <b>[Empty box]</b>: the motion stops exactly at the end point.</li> </ul>

Item	Description
4	Velocity ■ <b>0.001 ... 2 m/s</b>
5	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.3.2.2 "Option window "Motion parameters" (SLIN)" Page 259)

### 9.3.2.2 Option window "Motion parameters" (SLIN)



**Fig. 9-10: Option window "Motion parameters" (SLIN)**

Item	Description
1	Path acceleration. The value refers to the maximum value specified in the machine data. ■ <b>1 ... 100%</b>
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ <b>1 ... 100%</b>
3	This box is only displayed if <b>CONT</b> was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used.
4	Axis velocity. The value refers to the maximum value specified in the machine data. ■ <b>1 ... 100%</b>
5	Axis acceleration. The value refers to the maximum value specified in the machine data. ■ <b>1 ... 100%</b>
6	Orientation control selection.

### 9.3.3 Programming a SCIRC motion (individual motion)

**NOTICE** When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

#### Precondition

- A program is selected.
- Operating mode T1

#### Procedure

1. Move the TCP to the auxiliary point.
2. Position the cursor in the line after which the motion is to be inserted.  
(But not within a spline block. This opens another inline form.)  
(>>> 9.3.4.6 "Inline form for spline segment" Page 266)
3. Select the menu sequence **Commands > Motion > SCIRC**.
4. Set the parameters in the inline form.  
(>>> 9.3.3.1 "Inline form "SCIRC"" Page 260)
5. Press **Teach Aux**.
6. Move the TCP to the end point.
7. Press **Cmd OK**.

#### 9.3.3.1 Inline form "SCIRC"

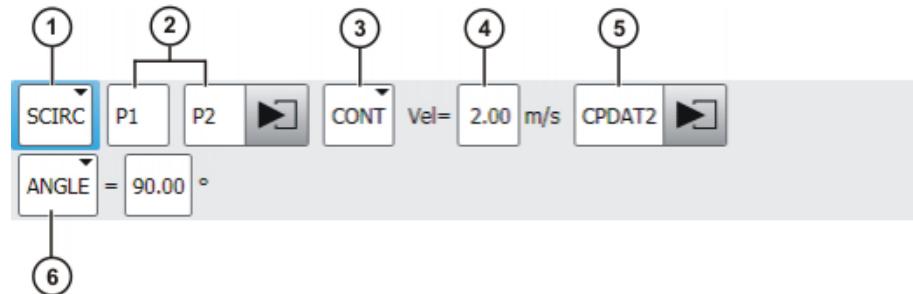


Fig. 9-11: Inline form "SCIRC" (individual motion)

Item	Description
1	Motion type <b>SCIRC</b>
2	Point names for auxiliary and end point. The system automatically assigns names. The names can be overwritten. (>>> 9.1 "Names in inline forms" Page 251) Touch the arrow to edit the point data. The corresponding option window is opened. (>>> 9.2.7 "Option window: Frames" Page 255)
3	<ul style="list-style-type: none"> <li>■ <b>CONT</b>: end point is approximated.</li> <li>■ <b>[Empty box]</b>: the motion stops exactly at the end point.</li> </ul>
4	Velocity <ul style="list-style-type: none"> <li>■ <b>0.001 ... 2 m/s</b></li> </ul>

Item	Description
5	<p>Name for the motion data set. The system automatically generates a name. The name can be overwritten.</p> <p>Touch the arrow to edit the point data. The corresponding option window is opened.</p> <p>(&gt;&gt;&gt; 9.3.3.2 "Option window "Motion parameters" (SCIRC)" Page 261)</p>
6	<p>Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point.</p> <ul style="list-style-type: none"> <li>■ Positive circular angle: the circular path is executed in the direction Start point → Auxiliary point → End point.</li> <li>■ Negative circular angle: the circular path is executed in the direction Start point → End point → Auxiliary point.</li> <li>■ <b>- 9,999° ... + 9,999°</b></li> </ul> <p>If a circular angle less than -400° or greater than +400° is entered, a request for confirmation is generated when the inline form is saved asking whether entry is to be confirmed or rejected.</p>

### 9.3.3.2 Option window "Motion parameters" (SCIRC)

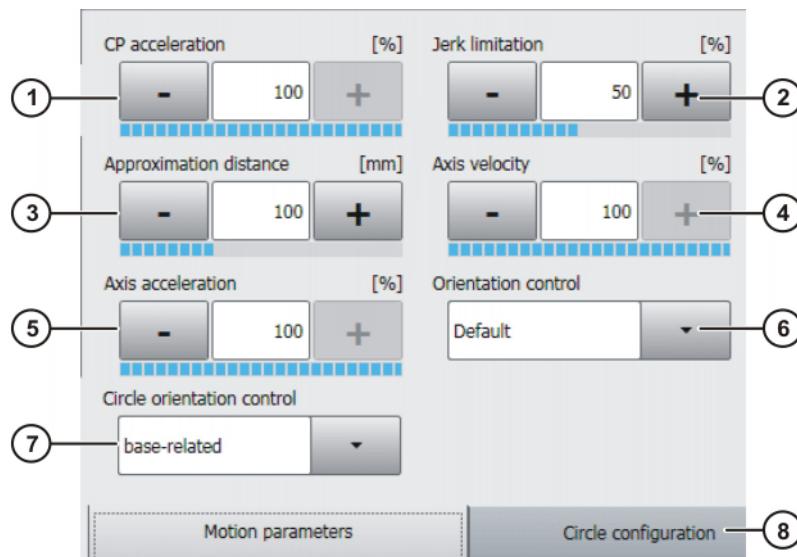


Fig. 9-12: Option window "Motion parameters" (SCIRC)

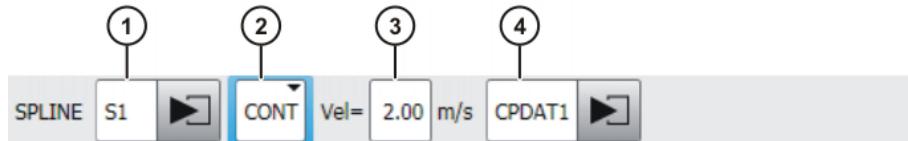
Item	Description
1	<p>Path acceleration. The value refers to the maximum value specified in the machine data.</p> <ul style="list-style-type: none"> <li>■ <b>1 ... 100%</b></li> </ul>
2	<p>Jerk limitation. The jerk is the change in acceleration.</p> <p>The value refers to the maximum value specified in the machine data.</p> <ul style="list-style-type: none"> <li>■ <b>1 ... 100%</b></li> </ul>

Item	Description
3	Furthest distance before the end point at which approximate positioning can begin.  The maximum permissible value is half the distance between the start point and the end point. If a higher value is entered, this is ignored and the maximum value is used.  This box is only displayed if <b>CONT</b> was selected in the inline form.
4	Axis velocity. The value refers to the maximum value specified in the machine data.  ■ <b>1 ... 100%</b>
5	Axis acceleration. The value refers to the maximum value specified in the machine data.  ■ <b>1 ... 100%</b>
6	Orientation control selection
7	Orientation control reference system selection.
8	Circular parameters are displayed on this tab. The parameters cannot be changed.

### 9.3.4 Programming a spline block

- Description** A spline block can be used to group together several SPL, SLIN and/or SCIRC segments to an overall motion. A spline block that contains no segments is not a motion statement.
- A spline block may contain the following:
- Spline segments (only limited by the memory capacity.)
  - PATH trigger
  - Comments and blank lines
  - Inline commands from technology packages that support the spline functionality
- A spline block must not include any other instructions, e.g. variable assignments or logic statements. A spline block does not trigger an advance run stop.
- Precondition**
- A program is selected.
  - Operating mode T1
- Procedure**
1. Position the cursor in the line after which the spline block is to be inserted.
  2. Select the menu sequence **Commands > Motion > SPLINE block**.
  3. Set the parameters in the inline form.  
(>>> 9.3.4.1 "Inline form for spline block" Page 263)
  4. Press **Cmd OK**.
  5. Press **Open/close fold**. Spline segments and other lines can now be inserted into the spline block.  
(>>> 9.3.4.4 "Programming an SPL or SLIN segment" Page 265)  
(>>> 9.3.4.5 "Programming an SCIRC segment" Page 265)  
(>>> 9.3.4.9 "Programming triggers in the spline block" Page 268)

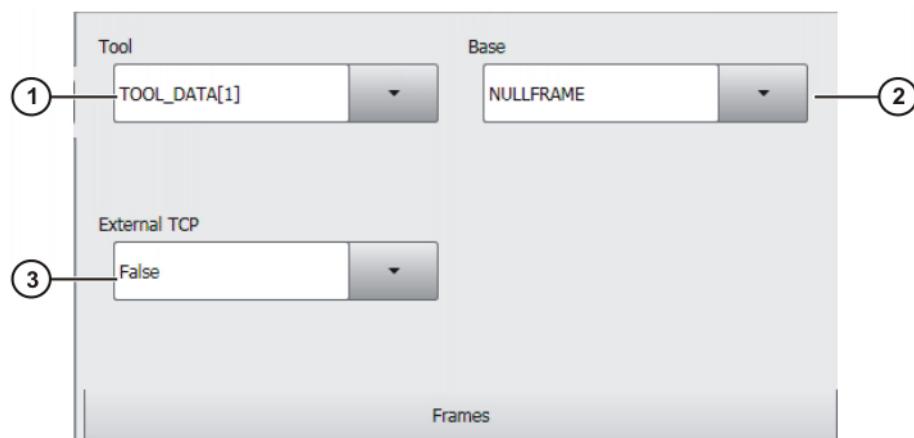
### 9.3.4.1 Inline form for spline block



**Fig. 9-13: Inline form for spline block**

Item	Description
1	Name of the spline block. The system automatically generates a name. The name can be overwritten. (>>> 9.1 "Names in inline forms" Page 251) Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 9.3.4.2 "Option window "Frames" (spline block)" Page 263)
2	■ <b>CONT</b> : end point is approximated. ■ <b>[Empty box]</b> : the motion stops exactly at the end point.
3	The velocity is valid by default for the entire spline block. It can also be defined separately for individual segments. ■ <b>0.001 ... 2 m/s</b>
4	Name for the motion data set. The system automatically generates a name. The name can be overwritten. Position the cursor in this box to edit the motion data. The corresponding option window is opened. (>>> 9.3.4.3 "Option window "Motion parameters" (spline block)" Page 264) The motion data are valid by default for the entire spline block. They can also be defined separately for individual segments.

### 9.3.4.2 Option window "Frames" (spline block)



**Fig. 9-14: Option window "Frames" (spline block)**

Item	Description
1	Tool selection. If <b>True</b> in the box <b>External TCP</b> : workpiece selection. ■ [1] ... [16]
2	Base selection. If <b>True</b> in the box <b>External TCP</b> : fixed tool selection. ■ [1] ... [32]
3	Interpolation mode ■ <b>False</b> : The tool is mounted on the mounting flange. ■ <b>True</b> : The tool is a fixed tool.

### 9.3.4.3 Option window “Motion parameters” (spline block)

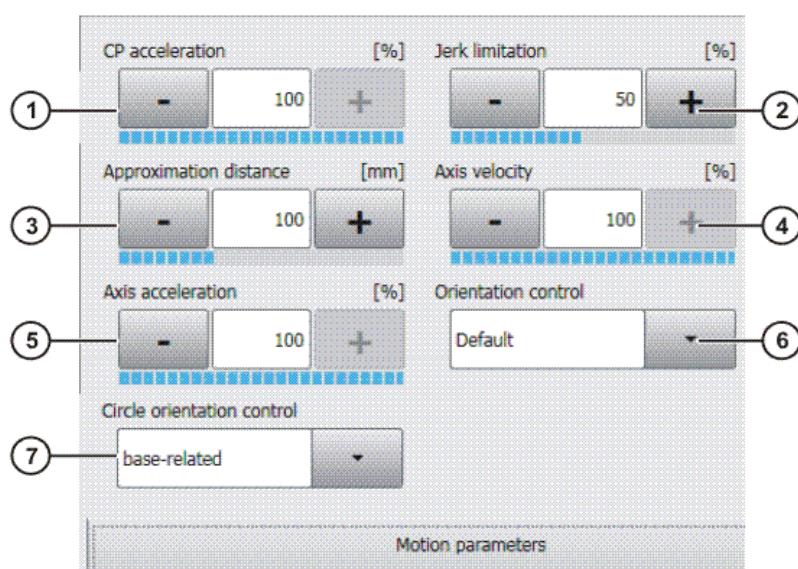


Fig. 9-15: Option window “Motion parameters” (spline block)

Item	Description
1	Path acceleration. The value refers to the maximum value specified in the machine data. ■ 1 ... 100%
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. ■ 1 ... 100%
3	This box is only displayed if <b>CONT</b> was selected in the inline form. Furthest distance before the end point at which approximate positioning can begin. The maximum distance is that of the last segment in the spline. If there is only one segment present, the maximum distance is half the segment length. If a higher value is entered, this is ignored and the maximum value is used.
4	Axis velocity. The value refers to the maximum value specified in the machine data. ■ 1 ... 100%

Item	Description
5	Axis acceleration. The value refers to the maximum value specified in the machine data. ■ 1 ... 100%
6	Orientation control selection.
7	Orientation control reference system selection. This parameter only affects SCIRC segments (if present) in the spline block.

#### 9.3.4.4 Programming an SPL or SLIN segment

**NOTICE**

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

**Precondition**

- A program is selected.
- Operating mode T1
- The spline block fold is open.

**Procedure**

1. Move the TCP to the end point.
2. Position the cursor in the line after which the segment is to be inserted in the spline block.
3. Select the menu sequence **Commands > Motion > SPL or SLIN**.
4. Set the parameters in the inline form.  
(>>> 9.3.4.6 "Inline form for spline segment" Page 266)
5. Press **Cmd OK**.

#### 9.3.4.5 Programming an SCIRC segment

**NOTICE**

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

**Precondition**

- A program is selected.
- Operating mode T1
- The spline block fold is open.

**Procedure**

1. Move the TCP to the auxiliary point.
2. Position the cursor in the line after which the segment is to be inserted in the spline block.
3. Select the menu sequence **Commands > Motion > SCIRC**.
4. Set the parameters in the inline form.  
(>>> 9.3.4.6 "Inline form for spline segment" Page 266)
5. Press **Teach Aux**.
6. Move the TCP to the end point.
7. Press **Cmd OK**.

### 9.3.4.6 Inline form for spline segment

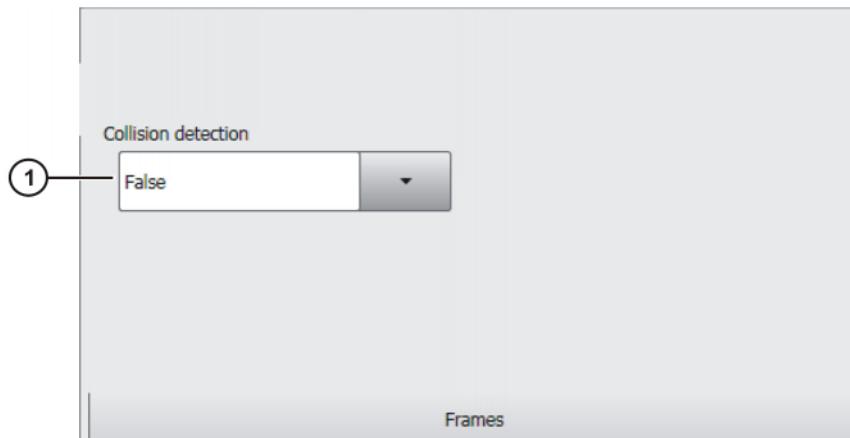


**Fig. 9-16: Inline form for spline segment**

The boxes in the inline form can be displayed or hidden one by one with **Toggle parameters**.

Item	Description
1	Motion type ■ <b>SPL</b> ■ <b>SLIN</b> ■ <b>SCIRC</b>
2	Point name for end point. Only for SCIRC: point names for auxiliary point and end point.  The system automatically generates a name. The name can be overwritten.  (>>> 9.1 "Names in inline forms" Page 251)  Touch the arrow to edit the point data. The corresponding option window is opened.  (>>> 9.3.4.7 "Option window "Frames" (spline segment)" Page 267)
3	Velocity  This only refers to the segment to which it belongs. It has no effect on subsequent segments.  ■ <b>0.001 ... 2 m/s</b>
4	Name for the motion data set. The system automatically generates a name. The name can be overwritten.  Touch the arrow to edit the point data. The corresponding option window is opened.  (>>> 9.3.4.8 "Option window "Motion parameters" (spline segment)" Page 267)  The motion data only refer to the segment to which they belong. They have no effect on subsequent segments.
5	Only available if the motion type <b>SCIRC</b> has been selected.  Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point.  ■ Positive circular angle: the circular path is executed in the direction Start point > Auxiliary point > End point. ■ Negative circular angle: the circular path is executed in the direction Start point > End point > Auxiliary point. ■ <b>- 9,999° ... + 9,999°</b>  If a circular angle less than -400° or greater than +400° is entered, a request for confirmation is generated when the inline form is saved asking whether entry is to be confirmed or rejected.

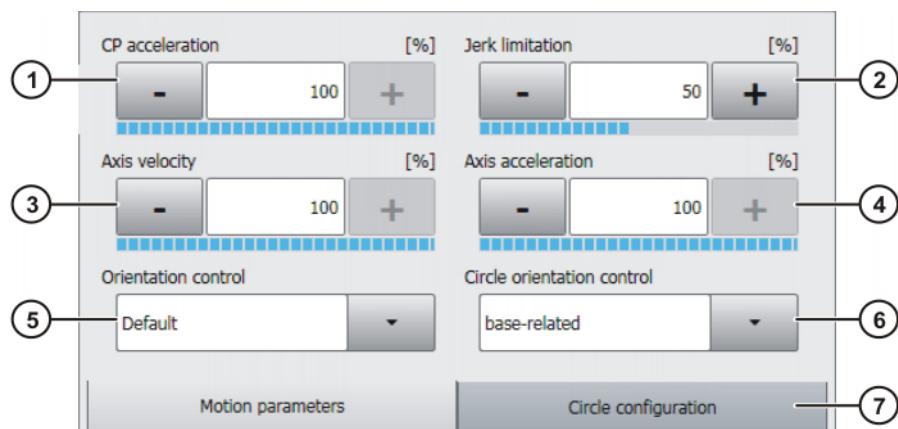
### 9.3.4.7 Option window "Frames" (spline segment)



**Fig. 9-17: Option window "Frames" (spline segment)**

Item	Description
1	<ul style="list-style-type: none"> <li>■ <b>True:</b> For this motion, the robot controller calculates the axis torques. These are required for collision detection.</li> <li>■ <b>False:</b> For this motion, the robot controller does not calculate the axis torques. Collision detection is thus not possible for this motion.</li> </ul>

### 9.3.4.8 Option window "Motion parameters" (spline segment)



**Fig. 9-18: Option window "Motion parameters" (spline segment)**

Item	Description
1	Path acceleration. The value refers to the maximum value specified in the machine data. <ul style="list-style-type: none"> <li>■ <b>1 ... 100 %</b></li> </ul>
2	Jerk limitation. The jerk is the change in acceleration. The value refers to the maximum value specified in the machine data. <ul style="list-style-type: none"> <li>■ <b>1 ... 100 %</b></li> </ul>
3	Axis velocity. The value refers to the maximum value specified in the machine data. <ul style="list-style-type: none"> <li>■ <b>1 ... 100 %</b></li> </ul>

Item	Description
4	Axis acceleration. The value refers to the maximum value specified in the machine data. ■ 1 ... 100 %
5	Orientation control selection
6	Only in the case of SCIRC segments: Orientation control reference system selection.
7	Only in the case of SCIRC segments: Circular parameters are displayed on this tab. The parameters cannot be changed.

### 9.3.4.9 Programming triggers in the spline block

#### Precondition

- A program is selected.
- Operating mode T1
- The spline block fold is open.

#### Procedure

1. Position the cursor in the line after which the trigger is to be inserted in the spline block.
2. Select the menu sequence **Commands > Logic > Spline trigger**.
3. The inline form **Set output** is displayed by default. A different inline form can be displayed by pressing the **Toggle type** button.
4. Set the parameters in the inline form.
5. Press **Cmd OK**.

#### Description

The specific inline form that is displayed depends on which type has been selected using **Toggle type**.

Inline form type	Description
<b>Set output</b>	The trigger sets an output. (>>> 9.3.4.10 "Inline form for spline trigger type "Set output"" Page 269)
<b>Set pulse output</b>	The trigger sets a pulse of a defined length. (>>> 9.3.4.11 "Inline form for spline trigger type "Set pulse output"" Page 270)
<b>Trigger assignment</b>	The trigger assigns a value to a variable. Only available in the user group "Expert". (>>> 9.3.4.12 "Inline form for spline trigger type "Trigger assignment"" Page 271)
<b>Trigger function call</b>	The trigger calls a subprogram. Only available in the user group "Expert". (>>> 9.3.4.13 "Inline form for spline trigger type "Trigger function call"" Page 271)



Further information on triggers, on offsetting the switching point and on the offset limits can be found here:  
(>>> 10.11 "Path-related switching actions (=Trigger)" Page 333)

### 9.3.4.10 Inline form for spline trigger type "Set output"

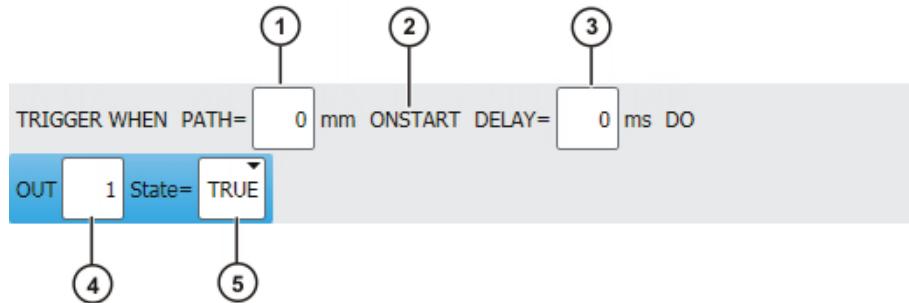


Fig. 9-19: Inline form for spline trigger type Set output

Item	Description
1	If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. If no shift in space is desired, enter the value 0. <ul style="list-style-type: none"> <li>Positive value: shifts the statement towards the end of the motion.</li> <li>Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle path</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
2	<b>Toggle OnStart</b> can be used to set or cancel the parameter <b>ONSTART</b> . <ul style="list-style-type: none"> <li>Without <b>ONSTART</b>: the PATH value refers to the end point.</li> <li>With <b>ONSTART</b>: the PATH value refers to the start point.</li> </ul>
3	If the statement is to be shifted in time (relative to the value in item 1), the desired duration must be specified here. If no shift in time is desired, enter the value 0. <ul style="list-style-type: none"> <li>Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms</li> <li>Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle Delay</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
4	Output number <ul style="list-style-type: none"> <li><b>1 ... 4096</b></li> </ul>
5	State to which the output is switched <ul style="list-style-type: none"> <li><b>TRUE</b></li> <li><b>FALSE</b></li> </ul>

### 9.3.4.11 Inline form for spline trigger type “Set pulse output”

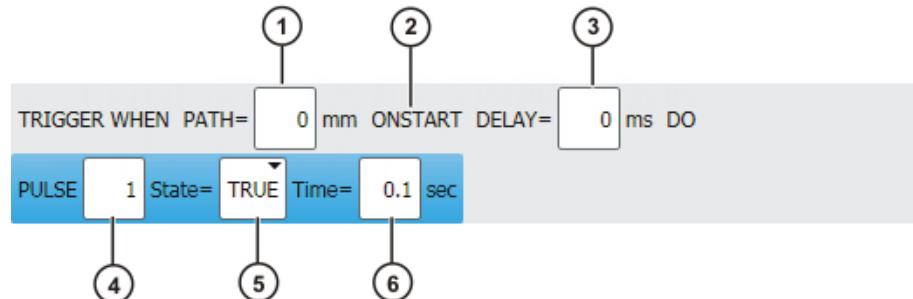
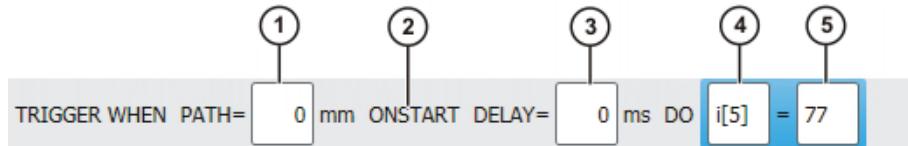


Fig. 9-20: Inline form for spline trigger type Set pulse output

Item	Description
1	If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. If no shift in space is desired, enter the value 0. <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion.</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group “Expert”: <b>Toggle path</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
2	<b>Toggle OnStart</b> can be used to set or cancel the parameter <b>ONSTART</b> . <ul style="list-style-type: none"> <li>■ Without <b>ONSTART</b>: the PATH value refers to the end point.</li> <li>■ With <b>ONSTART</b>: the PATH value refers to the start point.</li> </ul>
3	If the statement is to be shifted in time (relative to the value in item 1), the desired duration must be specified here. If no shift in time is desired, enter the value 0. <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group “Expert”: <b>Toggle Delay</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
4	Output number <ul style="list-style-type: none"> <li>■ <b>1 ... 4096</b></li> </ul>
5	State to which the output is switched <ul style="list-style-type: none"> <li>■ <b>TRUE</b>: “High” level</li> <li>■ <b>FALSE</b>: “Low” level</li> </ul>
6	Length of the pulse <ul style="list-style-type: none"> <li>■ <b>0.10 ... 3.00 s</b></li> </ul>

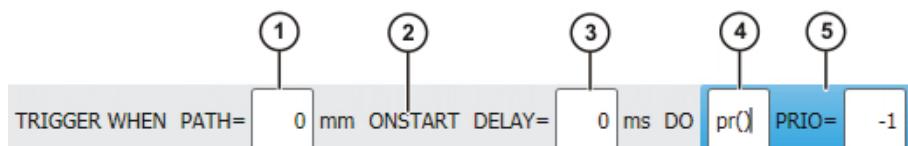
### 9.3.4.12 Inline form for spline trigger type “Trigger assignment”



**Fig. 9-21: Inline form for spline trigger type Trigger assignment**

Item	Description
1	If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. If no shift in space is desired, enter the value 0. <ul style="list-style-type: none"> <li>Positive value: shifts the statement towards the end of the motion.</li> <li>Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle path</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
2	<b>Toggle OnStart</b> can be used to set or cancel the parameter <b>ONSTART</b> . <ul style="list-style-type: none"> <li>Without <b>ONSTART</b>: the PATH value refers to the end point.</li> <li>With <b>ONSTART</b>: the PATH value refers to the start point.</li> </ul>
3	If the statement is to be shifted in time (relative to the value in item 1), the desired duration must be specified here. If no shift in time is desired, enter the value 0. <ul style="list-style-type: none"> <li>Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms</li> <li>Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle Delay</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)
4	Variable to which a value is to be assigned <b>Note:</b> Runtime variables cannot be used.
5	Value to be assigned to the variable

### 9.3.4.13 Inline form for spline trigger type “Trigger function call”



**Fig. 9-22: Inline form for spline trigger type Trigger function call**

Item	Description
1	If the statement is to be shifted in space, the desired distance from the start or end point must be specified here. If no shift in space is desired, enter the value 0. <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion.</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle path</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. ( <a href="#">&gt;&gt;&gt; 9.3.4.14 "Limits for functions in the spline trigger" Page 272</a> )
2	<b>Toggle OnStart</b> can be used to set or cancel the parameter <b>ONSTART</b> . <ul style="list-style-type: none"> <li>■ Without <b>ONSTART</b>: the PATH value refers to the end point.</li> <li>■ With <b>ONSTART</b>: the PATH value refers to the start point.</li> </ul>
3	If the statement is to be shifted in time (relative to the value in item 1), the desired duration must be specified here. If no shift in time is desired, enter the value 0. <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> Only for the user group "Expert": <b>Toggle Delay</b> makes it possible to enter a variable, constant or function in this box. The functions are subject to constraints. ( <a href="#">&gt;&gt;&gt; 9.3.4.14 "Limits for functions in the spline trigger" Page 272</a> )
4	Name of the subprogram to be called
5	A priority must be entered in the <b>PRIOR</b> box. Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: <b>PRIOR</b> = -1. <p>If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.</p>

#### 9.3.4.14 Limits for functions in the spline trigger

The values for **PATH** and **DELAY** can be assigned using functions. The following constraints apply to these functions:

- The KRL program containing the function must have the attribute **Hidden**.
- The function must be globally valid.
- The functions may only contain the following statements or elements:
  - Value assignments
  - IF statements
  - Comments
  - Blank lines
  - RETURN
  - Read system variable
  - Call predefined KRL function

### 9.3.5 Copying spline inline forms

#### Overview

The following copying operations can be carried out:

- Copy an individual motion into a spline block
- Copy a spline block
- Copy a spline segment to another spline block
- Copy a spline segment out of a spline block

#### Precondition

- Expert user group
- Program is selected or open.
- T1, T2 or AUT mode

#### Copy

##### **Copy an individual motion into a spline block:**

The following individual motions can be copied and pasted into a spline block:

- SLIN
- SCIRC
- LIN
- CIRC

Precondition:

- The following frame data (= data in the option window **Frames**) of the individual motion and the block are identical: **Tool**, **Base** and **Interpolation mode**

##### **Copy spline block:**

A spline block can be copied and pasted at a different point in the program. Only the empty block is pasted. It is not possible to copy a block and its contents at the same time.

The contents must be copied and pasted separately.

##### **Copy a spline segment to another spline block:**

One or more spline segments can be copied and pasted to another block.

Precondition:

- The following frame data (= data in the option window **Frames**) of the spline blocks are identical: **Tool**, **Base** and **Interpolation mode**

##### **Copy a spline segment out of a spline block:**

One or more spline segments can be copied and pasted outside a spline block. The motion types change as follows:

Spline segment...	... becomes an individual motion
SLIN	SLIN
SCIRC	SCIRC
SPL	PTP

- For individual SLIN, SCIRC motions: the frame and motion data from the segment are applied if available; otherwise, the data from the spline block are applied.
- For individual PTP motion: the position and frame data from the SPL are applied to the PTP. Motion data are not applied.

### 9.3.6 Converting spline inline forms from 8.1

#### Description

More parameters can be set in spline inline forms in KSS 8.2 than in KSS 8.1. This allows more detailed determination of the motion characteristics.

Programs with inline forms from 8.1 can be used in 8.2. For this, values must be assigned to the new parameters. This is done by opening the inline form and closing it again. Default values are automatically assigned to all new parameters.

**Precondition**

- A program is selected.
- Operating mode T1

**Procedure**

1. Position the cursor in the line with the inline form.
2. Press **Change**. The inline form is opened. The default values are automatically set for all new parameters.
3. If required, modify the values.
4. Press **Cmd OK**.
5. Repeat steps 1 to 4 for all spline inline forms in the program.

## 9.4 Modifying motion parameters

**Precondition**

- A program is selected.
- Operating mode T1

**Procedure**

1. Position the cursor in the line containing the instruction that is to be changed.
2. Press **Change**. The inline form for this instruction is opened.
3. Modify parameters.
4. Save changes by pressing **Cmd Ok**.

## 9.5 Modifying the coordinates of a taught point

**Description**

The coordinates of a taught point can be modified. This is done by moving to the new position and overwriting the old point with the new position.

