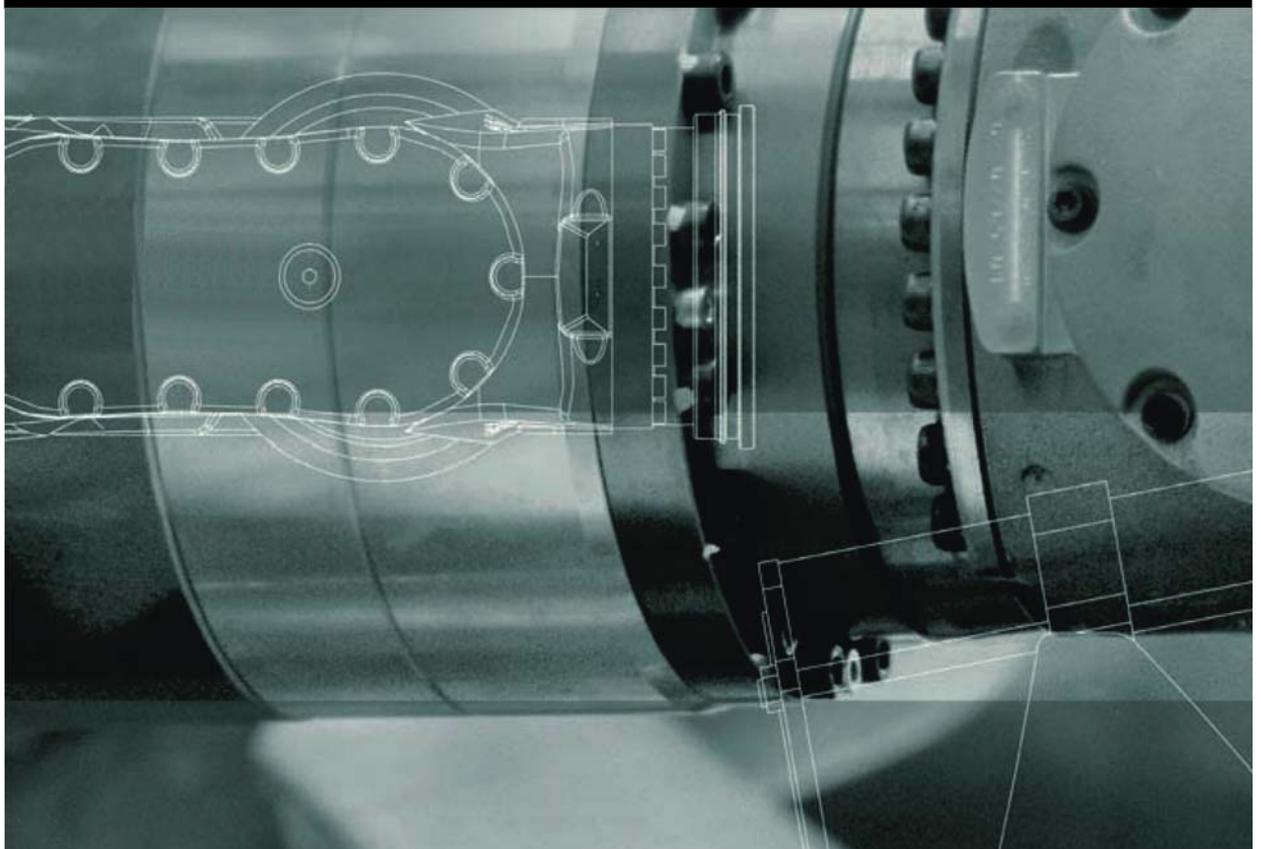


VW System Software

KUKA Roboter GmbH

VW System Software 8.1

Operating and Programming Instructions for System Integrators



Issued: 01.09.2010

Version: VSS 8.1 SI V2 en

© Copyright 2010

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 VSS 8.1 SI en
Bookstructure:	VSS 8.1 SI V2.3
Label:	VSS 8.1 SI V2 en

Contents

1	Introduction	13
1.1	Target group	13
1.2	Industrial robot documentation	13
1.3	Representation of warnings and notes	13
1.4	Trademarks	13
2	Product description	15
2.1	Overview of the industrial robot	15
2.2	Overview of the software components	15
2.3	Overview of VW System Software (VSS)	15
3	Safety	17
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	26
3.5.8	Enabling device	26
3.5.9	External enabling device	27
3.5.10	External safety stop 1 and external safety stop 2	27
3.5.11	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	28
3.6.4	Mechanical axis range limitation (optional)	28
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	32
3.8.3	Start-up and recommissioning	32
3.8.3.1	Start-up mode	34

3.8.4	Manual mode	34
3.8.5	Simulation	35
3.8.6	Automatic mode	35
3.8.7	Maintenance and repair	36
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
4	Operation	41
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	43
4.2	KUKA smartHMI user interface	45
4.2.1	Status bar	46
4.2.2	Keypad	47
4.3	Switching on the robot controller and starting the VSS	48
4.4	Calling a menu	48
4.5	Defining the start type for VSS	49
4.6	Start types	49
4.7	Exiting VSS	49
4.8	Switching the robot controller off	51
4.9	Setting the user interface language	51
4.10	Changing user group	51
4.11	Disabling the robot controller	52
4.12	Changing operating mode	53
4.13	Coordinate systems	53
4.14	Jogging the robot	55
4.14.1	"Jogging Options" window	56
4.14.1.1	"General" tab	56
4.14.1.2	"Keys" tab	57
4.14.1.3	"Mouse" tab	57
4.14.1.4	"Kcp Pos." tab	58
4.14.1.5	"Act. Tool/Base" tab	58
4.14.2	Activating the jog mode	59
4.14.3	Setting the jog override (HOV)	59
4.14.4	Selecting the tool and base	59
4.14.5	Axis-specific jogging with the jog keys	60
4.14.6	Cartesian jogging with the jog keys	60
4.14.7	Configuring the Space Mouse	60
4.14.8	Defining the alignment of the Space Mouse	62
4.14.9	Cartesian jogging with the Space Mouse	63
4.14.10	Incremental jogging	63
4.15	Jogging external axes	64
4.16	Bypassing workspace monitoring	65
4.17	Monitor functions	65
4.17.1	Displaying the actual position	65
4.17.2	Displaying digital inputs/outputs	66
4.17.3	Displaying cyclical flags and flags	68

4.17.4	Displaying analog inputs/outputs	69
4.17.5	Displaying timers and counters	70
4.17.6	Displaying binary inputs/outputs	71
4.17.7	Displaying process parameters	73
4.17.8	Displaying gun inputs/outputs	73
4.17.9	Displaying inputs/outputs for Automatic External	74
4.17.10	Displaying and modifying the value of a variable	75
4.17.11	Displaying the state of a variable	76
4.17.12	Displaying the variable overview and modifying variables	77
4.17.13	Displaying calibration data	78
4.17.14	Displaying information about the robot and robot controller	78
4.17.15	Displaying robot data	79
5	Start-up and recommissioning	81
5.1	Checking the machine data	81
5.2	Defining hardware options	81
5.3	Changing the safety ID of the PROFINET device	82
5.4	Jogging the robot without a higher-level safety controller	83
5.5	Checking the activation of the positionally accurate robot model	84
5.6	Activating palletizing mode	84
5.7	Mastering	85
5.7.1	Mastering methods	86
5.7.2	Moving axes to the pre-mastering position	87
5.7.3	Mastering with the EMD	88
5.7.3.1	First mastering with the EMD	88
5.7.3.2	Teach offset	91
5.7.3.3	Master load with offset	92
5.7.4	Mastering with the dial gauge	93
5.7.5	Mastering external axes	94
5.7.6	Manually unmastering axes	94
5.8	Calibration	95
5.8.1	Defining the tool direction	95
5.8.2	Tool calibration	95
5.8.2.1	TCP calibration: XYZ 4-Point method	97
5.8.2.2	TCP calibration: XYZ Reference method	98
5.8.2.3	Defining the orientation: ABC World method	99
5.8.2.4	Defining the orientation: ABC 2-Point method	101
5.8.2.5	Numeric input	102
5.8.3	Base calibration	102
5.8.3.1	3-point method	103
5.8.3.2	Indirect method	104
5.8.3.3	Numeric input	104
5.8.4	Fixed tool calibration	105
5.8.4.1	Calibrating an external TCP	105
5.8.4.2	Entering the external TCP numerically	107
5.8.4.3	Workpiece calibration: direct method	107
5.8.4.4	Workpiece calibration: indirect method	108
5.8.5	Renaming the tool/base	109
5.8.6	Calibrating the linear unit	109

5.8.6.1	Moving to the reference point from 2 different positions	110
5.8.6.2	Numeric input	111
5.9	Load data	111
5.9.1	Checking loads with KUKA.Load	111
5.9.2	Calculating payloads with KUKA.LoadDataDetermination	112
5.9.3	Entering payload data	112
5.9.4	Entering supplementary load data	112
6	Configuration	115
6.1	Configuring gun inputs/outputs	115
6.2	Configuring binary inputs/outputs	116
6.3	Configuring the variable overview	117
6.4	Configuring workspaces	119
6.4.1	Configuring Cartesian workspaces	119
6.4.2	Configuring axis-specific workspaces	122
6.4.3	Mode for workspaces	124
6.5	Warm-up	124
6.5.1	Configuring warm-up	124
6.5.2	Warm-up sequence	125
6.5.3	System variables for warm-up	126
6.6	Defining calibration tolerances	127
6.7	Submit interpreter	128
6.7.1	Function of the Submit interpreter	128
6.7.2	Manually stopping or deselecting the Submit interpreter	128
6.7.3	Manually starting the Submit interpreter	129
6.8	Configuring Automatic External	129
6.8.1	Configuring CELL.SRC	129
6.8.2	Configuring Automatic External inputs/outputs	130
6.8.3	Automatic External inputs	131
6.8.4	Automatic External outputs	132
6.8.5	Signal diagrams	134
6.9	Event planner	136
6.9.1	Configuring a data comparison	137
6.9.2	Configuring T1 and T2 Consistency, AUT and EXT Consistency	137
6.9.3	Configuring Logic Consistency	138
7	Program management	141
7.1	Navigator file manager	141
7.1.1	Selecting filters	142
7.1.2	Creating a new program	142
7.1.3	Renaming a file	143
7.2	Selecting and deselecting a program	143
7.3	Toggling between the Navigator and the program	143
7.4	Structure of a program (Folge)	144
7.5	Displaying/hiding program sections	145
7.5.1	Displaying PLC instructions	145
7.5.2	Activating detail view (ASCII mode)	145
7.5.3	Activating/deactivating the line break function	146
7.6	Starting a program	146

7.6.1	Selecting the program run mode	146
7.6.2	Program run modes	146
7.6.3	Advance run	147
7.6.4	Setting the program override (POV)	147
7.6.5	Switching drives on/off	148
7.6.6	Robot interpreter status indicator	148
7.6.7	Starting a program forwards (manual)	148
7.6.8	Starting a program backwards	149
7.6.9	Carrying out a block selection	149
7.6.10	Resetting a program	150
7.6.11	Starting Automatic External mode	150
7.6.12	Dry run	151
7.7	Editing a program	152
7.7.1	Inserting a comment	152
7.7.2	Deleting program lines	153
7.7.3	Additional editing functions	153
7.8	Printing a program	154
7.9	Archiving	154
7.9.1	Archiving	154
7.9.2	Menu item "Archive"	154
7.9.3	Restoring data	155
7.9.3.1	Restoring data via the menu	155
7.9.3.2	Restoring data from a standard archive	155
8	Basic principles of motion programming	157
8.1	Overview of motion types	157
8.2	Motion type PTP	157
8.3	Motion type LIN	158
8.4	Motion type CIR	158
8.5	Approximate positioning	159
8.6	Orientation control LIN, CIR	160
8.6.1	Combinations of \$ORI_TYPE and \$CIRC_TYPE	161
8.7	Motion type "Spline"	163
8.7.1	Velocity profile for spline motions	165
8.7.2	Block selection with spline motions	166
8.7.3	Modifications to spline blocks	167
8.7.4	Approximate positioning of spline motions	169
8.7.5	Replacing an approximated motion with a spline block	170
8.7.5.1	SLIN-SPL-SLIN transition	173
8.7.6	Programming tips for spline motions	173
8.8	Orientation control SPLINE	174
8.8.1	SCIRC: reference system for the orientation control	176
8.8.2	SCIRC: orientation behavior	177
8.9	Status and Turn	179
8.9.1	Status	180
8.9.2	Turn	182
8.10	Singularities	182
9	Programming for user group "User" (inline forms)	185

9.1	Programming motions	185
9.1.1	Programming a PTP motion	185
9.1.2	Inline form for PTP motions	185
9.1.3	Programming a LIN motion	186
9.1.4	Inline form for LIN motions	187
9.1.5	Programming a CIR motion	188
9.1.6	Inline form for CIR motions	188
9.2	Programming application-specific motions	189
9.2.1	Programming a KLIN motion	190
9.2.2	Inline form for KLIN motions	190
9.2.3	Programming a KCIR motion	191
9.2.4	Inline form for KCIR motions	192
9.2.5	Programming a sensor-monitored LIN motion	193
9.2.6	Inline form "LIN SUCHEN"	193
9.3	Modifying programmed motions	194
9.3.1	Modifying motion parameters	194
9.3.2	Modifying the coordinates of a taught point	194
9.3.3	Block function	194
9.3.3.1	Modifying blocks of motion parameters	195
9.3.3.2	Transforming blocks of coordinates	195
9.3.3.3	"Mirroring" window	198
9.3.3.4	"Point transformation - axis-specific" window	199
9.3.3.5	"Cartesian point transformation" window	200
9.4	Programming PLC instructions	201
9.4.1	Boolean operands	201
9.4.2	Arithmetic operands	201
9.4.3	Operators	202
9.4.4	Priority of the operators	202
9.4.5	Inserting a new operator or operand	202
9.4.6	Setting an output, cyclical flag (Merker) or flag	203
9.4.7	Inline form "A/M/F"	203
9.4.8	Setting an integer counter or binary output	204
9.4.9	Inline form "i/bin"	204
9.4.10	Starting a timer	205
9.4.11	Inline form "t= (Start)"	205
9.4.12	Stopping a timer	206
9.4.13	Inline form "t=STOP"	207
9.4.14	Programming an arithmetic comparison	207
9.4.15	Inline form "Compare"	207
9.4.16	Setting a pulse output	208
9.4.17	Inline form "Pulse"	209
9.4.18	Programming a FB ONL motion condition	209
9.4.19	Inline form "FB ONL"	210
9.4.20	Programming a FB PSPS motion condition	210
9.4.21	Inline form "FB PSPS"	211
9.4.22	Programming a signal-dependent wait function	211
9.4.23	Inline form "WARTE ONL/bis"	212
9.4.24	Programming a wait time	213
9.4.25	Inline form "WARTE ZEIT"	213

9.4.26	Switching interlocking on/off	214
9.4.27	Inline form "VERR."	215
9.4.28	Switching an Interbus segment or Interbus device on/off	216
9.4.29	Inline form "IBG"	216
9.4.30	Setting an analog output	217
9.4.31	Inline form "ana=KONST"	217
9.4.32	Inline form "ana=Vprop"	218
9.4.33	Inline form "ana=KST+P"	221
9.4.34	Activating/deactivating weaving	223
9.4.35	Weave patterns	223
9.4.36	Inline form "Weave"	224
9.4.37	Programming a path-related switching action	225
9.4.38	Inline form "BS A/F"	228
9.4.39	Inline form "BS bin/ana"	229
9.4.40	Modifying a switching point	231
9.4.41	Calling a macro	231
9.4.42	Inline form "SPSMAKRO"	232
9.4.43	Calling a subprogram	232
9.4.44	Inline form "UP"	233
9.4.45	Programming program loops	233
9.4.46	Inline form "REPEAT MAKRO/UP"	234
9.4.47	Calling a gun function	235
9.4.48	Inline form "ZANGE"	235
9.4.49	Calling VW_USER	236
9.4.50	Inline form "VW User"	236
9.5	Programming jump labels	237
9.5.1	Inline form "Label/SPSLabel"	237
9.5.2	Inline form "GOTO Label/GOTO SPSLabel"	238
9.6	Programming instructions in macros	239
9.6.1	Macros in VSS	239
9.6.2	MakroSAW	240
9.6.3	MakroSPS	240
9.6.4	Setting a position-dependent flag	240
9.6.5	Inline form "Position Flag"	240
9.6.6	MakroStep	241
9.6.7	Inline form "SCHRITT"	242
9.6.8	Inline form "Schritt" (transition)	242
9.6.9	MakroTrigger	243
9.6.10	Inline form "Trigger"	244
10	Programming for user group "Expert" (KRL syntax)	245
10.1	Overview of KRL syntax	245
10.2	Symbols and fonts	246
10.3	Important KRL terms	246
10.3.1	SRC files and DAT files	246
10.3.2	Subprograms and functions	247
10.3.3	Naming conventions and keywords	247
10.3.4	Data types	248
10.3.5	Areas of validity	249

10.3.6	Constants	251
10.4	Variables and declarations	251
10.4.1	DECL	251
10.4.2	ENUM	252
10.4.3	STRUC	254
10.5	Motion programming: PTP, LIN, CIRC	255
10.5.1	PTP	255
10.5.2	PTP_REL	256
10.5.3	LIN	257
10.5.4	LIN_REL	257
10.5.5	CIRC	259
10.5.6	CIRC_REL	260
10.6	Motion programming: spline	261
10.6.1	SPLINE ... ENDSPLINE	261
10.6.2	SLIN	262
10.6.3	SCIRC	263
10.6.4	SPL	265
10.6.5	TIME_BLOCK	265
10.7	Program execution control	269
10.7.1	CONTINUE	269
10.7.2	EXIT	270
10.7.3	FOR ... TO ... ENDFOR	270
10.7.4	GOTO	271
10.7.5	HALT	272
10.7.6	IF ... THEN ... ENDIF	272
10.7.7	LOOP ... ENDLOOP	273
10.7.8	REPEAT ... UNTIL	273
10.7.9	SWITCH ... CASE ... ENDSWITCH	274
10.7.10	WAIT FOR	275
10.7.11	WAIT SEC	275
10.7.12	WHILE ... ENDWHILE	276
10.8	Inputs/outputs	276
10.8.1	ANIN	276
10.8.2	ANOUT	277
10.8.3	PULSE	279
10.8.4	SIGNAL	282
10.9	Subprograms and functions	283
10.9.1	RETURN	283
10.10	Interrupt programming	284
10.10.1	BRAKE	284
10.10.2	INTERRUPT ... DECL ... WHEN ... DO	284
10.10.3	INTERRUPT	286
10.10.4	RESUME	287
10.11	Path-related switching actions (=Trigger)	289
10.11.1	TRIGGER WHEN DISTANCE	289
10.11.2	TRIGGER WHEN PATH	292
10.11.3	TRIGGER WHEN PATH (for SPLINE)	295
10.11.3.1	Spline: trigger point in the case of approximate positioning	297
10.12	Communication	299

10.13 System functions	299
10.13.1 VARSTATE()	299
10.14 Editing string variables	301
10.14.1 String variable length in the declaration	301
10.14.2 String variable length after initialization	302
10.14.3 Deleting the contents of a string variable	302
10.14.4 Extending a string variable	303
10.14.5 Searching a string variable	303
10.14.6 Comparing the contents of string variables	304
10.14.7 Copying a string variable	305
11 KRL function call with parameter transfer (=USER)	307
11.1 VW_USR_R	307
11.2 VW_USR_S	307
11.3 Configuring the inline form "VW User"	308
11.4 Example programs	312
11.4.1 Transferring parameters for LIN_REL motion to a function	312
11.4.2 Transferring parameters to a function for executing a pattern	314
12 Diagnosis	317
12.1 Logbook	317
12.1.1 Displaying the logbook	317
12.1.2 "Log" tab	318
12.1.3 "Filter" tab	319
12.1.4 Configuring the logbook	319
12.2 Displaying a wait condition	320
12.3 Displaying the caller stack	321
12.4 Displaying the reference list	322
12.5 Interlock diagnosis	323
12.5.1 Configuring interlock diagnosis	323
12.5.2 Carrying out interlock diagnosis	324
12.6 Displaying diagnostic data about the kernel system	325
13 Installation	327
13.1 Overview of the software components	327
13.2 Installing Windows and VSS (from image)	327
13.3 Installing additional software (via KUKA smartHMI)	328
13.4 VSS update	329
13.4.1 Update from a USB storage medium	329
13.4.2 Update from the network	330
14 Messages	331
14.1 Error messages of the positionally accurate robot model	331
15 KUKA Service	333
15.1 Requesting support	333
15.2 KUKA Customer Support	333
Index	341

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 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 VW 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

Warnings marked with this pictogram are relevant to safety and **must** be observed.



Danger!

This warning means that death, severe physical injury or substantial material damage **will** occur, if no precautions are taken.



Warning!

This warning means that death, severe physical injury or substantial material damage **may** occur, if no precautions are taken.



Caution!

This warning means that minor physical injuries or minor material damage **may** occur, if no precautions are taken.

Notes

Notes marked with this pictogram contain tips to make your work easier or references to further information.



Tips to make your work easier or references to further information.

1.4 Trademarks

Windows 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

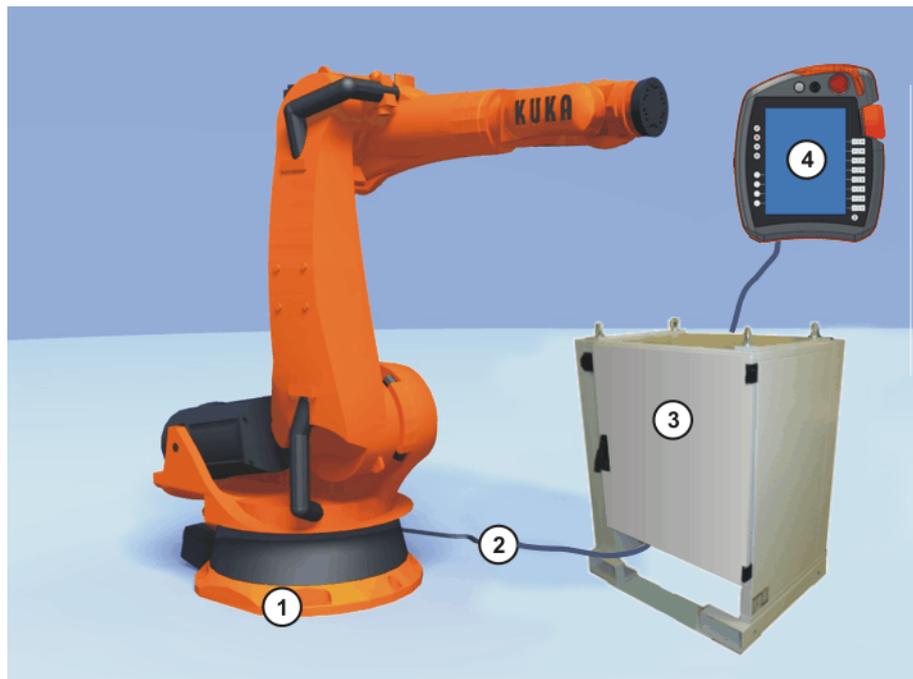


Fig. 2-1: Example of an industrial robot

- | | |
|---------------------|--------------------|
| 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:

- VW System Software 8.1
- Windows XP embedded 3.0

2.3 Overview of VW System Software (VSS)

Description The VW System Software (VSS) is responsible for all the basic operator control functions of the industrial robot.

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

smarthMI

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

Features:

- User management
- 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.



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

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 component.

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.

Misuse

Any use or application deviating from the intended use is deemed to be impermissible misuse. This includes e.g.:

- Transportation of persons and animals
- Use as a climbing aid
- Operation outside the permissible operating parameters
- Use in potentially explosive environments
- Operation without additional safeguards
- Outdoor operation

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.

Declaration of conformity

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.

Declaration of incorporation

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.
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. Note: 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. Note: 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. Note: This stop is called safety STOP 2 in this document.
Stop category 0	The drives are deactivated immediately and the brakes are applied. The manipulator and any external axes (optional) perform path-oriented braking. Note: 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. Note: 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. Note: 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 instruction 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.

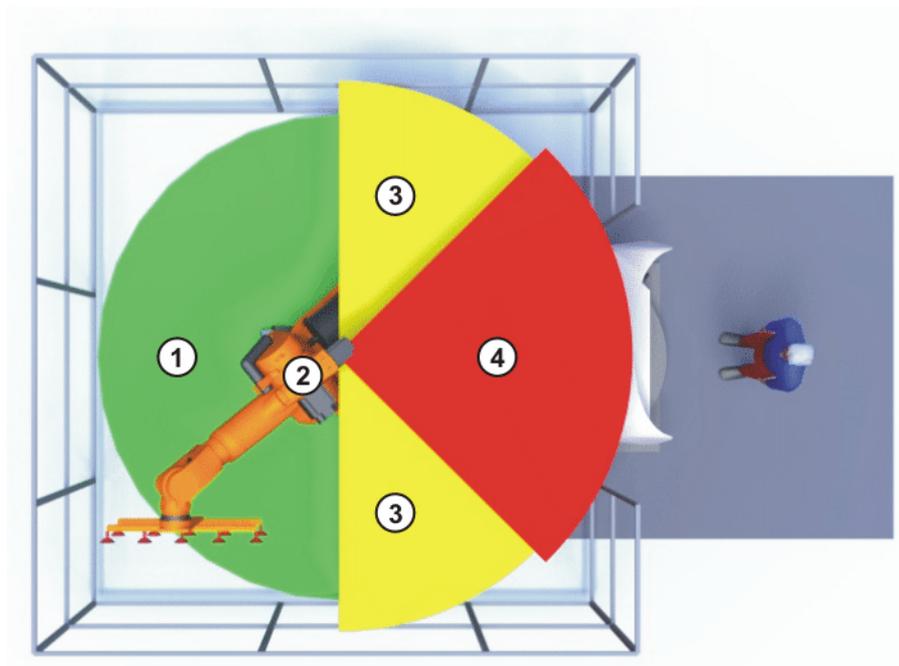


Fig. 3-1: Example of axis range A1

- 1 Workspace
- 2 Manipulator
- 3 Stopping distance
- 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 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 system
- Enabling device
- 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 Interface Board
- Resolver Digital Converter (RDC)
- KUKA Power Pack
- KUKA Servo Pack

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



Danger!

In the absence of functional 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 External (AUT EXT)



If the operating mode is changed during operation, the drives are immediately switched off. The industrial robot stops with a safety stop 2.

Operating mode	Use	Velocities
T1	For test operation, programming and teaching	<ul style="list-style-type: none"> ■ Program verification: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	<ul style="list-style-type: none"> ■ Program verification: Programmed velocity ■ Jog mode: Not possible
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC	<ul style="list-style-type: none"> ■ Program mode: Programmed velocity ■ Jog mode: Not possible

In order to be able to work on the robot in operating modes T1 and T2 with the safety gate open, the following jumpering options are available:

E2 keyswitch	T1 mode is active. The safety circuit "Safety gate open" is jumpered. The robot can be moved at Manual Reduced Velocity with the safety gate open.
E2+E7 keyswitch	T2 mode is active. The machine safety is jumpered. The robot can be moved at Manual High Velocity with the safety gate open.

Operating mode	Drives can be switched on	Safety gate	E2 key-switch	E7 key-switch
T1	Yes	Open	Yes	No
T1	Yes	Closed	Yes	No
T2	Yes	Open	Yes	Yes
T2	Yes	Closed	Yes	Yes
T2	Yes	Closed	No	No
T2	No	Open	No	No
----	No	Open	No	Yes
----	No	Closed	No	Yes

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.

In the test modes Manual Reduced Velocity (T1) and Manual High Velocity (T2), operator safety can be bypassed with the E2 and E2+E7 keyswitches.



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 26)

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. The KUKA safety controller generates a signal that prevents the higher-level controller from triggering an EMERGENCY STOP for the overall system.

**Warning!**

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 on every operator panel and anywhere else it may be necessary to trigger an EMERGENCY STOP. 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
- Start key released

**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 the customer interface.

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

3.5.10 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 there is no external signal present. If the external signal is active, the manipulator can be moved again. No acknowledgement is required.

3.5.11 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.



Warning!

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 333). 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 A 1 to A 3. 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).