**Precondition**

- A program is selected.
- Operating mode T1

**Procedure**

1. Move the TCP to the desired position.
2. Position the cursor in the line containing the motion instruction that is to be changed.
3. Press **Change**. The inline form for this instruction is opened.
4. For PTP and LIN motions: Press **Touch Up** to accept the current position of the TCP as the new end point.

For CIRC motions:

- Press **Teach Aux** to accept the current position of the TCP as the new auxiliary point.
- Or press **Teach End** to accept the current position of the TCP as the new end point.

5. Confirm the request for confirmation with **Yes**.
6. Save change by pressing **Cmd Ok**.

## 9.6 Programming logic instructions

### 9.6.1 Inputs/outputs

#### Digital inputs/outputs

The robot controller can manage up to 4096 digital inputs and 4096 digital outputs. The configuration is customer-specific.

### Analog inputs/outputs

The robot controller can manage 32 analog inputs and 32 analog outputs. The configuration is customer-specific.

Permissible range of values for inputs/outputs: -1.0 to +1.0. This corresponds to a voltage range from -10 V to +10 V. If the value is exceeded, the input/output takes the maximum value and a message is displayed until the value is back in the permissible range.

The inputs/outputs are managed via the following system variables:

	Inputs	Outputs
Digital	\$IN[1] ... \$IN[4096]	\$OUT[1] ... \$OUT[4096]
Analog	\$ANIN[1] ... \$ANIN[32]	\$ANOUT[1] ... \$ANOUT[32]

## 9.6.2 Setting a digital output - OUT

- Precondition**
- A program is selected.
  - Operating mode T1

- Procedure**
1. Position the cursor in the line after which the logic instruction is to be inserted.
  2. Select the menu sequence **Commands > Logic > OUT > OUT**.
  3. Set the parameters in the inline form.  
(>>> 9.6.3 "Inline form "OUT"" Page 275)
  4. Save instruction with **Cmd Ok**.

## 9.6.3 Inline form "OUT"

The instruction sets a digital output.

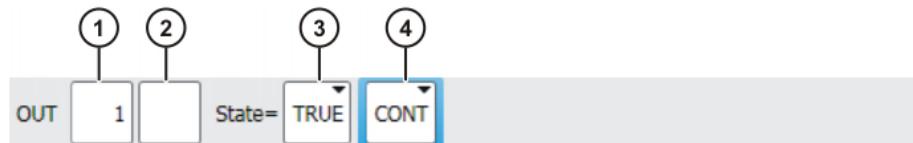


Fig. 9-23: Inline form "OUT"

Item	Description
1	Output number ■ 1 ... 4096
2	If a name exists for the output, this name is displayed. Only for the user group "Expert": A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
3	State to which the output is switched ■ TRUE ■ FALSE
4	■ CONT: Execution in the advance run ■ [Empty box]: Execution with advance run stop

## 9.6.4 Setting a pulse output - PULSE

### Precondition

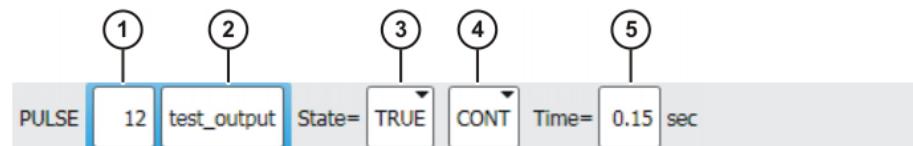
- A program is selected.
- Operating mode T1

### Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands > Logic > OUT > PULSE**.
3. Set the parameters in the inline form.  
(>>> 9.6.5 "Inline form "PULSE"" Page 276)
4. Save instruction with **Cmd Ok**.

## 9.6.5 Inline form "PULSE"

The instruction sets a pulse of a defined length.



**Fig. 9-24: Inline form "PULSE"**

Item	Description
1	Output number ■ <b>1 ... 4096</b>
2	If a name exists for the output, this name is displayed. Only for the user group "Expert": A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
3	State to which the output is switched ■ <b>TRUE</b> : "High" level ■ <b>FALSE</b> : "Low" level
4	■ <b>CONT</b> : Execution in the advance run ■ <b>[Empty box]</b> : Execution with advance run stop
5	Length of the pulse ■ <b>0.10 ... 3.00 s</b>

## 9.6.6 Setting an analog output - ANOUT

### Precondition

- A program is selected.
- Operating mode T1

### Procedure

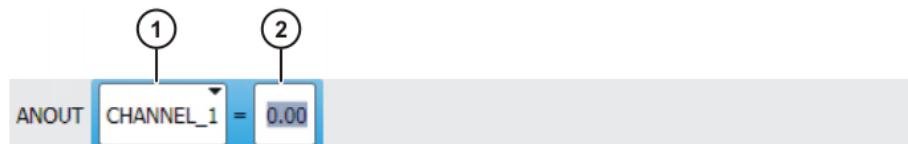
1. Position the cursor in the line after which the instruction is to be inserted.
2. Select **Commands > Analog output > Static or Dynamic**.
3. Set the parameters in the inline form.  
(>>> 9.6.7 "Inline form "ANOUT" (static)" Page 277)  
(>>> 9.6.8 "Inline form "ANOUT" (dynamic)" Page 277)
4. Save instruction with **Cmd Ok**.

### 9.6.7 Inline form “ANOUT” (static)

This instruction sets a static analog output.

A maximum of 8 analog outputs (static and dynamic together) can be used at any one time. ANOUT triggers an advance run stop.

The voltage is set to a fixed level by means of a factor. The actual voltage level depends on the analog module used. For example, a 10 V module with a factor of 0.5 provides a voltage of 5 V.



**Fig. 9-25: Inline form “ANOUT” (static)**

Item	Description
1	Number of the analog output ■ CHANNEL_1 ... CHANNEL_32
2	Factor for the voltage ■ 0 ... 1 (intervals: 0.01)

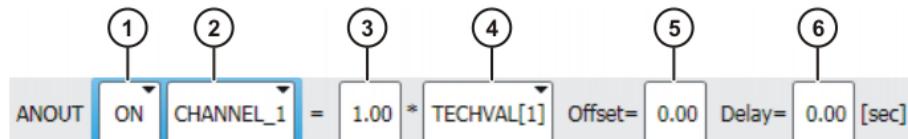
### 9.6.8 Inline form “ANOUT” (dynamic)

This instruction activates or deactivates a dynamic analog output.

A maximum of 4 dynamic analog outputs can be activated at any one time. ANOUT triggers an advance run stop.

The voltage is determined by a factor. The actual voltage level depends on the following values:

- Velocity or function generator  
For example, a velocity of 1 m/s with a factor of 0.5 results in a voltage of 5 V.
- Offset  
For example, an offset of +0.15 for a voltage of 0.5 V results in a voltage of 6.5 V.



**Fig. 9-26: Inline form “ANOUT” (dynamic)**

Item	Description
1	Activation or deactivation of the analog output ■ ON ■ OFF
2	Number of the analog output ■ CHANNEL_1 ... CHANNEL_32
3	Factor for the voltage ■ 0 ... 10 (intervals: 0.01)

Item	Description
4	<ul style="list-style-type: none"> <li>■ <b>VEL_ACT</b>: The voltage is dependent on the velocity.</li> <li>■ <b>TECHVAL[1] ... TECHVAL[6]</b>: The voltage is controlled by a function generator.</li> </ul>
5	Value by which the voltage is increased or decreased <ul style="list-style-type: none"> <li>■ <b>-1 ... +1</b> (intervals: 0.01)</li> </ul>
6	Time by which the output signal is delayed (+) or brought forward (-) <ul style="list-style-type: none"> <li>■ <b>-0.2 ... +0.5 s</b></li> </ul>

### 9.6.9 Programming a wait time - WAIT

- Precondition**
- A program is selected.
  - Operating mode T1
- Procedure**
1. Position the cursor in the line after which the logic instruction is to be inserted.
  2. Select the menu sequence **Commands > Logic > WAIT**.
  3. Set the parameters in the inline form.  
(*>>> 9.6.10 "Inline form "WAIT"" Page 278*)
  4. Save instruction with **Cmd Ok**.

### 9.6.10 Inline form "WAIT"

WAIT can be used to program a wait time. The robot motion is stopped for a programmed time. WAIT always triggers an advance run stop.

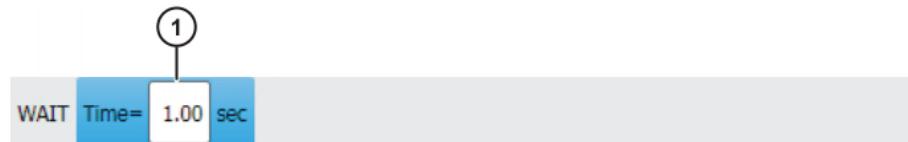


Fig. 9-27: Inline form "WAIT"

Item	Description
1	Wait time <ul style="list-style-type: none"> <li>■ <b>≥ 0 s</b></li> </ul>

### 9.6.11 Programming a signal-dependent wait function - WAITFOR

- Precondition**
- A program is selected.
  - Operating mode T1
- Procedure**
1. Position the cursor in the line after which the logic instruction is to be inserted.
  2. Select the menu sequence **Commands > Logic > WAITFOR**.
  3. Set the parameters in the inline form.  
(*>>> 9.6.12 "Inline form "WAITFOR"" Page 279*)
  4. Save instruction with **Cmd Ok**.

### 9.6.12 Inline form "WAITFOR"

The instruction sets a signal-dependent wait function.

If required, several signals (maximum 12) can be linked. If a logic operation is added, boxes are displayed in the inline form for the additional signals and links.

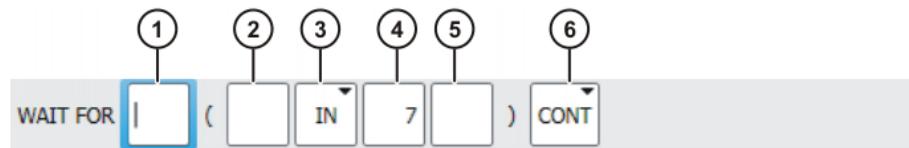


Fig. 9-28: Inline form "WAITFOR"

Item	Description
1	Add external logic operation. The operator is situated between the bracketed expressions. <ul style="list-style-type: none"> <li>■ <b>AND</b></li> <li>■ <b>OR</b></li> <li>■ <b>EXOR</b></li> </ul> Add NOT. <ul style="list-style-type: none"> <li>■ <b>NOT</b></li> <li>■ <b>[Empty box]</b></li> </ul> Enter the desired operator by means of the corresponding button.
2	Add internal logic operation. The operator is situated inside a bracketed expression. <ul style="list-style-type: none"> <li>■ <b>AND</b></li> <li>■ <b>OR</b></li> <li>■ <b>EXOR</b></li> </ul> Add NOT. <ul style="list-style-type: none"> <li>■ <b>NOT</b></li> <li>■ <b>[Empty box]</b></li> </ul> Enter the desired operator by means of the corresponding button.
3	Signal for which the system is waiting <ul style="list-style-type: none"> <li>■ <b>IN</b></li> <li>■ <b>OUT</b></li> <li>■ <b>CYCFLAG</b></li> <li>■ <b>TIMER</b></li> <li>■ <b>FLAG</b></li> </ul>
4	Number of the signal <ul style="list-style-type: none"> <li>■ <b>1 ... 4096</b></li> </ul>
5	If a name exists for the signal, this name is displayed. Only for the user group "Expert": A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
6	<ul style="list-style-type: none"> <li>■ <b>CONT</b>: Execution in the advance run</li> <li>■ <b>[Empty box]</b>: Execution with advance run stop</li> </ul>

### 9.6.13 Switching on the path - SYN OUT

#### Precondition

- A program is selected.
- Operating mode T1

#### Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands > Logic > OUT > SYN OUT**.
3. Set the parameters in the inline form.
 

(>>> 9.6.14 "Inline form "SYN OUT", option "START/END"" Page 280)  
   (>>> 9.6.15 "Inline form "SYN OUT", option "PATH"" Page 283)
4. Save instruction with **Cmd Ok**.

### 9.6.14 Inline form "SYN OUT", option "START/END"

A switching action can be triggered relative to the start or end point of a motion block. The switching action can be delayed or brought forward. The motion block can be a LIN, CIRC or PTP motion.

Possible applications include:

- Closing or opening the weld gun during spot welding
- Switching the welding current on/off during arc welding
- Starting or stopping the flow of adhesive in bonding or sealing applications.

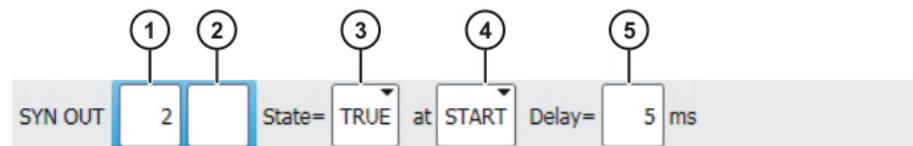


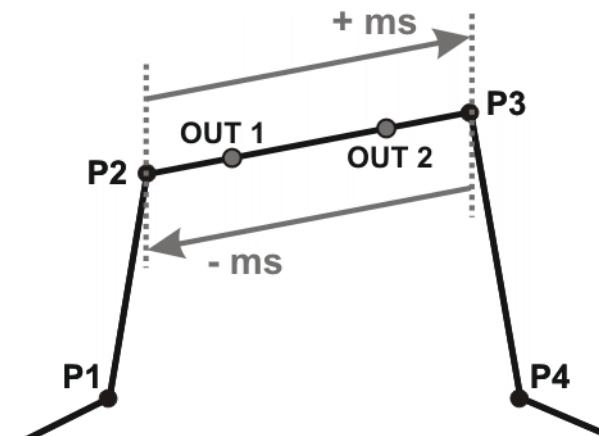
Fig. 9-29: Inline form "SYN OUT", option "START/END"

Item	Description
1	Output number  ■ 1 ... 4096
2	If a name exists for the output, this name is displayed.  Only for the user group "Expert":  A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
3	State to which the output is switched  ■ TRUE ■ FALSE
4	Point at which switching is carried out  ■ START: Switching is carried out at the start point of the motion block. ■ END: Switching is carried out at the end point of the motion block. ■ PATH:
5	Switching action delay  ■ -1,000 ... +1,000 ms  <b>Note:</b> The time specification is absolute. The switching point varies according to the velocity of the robot.

**Example 1**

Start point and end point are exact positioning points.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```



**Fig. 9-30**

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits.

Switching limits:

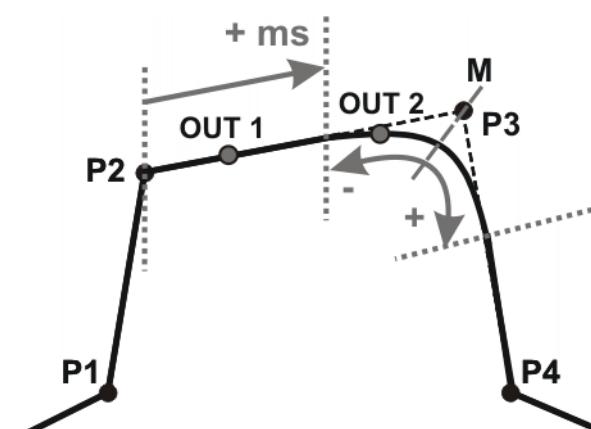
- START: The switching point can be delayed, at most, as far as exact positioning point P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as exact positioning point P2 (- ms).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

**Example 2**

Start point is exact positioning point, end point is approximated.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```



**Fig. 9-31**

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

#### Switching limits:

- START: The switching point can be delayed, at most, as far as the start of the approximate positioning range of P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P3 (-).  
The switching point can be delayed, at most, as far as the end of the approximate positioning range of P3 (+).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

#### Example 3

Start point and end point are approximated

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 CONT VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

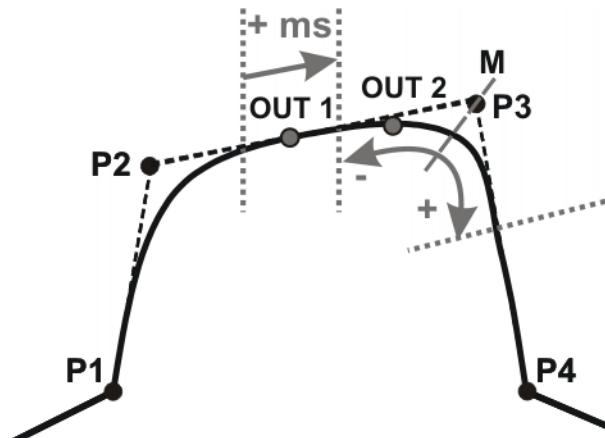


Fig. 9-32

OUT 1 and OUT 2 specify approximate positions at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

#### Switching limits:

- START: The switching point can be situated, at the earliest, at the end of the approximate positioning range of P2.  
The switching point can be delayed, at most, as far as the start of the approximate positioning range of P3 (+ ms).
- END: The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P3 (-).  
The switching point can be delayed, at most, as far as the end of the approximate positioning range of P3 (+).

If greater values are specified for the delay, the controller automatically switches at the switching limit.

### 9.6.15 Inline form "SYN OUT", option "PATH"

A switching action can be triggered relative to the end point of a motion block. The switching action can be shifted in space and delayed or brought forward. The motion block can be a LIN or CIRC motion. It must not be a PTP motion.

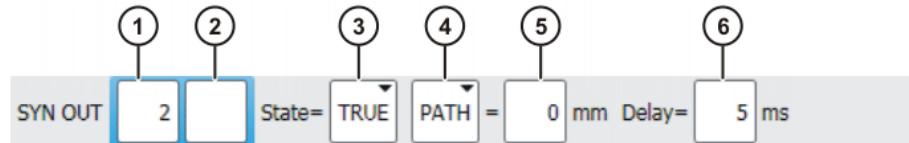


Fig. 9-33: Inline form "SYN OUT", option "PATH"

Item	Description
1	Output number ■ 1 ... 4096
2	If a name exists for the output, this name is displayed. Only for the user group "Expert": A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
3	State to which the output is switched ■ TRUE ■ FALSE
4	Point at which switching is carried out ■ PATH: Switching is carried out at the end point of the motion block. ■ START: (» 9.6.14 "Inline form "SYN OUT", option "START/END" » Page 280) ■ END: (» 9.6.14 "Inline form "SYN OUT", option "START/END" » Page 280)
5	Distance from the switching point to the end point ■ -2,000 ... +2,000 mm This box is only displayed if <b>PATH</b> has been selected.
6	Switching action delay ■ -1,000 ... +1,000 ms <b>Note:</b> The time specification is absolute. The switching point varies according to the velocity of the robot.

#### Example 1

Start point is exact positioning point, end point is approximated.

```
LIN P1 VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

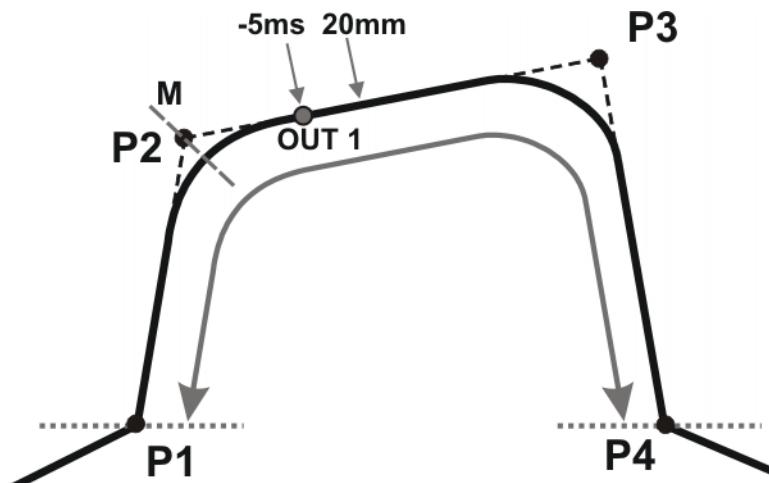


Fig. 9-34

OUT 1 specifies the approximate position at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- The switching point can be brought forward, at most, as far as exact positioning point P1.
- The switching point can be delayed, at most, as far as the next exact positioning point P4. If P3 was an exact positioning point, the switching point could be delayed, at most, as far as P3.

If greater values are specified for the shift in space or time, the controller automatically switches at the switching limit.

### Example 2

Start point and end point are approximated

```
LIN P1 CONT VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

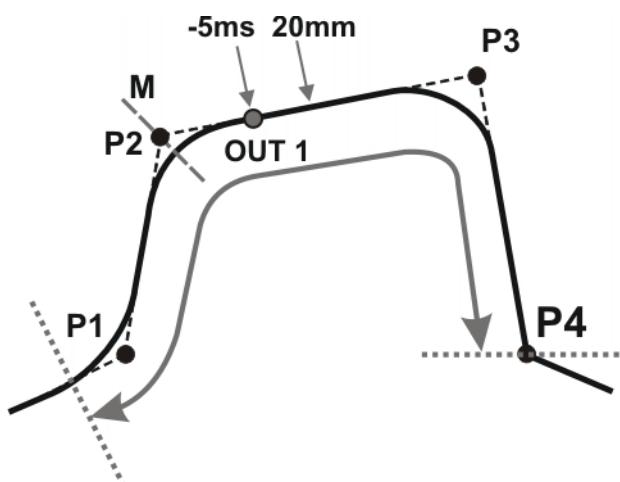


Fig. 9-35

OUT 1 specifies the approximate position at which switching is to occur. The dotted lines indicate the switching limits. M = middle of the approximate positioning range.

Switching limits:

- The switching point can be brought forward, at most, as far as the start of the approximate positioning range of P1.
- The switching point can be delayed, at most, as far as the next exact positioning point P4. If P3 was an exact positioning point, the switching point could be delayed, at most, as far as P3.

If greater values are specified for the shift in space or time, the controller automatically switches at the switching limit.

#### 9.6.16 Setting a pulse on the path - SYN PULSE

##### Precondition

- A program is selected.
- Operating mode T1

##### Procedure

1. Position the cursor in the line after which the logic instruction is to be inserted.
2. Select the menu sequence **Commands > Logic > OUT > SYN PULSE**.
3. Set the parameters in the inline form.  
(>>> 9.6.17 "Inline form "SYN PULSE"" Page 285)
4. Save instruction with **Cmd Ok**.

#### 9.6.17 Inline form "SYN PULSE"

A pulse can be triggered relative to the start or end point of a motion block. The pulse can be delayed or brought forward and shifted in space.

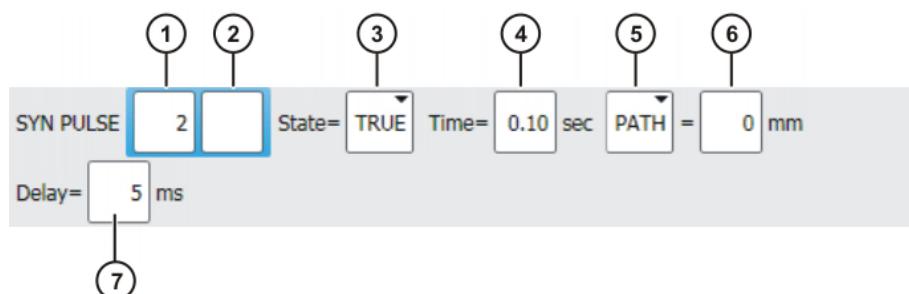


Fig. 9-36: Inline form "SYN PULSE"

Item	Description
1	Output number ■ 1 ... 4096
2	If a name exists for the output, this name is displayed. Only for the user group "Expert": A name can be entered by pressing <b>Long text</b> . The name is freely selectable.
3	State to which the output is switched ■ TRUE ■ FALSE
4	Duration of the pulse ■ 0.1 ... 3 s

Item	Description
5	<ul style="list-style-type: none"> <li>■ <b>START:</b> The pulse is triggered at the start point of the motion block.</li> <li>■ <b>END:</b> The pulse is triggered at the end point of the motion block.</li> </ul> <p>See SYN OUT for examples and switching limits.            (&gt;&gt;&gt; 9.6.14 "Inline form "SYN OUT", option "START/END""            Page 280)</p> <ul style="list-style-type: none"> <li>■ <b>PATH:</b> The pulse is triggered at the end point of the motion block.</li> </ul> <p>See SYN OUT for examples and switching limits.            (&gt;&gt;&gt; 9.6.15 "Inline form "SYN OUT", option "PATH"" Page 283)</p>
6	<p>Distance from the switching point to the end point</p> <ul style="list-style-type: none"> <li>■ <b>-2,000 ... +2,000 mm</b></li> </ul> <p>This box is only displayed if <b>PATH</b> has been selected.</p>
7	<p>Switching action delay</p> <ul style="list-style-type: none"> <li>■ <b>-1,000 ... +1,000 ms</b></li> </ul> <p><b>Note:</b> The time specification is absolute. The switching point varies according to the velocity of the robot.</p>

### 9.6.18 Modifying a logic instruction

- |                     |  |
|---------------------|--|
| <b>Precondition</b> | <ul style="list-style-type: none"> <li>■ A program is selected.</li> <li>■ Operating mode T1</li> </ul>  |
| <b>Procedure</b>    | <ol style="list-style-type: none"> <li>1. Position the cursor in the line containing the instruction that is to be changed.</li> <li>2. Press <b>Change</b>. The inline form for this instruction is opened.</li> <li>3. Change the parameters.</li> <li>4. Save changes by pressing <b>Cmd Ok</b>.</li> </ol> |

## 10 Programming for user group "Expert" (KRL syntax)

<b>NOTICE</b>	In the case of programs with the following axis motions or positions, the film of lubricant on the gear units of the axes may break down:
■ Motions <3°	
■ Oscillating motions	
■ Areas of gear units permanently facing upwards	
	It must be ensured that the gear units have a sufficient supply of oil. For this, in the case of oscillating motions or short motions (<3°), programming must be carried out in such a way that the affected axes regularly move more than 40° (e.g. once per cycle).
	In the case of areas of gear units permanently facing upwards, sufficient oil supply must be achieved by programming re-orientations of the in-line wrist. In this way, the oil can reach all areas of the gear units by means of gravity.
	Required frequency of re-orientations:
■ With low loads (gear unit temperature <+35 °C): daily	
■ With medium loads (gear unit temperature +35 °C to 55 °C): hourly	
■ With heavy loads (gear unit temperature >+55 °C): every 10 minutes	
	Failure to observe this precaution may result in damage to the gear units.

	If a selected program is edited in the user group "Expert", the cursor must then be removed from the edited line and positioned in any other line!
	Only in this way is it certain that the editing will be applied when the program is deselected again.

### 10.1 Overview of KRL syntax

Variables and declarations	
DECL	(>>> 10.4.1 "DECL" Page 293)
ENUM	(>>> 10.4.2 "ENUM" Page 295)
STRUC	(>>> 10.4.3 "STRUC" Page 296)

Motion programming	
PTP	(>>> 10.5.1 "PTP" Page 297)
PTP_REL	(>>> 10.5.2 "PTP_REL" Page 298)
LIN	(>>> 10.5.3 "LIN" Page 299)
LIN_REL	(>>> 10.5.4 "LIN_REL" Page 300)
CIRC	(>>> 10.5.5 "CIRC" Page 302)
CIRC_REL	(>>> 10.5.6 "CIRC_REL" Page 303)
SPLINE ... ENDSPLINE	(>>> 10.6.2 "SPLINE ... ENDSPLINE" Page 305)
SLIN	(>>> 10.6.3 "SLIN" Page 306)
SCIRC	(>>> 10.6.4 "SCIRC" Page 307)
SPL	(>>> 10.6.5 "SPL" Page 309)
TIME_BLOCK	(>>> 10.6.6 "TIME_BLOCK" Page 309)

Program execution control	
CONTINUE	(>>> 10.7.1 "CONTINUE" Page 314)
EXIT	(>>> 10.7.2 "EXIT" Page 314)
FOR ... TO ... ENDFOR	(>>> 10.7.3 "FOR ... TO ... ENDFOR" Page 315)

<b>Program execution control</b>	
GOTO	(>>> 10.7.4 "GOTO" Page 315)
HALT	(>>> 10.7.5 "HALT" Page 316)
IF ... THEN ... ENDIF	(>>> 10.7.6 "IF ... THEN ... ENDIF" Page 316)
LOOP ... ENDLOOP	(>>> 10.7.7 "LOOP ... ENDLOOP" Page 317)
REPEAT ... UNTIL	(>>> 10.7.8 "REPEAT ... UNTIL" Page 317)
SWITCH ... CASE ... END SWITCH	(>>> 10.7.9 "SWITCH ... CASE ... END SWITCH" Page 318)
WAIT ... FOR	(>>> 10.7.10 "WAIT FOR" Page 319)
WAIT ... SEC	(>>> 10.7.11 "WAIT SEC" Page 320)
WHILE ... END WHILE	(>>> 10.7.12 "WHILE ... END WHILE" Page 320)
<b>Inputs/outputs</b>	
ANIN	(>>> 10.8.1 "ANIN" Page 321)
ANOUT	(>>> 10.8.2 "ANOUT" Page 322)
PULSE	(>>> 10.8.3 "PULSE" Page 323)
SIGNAL	(>>> 10.8.4 "SIGNAL" Page 326)
<b>Subprograms and functions</b>	
RETURN	(>>> 10.9.1 "RETURN" Page 327)
<b>Interrupt programming</b>	
BRAKE	(>>> 10.10.1 "BRAKE" Page 328)
INTERRUPT	(>>> 10.10.3 "INTERRUPT" Page 330)
INTER- RUPT ... DECL ... WHEN ... DO	(>>> 10.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 329)
RESUME	(>>> 10.10.4 "RESUME" Page 332)
<b>Path-related switching actions (=Trigger)</b>	
TRIGGER WHEN DISTANCE	(>>> 10.11.1 "TRIGGER WHEN DISTANCE" Page 333)
TRIGGER WHEN PATH	(>>> 10.11.2 "TRIGGER WHEN PATH" Page 336)
TRIGGER WHEN PATH (for spline)	(>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 339)
<b>Communication</b>	
(>>> 10.12 "Communication" Page 343)	
<b>System functions</b>	
VARSTATE()	(>>> 10.13.1 "VARSTATE()" Page 343)
ROB_STOP()	(>>> 10.13.2 "ROB_STOP() and ROB_STOP_RELEASE()" Page 345)
<b>Manipulating string variables</b>	
(>>> 10.14 "Editing string variables" Page 346)	

## 10.2 Symbols and fonts

The following symbols and fonts are used in the syntax descriptions:

Syntax element	Representation
KRL code	<ul style="list-style-type: none"> <li>■ Courier font</li> <li>■ Upper-case letters</li> </ul> <p>Examples: GLOBAL; ANIN ON; OFFSET</p>
Elements that must be replaced by program-specific entries	<ul style="list-style-type: none"> <li>■ Italics</li> <li>■ Upper/lower-case letters</li> </ul> <p>Examples: <i>Distance</i>; <i>Time</i>; <i>Format</i></p>
Optional elements	<ul style="list-style-type: none"> <li>■ In angle brackets</li> </ul> <p>Example: &lt;STEP Increment&gt;</p>
Elements that are mutually exclusive	<ul style="list-style-type: none"> <li>■ Separated by the " " symbol</li> </ul> <p>Example: IN  OUT</p>

## 10.3 Important KRL terms

### 10.3.1 SRC files and DAT files

A KRL program generally consists of an **SRC file** and a **DAT file** of the same name.

- SRC file: contains the program code.
- DAT file: contains permanent data and point coordinates. The DAT file is also called a **data list**.

The SRC file and associated DAT file together are called a **module**.

Depending on the user group, programs in the Navigator are displayed as modules or individual files:

- User group "User"  
A program is displayed as a module. The SRC file and the DAT file exist in the background. They are not visible for the user and cannot be edited individually.
- User group "Expert"  
By default, the SRC file and the DAT file are displayed individually. They can be edited individually.

### 10.3.2 Subprograms and functions

#### Subprograms

Subprograms are programs which are accessed by means of branches from the main program. Once the subprogram has been executed, the main program is resumed from the line directly after the subprogram call.

- **Local subprograms** are contained in the same SRC file as the main program. They can be made to be recognized globally using the keyword GLOBAL .
- **Global subprograms** are programs with a separate SRC file of their own, which is accessed from another program by means of a branch.

#### Functions

Functions, like subprograms, are programs which are accessed by means of branches from the main program. In addition, however, they also have a data type and always return a value to the main program.

### 10.3.3 Naming conventions and keywords

#### Names

Examples of names in KRL: variable names, program names, point names

- Names in KRL can have a maximum length of 24 characters.
- Names in KRL can consist of letters (A-Z), numbers (0-9) and the signs “\_” and “\$”.
- Names in KRL must not begin with a number.
- Names in KRL must not be keywords.



The names of all system variables begin with the “\$” sign. To avoid confusion, do not begin the names of user-defined variables with this sign.

#### Keywords

Keywords are sequences of letters having a fixed meaning. They must not be used in programs in any way other than with this meaning. No distinction is made between uppercase and lowercase letters. A keyword remains valid irrespective of the way in which it is written.

**Example:** The sequence of letters CASE is an integral part of the KRL syntax SWITCH ... CASE ... ENDSWITCH. For this reason, CASE must not be used in any other way, e.g. as a variable name.

The system distinguishes between reserved and non-reserved keywords:

- Reserved keywords

These may only be used with their defined meaning.

- Non-reserved keywords

With non-reserved keywords, the meaning is restricted to a particular context. Outside of this context, a non-reserved keyword is interpreted by the compiler as a name.



In practice, it is not helpful to distinguish between reserved and non-reserved keywords. To avoid error messages or compiler problems, keywords are thus never used other than with their defined meaning.

#### Overview of important keywords:

All elements of the KRL syntax described in this documentation that are not program-specific are keywords.

The following important keywords are worth a particular mention:

AXIS	ENDFCT
BOOL	ENDFOR
CHAR	ENDIF
CAST_FROM	ENDLOOP
CAST_TO	ENDSWITCH
CCLOSE	ENDWHILE
CHANNEL	EXT
CIOCTL	EXTFCT
CONFIRM	FALSE
CONST	FRAME
COPEN	GLOBAL
CREAD	INT
CWRITE	MAXIMUM
DEF	MINIMUM
DEFAULT	POS
DEFDAT	PRIO

DEFFCT	PUBLIC
E6AXIS	SREAD
E6POS	SWRITE
END	REAL
ENDDAT	TRUE

### 10.3.4 Data types

#### Overview

There are 2 kinds of data types:

- User-defined data types  
User-defined data types are always derived from the data types ENUM or STRUC.
- Predefined data types, e.g.:
  - Simple data types
  - Data types for motion programming

The following simple data types are predefined:

Data type	Keyword	Description
Integer	INT	Integer ■ $-2^{31}-1 \dots 2^{31}-1$ Examples: 1; 32; 345
Real	REAL	Floating-point number ■ $+1.1E-38 \dots +3.4E+38$ Examples: 1.43; 38.50; 300.25
Boolean	BOOL	Logic state ■ TRUE ■ FALSE
Character	CHAR	1 character ■ ASCII character Examples: "A"; "1"; "q"

The following data types for motion programming are predefined:

#### Structure type AXIS

A1 to A6 are angle values (rotational axes) or translation values (translational axes) for the axis-specific movement of robot axes 1 to 6.

```
STRUCT AXIS REAL A1, A2, A3, A4, A5, A6
```

#### Structure type E6AXIS

E1 to E6 are angle values or translation values of the external axes 7 to 12.

```
STRUCT E6AXIS REAL A1, A2, A3, A4, A5, A6, E1, E2, E3, E4, E5, E6
```

#### Structure type FRAME

X, Y and Z are space coordinates, while A, B and C are the orientation of the coordinate system.

```
STRUCT FRAME REAL X, Y, Z, A, B, C
```

#### Structure types POS and E6POS

S (Status) and T (Turn) define axis positions unambiguously.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

### 10.3.5 Areas of validity

#### Local

	Data object	Area of validity
Variable		Valid in the program code between DEF and END containing the declaration of the variables.
Constant		Valid in the module to which the data list in which the constant was declared belongs.
User-defined data type		If the data type has been defined in an SRC file: valid at, or below, the program level in which it was declared. If the data type has been defined in a DAT file: valid in the SRC file that belongs to the DAT file.
Subprogram		Valid in the main program of the shared SRC file.
Function		Valid in the main program of the shared SRC file.
Interrupt		Valid at, or below, the programming level in which it was declared.

#### Global

The data objects referred to under "Local" are globally valid if they are declared using the keyword GLOBAL.



GLOBAL can only be used for variables and user-defined data types if they have been declared in a data list.

Variables and user-defined data types are also globally valid if they were declared in the section USER GLOBALS in \$CONFIG.DAT.

If there are local and global variables with the same name, the compiler uses the local variable within its area of validity.

Always globally valid:

- The first program in an SRC file. By default, it bears the name of the SRC file.
- Predefined data types
- KRL system variables
- Variables declared in \$CONFIG.DAT

#### Examples

The examples show where the keyword GLOBAL must be positioned.

##### Declaration of a global variable (only in a data list):

<DECL> GLOBAL Data type Variable name

##### Declaration of a global subprogram:

Main program

GLOBAL DEF Subprogram name ()

#### Restriction



Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.

Example:

In DEFDAT PROG(), the enumeration type SWITCH\_TYP has been defined with the keyword GLOBAL:

```
DEFDAT PROG()
GLOBAL ENUM SWITCH_TYP ON, OFF
...
```

If this data type is used in \$CONFIG.DAT, the compiler signals the error “*Type unknown: \*\*\* DECL SWITCH\_TYP MY\_VAR*”.

```
DEFDAT $CONFIG
DECL SWITCH_TYP MY_VAR
...
```

### 10.3.6 Constants

The value of a constant can no longer be modified during program execution after initialization. Constants can be used to prevent a value from being changed accidentally during program execution.

Constants must be declared and, at the same time, initialized in a data list. The data type must be preceded by the keyword CONST.

`DECL <GLOBAL> CONST Data type Variable name = Value`



The keyword CONST must only be used in data lists.

## 10.4 Variables and declarations

### 10.4.1 DECL

**Description** Declaration of variables, arrays and constants

**Syntax** **Declaration of variables**

Declaration of variables in programs:

`<DECL> Data type Name1 <, ..., NameN>`

Declaration of variables in data lists:

`<DECL> <GLOBAL> Data type Name1 <, ..., NameN>`

Declaration of variables in data lists with simultaneous initialization:

`<DECL> <GLOBAL> Data type Name = Value`

In the case of declaration with simultaneous initialization, a separate DECL declaration is required for each variable. It is not possible to declare and initialize several variables with a single DECL declaration.

**Declaration of arrays**

Declaration of arrays in programs:

`<DECL> Data type Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >`

Declaration of arrays in data lists:

`<DECL> <GLOBAL> Data type Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >`

For the declaration of arrays or constant arrays in data lists with simultaneous initialization:

- It is not permissible to declare and initialize in a single line. The initialization must, however, follow directly after the line containing the declaration. There must be no lines, including blank lines, in between.
- If several elements of an array are initialized, the elements must be specified in ascending sequence of the array index (starting from the right-hand array index).
- If the same character string is to be assigned to all of the elements of an array of type CHAR as a default setting, it is not necessary to initialize each array element individually. The right-hand array index is omitted. (No index is written for a one-dimensional array index.)

Declaration of arrays in data lists with simultaneous initialization:

```
<DECL> <GLOBAL> Data type Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1>] = Value1
<Name [1 <, 1, 2>] = Value2>
...
Name [Dimension1 <, Dimension2, Dimension3>] = ValueN
```

Declaration of constant arrays in data lists with simultaneous initialization:

```
DECL <GLOBAL> CONST Data type Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1>] = Value1
<Name [1 <, 1, 2>] = Value2>
...
Name [Dimension1 <, Dimension2, Dimension3>] = ValueN
```

#### Explanation of the syntax

Element	Description
DECL	DECL can be omitted if <i>Data type</i> is a predefined data type. If <i>Data type</i> is a user-defined data type, then DECL is obligatory.
GLOBAL	(>>> 10.3.5 "Areas of validity" Page 292)
CONST	The keyword CONST must only be used in data lists.
<i>Data type</i>	Specification of the desired data type
<i>Name</i>	Name of the object (variable, array or constant) that is being declared.
<i>Dimension</i>	Type: INT <i>Dimension</i> defines the number of array elements for the dimension in question. Arrays have a minimum of 1 and a maximum of 3 dimensions.
<i>Value</i>	The data type of <i>Value</i> must be compatible with <i>Data type</i> , but not necessarily identical. If the data types are compatible, the system automatically matches them.

#### Example 1

Declarations with predefined data types. The keyword DECL can also be omitted.

```
DECL INT X
DECL INT X1, X2
DECL REAL ARRAY_A[7], ARRAY_B[5], A
```

#### Example 2

Declarations of arrays with simultaneous initialization (only possible in data lists).

```

INT A[7]
A[1]=27
A[2]=313
A[6]=11
CHAR TEXT1[80]
TEXT1 []="message"
CHAR TEXT2[2,80]
TEXT2 [1, ]="first message"
TEXT2 [2, ]="second message"

```

## 10.4.2 ENUM

**Description** Definition of an enumeration type (= ENUM data type)

**Syntax** <GLOBAL> ENUM *NameEnumType Constant1*, ..., *ConstantN*

**Explanation of the syntax**

Element	Description
GLOBAL	(>>> 10.3.5 "Areas of validity" Page 292) <b>Note:</b> Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.
<i>NameEnum-Type</i>	Name of the new enumeration type. Recommendation: For user-defined data types, assign names ending in _TYPE, to distinguish them from variable names.
<i>Constant</i>	The constants are the values that a variable of the enumeration type can take. Each constant may only occur once in the definition of the enumeration type.

**Example 1** Definition of an enumeration type with the name COUNTRY\_TYPE.

```
ENUM COUNTRY_TYP SWITZERLAND, AUSTRIA, ITALY, FRANCE
```

Declaration of a variable of type COUNTRY\_TYPE:

```
DECL COUNTRY_TYP MYCOUNTRY
```

Initialization of the variable of type COUNTRY\_TYPE:

```
MYCOUNTRY = #AUSTRIA
```

**Example 2** An enumeration type with the name SWITCH\_TYPE and the constants ON and OFF is defined.

```

DEF PROG()
ENUM SWITCH_TYP ON, OFF
DECL SWITCH_TYP GLUE
    IF A>10 THEN
        GLUE=#ON
    ELSE
        GLUE=#OFF
    ENDIF
END

```

**Restriction**



Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.

Example:

In DEFDAT PROG(), the enumeration type SWITCH\_TYP has been defined with the keyword GLOBAL:

```
DEFDAT PROG()

GLOBAL ENUM SWITCH_TYP ON, OFF
...
```

If this data type is used in \$CONFIG.DAT, the compiler signals the error "Type unknown: \*\*\* DECL SWITCH\_TYP MY\_VAR".

```
DEFDAT $CONFIG

DECL SWITCH_TYP MY_VAR
...
```

### 10.4.3 STRUC

**Description** Definition of a structure type (= STRUC data type). Several data types are combined to form a new data type.

**Syntax**

```
<GLOBAL> STRUC NameStructureType Data type1 Component1A<, Component1B, ...> < , Data type2 Component2A<, Component2B, ...>>
```

**Explanation of the syntax**

Element	Description
GLOBAL	(>>> 10.3.5 "Areas of validity" Page 292)
NameStructureType	<b>Note:</b> Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.
Data type	TYPE: Any data type Structure types are also permissible as data types.
Component	Name of the component. It may only be used once in the structure type. Arrays can only be used as components of a structure type if they have the type CHAR and are one-dimensional. In this case, the array limit follows the name of the array in square brackets in the definition of the structure type.

**Value assignment**

There are 2 ways of assigning values to variables based on a STRUC data type:

- Assignment of values to several components of a variable: with an **aggregate**
- Assignment of a value to a single component of a variable: with the **point separator**

Information regarding the aggregate:

- The values of an aggregate can be simple constants or themselves aggregates; they may not, however, be variables (see also Example 3).
- Not all components of the structure have to be specified in an aggregate.
- The components do not need to be specified in the order in which they have been defined.
- Each component may only be contained once in an aggregate.
- The name of the structure type can be specified at the beginning of an aggregate, separated by a colon.

**Example 1**

Definition of a structure type CAR\_TYPE with the components AIR\_COND, YEAR and PRICE.

```
STRUC CAR_TYP BOOL AIR_COND, INT YEAR, REAL PRICE
```

Declaration of a variable of type CAR\_TYPE:

```
DECL CAR_TYP MYCAR
```

Initialization of the variable MYCAR of type CAR\_TYPE with an **aggregate**:



A variable based on a structure type does not have to be initialized with an aggregate. It is also possible to initialize the components individually with the point separator.

```
MYCAR = {CAR_TYP: PRICE 15000, AIR_COND TRUE, YEAR 2003}
```

Modification of an individual component using the **point separator**:

```
MYCAR.AIR_COND = FALSE
```

## Example 2

Definition of a structure type S\_TYPE with the component NUMBER of data type REAL and of the array component TEXT[80] of data type CHAR.

```
STRUC S_TYP REAL NUMBER, CHAR TEXT[80]
```

## Example 3

Example of aggregates as values of an aggregate:

```
STRUC INNER_TYP INT A, B, C
STRUC OUTER_TYP INNER_TYP Q, R
DECL OUTER_TYP MYVAR
...
MYVAR = {Q {A 1, B 4}, R {A 3, C 2}}
```

## 10.5 Motion programming: PTP, LIN, CIRC

### 10.5.1 PTP

#### Description

Executes a point-to-point motion to the end point. The coordinates of the end point are absolute.

#### Syntax

```
PTP End point <C_PTP <CP approximation>>
```

#### Explanation of the syntax

Element	Description
<i>End point</i>	Type: POS, E6POS, AXIS, E6AXIS, FRAME  The end point can be specified in Cartesian or axis-specific coordinates. Cartesian coordinates refer to the BASE coordinate system.  If not all components of the end point are specified, the controller takes the values of the previous position for the missing components.

Element	Description
C_PTP	Causes the end point to be approximated. The specification C_PTP is sufficient for PTP-PTP approximate positioning. In the case of PTP-CP approximation, i.e. if the approximated PTP block is followed by a LIN or CIRC block, <i>CP approximation</i> must also be specified.
CP approximation	Only for PTP-CP approximate positioning. This parameter defines the earliest point at which the approximate positioning can begin. The possible specifications are: <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>

**Example 1**

End point specified in Cartesian coordinates.

PTP {X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0,S 'B010',T 'B1010'}

**Example 2**

End point specified in axis-specific coordinates. The end point is approximated.

PTP {A1 10,A2 -80.6,A3 -50,A4 0,A5 14.2, A6 0} C\_PTP

**Example 3**

End point specified with only 2 components. For the rest of the components, the controller takes the values of the previous position.

PTP {Z 500,X 123.6}

**10.5.2 PTP\_REL****Description**

Executes a point-to-point motion to the end point. The coordinates of the end point are relative to the current position.



A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

**Syntax**PTP\_REL *End point* <C\_PTP <CP approximation>>

**Explanation of the syntax**

Element	Description
<i>End point</i>	<p>Type: POS, E6POS, AXIS, E6AXIS</p> <p>The end point can be specified in Cartesian or axis-specific coordinates. The controller interprets the coordinates as relative to the current position. Cartesian coordinates refer to the BASE coordinate system.</p> <p>If not all components of the end point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged.</p>
C_PTP	<p>Causes the end point to be approximated.</p> <p>The specification C_PTP is sufficient for PTP-PTP approximate positioning. In the case of PTP-CP approximation, i.e. if the approximated PTP block is followed by a LIN or CIRC block, <i>CP approximation</i> must also be specified.</p>
<i>CP approximation</i>	<p>Only for PTP-CP approximate positioning. This parameter defines the earliest point at which the approximate positioning can begin. The possible specifications are:</p> <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>

**Example 1**

Axis 2 is moved 30 degrees in a negative direction. None of the other axes moves.

```
PTP_REL {A2 -30}
```

**Example 2**

The robot moves 100 mm in the X direction and 200 mm in the negative Z direction from the current position. Y, A, B, C and S remain constant. T is calculated in relation to the shortest path.

```
PTP_REL {X 100,Z -200}
```

**10.5.3 LIN****Description**

Executes a linear motion to the end point. The coordinates of the end point are absolute.

**Syntax**

LIN *End point* <*CP approximation*>

Explanation of the syntax	Element	Description
	<i>End point</i>	Type: POS, E6POS, FRAME
		If not all components of the end point are specified, the controller takes the values of the previous position for the missing components.
		The Status and Turn specifications for an end point of type POS or E6POS are ignored in the case of LIN (and CIRC) motions.
		The coordinates refer to the BASE coordinate system.
	<i>CP approximation</i>	<p>This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are:</p> <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>

**Example**

End point with two components. For the rest of the components, the controller takes the values of the previous position.

```
LIN {Z 500,X 123.6}
```

**10.5.4 LIN\_REL****Description**

Executes a linear motion to the end point. The coordinates of the end point are relative to the current position.



A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

**Syntax**

```
LIN_REL End point <CP approximation> <#BASE | #TOOL>
```

**Explanation of  
the syntax**

Element	Description
<i>End point</i>	<p>Type: POS, E6POS, FRAME</p> <p>The end point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates can refer to the BASE or TOOL coordinate system.</p> <p>If not all components of the end point are specified, the controller automatically sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged.</p> <p>The Status and Turn specifications for an end point of type POS or E6POS are disregarded in the case of LIN motions.</p>
<i>CP approximation</i>	<p>This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are:</p> <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>
#BASE, #TOOL	<ul style="list-style-type: none"> <li>■ #BASE Default setting. The coordinates of the end point refer to the BASE coordinate system.</li> <li>■ #TOOL The coordinates of the end point refer to the TOOL coordinate system.</li> </ul> <p>The specification of #BASE or #TOOL refers only to the corresponding LIN_REL statement. It has no effect on subsequent statements.</p>

**Example 1**

The TCP moves 100 mm in the X direction and 200 mm in the negative Z direction from the current position in the BASE coordinate system. Y, A, B, C and S remain constant. T is determined by the motion.

```
LIN_REL {X 100, Z -200}
```

**Example 2**

The TCP moves 100 mm from the current position in the negative X direction in the TOOL coordinate system. Y, Z, A, B, C and S remain constant. T is determined by the motion.

This example is suitable for moving the tool backwards against the tool direction. The precondition is that the tool direction has been calibrated along the X axis.

```
LIN_REL {X -100} #TOOL
```

## 10.5.5 CIRC

**Description** Executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion.

The coordinates of the auxiliary point and end point are absolute.

**Syntax** CIRC *Auxiliary point*, *End point*<, CA *Circular angle*> <*CP approximation*>

**Explanation of the syntax**

Element	Description
<i>Auxiliary point</i>	<p>Type: POS, E6POS, FRAME</p> <p>If not all components of the auxiliary point are specified, the controller takes the values of the previous position for the missing components.</p> <p>The orientation angles and the Status and Turn specifications for an auxiliary point are always disregarded.</p> <p>The auxiliary point cannot be approximated. The motion always stops exactly at this point.</p> <p>The coordinates refer to the BASE coordinate system.</p>
<i>End point</i>	<p>Type: POS, E6POS, FRAME</p> <p>If not all components of the end point are specified, the controller takes the values of the previous position for the missing components.</p> <p>The Status and Turn specifications for an end point of type POS or E6POS are ignored in the case of CIRC (and LIN) motions.</p> <p>The coordinates refer to the BASE coordinate system.</p>
<i>Circular angle</i>	<p>Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point.</p> <p>Unit: degrees. There is no limit; in particular, a circular angle greater than 360° can be programmed.</p> <ul style="list-style-type: none"> <li>■ Positive circular angle: the circular path is executed in the direction Start point → Auxiliary point → End point.</li> <li>■ Negative circular angle: the circular path is executed in the direction Start point → End point → Auxiliary point.</li> </ul>
<i>CP approximation</i>	<p>This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are:</p> <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>

**Example**

The end point of the circular motion is defined by a circular angle of 260°. The end point is approximated.

```
CIRC {X 5,Y 0, Z 9.2},{X 12.3,Y 0,Z -5.3,A 9.2,B -5,C 20}, CA 260
C_ORI
```

**10.5.6 CIRC\_REL****Description**

Executes a circular motion. An auxiliary point and an end point must be specified in order for the controller to be able to calculate the circular motion.

The coordinates of the auxiliary point and end point are relative to the current position.



A REL statement always refers to the current position of the robot. For this reason, if a REL motion is interrupted, the robot executes the entire REL motion again, starting from the position at which it was interrupted.

**Syntax**

`CIRC_REL Auxiliary point, End point<, CA Circular angle> <CP approximation>`

**Explanation of the syntax**

Element	Description
<i>Auxiliary point</i>	<p>Type: POS, E6POS, FRAME</p> <p>The auxiliary point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates refer to the BASE coordinate system.</p> <p>If \$ORI_TYPE, Status and/or Turn are specified, these specifications are ignored.</p> <p>If not all components of the auxiliary point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged.</p> <p>The orientation angles and the Status and Turn specifications for an auxiliary point are disregarded.</p> <p>The auxiliary point cannot be approximated. The motion always stops exactly at this point.</p>
<i>End point</i>	<p>Type: POS, E6POS, FRAME</p> <p>The end point must be specified in Cartesian coordinates. The controller interprets the coordinates as relative to the current position. The coordinates refer to the BASE coordinate system.</p> <p>If not all components of the end point are specified, the controller sets the value of the missing components to 0. In other words, the absolute values of these components remain unchanged.</p> <p>The Status and Turn specifications for an end point of type POS or E6POS are disregarded.</p>

Element	Description
<i>Circular angle</i>	<p>Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point.</p> <p>Unit: degrees. There is no limit; in particular, a circular angle <math>&gt; 360^\circ</math> can be programmed.</p> <ul style="list-style-type: none"> <li>■ Positive circular angle: the circular path is executed in the direction Start point → Auxiliary point → End point.</li> <li>■ Negative circular angle: the circular path is executed in the direction Start point → End point → Auxiliary point.</li> </ul>
<i>CP approximation</i>	<p>This parameter causes the end point to be approximated. It also defines the earliest point at which the approximate positioning can begin. The possible specifications are:</p> <ul style="list-style-type: none"> <li>■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS.</li> <li>■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI.</li> <li>■ C_VEL Velocity parameter: Approximation starts, at the earliest, when the velocity in the deceleration phase to the end point falls below the value of \$APO.CVEL.</li> </ul>

**Example**

The end point of the circular motion is defined by a circular angle of 500°. The end point is approximated.

```
CIRC_REL {X 100,Y 3.2,Z -20}, {Y 50}, CA 500 C_VEL
```

## 10.6 Motion programming: spline

### 10.6.1 Higher motion profile for spline motions

There are 2 motion profiles available for spline motions. The motion profile is selected using the system variable \$SPL\_VEL\_MODE.

**Overview****\$SPL\_VEL\_MODE:**

- \$SPL\_VEL\_MODE = #OPT: higher motion profile  
This motion profile is activated by default when the System Software is installed for the first time.
- \$SPL\_VEL\_MODE = #CART: conventional motion profile  
This motion profile is activated by default if a System Software version < 8.2 had previously been installed.

The higher motion profile enables the robot controller already to take axis-specific restrictions into consideration during path planning, and not wait until they are detected by monitoring functions during execution. All applications can generally be executed faster with the higher motion profile.

If a path is executed with the different motion profiles, the time response of the robot changes, but not the path in space.

**\$SPL\_VEL\_RESTR:**

If \$SPL\_VEL\_MODE = #OPT, it is possible to activate additional Cartesian restrictions by means of \$SPL\_VEL\_RESTR. These are:

- Maximum orientation velocity (component ORI\_VEL)
- Maximum Cartesian acceleration (component CART\_ACC)
- Maximum orientation acceleration (component ORI\_ACC)
- Maximum Cartesian jerk (component CART\_JERK)
- Maximum orientation jerk (component ORI\_JERK)
- Maximum robot axis acceleration (component ROB\_ACC)
- Maximum robot axis jerk (component ROB\_JERK)

The restrictions are deactivated by default. #ON activates them; #OFF deactivates them. Example:

```
$SPL_VEL_RESTR={ORI_VEL=#OFF, CART_ACC=#ON, ..., ROB_JERK=#OFF}
```

#### **Changes:**

Changing \$SPL\_VEL\_MODE or \$SPL\_VEL\_RESTR triggers an advance run stop.

\$SPL\_VEL\_MODE and \$SPL\_VEL\_RESTR can be changed as follows:

- Directly during program execution
- In the WITH line of a spline block
- In the WITH line of an individual spline motion
- By means of the variable correction function

It is not possible to change \$SPL\_VEL\_MODE or \$SPL\_VEL\_RESTR in the WITH line of a segment within a spline block.

## 10.6.2 SPLINE ... ENDSPLINE

<b>Description</b>	<p>A spline block may contain the following:</p> <ul style="list-style-type: none"> <li>■ Spline segments (only limited by the memory capacity.)</li> <li>■ PATH trigger</li> <li>■ 1 time block (&gt;&gt;&gt; 10.6.6 "TIME_BLOCK" Page 309)</li> <li>■ Comments</li> <li>■ Blank lines</li> </ul> <p>A spline block must not include any other instructions, e.g. variable assignments or logic statements. There must be no trigger between the last segment and ENDSPLINE.</p> <p>A spline block does not trigger an advance run stop.</p> <p>The following components must be specified for the first point in a spline block:</p> <ul style="list-style-type: none"> <li>■ X, Y, Z</li> <li>■ A, B, C</li> <li>■ E1 to E6 (if present)</li> </ul> <p>This also applies if the first point of the auxiliary point is a SCIRC segment.</p>
<b>Syntax</b>	<pre>SPLINE &lt;WITH SysVarSpline = Value1 &lt;,Value2 , ..., ValueN&gt; &gt;   Spline segment1   &lt;Trigger&gt;   ...   &lt;Spline segmentN-1</pre>

```
<Trigger>
<Spline segmentN>
ENDSPLINE <C_DIS>
```

**Explanation of the syntax**

Element	Description
SysVar-Spline	<p>The following system variables may be used:</p> <p>\$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$APO, \$BASE, \$CIRC_TYPE, \$IPO_MODE, \$GEAR_JERK, \$JERK, \$LOAD, \$ORI_TYPE, \$SPL_VEL_MODE, \$SPL_VEL_RESTR, \$TOOL, \$VEL, \$VEL_AXIS, \$VEL_EXTAX</p> <p>(&gt;&gt;&gt; 10.6.7 "System variables that affect the spline?" Page 313)</p>
Value	<p>Value assignment to the system variable. The value is valid for every segment in the spline block except segments to which a value is assigned separately.</p> <p>A value can also be assigned to the system variables \$ACC_EXTAX, \$GEAR_JERK, \$JERK, \$VEL and \$VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger.</p> <p>(&gt;&gt;&gt; 9.3.4.14 "Limits for functions in the spline trigger" Page 272)</p>
C_DIS	<p>Approximate the end point of the spline block (= last point in the block).</p> <p>\$APO is used to define the earliest point at which the approximate positioning can begin.</p>

**Example**

```
SPLINE
SPL P1
TRIGGER WHEN PATH=GET_PATH() ONSTART DELAY=0 DO <subprog> PRIO=-1
SPL P2
SLIN P3
SPL P4
SCIRC P5, P6 WITH $VEL.CP=0.2
SPL P7 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
SCIRC P8, P9
SPL P10
ENDSPLINE
```

### 10.6.3 SLIN

**Description**

SLIN can be programmed as an individual block or as a segment in a spline block. It is possible to copy an individual SLIN motion into a spline block, but only if it does not contain an assignment to system variables that are not permissible there.

**Syntax**

```
SLIN End point <WITH SysVarSlin = Value1 <, Value2 , ... , ValueN>> <C_DIS>
```

**Explanation of  
the syntax**

Element	Description
End point	<p>Type: POS, E6POS, FRAME</p> <p>The coordinates refer to the BASE coordinate system.</p> <ul style="list-style-type: none"> <li>■ The following applies for spline segments: If not all components of the end point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely.</li> <li>■ The following applies for individual motions: If not all components of the end point are specified, the controller takes the values of the previous position for the missing components. If the previous position is the end point of a circle with a circular angle, the values of the programmed end point are applied, not the values resulting from the circular angle. If no previous position is known (in the case of a block selection), the missing components are taken from the current robot position.</li> </ul>
SysVarSlin	<p>The following system variables may be used:</p> <p>\$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$GEAR_JERK, \$JERK, \$ORI_TYPE, \$VEL, \$VEL_AXIS, \$VEL_EXTAX</p> <ul style="list-style-type: none"> <li>■ The following may also be used for SLIN in the spline block: \$EX_AX_IGNORE</li> <li>■ The following may also be used for individual SLIN motions: \$APO, \$BASE, \$IPO_MODE, \$LOAD, \$SPL_VEL_MODE, \$SPL_VEL_RESTR, \$TOOL</li> </ul> <p>(&gt;&gt;&gt; 10.6.7 "System variables that affect the spline?" Page 313)</p>
Value	<p>Value assignment to the system variable.</p> <p>In the case of SLIN segments: The assignment applies only for this segment. It has no effect on subsequent segments.</p> <p>A value can also be assigned to the system variables \$ACC_EXTAX, \$GEAR_JERK, \$JERK, \$VEL and \$VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger.</p> <p>(&gt;&gt;&gt; 9.3.4.14 "Limits for functions in the spline trigger" Page 272)</p>
C_DIS	<p>Only possible for individual SLIN motions: Approximate the end point.</p> <p>\$APO is used to define the earliest point at which the approximate positioning can begin.</p>

**10.6.4 SCIRC**
**Description**

SCIRC can be programmed as an individual block or as a segment in a spline block. It is possible to copy an individual SCIRC motion into a spline block, but only if it does not contain an assignment to system variables that are not permissible there.