**Warning!**

The motors reach temperatures during operation which can cause burns to the skin. Contact should 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:

- Rating plates
- Warning labels
- Safety symbols
- Designation labels
- Cable markings
- Identification 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 measures	T1	T2	AUT EXT
Operator safety	- *	- *	active
EMERGENCY STOP device	active	active	active
Enabling device	active	active	-
Reduced velocity during program verification	active	-	-
Jog mode	active	active	-
Software limit switches	active	active	active

* In T1 and T2 modes, operator safety can be bypassed with the E2 and E2+E7 keyswitches.

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!

**Warning!**

The motors reach temperatures during operation which can cause burns to the skin. Contact should 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.

**Warning!**

The operator must ensure that decoupled KCPs 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 this precaution may result in death, severe physical injuries or considerable damage to property.

External keyboard, external mouse

An external keyboard and/or external mouse may only be used if the following conditions are met:

- Start-up or maintenance work is being carried out.
- The drives are switched off.
- There are no persons in the danger zone.

The KCP must not be used as long as an external keyboard and/or external mouse are connected.

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.

Faults

The following tasks must be carried out in the case of faults in the industrial robot:

- Switch off the robot controller and secure it (e.g. with a padlock) to prevent unauthorized persons from switching it on again.
- Indicate the fault by means of a label with a corresponding warning (tag-out).
- Keep a record of the faults.
- Eliminate the fault and carry out a function test.

Modifications

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.

New or modified programs must always be tested first in Manual Reduced Velocity mode (T1).

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.

3.8.2 Transportation

Manipulator	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.
Robot controller	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. Transportation must be carried out in accordance with the operating instructions or assembly instructions of the robot controller.
External axis (optional)	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.



Warning!

If additional components (e.g. cables), that 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.



Caution!

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.

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 robot 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.



Warning!

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 updated.



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 robot can be moved in T1 in the absence of the safety periphery.

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 robot's danger zone.
- An unauthorized person moves the robot.
- In a hazardous situation, a disabled external EMERGENCY STOP device is actuated and the robot 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 robot'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 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).



Danger!

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 robot.

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
- Teaching
- 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.
- Components, 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 unauthorized persons from switching it 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.

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 600 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 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 configuration 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
2006/42/EC	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
2004/108/EC	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
97/23/EC	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	1997
EN ISO 13850	Safety of machinery: Emergency stop - Principles for design	2008
EN ISO 13849-1	Safety of machinery: Safety-related parts of control systems - Part 1: General principles for design	2008
EN ISO 13849-2	Safety of machinery: Safety-related parts of control systems - Part 2: Validation	2008
EN ISO 12100-1	Safety of machinery: Basic concepts, general principles for design - Part 1: Basic terminology, methodology	2003
EN ISO 12100-2	Safety of machinery: Basic concepts, general principles for design - Part 2: Technical principles	2003
EN ISO 10218-1	Industrial robots: Safety	2008
EN 614-1	Safety of machinery: Ergonomic design principles - Part 1: Terminology and general principles	2006
EN 61000-6-2	Electromagnetic compatibility (EMC): Part 6-2: Generic standards; Immunity for industrial environments	2005
EN 61000-6-4	Electromagnetic compatibility (EMC): Part 6-4: Generic standards; Emission standard for industrial environments	2007
EN 60204-1	Safety of machinery: Electrical equipment of machines - Part 1: General requirements	2006



EN ISO 10218-1, Annex B, specifies the need for information about the stopping time and distance. These have not yet been determined in full for all robot types in conjunction with the KR C4 robot controller. In this respect, the industrial robot does not conform to the requirements of EN ISO 10218-1.

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	Unplug the smartPAD. (>>> 4.1.3 "Disconnecting and connecting the smartPAD" Page 43)
2	Keyswitch 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. (>>> 4.12 "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. (>>> 4.14 "Jogging the robot" Page 55)
5	Jog keys. For moving the robot manually. (>>> 4.14 "Jogging the robot" Page 55)
6	Key for setting the program override
7	Key for setting the jog override
8	Menu key. Shows the menu items on the smartHMI. (>>> 4.4 "Calling a menu" Page 48)
9	Technology keys. The technology 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 | Enabling switch |
| 2 | Start key (green) | 5 | Identification plate |
| 3 | Enabling switch | | |

Description

Element	Description
Identification plate	Identification plate
Start key	The Start key is used to start a program.
Enabling switch	<p>The enabling switch has 3 positions:</p> <ul style="list-style-type: none"> ■ Not pressed ■ Center position ■ Panic position <p>The enabling switch must be held in the center position in the operating mode Test (T1 or T2) in order to be able to jog the robot.</p> <p>In the operating mode Automatic External, the enabling switch has no function.</p>

4.1.3 Disconnecting and connecting the smartPAD

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

**Warning!**

- 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

1. Press the disconnect button on the smartPAD.

A message and a counter are displayed on the smartHMI. While the counter is running, 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.

A smartPAD can be reconnected to the robot controller at any time. Precondition: Same smartPAD variant as the disconnected device. The smartHMI is automatically displayed again.

4.2 KUKA smartHMI user interface

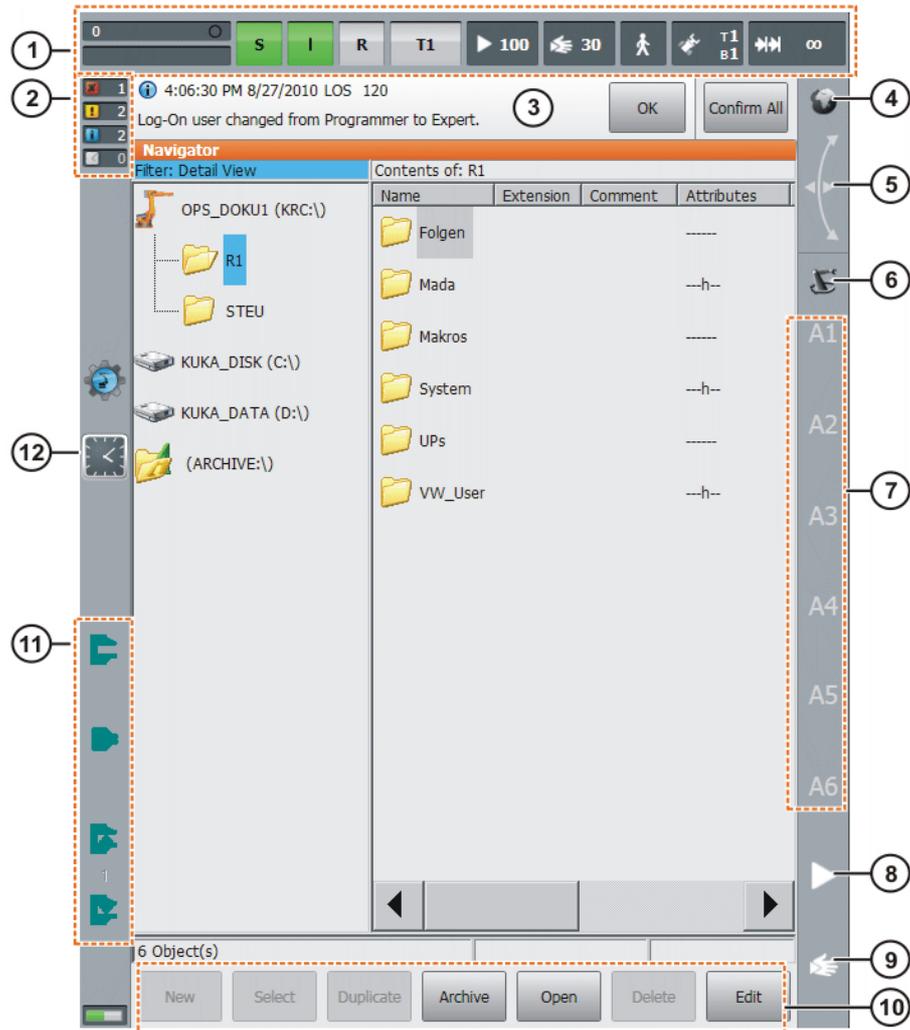


Fig. 4-3: KUKA smartHMI user interface

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 OK . All acknowledgeable messages can be acknowledged at once with All OK .
4	Space Mouse 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	Space Mouse alignment indicator Touching this indicator opens a window in which the current alignment of the Space Mouse is indicated and can be changed. (>>> 4.14.8 "Defining the alignment of the Space Mouse" Page 62)
6	Jog keys 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 (>>> 7.6.4 "Setting the program override (POV)" Page 147)
9	Jog override (>>> 4.14.3 "Setting the jog override (HOV)" Page 59)
10	Softkey bar. The softkeys change dynamically and always refer to the window that is currently active in the smartHMI. At the right-hand end is the Edit softkey. This can be used to call numerous commands relating to the Navigator.
11	Technology key labels The icons displayed here depend on which technology packages are installed.
12	Clock The clock displays the system time. Touching the clock displays the system time in digital format, together with the current date.

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

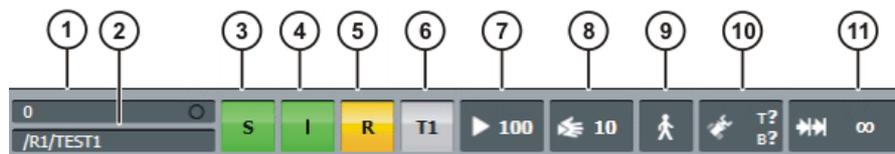


Fig. 4-4: KUKA smartHMI status bar

Item	Description
1	Robot name. The robot name can be changed. (>>> 4.17.15 "Displaying robot data" Page 79)
2	Currently selected program
3	Submit interpreter status indicator (>>> 6.7 "Submit interpreter" Page 128)

Item	Description
4	Drives status indicator. The drives can be switched on or off here. (>>> 7.6.5 "Switching drives on/off" Page 148)
5	Robot interpreter status indicator. Programs can be reset or canceled here. (>>> 7.6.6 "Robot interpreter status indicator" Page 148) (>>> 7.2 "Selecting and deselecting a program" Page 143) (>>> 7.6.10 "Resetting a program" Page 150)
6	Current operating mode (>>> 4.12 "Changing operating mode" Page 53)
7	POV status indicator. Indicates the current program override. (>>> 7.6.4 "Setting the program override (POV)" Page 147)
8	HOV status indicator. Indicates the current jog override. (>>> 4.14.3 "Setting the jog override (HOV)" Page 59)
9	Program run mode status indicator. Indicates the current program run mode. (>>> 7.6.2 "Program run modes" Page 146)
10	Tool/base status indicator. Indicates the current tool and base. (>>> 4.14.4 "Selecting the tool and base" Page 59)
11	Incremental jogging status indicator. (>>> 4.14.10 "Incremental jogging" Page 63)

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.

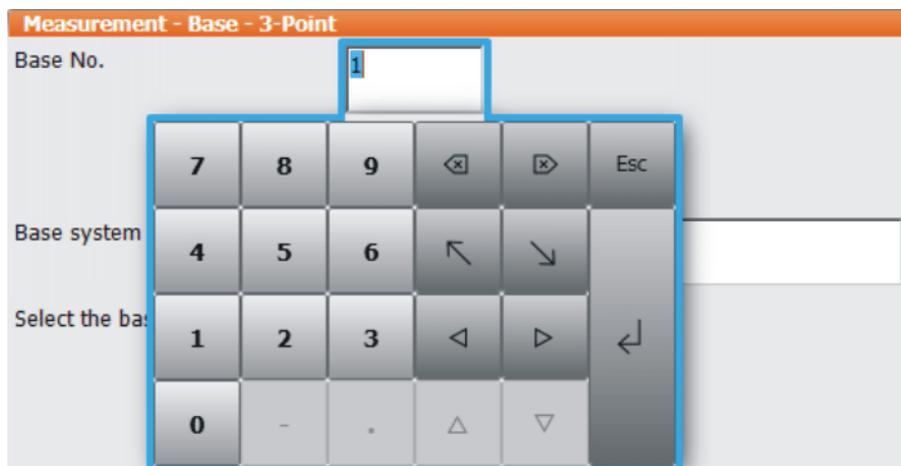


Fig. 4-5: Example keypad

4.3 Switching on the robot controller and starting the VSS

Procedure

- Turn the main switch on the robot controller to ON.
The operating system and the VSS start automatically.

If the VSS does not start automatically, e.g. because the Startup function has been disabled, execute the file StartKRC.exe in the directory C:\KRC.

If the robot controller is logged onto the network, the start may take longer.

4.4 Calling a menu

Procedure

- Press Menu key on the KCP. The **Main Menu** window is opened.
The display is always the same as that which was in the window before it was last closed.

Description

Properties of the **Main Menu** window:

- The main menu is displayed in the left-hand column.
- Touching a menu item that contains an arrow opens the corresponding submenu (e.g. **Configure**).
Depending on how many nested submenus are open, the **Main Menu** column may no longer be visible, with only the submenus remaining visible.
- The arrow key in the top right-hand corner closes the most recently opened submenu.
- The Home key in the top right-hand corner closes all open submenus.
- 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.
- The white cross on the left-hand side closes the window.

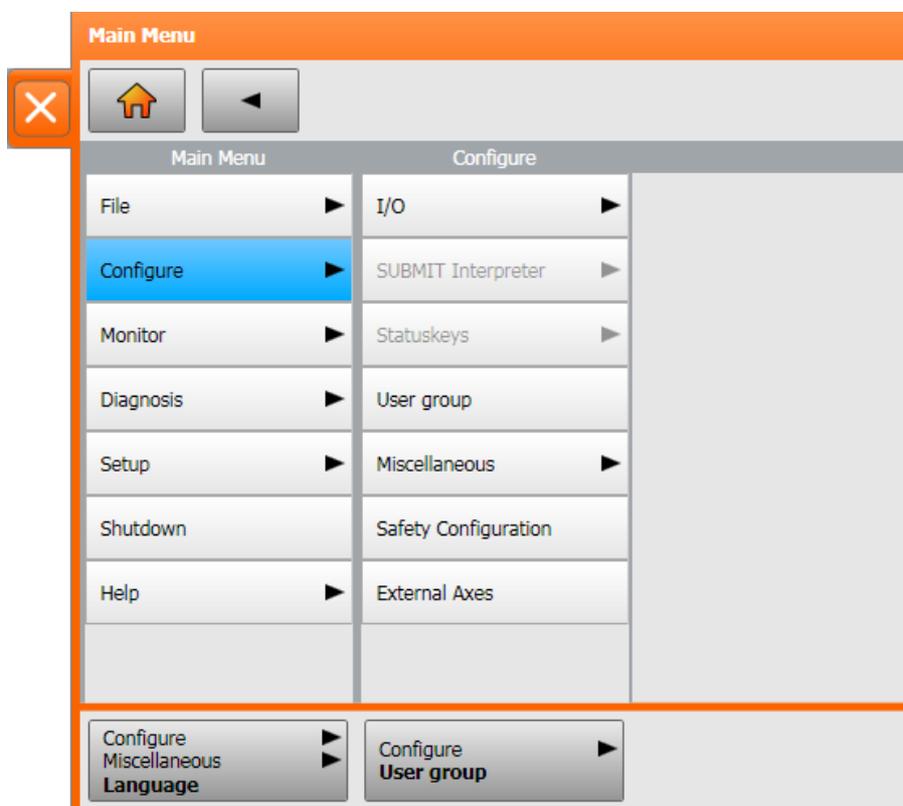


Fig. 4-6: Example: Configure submenu is open.

4.5 Defining the start type for VSS

This function defines how the VSS 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 VSS boots with a cold start.



If the robot controller detects a system error or modified data, the VSS always starts with a cold start – irrespective of the selected start type.

Precondition ■ Expert user group

Procedure

1. Select the menu item **Shutdown**. A window opens.
2. Select the start type: **Cold start** or **Hibernate**.
3. Close the window. The selected start type is applied.

4.6 Start types

Start type	Description
Cold start	After a cold start the robot controller displays the Navigator. No program is selected. The controller is completely reinitialized, e.g. all user outputs are set to FALSE.
Hibernate	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. 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.

4.7 Exiting VSS

Precondition ■ Operating mode T1 or T2.

Procedure

1. Select the menu item **Shutdown**.
2. Select the desired options.
3. Press **Shut down KRC**. Confirm the request for confirmation with **Yes**. The VSS is terminated.



Caution!

If, on shutting down, the option **with restart** was selected, the main switch on the robot controller must not be pressed until the reboot has been completed. System files may otherwise be destroyed.

If this option was not selected on shutting down, the main switch can be pressed once the controller has shut down.



If the robot controller detects a system error or modified data, the VSS always starts with a cold start – irrespective of the selected start type.

Description

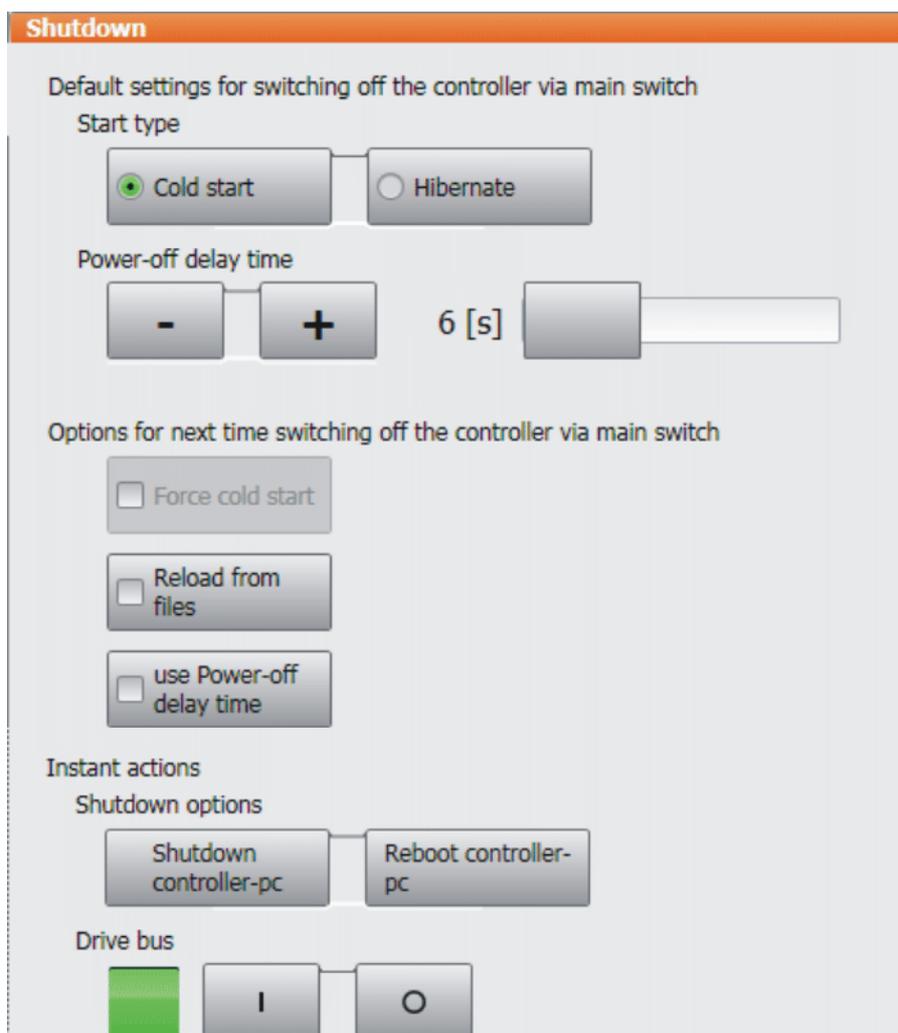


Fig. 4-7: Shutdown with on/off options window

The following options are available:

Option	Description
Cold start	<p>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.)</p> <p>The setting can only be modified in the user group "Expert".</p> <p>(>>> 4.6 "Start types" Page 49)</p>
Hibernate	<p>After a power failure, the robot controller starts with a start after Hibernate. (A power failure and start are generally triggered by switching the main switch on the robot controller off and on.)</p> <p>The setting can only be modified in the user group "Expert".</p> <p>(>>> 4.6 "Start types" Page 49)</p>

Option	Description
Power-off delay time	Wait time before the robot controller is shut down. The purpose of the wait time is to ensure that the system does not immediately shut down, for example, in the event of a very sudden, brief power failure, but bridges the power failure for the duration of the wait time. This value can only be changed in the user group "Expert".
Use Power-off delay time	If the wait time is to be ignored, it can be deactivated here for the next shutdown procedure.
Force cold start	The next start is a cold start. This also applies if the option Hibernate is selected under Default start type . This setting only applies to the next start. It can only be changed in the user group "Expert".
Shut down Windows	Windows is shut down. This setting can only be modified in the user group "Expert". Shut down Windows is automatically active if Hibernate has been selected. In this case, Windows is also restarted with Hibernate.
with restart	The robot controller is immediately restarted after shutdown.
Shut down KRC	Only available in operating modes T1 and T2. If the option Shut down Windows is active, then Windows is also shut down.
Drive bus off / Drive bus on	Only available in operating modes T1 and T2. The drive bus can be switched off or on. Drive bus status indicator: <ul style="list-style-type: none"> ■ Green: Drive bus is on. ■ Red: Drive bus is off. ■ Gray: Status of the drive bus is unknown.

4.8 Switching the robot controller off

Procedure

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



Caution!

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

4.9 Setting the user interface language

Procedure

1. Select the menu sequence **Configure > Miscellaneous > Language**.
2. Select the desired language. Confirm with **OK**.

4.10 Changing user group

Description

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

- **User**
User group for the operator. This is the default user group.
- **Expert**
User group for the programmer.
- **15 (optional)**
If user group 15 is installed, the user group “User” can only view programs and no longer edit them.



User group 15 can be selected when installing VSS and thus included in the installation.

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

If the mode is switched to 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.

Procedure

1. Connect KUKA USB stick with the data from the key CD.



Caution!

The only USB stick that may be used is the KUKA USB stick. Data may be lost or modified if any other USB stick is used.

2. Select the menu sequence **Configure > User group**. The current user group is displayed.
3. Press **Default** to switch to the default user group.
Press **Log On...** to switch to the user group “Expert”. The user group “Expert” is displayed. Confirm with **Log On**.
If prompted: Enter password and confirm with **Log On**.

4.11 Disabling the robot controller

Description

The robot controller can be disabled. It is then disabled for all actions except logging back on.

The robot controller cannot be disabled in the default user group.

Precondition

- The default user group is not selected.

Procedure

1. Select the menu sequence **Configure > User group**.
2. Press **Lock**. The robot controller is then disabled for all actions except logging on. The current user group is displayed.
3. Log back on:
 - Log on as the default user: Press **Default**.
 - Log on as a different user: Press **Log On....** Select the desired user group and confirm with the **Log On...** softkey.
If prompted: Enter password and confirm with **Log On**.



If you log onto the same user group as before, all the windows and programs of the previous user remain open. No data are lost.

If you log onto a different user group, the windows and programs of the previous user may be closed. Data can be lost!

4.12 Changing operating mode



If the operating mode is changed during operation, the drives are immediately switched off. The industrial robot stops with a safety stop 2.

Precondition ■ Key for the switch for calling the connection manager

Procedure

1. On the KCP, 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 smart-PAD.

Operating mode	Use	Velocities
T1	For test operation, programming and teaching	<ul style="list-style-type: none"> ■ Program verification: Programmed velocity, maximum 250 mm/s ■ Jog mode: Jog velocity, maximum 250 mm/s
T2	For test operation	<ul style="list-style-type: none"> ■ Program verification: Programmed velocity ■ Jog mode: Not possible
AUT EXT	For industrial robots with higher-level controllers, e.g. PLC	<ul style="list-style-type: none"> ■ Program mode: Programmed velocity ■ Jog mode: Not possible

4.13 Coordinate systems

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

- WORLD
- ROBROOT
- BASE
- TOOL

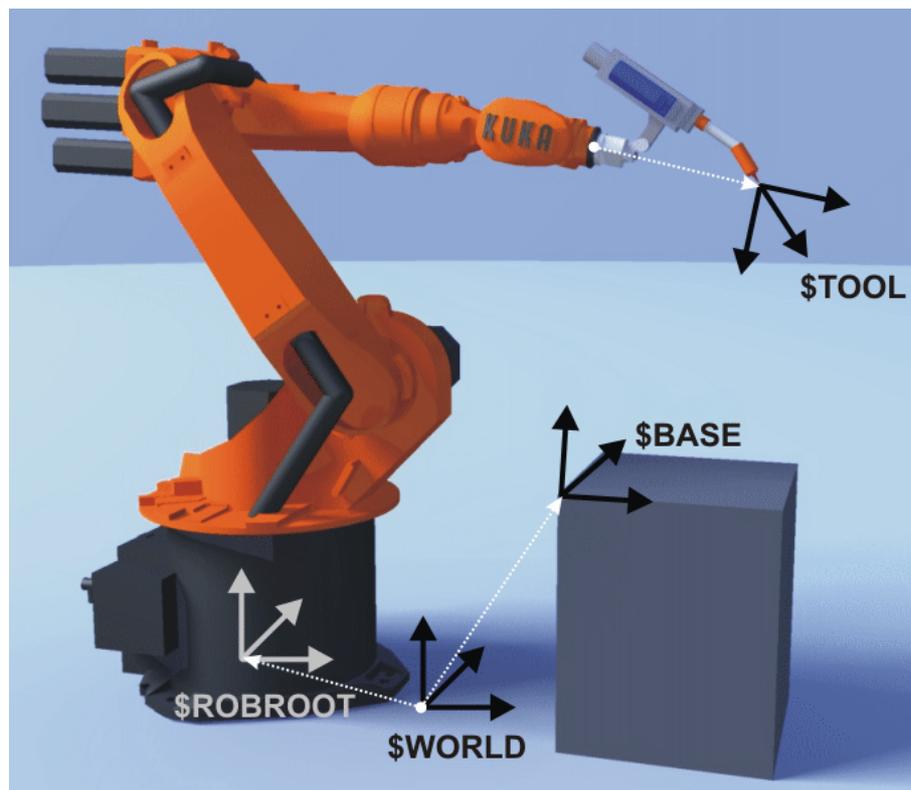


Fig. 4-8: Overview of coordinate systems

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.

(>>> 5.8.3 "Base calibration" Page 102)

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.

(>>> 5.8.2 "Tool calibration" Page 95)

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

4.14 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.

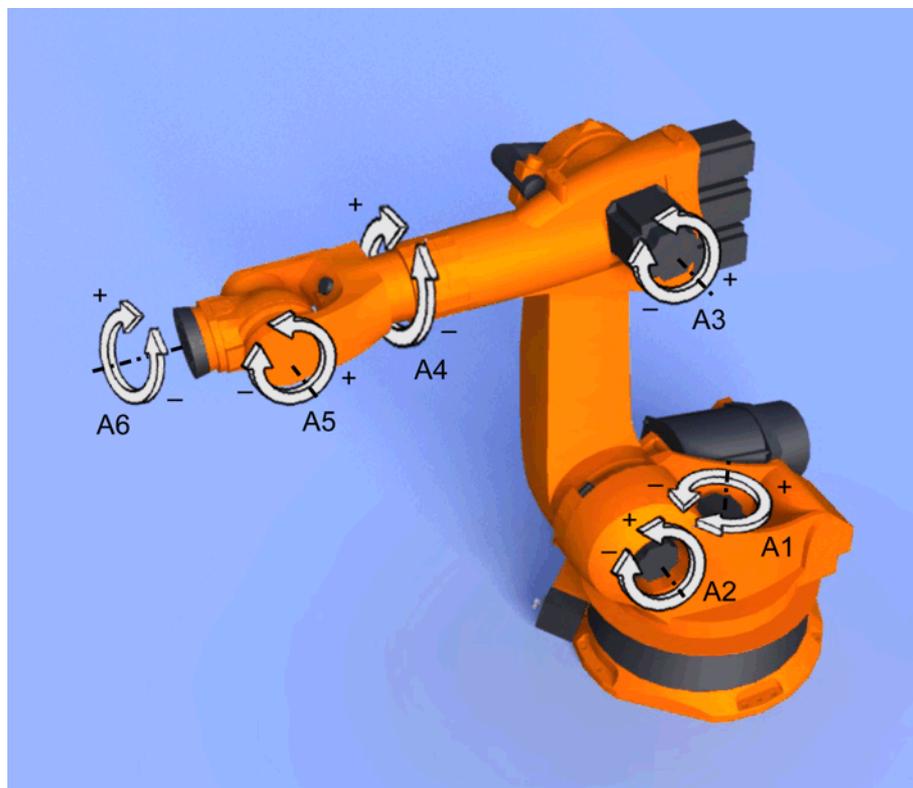


Fig. 4-9: Axis-specific jogging

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	(>>> 4.14.6 "Cartesian jogging with the jog keys" Page 60)	(>>> 4.14.5 "Axis-specific jogging with the jog keys" Page 60)
Space Mouse	(>>> 4.14.9 "Cartesian jogging with the Space Mouse" Page 63)	Axis-specific jogging with the Space Mouse is possible, but is not described here.