**Syntax**

SCIRC *Auxiliary point*, *End point* <, CA *Auxiliary point*> <WITH *SysVarScirc* = *Value1* <, *Value2* , ..., *ValueN*>> <*C\_DIS*>

**Explanation of  
the syntax**

Element	Description
Auxiliary point End point	<p>Type: POS, E6POS, FRAME</p> <p>The coordinates refer to the BASE coordinate system.</p> <ul style="list-style-type: none"> <li>■ The following applies for spline segments: If not all components of the point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely.</li> <li>■ The following applies for individual motions: If not all components of the point are specified, the controller takes the values of the previous position for the missing components. If the previous position is the end point of a circle with a circular angle, the values of the programmed end point are applied, not the values resulting from the circular angle. If no previous position is known (in the case of a block selection), the missing components are taken from the current robot position.</li> </ul>
Circular angle	<p>Specifies the overall angle of the circular motion. This makes it possible to extend the motion beyond the programmed end point or to shorten it. The actual end point thus no longer corresponds to the programmed end point.</p> <p>Unit: degrees. There is no limit; in particular, a circular angle greater than 360° can be programmed.</p> <ul style="list-style-type: none"> <li>■ Positive circular angle: the circular path is executed in the direction Start point → Auxiliary point → End point.</li> <li>■ Negative circular angle: the circular path is executed in the direction Start point → End point → Auxiliary point.</li> </ul>
SysVarScirc	<p>The following system variables may be used:</p> <p>\$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$CIRC_MODE, \$CIRC_TYPE, \$GEAR_JERK, \$JERK, \$ORI_TYPE, \$VEL, \$VEL_AXIS, \$VEL_EXTAX</p> <ul style="list-style-type: none"> <li>■ The following may also be used for SCIRC in the spline block: \$EX_AX_IGNORE</li> <li>■ The following may also be used for individual SCIRC motions: \$APO, \$BASE, \$IPO_MODE, \$LOAD, \$SPL_VEL_MODE, \$SPL_VEL_RESTR, \$TOOL</li> </ul> <p>(&gt;&gt;&gt; 10.6.7 "System variables that affect the spline?" Page 313)</p>
Value	<p>Value assignment to the system variable.</p> <p>In the case of SCIRC segments: The assignment applies only for this segment. It has no effect on subsequent segments.</p> <p>A value can also be assigned to the system variables \$ACC_EXTAX, \$GEAR_JERK, \$JERK, \$VEL and \$VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger.</p> <p>(&gt;&gt;&gt; 9.3.4.14 "Limits for functions in the spline trigger" Page 272)</p>
C_DIS	<p>Only possible for individual SCIRC motions: Approximate the end point.</p> <p>\$APO is used to define the earliest point at which the approximate positioning can begin.</p>

**Example**

```
SCIRC P2, P3 WITH $CIRC_TYPE=#PATH
```

**10.6.5 SPL**

**Description** SPL can only be programmed as a segment in a spline block. It is not possible to program an individual SPL motion.

**Syntax** SPL *End point* <WITH *SysVarSpl* = *Value1* <, *Value2* , ..., *ValueN*>>

**Explanation of the syntax**

Element	Description
End point	Type: POS, E6POS, FRAME  The coordinates refer to the BASE coordinate system.  If not all components of the end point are specified, the controller takes the values of the previous segment for the missing components. The first segment in the spline block must be specified completely.
SysVarSpl	The following system variables may be used:  \$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$CIRC_TYPE, \$EX_AX_IGNORE, \$GEAR_JERK, \$JERK, \$ORI_TYPE, \$VEL, \$VEL_AXIS, \$VEL_EXTAX  (>>> 10.6.7 "System variables that affect the spline?" Page 313)
Value	Value assignment to the system variable. The assignment applies only for this segment. It has no effect on subsequent segments.  A value can also be assigned to the system variables \$ACC_EXTAX, \$GEAR_JERK, \$JERK, \$VEL and \$VEL_EXTAX via a function. The same restrictions apply to these functions as to functions in the spline trigger.  (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)

**Example**

```
SPL P4 WITH $ACC=(CP 2.0, ORI1 200, ORI2 200)
```

**10.6.6 TIME\_BLOCK**

**Description** TIME\_BLOCK can be used to execute a spline block or part of one in a defined time. It is also possible to allocate time components to areas of TIME\_BLOCK.

Points can be modified in, added to or removed from the spline block without changing the time specifications. This enables the user to correct the Cartesian path and retain the existing timing.

A spline block may include 1 time block, i.e. 1 statement of the type TIME\_BLOCK START ... TIME\_BLOCK END. This, in turn, may contain any number of TIME\_BLOCK PART statements. The time block can only be used in spline blocks.

**Syntax**

SPLINE

<*Spline segment...*>

...

**TIME\_BLOCK START**

<*Spline segment...*

```
...
<TIME_BLOCK PART = Component[1]>
...
Spline segmentN
...
<TIME_BLOCK PART = Component[N]>
TIME_BLOCK END = Overall time
<Spline segmentN+1>
...
ENDSPLINE
```

#### Explanation of the syntax

It is not essential for there to be spline segments before TIME\_BLOCK START and after TIME\_BLOCK END. It is nonetheless advisable to program as follows:

- There is at least 1 spline segment between SPLINE and TIME\_BLOCK START.
- There is at least 1 spline segment between TIME\_BLOCK END and END-SPLINE.

Advantages:

- The programmed overall time is maintained exactly even in the case of approximate positioning.
- In the case of a block selection, the original velocity profile is maintained if there is at least 1 spline segment between SPLINE and TIME\_BLOCK START.
- Segments before TIME\_BLOCK START make it possible to accelerate to the desired velocity.

Element	Description
Component[x]	<p>Type: INT or REAL; constant, variable or function</p> <p>Desired time component [x] for the following distance:</p> <ul style="list-style-type: none"> <li>■ From the point before TIME_BLOCK PART = Component[x-1] to the point before TIME_BLOCK PART = Component[x]</li> <li>■ If TIME_BLOCK PART = Component[x-1] does not exist: From the point before TIME_BLOCK START to the point before TIME_BLOCK PART = Component[x]</li> </ul> <p>The time components are maintained as accurately as possible by the robot controller. Generally, however, they are not maintained exactly.</p> <p>The user can assign the components in such a way that they add up to 100. The components can then be considered as percentages. The components do not have to add up to 100, however, and can have any sum! The robot controller always equates <i>Overall time</i> to the sum of the components. This allows the components to be used very flexibly and also changed.</p> <p>If time components are assigned, there must always be a TIME_BLOCK PART before TIME_BLOCK END. There must be no segments in between.</p>
Overall time	<p>Type: INT or REAL; constant, variable or function; unit: s</p> <p>Time in which the following distance is traveled:</p> <ul style="list-style-type: none"> <li>■ From the point before TIME_BLOCK START to the point before TIME_BLOCK END</li> </ul> <p>The value must be greater than 0. The overall time is maintained exactly. If this time cannot possibly be maintained, e.g. because too short a time has been programmed, the robot executes the motion in the fastest possible time. In T1 and T2, a message is also displayed.</p>

If the value for *Component[x]* or *Overall time* is assigned via a function, the same restrictions apply as for the functions in the spline trigger.

(>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)

## Example

```

SPLINE
  SLIN P1
  SPL P2
  TIME_BLOCK START
    SLIN P3
  TIME_BLOCK PART = 12.7
    SPL P4
    SPL P5
    SPL P6
  TIME_BLOCK PART = 56.4
    SCIRC P7, P8
    SPL P9
  TIME_BLOCK PART = 27.8
  TIME_BLOCK END = 3.9
  SLIN P10
ENDSPLINE

```

Points P2 to P9 are executed exactly in the programmed time of 3.9 s.

The robot controller equates the overall time of 3.9 s to the sum of the components, i.e. 96.9. This gives the value of the components in seconds.

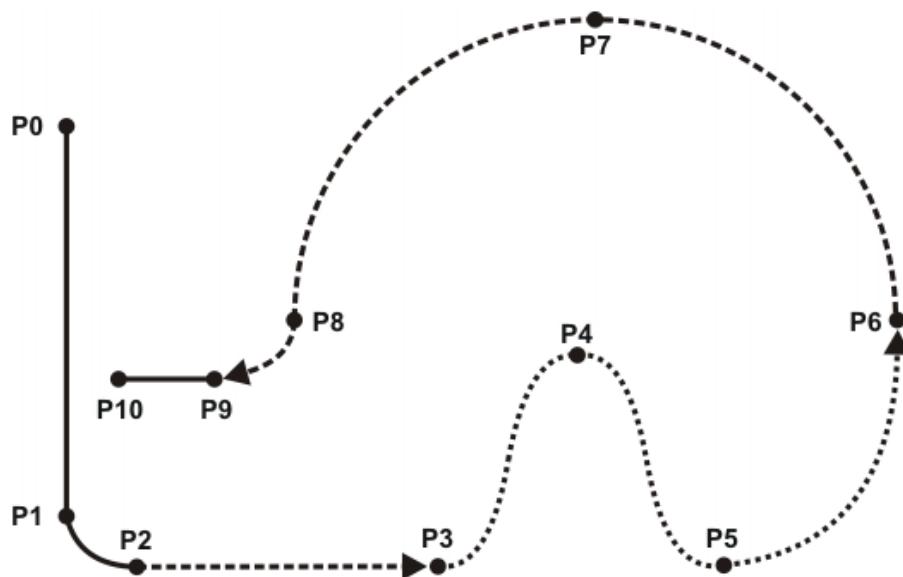


Fig. 10-1: Example TIME\_BLOCK

#### Block selection



If \$SPL\_VEL\_MODE = #OPT: in the case of a block selection, the robot controller ignores the time block and generates the following message: **Time block ignored due to BCO run..**

In the case of a block selection, the original velocity profile is maintained if there is at least 1 spline segment between SPLINE and TIME\_BLOCK START.

(A block selection to TIME\_BLOCK is the equivalent of a block selection to the subsequent motion block.)

#### Example: Block selection with segment before TIME\_BLOCK START

```
LIN P1
SPLINE
SPL P2
TIME_BLOCK START
SLIN P3
TIME_BLOCK PART = 0.5
SPL P4
SPL P5
SPL P6
TIME_BLOCK PART = 2.2
TIME_BLOCK END = 2.7
...
```

- The defined overall time applies from the point before TIME\_BLOCK START, i.e. P2.
- A block selection is now made to P5.
- For continued motion from P5, the robot controller plans the velocity for the entire spline block.

The first point in the spline block is P2. The velocity is thus planned from the same point as if the spline were to be executed during normal program execution (= from P1).

#### Example: Block selection without segment before TIME\_BLOCK START

```

LIN P1
SPLINE
  TIME_BLOCK START
    SLIN P2
  TIME_BLOCK PART = 0.5
    SPL P3
    SPL P4
    SPL P5
  TIME_BLOCK PART = 2.2
  TIME_BLOCK END = 2.7
...

```

- The defined overall time applies from the point before TIME\_BLOCK START, i.e. P1.
- A block selection is now made to P5.
- For continued motion from P5, the robot controller plans the velocity for the entire spline block.

The first point in the spline block is P2. P1 is situated before the spline block and can thus no longer be taken into consideration by the robot controller.

It is thus not possible to plan the velocity for the spline block as exactly as if it were to be executed during normal program execution (= from P1).

#### **System variable**

The data of the time-based spline can be read via the system variable \$PATH-TIME. \$PATHTIME is filled with the data as soon as the robot controller has completed the planning of the spline block. The data are retained until the next spline block has been planned.

\$PATHTIME is a structure and consists of the following components:

Component	Description
REAL \$PATHTIME.TOTAL	Overall time actually required (s)
REAL \$PATHTIME.SCHEDULED	Overall time planned (s)
REAL \$PATHTIME.PROGRAMMED	Overall time programmed (s)
INT \$PATHTIME.N_SECTIONS	Number N of time components
REAL \$PATHTIME.MAX_DEV	Maximum deviation of all time components between the programmed time and the planned time (%)
INT \$PATH-TIME.MAX_DEV_SECTION	Number of the time component with the greatest deviation between the programmed time and the planned time

#### **10.6.7 System variables that affect the spline?**

In the case of some system variables that can be set in WITH lines with splines, it depends on the state of \$SPL\_VEL\_MODE whether or not the system variable has an effect on spline motions.

System variable	\$SPL_VEL_MODE=#OPT	\$SPL_VEL_MODE=#CART
\$ACC.CP	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$ACC.ORI1	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$ACC_AXIS[]	Yes	No
\$GEAR_JERK	Yes	No
\$JERK.AX.E1 ... \$JERK.AX.E6	Yes	No

System variable	\$SPL_VEL_MODE=#OPT	\$SPL_VEL_MODE=#CART
\$JERK.AX.A1 ... \$JERK.AX.A6	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$JERK.CP	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$JERK.ORI	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$SPL_VEL_RESTR	Yes	No
\$VEL.CP	Yes	Yes
\$VEL.ORI1	By default no, but can be activated using \$SPL_VEL_RESTR	Yes
\$VEL_AXIS[]	Yes	No

## 10.7 Program execution control

### 10.7.1 CONTINUE

**Description** Prevents an advance run stop that would otherwise occur in the following program line.



CONTINUE always applies to the following program line, even if this is a blank line!

#### Syntax

CONTINUE

#### Example

Preventing both advance run stops:

```
CONTINUE
$OUT[1]=TRUE
CONTINUE
$OUT[2]=FALSE
```



In this case, the outputs are set in the advance run. The exact point at which they are set cannot be foreseen.

### 10.7.2 EXIT

**Description** Exit from a loop. The program is then continued after the loop. EXIT may be used in any loop.

#### Syntax

EXIT

#### Example

The loop is exited when \$IN[1] is set to TRUE. The program is then continued after ENDLOOP.

```
DEF EXIT_PROG()
PTP HOME
LOOP
    PTP POS_1
    PTP POS_2
    IF $IN[1] == TRUE THEN
        EXIT
    ENDIF
    CIRC HELP_1, POS_3
    PTP POS_4
ENDLOOP
PTP HOME
END
```

### 10.7.3 FOR ... TO ... ENDFOR

**Description** A statement block is repeated until a counter exceeds or falls below a defined value.

After the last execution of the statement block, the program is resumed with the first statement after ENDFOR. The loop execution can be exited prematurely with EXIT.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

**Syntax**

```
FOR Counter = Start TO End <STEP Increment>
<Statements>
ENDFOR
```

**Explanation of the syntax**

Element	Description
<i>Counter</i>	Type: INT  Variable that counts the number of times the loop has been executed. The preset value is <i>Start</i> . The variable must first be declared.  The value of <i>Counter</i> can be used in statements inside and outside of the loop. Once the loop has been exited, <i>Counter</i> retains its most recent value.
<i>Start; End</i>	Type: INT  <i>Counter</i> must be preset to the value <i>Start</i> . Each time the loop is executed, the value of <i>Counter</i> is automatically increased by the increment. If the value exceeds or falls below the <i>End</i> value, the loop is terminated.
<i>Increment</i>	Type: INT  Value by which <i>Counter</i> is changed every time the loop is executed. The value may be negative. Default value: 1.  <ul style="list-style-type: none"> <li>■ Positive value: the loop is ended if <i>Counter</i> is greater than <i>End</i>.</li> <li>■ Negative value: the loop is ended if <i>Counter</i> is less than <i>End</i>.</li> </ul> The value may not be either zero or a variable.

**Example**

The variable *B* is incremented by 1 after each of 5 times the loop is executed.

```
INT A
...
FOR A=1 TO 10 STEP 2
    B=B+1
ENDFOR
```

### 10.7.4 GOTO

**Description** Unconditional jump to a specified position in the program. Program execution is resumed at this position.

The destination must be in the same subprogram or function as the GOTO statement.

The following jumps are not possible:

- Into an IF statement from outside.
- Into a loop from outside.

- From one CASE statement to another CASE statement.



GOTO statements lead to a loss of structural clarity within a program. It is better to work with IF, SWITCH or a loop instead.

## Syntax

`GOTO Label`

...

*Label:*

## Explanation of the syntax

Element	Description
<i>Label</i>	Position to which a jump is made. At the destination position, <i>Label</i> must be followed by a colon.

## Example 1

Unconditional jump to the program position GLUESTOP.

```
GOTO GLUESTOP
...
GLUESTOP:
```

## Example 2

Unconditional jump from an IF statement to the program position END.

```
IF X>100 THEN
  GOTO ENDE
ELSE
  X=X+1
ENDIF
A=A*X
...
ENDE:
END
```

## 10.7.5 HALT

### Description

Stops the program. The last motion instruction to be executed will, however, be completed.

Execution of the program can only be resumed using the Start key. The next instruction after HALT is then executed.



In an interrupt program, program execution is only stopped after the advance run has been completely executed.

## Syntax

`HALT`

## 10.7.6 IF ... THEN ... ENDIF

### Description

Conditional branch. Depending on a condition, either the first statement block (THEN block) or the second statement block (ELSE block) is executed. The program is then continued after ENDIF.

The ELSE block may be omitted. If the condition is not satisfied, the program is then continued at the position immediately after ENDIF.

There is no limit on the number of statements contained in the statement blocks. Several IF statements can be nested in each other.

## Syntax

`IF Condition THEN`

*Statements*

`<ELSE`

*Statements>*

ENDIF

### Explanation of the syntax

Element	Description
<i>Condition</i>	Type: BOOL  Possible: <ul style="list-style-type: none"> <li>■ Variable of type BOOL</li> <li>■ Function of type BOOL</li> <li>■ Logic operation, e.g. a comparison, with a result of type BOOL</li> </ul>

#### Example 1

IF statement without ELSE

```
IF A==17 THEN
    B=1
ENDIF
```

#### Example 2

IF statement with ELSE

```
IF $IN[1]==TRUE THEN
    $OUT[17]=TRUE
ELSE
    $OUT[17]=FALSE
ENDIF
```

### 10.7.7 LOOP ... ENDLOOP

#### Description

Loop that endlessly repeats a statement block. The loop execution can be exited with EXIT.  
  
Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

#### Syntax

LOOP

*Statements*

ENDLOOP

#### Example

The loop is executed until input \$IN[30] is set to true.

```
LOOP
    LIN P_1
    LIN P_2
    IF $IN[30]==TRUE THEN
        EXIT
    ENDIF
ENDLOOP
```

### 10.7.8 REPEAT ... UNTIL

#### Description

Non-rejecting loop. Loop that is repeated until a certain condition is fulfilled.

The statement block is executed at least once. The condition is checked after each loop execution. If the condition is met, program execution is resumed at the first statement after the UNTIL line.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

#### Syntax

REPEAT

*Statements*

`UNTIL Termination condition`

#### Explanation of the syntax

Element	Description
<i>Termination condition</i>	<p>Type: BOOL</p> <p>Possible:</p> <ul style="list-style-type: none"> <li>■ Variable of type BOOL</li> <li>■ Function of type BOOL</li> <li>■ Logic operation, e.g. a comparison, with a result of type BOOL</li> </ul>

#### Example 1

The loop is to be executed until \$IN[1] is true.

```
R=1
REPEAT
    R=R+1
UNTIL $IN[1]==TRUE
```

#### Example 2

The loop is executed once, even though the termination condition is already fulfilled before the loop execution, because the termination condition is not checked until the end of the loop. After execution of the loop, R has the value 102.

```
R=101
REPEAT
    R=R+1
UNTIL R>100
```

### 10.7.9 SWITCH ... CASE ... ENDSWITCH

#### Description

Selects one of several possible statement blocks, according to a selection criterion. Every statement block has at least one identifier. The block whose identifier matches the selection criterion is selected.

Once the block has been executed, the program is resumed after ENDSWITCH.

If no identifier agrees with the selection criterion, the DEFAULT block is executed. If there is no DEFAULT block, no block is executed and the program is resumed after ENDSWITCH.



The SWITCH statement cannot be prematurely exited using EXIT.

#### Syntax

```
SWITCH Selection criterion
CASE Identifier1 <, Identifier2, . . . >
Statement block
<CASE IdentifierM <, IdentifierN, . . . >
Statement block >
<DEFAULT
Default statement block>
ENDSWITCH
```

There must be no blank line or comment between the SWITCH line and the first CASE line. DEFAULT may only occur once in a SWITCH statement.

**Explanation of the syntax**

Element	Description
<i>Selection criterion</i>	Type: INT, CHAR, ENUM This can be a variable, a function call or an expression of the specified data type.
<i>Identifier</i>	Type: INT, CHAR, ENUM The data type of the identifier must match the data type of the selection criterion. A statement block can have any number of identifiers. Multiple block identifiers must be separated from each other by a comma.

**Example 1**

Selection criterion and identifier are of type INT.

```
INT VERSION
...
SWITCH VERSION
    CASE 1
        UP_1()
    CASE 2,3
        UP_2()
        UP_3()
        UP_3A()
    DEFAULT
        ERROR_UP()
ENDSWITCH
```

**Example 2**

Selection criterion and identifier are of type CHAR. The statement `SP_5()` is never executed here because the identifier `C` has already been used.

```
SWITCH NAME
    CASE "A"
        UP_1()
    CASE "B", "C"
        UP_2()
        UP_3()
    CASE "C"
        UP_5()
ENDSWITCH
```

### 10.7.10 WAIT FOR

**Description**

Stops the program until a specified condition is fulfilled. Program execution is then resumed.



If, due to incorrect formulation, the expression can never take the value TRUE, the compiler does not recognize this. In this case, execution of the program will be permanently halted because the program is waiting for a condition that cannot be fulfilled.

**Syntax**

`WAIT FOR Condition`

**Explanation of the syntax**

Element	Description
<i>Condition</i>	Type: BOOL Condition, the fulfillment of which allows program execution to be resumed. <ul style="list-style-type: none"> <li>■ If the condition is already TRUE when WAIT is called, program execution is not halted.</li> <li>■ If the condition is FALSE, program execution is stopped until the condition is TRUE.</li> </ul>

**Example**

Interruption of program execution until \$IN[17] is TRUE:

```
WAIT FOR $IN[17]
```

Interruption of program execution until BIT1 is FALSE:

```
WAIT FOR BIT1==FALSE
```

**10.7.11 WAIT SEC****Description**

Halts execution of the program and continues it after a wait time. The wait time is specified in seconds.

**Syntax**

```
WAIT SEC Wait time
```

**Explanation of the syntax**

Element	Description
<i>Wait time</i>	Type: INT, REAL  Number of seconds for which program execution is to be interrupted. If the value is negative, the program does not wait. With small wait times, the accuracy is determined by a multiple of 12 ms.

**Example**

Interruption of program execution for 17.156 seconds:

```
WAIT SEC 17.156
```

Interruption of program execution in accordance with the variable value of V\_WAIT in seconds:

```
WAIT SEC V_ZEIT
```

**10.7.12 WHILE ... ENDWHILE****Description**

Rejecting loop. Loop that is repeated as long as a certain condition is fulfilled.

If the condition is not met, program execution is resumed at the first statement after the ENDWHILE line. The condition is checked before each loop execution. If the condition is not already fulfilled beforehand, the statement block is not executed.

Loops can be nested. In the case of nested loops, the outer loop is executed completely first. The inner loop is then executed completely.

**Syntax**

```
WHILE Repetition condition
```

```
Statement block
```

```
ENDWHILE
```

**Explanation of the syntax**

Element	Description
<i>Repetition condition</i>	Type: BOOL  Possible: <ul style="list-style-type: none"><li>■ Variable of type BOOL</li><li>■ Function of type BOOL</li><li>■ Logic operation, e.g. a comparison, with a result of type BOOL</li></ul>

**Example 1**

The loop is executed 99 times. After execution of the loop, W has the value 100.

```

W=1
WHILE W<100
    W=W+1
ENDWHILE

```

**Example 2**

The loop is executed as long as \$IN[1] is true.

```

WHILE $IN[1]==TRUE
    W=W+1
ENDWHILE

```

## 10.8 Inputs/outputs

### 10.8.1 ANIN

**Description** Cyclical reading (every 12 ms) of an analog input.  
ANIN triggers an advance run stop.

**Syntax** Starting cyclical reading:

ANIN ON *Value* = *Factor* \* *Signal name* \* <*#Offset*>

Ending cyclical reading:

ANIN OFF *Signal name*



The robot controller has 32 analog inputs (\$ANIN[1] ... \$ANIN[32]).

■ A maximum of three ANIN ON statements can be used at the same time.

- A maximum of two ANIN ON statements can use the same variable *Value* or access the same analog input.
- All of the variables used in an ANIN statement must be declared in data lists (locally or in \$CONFIG.DAT).

**Explanation of the syntax**

Element	Description
<i>Value</i>	Type: REAL  The result of the cyclical reading is stored in <i>Value</i> . <i>Value</i> can be a variable or a signal name for an output.
<i>Factor</i>	Type: REAL  Any factor. It can be a constant, variable or signal name.
<i>Signal name</i>	Type: REAL  Specifies the analog input. <i>Signal name</i> must first have been declared with SIGNAL . It is not possible to specify the analog input \$ANIN[x] directly instead of the signal name.  The values of an analog input \$ANIN[x] range between +1.0 and -1.0 and represent a voltage of +10 V to -10 V.
<i>Offset</i>	Type: REAL  It can be a constant, variable or signal name.

**Example**

In this example, the program override (= system variable \$OV\_PRO) is defined by means of the analog input \$ANIN[1].

\$ANIN[1] must first be linked to a freely selected signal name, in this case SIGNAL\_1, in the declaration section.

```

SIGNAL SIGNAL_1 $ANIN[1]
...
ANIN ON $OV_PRO = 1.0 * SIGNAL_1

```

The cyclical scanning of `SIGNAL_1` is ended using the `ANIN OFF` statement.

```
ANIN OFF SIGNAL_1
```

### 10.8.2 ANOUT

<b>Description</b>	Cyclical writing (every 12 ms) to an analog output. ANOUT triggers an advance run stop.
<b>Syntax</b>	<p>Starting cyclical writing:</p> <pre>ANOUT ON Signal name = Factor * Control element &lt;±Offset&gt; &lt;DELAY = ±Time&gt; &lt;MINIMUM = Minimum value&gt; &lt;MAXIMUM = Maximum value&gt;</pre> <p>Ending cyclical writing:</p> <pre>ANOUT OFF Signal name</pre>



The robot controller has 32 analog outputs (\$ANOUT[1] ... \$ANOUT[32]).

- A maximum of four ANOUT ON statements can be used at the same time.
- All of the variables used in an ANOUT statement must be declared in data lists (locally or in \$CONFIG.DAT).

#### Explanation of the syntax

Element	Description
<i>Signal name</i>	Type: REAL  Specifies the analog output. <i>Signal name</i> must first have been declared with SIGNAL . It is not possible to specify the analog output \$ANOUT[x] directly instead of the signal name.  The values of an analog output \$ANOUT[x] range between +1.0 and -1.0 and represent a voltage of +10 V to -10 V.
<i>Factor</i>	Type: REAL  Any factor. It can be a constant, variable or signal name.
<i>Control element</i>	Type: REAL  It can be a constant, variable or signal name.
<i>Offset</i>	Type: REAL  It can be a constant, variable or signal name.
<i>Time</i>	Type: REAL  Unit: seconds. By using the keyword DELAY and entering a positive or negative amount of time, the output signal can be delayed (+) or set early (-).
<i>Minimum value, Maximum value</i>	Type: REAL  Minimum and/or maximum voltage to be present at the output. The actual value does not fall below/exceed these values, even if the calculated values fall outside this range.  Permissible values: -1.0 to +1.0 (corresponds to -10 V to +10 V).  It can be a constant, variable, structure component or array element. The minimum value must always be less than the maximum value. The sequence of the keywords MINIMUM and MAXIMUM must be observed.

**Example**

In this example, the output \$ANOUT[5] controls the adhesive output.

A freely selected name, in this case GLUE, is assigned to the analog output in the declaration section. The amount of adhesive is to be dependent on the current path velocity (= system variable \$VEL\_ACT). Furthermore, the output signal is to be generated 0.5 seconds early. The minimum voltage is to be 3 V.

```
SIGNAL GLUE $ANOUT[5]
...
ANOUT ON GLUE = 0.5 * $VEL_ACT DELAY=-0.5 MINIMUM=0.30
```

The cyclical analog output is ended by using ANOUT OFF:

```
ANOUT OFF GLUE
```

**10.8.3 PULSE****Description**

Sets a pulse. The output is set to a defined level for a specified duration. The output is then reset automatically by the system. The output is set and reset irrespective of the previous level of the output.

At any one time, pulses may be set at a maximum of 16 outputs.

If PULSE is programmed before the first motion block, the pulse duration also elapses if the Start key is released again and the robot has not yet reached the BCO position.

The PULSE statement triggers an advance run stop. It is only executed concurrently with robot motion if it is used in a TRIGGER statement.



The pulse is not terminated in the event of an EMERGENCY STOP, an operator stop or an error stop!

**Syntax**

PULSE (*Signal*, *Level*, *Pulse duration*)

**Explanation of the syntax**

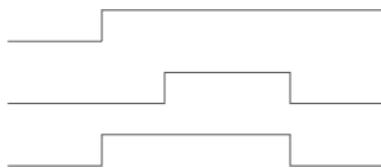
Element	Description
<i>Signal</i>	Type: BOOL  Output to which the pulse is to be fed. The following are permitted: <ul style="list-style-type: none"><li>■ OUT[No]</li><li>■ Signal variable</li></ul>
<i>Level</i>	Type: BOOL  Logic expression: <ul style="list-style-type: none"><li>■ TRUE represents a positive pulse (high).</li><li>■ FALSE represents a negative pulse (low).</li></ul>
<i>Pulse duration</i>	Type: REAL  Range of values: 0.1 to 3.0 seconds. Pulse durations outside this range trigger a program stop.  Pulse interval: 0.1 seconds, i.e. the pulse duration is rounded up or down. The PULSE statement is executed in the controller at the low-priority clock rate. This results in a tolerance in the order of the pulse interval (0.1 seconds). The time deviation is about 1% - 2% on average. The deviation is about 13% for very short pulses.

**\$OUT+PULSE**

If an output is already set before the pulse, it will be reset by the falling edge of the pulse.

```
$OUT[50] = TRUE  
PULSE ($OUT[50],TRUE,0.5)
```

Actual pulse characteristic at output 50:

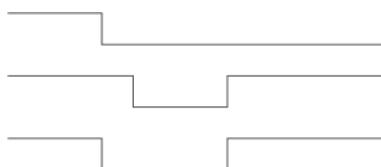


**Fig. 10-2: \$OUT+PULSE, example 1**

If a negative pulse is applied to an output that is set to Low, the output remains Low until the end of the pulse and is then set to High:

```
$OUT[50] = FALSE  
PULSE ($OUT[50],FALSE,0.5)
```

Actual pulse characteristic at output 50:



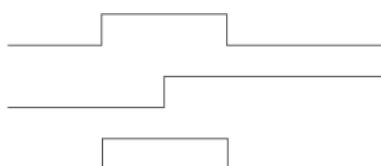
**Fig. 10-3: \$OUT+PULSE, example 2**

#### PULSE+\$OUT

If the same output is set during the pulse duration, it will be reset by the falling edge of the pulse.

```
PULSE ($OUT[50],TRUE,0.5)  
$OUT[50] = TRUE
```

Actual pulse characteristic at output 50:

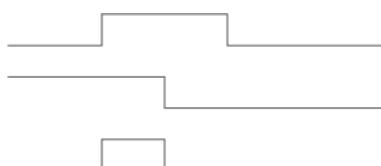


**Fig. 10-4: PULSE+\$OUT, example 1**

If the output is reset during the pulse duration, the pulse duration is reduced accordingly:

```
PULSE ($OUT[50],TRUE,0.5)  
$OUT[50] = FALSE
```

Actual pulse characteristic at output 50:

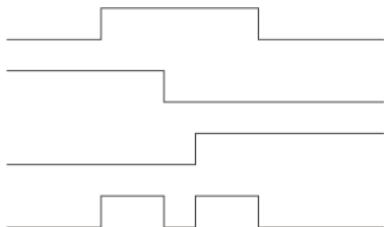


**Fig. 10-5: PULSE+\$OUT, example 2**

If an output is set to FALSE during a pulse and then back to TRUE, the pulse is interrupted and then resumed when the output is set to TRUE. The overall duration from the first rising edge to the last falling edge (i.e. including the duration of the interruption) corresponds to the duration specified in the PULSE statement.

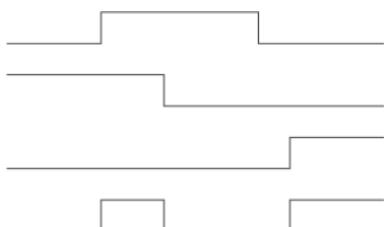
```
PULSE ($OUT[50], TRUE, 0.8)
$OUT[50]=FALSE
$OUT[50]=TRUE
```

Actual pulse characteristic at output 50:



**Fig. 10-6: PULSE+\$OUT, example 3**

The actual pulse characteristic is only specified as above if \$OUT[x]=TRUE is set during the pulse. If \$OUT[x]=TRUE is not set until after the pulse (see line 3), then the actual pulse characteristic is as follows (line 4):



**Fig. 10-7: PULSE+\$OUT, example 4**

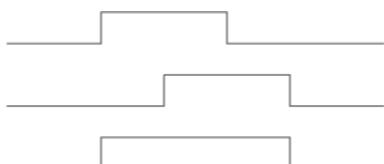
#### PULSE+PULSE

If several PULSE statements overlap, it is always the last PULSE statement that determines the end of the overall pulse duration.

If a pulse is activated again before the falling edge, the duration of the second pulse starts at this moment. The overall pulse duration is thus shorter than the sum of the values of the first and second pulses:

```
PULSE ($OUT[50], TRUE, 0.5)
PULSE ($OUT[50], TRUE, 0.5)
```

Actual pulse characteristic at output 50:

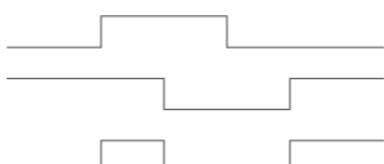


**Fig. 10-8: PULSE+PULSE, example 1**

If, during the pulse duration of a positive pulse, a negative pulse is sent to the same output, only the second pulse is taken into consideration from this moment onwards:

```
PULSE ($OUT[50], TRUE, 0.5)
PULSE ($OUT[50], FALSE, 0.5)
```

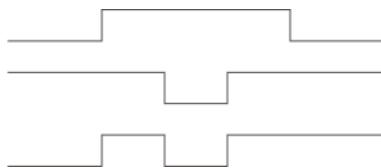
Actual pulse characteristic at output 50:



**Fig. 10-9: PULSE+PULSE, example 2**

```
PULSE ($OUT[50], TRUE, 3.0)
PULSE ($OUT[50], FALSE, 1.0)
```

Actual pulse characteristic at output 50:



**Fig. 10-10: PULSE+PULSE, example 3**

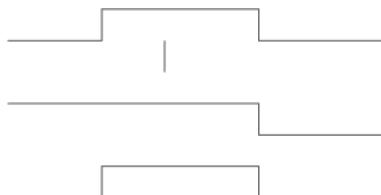
#### PULSE+END

If a pulse is programmed before the END statement, the duration of program execution is increased accordingly.

```
PULSE ($OUT[50], TRUE, 0.8)
END
```

Program active

Actual pulse characteristic at output 50:



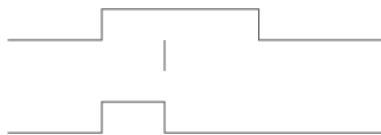
**Fig. 10-11: PULSE+END, example**

#### PULSE+RESET/ CANCEL

If program execution is reset (RESET) or aborted (CANCEL) while a pulse is active, the pulse is immediately reset:

```
PULSE ($OUT[50], TRUE, 0.8)
RESET OR CANCEL
```

Actual pulse characteristic at output 50:



**Fig. 10-12: PULSE+RESET, example**

### 10.8.4 SIGNAL

#### Description

SIGNAL declarations must appear in the declaration section.

- SIGNAL links predefined signal variables for inputs or outputs with a name.  
A SIGNAL declaration is required in order to be able to address an analog input or output. An input or output may appear in several SIGNAL declarations.
- SIGNAL declarations that are predefined in the system can be deactivated by means of SIGNAL in conjunction with the keyword FALSE.  
Can only be used in KRC:\STEU:\MADA:\\$machine.dat.

#### Syntax

Declaration of signal names for inputs and outputs:

```
SIGNAL Signal name Signal variable <TO Signal variable>
```

Deactivation of a SIGNAL declaration predefined in the system:

SIGNAL *System signal name* FALSE

### Explanation of the syntax

Element	Description
<i>Signal name</i>	Any name
<i>Signal variable</i>	Predefined signal variable. The following types are available: <ul style="list-style-type: none"> <li>■ \$IN[x]</li> <li>■ \$OUT[x]</li> <li>■ \$ANIN[x]</li> <li>■ \$ANOUT[x]</li> </ul>
TO	Groups together several consecutive binary inputs or outputs (max. 32) to form a digital input or output. The combined signals can be addressed with a decimal name, a hexadecimal name (prefix H) or with a bit pattern name (prefix B). They can also be processed with Boolean operators.
<i>System signal name</i>	Signal name predefined in the system, e.g. \$T1.
FALSE	Deactivates a SIGNAL declaration predefined in the system. The inputs or outputs to which the SIGNAL declaration refers are thus available again for other purposes.  FALSE is not a Boolean value here, but a keyword. The option TRUE is not available. If the SIGNAL declaration that has been deactivated by means of FALSE is to be reactivated, the program line containing the entry FALSE must be deleted.

#### Example 1

The output \$OUT[7] is assigned the name START\_PROCESS. The output \$OUT[7] is set.

```
SIGNAL START_PROCESS $OUT[7]
START_PROCESS = TRUE
```

#### Example 2

The outputs \$OUT[1] to \$OUT[8] are combined to form one digital output under the name OUTWORT. The outputs \$OUT[3], \$OUT[4], \$OUT[5] and \$OUT[7] are set.

```
SIGNAL OUTWORT $OUT[1] TO $OUT[8]
OUTWORT = 'B01011100'
```

## 10.9 Subprograms and functions

### 10.9.1 RETURN

#### Description

Jump from a subprogram or function back to the program from which the subprogram or function was called.

#### Subprograms

RETURN can be used to return to the main program if a certain condition is met in the subprogram. No values from the subprogram can be transferred to the main program.

#### Functions

Functions must be ended by a RETURN statement containing the value that has been determined. The determined value is hereby transferred to the program from which the function was called.

**Syntax**

For subprograms:

RETURN

For functions:

RETURN *Function value*

**Explanation of the syntax**

Element	Description
<i>Function value</i>	Type: The data type of <i>Function value</i> must match the data type of the function.  <i>Function value</i> is the value determined by the function. The value can be specified as a constant, a variable or an expression.

**Example 1**

Return from a subprogram to the program from which it was called, dependent on a condition.

```
DEF PROG_2 ()
  ...
  IF $IN[5]==TRUE THEN
    RETURN
  ...
END
```

**Example 2**

Return from a function to the program from which it was called. The value X is transferred.

```
DEFFCT INT CALCULATE (X:IN)
  INT X
  X=X*X
  RETURN X
ENDFCT
```

## 10.10 Interrupt programming

### 10.10.1 BRAKE

**Description**

BRAKE stops the robot.



BRAKE may only be used in an interrupt program.

The interrupt program is not continued until the robot has come to a stop. The robot motion is resumed as soon as the interrupt program has been completed.

**Syntax**

BRAKE <F>

**Explanation of the syntax**

Element	Description
F	F triggers a STOP 1.  In the case of a BRAKE statement without F, the robot brakes with a STOP 2.

**Example**

(>>> 10.10.3 "INTERRUPT" Page 330)

## 10.10.2 INTERRUPT ... DECL ... WHEN ... DO

### Description

In the case of a defined event, e.g. an input, the controller interrupts the current program and executes a defined subprogram. The event and the subprogram are defined by INTERRUPT ... DECL ... WHEN ... DO.

Once the subprogram has been executed, the interrupted program is resumed at the point at which it was interrupted. Exception: RESUME.

A subprogram called by an interrupt is called an interrupt program.

A maximum of 32 interrupts may be declared simultaneously. An interrupt declaration may be overwritten by another at any time.



The interrupt declaration is a statement. It must be situated in the statements section of the program and not in the declaration section!



When first declared, an interrupt is deactivated. The interrupt must be activated before the system can react to the defined event!  
(>>> 10.10.3 "INTERRUPT" Page 330)

### Syntax

<GLOBAL> INTERRUPT DECL *Prio* WHEN *Event* DO *Subprogram*

### Explanation of the syntax

Element	Description
GLOBAL	An interrupt is only recognized at, or below, the level in which it is declared. In other words, an interrupt declared in a subprogram is not recognized in the main program (and cannot be activated there). If an interrupt is also to be recognized at higher levels, the declaration must be preceded by the keyword GLOBAL.
<i>Prio</i>	Type: INT  If several interrupts occur at the same time, the interrupt with the highest priority is processed first, then those of lower priority. 1 = highest priority.  Priorities 1, 2, 4 to 39 and 81 to 128 are available.  <b>Note:</b> Priorities 3 and 40 to 80 are reserved for use by the system. They must <b>not</b> be used by the user because this would cause system-internal interrupts to be overwritten and result in errors.
<i>Event</i>	Type: BOOL  Event that is to trigger the interrupt. Structure components are impermissible. The following are permitted: <ul style="list-style-type: none"> <li>■ a Boolean constant</li> <li>■ a Boolean variable</li> <li>■ a signal name</li> <li>■ a comparison</li> <li>■ a simple logic operation: NOT, OR, AND or EXOR</li> </ul>
<i>Subprogram</i>	The name of the interrupt program to be executed. Runtime variables may not be transferred to the interrupt program as parameters, with the exception of variables declared in a data list.

### Example 1

Declaration of an interrupt with priority 23 that calls the subprogram SP1 if \$IN[12] is true. The parameters 20 and VALUE are transferred to the subprogram.

```
INTERRUPT DECL 23 WHEN $IN[12]==TRUE DO UP1(20,WERT)
```

### Example 2

Two objects, the positions of which are detected by two sensors connected to inputs 6 and 7, are located on a programmed path. The robot is to be moved subsequently to these two positions.

For this purpose, the two detected positions are saved as points P\_1 and P\_2. These points are then addressed in the second section of the main program.

If the robot controller detects an event defined by means of INTERRUPT ... DECL ... WHEN ... DO, it always saves the current robot position in the system variables \$AXIS\_INT (axis-specific) and \$POS\_INT (Cartesian).

Main program:

```
DEF PROG()
...
INTERRUPT DECL 10 WHEN $IN[6]==TRUE DO UP1()
INTERRUPT DECL 20 WHEN $IN[7]==TRUE DO UP2()
...
INTERRUPT ON
LIN START
LIN END
INTERRUPT OFF
LIN P_1
LIN P_2
...
END
```

Local interrupt program 1:

```
DEF UP1()
P_1=$POS_INT
END
```

Local interrupt program 2:

```
DEF UP2()
P_2=$POS_INT
END
```

### 10.10.3 INTERRUPT

#### Description

Executes one of the following actions:

- Activates an interrupt.
- Deactivates an interrupt.
- Disables an interrupt.
- Enables an interrupt.

The interrupt must previously have been declared. ([>>> 10.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 329](#))

#### Syntax

INTERRUPT *Action* <*Number*>

## Explanation of the syntax

Element	Description
Action	<ul style="list-style-type: none"> <li>■ ON: Activates an interrupt.</li> <li>■ OFF: Deactivates an interrupt.</li> <li>■ DISABLE: Disables an activated interrupt.</li> <li>■ ENABLE: Enables a disabled interrupt.</li> </ul>
Number	<p>Type: INT</p> <p>Number (= priority) of the interrupt to which the Action is to refer.</p> <p>Number can be omitted. In this case, ON or OFF refers to all declared interrupts, while DISABLE or ENABLE refers to all active interrupts.</p>



Up to 16 interrupts may be active at any one time. In this regard, particular attention must be paid to the following:

- If, in the case of INTERRUPT ON, the Number is omitted, all declared interrupts are activated. The maximum permissible total of 16 may not be exceeded, however.
- If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed.



If, in the interrupt declaration, a Boolean variable, e.g. an input, has been defined as the Event:

- In this case, the interrupt is triggered by a change of state, e.g. in the case of \$IN[x]==TRUE by the change from FALSE to TRUE. The state must therefore not already be present at INTERRUPT ON, as the interrupt is not then triggered!
- Furthermore, the following must also be considered in this case: the change of state must not occur until at least one interpolation cycle after INTERRUPT ON.

(This can be achieved by programming a WAIT SEC 0.012 after INTERRUPT ON. If no advance run stop is desired, a CONTINUE command can also be programmed before the WAIT SEC.)

The reason for this is that INTERRUPT ON requires one interpolation cycle (= 12 ms) before the interrupt is actually activated. If the state changes before this, the interrupt cannot detect the change.

### Example 1

The interrupt with priority 2 is activated. (The interrupt must already be declared.)

```
INTERRUPT ON 2
```

### Example 2

A non-path-maintaining EMERGENCY STOP is executed via the hardware during application of adhesive. The application of adhesive is stopped by the program and the adhesive gun is repositioned onto the path after enabling (by input 10).

```
DEF PROG()
...
INTERRUPT DECL 1 WHEN $STOPMESS DO STOP_PROG()
LIN P_1
INTERRUPT ON
LIN P_2
INTERRUPT OFF
...
END
```

```
DEF STOP_PROG()
BRAKE F
GLUE=FALSE
WAIT FOR $IN[10]
LIN $POS_RET
GLUE=TRUE
END
```

#### 10.10.4 RESUME

##### Description

RESUME may only occur in interrupt programs. (Not in interrupt programs, however, that are called by an interrupt that is declared as GLOBAL.) RESUME cancels all running interrupt programs and subprograms up to the level at which the current interrupt was declared.

When the RESUME statement is activated, the advance run pointer must not be at the level where the interrupt was declared, but at least one level lower.

Changing the variable \$BASE in the interrupt program only has an effect there. The computer advance run, i.e. the variable \$ADVANCE, must not be modified in the interrupt program.

The behavior of the robot controller after RESUME depends on the following motion instruction:

- PTP instruction: is executed as a PTP motion.
- LIN instruction: is executed as a LIN motion.
- CIRC instruction: is always executed as a LIN motion!

Following a RESUME statement, the robot is not situated at the original start point of the CIRC motion. The motion will thus differ from how it was originally planned; this can potentially be very dangerous, particularly in the case of CIRC motions.



If the first motion instruction after RESUME is a CIRC motion, this is always executed as LIN! This must be taken into consideration when programming RESUME statements. The robot must be able to reach the end point of the CIRC motion safely, by means of a LIN motion, from any position in which it could find itself when the RESUME statement is executed.  
Failure to observe this may result in death to persons, physical injuries or damage to property.

##### Syntax

RESUME

##### Example

The robot is to search for a part on a path. The part is detected by means of a sensor at input 15. Once the part has been found, the robot is not to continue to the end point of the path, but to return to the interrupt position and pick up the part. The main program is then to be resumed.

Motions that are to be canceled by means of BRAKE and RESUME must be programmed in a subprogram. (Here SEARCH().)

Main program:

```
DEF PROG()
INI
...
INTERRUPT DECL 21 WHEN $IN[15] DO FOUND()
PTP HOME
...
SEARCH()
$ADVANCE=3
...
END
```

Subprogram with search path:

When the RESUME statement is activated, the advance run pointer must not be at the level where the current interrupt was declared. To prevent this, the advance run is set to 0 here in the subprogram.

```
DEF SEARCH()
INTERRUPT ON 21
LIN START_SEARCH
LIN END_SEARCH
$ADVANCE=0
...
END
```

Interrupt program:

LIN \$POS\_INT is the return motion to the position at which the interrupt was triggered. After LIN \$POS\_INT (in the example: ...), the robot grips the part. RESUME causes the main program to be resumed after the part has been gripped. Without the RESUME statement, the subprogram SEARCH would be resumed after END.

```
DEF FOUND()
INTERRUPT OFF
BRAKE
LIN $POS_INT
...
RESUME
END
```

## 10.11 Path-related switching actions (=Trigger)

### 10.11.1 TRIGGER WHEN DISTANCE

**Description** The Trigger triggers a defined statement. The statement refers to the start point or end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.

The statement can be shifted in time. It is then not triggered exactly at the start or end point, but brought forward or delayed.

**Syntax** TRIGGER WHEN DISTANCE=*Position* DELAY=*Time* DO *Statement* <PRIO=*Priority*>

Explanation of the syntax	Element	Description
	<i>Position</i>	<p>Type: INT; variable or constant</p> <p>Defines the point at which the statement is triggered. Possible values: 0 or 1.</p> <ul style="list-style-type: none"> <li>■ <b>0:</b> The statement is triggered at the start point of the motion block. If the start point is approximated, the statement is triggered at the end of the approximate positioning arc.</li> <li>■ <b>1:</b> The statement is triggered at the end point. If the end point is approximated, the statement is triggered in the middle of the approximate positioning arc.</li> </ul>
	<i>Time</i>	<p>Type: REAL; variable or constant; unit: ms</p> <p>Shifts the statement in time. Obligatory specification: if no time shift is desired, set <i>Time</i> = 0.</p> <p>The statement cannot be shifted freely in time. The shifts that are available depend on the value selected for <i>Position</i>:</p> <ul style="list-style-type: none"> <li>■ <b>Position = 0 (start point)</b> In this case, the statement can only be triggered with a delay, i.e. it is only possible to select a positive value for <i>Time</i>. The statement can be delayed, <b>at most, as far as the end point</b>. If the end point is approximated, the statement can be delayed, at most, as far as the start of the approximate positioning arc.</li> <li>■ <b>Position = 1 (end point)</b> In this case, a distinction must be made as to whether the end point is an exact positioning point or an approximate positioning point. <ul style="list-style-type: none"> <li>■ <b>Exact positioning point:</b> In this case, the statement can only be triggered earlier, i.e. it is only possible to select a negative value for <i>Time</i>. The statement can be brought forward, <b>at most, as far as the start point</b>. If the start point is approximated, the statement can be brought forward, at most, as far as the end of the approximate positioning arc.</li> <li>■ <b>Approximate positioning point:</b> In this case, the statement can be triggered earlier or with a delay, i.e. it is possible to select a negative or positive value for <i>Time</i>. The statement can be shifted, <b>at most, as far as the start or end of the approximate positioning arc</b> of the end point.</li> </ul> </li> </ul>

Element	Description
<i>Statement</i>	<p>Possible:</p> <ul style="list-style-type: none"> <li>■ Assignment of a value to a variable <b>Note:</b> There must be no runtime variable on the left-hand side of the assignment.</li> <li>■ OUT statement</li> <li>■ PULSE statement</li> <li>■ Subprogram call. In this case, <i>Priority</i> must be specified.</li> </ul>
<i>Priority</i>	<p>Type: INT; variable or constant</p> <p>Priority of the trigger. Only relevant if <i>Statement</i> calls a subprogram, and then obligatory.</p> <p>Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: PRIO = -1.</p> <p>If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.</p>



If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

## System variables

Useful system variables for working with triggers:

System variable	Data type	Description
\$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
\$DISTANCE	REAL	Overall length of current CP motion

### Example 1

130 milliseconds after P\_2, \$OUT[8] is set to TRUE.

```
LIN P_2
TRIGGER WHEN DISTANCE=0 DELAY=130 DO $OUT[8]=TRUE
LIN P_3
```

### Example 2

In the middle of the approximate positioning arc of P\_5, the subprogram MY\_SUBPROG with priority 5 is called.

```
PTP P_4
TRIGGER WHEN DISTANCE=1 DELAY=0 DO MY_SUBPROG() PRIO=5
PTP P_5 C_DIS
```

### Example 3

Explanation of the diagram:

In the diagram, the approximate positions in which the Triggers would be triggered are indicated by arrows.

In addition to these points, the start, middle and end of each approximate positioning arc are indicated.

```

1 DEF PROG()
2 ...
3 PTP P_0
4 TRIGGER WHEN DISTANCE=0 DELAY=40 DO A=12
5 ...
6 TRIGGER WHEN DISTANCE=1 DELAY=-20 DO UP1() PRIO=10
7 ...
8 LIN P_1
9 TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(A) PRIO=5
10 ...
11 TRIGGER WHEN DISTANCE=1 DELAY=15 DO B=1
12 ...
13 LIN P_2 C_DIS
14 TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(B) PRIO=12
15 ...
16 TRIGGER WHEN DISTANCE=1 DELAY=0 DO UP(A,B,C) PRIO=6
17 ...
18 LIN P_3 C_DIS
19 TRIGGER WHEN DISTANCE=0 DELAY=50 DO UP2(A) PRIO=4
20 ...
21 TRIGGER WHEN DISTANCE=1 DELAY=-80 DO A=0
22 ...
23 LIN P_4
24 ...
25 END

```

Line	Description
4	Switching range: 0 - 1
6	Switching range: 0 - 1
9	Switching range: 1 - 2*start
11	Switching range: 2*start - 2*end
14	Switching range: 2*end - 3*start
16	Switching range: 3*start - 3*end
19	Switching range: 3*end - 4
21	Switching range: 3*end - 4

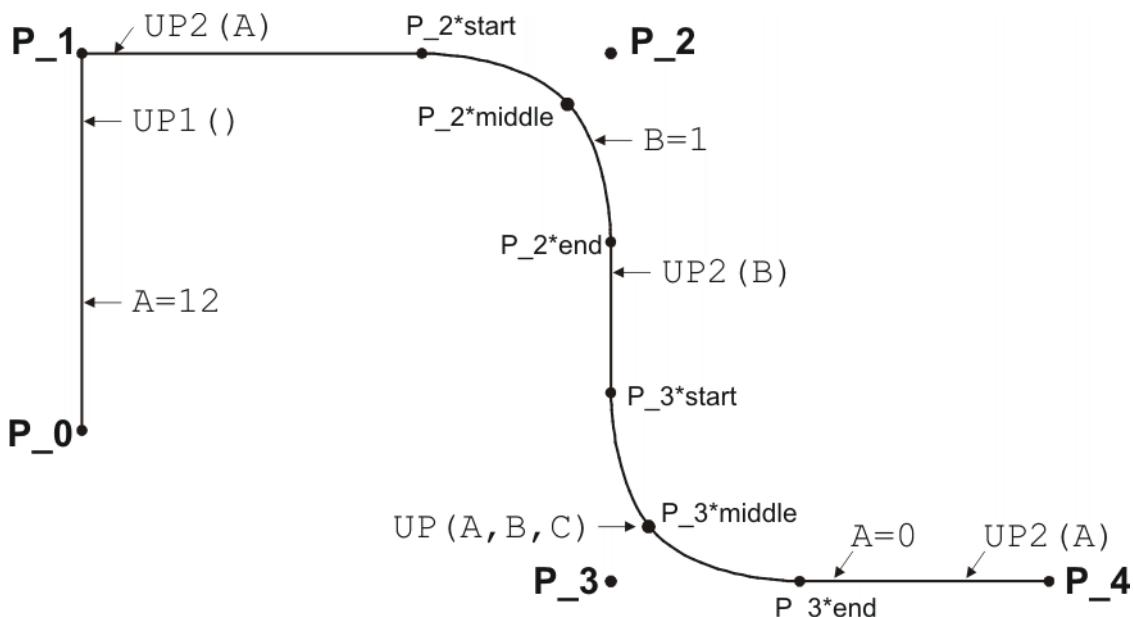


Fig. 10-13: Example of TRIGGER WHEN DISTANCE

### 10.11.2 TRIGGER WHEN PATH

#### Description

The Trigger triggers a defined statement. The statement refers to the end point of the motion block in which the Trigger is situated in the program. It is possible

to shift the statement in time and/or space so that it is not triggered exactly at the end point, but before or after it.



The end point must be LIN or CIRC. It must not be PTP.

The statement is executed parallel to the robot motion.

## Syntax

```
TRIGGER WHEN PATH =Distance DELAY=Time DO Statement <PRIO=Priority>
```

## Explanation of the syntax

Element	Description
<i>Distance</i>	<p>Type: REAL; variable or constant; unit: mm</p> <p>Obligatory specification. If no shift in space is desired, set <i>Distance</i> = 0.</p> <p>If the statement is to be shifted in space, the desired distance from the end point must be specified here. If this end point is approximated, <i>Distance</i> is the distance to the position on the approximate positioning arc closest to the end point.</p> <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion.</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> <p>The statement cannot be shifted freely. Maximum possible shift: see below, section "Switching range".</p>
<i>Time</i>	<p>Type: REAL; variable or constant; unit: ms</p> <p>Obligatory specification. If no shift in time is desired, set <i>Time</i> = 0.</p> <p>If the statement is to be shifted in time (relative to PATH), the desired duration must be specified here.</p> <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion.</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul> <p>The statement cannot be shifted freely. Maximum possible shift: see below, section "Switching range".</p>

Element	Description
<i>Statement</i>	<p>Possible:</p> <ul style="list-style-type: none"> <li>■ Assignment of a value to a variable <b>Note:</b> There must be no runtime variable on the left-hand side of the assignment.</li> <li>■ OUT statement</li> <li>■ PULSE statement</li> <li>■ Subprogram call. In this case, <i>Priority</i> must be specified.</li> </ul>
<i>Priority</i>	<p>Type: INT; variable or constant</p> <p>Priority of the trigger. Only relevant if <i>Statement</i> calls a subprogram, and then obligatory.</p> <p>Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: PRIO = -1.</p> <p>If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.</p>



If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

## System variables

Useful system variables for working with triggers:

System variable	Data type	Description
\$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
\$DISTANCE	REAL	Overall length of current CP motion

## Switching range

- Shift towards the end of the motion:

A statement can be shifted, **at most, as far as the next exact positioning point** after TRIGGER WHEN PATH (skipping all approximate positioning points).

In other words, if the end point is an exact positioning point, the statement cannot be shifted beyond the end point.

- Shift towards the start of the motion:

A statement can be shifted, **at most, as far as the start point of the motion block** (i.e. as far as the last point before TRIGGER WHEN PATH).

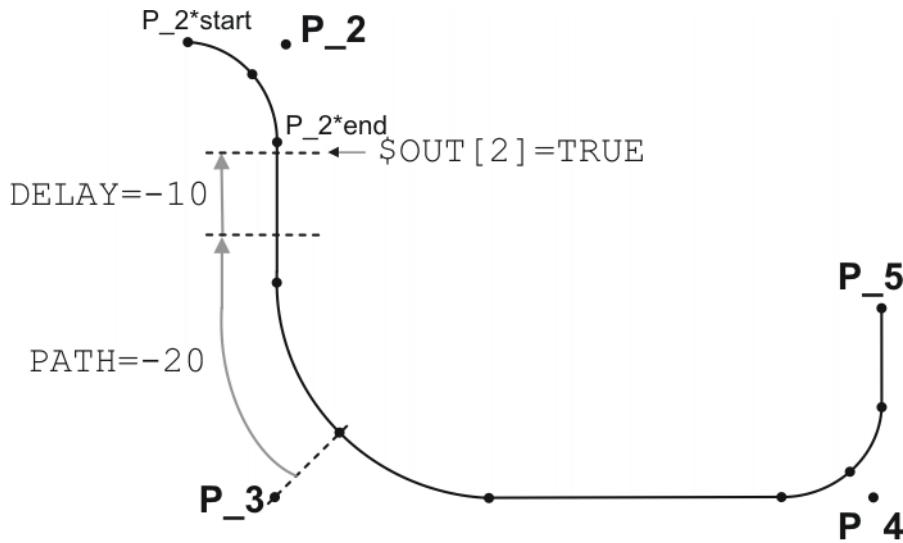
If the start point is an approximated LIN or CIRC point, the statement can be brought forward, at most, as far as the start of its approximate positioning arc.

If the start point is an approximated PTP point, the statement can be brought forward, at most, as far as the end of its approximate positioning arc.

## Example

```
LIN P_2 C_DIS
TRIGGER WHEN PATH = -20.0 DELAY= -10 DO $OUT[2]=TRUE
LIN P_3 C_DIS
LIN P_4 C_DIS
LIN P_5
```

In the diagram, the approximate position in which the  $\$OUT[2]=TRUE$  statement would be triggered is indicated by an arrow.



**Fig. 10-14: Example of TRIGGER WHEN PATH**

Switching range:  $P_2^{*start}$  to  $P_5$ .

If  $P_2$  were not approximated, the switching range would be  $P_2$  to  $P_5$ .

The switching range goes to  $P_5$  because  $P_5$  is the next exact positioning point after the TRIGGER statement. If  $P_3$  were not approximated, the switching range would be  $P_2$  to  $P_3$ , as  $P_3$  is the next exact positioning point in the program after the Trigger statement.

### 10.11.3 TRIGGER WHEN PATH (for SPLINE)

<b>Description</b>	This trigger can only be used in spline blocks.  The Trigger triggers a defined statement. The statement refers to the start point or end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.  The statement can be shifted in time and/or space. It is then not triggered exactly at the start or end point, but beforehand or afterwards.
<b>Syntax</b>	<code>TRIGGER WHEN PATH = <i>Distance</i> &lt;ONSTART&gt; <i>DELAY</i> = <i>Time</i> DO <i>Statement</i> &lt;PRIO = <i>Priority</i>&gt;</code>
<b>Functions</b>	<code>PATH</code> and <code>DELAY</code> can call functions. The functions are subject to constraints. (>>> 9.3.4.14 "Limits for functions in the spline trigger" Page 272)

**Explanation of  
the syntax**

Element	Description
<i>Distance</i>	<p>Type: REAL; variable, constant or function; unit: mm</p> <p>Obligatory specification. If no shift in space is desired, set <i>Distance</i> = 0.</p> <p>If the statement is to be shifted in space, the desired distance from the start or end point must be specified here.</p> <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion.</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul>
ONSTART	<p>The PATH value refers to the start point.</p> <p>Without ONSTART: the PATH value refers to the end point.</p> <p>(&gt;&gt;&gt; 10.11.3.1 "Spline: trigger point in the case of approximate positioning" Page 341)</p>
<i>Time</i>	<p>Type: REAL; variable, constant or function; unit: ms</p> <p>Obligatory specification. If no shift in time is desired, set <i>Time</i> = 0.</p> <p>If the statement is to be shifted in time (relative to PATH), the desired duration must be specified here.</p> <ul style="list-style-type: none"> <li>■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms</li> <li>■ Negative value: shifts the statement towards the start of the motion.</li> </ul>
<i>Statement</i>	<p>Possible:</p> <ul style="list-style-type: none"> <li>■ Assignment of a value to a variable</li> <li>■ <b>Note:</b> There must be no runtime variable on the left-hand side of the assignment.</li> <li>■ OUT statement</li> <li>■ PULSE statement</li> <li>■ Subprogram call. In this case, <i>Priority</i> must be specified.</li> </ul>
<i>Priority</i>	<p>Type: INT; variable or constant</p> <p>Priority of the trigger. Only relevant if <i>Statement</i> calls a subprogram, and then obligatory.</p> <p>Priorities 1, 2, 4 to 39 and 81 to 128 are available. Priorities 3 and 40 to 80 are reserved for cases in which the priority is automatically assigned by the system. If the priority is to be assigned automatically by the system, the following is programmed: PRIO = -1.</p> <p>If several triggers call subprograms at the same time, the trigger with the highest priority is processed first, then the triggers of lower priority. 1 = highest priority.</p>



If a trigger calls a subprogram, it counts as an active interrupt until the subprogram has been executed. Up to 16 interrupts may be active at any one time.