4.14.1 “Jogging 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.

4.14.1.1 “General” tab

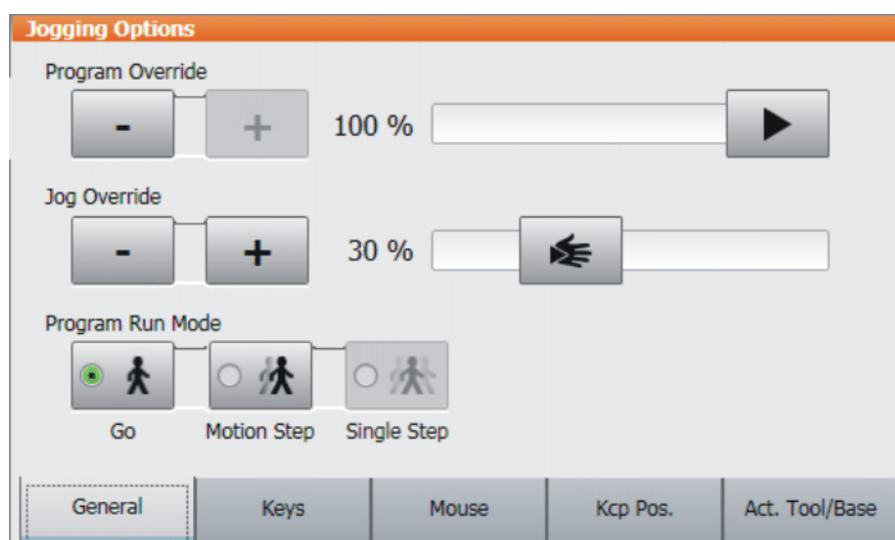


Fig. 4-10: General tab

Description

Item	Description
1	Set program override (>>> 7.6.4 "Setting the program override (POV)" Page 147)
2	Set jog override (>>> 4.14.3 "Setting the jog override (HOV)" Page 59)
3	Select the program run mode (>>> 7.6.2 "Program run modes" Page 146)

4.14.1.2 “Keys” tab

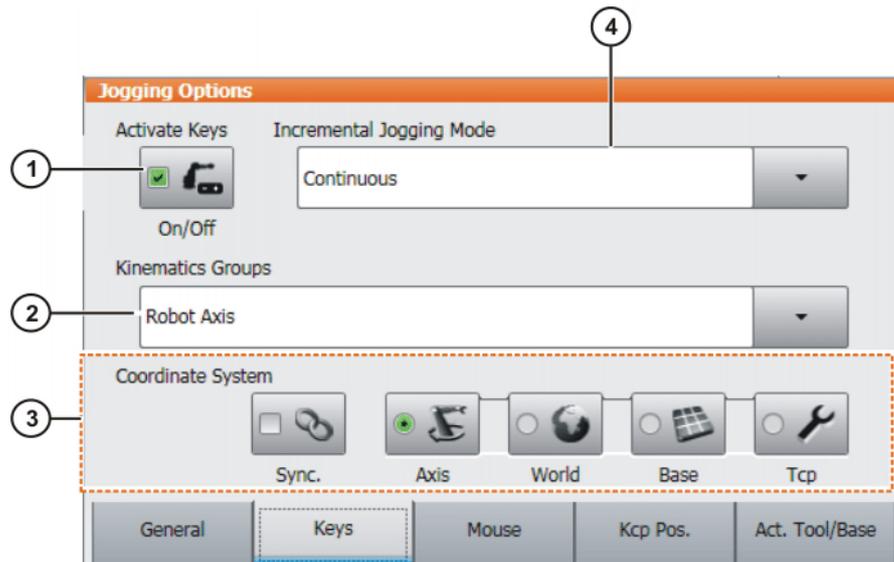


Fig. 4-11: Keys tab

Description

Item	Description
1	Activate jog mode “Jog keys” (>>> 4.14.2 "Activating the jog mode" Page 59)
2	Select a kinematics group. The kinematics group defines the axes to which the jog keys refer. Default: Robot axes (= A1 to A6) Depending on the system configuration, other kinematics groups may be available. (>>> 4.15 "Jogging external axes" Page 64)
3	Select the coordinate system for jogging with the jog keys
4	Incremental jogging (>>> 4.14.10 "Incremental jogging" Page 63)

4.14.1.3 “Mouse” tab

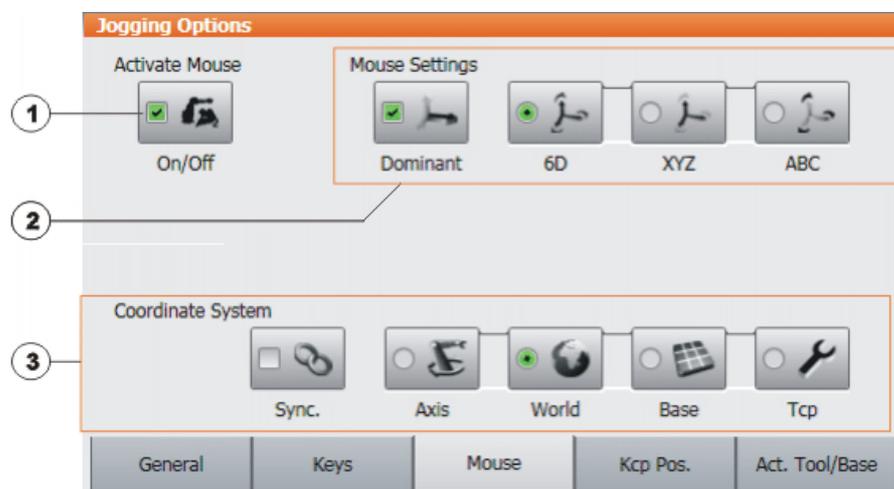


Fig. 4-12: Mouse tab

Description

Item	Description
1	Activate jog mode "Space Mouse" (>>> 4.14.2 "Activating the jog mode" Page 59)
2	Configure the Space Mouse (>>> 4.14.7 "Configuring the Space Mouse" Page 60)
3	Select the coordinate system for jogging with the Space Mouse

4.14.1.4 "Kcp Pos." tab

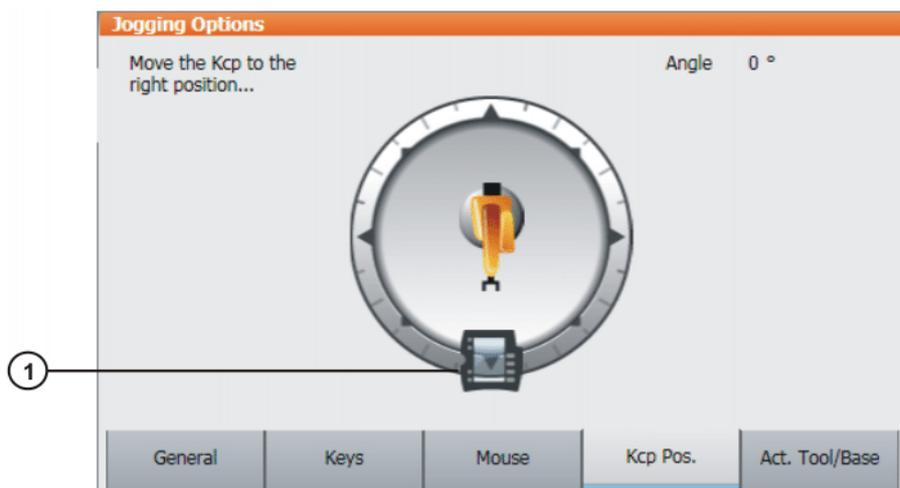


Fig. 4-13: Kcp Pos. tab

Description

Item	Description
1	(>>> 4.14.8 "Defining the alignment of the Space Mouse" Page 62)

4.14.1.5 "Act. Tool/Base" tab

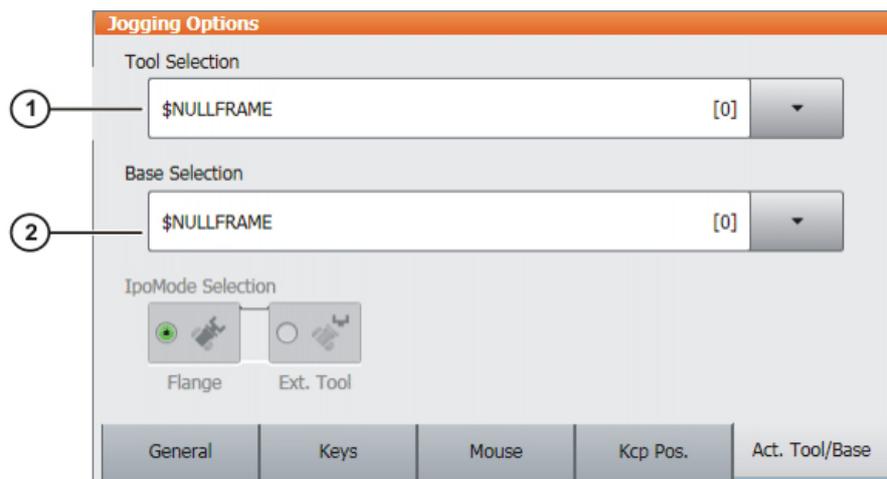


Fig. 4-14: Act. Tool/Base tab

Description

Item	Description
1	The current tool is displayed here. A different tool can be selected. (>>> 4.14.4 "Selecting the tool and base" Page 59) The display Unknown [?] means that no tool has yet been calibrated.
2	The current base is displayed here. A different base can be selected. (>>> 4.14.4 "Selecting the tool and base" Page 59) The display Unknown [?] means that no base has yet been calibrated.

4.14.2 Activating the jog mode**Procedure**

1. Open the **Jogging Options** window.
(>>> 4.14.1 "Jogging Options" window" Page 56)
2. To activate the jog mode "Jog keys":
On the **Keys** tab, activate the **Activate Keys** check box.
To activate the jog mode "Space Mouse":
On the **Mouse** tab, activate the **Activate Mouse** check box.

Description

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.

4.14.3 Setting the jog override (HOV)**Description**

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.

Procedure

1. Touch the **HOV** status indicator. The **Jog Override** window is opened.
2. Set the desired value. It can be set using either the plus/minus keys or by means of the slide controller.
 - Plus/minus keys: The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%
 - Slide controller: The override can be adjusted in 1% steps.
3. Touch the **HOV** status indicator again. (Or touch the area outside the window.)
The window closes and the selected override value is applied.



The **Jogging Options** window can be opened via **Options** in the **Jog Override** window.

Alternative procedure

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%.

4.14.4 Selecting the tool and base**Description**

A maximum of 32 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.

4.14.5 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 the menu sequence **Monitor > Rob. Position**.

4.14.6 Cartesian jogging with the jog keys

- Precondition**
- The jog mode "Jog keys" is active.
 - Operating mode T1
 - Tool and base have been selected.
(>>> 4.14.4 "Selecting the tool and base" Page 59)

- 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 the menu sequence **Monitor > Rob. Position**.

4.14.7 Configuring the Space Mouse

- Procedure**
1. Open the **Jogging Options** window and select the **Mouse** tab.
(>>> 4.14.1 "'Jogging 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

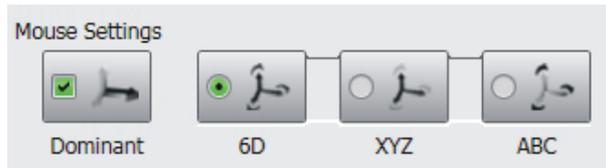


Fig. 4-15: Mouse settings

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

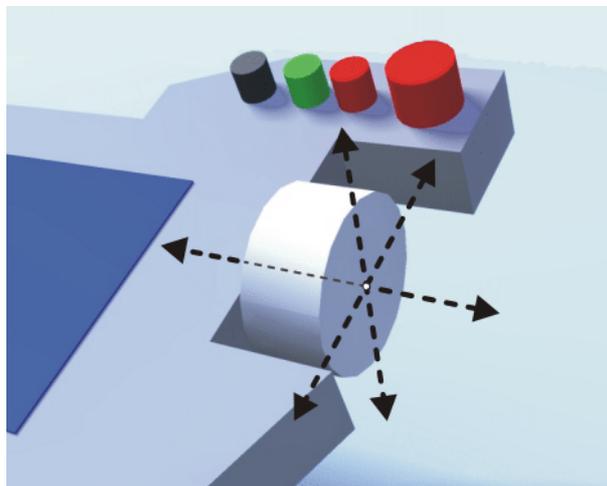


Fig. 4-16: Pushing and pulling the Space Mouse

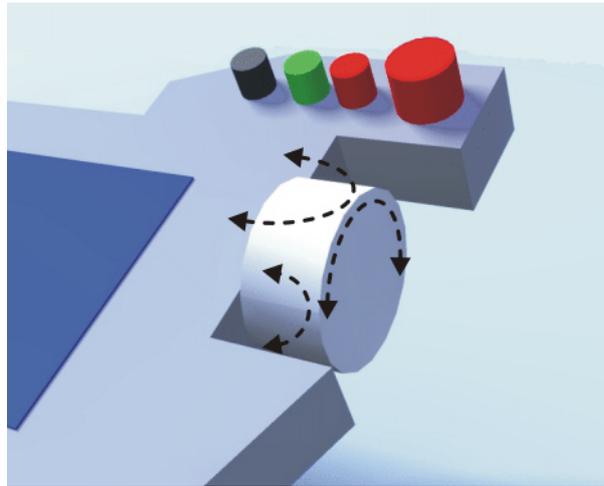


Fig. 4-17: Rotating and tilting the Space Mouse

4.14.8 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.

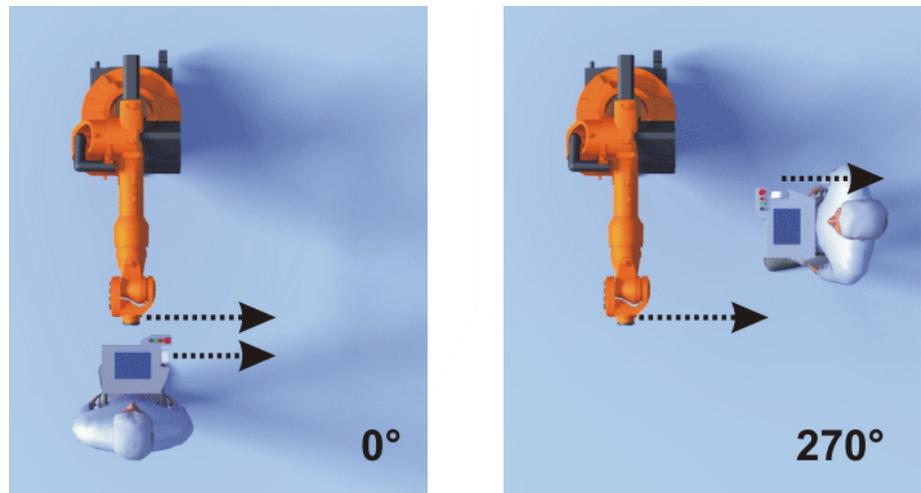


Fig. 4-18: Space Mouse: 0° and 270°

Precondition

- Operating mode T1

Procedure

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



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

4.14.9 Cartesian jogging with the Space Mouse

- Precondition**
- The jog mode “Space Mouse” is active.
 - Operating mode T1
 - Tool and base have been selected.
(>>> 4.14.4 "Selecting the tool and base" Page 59)
 - The Space Mouse is configured.
(>>> 4.14.7 "Configuring the Space Mouse" Page 60)
 - The alignment of the Space Mouse has been defined.
(>>> 4.14.8 "Defining the alignment of the Space Mouse" Page 62)

- 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 the menu sequence **Monitor > Rob. Position**.

4.14.10 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
Continuous	Incremental jogging is deactivated.
100 m / 10°	1 increment = 100 mm or 10°
10 mm / 3°	1 increment = 10 mm or 3°
1 mm / 1°	1 increment = 1 mm or 1°
0.1 mm / 0.005°	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.

4.15 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 are jogged using the jog keys.

Precondition

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

Procedure

1. Select the desired kinematics group, e.g. **External Axes**, on the **Keys** tab in the **Jogging Options** window.



The type and number of kinematics groups available depend on the system configuration.

2. Set jog override.
3. Hold down the enabling switch.
The axes of the selected kinematics group 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.

Description Depending on the system configuration, the following kinematics groups may be available.

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

4.16 Bypassing workspace monitoring

Description Workspaces can be configured for a robot. Workspaces serve to protect the system.

There are 2 types of workspace:

- The workspace is an exclusion zone.
The robot may only move outside the workspace.
- Only the workspace is a permitted zone.
The robot may not move outside the workspace.

Exactly what reactions occur when the robot violates a workspace depends on the configuration. (>>> 6.4 "Configuring workspaces" Page 119)

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.

Precondition

- User group "Expert"
- Operating mode T1

Procedure

1. Select the menu sequence **Configure > Miscellaneous > WorkSpace-Monitor > Override**.
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.

4.17 Monitor functions

4.17.1 Displaying the actual position

Procedure

1. Select the menu sequence **Monitor > Rob. Position**. The Cartesian actual position is displayed.
2. To display the axis-specific actual position, press **Axis spec..**
3. To display the Cartesian actual position again, press **Cartesian**.

Description

Actual position, Cartesian:

The current position (X, Y, Z) and orientation (A, B, C) of the TCP are displayed. In addition to this, the current TOOL and BASE coordinate systems and the Status and Turn are 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 (Cartesian)			
Name	Value	Unit	
Position			
X	2930.00	mm	
Y	0.00	mm	
Z	695.00	mm	
Orientation			
A	0.00	deg	
B	90.00	deg	
C	0.00	deg	
Robot Position			
S	010	bin	
T	000000	bin	

Fig. 4-20: Actual position, Cartesian

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. 4-21: Actual position, axis-specific

4.17.2 Displaying digital inputs/outputs

Procedure

1. Select the menu sequence **Monitor > I/O > Digital I/O**.
2. To display a specific input/output:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.

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

Description

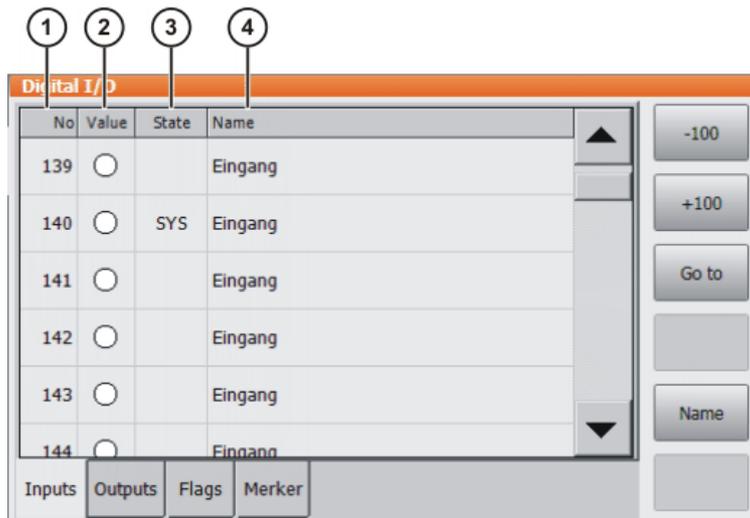


Fig. 4-22: Digital inputs

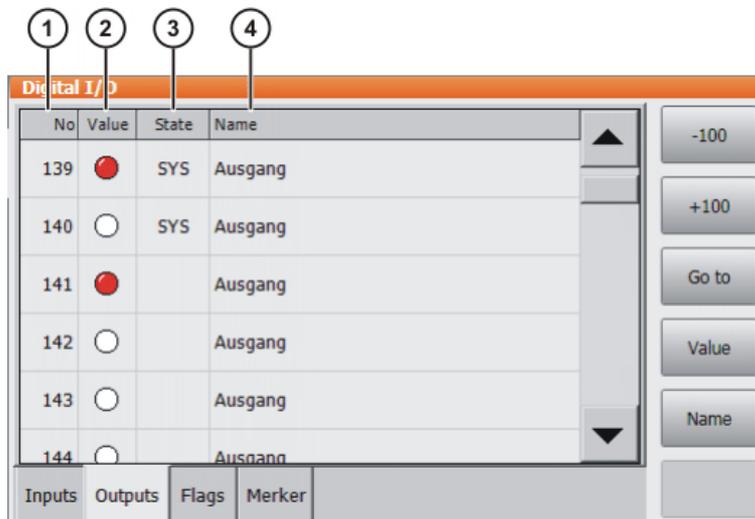


Fig. 4-23: Digital outputs

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 softkeys are available:

Softkey	Description
Value	Toggles the selected output between TRUE and FALSE. Precondition: The enabling switch is pressed. This softkey is not available in AUT EXT mode or for inputs.
Name	The name of the selected input or output can be changed.

4.17.3 Displaying cyclical flags and flags

Procedure

1. Select the menu sequence **Monitor > Merker** or **Flags**.
2. To display a specific cyclical flag (Merker) or flag:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.

The display jumps to the cyclical flag (Merker) or flag with this number.

Description

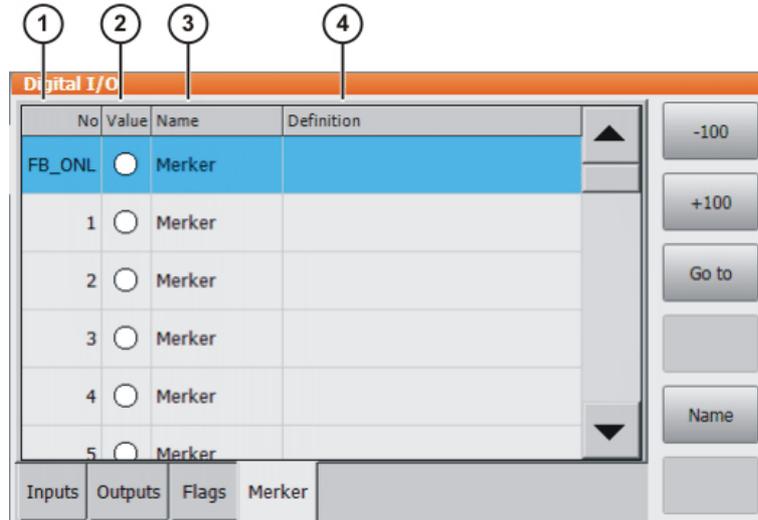


Fig. 4-24: Merker

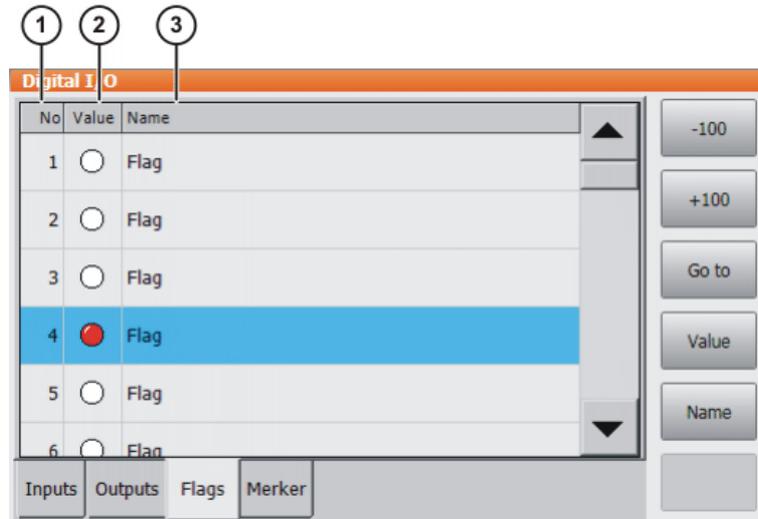


Fig. 4-25: Flags

Item	Description
1	Number of the flag or cyclical flag
2	Value of the flag or cyclical flag. A flag or cyclical flag is indicated in red if it is set. Note: A maximum of 64 cyclical flags may be active simultaneously.
3	Name of the flag or cyclical flag
4	Definition of the cyclical flag The conditions linked to the setting of a cyclical flag are indicated here.

The following softkeys are available:

Softkey	Description
Value	Toggles the selected flag between TRUE and FALSE. Precondition: The enabling switch is pressed. This softkey is not available for cyclical flags.
Name	The name of the selected flag or cyclical flag can be changed.

4.17.4 Displaying analog inputs/outputs

Procedure

1. Select the menu sequence **Monitor > I/O > Analog I/O**.
2. To display a specific input/output:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.

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

Description

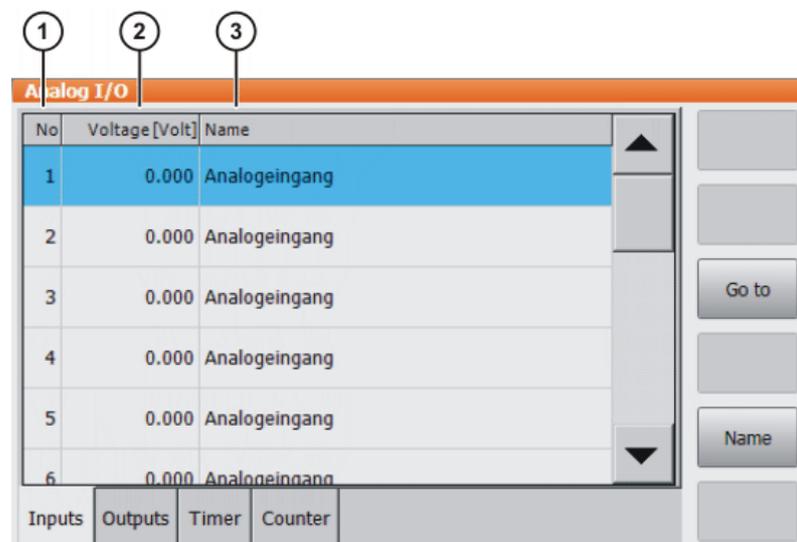


Fig. 4-26: Analog inputs

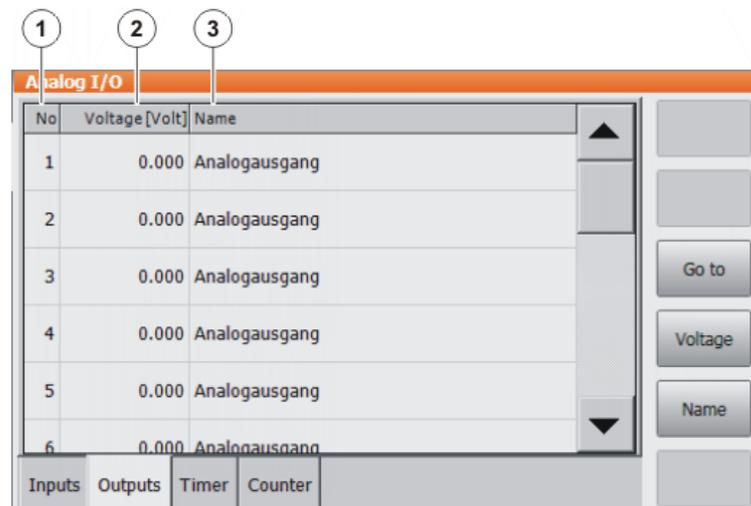


Fig. 4-27: Analog outputs

Item	Description
1	Input/output number
2	Input/output voltage ■ -10 ... 10 V
3	Name of the input/output

The following softkeys are available:

Softkey	Description
Voltage	A voltage can be entered for the selected output. ■ -10 ... 10 V This softkey is not available for inputs.
Name	The name of the selected input or output can be changed.

4.17.5 Displaying timers and counters

Procedure

1. Select the menu sequence **Monitor > Timer** or **Counter**.
2. To display a specific timer or counter:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.

The display jumps to the timer or counter with this number.

Description

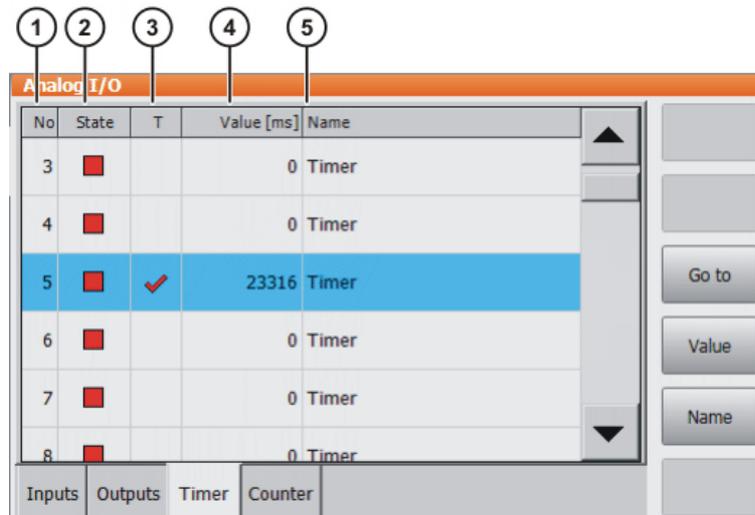


Fig. 4-28: Timers

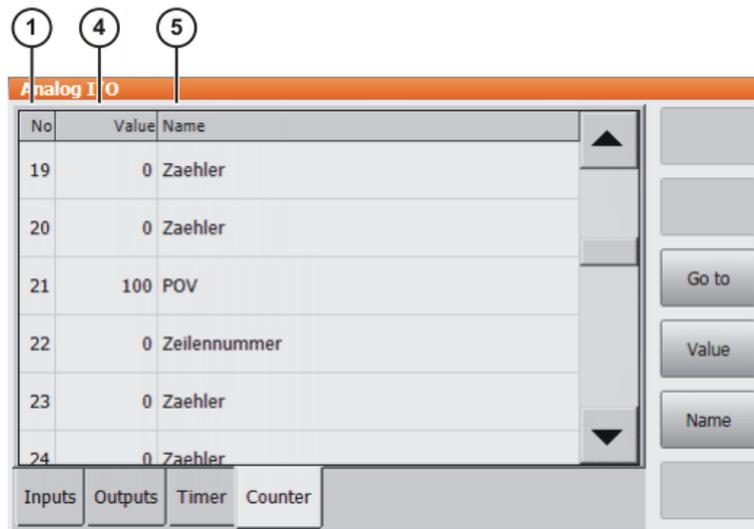


Fig. 4-29: Counter

Item	Description
1	Number of the timer or counter
2	Status of the timer <ul style="list-style-type: none"> ■ If the timer is activated, this is indicated in green. ■ If the timer is deactivated, this is indicated in red.
3	State of the timer <ul style="list-style-type: none"> ■ If the value of the timer is > 0, the timer flag is set (red check mark). ■ If the value of the timer is ≤ 0, no timer flag is set.
4	Value of the timer or counter <ul style="list-style-type: none"> ■ Counter 21 indicates the current program override value in %. ■ Counter 22 indicates the current line number in the editor.
5	Name of the timer or counter

The following softkeys are available:

Softkey	Description
Status	Toggles the selected timer between TRUE and FALSE. Precondition: The enabling switch is pressed. This softkey is not available for counters.
Value	A value can be entered for the selected timer or counter.
Name	The name of the selected timer or counter can be changed.

4.17.6 Displaying binary inputs/outputs

Procedure

1. Select the menu sequence **Monitor > I/O > Binary I/O**.
2. To display a specific input/output:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.

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

Description

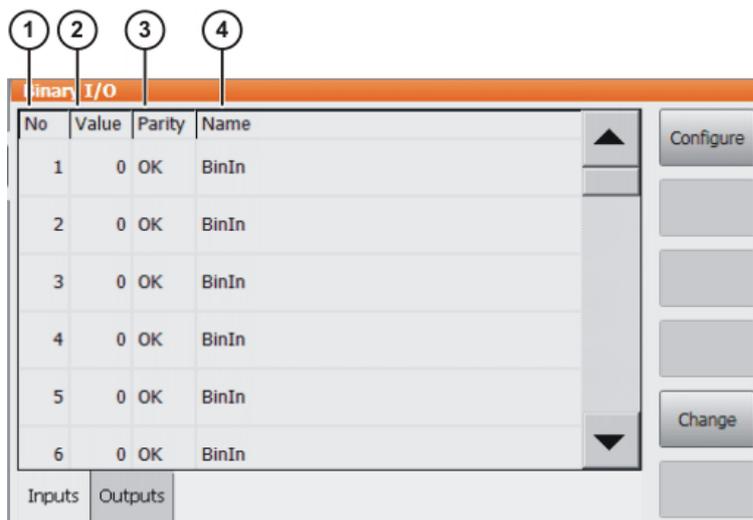


Fig. 4-30: Binary inputs

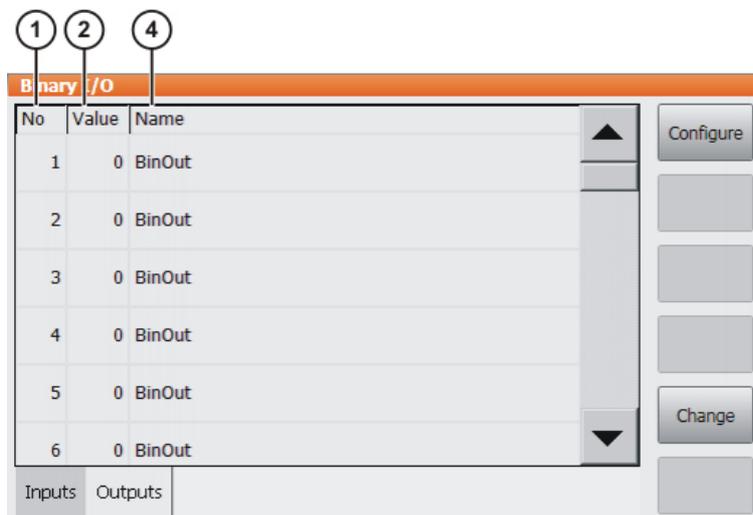


Fig. 4-31: Binary outputs

Item	Description
1	Number of the input/output
2	Value of the input/output
3	Parity of the input
4	Name of the input/output

The following softkeys are available:

Softkey	Description
Configure	Switches to the configuration of the binary I/O. (>>> 6.2 "Configuring binary inputs/outputs" Page 116)
Change	<ul style="list-style-type: none"> ■ The value of the selected output can be changed. Precondition: Enabling switch is pressed. ■ The name of the input or output can be changed.

4.17.7 Displaying process parameters

- Procedure**
1. Select the menu sequence **Monitor > Process Parameter**.
 2. To display a specific parameter:
 - Select any cell in the **No.** column.
 - Enter the number using the keypad.
 The display jumps to the parameter with this number.

Description

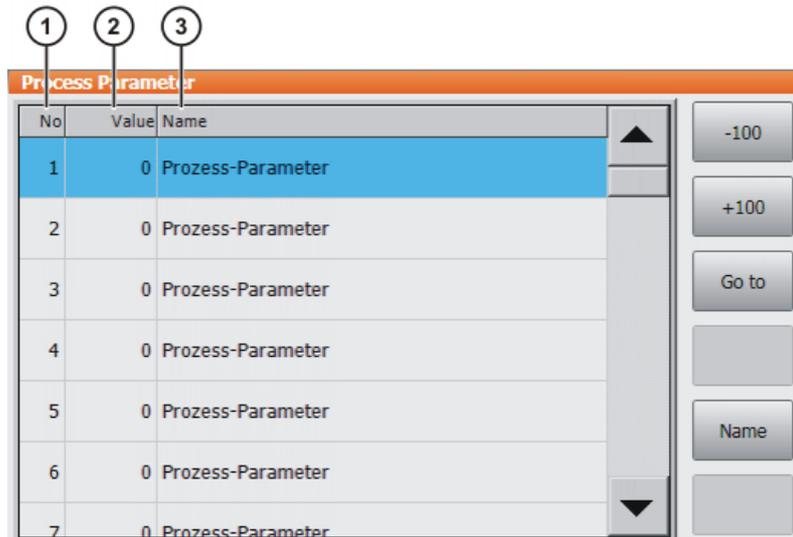


Fig. 4-32: Process parameters

Item	Description
1	Number of the process parameter
2	Value of the process parameter
3	Name of the process parameter

The following softkeys are available:

Softkey	Description
Value	A value can be entered for the selected process parameter. ■ -99,999 ... +99,999 This softkey is only available in the user group "Expert".
Name	The name of the selected process parameter can be changed.

4.17.8 Displaying gun inputs/outputs

- Procedure**
- Select the menu sequence **Monitor > I/O > Gun**.

Description

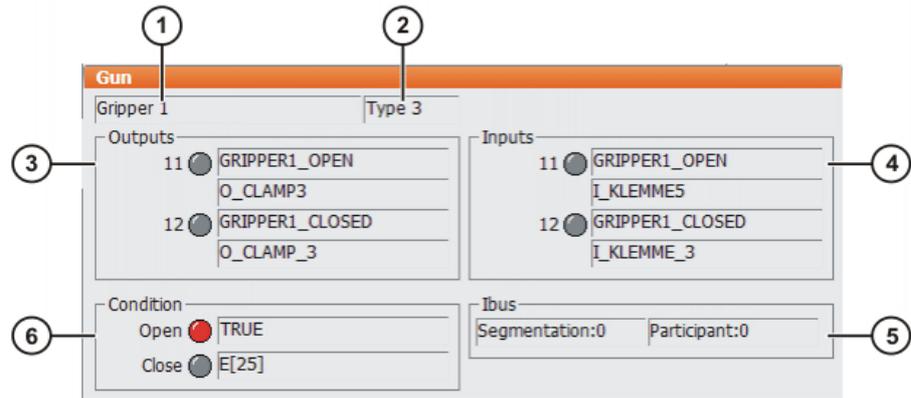


Fig. 4-33: Gun I/O

Item	Description
1	Gun number
2	Gun type
3	Outputs for opening and closing the gun The icon is red if an output is set.
4	Inputs for opening and closing the gun The icon is red if an input is set.
5	Number of the Interbus segment or device The group Ibus is only displayed if a gun is configured with Interbus. (>>> 6.1 "Configuring gun inputs/outputs" Page 115)
6	Condition for manual opening and closing of the gun When the condition has been met, it is indicated in red.

The following softkeys are available:

Softkey	Description
Continue	Switches to the display of the next configured gun.
Previous	Returns to the previous display.

4.17.9 Displaying inputs/outputs for Automatic External

Procedure

- Select the menu sequence **Monitor > I/O > Automatic External**.

Description

St.	Term	Type	Name	Value
1	currently selected Folgennumber	Var	P_NR	1
2	Drives EIN in EXT of	E22	ANTEIN	140
3	Folgestart	E22	SRB	137
4	Format for choice of Folgen (0=Int; 1=1 aus n)	Var	P_TYPE	1
5	First input for Folgennumber	E22	P_FBIT	129
6	Length of Folgennumber	Var	P_LEN	8

Fig. 4-34: Automatic External inputs (detail view)

St.	Term	Type	Name	Value
1	currently selected Folgennumber	Var	P_NR	1
2	First output for Folgenspiegelung	IO	R_FBIT	129
3	Last Folgennumber in Automatic mode	Var	P_OLD	0
4	Robot control ready	IO	BEREIT	137
5	Programming mode	IO	RK23	138
6	Single Step	IO	RK8	139
7	Automatic External	IO	RK9	140
8	Actuator release Start	IO	RK100	141
9	Wait for process input	IO	WPROZ	142
10	Wait for Slave input	IO	WSLAV	143
11	Last point reached	IO	LPKT	145
12	Robot in basic position of Folge	IO	PF0	146
13	Robot on path	IO	SAK	147
14	Internal emergency stop	IO	IntEstop	1002

Fig. 4-35: Automatic External outputs (detail view)

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

Softkeys	Description
Config.	Switches to the configuration of the Automatic External interface. (>>> 6.8 "Configuring Automatic External" Page 129)
Inputs/Outputs	Toggles between the windows for inputs and outputs.
Details/Normal	Toggles between the Details and Normal views.

4.17.10 Displaying and modifying the value of a variable

Precondition ■ Expert user group

Procedure

1. Select the menu sequence **Monitor > Variable > Single**.
The **Variable Overview - Single** window is opened.
2. Enter the name of the variable in the **Name** box.
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 Enter.

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.

5. Enter the desired value in the **New Value** box.

6. Press Enter.

The new value is displayed in the **Current value** box.

Description

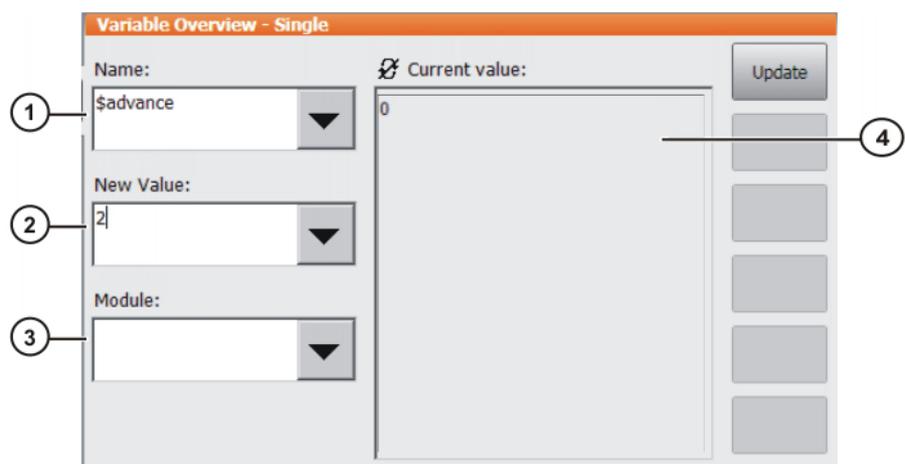


Fig. 4-36: Variable Overview - Single window

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 Module box is irrelevant.
4	This box has two states: <ul style="list-style-type: none"> ■ : The displayed value is not refreshed automatically. ■ : The displayed value is refreshed automatically. Switching between the states: <ul style="list-style-type: none"> ■ Press Refresh.

4.17.11 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. Select the menu sequence **Monitor > Variable > Single**.

The **Variable Overview - Single** window is opened.

2. Enter the following in the **Name** box: =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.

4.17.12 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.

(>>> 6.3 "Configuring the variable overview" Page 117)



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

Procedure

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

Description

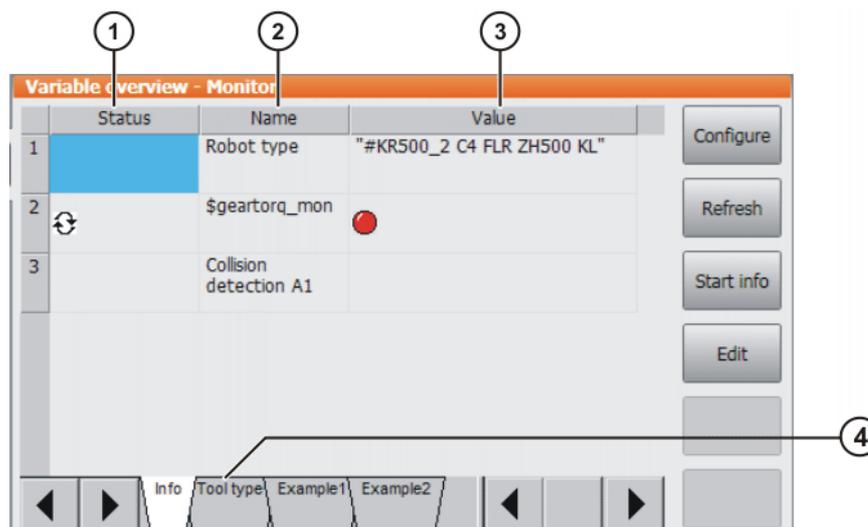


Fig. 4-37: Variable overview - Monitor window

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

The following softkeys are available:

Softkey	Description
Config.	Switches to the configuration of the variable overview. (>>> 6.3 "Configuring the variable overview" Page 117) This softkey is not available in the user group "User".
Refresh all	Refreshes the display.
Cancel Info	Deactivates the automatic refreshing function.
Start info	Activates the automatic refreshing function. A maximum of 12 variables per group can be refreshed automatically.
Edit	Switches the current cell to edit mode so that the name or value can be modified. In the Value column, this softkey changes the state of inputs/outputs (TRUE/FALSE). This softkey is only available in the user group "User" if it has been enabled in the configuration. Note: The values of write-protected variables cannot be changed.

4.17.13 Displaying calibration data

Procedure

1. Select the menu sequence **Setup > Measure > Measurement Points** and the desired menu item:
 - **Tool type**
 - **Base type**
 - **External axis**
2. Enter the number of the tool, base or external kinematic system.
The calibration method and the calibration data are displayed.

4.17.14 Displaying information about the robot and robot controller

Procedure

- Select the menu sequence **Help > Info**.

Description

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

The tabs contain the following information:

Tab	Description
Info	<ul style="list-style-type: none"> ■ Robot controller type ■ Robot controller version ■ User interface version ■ Kernel system version
Robot	<ul style="list-style-type: none"> ■ Robot name ■ Robot type and configuration ■ Service life 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. ■ Number of axes ■ List of external axes ■ Machine data version
System	<ul style="list-style-type: none"> ■ Control PC name ■ Operating system versions ■ Storage capacities
Options	Additionally installed options and technology packages
Comments	Additional comments
Modules	Names and versions of important system files The Save softkey exports the contents of the Modules tab to the file C:\KRC\ROBOTER\LOG\OCXVER.TXT.

4.17.15 Displaying robot data

Procedure

- Select the menu sequence **Setup > Robot Data**.

Description

The screenshot shows the 'Robot Data' window with the following fields and values:

- Serial Number:** 0 (labeled 1)
- Runtime [Hours]:** 0.00 (labeled 2)
- Machine Data:** #KR2210_2 S C4 FLR ZH210 (labeled 3)
- Robot Name:** RName (labeled 4)

Fig. 4-38: 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 Start-up and recommissioning

5.1 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 VSS version. This is ensured if only the machine data from the CD with the VSS, that is also installed, are used.



Warning!

The 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.

KUKA Roboter GmbH Augsburg Germany			
Typ	Type	Type	KR XXX LXXX Xx-2 K-W-F XxxXYZ
Artikel-Nr.	Article-No.	No.d'article	XXXXXXXXXX
Serie-Nr.	Serial-No.	No.Série	XXXXXX
Hergestellt	Manufactured	Fabriqué	2004-02
Gewicht	Weight	Poids	1200 kg
\$TRAFONAME[]="#....."			TRAF01513321654984649352841
...\MADA\			MADA15133216549846493554861

Fig. 5-1: Rating plate

Procedure

1. Select the menu sequence **Setup > 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**.

5.2 Defining hardware options

- Precondition**
- User group "Expert"
 - Or if KUKA.SafeOperation is installed: user group "Safety Maintenance"

Procedure

1. Select the menu sequence **Setup > Service > Safety configuration**.
2. Press **Hardware options**.
3. Modify hardware options and press **Save**.

Description

The screenshot shows a window titled "Hardware options" with the following settings:

- ProfiNet:** A checkbox that is checked.
- Peripheral contactor:** A dropdown menu with "not used" selected.
- Operator safety acknowledgment:** A dropdown menu with "external unit" selected.

At the bottom of the window are three buttons: "Back", "Save", and "Revert changes".

Fig. 5-2: Hardware options

Parameter	Description
ProfiNet	TRUE = Bus is enabled. (Default) FALSE = Bus is disabled.
Peripheral contactor	Main contactor 2 can be used as a peripheral contactor, i.e. as a switching element for the power supply to peripheral devices. not used: Peripheral contactor is not used. (Default) by ProfiSafe: The peripheral contactor is switched by PROFIsafe via input US2. automatically: The peripheral contactor is switched in accordance with the motion enable. If motion enable is present, the contactor is energized.
Operator safety acknowledgement	If the Operator Safety signal is lost and reset in Automatic mode, it must be acknowledged before operation can be continued. external unit: Acknowledgement is given by the system PLC. by acknowledgement button: 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.

5.3 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 "Expert"
- Or if KUKA.SafeOperation is installed: user group "Safety Maintenance"

Procedure

1. Select the menu sequence **Setup > Service > Safety configuration**.
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 softkey **Save** 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.

5.4 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 there is a connection to a higher-level safety controller, Start-up mode cannot be activated. If the robot is in Start-up mode and a connection to a higher-level safety controller is established, Start-up mode is automatically deactivated.



Danger!

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 34)

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

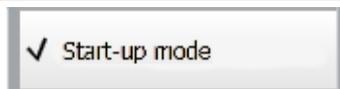
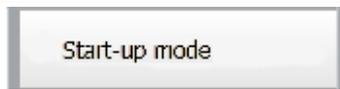
- The safety gate is open.
- The external EMERGENCY STOP is not active.
- Safety stop 2 is not requested.

Precondition

- No connection to a higher-level safety controller
- Operating mode T1

Procedure

- Select the menu sequence **Setup > Service > Start-up mode**.

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

5.5 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. Select the menu sequence **Help > Info**.
2. Check on the **Robot** tab that the positionally accurate robot model is activated. (= specification **Position accurate robot**).

5.6 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.

Precondition

- The robot is mastered.
- There is no load on the robot; i.e. there is no tool, workpiece or supplementary load mounted.

Procedure

- Activate palletizing mode in the program as follows:

```
$PAL_MODE = TRUE
```

Alternative procedure

1. Set \$PAL_MODE to TRUE via the variable correction function.
2. The following message is displayed: *Palletizing mode: Move axis A4 [direction] into position.*
Move A4 in the direction specified in the message [+/-].
3. Once A4 has reached its position, the following message is displayed: *Palletizing mode: Move axis A5 [direction] into position.*
Move A5 in the direction specified in the message [+/-].

Restrictions

- After every cold restart of the robot controller, \$PAL_MODE is automatically set to FALSE.



Recommendation: Integrate \$PAL_MODE = TRUE into the initialization section of all programs for the palletizing robot.

- In the case of robots with palletizing mode active, payload determination with KUKA.LoadDataDetermination is not possible.

**Caution!**

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.

- If palletizing mode is active, the robot cannot be mastered. If mastering is nonetheless required, proceed as follows:
 - a. Remove all loads from the robot.
 - b. Set \$PAL_MODE to FALSE via the variable correction function.
 - c. Master the robot.
 - d. Set \$PAL_MODE to TRUE.
(Not necessary if \$PAL_MODE = TRUE is in the initialization section of all programs for the palletizing robot.)
 - e. Move the robot to the palletizing position.
 - f. Re-attach all loads to the robot.

5.7 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	---
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. (>>> 5.7.6 "Manually unmastering axes" Page 94)
After an impact with an end stop at more than 250 mm/s	
After a collision	

5.7.1 Mastering methods

Overview

A robot can be mastered in the following ways:

- With the EMD (Electronic Mastering Device)
(>>> 5.7.3 "Mastering with the EMD" Page 88)
- With the dial gauge
(>>> 5.7.4 "Mastering with the dial gauge" Page 93)

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



EMD mastering is recommended.

5.7.2 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:

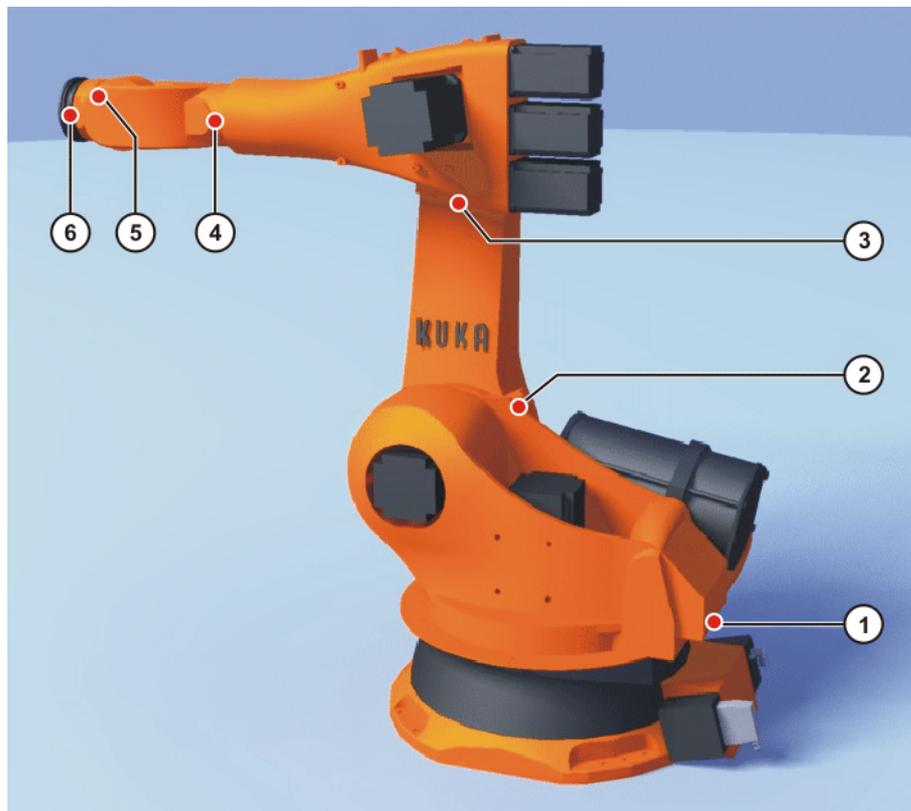


Fig. 5-5: Mastering marks on the robot



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 **Axis** 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.



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°.

5.7.3 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>First mastering</p> <p>(>>> 5.7.3.1 "First mastering with the EMD" Page 88)</p> <p>First mastering is carried out without a load.</p>
2	<p>Teach offset</p> <p>(>>> 5.7.3.2 "Teach offset" Page 91)</p> <p>"Teach offset" is carried out with a load. The difference from the first mastering is saved.</p>
3	<p>If required: Master load with offset</p> <p>(>>> 5.7.3.3 "Master load with offset" Page 92)</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"> ■ 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.

5.7.3.1 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
- User group "Expert"

Procedure

**Caution!**

The EMD must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMD, always remove the signal cable from the EMD first, then remove the EMD from the gauge cartridge. Otherwise, the signal cable could be damaged.

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

1. Select the menu sequence **Setup > Master > EMD > First mastering**.
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.

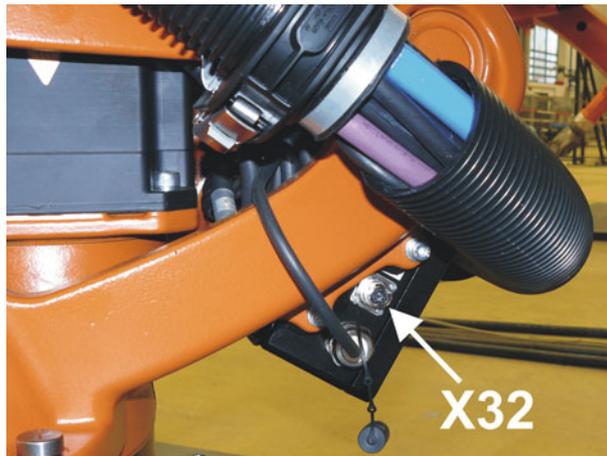


Fig. 5-6: Removing cover from X32

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.

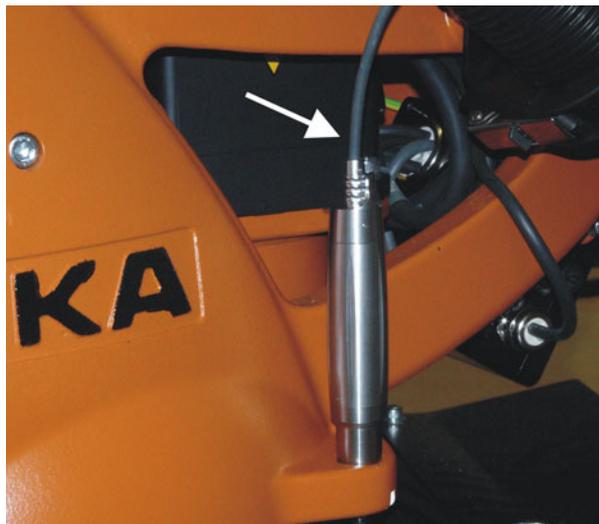


Fig. 5-10: Attaching signal cable to EMD

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.