**System variables**

Useful system variables for working with triggers:

System variable	Data type	Description
\$DIST_NEXT	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
\$DISTANCE	REAL	Overall length of current CP motion

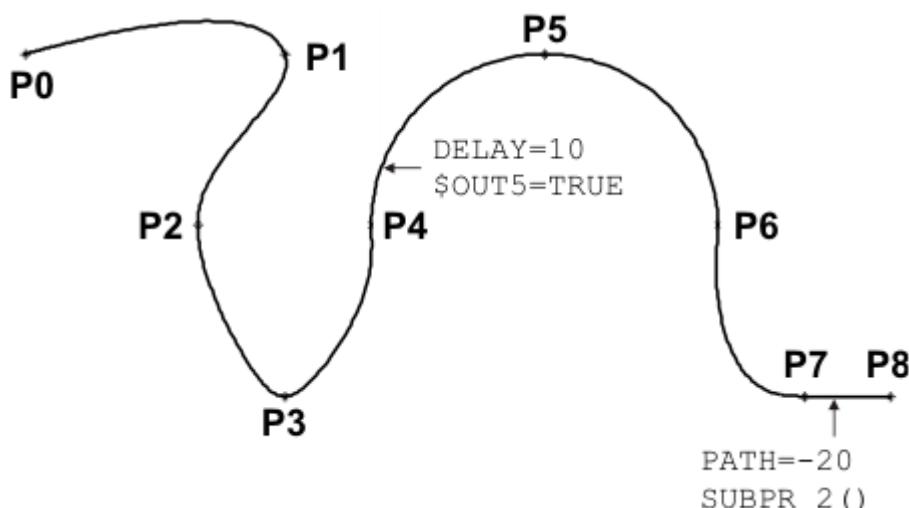
**Example**

```

1 PTP P0
2 SPLINE
3 SPL P1
4 SPL P2
5 SPL P3
6 SPL P4
7 TRIGGER WHEN PATH=0 ONSTART DELAY=10 DO $OUT[5]=TRUE
8 SCIRC P5, P6
9 SPL P7
10 TRIGGER WHEN PATH=-20.0 DELAY=0 DO SUBPR_2() PRIO=-1
11 SLIN P8
12 ENDSPLINE

```

The Trigger in line 10 would have the same result if it was positioned directly before the spline block (i.e. between line 1 and line 2). In both cases, it refers to the last point of the spline motion: P8.



**Fig. 10-15: Example of TRIGGER WHEN PATH (for spline)**

#### 10.11.3.1 Spline: trigger point in the case of approximate positioning

The point at which the trigger is triggered is the end point of the motion block in which the trigger is situated. If ONSTART is used, it is the start point.

If this end or start point is approximated, the trigger point is transferred onto the approximate positioning arc by a distance corresponding to the approximation distance.

- The trigger point is not generally in the middle of the approximate positioning arc.
- Triggers that are triggered at the same point in the case of exact positioning are triggered at different points in the case of approximate positioning. The trigger that refers to the end point is triggered later than the trigger that refers to the start point

**Example**

Trigger 1 and trigger 2 would be triggered at the same point if P3 was not approximated.

- Trigger 1 refers to the end point of the block in which it is situated, i.e. P3.
- Trigger 2 (= with ONSTART) refers to the start point of the block in which it is situated, i.e. also P3.

**Trigger 1:**

```
SPLINE
...
SLIN P2
TRIGGER WHEN PATH=0 DELAY=0 DO ...
SLIN P3
ENDSPLINE C_DIS
SPLINE
SLIN4
...
ENDSPLINE
```

**Trigger 2:**

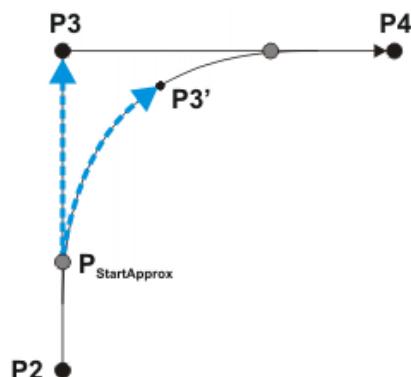
```
SPLINE
...
SLIN P2
SLIN P3
ENDSPLINE C_DIS
SPLINE
TRIGGER WHEN PATH=0 ONSTART DELAY=0 DO ...
SLIN4
...
ENDSPLINE
```

P3 is approximated, however. It is transferred onto the approximate positioning arc by a distance corresponding to the approximation distance (= P3').

**Trigger 1:**

Trigger 1 is situated in the block P2 --> P3 and refers to the end point. The robot controller calculates how far the distance would be from the start of the approximate positioning arc to the end point with exact positioning. This distance is then applied to the approximate positioning arc.

The distance  $P_{StartApprox} \rightarrow P3'$  is the same as  $P_{StartApprox} \rightarrow P3$ .



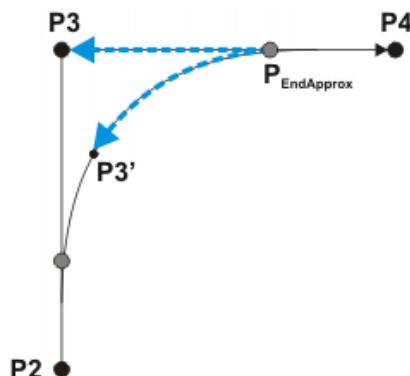
**Fig. 10-16: Trigger 1: Trigger point in the case of approximate positioning**

<b>P3</b>	Trigger point in the case of exact positioning
<b>P3'</b>	Trigger point in the case of approximate positioning
<b>P<sub>StartApprox</sub></b>	Start of the approximate positioning arc
<b>P<sub>EndApprox</sub></b>	End of the approximate positioning arc

**Trigger 2:**

Trigger 2 is situated in the block P3 --> P4 and refers to the start point. The robot controller calculates how far the distance would be from the end of the approximate positioning arc back to the start point with exact positioning. This distance is then applied to the approximate positioning arc.

The distance  $P_{EndApprox} \rightarrow P3'$  is the same as  $P_{EndApprox} \rightarrow P3$ .



**Fig. 10-17: Trigger 2: Trigger point in the case of approximate positioning**

## 10.12 Communication

Information about the following statements is contained in the Expert documentation CREAD/CWRITE.

- CAST\_FROM
- CAST\_TO
- CCLOSE
- CHANNEL
- CIOCTL
- COPEN
- CREAD
- CWRITE
- SREAD
- SWRITE

## 10.13 System functions

### 10.13.1 VARSTATE()

**Description** VARSTATE() can be used to monitor the state of a variable.

VARSTATE() is a function with a return value of type VAR\_STATE.

VAR\_STATE is an enumeration type that is defined as follows in the system:

```
ENUM VAR_STATE DECLARED, INITIALIZED, UNKNOWN
```

VARSTATE is defined as follows in the system:

```
VAR_STATE VARSTATE (CHAR VAR_STR[80]:IN)
```

**Example 1**

```

DEF PROG1()
INT MYVAR
...
IF VARSTATE ("MYVAR") ==#UNKNOWN THEN
    $OUT[11]=TRUE
ENDIF
...
IF VARSTATE ("MYVAR") ==#DECLARED THEN
    $OUT[12]=TRUE
ENDIF
...
IF VARSTATE ("ANYVAR") ==#UNKNOWN THEN
    $OUT[13]=TRUE
ENDIF
...
MYVAR=9
...
IF VARSTATE ("MYVAR") ==#DECLARED THEN
    $OUT[14]=TRUE
ENDIF
...
IF VARSTATE ("MYVAR") ==#INITIALIZED THEN
    $OUT[15]=TRUE
ENDIF
...
END

```

**Explanation of the state monitoring:**

- The first IF condition is false, as MYVAR has already been declared. Output 11 is not set.
- The second IF condition is true, as MYVAR has been declared. Output 12 is set.
- The third IF condition is true, on the condition that there is also no variable with the name ANYVAR in \$CONFIG.DAT. Output 13 is set.
- The fourth IF condition is false, as MYVAR has not only been declared, but has also already been initialized here. Output 14 is not set.
- The fifth IF condition is true, as MYVAR has been initialized. Output 15 is set.

**Example 2**

```

DEF PROG2()
INT MYVAR
INT YOURVAR
DECL VAR_STATE STATUS
...
STATUS=VARSTATE ("MYVAR")
UP()
...
STATUS=VARSTATE ("YOURVAR")
UP()
...
END

```

```

DEF UP()
...
IF VARSTATE ("STATUS") ==#DECLARED THEN
    $OUT[100]=TRUE
ENDIF
...
END

```

**Explanation of the state monitoring:**

In this example, the state is monitored indirectly, i.e. via an additional variable. The additional variable must be of type VAR\_STATE. The keyword DECL must not be omitted in the declaration. The name of the additional variable may be freely selected. In this example it is STATUS.

### 10.13.2 ROB\_STOP() and ROB\_STOP\_RELEASE()

**Description** The statement ROB\_STOP() stops the robot and prevents further motions. This affects all possible motions, irrespective of whether they arise from program execution, manual jogging or command mode.

The statement ROB\_STOP\_RELEASE() cancels a blockade caused by ROB\_STOP().



ROB\_STOP() and ROB\_STOP\_RELEASE() can only be used in submit programs, not robot programs.

If ROB\_STOP() is called several times in succession without a ROB\_STOP\_RELEASE() in between, only the first call has any effect. The subsequent calls have no effect, i.e. they do not trigger a stop or cause a message to be generated.

If ROB\_STOP\_RELEASE() is called without ROB\_STOP() being called first, this has no effect.

**Messages** ROB\_STOP() triggers the following status message: **Robot stopped by submit**

ROB\_STOP\_RELEASE() in a Test mode triggers the following acknowledgement message: **Ackn. Robot stopped by submit**

ROB\_STOP\_RELEASE() in an Automatic mode triggers no message.

#### Syntax

`result = ROB_STOP (stop_type: IN)`

#### Explanation of the syntax

Element	Description
<code>result</code>	Type: BOOL  Variable for the return value. Return value: <ul style="list-style-type: none"><li>■ TRUE: The stop has been executed.</li><li>■ FALSE: An invalid parameter has been transferred for <code>stop_type</code>.</li></ul>
<code>stop_type</code>	Type: ROB_STOP_T  Stop type to be used to stop the robot: <ul style="list-style-type: none"><li>■ #RAMP_DOWN: ramp stop</li><li>■ #PATH_MAINTAINING: path-maintaining EMERGENCY STOP</li></ul> Other stop types are not possible.

#### ProConOS

The "Robot stop" function can also be used from ProConOS. The following functions are available for this:

- PLC\_ROB\_STOP()  
The desired stop type is defined with PLC\_ROB\_STOP\_RAMP\_DOWN or PLC\_ROB\_STOP\_PATH\_MAINT.
- PLC\_ROB\_STOP\_RELEASE()

The effect is the same, irrespective of whether a stop is triggered by submit or by ProConOS. ProConOS generates its own message texts; these are:

- **Robot stopped by SoftPLC (Name of the task calling it)**
- **Ackn. Robot stopped by SoftPLC**

If a stop is requested by both submit and ProConOS, this is indicated in each case by a status message. A maximum of 2 status messages can thus be dis-

played for an executed stop. In this case, robot motion cannot be resumed until the blockade has been canceled by both submit and ProConOS.

If different stop types are requested by submit and ProConOS, the type actually carried out is generally the one that was requested first.



A stop triggered by ProConOS cannot be canceled by a submit program and vice versa.

## 10.14 Editing string variables

Various functions are available for editing string variables. The functions can be used in SRC files, in SUB files and in the variable correction function.

### 10.14.1 String variable length in the declaration

**Description** The function `StrDeclLen()` determines the length of a string variable according to its declaration in the declaration section of a program.

**Syntax** `Length = StrDeclLen (StrVar[])`

**Explanation of the syntax**

Element	Description
Length	Type: INT  Variable for the return value. Return value: Length of the string variable as declared in the declaration section
StrVar[]	Type: CHAR array  String variable whose length is to be determined  Since the string variable StrVar[ ] is an array of type CHAR, individual characters and constants are not permissible for length determination.

**Example**

```

1 CHAR ProName[24]
2 INT StrLength
...
3 StrLength = StrDeclLen(ProName)
4 StrLength = StrDeclLen($Trace.Name[ ])

```

Line	Description
3	StrLength = 24
4	StrLength = 64

### 10.14.2 String variable length after initialization

**Description** The function `StrLen()` determines the length of the character string of a string variable as defined in the initialization section of the program.

**Syntax** `Length = StrLen (StrVar)`

**Explanation of the syntax**

Element	Description
Length	Type: INT  Variable for the return value. Return value: Number of characters currently assigned to the string variable
StrVar	Type: CHAR  Character string or variable whose length is to be determined

**Example**

```

1  CHAR PartA[50]
2  INT AB
...
3  PartA[] = "This is an example"
4  AB = StrLen(PartA[])

```

Line	Description
4	AB = 18

### 10.14.3 Deleting the contents of a string variable

**Description** The function `StrClear()` deletes the contents of a string variable.

**Syntax** `Result = StrClear(StrVar[])`

**Explanation of the syntax**

Element	Description
Result	Type: BOOL  Variable for the return value. Return value: <ul style="list-style-type: none"><li>■ The contents of the string variable have been deleted: TRUE</li><li>■ The contents of the string variable have not been deleted: FALSE</li></ul>
StrVar[]	Type: CHAR array  Variable whose character string is to be deleted

**Example**

```

IF (NOT StrClear($Loop_Msg[])) THEN
HALT
ENDIF

```



The function can be used within IF branches without the return value being explicitly assigned to a variable. This applies to all functions for editing string variables.

### 10.14.4 Extending a string variable

**Description** The function `StrAdd()` can be used to expand a string variable with the contents of another string variable.

**Syntax** `Sum = StrAdd(StrDest[], StrToAdd[])`

**Explanation of  
the syntax**

Element	Description
Sum	Type: INT  Variable for the return value. Return value: Sum of StrDest[ ] and StrToAdd[ ]  If the sum is longer than the previously defined length of StrDest[], the return value is 0. This is also the case if the sum is greater than 470 characters.
StrDest[]	Type: CHAR array  The string variable to be extended  Since the string variable StrDest[ ] is an array of type CHAR, individual characters and constants are not permissible.
StrToAdd[]	Type: CHAR array  The character string by which the variable is to be extended

**Example**

```

1 DECL CHAR A[50], B[50]
2 INT AB, AC
...
3 A[] = "This is an "
4 B[] = "example"
5 AB = StrAdd(A[],B[])

```

Line	Description
5	A[] = "This is an example" AB = 18

#### 10.14.5 Searching a string variable

**Description**

The function `StrFind()` can be used to search a string variable for a character string.

**Syntax**

`Result = StrFind(StartAt, StrVar[], StrFind[], CaseSens)`

**Explanation of  
the syntax**

Element	Description
Result	Type: INT  Variable for the return value. Return value: Position of the first character found. If no character is found, the return value is 0.
StartAt	Type: INT  The search is started from this position.
StrVar[]	Type: CHAR array  The string variable to be searched
StrFind[]	Type: CHAR array  The character string that is being looked for.
CaseSens	<ul style="list-style-type: none"> <li>■ #CASE_SENS: Upper and lower case are taken into consideration.</li> <li>■ #NOT_CASE_SENS: Upper and lower case are ignored.</li> </ul>

**Example**

```

1 DECL CHAR A[5]
2 INT B
3 A[]="ABCDE"
4 B = StrFind(1, A[], "AC", #CASE_SENS)
5 B = StrFind(1, A[], "a", #NOT_CASE_SENS)
6 B = StrFind(1, A[], "BC", #Case_Sens)
7 B = StrFind(1, A[], "bc", #NOT_CASE_SENS)

```

Line	Description
4	B = 0
5	B = 1
6	B = 2
7	B = 2

**10.14.6 Comparing the contents of string variables**

**Description** The function `StrComp()` can be used to compare two string variables.

**Syntax** `Comp = StrComp(StrComp1[], StrComp2[], CaseSens)`

**Explanation of the syntax**

Element	Description
Comp	Type: BOOL  Variable for the return value. Return value: <ul style="list-style-type: none"><li>■ The character strings match: TRUE</li><li>■ The character strings do not match: FALSE</li></ul>
StrComp1[]	Type: CHAR array  String variable that is compared with StrComp2[].
StrComp2[]	Type: CHAR array  String variable that is compared with StrComp1[].
CaseSens	<ul style="list-style-type: none"><li>■ #CASE_SENS: Upper and lower case are taken into consideration.</li><li>■ #NOT_CASE_SENS: Upper and lower case are ignored.</li></ul>

**Example**

```

1 DECL CHAR A[5]
2 BOOL B
3 A[]="ABCDE"
4 B = StrComp(A[], "ABCDE", #CASE_SENS)
5 B = StrComp(A[], "abcde", #NOT_CASE_SENS)
6 B = StrComp(A[], "abcd", #NOT_CASE_SENS)
7 B = StrComp(A[], "acbde", #NOT_CASE_SENS)

```

Line	Description
4	B = TRUE
5	B = TRUE
6	B = FALSE
7	B = FALSE

**10.14.7 Copying a string variable**

**Description** The function `StrCopy()` can be used to copy the contents of a string variable to another string variable.

**Syntax** `Copy = StrCopy(StrDest[], StrSource[])`

**Explanation of  
the syntax**

Element	Description
Copy	Type: BOOL  Variable for the return value. Return value: <ul style="list-style-type: none"><li>■ The string variable was copied successfully: TRUE</li><li>■ The string variable was not copied: FALSE</li></ul>
StrDest[]	Type: CHAR array  The character string is copied to this string variable.  Since StrDest[ ] is an array of type CHAR, individual characters and constants are not permissible.
StrSource[]	Type: CHAR array  The contents of this string variable are copied.

**Example**

```
1 DECL CHAR A[25], B[25]
2 DECL BOOL C
3 A[] = ""
4 B[] = "Example"
5 C = StrCopy(A[], B[])
```

Line	Description
5	A[ ] = "Example" C = TRUE

## 11 Submit interpreter

### 11.1 Function of the Submit interpreter

<b>Function</b>	2 tasks run in parallel on the robot controller:
■ Robot interpreter	The motion program runs in the robot interpreter.
■ Submit interpreter	A SUB program runs in the Submit interpreter. A SUB program can perform operator control or monitoring tasks. Examples: monitoring of safety equipment; monitoring of a cooling circuit. This means that no PLC is required for smaller applications, as the robot controller can perform such tasks by itself.
	The Submit interpreter starts automatically when the robot controller is switched on. The program defined in the file KRC/STEU/MADA/\$custom.dat is started. By default, this is SPS.SUB.

```
$PRO_I_O[]="/R1/SPS()"
```

The Submit interpreter can be stopped or deselected manually and can also be restarted. It is also possible to start a SUB program other than the one entered in \$custom.dat.

(>>> 11.2 "Manually stopping or deselecting the Submit interpreter"  
Page 352)

(>>> 11.3 "Manually starting the Submit interpreter" Page 352)

SUB programs are always files with the extension \*.SUB. The program SPS.SUB can be modified and new SUB programs can be created.

(>>> 11.4 "Modifying the program SPS.SUB" Page 353)

(>>> 11.5 "Creating a new SUB program" Page 353)



The Submit interpreter must not be used for time-critical applications! A PLC must be used in such cases. Reasons:

- The Submit interpreter shares system resources with the robot interpreter, which has the higher priority. The Submit interpreter is thus not executed at the robot controller's interpolation cycle rate of 12 ms. Furthermore, the runtime of the Submit interpreter is irregular.
- The runtime of the Submit interpreter is influenced by the number of lines in the SUB program. Even comment lines and blank lines have an effect.



If a system file, e.g. \$config.dat or \$custom.dat, is modified in such a way that errors are introduced, the Submit interpreter is automatically deselected. Once the error in the system file has been rectified, the Submit interpreter must be reselected manually.

### Display

The program SPS.SUB can be found in the "System" folder in the Navigator. SUB files are only visible in the Navigator in the user group "Expert".

By default, the execution of a selected SUB program is not displayed. This can be changed using the system variable \$INTERPRETER. The SUB program can only be displayed, however, if a motion program is selected at the same time.

Value of \$INTERPRETER	Description
1	The selected motion program is displayed in the editor (default).
0	The selected SUB program is displayed in the editor.

## 11.2 Manually stopping or deselecting the Submit interpreter

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ User group "Expert".</li> <li>■ Operating mode T1 or T2.</li> </ul>						
<b>Procedure</b>	<ul style="list-style-type: none"> <li>■ In the main menu, select <b>Configuration &gt; SUBMIT interpreter &gt; Stop or Deselect.</b></li> </ul>						
<b>Alternative procedure</b>	<ul style="list-style-type: none"> <li>■ In the status bar, touch the <b>Submit interpreter</b> status indicator. A window opens. Select <b>Stop or Deselect.</b></li> </ul>						
<b>Description</b>	<table border="1"> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><b>Stop</b></td> <td>The Submit interpreter is stopped. When it is restarted, the SUB program is resumed at the point at which it was stopped.</td> </tr> <tr> <td><b>Deselect</b></td> <td>The Submit interpreter is deselected. A different SUB program can now be selected.</td> </tr> </tbody> </table>	Command	Description	<b>Stop</b>	The Submit interpreter is stopped. When it is restarted, the SUB program is resumed at the point at which it was stopped.	<b>Deselect</b>	The Submit interpreter is deselected. A different SUB program can now be selected.
Command	Description						
<b>Stop</b>	The Submit interpreter is stopped. When it is restarted, the SUB program is resumed at the point at which it was stopped.						
<b>Deselect</b>	The Submit interpreter is deselected. A different SUB program can now be selected.						

Once the Submit interpreter has been stopped or deselected, the corresponding icon in the status bar is red or gray.

Icon	Color	Description
	Red	Submit interpreter has been stopped.
	Gray	Submit interpreter is deselected.

## 11.3 Manually starting the Submit interpreter

<b>Precondition</b>	<ul style="list-style-type: none"> <li>■ User group "Expert"</li> <li>■ Operating mode T1 or T2</li> <li>■ Submit interpreter has been stopped or deselected.</li> </ul>
<b>Procedure</b>	<ul style="list-style-type: none"> <li>■ In the main menu, select <b>Configuration &gt; SUBMIT interpreter &gt; Start/ select.</b></li> </ul>
<b>Alternative procedure</b>	<ul style="list-style-type: none"> <li>■ In the status bar, touch the <b>Submit interpreter</b> status indicator. A window opens. Select <b>Start/select.</b></li> </ul>
<b>Description</b>	<p>If the Submit interpreter is deselected, the command <b>Start&gt;Select</b> selects the SUB program defined in the file \$custom.dat.</p> <p>If the Submit interpreter has been stopped, the command <b>Start&gt;Select</b> resumes the selected SUB program at the point at which it was stopped.</p> <p>Once the Submit interpreter has been started, the corresponding icon in the status bar is green.</p>

Icon	Color	Description
	Yellow	Submit interpreter is selected. The block pointer is situated on the first line of the selected SUB program.
	Green	Submit interpreter is running.

## 11.4 Modifying the program SPS.SUB

- Precondition**
- The program SPS.SUB is not selected or has been stopped.
  - User group “Expert”

- Procedure**
1. Select the program SPS.SUB in the Navigator and press **Open**.
  2. Enter changes.
    - Enter initializations in the USER INIT Fold. This Fold is located in theINI Fold.
    - Enter all other changes in the USER PLC Fold.
  3. Close the program. Respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.
  4. The program SPS.SUB can now be started via the main menu with **Configuration > SUBMIT interpreter > Start**.

- Description** Structure of the program SPS.SUB:

```

1 DEF SPS ( )
2 DECLARATIONS
3INI
4
5 LOOP
6   WAIT FOR NOT ($POWER_FAIL)
7   TORQUE_MONITORING()
8
9 USER PLC
10 ENDLOOP

```

Line	Description
2	Declaration section
3	Initialization section. For statements that are only to be executed once after the system has booted.
5 ... 10	LOOP statement. For programs that are to run continuously in the background.
9	Fold for user-specific adaptations

The program SPS.SUB may also contain Folds for user-specific adaptations for technology packages, e.g. the Fold GRIPPERTECH PLC. The Folds that are actually present depend on what technologies are installed.

## 11.5 Creating a new SUB program

- Precondition**
- Expert user group

- |                    |  |
|--------------------|--|
| <b>Procedure</b>   | <ol style="list-style-type: none"> <li>1. In the directory structure, select the folder in which the program is to be created. (Not all folders allow the creation of programs within them.)</li> <li>2. Press <b>New</b>.<br/>The <b>Template selection</b> window is opened.</li> <li>3. Select the template <b>Submit</b> or <b>Expert Submit</b> and confirm with <b>OK</b>.</li> <li>4. Enter a name for the program and confirm it with <b>OK</b>.</li> </ol>  |
| <b>Description</b> | <p>Newly created SUB programs contain no Folds for the installed technology packages.</p> <p>These Folds are present by default in the program SPS.SUB. If the program SPS.SUB is to be replaced with a new SUB program, these Folds must be copied into the new program. The technology packages are otherwise no longer fully operational.</p> <ul style="list-style-type: none"> <li>■ The template <b>Submit</b> generates a SUB file with the following structure:</li> </ul> <pre style="border: 1px solid black; padding: 5px; margin-top: 10px;">1 DECLARATIONS 2INI 3 4LOOP 5USER PLC 6ENDLOOP 7USER SUBROUTINE</pre> |
- 
- | Line    | Description   |
|---------|---|
| 1       | Declaration section   |
| 2       | Initialization section. For statements that are only to be executed once after the system has booted.                 |
| 4, 5, 6 | LOOP statement containing the Fold USER PLC. The Fold is for programs that are to run continuously in the background. |
| 7       | For user-specific subroutines   |
- 
- The template **Expert Submit** generates an empty SUB file. With this template, everything has to be programmed by the user.



Use a LOOP statement when programming. SUB programs without a LOOP statement are only executed once by the Submit interpreter. It is then automatically deselected.

## 11.6 Programming

- |                 |  |
|-----------------|--|
| <b>KRL code</b> | <p>Almost all KRL instructions can be used in a SUB program. The following statements are not possible, however:</p> <ul style="list-style-type: none"> <li>■ Statements for robot motions</li> </ul> <p>Robot motions can only be interpreted by the robot interpreter. For this reason, SRC programs containing motion commands cannot be called as subprograms from a SUB program.</p> <ul style="list-style-type: none"> <li>■ Statements referring to robot motions</li> </ul> <p>These include BRAKE and all TRIGGER statements.</p> |
|-----------------|--|

Motion commands for external axes can be used in a SUB program. Example:

```
IF (($IN[12] == TRUE) AND ( NOT $IN[13] == TRUE)) THEN
ASYPTP {E2 45}
ASYPTP {E3 200}
...
IF ((NOT $IN[12] == TRUE) AND ($IN[13] == TRUE)) THEN
ASYPTP {E2 0}
ASYPTP {E3 90}
```

External axes E2 and E3 are moved in accordance with specific inputs.

WAIT statements or wait loops stop the cycle.

### System variables

The Submit interpreter has read-access to all system variables and write-access to many of them. Access works even if the system variables are being used in parallel by a motion program.

If a system variable to which the Submit interpreter does not have write-access is modified in a SUB program, an error message is generated when the program is started and the Submit interpreter stops.

System variables that are frequently required in SUB programs:

\$MODE_OP = Value	
Value	Description
#T1	Robot controller is in T1 mode.
#T2	Robot controller is in T2 mode.
#AUT	Robot controller is in Automatic mode.
#EX	Robot controller is in Automatic External mode.
#INVALID	Robot controller has no defined state.

\$OV_PRO = Value		
Element	Data type	Description
Value (%)	INT	Program override value

Example:

If the programmed velocity is not reached, output 2 is set to FALSE.

```
...
IF (( $MODE_OP == #T1) OR ($OV_PRO < 100)) THEN
$OUT[2] = FALSE
ENDIF
...
```



**WARNING** In the test modes, \$OV\_PRO must not be written to by the Submit interpreter, because the change may be unexpected for operators working on the industrial robot. Death to persons, severe physical injuries and considerable damage to property may result.



**WARNING** If possible, do not modify safety-relevant signals and variables (e.g. operating mode, EMERGENCY STOP, safety gate contact) via the Submit interpreter. If modifications are nonetheless required, all safety-relevant signals and variables must be linked in such a way that they cannot be set to a dangerous state by the Submit interpreter or PLC.

### Inputs/outputs

The Submit interpreter can access the inputs and outputs of the robot controller.



**WARNING** No check is made to see if the robot interpreter and Submit interpreter are accessing the same output simultaneously, as this may even be desired in certain cases. The user must therefore carefully check the assignment of the outputs. Otherwise, unexpected output signals may be generated, e.g. in safety equipment. Death, serious physical injuries or major damage to property may result.

### Subprograms

Other programs can be called as subprograms in a SUB program. The following are possible:

- Other SUB programs

- SRC programs without statements for robot motions

Example:

CELL.SRC can be called from the program SPS.SUB with a CWRITE statement and RUN. The call only takes effect in the case of a cold start.

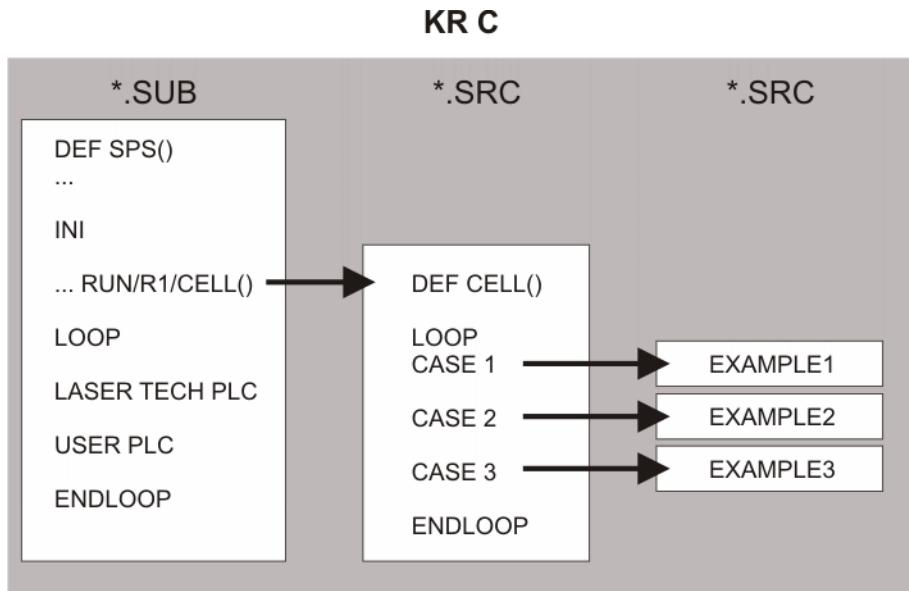


Fig. 11-1: SPS.SUB calls CELL.SRC



Further information about the program CELL.SRC can be found in this documentation.

(>>> 6.10.1 "Configuring CELL.SRC" Page 159)

Further information about CWRITE statements can be found in the Expert documentation CREAD/CWRITE.

### Communication

The flags of the robot controller can be used to enable the exchange of binary information between a running motion program and a SUB program. A flag is set by the Submit interpreter and read by the robot interpreter.

## 12 Diagnosis

### 12.1 Logbook

#### 12.1.1 Displaying the logbook

The operator actions on the KCP are automatically logged. The command **Logbook** displays the logbook.