5.7.3.2 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



Caution!

The EMD must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMD, always remove the signal cable from the EMD first, then remove the EMD from the gauge cartridge. Otherwise, the signal cable could be damaged.

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

1. Select the menu sequence **Setup > Master > EMD > Teach offset**.
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 **Teach**.
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. Confirm with **OK**. 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.

5.7.3.3 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



Caution!

The EMD must always be screwed onto the gauge cartridge without the signal cable attached; only then may the signal cable be attached. When removing the EMD, always remove the signal cable from the EMD first, then remove the EMD from the gauge cartridge. Otherwise, the signal cable could be damaged.

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

1. Select the menu sequence **Setup > Master > EMD > Master load**.
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.

5.7.4 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 **Axes** has been selected.
 - No program is selected.
 - Operating mode T1
 - User group “Expert”
- Procedure**
1. Select the menu sequence **Setup > 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.



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.

5.7.5 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.

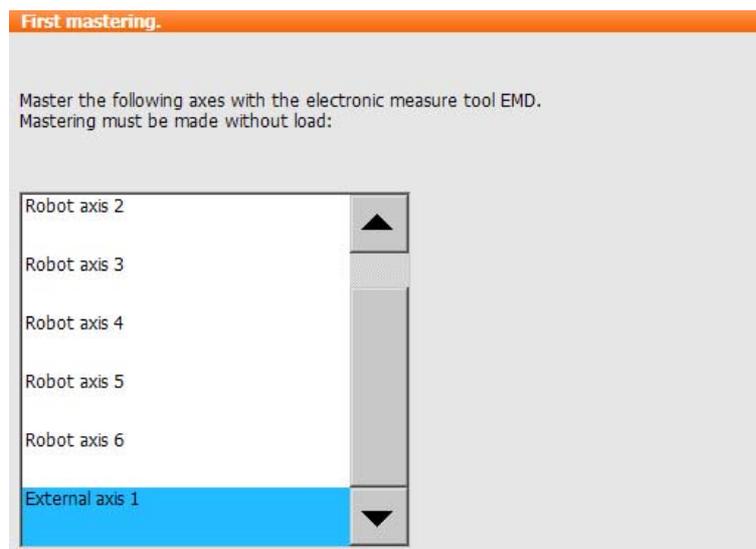


Fig. 5-12: Selection list of axes to be mastered



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.

5.7.6 Manually unmastering axes

Description

The mastering values of the individual axes can be deleted. The axes do not move during unmastering.



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.



Warning!

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.

Precondition ■ No program is selected.

Procedure

1. Select the menu sequence **Setup > Master > UnMaster**. A window opens.
2. Select the axis to be unmastered.
3. Press **UnMaster**. The mastering data of the axis are deleted.
4. Repeat steps 2 and 3 for all axes to be unmastered.
5. Close the window.

5.8 Calibration

5.8.1 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.

It may be desirable to change the default tool direction, for example, when working with the spline function generator.



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.

5.8.2 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. (>>> 5.8.4 "Fixed tool calibration" Page 105)

Advantages of the 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 32 TOOL coordinate systems can be saved. Variable: TOOL_DATA[1...32].

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

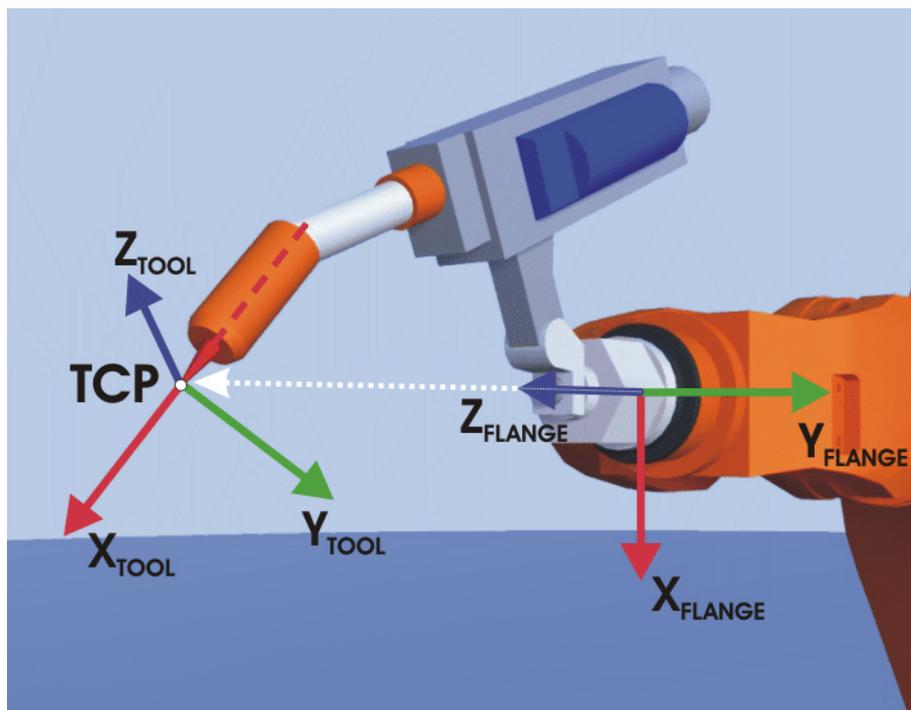


Fig. 5-13: TCP calibration principle

Overview

Tool calibration consists of 2 steps:

Step	Description
1	<p>Definition of the origin of the TOOL coordinate system</p> <p>The following methods are available:</p> <ul style="list-style-type: none"> ■ XYZ 4-Point (>>> 5.8.2.1 "TCP calibration: XYZ 4-Point method" Page 97) ■ XYZ Reference (>>> 5.8.2.2 "TCP calibration: XYZ Reference method" Page 98)
2	<p>Definition of the orientation of the TOOL coordinate system</p> <p>The following methods are available:</p> <ul style="list-style-type: none"> ■ ABC World (>>> 5.8.2.3 "Defining the orientation: ABC World method" Page 99) ■ ABC 2-Point (>>> 5.8.2.4 "Defining the orientation: ABC 2-Point method" Page 101)

If the calibration data are already known, they can be entered directly.

(>>> 5.8.2.5 "Numeric input" Page 102)

5.8.2.1 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.

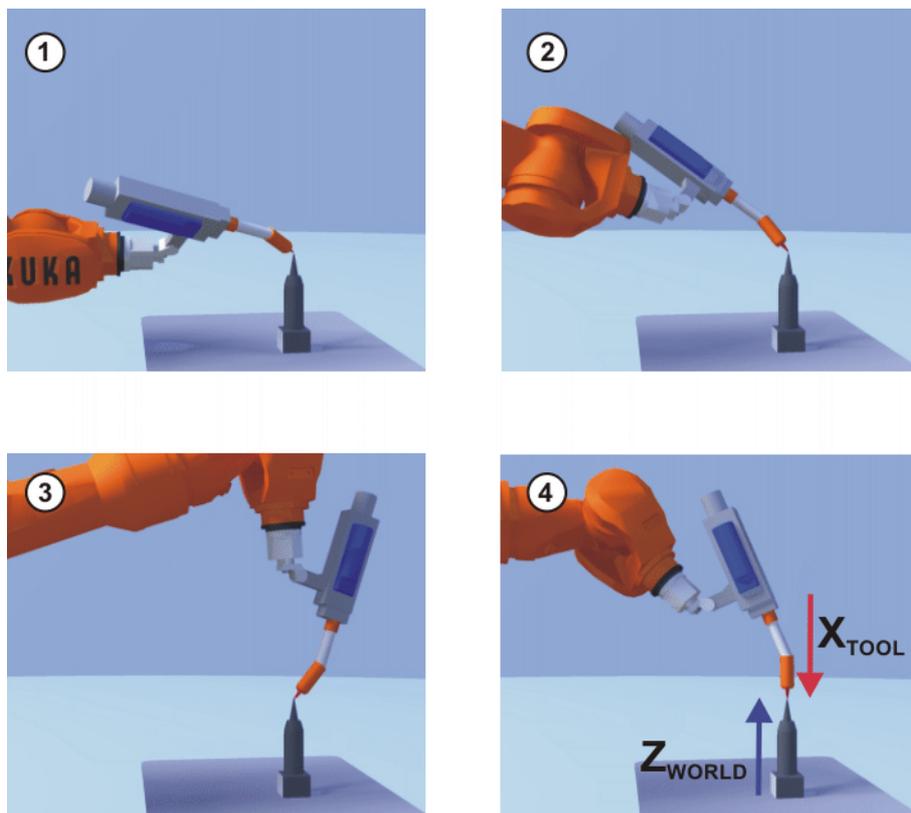


Fig. 5-14: XYZ 4-Point method

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Tool > XYZ 4-Point**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Continue**.
3. Move the TCP to a reference point. Press **Measure**. Confirm the request for confirmation with **Yes**.
4. Move the TCP to the reference point from a different direction. Press **Measure**. Confirm the request for confirmation with **Yes**.
5. Repeat step 4 twice.
6. Press **Save**.

5.8.2.2 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.

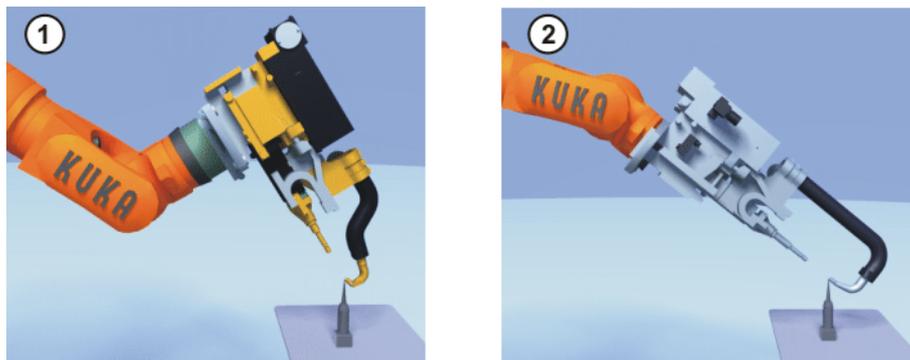


Fig. 5-15: XYZ Reference method

- Precondition**
- A previously calibrated tool is mounted on the mounting flange.
 - Operating mode T1
- Preparation**
- Calculate the TCP data of the calibrated tool:
1. Select the menu **Setup > Measure > Tool > XYZ Reference**.
 2. Enter the number of the calibrated tool.
 3. Note the X, Y and Z values.
 4. Close the window.
- Procedure**
1. Select the menu **Setup > Measure > Tool > XYZ Reference**.
 2. Assign a number and a name for the new tool. Confirm with **Continue**.
 3. Enter the TCP data of the calibrated tool. Confirm with **Continue**.
 4. Move the TCP to a reference point. Press **Measure**. Confirm 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 **Measure**. Confirm the request for confirmation with **Yes**.
 7. Press **Save**.

5.8.2.3 Defining the orientation: ABC World method

Description

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:

- **5D**: 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
- **6D**: The directions of all 3 axes are communicated to the robot controller.
Area of application: e.g. for weld guns, grippers or adhesive nozzles

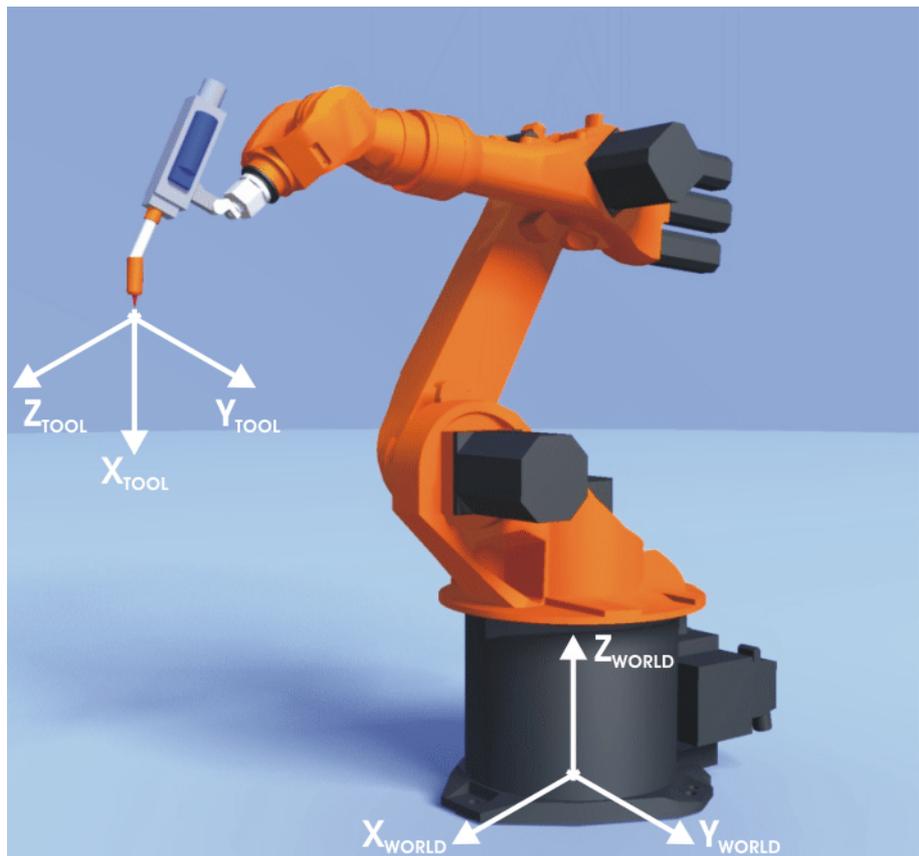


Fig. 5-16: ABC World method

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Tool > ABC World**.
2. Enter the number of the tool. Confirm with **Continue**.
3. Enter the tool direction of the tool in the box **Tool working direction** (by default, only **X-Axis** is available).
4. Select a variant in the box **5D/6D**. Confirm with **Continue**.
5. If **5D** is selected:
Align $+X_{TOOL}$ parallel to $-Z_{WORLD}$. ($+X_{TOOL}$ = tool direction)
- If **6D** is selected:
Align the axes of the TOOL coordinate system as follows.
 - $+X_{TOOL}$ parallel to $-Z_{WORLD}$. ($+X_{TOOL}$ = tool direction)
 - $+Y_{TOOL}$ parallel to $+Y_{WORLD}$
 - $+Z_{TOOL}$ parallel to $+X_{WORLD}$



This is the default alignment. Depending on customer-specific settings, the axes may be aligned differently.

6. Confirm with **Continue**.
7. Press **Measure**. Confirm the request for confirmation with **Yes**.
8. Press **Save**.

5.8.2.4 Defining the orientation: ABC 2-Point method

Description

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.

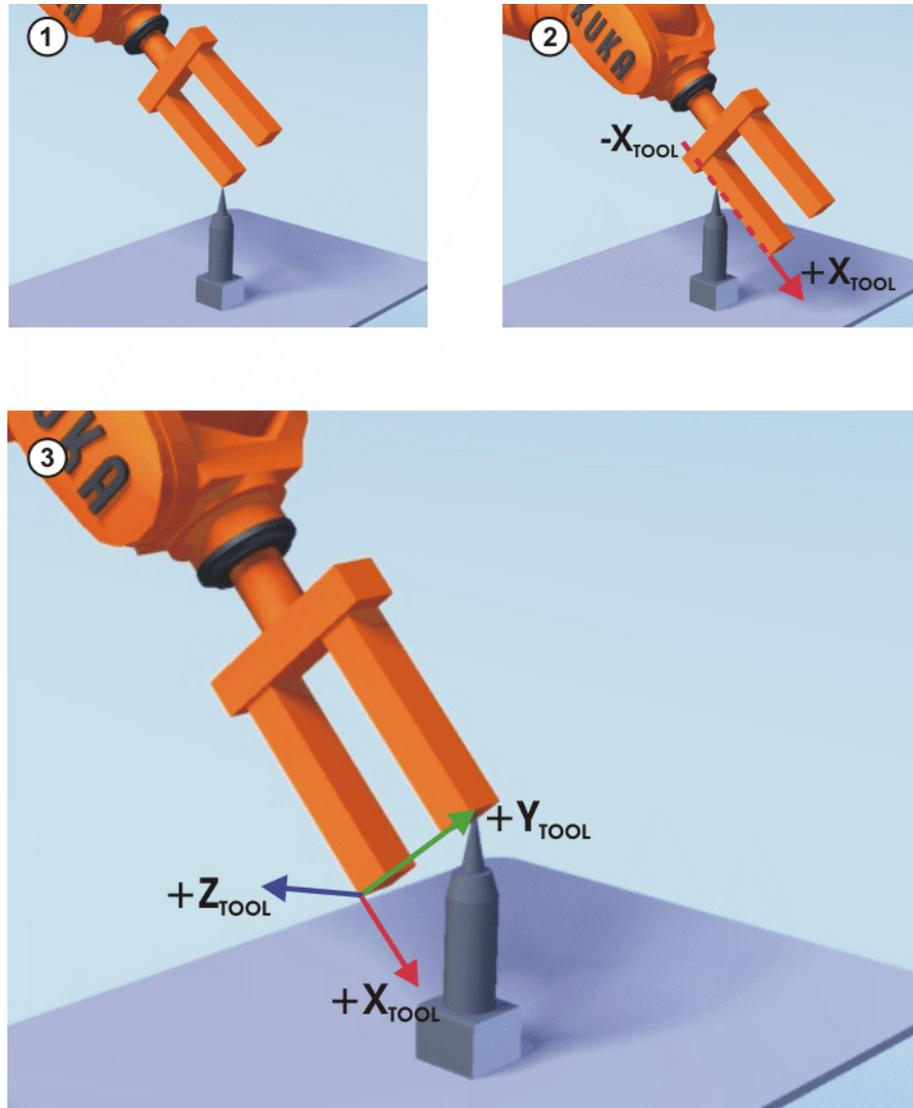


Fig. 5-17: ABC 2-Point method

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Tool > ABC 2-Point**.
2. Enter the number of the mounted tool. Confirm with **Continue**.
3. Move the TCP to any reference point. Press **Measure**. Confirm the request for confirmation with **Yes**.
4. By default, the tool direction is the X axis. 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 **Measure**. Confirm 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 **Measure**. Confirm the request for confirmation with **Yes**.

6. Press **Save**.

5.8.2.5 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

Procedure

1. Select the menu **Setup > Measure > Tool > Numeric Input**.
2. Assign a number and a name for the tool to be calibrated. Confirm with **Continue**.
3. Enter data. Confirm with **Continue**.
4. Either: press **Save** to save the data, then close the window.
Or: press **Load data** in order to be able to enter the tool load data directly.
5. Only if **Load data** has been selected in step 4:
 - a. Enter data.
(>>> 5.9.3 "Entering payload data" Page 112)
Confirm with **Continue**.
 - b. Press **Save** to save the data. Then close the window.

5.8.3 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.

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 (>>> 5.8.3.1 "3-point method" Page 103)
- Indirect method (>>> 5.8.3.2 "Indirect method" Page 104)

If the calibration data are already known, they can be entered directly.
(>>> 5.8.3.3 "Numeric input" Page 104)

5.8.3.1 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.

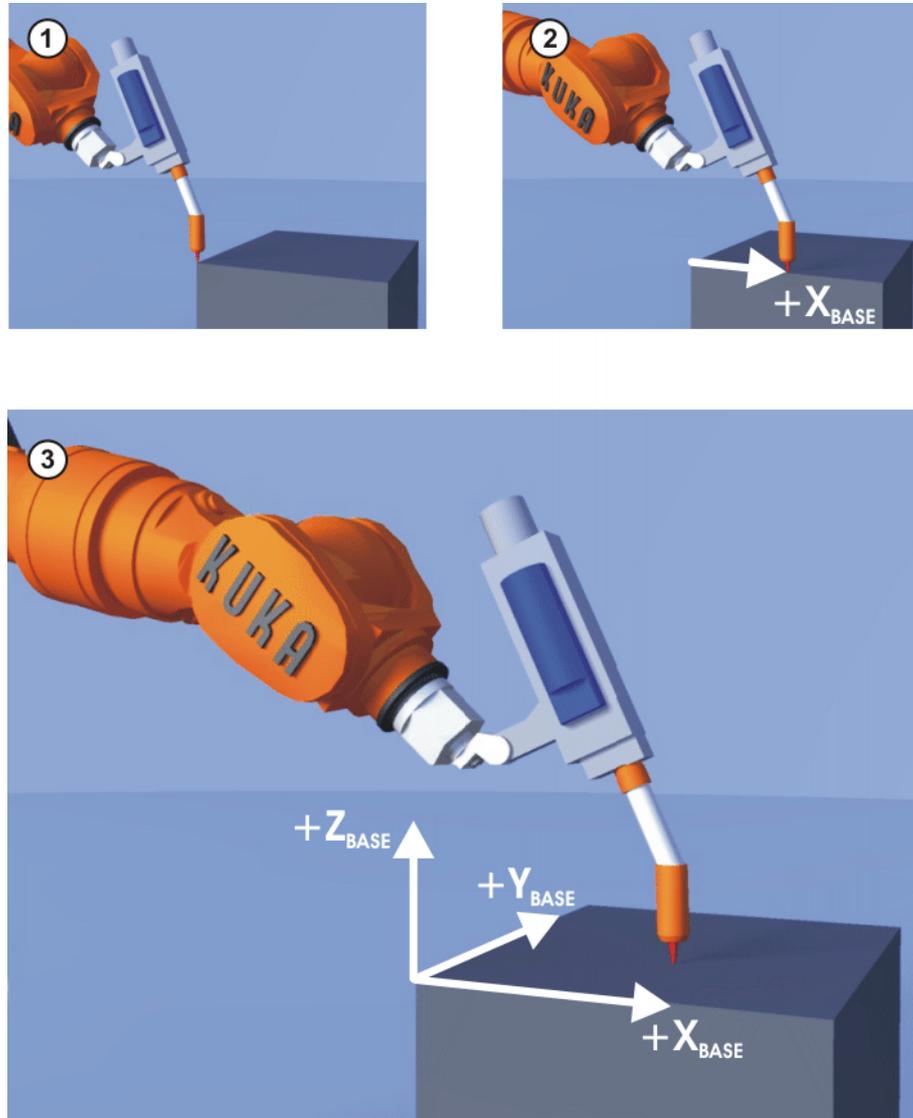


Fig. 5-18: 3-point method

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Base > ABC 3-Point**.
2. Assign a number and a name for the base. Confirm with **Continue**.
3. Enter the number of the mounted tool. Confirm with **Continue**.
4. Move the TCP to the origin of the new base.
Press **Measure**. Confirm the request for confirmation with **Yes**.
5. Move the TCP to a point on the positive X axis of the new base.
Press **Measure**. Confirm the request for confirmation with **Yes**.
6. Move the TCP to a point in the XY plane with a positive Y value.
Press **Measure**. Confirm the request for confirmation with **Yes**.

7. Press **Save**.

5.8.3.2 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.

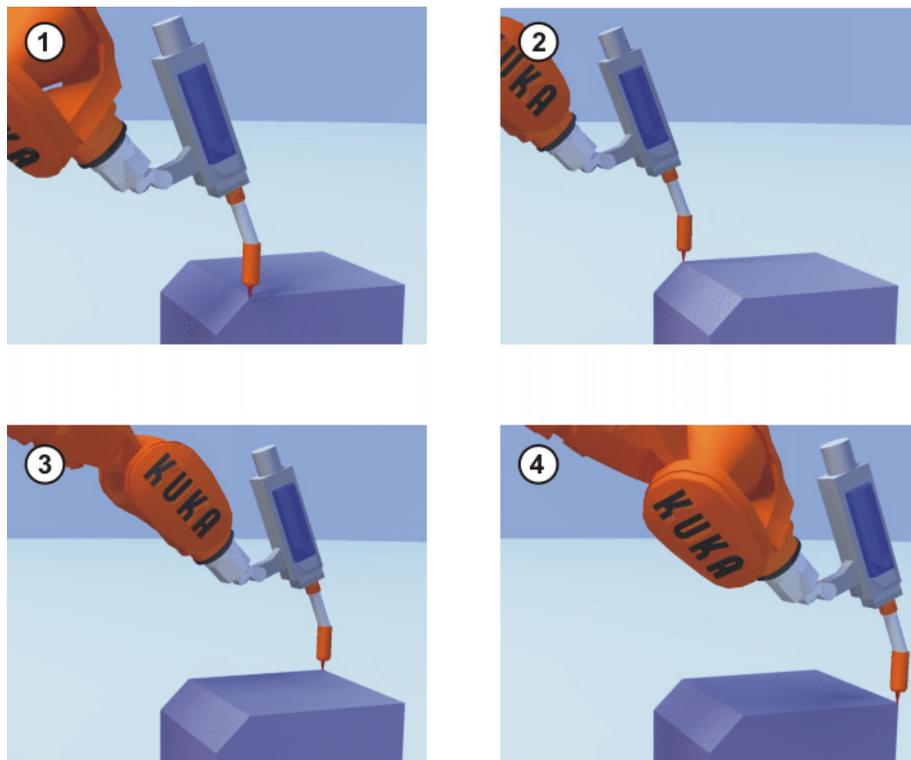


Fig. 5-19: Indirect method

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. Select the menu **Setup > Measure > Base > Indirect**.
2. Assign a number and a name for the base. Confirm with **Continue**.
3. Enter the number of the mounted tool. Confirm with **Continue**.
4. Enter the coordinates of a known point in the new base and move the TCP to this point.
Press **Measure**. Confirm the request for confirmation with **Yes**.
5. Repeat step 4 three times.
6. Press **Save**.

5.8.3.3 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

- Procedure**
1. Select the menu **Setup > Measure > Base > Numeric Input**.
 2. Assign a number and a name for the base. Confirm with **Continue**.
 3. Enter data. Confirm with **Continue**.
 4. Press **Save**.

5.8.4 Fixed tool calibration

Overview Calibration of a fixed tool consists of 2 steps:

Step	Description
1	<p>Calibration of the TCP of the fixed tool</p> <p>(>>> 5.8.4.1 "Calibrating an external TCP" Page 105)</p> <p>The TCP of a fixed tool is called an external TCP. If the calibration data are already known, they can be entered directly.</p> <p>(>>> 5.8.4.2 "Entering the external TCP numerically" Page 107)</p>
2	<p>Calibration of the workpiece</p> <p>The following methods are available:</p> <ul style="list-style-type: none"> ■ Direct method (>>> 5.8.4.3 "Workpiece calibration: direct method" Page 107) ■ Indirect method (>>> 5.8.4.4 "Workpiece calibration: indirect method" Page 108)

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 32 TOOL coordinate systems can be saved.

5.8.4.1 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.

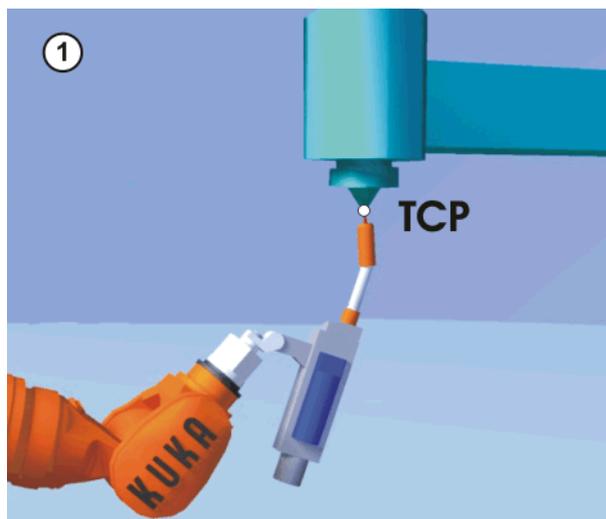


Fig. 5-20: Moving to the external TCP

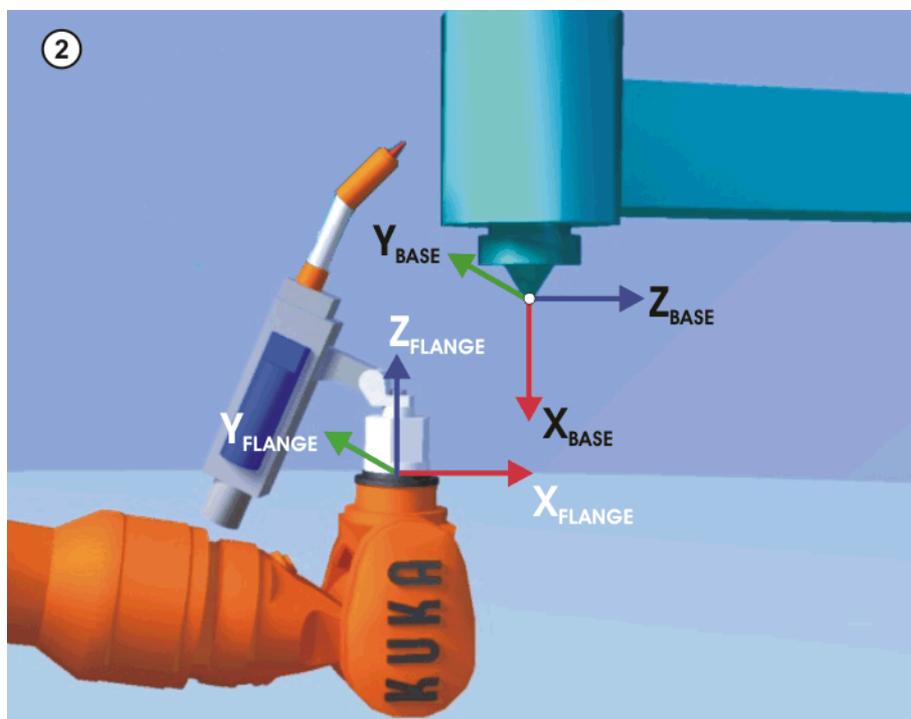


Fig. 5-21: Aligning the coordinate systems parallel to one another

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Fixed tool > Tool**.
2. Assign a number and a name for the fixed tool. Confirm with **Continue**.
3. Enter the number of the calibrated tool. Confirm with **Continue**.
4. Enter the tool direction of the tool in the box **Tool working direction** (by default, only **X-Axis** is available).
5. Select a variant in the box **5D/6D**. Confirm with **Continue**.
6. Move the TCP of the calibrated tool to the TCP of the fixed tool. Press **Measure**. Answer the request for confirmation with **Yes**.
7. 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 $+X_{FLANGE}$



This is the default alignment. Depending on customer-specific settings, the axes may be aligned differently.

8. Press **Measure**. Answer the request for confirmation with **Yes**.
9. Press **Save**.

5.8.4.2 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)

Procedure

1. Select the menu **Setup > Measure > Fixed tool > Numeric Input**.
2. Assign a number and a name for the fixed tool. Confirm with **Continue**.
3. Enter data. Confirm with **Continue**.
4. Press **Save**.

5.8.4.3 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.

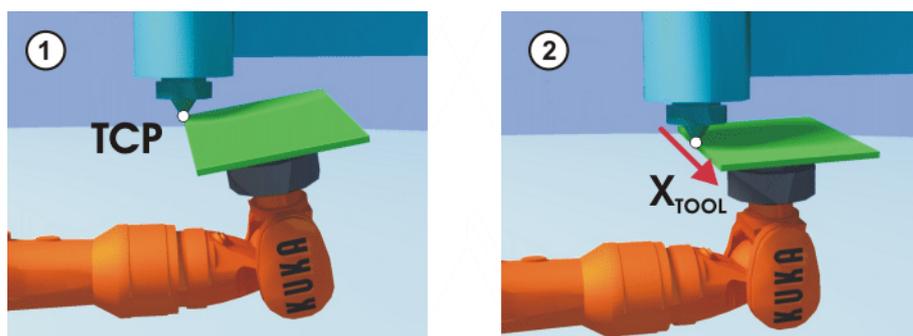


Fig. 5-22

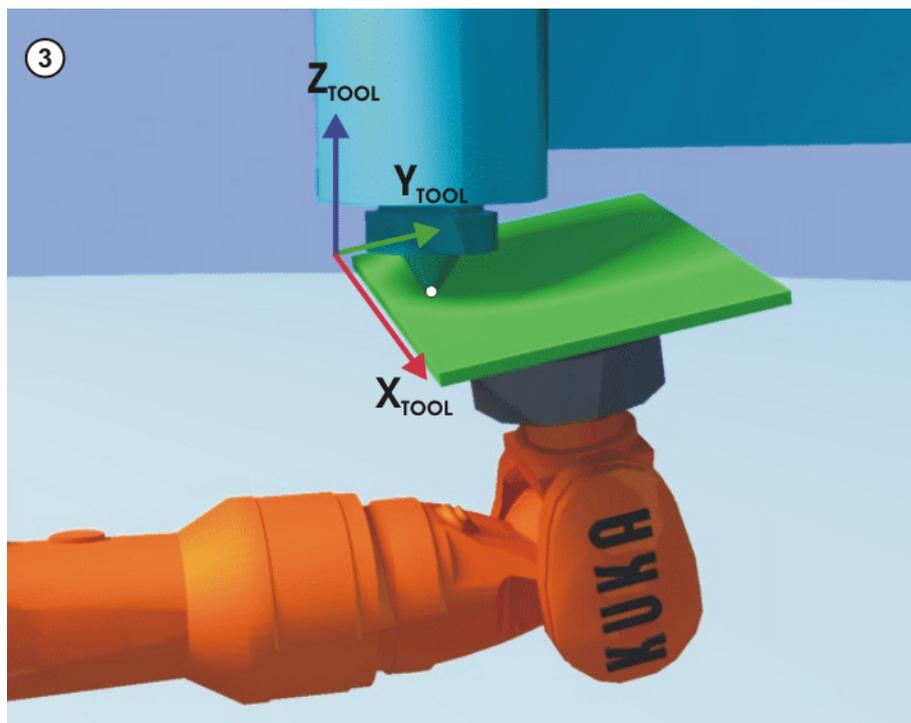


Fig. 5-23: Workpiece calibration: direct method

Precondition

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

Procedure

1. Select the menu **Setup > Measure > Fixed tool > Workpiece > Direct measuring**.
2. Assign a number and a name for the workpiece. Confirm with **Continue**.
3. Enter the number of the fixed tool. Confirm with **Continue**.
4. Move the origin of the workpiece coordinate system to the TCP of the fixed tool.
Press **Measure**. Confirm 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 **Measure**. Confirm 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 **Measure**. Confirm the request for confirmation with **Yes**.
7. Press **Save**.

5.8.4.4 Workpiece calibration: indirect method

Description

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.

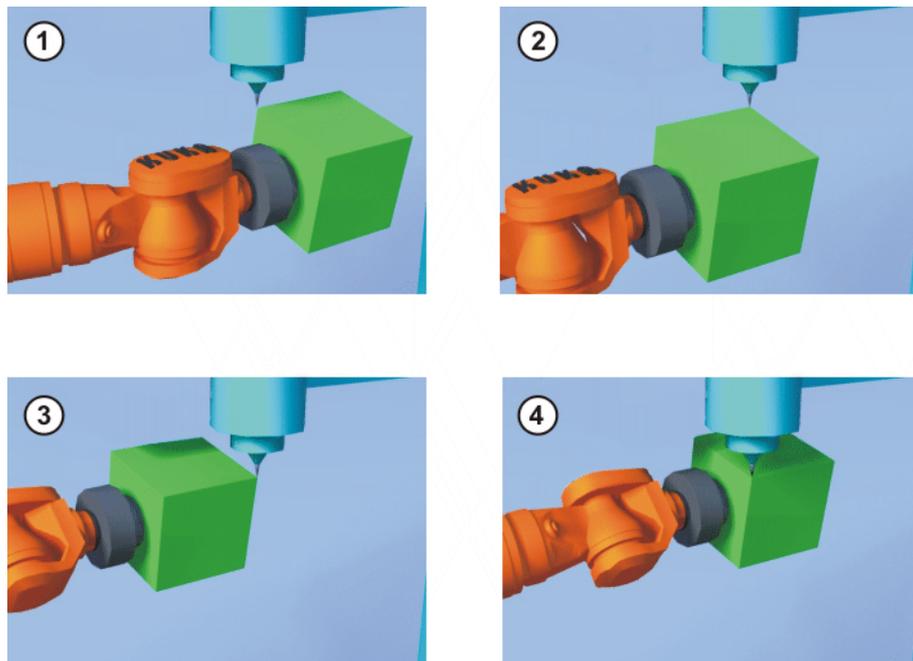


Fig. 5-24: Workpiece calibration: indirect method

- Precondition**
- A previously calibrated fixed tool is mounted.
 - The workpiece to be calibrated is mounted on the mounting flange.
 - The coordinates of 4 points of the new workpiece are known, e.g. from CAD data. The 4 points are accessible to the TCP.
 - Operating mode T1
- Procedure**
1. Select the menu **Setup > Measure > Fixed tool > Workpiece > Indirect measuring**.
 2. Assign a number and a name for the workpiece. Confirm with **Continue**.
 3. Enter the number of the fixed tool. Confirm with **Continue**.
 4. Enter the coordinates of a known point on the workpiece and move this point to the TCP of the fixed tool.
Press **Measure**. Confirm the request for confirmation with **Yes**.
 5. Repeat step 4 three times.
 6. Press **Save**.

5.8.5 Renaming the tool/base

- Procedure**
1. Select the menu **Setup > Measure > Tool or Base > Change name**.
 2. Select the tool or base and press the **Name** softkey.
 3. Enter the new name and confirm with the **Save** softkey.

5.8.6 Calibrating the 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.



Fig. 5-25: ROBROOT kinematic system – linear unit

The linear unit is a ROBROOT kinematic system in which \$WORLD is offset to \$ROBROOT, i.e. the position of \$WORLD changes every time the linear unit is moved. The reference point for calibration of the linear unit is thus the ERSYSROOT coordinate system.

The ERSYSROOT coordinate system is a Cartesian coordinate system which is located at the root point of the linear unit. It defines the offset between the root point of the linear unit and the robot base flange. The position of the coordinate system (\$ERSYSROOT) can be configured in the machine data.



If the calibration data are already known, they can be entered directly. (>>> 5.8.6.2 "Numeric input" Page 111)

5.8.6.1 Moving to the reference point from 2 different positions

Description

During calibration of the linear unit, the TCP of a calibrated tool is moved twice to a reference point in space. The reference point can be freely selected. The position of the robot on the linear unit from which the reference point is approached must be different each time. The robot controller uses the different positions to calculate the alignment of the robot on the linear unit.



The 2 positions of the robot on the linear unit from which the reference point is approached must be sufficiently far apart.

Precondition

- The machine data of the kinematic system, including transformation, have been loaded into the robot controller.
- A previously calibrated tool is mounted on the mounting flange.
- Operating mode T1 or T2



Detailed information about creating machine data for external kinematic systems can be found in the documentation **External axes**.

Procedure

1. Select the menu sequence **Setup > Measure > External kinematic > Linear Track**.
2. Enter the number of the mounted tool. Confirm with **Continue**.
3. Move the linear unit with the jog key "+".
4. Specify whether the linear unit is moving to "+" or "-". Confirm with **Continue**.

5. Move linear unit.
6. Move the TCP to the reference point.
7. Press **Measure**. Confirm with **Continue**.
8. Repeat steps 5 to 7.
9. Press **Save**.

5.8.6.2 Numeric input

- Precondition**
- No program is open or selected.
 - The following numerical values are known, e.g. from CAD data:
 - Distance between the robot base flange and the origin of the ERSYS-ROOT coordinate system (X, Y, Z)
 - Orientation of the robot base flange relative to the ERSYSROOT coordinate system (A, B, C)
- Procedure**
1. Select the menu **Setup > Measure > External kinematic > Linear Track - (numeric)**.
 2. Move the linear unit with the jog key "+".
 3. Specify whether the linear unit is moving to "+" or "-". Confirm with **Continue**.
 4. Enter data. Confirm with **Continue**.
 5. Press **Save**.
 6. Confirm the request for confirmation with **Yes**. The position data are applied automatically.



Caution!

If the linear unit configuration is changed, the robot controller recalculates positions that have already been programmed. In this case, the taught points must be checked in jog mode. Otherwise, collisions and significant damage to property could occur.

5.9 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.



Warning!

If a robot is operated with incorrect load data or an unsuitable load, this can result in danger to life and limb and/or substantial material damage.

Sources

Load data can be obtained from the following sources:

- Software option KUKA.LoadDataDetermination (only for payloads on the flange)
- Manufacturer information
- Manual calculation
- CAD programs

5.9.1 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.



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

5.9.2 Calculating payloads with KUKA.LoadDataDetermination

KUKA.LoadDataDetermination can be used to calculate payloads exactly and transfer them to the robot controller.



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

5.9.3 Entering payload data

- Description** 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.Load-DataDetermination and the robot is suitable for these payloads.
- Procedure**
1. Select the menu **Setup > Measure > Tool > Payload data**.
 2. Enter the number of the tool in the box **Tool no.**. Confirm with **Continue**.
 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.)
 4. Confirm with **Continue**.
 5. Press **Save**.

5.9.4 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. Select the menu sequence **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**.

6 Configuration

6.1 Configuring gun inputs/outputs

- Procedure**
1. Select the menu sequence **Configure > I/O > Gun**.
 2. Set the desired gun parameters.
 3. Save the configuration with **Change**.

Description

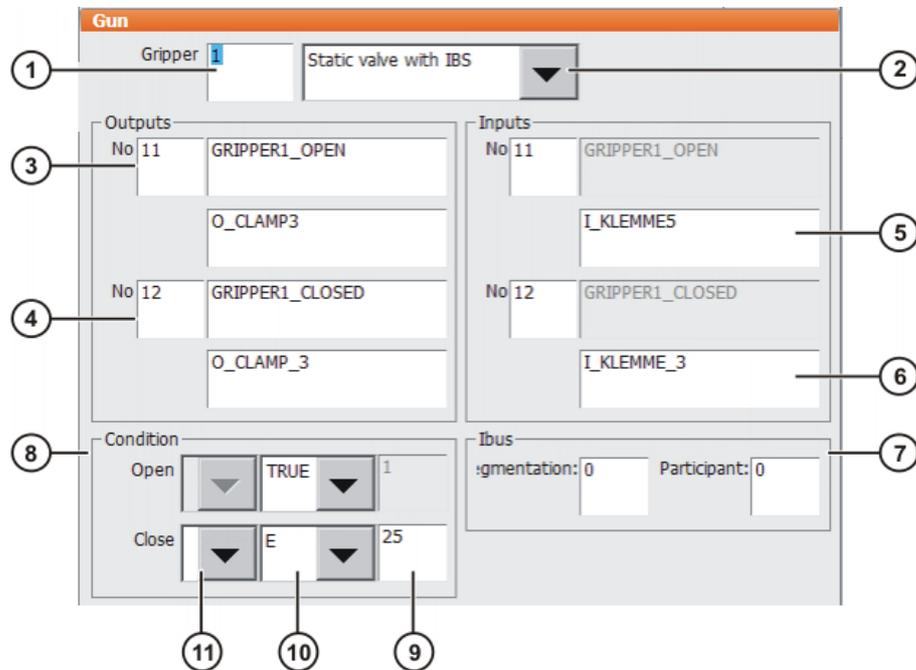


Fig. 6-1: Configuring gun inputs/outputs

Item	Description
1	Enter the gun number. A maximum of 32 guns is possible.
2	Select the gun type. The following gun types are available for selection as standard: <ul style="list-style-type: none"> ■ Static valve (type 1) ■ Pulse valve (type 2) ■ Static valve with IBS (type 3) ■ Pulse valve with IBS (type 4) ■ I/O servo gun (type 6) ■ No gun <p>Note: Optionally, the servo gun (type 5) is available. Precondition: servo gun has been installed.</p>
3	<ul style="list-style-type: none"> ■ Define an output for opening the gun. ■ Define a name for the function and a name for the assignment of the output (maximum 20 characters). The name for the function is automatically applied in the Inputs group.
4	<ul style="list-style-type: none"> ■ Define an output for closing the gun. ■ Define a name for the function and a name for the assignment of the output (maximum 20 characters). The name for the function is automatically applied in the Inputs group.

Item	Description
5	<ul style="list-style-type: none"> ■ Define an input for opening the gun. ■ Define a name for the assignment. <p>The Inputs group is not displayed if a servo gun is selected. (Type 5 or type 6)</p>
6	<ul style="list-style-type: none"> ■ Define an input for closing the gun. ■ Define a name for the assignment. <p>The Inputs group is not displayed if a servo gun is selected. (Type 5 or type 6)</p>
7	<p>Enter the number of the Interbus segment and device.</p> <p>The group Ibus is only displayed if a gun with Interbus is selected. (Type 3 or type 4)</p> <p>Note: To deactivate an active alternative bus segment, the value 0 must be entered in both boxes.</p> <p>(>>> 9.4.29 "Inline form "IBG"" Page 216)</p>
8	Define a condition for manual opening and closing of the gun.
9	<p>Input box for the operand number</p> <p>The input box is not available if the operands TRUE and FALSE are selected.</p>
10	<p>The following operands are available: TRUE, FALSE, E, A, M, F, T, S</p> <p>(>>> 9.4.1 "Boolean operands" Page 201)</p>
11	<p>The following operators are available: _, !</p> <p>(>>> 9.4.3 "Operators" Page 202)</p> <p>The list box is not available if the operands TRUE and FALSE are selected.</p>

The following softkeys are available:

Softkey	Description
Previous	Switches to the previous gun number.
Continue	Switches to the next gun number.
Change	Saves the configuration.

6.2 Configuring binary inputs/outputs

Procedure

1. Select the menu sequence **Configure > I/O > Binary I/O**.
2. In the **Start** column, select the cell to be edited and press **Change**.
3. Enter the desired start bit of the input/output and save it by pressing **OK**.
4. Repeat steps 2 and 3 for the bit width of the input/output to be edited in the **Length** column.
5. If necessary, activate the parity bit. To do so, select the desired cell in the **Parity** column and select the parity of the input/output by pressing **Change**.
6. Repeat steps 2 and 3 as required to change the name of the input/output.
7. Exit the configuration by pressing **Close**.

Description

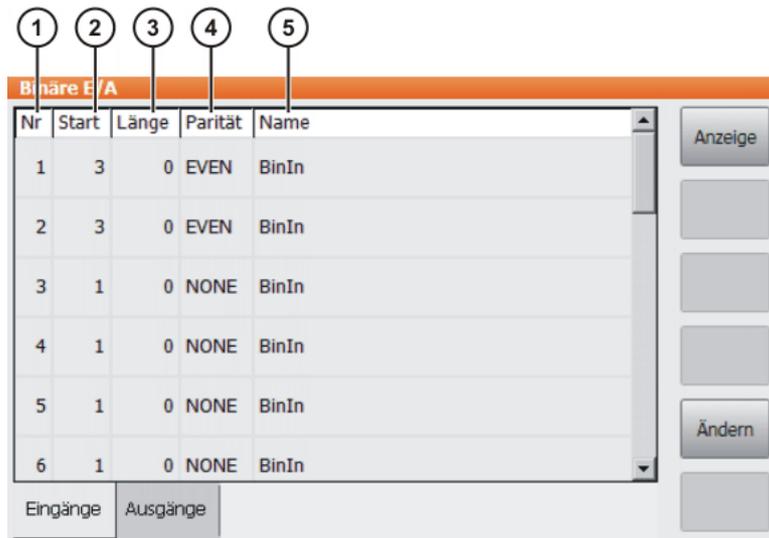


Fig. 6-2: Binary I/O configuration

Item	Description
1	Number of the input/output <ul style="list-style-type: none"> ■ 1 ... 20
2	Start bit of the input/output <ul style="list-style-type: none"> ■ 1 ... 1,024
3	Bit width of the input/output <ul style="list-style-type: none"> ■ 1 ... 16
4	Parity bit of the input/output <ul style="list-style-type: none"> ■ NONE: Parity bit is not activated (default). ■ EVEN: Parity bit is activated. <ul style="list-style-type: none"> ■ If the parity sum is even, the parity bit has the value 0. ■ If the parity sum is odd, the parity bit has the value 1. ■ ODD: Parity bit is activated. <ul style="list-style-type: none"> ■ If the parity sum is odd, the parity bit has the value 0. ■ If the parity sum is even, the parity bit has the value 1.
5	Name of the input/output

The following softkeys are available:

Softkey	Description
Monitor	Switches to the display of the binary I/O. (>>> 4.17.6 "Displaying binary inputs/outputs" Page 71)
Change	The selected cell is made available for editing. It is possible to toggle between the parities NONE, EVEN and ODD in the Parity column.

6.3 Configuring the variable overview

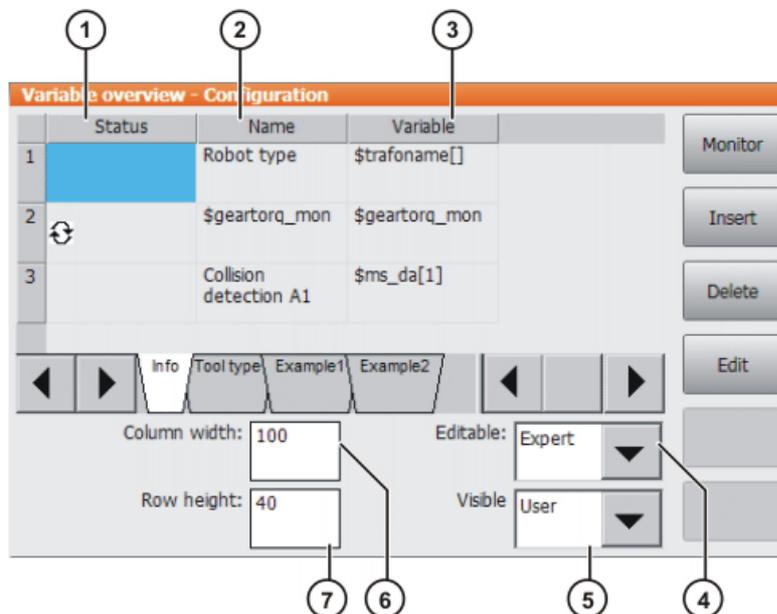
This is where the variables to be displayed in the variable overview and the number of groups are defined. A maximum of 10 groups is possible. A maximum of 25 variables per group is possible. System variables and user-defined variables can be displayed.

Precondition

- Expert user group

Procedure

- Select the menu sequence **Monitor > Variable > Overview > Configure**. The **Variable overview - Configuration** window is opened.
- Make the desired settings. To edit a cell, select it.
- Press **OK** to save the configuration and close the window.

Description**Fig. 6-3: 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
3	Path and name of the variable Note: For system variables, the name is sufficient. Other variables must be specified as follows: <i>/R1/Program name/ Variable name</i> Do not specify a folder between <i>/R1/</i> 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 softkeys are available:

Softkey	Description
Monitor	Switches to the variable overview. (>>> 4.17.12 "Displaying the variable overview and modifying variables" Page 77)
Insert	Shows additional softkeys: <ul style="list-style-type: none"> ■ Row above: Inserts a new row above the one currently selected. ■ Row below: Inserts a new row below the one currently selected. ■ Group before: Inserts a new group to the left of the one currently selected. ■ Group after: Inserts a new group to the right of the one currently selected.
Delete	Shows additional softkeys: <ul style="list-style-type: none"> ■ Row: The selected row is deleted. ■ Group: The current group is deleted.

6.4 Configuring workspaces

Workspaces can be configured for a robot. Workspaces serve to protect the system.

There are 2 types of workspace:

- The workspace is an exclusion zone.
The robot may only move outside the workspace.
- Only the workspace is a permitted zone.
The robot may not move outside the workspace.

A maximum of 8 Cartesian (=cubic) and 8 axis-specific workspaces can be configured at any one time. The workspaces can overlap.

(>>> 6.4.1 "Configuring Cartesian workspaces" Page 119)

(>>> 6.4.2 "Configuring axis-specific workspaces" Page 122)

Exactly what reactions occur when the robot violates a workspace depends on the configuration.

6.4.1 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

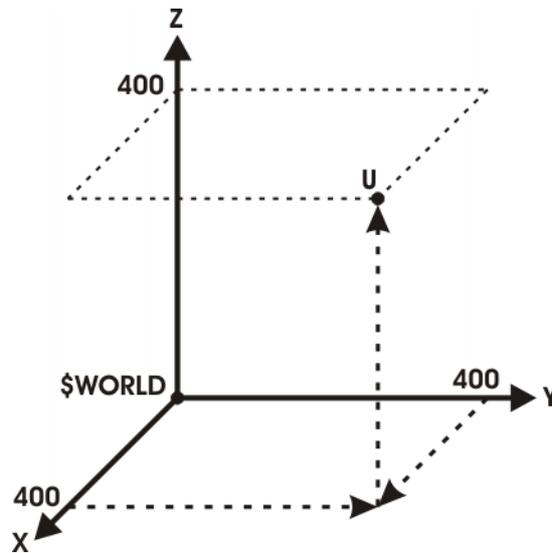


Fig. 6-4: Cartesian workspace, origin U

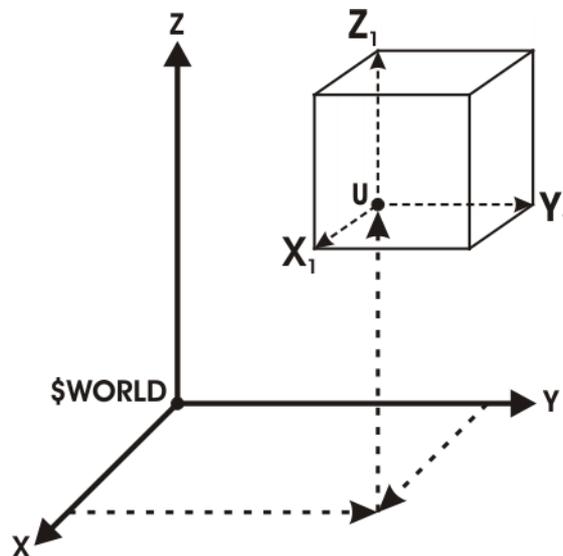


Fig. 6-5: Cartesian workspace, dimensions

Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

1. Select the menu sequence **Configure > Miscellaneous > WorkSpace-Monitor > 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.

Cartesian workspaces

① No. 1 Name: WORKSPACE 1 ②

③ Origin X: 500 A: 0.00 Y: 500 B: 0.00 Z: 1000 C: 0.00 Distance to origin X1: 100 X2: -100 Y1: 100 Y2: -1000.00 Z1: 100 Z2: -100 ④

Mode OFF ⑤

Fig. 6-6: Configuring a Cartesian workspace

Item	Description
1	Number of the workspace (max. 8)
2	Designation of the workspace
3	Origin and orientation of the workspace relative to the WORLD coordinate system
4	Dimensions of the workspace in mm
5	Mode (>>> 6.4.3 "Mode for workspaces" Page 124)

Signals

① Cartesian 1: 984 2: 985 3: 986 4: 987 5: FALSE 6: FALSE 7: FALSE 8: FALSE

② Axis spec. 1: FALSE 2: FALSE 3: FALSE 4: FALSE 5: FALSE 6: FALSE 7: FALSE 8: FALSE

Fig. 6-7: 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.

6.4.2 Configuring axis-specific workspaces

Description

Axis-specific workspaces can be used to restricted yet further the areas defined by the software limit switches in order to protect the robot, tool or work-piece.

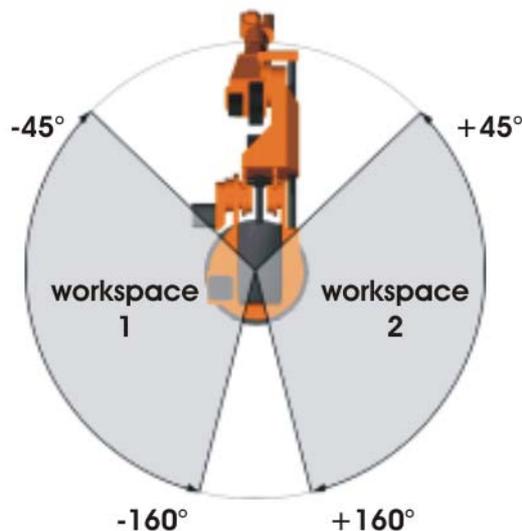


Fig. 6-8: Example of axis-specific workspaces for A1

Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

1. Select the menu sequence **Configure > Miscellaneous > WorkSpace-Monitor > 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. 6-9: 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 (>>> 6.4.3 "Mode for workspaces" Page 124)

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. 6-10: 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.