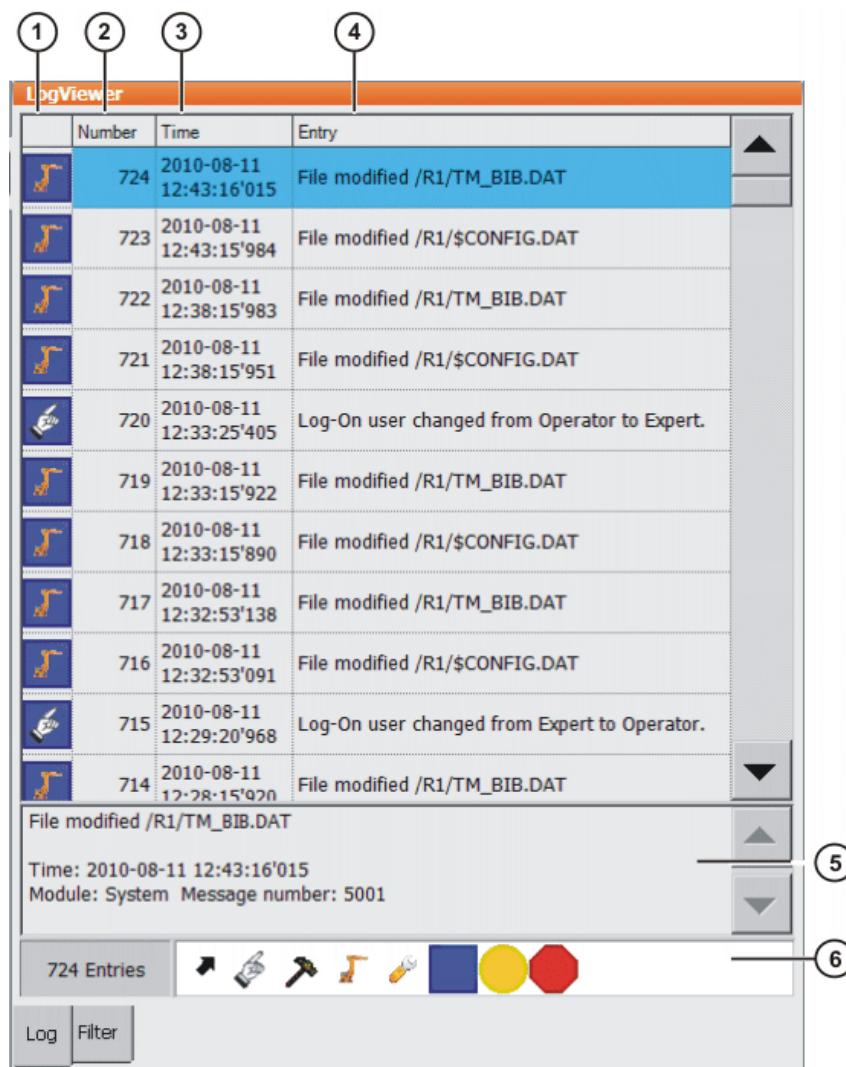
##### Procedure

- In the main menu, select **Diagnosis > Logbook > Display**.

The following tabs are available:

- **Log** ([>>> 12.1.2 "Log" tab Page 357](#))
- **Filter** ([>>> 12.1.3 "Filter" tab Page 358](#))

#### 12.1.2 "Log" tab



**Fig. 12-1: Logbook, Log tab**

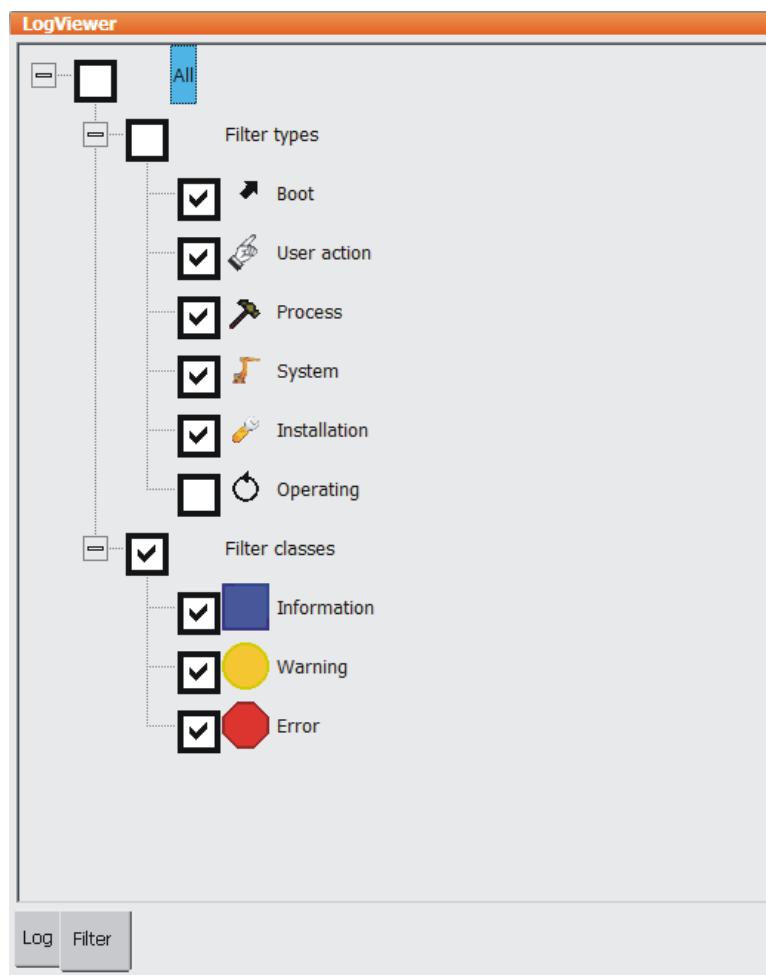
Item	Description
1	Type of log event  Example  Filter type "Note" + filter class "System" = note originated by the kernel system of the robot. The individual filter types and filter classes are listed on the <b>Filter</b> tab.
2	Log event number
3	Date and time of the log event
4	Brief description of the log event
5	Detailed description of the selected log event
6	Indication of the active filter

The following buttons are available:

Button	Description
<b>Export</b>	Exports the log data as a text file. (>>> 12.1.4 "Configuring the logbook" Page 359)
<b>Update</b>	Refreshes the log display.

### 12.1.3 “Filter” tab

The **Filter** tab is only displayed if the log **User-defined** has been selected.



**Fig. 12-2: Logbook, Filter tab**

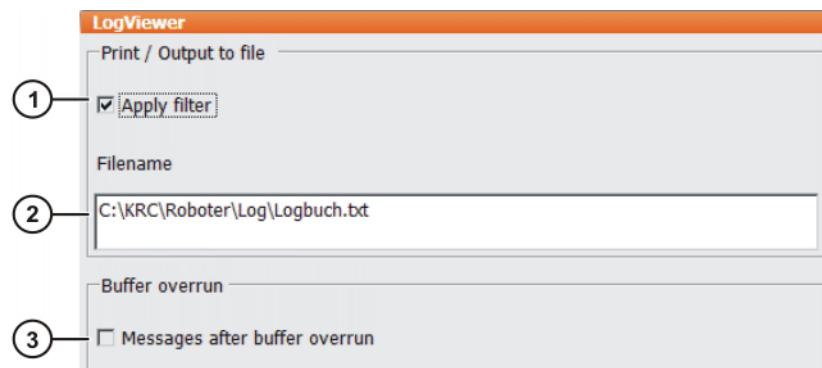
#### 12.1.4 Configuring the logbook

**Precondition** ■ Expert user group

**Procedure**

1. In the main menu, select **Diagnosis > Logbook > Configuration**. A window opens.
2. Make the desired settings.
3. Press **OK** to save the configuration and close the window.

**Description**



**Fig. 12-3: Logbook configuration window**

Item	Description
1	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> the log events selected with the filter are saved in the text file.</li> <li>■ <b>Check box not active:</b> all log events are saved in the text file.</li> </ul>
2	Enter the path and name of the text file. Default path: C:\KRC\ROBOTER\LOG\LOGBUCH.TXT
3	<ul style="list-style-type: none"> <li>■ <b>Check box active:</b> log data deleted because of a buffer overflow are indicated in gray in the text file.</li> <li>■ <b>Check box not active:</b> log data deleted because of a buffer overflow are not indicated in the text file.</li> </ul>

## 12.2 Displaying the caller stack

This function displays the data for the process pointer (\$PRO\_IP).

### Precondition

- User group "Expert"
- Program is selected.

### Procedure

- In the main menu, select **Diagnosis > Caller stack**.

### Description

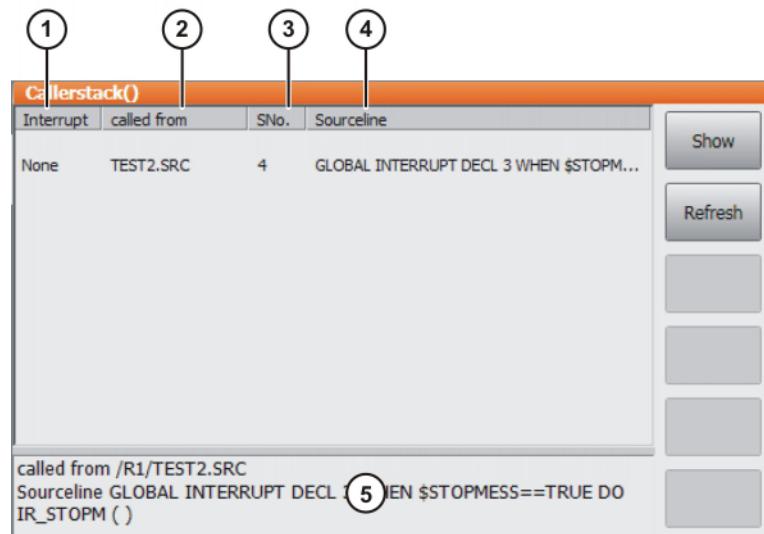
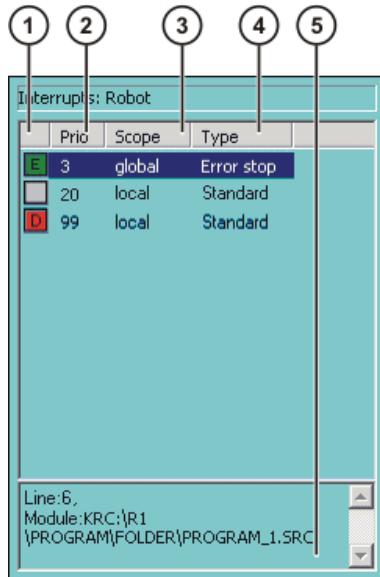


Fig. 12-4: Caller Stack window

Item	Description
1	<ul style="list-style-type: none"> <li>■ <b>None:</b> Call not initiated by interrupt</li> <li>■ <b>[No.]:</b> Call initiated by interrupt with the number [No.]</li> </ul>
2	This file contains the call.
3	The program line with this number contains the call. Preconditions in the program for the correct line to be determined using the number: <ul style="list-style-type: none"> <li>■ Detail view (ASCII mode) is activated.</li> <li>■ All Point PLCs are open.</li> </ul>
4	Source line
5	Detailed information about the entry selected in the list

## 12.3 Displaying interrupts

- Assumption** ■ “Expert” user group
- Procedure** ■ In the main menu, select **Display > Diagnosis > Interrupts**.
- Description**



**Fig. 12-5: Interrupts**

Item	Description
1	Status of the interrupt <ul style="list-style-type: none"> <li>■ <b>E</b> Interrupt ON or ENABLE</li> <li>■ <b>D</b> Interrupt DISABLE</li> <li>■ <b> </b> Interrupt OFF or not activated</li> </ul>
2	Number/priority of the interrupt
3	Validity range of the interrupt: global or local
4	Type of interrupt, dependent on the defined event in the interrupt declaration <ul style="list-style-type: none"> <li>■ <b>Default:</b> e.g. \$IN[1...4096]</li> <li>■ <b>Error stop:</b> \$STOPMESS</li> <li>■ <b>EMERGENCY STOP:</b> \$ALARM_STOP</li> <li>■ <b>Measurement (Fast Measurement):</b> \$MEAS_PULSE[1...5]</li> <li>■ <b>Trigger:</b> Trigger subprogram</li> </ul>
5	Module and program line of the interrupt declaration

The following buttons are available:

Button	Description
<b>Submit/Robot</b>	Toggles between the displays for robot interrupts and Submit interrupts.
<b>Update</b>	Refreshes the display.

## 12.4 Displaying diagnostic data about the kernel system

<b>Description</b>	The menu item <b>Diagnostic Monitor</b> makes it possible to display a wide range of diagnostic data concerning numerous software modules of the kernel system.
	<p>Examples:</p> <ul style="list-style-type: none"><li>■ <b>Module Kcp3 Driver</b> (= drive for the smartPAD)</li><li>■ Network driver</li></ul> <p>The data displayed depend on the selected module. The display includes states, fault counters, message counters, etc.</p>

- Procedure**
1. Select **Diagnosis > Diagnostic monitor** in the main menu.
  2. Select a software module in the **Module** box.  
Diagnostic data are displayed for the selected module.

## 12.5 Compressing data for error analysis at KUKA

<b>Description</b>	If it is necessary for an error to be analyzed by KUKA Roboter GmbH, this procedure can be used to compress the data for sending to KUKA. The procedure generates a ZIP file in the directory C:\KUKA\KRCDiag. This contains the data required by KUKA Roboter GmbH to analyze an error (including information about system resources, screenshots, and much more.)
<b>Procedure via "Diagnosis"</b>	<ul style="list-style-type: none"><li>■ In the main menu, select <b>Diagnosis &gt; KrcDiag</b>. The data are compressed. Progress is displayed in a window. Once the operation has been completed, this is also indicated in the window. The window is then automatically hidden again.</li></ul>
<b>Procedure via smartPAD</b>	This procedure uses keys on the smartPAD instead of menu items. It can thus also be used if the smartHMI is not available, due to Windows problems for example. <p><b>Precondition:</b></p> <ul style="list-style-type: none"><li>■ The smartPAD is connected to the robot controller.</li><li>■ The robot controller is switched on.</li></ul> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> The keys must be pressed within 2 seconds. Whether or not the main menu and keypad are displayed in the smartHMI is irrelevant.</div> <ol style="list-style-type: none"><li>1. Press the “Main menu” key and hold it down.</li><li>2. Press the keypad key twice.</li><li>3. Release the “Main menu” key. The data are compressed. Progress is displayed in a window. Once the operation has been completed, this is also indicated in the window. The window is then automatically hidden again.</li></ol>
<b>Procedure via "Archive"</b>	Alternatively, the data can also be compressed via <b>File &gt; Archive &gt; [...]</b> . In this way, the data can be stored on a USB stick or network path. (>>> 7.8 "Archiving and restoring data" Page 216)

## 13 Installation

The robot controller is supplied with a Windows operating system and an operational version of the KUKA System Software (KSS). Therefore, no installation is required during initial start-up.

Installation becomes necessary, for example, in the event of the hard drive being damaged and exchanged.



The robot controller may only be operated using the software provided with the controller by KUKA.

KUKA Roboter GmbH must be consulted if different software is to be used. ([>>> 15 "KUKA Service" Page 375](#))

### 13.1 Overview of the software components

**Overview** The following software components are used:

- KUKA System Software 8.2
- Windows XPe V3.0.0

### 13.2 Installing Windows and KSS (from image)

**Precondition**

- KUKA.USBSystem stick with an image of the robot controller  
The stick can be purchased from KUKA Roboter GmbH.
- 1 GB RAM
- The robot controller is switched off.



Only the KUKA.USBSystem stick can be used. Installation from a different stick is not possible.

**Procedure**

1. Connect the USB stick to the robot controller.
2. Switch on the robot controller. The Windows installation starts.
3. Disconnect the USB stick when all 6 LEDs on the CSP are permanently lit.  
The robot controller restarts.
4. Select the desired language. Confirm with **Next**.
5. Information about the installation and copyright are displayed. Confirm with **Next**.
6. Specify whether the robot controller is an Office PC. This is generally not the case. (Office PC = PC for KUKA personnel for tests.) Confirm with **Next**.
7. Only if a KSS version is already installed: It is possible to select whether data are to be retained from the existing installation. Confirm with **Next**.  
Or: If the suggested type does not correspond to the type that is being used, select a different type. Then confirm with **Next**.  
If, under step 7, the data from an existing installation have been retained, the system does not suggest a robot type, but retains the previous one.
9. A summary of the setup settings is displayed. Confirm with **Next**. The KSS installation starts.
10. A prompt to enter a unique computer name is displayed. A unique name is always suggested. The suggestion can be either accepted or altered. Confirm with **Run Now**.  
The robot controller restarts. When it has booted, installation is complete.



Following installation or update of the KUKA System Software, the robot controller always performs an initial cold start.

### 13.3 Installing additional software

This function can be used to install additional software, e.g. technology packages. New programs and updates can be installed. The software is installed from a USB stick. Alternatively, it can also be installed via a network path.

**NOTICE**

Use only a KUKA.USB data stick: if a different USB stick is used, data may be lost or changed.

The system checks whether the additional software is relevant for the KSS. If not, the system rejects the installation. If a software package that the system has rejected is nonetheless to be installed, KUKA Roboter GmbH must be contacted. (>>> 15 "KUKA Service" Page 375)

This function can also be used to uninstall additional software. Multiple additional programs can be installed or uninstalled one after the other.

This function is also used to select the path for a KSS update from the network. This function cannot be used, however, to install the KSS update.

(>>> 13.4 "KSS update" Page 365)

**Precondition**

- Software on KUKA.USBDATA stick
- No program is selected.
- T1 or T2 operating mode
- "Expert" user group

**Procedure**

1. Plug in USB stick.
2. Select **Start-up > Install additional software** in the main menu.
3. Press **New software**. If a software package that is on the USB stick is not yet displayed, press **Refresh**.
4. Select the software to be installed and press **Install**. Reply to the request for confirmation with **Yes**. The files are copied onto the hard drive.
5. Repeat step 4 if another software package is to be installed from this stick.
6. Remove the USB stick when the LED on the stick is no longer lit.
7. It may be necessary to reboot the controller, depending on the additional software. In this case, a corresponding prompt is displayed. Confirm with **OK** and reboot the robot controller. Installation is resumed and completed.

**Description**

The following buttons are available:

Button	Description
<b>New software</b>	All programs available for installation are displayed.
<b>Back</b>	Additional software already installed is displayed.
<b>Refresh</b>	Refreshes the display, e.g. after a USB stick has been connected.
<b>Install</b>	Displays additional buttons: <ul style="list-style-type: none"><li>■ <b>Yes</b>: The selected software is installed. If it is necessary to reboot the controller, this is indicated by a message.</li><li>■ <b>No</b>: The software is not installed.</li></ul>

Button	Description
<b>Configuration</b>	<p>This button is only displayed if <b>New software</b> has been pressed.</p> <p>Paths for the installation of additional software or KSS updates can be selected and saved here.</p> <p>Displays additional buttons:</p> <ul style="list-style-type: none"> <li>■ <b>Browse</b>: A new path can be selected.</li> <li>■ <b>Save</b>: Saves the displayed paths.</li> </ul>
<b>Uninstall</b>	<p>Displays additional buttons:</p> <ul style="list-style-type: none"> <li>■ <b>Yes</b>: The selected software is uninstalled.</li> <li>■ <b>No</b>: The software is not uninstalled.</li> </ul>

## 13.4 KSS update

**Description** This function can be used to install KSS updates, e.g. from KSS 8.2.0 to KSS 8.2.1.

It is advisable to archive all relevant data before updating a software package. If necessary, the old version can be restored in this way. It is also advisable to archive the new version after carrying out the update.



This function cannot be used to install new versions, e.g. from KSS 8.1 to KSS 8.2. KUKA Roboter GmbH must be consulted before a new version is installed.

(>>> 15 "KUKA Service" Page 375)

This function cannot be used to install updates of additional software, such as technology packages.

(>>> 13.3 "Installing additional software" Page 364)



**WARNING** If machine data are reloaded after an update, the version of the machine data must correspond exactly to the KSS version. This is ensured if the machine data supplied together with the KSS release are used.

The robot must not be moved if incorrect machine data are loaded. Personal injuries or damage to property may result in this case.

### Overview

There are 2 ways of installing a KSS update:

- From KUKA.USBData stick  
(>>> 13.4.1 "Update from the KUKA USB stick" Page 365)
- From the network  
(>>> 13.4.2 "Update from the network" Page 366)



Following installation or update of the KUKA System Software, the robot controller always performs an initial cold start.

### 13.4.1 Update from the KUKA USB stick



**NOTICE** Use only a KUKA.USB data stick: if a different USB stick is used, data may be lost or changed.

#### Precondition

- Software on KUKA.USBData stick

The software must be copied onto the stick with the file Setup.exe at the highest level (i.e. not in a folder).

- No program is selected.
- T1 or T2 operating mode

<b>Procedure</b>	<ol style="list-style-type: none"><li>1. Plug in USB stick.</li><li>2. In the main menu, select <b>Start-up &gt; Software update &gt; Automatic</b>.</li><li>3. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing <b>Yes</b>.</li><li>4. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.</li><li>5. Wait until the computer has shut down completely.</li><li>6. Remove the USB stick when the LED on the stick is no longer lit.</li><li>7. Switch the controller back on</li></ol> <p>Once the update has been completed, the computer is automatically shut down and rebooted.</p>
------------------	---

### 13.4.2 Update from the network

<b>Description</b>	<p>In the case of an update from the network, the installation data are copied from the network to the local drive D:\. If there is already a copy of a system software version present on drive D:\, that copy will now be overwritten.</p> <p>Installation is started on completion of the copying operation.</p>
<b>Precondition</b>	<p>For the preparation:</p> <ul style="list-style-type: none"><li>■ No program is selected.</li><li>■ T1 or T2 operating mode</li><li>■ “Expert” user group</li></ul> <p>For the procedure:</p> <ul style="list-style-type: none"><li>■ No program is selected.</li><li>■ T1 or T2 operating mode</li></ul>
<b>Preparation</b>	<p>Configure the network path from which the update installation is to be carried out:</p> <ol style="list-style-type: none"><li>1. Select <b>Start-up &gt; Install additional software</b> in the main menu.</li><li>2. Press <b>New software</b>.</li><li>3. Press <b>Configuration</b>.</li><li>4. Select the <b>Installation path for KRC update via the network</b> box. Press <b>Browse</b>.</li><li>5. Select the desired network path. Press <b>Save</b>.</li><li>6. The selected path is displayed in the <b>Installation path for KRC update via the network</b> box. Press <b>Save</b>.</li><li>7. Close the window.</li></ol>
<b>Procedure</b>	<p> It is only necessary to configure the network path once. It remains saved for subsequent updates.</p> <ol style="list-style-type: none"><li>1. In the main menu, select <b>Start-up &gt; Software update &gt; Net</b>.</li><li>2. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing <b>Yes</b>.  Depending on the network utilization, the procedure may take up to 15 min.</li><li>3. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.</li></ol>

4. Wait until the computer has shut down completely. Then switch the controller back on.
5. Once the update has been completed, the computer is automatically shut down and rebooted.



## 14 Messages

### 14.1 Automatic External error messages

No.	Message text	Cause
P00:1	PGNO_TYPE incorrect value permissible values (1,2,3)	The data type for the program number was entered incorrectly.
P00:2	PGNO_LENGTH incorrect value Range of values $1 \leq \text{PGNO\_LENGTH} \leq 16$	The selected program number length in bits was too high.
P00:3	PGNO_LENGTH incorrect value permissible values (4,8,12,16)	If BCD format was selected for reading the program number, a corresponding number of bits must also be set.
P00:4	PGNO_FBIT incorrect value not in the \$IN range	The value "0" or a non-existent input was specified for the first bit of the program number.
P00:7	PGNO_REQ incorrect value not in the \$OUT range	The value "0" or a non-existent output was specified for the output via which the program number is to be requested.
P00:10	Transmission error incorrect parity	Discrepancy detected when checking parity. A transmission error must have occurred.
P00:11	Transmission error incorrect program number	The higher-level controller has transferred a program number for which there is no CASE branch in the file CELL.SRC.
P00:12	Transmission error incorrect BCD encoding	The attempt to read the program number in BCD format led to an invalid result.
P00:13	Incorrect operating mode	The I/O interface output has not been activated, i.e. the system variable \$I_O_ACTCONF currently has the value FALSE. This can have the following causes: <ul style="list-style-type: none"> <li>■ The mode selector switch is not in the "Automatic External" position.</li> <li>■ The signal \$I_O_ACT currently has the value FALSE.</li> </ul>
P00:14	Move to Home position in operating mode T1	The robot has not reached the HOME position.
P00:15	Incorrect program number	More than one input set with "1 of n".

## 14.2 Brake test messages

No.	Message	Description
27001	Brake X has reached the wear limit	<p>Cause: The brake is worn. It will soon be identified as defective. Until then, the robot can be moved without restrictions.</p> <p>Remedy:</p> <ul style="list-style-type: none"> <li>■ Repeat the brake test for confirmation. If the same result is obtained, exchange the brake now and perform a brake test again.</li> <li>■ Or leave the message until the brake has been tested as defective and then exchange the brake</li> </ul>
27002	Cyclical check for brake test request not made	<p>Cause: The brake test has been requested. No brake test was performed within 2 hours of the request.</p> <p>Remedy:</p> <ul style="list-style-type: none"> <li>■ Integrate the program BrakeTestReq.SRC into the application program in such a way that it is cyclically called as a subprogram. If a brake test is requested, the robot detects this and performs the brake test immediately.</li> <li>■ Or acknowledge the motion enables and perform a brake test.</li> </ul>
27004	Brake test required	<p>Cause: The brake test cycle time has elapsed, the robot controller has been rebooted or the brake test has been requested externally.</p> <p>Remedy: Perform brake test.</p>
27007	Insufficient holding torque of brake X	<p>Cause: The brake for axis X is defective.</p> <p>Remedy: If no additional message is displayed for which a different remedy is specified, exchange the brake and perform a brake test again.</p>
27009	Brake X OK	Cause: Brake X has been tested and is OK.
27010	Unable to verify performance of brake X	<p>Cause: The start position of the brake test is not suitable for testing the performance of the brake.</p> <p>Remedy: Change the start position.</p>
27011	Test of brake X not completed	<p>Cause: The test was aborted, either by the operator or due to a fault not attributable to the brake test.</p> <p>Remedy: Repeat the test.</p>
27012	Brake test successful	<p>Cause: No brake is defective. Message 27009 has been displayed for all brakes.</p> <p>It is possible that one or more brakes may have reached the wear limit. This is also indicated by message 27001.</p>
27017	Friction impedes verification of brake X	<p>May be generated as an additional message with 27007 or 27023.</p> <p>Cause: The brake cannot be tested due to friction.</p> <p>Remedy: Warm up the robot and repeat the test.</p>
27018	No application of brake X observed	<p>May be generated as an additional message with 27007.</p> <p>Cause: The brake application time is significantly longer than specified.</p>

No.	Message	Description
27019	Full torque of brake X could not be verified, but holding torque is reached	<p>May be generated as an alternative to 27009.</p> <p>Cause: Brake X has been tested and is OK. The braking torque in the machine data is greater than the required holding torque and cannot be verified due to the limited power of the drive.</p> <p>Remedy: No remedy required.</p>
27020	Asynchronous axes prevent execution of brake test	<p>Cause: At least one axis has been switched to asynchronous mode with \$ASYNC_AXIS or \$EX_AXIS_ASYNC.</p> <p>Remedy:</p> <ul style="list-style-type: none"> <li>■ In the case of \$EX_AX_ASYNC, exclude the axis from the test.</li> <li>■ In the case of \$ASYNC_AXIS, the cyclical call of the brake test must be shifted to a point at which all axes are switched to synchronous mode.</li> </ul>
27021	Simulated axes prevent execution of brake test	<p>Cause: At least one axis has been switched to simulation with \$SIMULATED_AXIS.</p> <p>Remedy: Deactivate axis simulation.</p>
27022	T1 mode prevents brake test execution	<p>Cause: The test cannot be executed with the required velocity.</p> <p>Remedy: Change operating mode.</p>
27023	Automatic safety factor reduction for brake X	<p>Cause:</p> <ul style="list-style-type: none"> <li>■ The start position of the brake test is not suitable for early detection of brake wear.</li> <li>■ Or unfavorable ratio between braking torque and rated motor torque.</li> </ul> <p>Effect: Worn brakes may be detected later than usual.</p> <p>Remedy:</p> <ul style="list-style-type: none"> <li>■ If worn brakes are to be detected as early as possible, change the start position of the test.</li> <li>■ If it is not relevant whether or not worn brakes are detected as early as possible, no remedy is required.</li> </ul>
---	Maximum motion of axis X exceeded	<p>Cause: During the brake test, the robot exceeded the maximum motion range of axis X.</p> <p>Remedy: Contact KUKA Roboter GmbH.</p>
---	If you trigger a safety function (E-Stop, operating mode change, opening the safety gate) that causes the drives to be switched off, the robot may sag. Select strategy.	<p>Cause: One or more brakes have been detected as defective.</p> <p>Remedy:</p> <ul style="list-style-type: none"> <li>■ Press <b>Repeat</b> to repeat the brake test.</li> <li>■ Press <b>Park pos.</b> to move the robot to the parking position.</li> </ul>

No.	Message	Description
- - -	Performing manual brake test - please acknowledge	Cause: The program BrakeTestReq.SRC has been started manually. It must be confirmed that the test is to be performed.  Remedy: Confirm the message by pressing <b>Ackn..</b> . Press the Start key to resume the program and perform the test.
- - -	Performing self-test for brake test - please acknowledge	Cause: The program BrakeTestSelfTest.SRC has been started. It must be confirmed that the test is to be performed.  Remedy: Confirm the message by pressing <b>Ackn..</b> . Press the Start key to resume the program and perform the test.

### 14.3 Error messages of the positionally accurate robot model

No.	Message	Description
272	No robot number programmed	Cause: The file ROBOTINFO.XML is missing on the RDC, e.g. after exchange of the RDC memory chip.  Remedy: Create the file ROBOTINFO.XML on the RDC again.
276	Wrong machine data for this robot type	Cause: The machine data loaded in the robot controller do not correspond to the robot type. The robot must not be moved.  Remedy: Load the correct machine data.
384	Not enough memory: Cannot load precision robot model	Cause: It was not possible to create the positionally accurate robot model due to insufficient memory capacity. Positioning accuracy is not possible.  Remedy: Increase the memory capacity.
417	Absolute accuracy model not active	Cause: The positionally accurate robot model was deactivated.  Remedy: Contact KUKA Roboter GmbH. ( <b>&gt;&gt;&gt; 15 "KUKA Service"</b> Page 375)
1446	Value assignment inadmissible	Cause: <ul style="list-style-type: none"><li>■ A system variable was assigned an inadmissible value, e.g. \$SPEED.ORI1 ≤ 0.0 or &gt;\$SPEED_MA.ORI1.</li><li>■ An inadmissible value was assigned to \$BASE or \$TOOL.</li></ul> Remedy: Check the system variables and assign a correct value. Acknowledge the message.
2972	Checksum error File %1	Cause: Error in the file indicated: The parameters are not applied.  Remedy: Install the correct file.
2973	Precision Robot Model Parameterfile wrong Serial-number	Cause: The XPid file PREC_ROBXXXXXXXXXX.XML contains a serial number that does not match the robot. The positionally accurate robot model cannot be initialized.  Remedy: <ul style="list-style-type: none"><li>■ Use the correct XPid file.</li><li>■ Use the correct serial number.</li></ul>

No.	Message	Description
2974	Precision robot: Model parameter not consistent with machine data	Cause: The machine data have been modified. The positionally accurate robot model cannot be initialized.  Remedy: Recalibrate the robot with KUKA.XROB RCS.
3047	Error loading high precision model. Reason code 0x1011	Cause: The positionally accurate robot model could not be loaded during the update because the robot moved.  Remedy: Stop the robot and repeat the update.
	Error loading high precision model. Reason code 0x1012	Cause: The positionally accurate robot model could not be accepted during loading because the robot moved.  Remedy: Stop the robot and load the model.
	Error loading high precision model. Reason code 0x1013	Cause: The positionally accurate robot model could not be accepted because it was not possible to allocate the memory.  Remedy: Reboot the robot controller with a cold restart and load the model.