6.4.3 Mode for workspaces

Mode	Description
#OFF	Workspace monitoring is deactivated.
#INSIDE	<ul style="list-style-type: none"> ■ Cartesian workspace: The defined output is set if the TCP is located inside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located inside the workspace.
#OUTSIDE	<ul style="list-style-type: none"> ■ Cartesian workspace: The defined output is set if the TCP is located outside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located outside the workspace.
#INSIDE_STOP	<ul style="list-style-type: none"> ■ Cartesian workspace: The defined output is set if the TCP is located inside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located inside the workspace. <p>The robot is also stopped and a message is displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed.</p> <p>(>>> 4.16 "Bypassing workspace monitoring" Page 65)</p>
#OUTSIDE_STOP	<ul style="list-style-type: none"> ■ Cartesian workspace: The defined output is set if the TCP is located outside the workspace. ■ Axis-specific workspace: The defined output is set if the axis is located outside the workspace. <p>The robot is also stopped and a message is displayed. The robot cannot be moved again until the workspace monitoring is deactivated or bypassed.</p> <p>(>>> 4.16 "Bypassing workspace monitoring" Page 65)</p>

6.5 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.

6.5.1 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.
(>>> 6.5.3 "System variables for warm-up" Page 126)
 3. Close the file. Respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

6.5.2 Warm-up sequence

- Precondition**
- \$WARMUP_RED_VEL =TRUE
 - AUT EXT mode
 - 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
Monitoring active	The motor currents are monitored.
Monitoring inactive	The motor currents are not monitored.

Entry	Meaning
Controlling active	The velocity is reduced.
Controlling inactive	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
...
```

6.5.3 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).



If one of the values is outside the permissible range, the warm-up function is not active and the velocity is not reduced.

System variable	Description
\$WARMUP_RED_VEL	<ul style="list-style-type: none"> ■ TRUE: Warm-up functionality is activated. ■ FALSE: Warm-up functionality is deactivated (default). <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"> ■ > 0.0 min
\$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"> ■ > 0.0 min
\$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 = $(\\$CURR_LIM * \\$CURR_MAX) / 100$</p> <ul style="list-style-type: none"> ■ 0 ... 100%

System variable	Description
\$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"> 0 ... 100%
\$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"> > 0.0%/s

6.6 Defining calibration tolerances



Only modify the default values in exceptional cases. Otherwise, increased error messages and inaccuracy may result.

Precondition

- Expert user group

Procedure

- Select the menu sequence **Setup > Measure > Tolerances**.

Description

The screenshot shows a dialog box titled "Measurement - Tolerances". It contains four sections, each with a circled number and a text input field:

- 1. Tool measurement: Minimum Distance (TOOL) [mm]: 8.00
- 2. Base measurement: Minimum Distance (BASE) [mm]: 50.00
- 3. Base measurement: Minimum Angle [°]: 2.50
- 4. Maximum error [mm]: 5.00

Fig. 6-11: Default error tolerances

Item	Description
1	<p>The minimum distance for tool calibration.</p> <ul style="list-style-type: none"> 0 ... 200 mm
2	<p>The minimum distance for base calibration.</p> <ul style="list-style-type: none"> 0 ... 200 mm
3	<p>The minimum angle between the straight lines through the 3 calibration points in base calibration.</p> <ul style="list-style-type: none"> 0 ... 360°
4	<p>Maximum error in calculation.</p> <ul style="list-style-type: none"> 0 ... 200 mm

The following softkeys are available:

Softkeys	Description
Default	Restores the default settings. The data must then be saved by pressing OK .

6.7 Submit interpreter

6.7.1 Function of the Submit interpreter

Function

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.

6.7.2 Manually stopping or deselecting the Submit interpreter

Precondition

- User group "Expert".
- Operating mode T1 or T2.

Procedure

- Select the menu sequence **Configure > SUBMIT Interpreter > Stop or Cancel**.

Alternative procedure

- In the status bar, touch the **Submit interpreter** status indicator. A window opens.
Select **Stop** or **Deselect**.

Description

Command	Description
Stop	The Submit interpreter is stopped. When it is restarted, the SUB program is resumed at the point at which it was stopped.
Deselect	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.

6.7.3 Manually starting the Submit interpreter

- Precondition**
- User group "Expert"
 - Operating mode T1 or T2
 - Submit interpreter has been stopped or deselected.
- Procedure**
- Select the menu sequence **Configure > SUBMIT Interpreter > Start / select**.
- Alternative procedure**
- In the status bar, touch the **Submit interpreter** status indicator. A window opens.
Select **Start / select**.

Description If the Submit interpreter is deselected, the command **Start/Select** selects the SUB program defined in the file \$custom.dat.

If the Submit interpreter has been stopped, the command **Start/Select** resumes the selected SUB program at the point at which it was stopped.

Once the Submit interpreter has been started, the corresponding icon in the status bar is green.

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.

6.8 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. (>>> 6.8.1 "Configuring CELL.SRC" Page 129)
2	Configuration of the inputs/outputs of the Automatic External interface. (>>> 6.8.2 "Configuring Automatic External inputs/outputs" Page 130)

6.8.1 Configuring CELL.SRC

In Automatic External mode, Folgen are called using the program CELL.SRC.

- Procedure**
1. Select the menu sequence **Setup > 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 **\$STRAFONAME()="#"**

Description

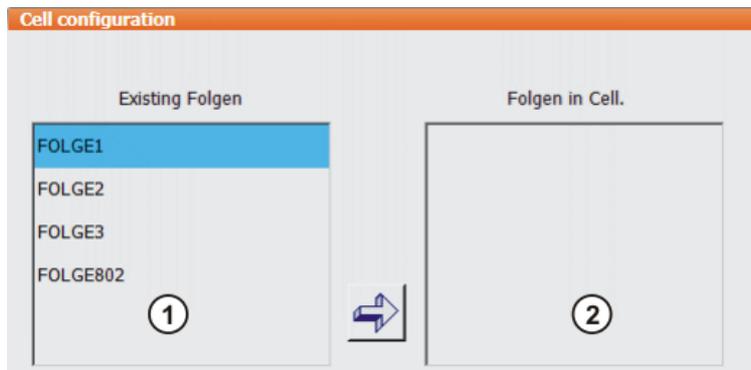


Fig. 6-12: Cell configuration window

Item	Description
1	All Folgen available for configuration of CELL.SRC are displayed in the Existing Folgen window.
2	All Folgen located in CELL.SRC are displayed in the Folgen in Cell window.

The following softkeys are available:

Softkey	Description
Add	Inserts a selected Folge into CELL.SRC. Only available in the Existing Folgen window.
Delete	Deletes a selected Folge from CELL.SRC. Only available in the Folgen in Cell window.

6.8.2 Configuring Automatic External inputs/outputs

Procedure

1. Select the menu sequence **Configure > I/O > 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.



Descriptions of the individual inputs and outputs:

- (>>> 6.8.3 "Automatic External inputs" Page 131)
- (>>> 6.8.4 "Automatic External outputs" Page 132)

Description

Automatik Extern-Konfiguration: Eingänge				
	Bezeichnung	Typ	Name	Wert
1	Antriebe EIN in EXT von Betriebsmittelsteuerung	E/A	ANTEIN	140
2	Folgestart	E/A	SRB	137
3	Format für Folgenauswahl (0=Int; 1=1 aus n)	Var	P_TYPE	1
4	Erster Eingang für Folgennummern	E/A	P_FBIT	129
5	Länge für Folgennummern	Var	P_LEN	8

Fig. 6-13: Configuring Automatic External inputs

Automatic External - Configuration: Outputs				
	Term	Type	Name	Value
1	First output for Folgenspiegelung	E/A	R_FBIT	129
2	Last Folgennummer in Automatic mode	Var	P_OLD	0
3	Robot control ready	E/A	BEREIT	137
4	Programming mode	E/A	RK23	138
5	Single Step	E/A	RK8	139
6	Automatic External	E/A	RK9	140
7	Actuator release Start	E/A	RK100	141
8	Wait for process input		WBP07	142

Fig. 6-14: Configuring Automatic External outputs

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

6.8.3 Automatic External inputs

ANTEIN

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.

SRB

If there is a high-level pulse of at least 20 ms duration at this input, the higher-level controller starts the selected Folge.



This signal does not need to be reset by the higher-level controller. It can be present at the input for several consecutive Folgen.

**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.

P_TYPE

Type: Variable

This variable defines the format in which the Folge number sent by the higher-level controller is read.

Value	Description	Example
0	Read as binary number. The Folge number is transmitted by the higher-level controller as a binary coded integer. The parity of the value is not checked.	0 0 1 0 0 1 1 1 => PGNO = 39
1	Read as "1 of n". The Folge number is transmitted by the higher-level controller or the periphery as a "1 of n" coded value.	0 0 0 0 0 0 1 => PGNO = 1 0 0 0 0 1 0 0 0 => PGNO = 4

P_FBIT

Input representing the first bit of the Folge number. Range of values: 1 ... 4096.

Example: P_FBIT = 5 => the external Folge number begins with input 5.

P_LEN

Type: Variable

This variable determines the number of bits in the Folge number sent by the higher-level controller. Range of values: 1 ... 16.

Example: P_LEN = 4 => the external Folge number is 4 bits long.

6.8.4 Automatic External outputs

R_FBIT

Output representing the first bit of the Folge number.

The size of the output area depends on the number of bits defining the Folge number (P_LEN).

R_FBIT is set to FALSE if the higher-level controller sends a Folge number that has not been assigned or that does not correspond to the format P_TYPE.

(>>> 6.8.3 "Automatic External inputs" Page 131)

P_OLD

Type: Variable

This variable indicates the last Folge number in Automatic External mode. Range of values: 1 ... 999

BEREIT

By setting this output, the robot controller communicates to the higher-level controller that it is ready.

This output is reset in the event of an error in the robot controller that can be eliminated by the user via the KCP. A corresponding error message is displayed on the KCP, e.g. in the case of an Interbus master error or an impermissible operating mode.

RK23

This output is TRUE if operating mode T1 or T2 is set and no program is running.

RK8	This output is TRUE if operating mode T1 or T2 is set.
RK9	This output is TRUE if Automatic External mode is set.
RK100	This output is TRUE if there is no EMERGENCY STOP situation and the safety gates are closed.



To ensure a path-maintaining stop in the case of an EMERGENCY STOP, this output is not set to FALSE until the robot motion has stopped.

WPROZ	By setting this output, the robot controller communicates to the higher-level controller that it is waiting for a process input.
WSLAV	By setting this output, the robot controller communicates to the higher-level controller that it is waiting for a slave input.
LPKT	By setting this output, the robot controller communicates to the higher-level controller that the robot has successfully reached the last programmed point in a Folge.
PF0	By setting this output, the robot controller communicates to the higher-level controller that the robot is situated at the zero point PF0 of the Folge. The Folge is started from this home position.
SAK	This output remains set to TRUE as long as the robot stays on its programmed path. If it leaves the path, the output is set to FALSE.
IntEstop	This output is set to FALSE if the EMERGENCY STOP button on the KCP is pressed.

6.8.5 Signal diagrams

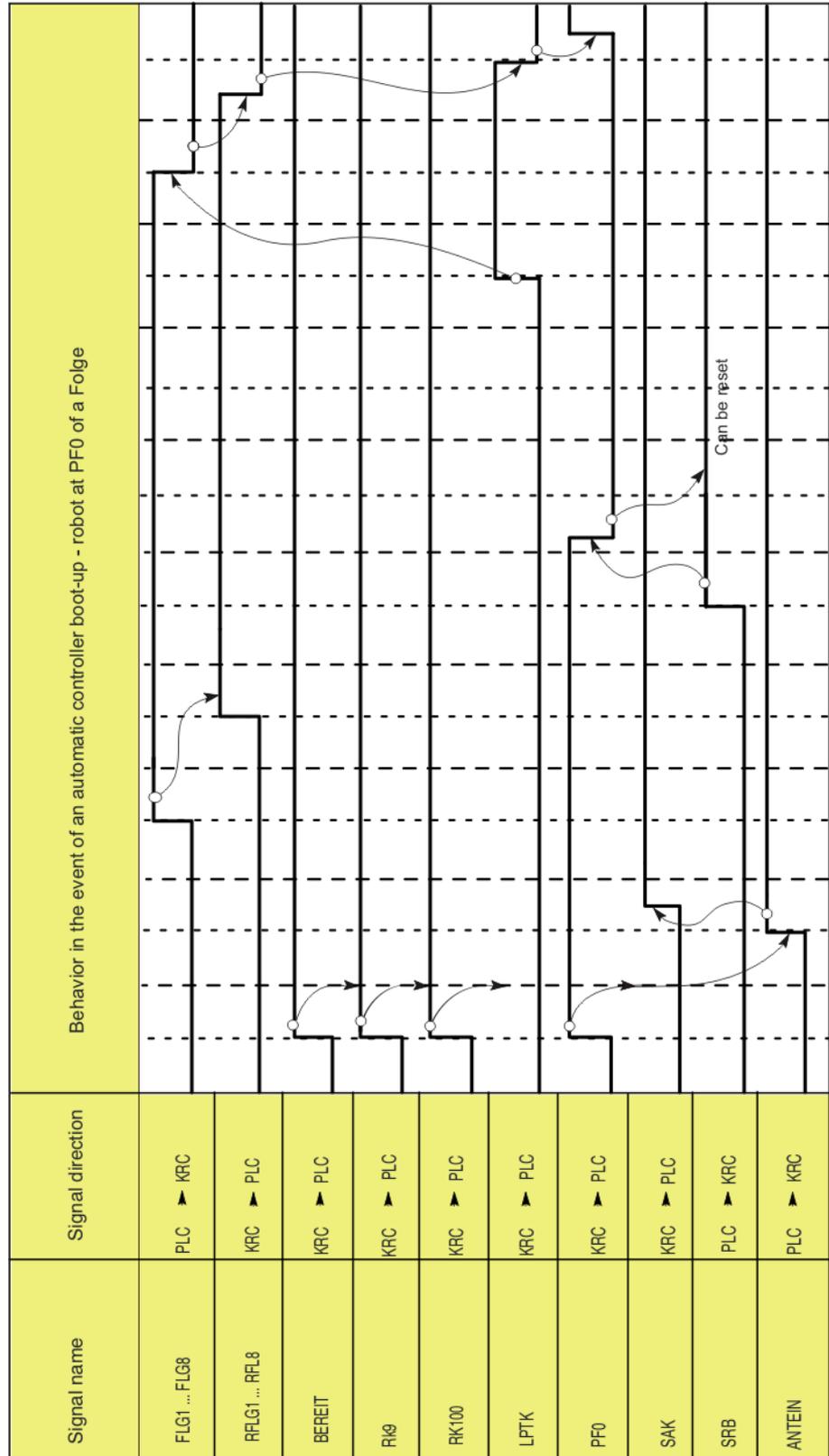


Fig. 6-15: Automatic controller boot-up – robot at PF0

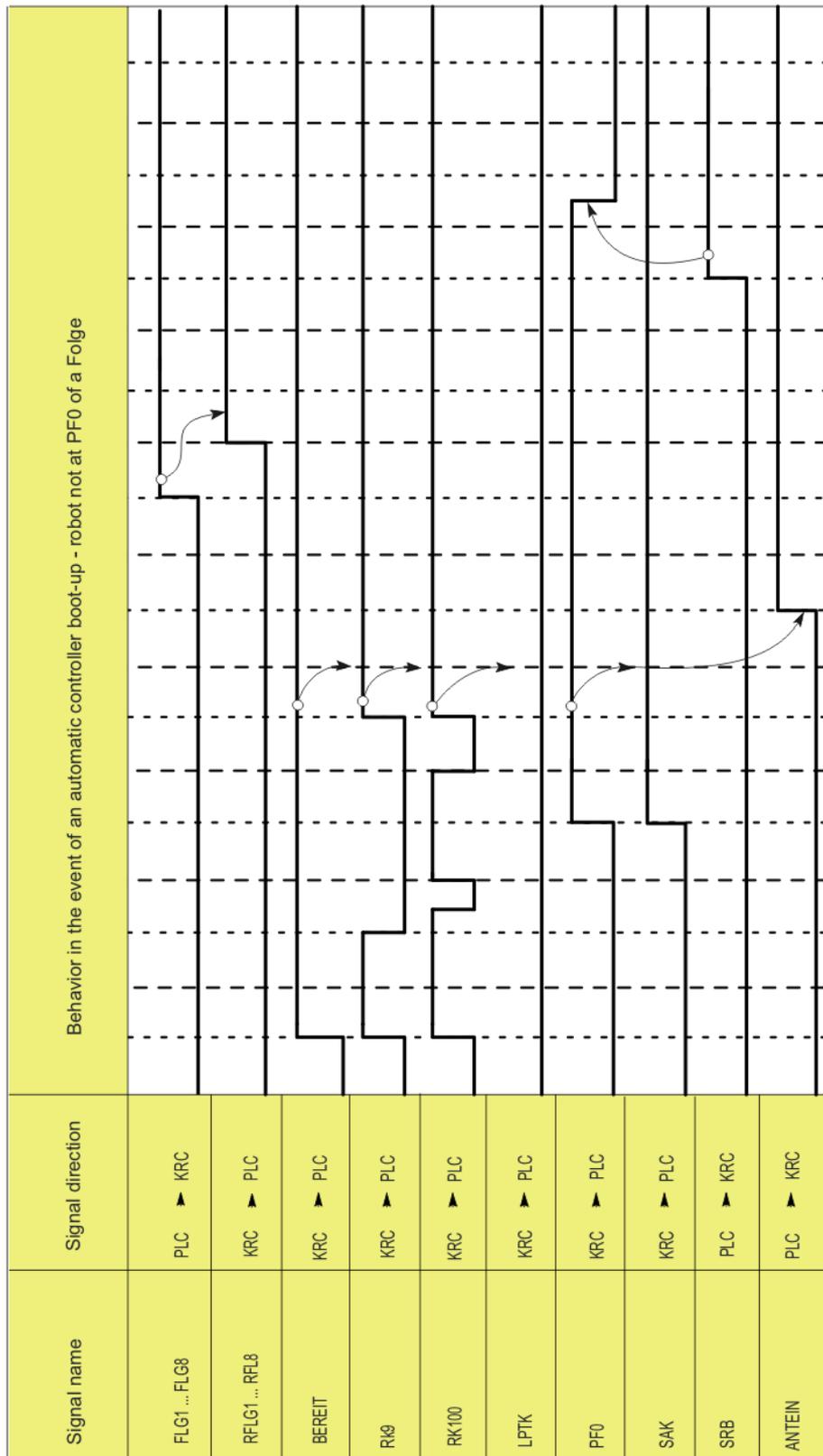


Fig. 6-16: Automatic controller boot-up – robot not at PF0

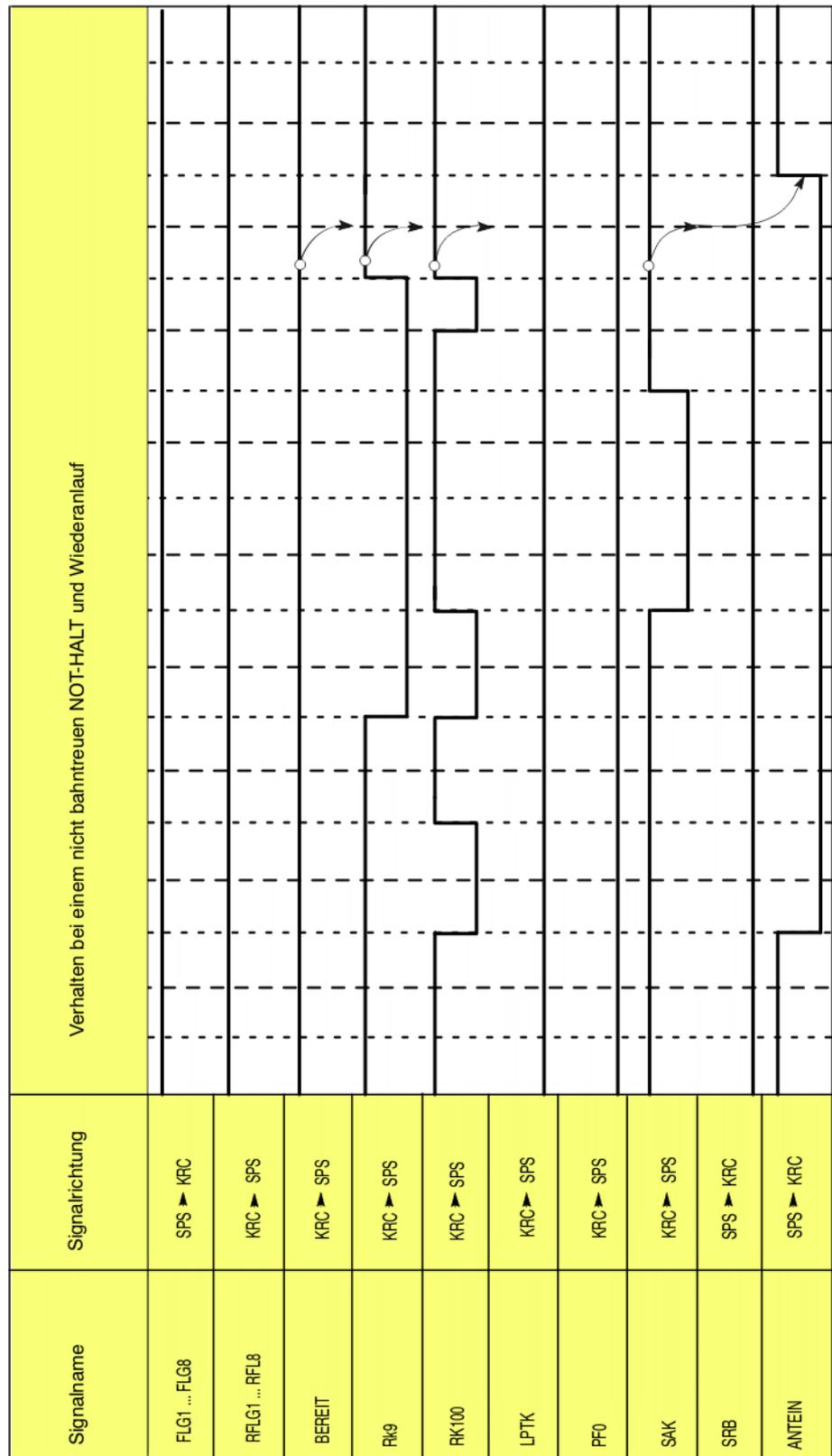


Fig. 6-17: Restart after non-path-maintaining EMERGENCY STOP

6.9 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.

6.9.1 Configuring a data comparison

Precondition ■ Expert user group

- Procedure**
1. Select the menu sequence **Configure > Miscellaneous > Event planner**.
 2. Open the tree structure in the left-hand part of the window.
 3. Select the desired action:
 - **T1 and T2 Consistency**: Compare data in operating modes T1 and T2 at regular intervals.
 - **AUT and EXT Consistency**: Compare data in operating mode Automatic External at regular intervals.
 - **Logic Consistency**: Compare data after a change of operating mode or after online optimizing.
Online optimizing is the modification of the parameters of a program during operation.
 4. Make the desired settings in the right-hand part of the window.
(>>> 6.9.2 "Configuring T1 and T2 Consistency, AUT and EXT Consistency" Page 137)
(>>> 6.9.3 "Configuring Logic Consistency" Page 138)
 5. Press **Save**.

6.9.2 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. 6-18: Configuring T1 and T2 Consistency

Item	Description
1	<ul style="list-style-type: none"> ■ Check box active: data comparison is activated. ■ Check box not active: data comparison is deactivated.
2	Enter date and time in the format indicated.
3	<ul style="list-style-type: none"> ■ Check box active: interval is activated. ■ Check box not active: interval is deactivated.



Caution!

If the intervals selected are too small, this can result in damage to the hard drive. An interval of several minutes is recommended.

6.9.3 Configuring Logic Consistency

Description

The following settings are possible here:

Name Logic Consistency

Description Ensure data consistency between HDD and Kernel System when operation mode changes.

Last run 11.08.2010 12:33:30

1 Enabled

Ensure consistency

2 When switch to T1.

3 When switch to T2.

4 When switch to Automatic.

5 When switch to Automatic Extern.

6 After online optimization.

Fig. 6-19: Configuring Logic Consistency

Item	Description
1	<ul style="list-style-type: none"> ■ Check box active: data comparison is activated. ■ Check box not active: data comparison is deactivated.
2	<ul style="list-style-type: none"> ■ Check box active: data are compared when operating mode is switched to T1. ■ Check box not active: data comparison is deactivated.
3	<ul style="list-style-type: none"> ■ Check box active: data are compared when operating mode is switched to T2. ■ Check box not active: data comparison is deactivated.
4	No function in VSS
5	<ul style="list-style-type: none"> ■ Check box active: data are compared when operating mode is switched to Automatic External. ■ Check box not active: data comparison is deactivated.
6	<ul style="list-style-type: none"> ■ Check box active: data are compared after online optimizing. ■ Check box not active: data comparison is deactivated.

7 Program management

7.1 Navigator file manager

Overview

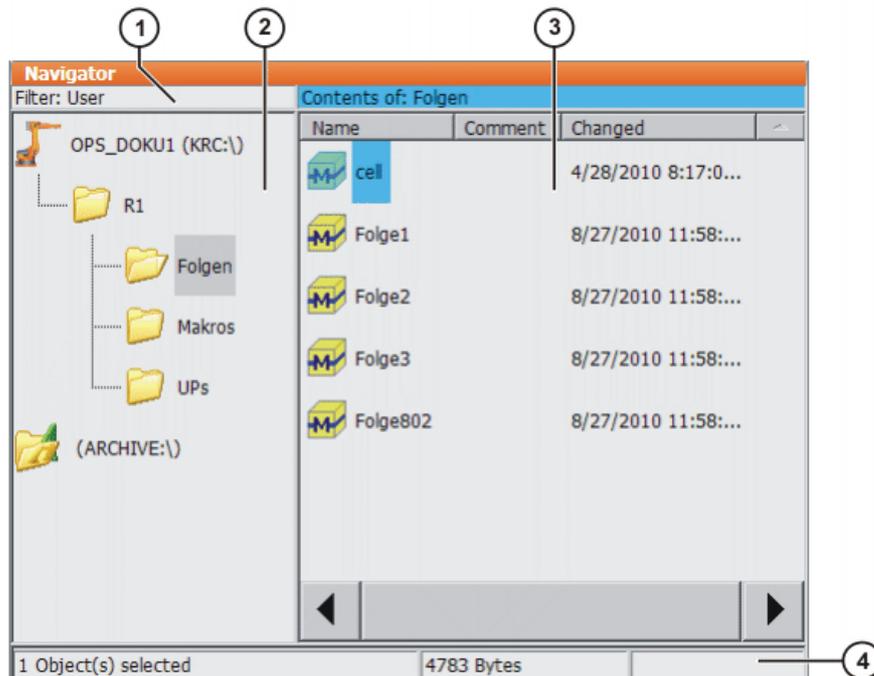


Fig. 7-1: Navigator

- | | | | |
|---|---------------------|---|------------|
| 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.
- 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

Column	Description
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
Modified	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 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 softkey **Program > Filter**.
2. Select the desired filter in the left-hand section of the Navigator.
3. Confirm with **OK**.

7.1.2 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 **Folgen**. (Not all folders allow the creation of programs within them.)
2. Press the **New** softkey.
3. Only in the user group "Expert":
The **Template selection** window is opened. Select the desired template and confirm with **OK**.
4. Enter a number for the Folge or 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.

7.1.3 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 the softkey **Program > Rename**.
4. Overwrite the file name with the new name and confirm with **OK**.

7.2 Selecting and deselecting a program

Description Before a program can be started or edited, it must first be selected. Following program execution or editing, it must be deselected again.



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.

Procedure

1. Select the program in the Navigator.
If the program is displayed as a SRC file and a DAT file, the SRC file or the DAT file can be selected.
2. Press **Select**.
3. Execute or edit the program.
4. Select the softkey **Program > Cancel program**.

Alternative procedure: ■ In the status bar, touch the **Robot interpreter** status indicator. A window opens.
Deselect Select **Cancel program**.

7.3 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: press **Program > Navigator**.
- Toggling from the Navigator to the program: press the **PROGRAM** softkey.

Program is opened:

- Toggling from the program to the Navigator: press **Program > Navigator**.
- Toggling from the Navigator to the program: press the **EDITOR** softkey.



Programs that are running or have been interrupted must first be stopped before the menu sequences and softkeys referred to above are available.

7.4 Structure of a program (Folge)