## 15 KUKA Service

### 15.1 Requesting support

**Introduction** The KUKA Roboter GmbH documentation offers information on operation and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

**Information** The following information is required for processing a support request:

- Model and serial number of the robot
- Model and serial number of the controller
- Model and serial number of the linear unit (if applicable)
- Version of the KUKA System Software
- Optional software or modifications
- Archive of the software  
For KUKA System Software V8: instead of a conventional archive, generate the special data package for fault analysis (via **KrcDiag**).
- Application used
- Any external axes used
- Description of the problem, duration and frequency of the fault

### 15.2 KUKA Customer Support

**Availability** KUKA Customer Support is available in many countries. Please do not hesitate to contact us if you have any questions.

**Argentina** Ruben Costantini S.A. (Agency)  
Luis Angel Huergo 13 20  
Parque Industrial  
2400 San Francisco (CBA)  
Argentina  
Tel. +54 3564 421033  
Fax +54 3564 428877  
[ventas@costantini-sa.com](mailto:ventas@costantini-sa.com)

**Australia** Headland Machinery Pty. Ltd.  
Victoria (Head Office & Showroom)  
95 Highbury Road  
Burwood  
Victoria 31 25  
Australia  
Tel. +61 3 9244-3500  
Fax +61 3 9244-3501  
[vic@headland.com.au](mailto:vic@headland.com.au)  
[www.headland.com.au](http://www.headland.com.au)

<b>Belgium</b>	KUKA Automatisering + Robots N.V. Centrum Zuid 1031 3530 Houthalen Belgium Tel. +32 11 516160 Fax +32 11 526794 <a href="mailto:info@kuka.be">info@kuka.be</a> <a href="http://www.kuka.be">www.kuka.be</a>
<b>Brazil</b>	KUKA Roboter do Brasil Ltda. Avenida Franz Liszt, 80 Parque Novo Mundo Jd. Guançã CEP 02151 900 São Paulo SP Brazil Tel. +55 11 69844900 Fax +55 11 62017883 <a href="mailto:info@kuka-roboter.com.br">info@kuka-roboter.com.br</a>
<b>Chile</b>	Robotec S.A. (Agency) Santiago de Chile Chile Tel. +56 2 331-5951 Fax +56 2 331-5952 <a href="mailto:robotec@robotec.cl">robotec@robotec.cl</a> <a href="http://www.robotec.cl">www.robotec.cl</a>
<b>China</b>	KUKA Automation Equipment (Shanghai) Co., Ltd. Songjiang Industrial Zone No. 388 Minshen Road 201612 Shanghai China Tel. +86 21 6787-1808 Fax +86 21 6787-1805 <a href="mailto:info@kuka-sha.com.cn">info@kuka-sha.com.cn</a> <a href="http://www.kuka.cn">www.kuka.cn</a>
<b>Germany</b>	KUKA Roboter GmbH Zugspitzstr. 140 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 <a href="mailto:info@kuka-roboter.de">info@kuka-roboter.de</a> <a href="http://www.kuka-roboter.de">www.kuka-roboter.de</a>

<b>France</b>	KUKA Automatisme + Robotique SAS Techvallée 6, Avenue du Parc 91140 Villebon S/Yvette France Tel. +33 1 6931660-0 Fax +33 1 6931660-1 <a href="mailto:commercial@kuka.fr">commercial@kuka.fr</a> <a href="http://www.kuka.fr">www.kuka.fr</a>
<b>India</b>	KUKA Robotics India Pvt. Ltd. Office Number-7, German Centre, Level 12, Building No. - 9B DLF Cyber City Phase III 122 002 Gurgaon Haryana India Tel. +91 124 4635774 Fax +91 124 4635773 <a href="mailto:info@kuka.in">info@kuka.in</a> <a href="http://www.kuka.in">www.kuka.in</a>
<b>Italy</b>	KUKA Roboter Italia S.p.A. Via Pavia 9/a - int.6 10098 Rivoli (TO) Italy Tel. +39 011 959-5013 Fax +39 011 959-5141 <a href="mailto:kuka@kuka.it">kuka@kuka.it</a> <a href="http://www.kuka.it">www.kuka.it</a>
<b>Japan</b>	KUKA Robotics Japan K.K. Daiba Garden City Building 1F 2-3-5 Daiba, Minato-ku Tokyo 135-0091 Japan Tel. +81 3 6380-7311 Fax +81 3 6380-7312 <a href="mailto:info@kuka.co.jp">info@kuka.co.jp</a>
<b>Korea</b>	KUKA Robotics Korea Co. Ltd. RIT Center 306, Gyeonggi Technopark 1271-11 Sa 3-dong, Sangnok-gu Ansan City, Gyeonggi Do 426-901 Korea Tel. +82 31 501-1451 Fax +82 31 501-1461 <a href="mailto:info@kukakorea.com">info@kukakorea.com</a>

<b>Malaysia</b>	KUKA Robot Automation Sdn Bhd South East Asia Regional Office No. 24, Jalan TPP 1/10 Taman Industri Puchong 47100 Puchong Selangor Malaysia Tel. +60 3 8061-0613 or -0614 Fax +60 3 8061-7386 <a href="mailto:info@kuka.com.my">info@kuka.com.my</a>
<b>Mexico</b>	KUKA de Mexico S. de R.L. de C.V. Rio San Joaquin #339, Local 5 Colonia Pensil Sur C.P. 11490 Mexico D.F. Mexico Tel. +52 55 5203-8407 Fax +52 55 5203-8148 <a href="mailto:info@kuka.com.mx">info@kuka.com.mx</a>
<b>Norway</b>	KUKA Sveiseanlegg + Roboter Bryggeveien 9 2821 Gjøvik Norway Tel. +47 61 133422 Fax +47 61 186200 <a href="mailto:geir.ulsrud@kuka.no">geir.ulsrud@kuka.no</a>
<b>Austria</b>	KUKA Roboter Austria GmbH Vertriebsbüro Österreich Regensburger Strasse 9/1 4020 Linz Austria Tel. +43 732 784752 Fax +43 732 793880 <a href="mailto:office@kuka-roboter.at">office@kuka-roboter.at</a> <a href="http://www.kuka-roboter.at">www.kuka-roboter.at</a>
<b>Poland</b>	KUKA Roboter Austria GmbH Spółka z ograniczoną odpowiedzialnością Oddział w Polsce Ul. Porcelanowa 10 40-246 Katowice Poland Tel. +48 327 30 32 13 or -14 Fax +48 327 30 32 26 <a href="mailto:ServicePL@kuka-roboter.de">ServicePL@kuka-roboter.de</a>

<b>Portugal</b>	KUKA Sistemas de Automatización S.A. Rua do Alto da Guerra n° 50 Armazém 04 2910 011 Setúbal Portugal Tel. +351 265 729780 Fax +351 265 729782 <a href="mailto:kuka@mail.telepac.pt">kuka@mail.telepac.pt</a>
<b>Russia</b>	OOO KUKA Robotics Rus Webnaja ul. 8A 107143 Moskau Russia Tel. +7 495 781-31-20 Fax +7 495 781-31-19 <a href="http://kuka-robotics.ru">kuka-robotics.ru</a>
<b>Sweden</b>	KUKA Svetsanläggningar + Robotar AB A. Odhners gata 15 421 30 Västra Frölunda Sweden Tel. +46 31 7266-200 Fax +46 31 7266-201 <a href="mailto:info@kuka.se">info@kuka.se</a>
<b>Switzerland</b>	KUKA Roboter Schweiz AG Industriestr. 9 5432 Neuenhof Switzerland Tel. +41 44 74490-90 Fax +41 44 74490-91 <a href="mailto:info@kuka-roboter.ch">info@kuka-roboter.ch</a> <a href="http://www.kuka-roboter.ch">www.kuka-roboter.ch</a>
<b>Spain</b>	KUKA Robots IBÉRICA, S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 8142-353 Fax +34 93 8142-950 <a href="mailto:Comercial@kuka-e.com">Comercial@kuka-e.com</a> <a href="http://www.kuka-e.com">www.kuka-e.com</a>

<b>South Africa</b>	Jendamark Automation LTD (Agency) 76a York Road North End 6000 Port Elizabeth South Africa Tel. +27 41 391 4700 Fax +27 41 373 3869 <a href="http://www.jendamark.co.za">www.jendamark.co.za</a>
<b>Taiwan</b>	KUKA Robot Automation Taiwan Co., Ltd. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 <a href="mailto:info@kuka.com.tw">info@kuka.com.tw</a> <a href="http://www.kuka.com.tw">www.kuka.com.tw</a>
<b>Thailand</b>	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 <a href="mailto:atika@ji-net.com">atika@ji-net.com</a> <a href="http://www.kuka-roboter.de">www.kuka-roboter.de</a>
<b>Czech Republic</b>	KUKA Roboter Austria GmbH Organisation Tschechien und Slowakei Sezemická 2757/2 193 00 Praha Horní Počernice Czech Republic Tel. +420 22 62 12 27 2 Fax +420 22 62 12 27 0 <a href="mailto:support@kuka.cz">support@kuka.cz</a>
<b>Hungary</b>	KUKA Robotics Hungaria Kft. Fö út 140 2335 Taksony Hungary Tel. +36 24 501609 Fax +36 24 477031 <a href="mailto:info@kuka-robotics.hu">info@kuka-robotics.hu</a>

**USA**

KUKA Robotics Corp.  
22500 Key Drive  
Clinton Township  
48036  
Michigan  
USA  
Tel. +1 866 8735852  
Fax +1 586 5692087  
[info@kukarobotics.com](mailto:info@kukarobotics.com)  
[www.kukarobotics.com](http://www.kukarobotics.com)

**UK**

KUKA Automation + Robotics  
Hereward Rise  
Halesowen  
B62 8AN  
UK  
Tel. +44 121 585-0800  
Fax +44 121 585-0900  
[sales@kuka.co.uk](mailto:sales@kuka.co.uk)



# Index

## Symbols

\_TYP 295  
 \_TYPE 296  
 #BSTEP 208  
 #CSTEP 208  
 #IGNORE 240  
 #ISTEP 208  
 #MSTEP 208  
 #PSTEP 208  
 \$ 290  
 \$ACCU\_STATE 80  
 \$ADAP\_ACC 152, 157  
 \$ADVANCE 208  
 \$ANIN 275  
 \$ANOUT 275  
 \$BRAKE\_SIG 181  
 \$BRAKES\_OK 195  
 \$BRAKETEST\_MONTIME 195  
 \$BRAKETEST\_REQ\_EX 195  
 \$BRAKETEST\_REQ\_INT 195  
 \$BRAKETEST\_WARN 195  
 \$BRAKETEST\_WORK 195  
 \$CIRC\_MODE 242, 308  
 \$CIRC\_TYPE 241  
 \$COLL\_ALARM 153  
 \$COLL\_ENABLE 153  
 \$COOLDOWN\_TIME 151  
 \$HOLDING\_TORQUE 183  
 \$IN 275  
 \$LDC\_CONFIG 129  
 \$LDC\_LOADED 128  
 \$LDC\_RESULT 129  
 \$ORI\_TYPE 226, 239  
 \$OUT 275  
 \$PAL\_MODE 87  
 \$PATHTIME 313  
 \$PRO\_IP 360  
 \$PRO\_MODE 208  
 \$ROBRUNTIME 78, 79  
 \$SPL\_VEL\_MODE 304, 313  
 \$SPL\_VEL\_RESTR 314  
 \$TOOL\_DIRECTION 102  
 \$TORQ\_DIFF 153, 157  
 \$TORQ\_DIFF2 153  
 \$TORQMON\_COM\_DEF 153  
 \$TORQMON\_DEF 153  
 \$TORQMON\_TIME 153, 157  
 \$TORQUE\_AXIS\_ACT 181, 183  
 \$TORQUE\_AXIS\_LIMITS 182  
 \$TORQUE\_AXIS\_MAX 182  
 \$TORQUE\_AXIS\_MAX\_0 182  
 \$WARMUP\_CURR\_LIMIT 151  
 \$WARMUP\_MIN\_FAC 151  
 \$WARMUP\_RED\_VEL 151  
 \$WARMUP\_SLEW\_RATE 151  
 \$WARMUP\_TIME 151

## Numbers

2004/108/EC 39  
 2006/42/EC 39  
 3-point method 110  
 89/336/EEC 39  
 95/16/EC 39  
 97/23/EC 39

**A**

ABC 2-point method 107  
 ABC World method 107  
 Accessories 15, 17  
 Actual position 67  
 Administrator 53  
 Advance run 208  
 Advance run stop 314  
 Analog inputs 321  
 Analog outputs 322  
 ANIN 321  
 ANOUT 276, 322  
 Applied norms and regulations 39  
 Approximate positioning 224, 256  
 Archiving overview 216  
 Archiving, logbook 218  
 Archiving, network 218  
 Archiving, to USB stick 217  
 Areas of validity 292  
 AUT and EXT Consistency 189  
 AUT and EXT consistency 189  
 Automatic External error messages 369  
 Automatic mode 35  
 Auxiliary point 224, 302, 303  
 Axis range 19  
 Axis range limitation 27  
 Axis range monitoring 28

**B**

Backward motion 211  
 Base calibration 109  
 BASE coordinate system 54, 109  
 Battery state 80  
 Block pointer 203  
 Block selection 211, 232  
 BRAKE 328  
 Brake defect 30  
 Brake test 191  
 Brake test cycle time 192  
 Brake test, function test 197  
 Brake test, messages 370  
 Brake test, programs 192  
 Brake test, signals 193, 194  
 Brake test, teaching positions 195  
 Brake, defective 196  
 BrakeTestBack.SRC 193, 196  
 BrakeTestPark.SRC 192, 196  
 BrakeTestReq.SRC 192, 197  
 BrakeTestSelfTest.SRC 193, 198  
 BrakeTestStart.SRC 193, 196

- Braking distance 19  
Branch, conditional 316
- C**
- Calibrating an external kinematic system 120
  - Calibration 102
  - Calibration points (menu item) 77
  - Calibration tolerances, defining 158
  - Calibration, base 109
  - Calibration, external TCP 113
  - Calibration, fixed tool 112
  - Calibration, linear unit 118
  - Calibration, root point, kinematic system 120
  - Calibration, tool 102
  - Calibration, TOOL kinematic system 124
  - Calibration, workpiece 112
  - Caller stack 360
  - Caller stack (menu item) 360
  - CASE 318
  - CAST\_FROM 343
  - CAST\_TO 343
  - CCLOSE 343
  - CE mark 18
  - CELL.SRC 212
  - CHANNEL 343
  - CIOCTL 343
  - CIRC 302
  - CIRC motion 253
  - CIRC\_REL 303
  - CIRC, motion type 224
  - Circular angle 261, 266, 302, 304
  - Circular motion 302, 303
  - Cleaning work 36
  - Close all FOLDs (menu item) 207
  - Cold start 52
  - Cold start, initial 49, 51, 52, 364, 365
  - Collision detection 151, 153
  - Collision detection (menu item) 153, 154
  - Collision detection, Automatic External 155
  - Collision detection, offset 154
  - Collision detection, system variables 152
  - Collision detection, variable 155
  - Comment 213
  - Comparison, data from kernel system and hard drive 188
  - Conditional branch 316
  - Configuration 133
  - Configuration (menu item) 142
  - Configuring CELL.SRC 159
  - Connecting cables 15, 17
  - Connection manager 42
  - Constants 293
  - CONTINUE 314
  - Continuous Path 223
  - Coordinate system for jog keys 46
  - Coordinate system for Space Mouse 45
  - Coordinate systems 54
  - Coordinate systems, angles 55
  - Coordinate systems, orientation 55
  - COPEN 343
  - Copy 215
  - Counterbalancing system 36
  - Counters, displaying 75
  - CP motions 223
  - CREAD 343
  - Creating a new folder 200
  - Creating a new program 201
  - Cut 215
  - CWRITE 343
- D**
- Danger zone 19
  - DAT 289
  - Data list 289
  - Data types 291
  - Data, restoring 219
  - DECL 293
  - Declaration of conformity 18
  - Declaration of incorporation 17, 18
  - Decommissioning 37
  - DEF line (menu item) 205
  - DEF line, displaying/hiding 205
  - DEFAULT 318
  - Deleting mastering 99
  - Description, signal declarations 194
  - Deselecting a program 202
  - Detail view (ASCII) (menu item) 206
  - Detail view, activating 205
  - Diagnosis 357
  - Diagnostic monitor (menu item) 362
  - Dial gauge 97
  - Directory structure 199
  - Display (menu item) 72
  - Displaying a variable, single 70, 71
  - Displaying the logbook 357
  - Displaying variables, in overview 72
  - Displaying, robot controller information 78
  - Displaying, robot information 78
  - Disposal 37
  - DISTANCE 333
  - DNS, configuring 138
  - Documentation, industrial robot 13
  - Drive bus 51
- E**
- EC declaration of conformity 18
  - Edit (button) 46
  - Editing a program 212
  - Editor 201
  - Electronic Mastering Device 92
  - ELSE 316
  - EMC Directive 18, 39
  - EMD 92
  - EMERGENCY STOP 42
  - EMERGENCY STOP button 25, 32
  - EMERGENCY STOP device 25, 30
  - EMERGENCY STOP, external 25, 32
  - EMERGENCY STOP, local 32
  - EN 60204-1 39
  - EN 61000-6-2 39
  - EN 61000-6-4 39
  - EN 614-1 39

- EN ISO 10218-1 39  
 EN ISO 12100-1 39  
 EN ISO 12100-2 39  
 EN ISO 13849-1 39  
 EN ISO 13849-2 39  
 EN ISO 13850 39  
 Enabling device 26, 30  
 Enabling device, external 26  
 Enabling switch 43  
 Enabling switches 26  
 ENDFOR 315  
 ENDIF 316  
 Endless loop 317  
 ENDLOOP 317  
 ENDSPLINE 305  
 ENDSWITCH 318  
 ENDWHILE 320  
 ENUM 295  
 Enumeration type 295  
 error messages, Automatic External 369  
 Error messages, positionally accurate robot 372  
 Event planner (menu item) 188  
 EXIT 314, 317  
 Exiting, KSS 49  
 External axes 17, 20, 67, 78  
 External kinematic system, calibration 120
- F**  
 F 328  
 FALSE 327  
 Faults 31  
 File list 199  
 File, renaming 201  
 Filter 200  
 Find 216  
 First mastering 92  
 Fixed tool, calibration 112  
 Flags, displaying 73, 74  
 FLANGE coordinate system 55, 103  
 Folder, creating 200  
 Folds 206  
 Folds, creating 214  
 Folds, displaying 206  
 Fonts 288  
 FOR 319  
 FOR ... TO ... ENDFOR 315  
 Function test 32  
 Functions 289
- G**  
 General safety measures 30  
 Global 292  
 GLOBAL (interrupt declaration) 329  
 GOTO 315  
 Guard interlock 24
- H**  
 HALT 316  
 Hardware, options 84  
 Hazardous substances 37  
 Header 199
- Hibernate 52  
 Higher motion profile 304  
 HOME position 205  
 HOV 60
- I**  
 I/O driver, reconfiguring 141  
 I/Os, reconfiguring 141  
 Identification plate 43  
 IF ... THEN ... ENDIF 316  
 Impact 153, 154  
 Increment 65  
 Incremental jogging 65  
 Indirect method 111  
 Industrial robot 15, 17  
 Info (menu item) 78  
 Inline forms 251  
 Inputs/outputs, analog 69, 275  
 Inputs/outputs, Automatic External 69, 160  
 Inputs/outputs, digital 67, 274  
 Installation 363  
 Intended use 17  
 INTERN.ZIP 217, 218  
 Interpolation mode 255, 264  
 INTERRUPT 329, 330  
 Interrupt 328  
 Interrupt program 329  
 Interrupts 361  
 Introduction 13  
 IP addresses 133
- J**  
 Jerk 259, 261, 264, 267  
 Jerk limitation 259, 261, 264, 267  
 Jog keys 42, 56, 61  
 Jog mode 27, 30  
 Jog mode "Jog keys" 58  
 Jog mode "Space Mouse" 58  
 Jog mode, activating 60  
 Jog override 60  
 Jogging, axis-specific 55, 61  
 Jogging, Cartesian 55, 61, 64  
 Jogging, external axes 66  
 Jogging, robot 55  
 Jump 315
- K**  
 KCP 19, 30, 41  
 Keyboard 42  
 Keyboard key 42  
 Keyboard, external 31  
 Keypad 47  
 Keywords 290  
 Kinematics group 46, 58  
 KLI, configuring 133  
 KRL syntax 287  
 KUKA Control Panel 41  
 KUKA Customer Support 78, 375  
 KUKA Line Interface, configuring 133  
 KUKA smartHMI 45  
 KUKA smartPAD 19, 41

KUKA.Load 126  
KUKA.LoadDataDetermination 126

**L**

Labeling 29  
Language 52  
Liability 17  
LIN 299  
LIN motion 252  
LIN\_REL 300  
LIN, motion type 223  
Line break (menu item) 206  
Linear motion 299, 300  
Linear unit 17, 117  
Load data 126  
Logbook 357  
Logbook, configuring 359  
Logic Consistency 190  
Logic consistency 189  
LOOP ... ENDLOOP 317  
Loss of mastering 92, 96  
Low Voltage Directive 18

**M**

Machine data 33, 78, 79, 83, 88, 365  
Machine data, copying 88  
Machinery Directive 18, 39  
Main menu, calling 48  
Maintenance 35, 129  
Manipulator 15, 17, 19, 22  
Manual mode 34  
Mastering 89  
Mastering after maintenance work 98  
Mastering marks 91  
Mastering methods 90  
Mechanical axis range limitation 27  
Mechanical end stops 27  
Messages 369  
Messages, brake test 370  
Minimizing smartHMI 48  
Mode selection 23, 24  
Modifying a logic instruction 286  
Modifying a variable 70  
Modifying coordinates 274  
Modifying motion parameters 274  
Modifying variables 72  
Module 289  
Monitoring, velocity 27  
Motion profile, conventional 304  
Motion programming, basic principles 223  
Motion types 223  
Motor, exchange 99  
Mouse, external 31

**N**

Name, archive 79  
Name, control PC 78  
Name, robot 78, 79  
Names 290  
Navigator 199  
Non-rejecting loop 317

Numeric entry, external TCP 114  
Numeric entry, external tool 125  
Numeric entry, linear unit 119  
Numeric entry, root point, kinematic system 122  
Numeric input, base 112  
Numeric input, tool 109

**O**

Offset 92, 95, 277  
OLDC 127  
Online load data check 127  
Online optimizing 189, 191  
Open all FOLDs (menu item) 207  
Opening a program 201  
Operating hours 79  
Operating hours meter 79  
Operating mode, changing 53  
Operation 41  
Operator 19, 21, 52  
Operator safety 23, 24, 30  
Operator safety acknowledgement 85  
Options 15, 17  
Orientation control 256  
Orientation control (spline) 262, 265, 268  
Orientation control, LIN, CIRC 226  
Orientation control, SPLINE 239  
OUT 275  
Output, analog 276  
Output, digital 275  
Overload 30  
Override 60, 209  
Override (menu item) 67  
Overview of the industrial robot 15

**P**

Palletizing robots 87, 104, 109  
Panic position 26  
Password, changing 143  
Paste 215  
PATH 336, 339  
Payload data 126  
Payload data (menu item) 127  
Performance Level 23  
Performing a manual brake test 196  
Peripheral contactor 85  
Personnel 20  
Pinning 219, 222  
Plant integrator 20  
PLC\_ROB\_STOP\_RELEASE() 345  
PLC\_ROB\_STOP() 345  
Point-to-point 223  
Point-to-point motion 297, 298  
Positionally accurate robot, checking activation 87  
Positionally accurate robot, error messages 372  
Positioner 17, 120  
POV 209  
Pre-mastering position 91  
Pressure Equipment Directive 36, 39  
Preventive maintenance work 36  
Printing a program 216

Priority 329, 335, 338, 340  
 Product description 15  
 PROFINET interface 134  
 Program lines, deleting 214  
 Program management 199  
 Program override 209  
 Program run mode, selecting 207  
 Program run modes 208  
 Program, closing 203  
 Program, creating 201  
 Program, deselecting 202  
 Program, opening 201  
 Program, selecting 201  
 Program, starting 210  
 Program, stopping 210, 212  
 Programmer 53  
 Programming, Expert 287  
 Programming, inline forms 251  
 Programming, KRL syntax 287  
 Programming, User 251  
 Project management (window) 221  
 Protective equipment 27  
 PTP 297  
 PTP motion 251  
 PTP\_REL 298  
 PTP, motion type 223  
 PULSE 276, 323  
 Pulse 276, 323  
 Pulse, path-related 285

**R**

Rating plate 83  
 RDC, exchange 99  
 Reaction distance 19  
 Recommissioning 32, 83  
 Reference mastering 98  
 Rejecting loop 320  
 Release device 28  
 Renaming a file 201  
 Renaming the base 117  
 Renaming the tool 117  
 Repair 35  
 REPEAT ... UNTIL 317  
 Replace 216  
 Resetting a program 211  
 RESUME 332  
 RETURN 327  
 ROB\_STOP\_RELEASE() 345  
 ROB\_STOP() 345  
 Robot controller 15, 17  
 Robot data (menu item) 78  
 ROBROOT coordinate system 54

**S**

Safe operational stop 19, 26  
 Safeguards, external 29  
 Safety 17  
 Safety controller 23  
 Safety functions 30  
 Safety functions, overview 23  
 Safety instructions 13

Safety STOP 0 19  
 Safety STOP 1 19  
 Safety STOP 2 19  
 Safety STOP 0 19  
 Safety STOP 1 19  
 Safety STOP 2 19  
 Safety stop, external 26, 27  
 Safety zone 19, 21, 22  
 Safety, general 17  
 SCIRC 307  
 SCIRC motion, programming 260  
 SCIRC segment, programming 265  
 SEC 320  
 Selecting a program 201  
 Selecting the base 60  
 Selecting the tool 60  
 Serial number 79  
 Service life 78  
 Service, KUKA Roboter 375  
 SET\_TORQUE\_LIMITS 177, 180  
 Shutdown (menu item) 49  
 SIGNAL 326  
 Signal diagrams 168  
 Signals, brake test 194  
 Signals, brake test 193  
 Simulation 35  
 Single (menu item) 70, 71, 157  
 Single point of control 37  
 Singularities 248  
 SLIN 306  
 SLIN motion, programming 258  
 SLIN segment, programming 265  
 smartHMI 16, 45  
 smartPAD 19, 41  
 Soft axes 174  
 Software 15, 17  
 Software components 15, 363  
 Software limit switches 27, 30, 99  
 Software limit switches, modifying 100  
 Space Mouse 42, 56, 61, 63, 64  
 Special characters 251  
 SPL 309  
 SPL segment, programming 265  
 SPLINE 339  
 SPLINE ... ENDSPLINE 305  
 Spline block, programming 262  
 Spline motion, orientation control 262, 265, 268  
 Spline, motion type 229  
 SPOC 37  
 SPS.SUB, modifying 353  
 SRC 289  
 SREAD 343  
 Stamp 213  
 Start backwards key 42  
 Start key 42, 43  
 Start type, KSS 49  
 Start types 52  
 Start-up 32, 83  
 Start-up mode 33  
 Start-up wizard 83  
 Starting a program, automatic 210

- Starting a program, backwards 211  
Starting a program, manual 210  
Starting Automatic External mode 212  
Starting the KSS 48  
Status 244  
Status bar 45, 46, 199  
Status keys 42  
STEP 315  
STOP 0 18, 20  
STOP 1 18, 20  
STOP 2 18, 20  
Stop category 0 20  
Stop category 1 20  
Stop category 2 20  
STOP key 42  
Stop reactions 22  
Stopping a program 210, 212  
Stopping distance 19, 22  
Stopping the robot 328, 345  
Storage 37  
Storage capacities 78  
String variable length after initialization 346  
String variable length in the declaration 346  
String variable, deleting contents 347  
String variables 346  
String variables, comparing contents 349  
String variables, copying 349  
String variables, extending 347  
String variables, searching 348  
STRUC 296  
Structure type 296  
Submit interpreter 47, 351  
Submit interpreter, modifying SPS.SUB 353  
Submit interpreter, starting 352  
Submit interpreter, stopping 352  
Subprograms 289  
Supplementary load data (menu item) 127  
Support request 375  
SWITCH ... CASE ... END SWITCH 318  
Switching action, path-related 280  
Switching on the robot controller 48  
SWRITE 343  
Symbols 288  
SYN OUT 280  
SYN PULSE 285  
System integrator 18, 20
- T**
- T1 20  
T1 and T2 Consistency 189  
T1 and T2 consistency 189  
T2 20  
TCP 102  
TCP, external 112  
Teach pendant 15, 17  
Teaching 274  
Technology packages 16, 78, 251  
Terms used, safety 18  
Time block 309  
TIME\_BLOCK 309  
Timers, displaying 76
- tm\_useraction 152  
TMx 155  
Tool calibration 102  
Tool Center Point 102  
TOOL coordinate system 54, 102  
Tool direction 102  
Tool, external 124  
Torque 153, 154  
Torque mode, diagnosis 181  
Torque mode, examples 176, 184  
Torque mode, overview 174  
Torque monitoring 156  
Torque monitoring (menu item) 157  
Touch screen 41, 47  
Trademarks 14  
Training 13  
Transport position 31  
Transportation 31  
TRIGGER 333, 336  
TRIGGER, for spline 268, 339  
Turn 244  
Turn-tilt table 17, 120  
Type, robot 78  
Type, robot controller 78
- U**
- Unmastering 99  
UNTIL 317  
Update 365  
USB connection 43  
Use, contrary to intended use 17  
Use, improper 17  
User 19, 20  
User group, changing 52  
User group, default 52  
User interface 45
- V**
- Variable overview, configuring 142  
VARSTATE() 71, 343  
Velocity 60, 209  
Velocity monitoring 27  
Version, kernel system 78  
Version, operating system 78  
Version, robot controller 78  
Version, user interface 78  
Voltage 69, 275, 277
- W**
- WAIT 278, 319, 320  
WAIT FOR 319  
Wait function, signal-dependent 278  
WAIT SEC 320  
Wait time 278, 320  
WAITFOR 278  
Warm-up 149  
Warnings 13  
WHILE ... END WHILE 320  
Windows interface 133, 134  
Working range limitation 27  
Workpiece base calibration 122

Workpiece base, numeric entry 124  
Workspace 19, 21, 22  
Workspace monitoring, bypassing 66  
Workspaces, axis-specific 143  
Workspaces, Cartesian 143  
Workspaces, cubic 143  
Workspaces, mode 148  
WORLD coordinate system 54  
Wrist root point 149

**X**

XYZ 4-point method 104  
XYZ Reference method 106