```

Editor
1 PTP UB=100% UE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s]

   Warte auf Folgenstart

2 PTP UB=100% UE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P

   1: M1 = (A1 & A2) & (A3 & A4)

   2: M2 = (A5 & A6) & (A7 & A8)

   3: M3 = (M1 & M2)

   4: WARTE BIS A3

3 PTP UB=100% UE=100% ACC=100% Wzg=1 SPSTrig=0[1/100s]

4 LIN UB=1500[mm/s] UE=100% ACC=100% Wzg=1
  ↳ SPSTrig=0[1/100s]

5 PTP UB=100% UE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P

6 LIN UB=1500[mm/s] UE=100% ACC=100% Wzg=1
  ↳ SPSTrig=0[1/100s]

7 PTP UB=100% UE=100% ACC=100% Wzg=1 SPSTrig=0[1/100s]

   1:

```

Fig. 7-2: Example

Line	Description
1.1	PTP motion to the zero point PF0 of the Folge, i.e. to the point from which the Folge is started. (>>> 9.1.1 "Programming a PTP motion" Page 185)
1.2	Wait for Folge start The robot controller is waiting for the "Folge start" signal (SRB) from the higher-level controller. Once the signal has been set, the Folge is started.
2	Open Point PLC with PLC instructions. (>>> 9.4 "Programming PLC instructions" Page 201) Note: The Point PLC is not automatically displayed in the selected program. (>>> 7.5.1 "Displaying PLC instructions" Page 145)
4	LIN motion (>>> 9.1.3 "Programming a LIN motion" Page 186)
7	Open Point PLC without PLC instructions.

Motions that are linked to PLC instructions are indicated in the program by means of the following abbreviations:

Abbrevia tion	Description
F	The Point PLC contains a FB ONL motion condition. (>>> 9.4.19 "Inline form "FB ONL"" Page 210)
P	The Point PLC contains PLC instructions with line numbering.
U	A subprogram is called in the Point PLC. (>>> 9.4.44 "Inline form "UP"" Page 233)

All PLC instructions with line numbering are executed at the end point of the motion. They can be brought forward in time so that they are executed before the end point is reached. For this purpose, the PLC trigger in the inline form of the motion must be programmed accordingly.

PLC instructions without line numbers can be positioned before or after numbered PLC instructions:

- If they are positioned before them, they are executed on the way to the end point and asynchronously in relation to the PLC trigger, i.e. the FB ONL motion condition, the WAIT ONL wait function or the interlock instruction.
- If they are positioned after them, they are executed at the end point as soon as all numbered PLC instructions have been executed, e.g. a subprogram call.

7.5 Displaying/hiding program sections

7.5.1 Displaying PLC instructions

Description PLC instructions and other program sections are hidden in the Point PLC. In this way, the Point PLCs make programs more transparent. The hidden program sections are processed during program execution in exactly the same way as normal program sections.

If a program is deselected, all open Point PLCs are automatically closed.

Precondition ■ Program is selected or open.

Procedure

1. Position the cursor in the line containing the Point PLC.
2. Press **PLC Open**. The Point PLC is opened.
3. To close the Point PLC, press **PLC Close**.

Alternatively, select the softkey **Program > SPS > Open all** or **Close all** to open or close all the Point PLCs in a program at once.

7.5.2 Activating detail view (ASCII mode)

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

Procedure ■ Select the menu sequence **Configure > Misc. > Editor > ASCII Mode**.
Check mark activated in menu: ASCII mode is activated.
Check mark not activated in menu: ASCII mode is deactivated.

7.5.3 Activating/deactivating the line break function

Description 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. 7-3: Line break

The line break function is activated and deactivated separately for opened and selected programs.

- Precondition**
- User group "Expert"
 - Program is selected or open.
- Procedure**
- Select the menu sequence **Configure > Miscellaneous > Editor > Line-break**.
- Check mark activated in menu: line break function is activated.
 Check mark not activated in menu: line break function is deactivated.

7.6 Starting a program

7.6.1 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.
 (>>> 7.6.2 "Program run modes" Page 146)
- The window closes and the selected program run mode is applied.

7.6.2 Program run modes

Program run mode	Description
Go #GO	The program is executed through to the end without stopping.
Motion Step #MSTEP	The program is executed with a stop after each motion block. The Start key must be pressed again for each motion block.
Single Step #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. Single Step 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.

7.6.3 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: 1
- 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. set output.

7.6.4 Setting the program override (POV)

Description 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.

Procedure

1. Touch the **POV** status indicator. The **Program override** window is opened.
2. Set the desired value. It can be set using either the plus/minus keys or by means of the slide controller.
 - Plus/minus keys: The value can be set to 100%, 75%, 50%, 30%, 10%, 3%, 1%
 - Slide controller: The override can be adjusted in 1% steps.
3. Touch the **POV** status indicator again. (Or touch the area outside the window.)

The window closes and the selected override value is applied.



The **Jogging Options** window can be opened via **Options** in the **Program override** window.

Alternative procedure

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%.

7.6.5 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.

7.6.6 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.
	Red	The selected and started program has been stopped.
	Black	The block pointer is situated at the end of the selected program.

7.6.7 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":



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.

7.6.8 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.

7.6.9 Carrying out a block selection

Description A program can be started at any point by means of a block selection. The **MakroSAW** macro is automatically executed every time block selection is carried out. The outputs that are reset in the event of a block selection, e.g. the interlock outputs to other robots, are defined in this macro.

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 **Line Select**.
The block pointer at the selected motion block indicates that the MakroSAW macro is being executed.

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. It is not necessary to carry out a BCO run again.

7.6.10 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	<ul style="list-style-type: none"> Program is selected.
Procedure	<ul style="list-style-type: none"> Select the softkey Program > Reset program.
Alternative procedure	<ul style="list-style-type: none"> In the status bar, touch the Robot interpreter status indicator. A window opens. Select Reset program.

7.6.11 Starting Automatic External mode

Precondition	<ul style="list-style-type: none"> Operating mode T1 or T2 Inputs/outputs for Automatic External and the program CELL.SRC are configured.
Procedure	<ol style="list-style-type: none"> 1. Select the program CELL.SRC in the directory C:\KRC\R1\Folgen. 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 EXT mode.
5. Start the program from a higher-level controller (PLC).

**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.



To stop a program that has been started in Automatic mode, press the STOP key.

7.6.12 Dry run

Description

The dry run is carried out if program execution is interrupted in Automatic mode and the robot leaves the programmed path.



If the dry run is caused in a subprogram, the dry run is only executed within the subprogram. The program from which the subprogram was called is not involved in the dry run.

The following cases are possible:

- Block selection
- Program deselection
- Backward motion
- The robot leaves the path due to a non-path-maintaining stop, a defect in the holding brakes or jogging by the user.

In these cases, the controller sets the digital, binary and analog outputs, cyclical flags and flags to FALSE. The robot then moves to the next programmed position on the path. Once this position has been reached, the dry run is executed. The previously reset signals are set as they would be at this position following program execution. Program execution can be resumed.



Caution!

If program execution is resumed following a dry run, the controller may behave differently from how it was programmed.

The following differences between dry run and program execution must be borne in mind:

- During the dry run, only the current state of an input can be accessed. This can have a different value during program execution.
- System outputs and outputs whose setting is dependent on inputs are not taken into consideration.
- Program loops, jump and wait statements and subprogram calls are not taken into consideration.
- Counting operations and timers are not taken into consideration.

Block selection

```

1 PTP VB=10% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P
  Warten auf Folgenstart
2 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P
3 LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s]
4 LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s] P
  1: A1 = EIN
  2: A2 = EIN
5  LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s] P
  1: A3 = EIN
  2: A4 = EIN
6 LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s]
7 LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s] P
  1: A1 = AUS
  2: A2 = AUS
  3: A3 = AUS
  4: A4 = AUS
8 LIN VB=250[mm/s] VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s]
9 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P

```

Fig. 7-4: Example of a block selection

Program execution is interrupted in line 6, for example, by the block selection to line 5. In line 6, outputs A1 to A4 are TRUE. These outputs are now set to FALSE. When the BCO run is performed, the selected motion block is approached slowly by the direct route. The robot stops once it has reached line 5, the dry run is carried out and outputs A1 and A2 are reset to TRUE. Program execution can be resumed.

Program deselection

If the program is deselected, all outputs, cyclical flags and flags are set to FALSE.

Backward motion

If the START minus key is pressed, all outputs, cyclical flags and flags are set to FALSE. The programmed path is executed backwards. Logic statements and subprogram calls are not taken into consideration. If the direction is changed again by pressing the START plus key, the dry run is performed.

Leaving the path

When the path is left, only the digital and analog outputs are reset. The state of the binary outputs, cyclical flags and flags remains unchanged. Using the Start key, the robot can be repositioned to the point at which it left the path. Following the dry run, the digital and analog outputs are reset to the state they had immediately before the path was left.

7.7 Editing 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.

7.7.1 Inserting a comment**Precondition**

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line **after** which the comment is to be inserted.
2. Select the softkey **Commands > Comment**.
3. An inline form is opened. Enter the desired text.
4. Press **Cmd OK**.

7.7.2 Deleting program lines

- Precondition**
- Program is selected or open.
 - Operating mode T1 or T2

- 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 softkey **Edit > Delete**.
 3. Confirm the request for confirmation with **Yes**.



Lines cannot be restored once they have been deleted!



If a program line containing a motion instruction is deleted, the point name and coordinates remain saved in the DAT file. The point can be used in other motion instructions and does not need to be taught again.

7.7.3 Additional editing functions

The following additional program editing functions can be called using the **Program** softkey:

Copy

Precondition:

- Program is selected or open.
- User group "Expert"
- Operating mode T1 or T2

Paste

Precondition:

- Program is selected or open.
- User group "Expert"
- Operating mode T1 or T2

Cut

Precondition:

- Program is selected or open.
- User group "Expert"
- Operating mode T1 or T2

Find

Precondition:

- Program is selected or open.

Replace

Precondition:

- Program has been opened.
- User group "Expert"

7.8 Printing a program

- Procedure**
1. Select the program in the Navigator. Multiple program selection is also possible.
 2. Select the softkey **Program > Print**.

7.9 Archiving

7.9.1 Archiving

Description Archiving is always carried out to D:\, irrespective of whether a USB medium is connected or not. The file INTERN.ZIP is generated.

For the user group "Expert", the archived files are displayed in the directory ARCHIVE:\Intern in the Navigator.

Precondition For archiving to a **USB medium**:

- A writable USB medium is connected.

- Procedure**
1. Select the menu sequence **File > Archive > USB** and the desired menu item.
 2. Confirm the request for confirmation with **Yes**.
Once the archiving is completed, this is indicated in the message window. The file ARCHIVE.ZIP is generated by default on the USB storage medium.



Caution!

In the case of archiving to a USB medium: the medium must not be removed until the LED on the USB medium is no longer lit. Otherwise, the medium could be damaged.

7.9.2 Menu item "Archive"

Description The following menu items are available for archiving.

Menu item	Description
All	The data that are required to restore an existing system are archived.
Applications > Folgen	All user-defined Folgen and their corresponding system files are archived.
Applications > Ups	All user-defined subprograms and their corresponding system files are archived.
Applications > Makros	All macros and their corresponding system files are archived.
Applications > VW-User	All user-defined transfer parameters to the VW_User module are archived.
System > Machine Data	The machine data are archived.
System > Configuration	The configuration data are archived.
System > I/O Longtexts	The long text names of the inputs/outputs are archived.
System > Drivers	The I/O drivers are archived.

Menu item	Description
System > Log Data	The log files are archived.
System > Process Parameter	The process parameters are archived.

- If archiving is carried out using the menu item **All**, an existing archive will be overwritten.
- If archiving is carried out using a menu item other than **All** 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.

7.9.3 Restoring data



Caution!

Only VSS 8.1 archives may be loaded into VSS 8.1. If other archives are loaded, the following may occur:

- Error messages
- Robot controller is not operable.
- Personal injury and damage to property.

If the archive files are not the same version as the system files, an error message is generated. Similarly, if the version of the archived technology packages does not match the installed version, an error message is generated.

7.9.3.1 Restoring data via the menu

Description With the exception of **Log Data**, the menu items available for restoring data are the same as those available for archiving.

Precondition ■ A USB medium with the archive is connected.

Procedure

1. Select the menu sequence **File > Restore > External USB stick**.
2. Confirm the request for confirmation with **Yes**.
A message indicates completion of the restoration process.



Caution!

In the case of restoring data from a USB medium: the medium must not be removed until the LED on the USB medium is no longer lit. Otherwise, the medium could be damaged.

3. Reboot the robot controller.

7.9.3.2 Restoring data from a standard archive

Description With the exception of **Log Data** and **Process Parameter**, the menu items available for restoring data from a standard archive are the same as those available for archiving.



Standard archives can be used by different robots and are always saved as ARCHIVE.ZIP.

Precondition ■ The robot name is configured in the robot controller as the name of the archive.

- A USB medium with the archive is connected.

Procedure

1. Select the menu sequence **File > Restore > S_Archive** and the desired menu item.
2. Confirm the request for confirmation with **Yes**.
A message indicates completion of the restoration process.

**Caution!**

In the case of restoring data from a USB medium: the medium must not be removed until the LED on the USB medium is no longer lit. Otherwise, the medium could be damaged.

3. Reboot the robot controller.

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 157)
- Linear motions (LIN)
(>>> 8.3 "Motion type LIN" Page 158)
- Circular motions (CIR)
(>>> 8.4 "Motion type CIR" Page 158)
- Spline motions
(>>> 8.7 "Motion type "Spline"" Page 163)

LIN, CIR 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.

In VSS, application-specific motions based on these standard motions can be programmed.

(>>> 9.2 "Programming application-specific motions" Page 189)

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.

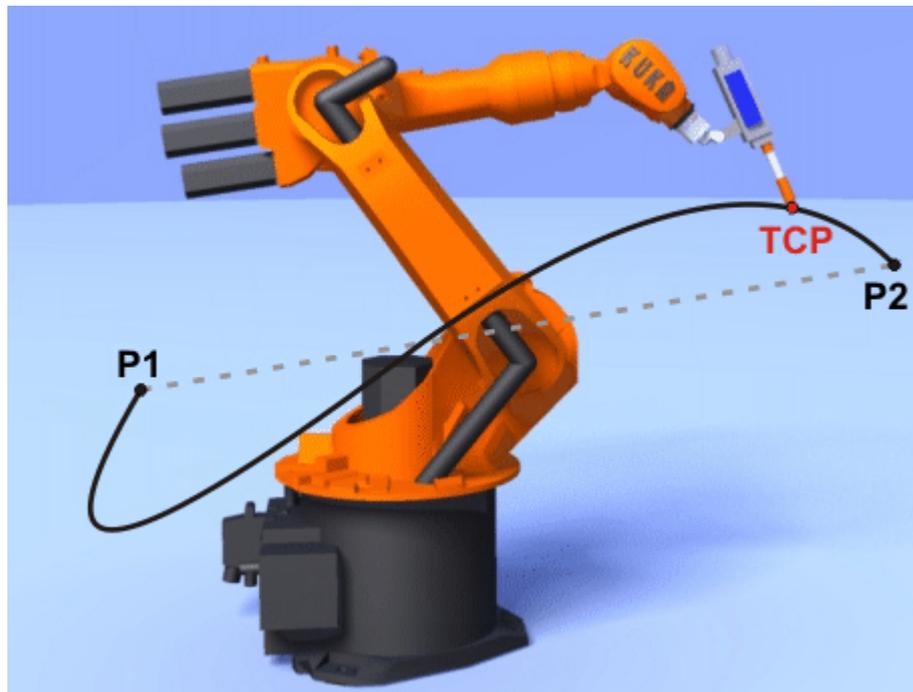


Fig. 8-1: PTP motion

8.3 Motion type LIN

The robot guides the TCP at a defined velocity along a straight path to the end point.

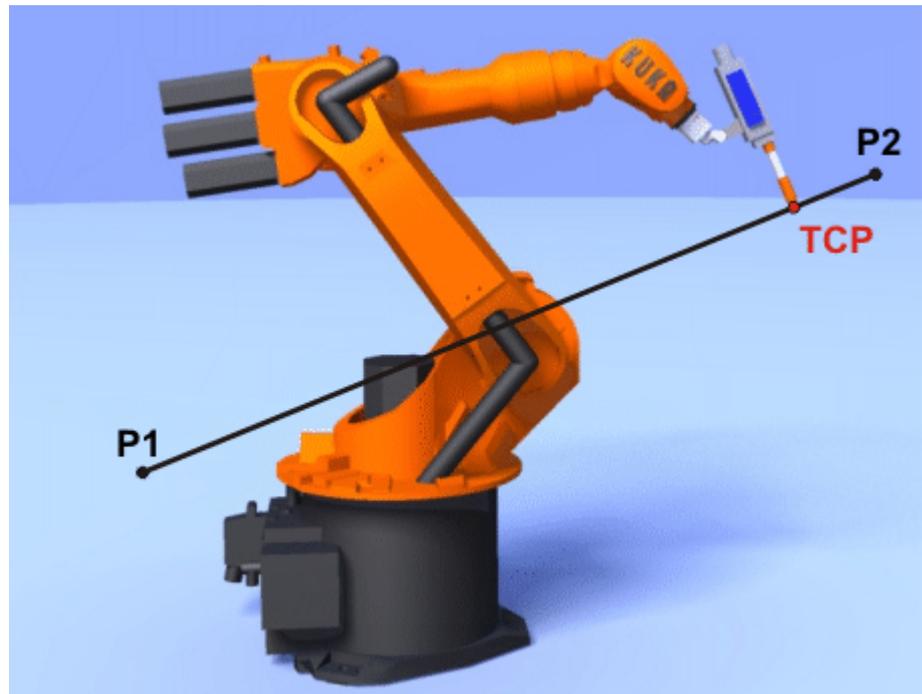


Fig. 8-2: LIN motion

8.4 Motion type CIR

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.

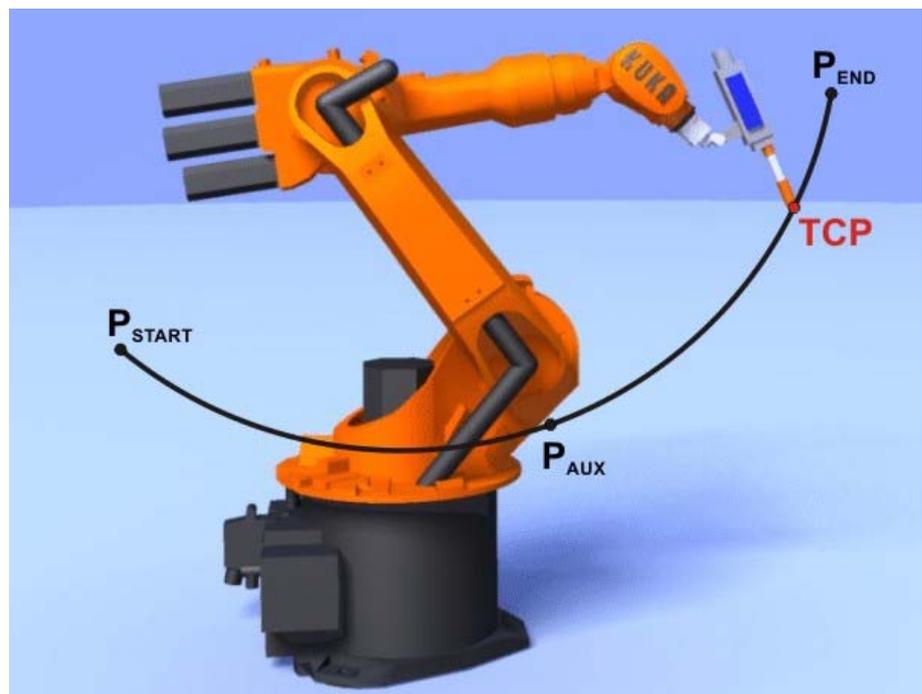


Fig. 8-3: CIR motion

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.

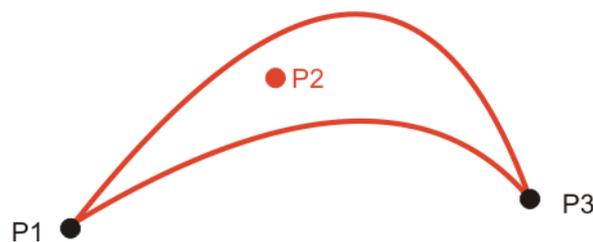


Fig. 8-4: PTP motion, P2 is approximated

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.

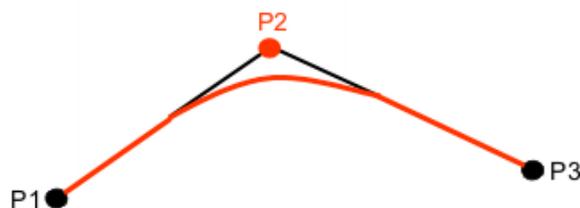


Fig. 8-5: LIN motion, P2 is approximated

CIR 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.

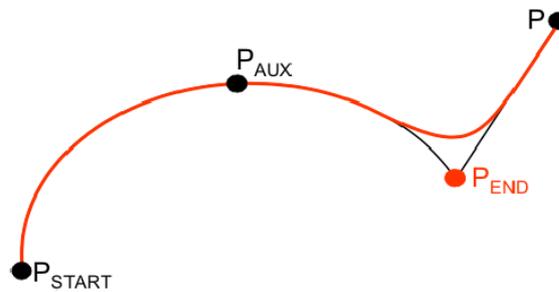


Fig. 8-6: CIR motion, P_{END} is approximated

8.6 Orientation control LIN, CIR

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.



In the case of programming with inline forms, the orientation control cannot be changed.

The orientation control for LIN and CIR motions is defined via the system variable \$ORI_TYPE.

LIN motion

\$ORI_TYPE =	Description
#CONSTANT	<p>Constant orientation control</p> <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>
#VAR	<p>Standard (default)</p> <p>The orientation of the TCP changes continuously during the motion.</p> <p>Note: If, with Standard, the robot passes through a wrist axis singularity, use Wrist PTP instead.</p>
#JOINT	<p>Wrist PTP</p> <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>Note: Use Wrist PTP if, with Standard, the robot passes through a wrist axis singularity. The orientation of the TCP changes continuously during the motion, but not uniformly. Wrist PTP 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**.

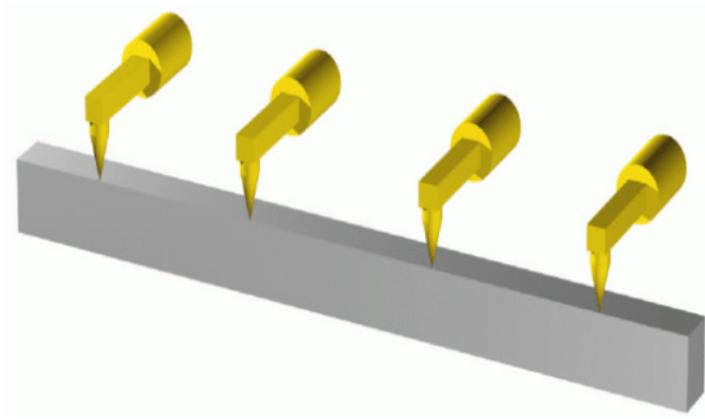


Fig. 8-7: Orientation control - Constant

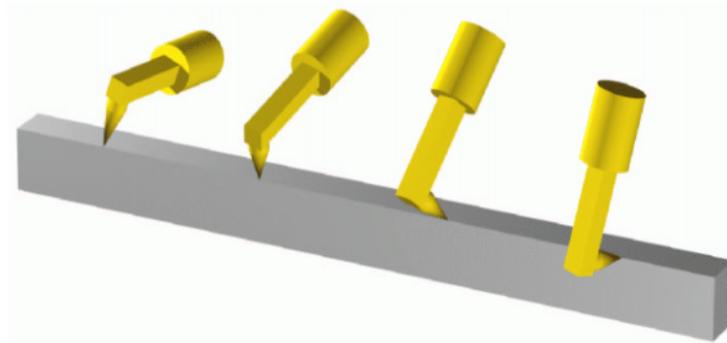


Fig. 8-8: Standard or Wrist PTP

CIR motion

During CIR 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 CIR motions as for LIN motions.

It is also possible to define for CIR motions whether the orientation control is to be base-related or path-related. This is defined via the system variable \$CIRC_TYPE.

\$CIRC_TYPE =	Description
#BASE	Base-related orientation control during the circular motion
#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 161)

8.6.1 Combinations of \$ORI_TYPE and \$CIRC_TYPE

\$ORI_TYPE = #CONSTANT, \$CIRC_TYPE = #PATH:

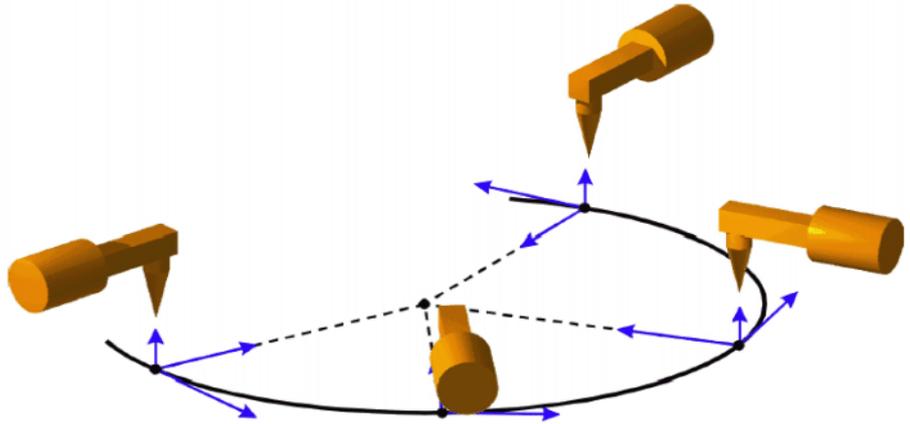


Fig. 8-9: Constant orientation, path-related

$\$ORI_TYPE = \#VAR, \$CIRC_TYPE = \#PATH:$

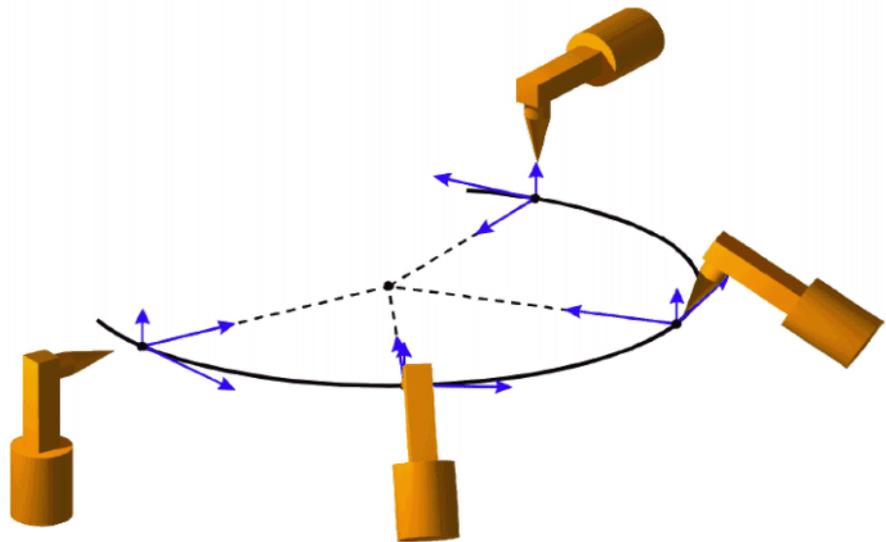


Fig. 8-10: Variable orientation, path-related

$\$ORI_TYPE = \#CONSTANT, \$CIRC_TYPE = \#BASE:$

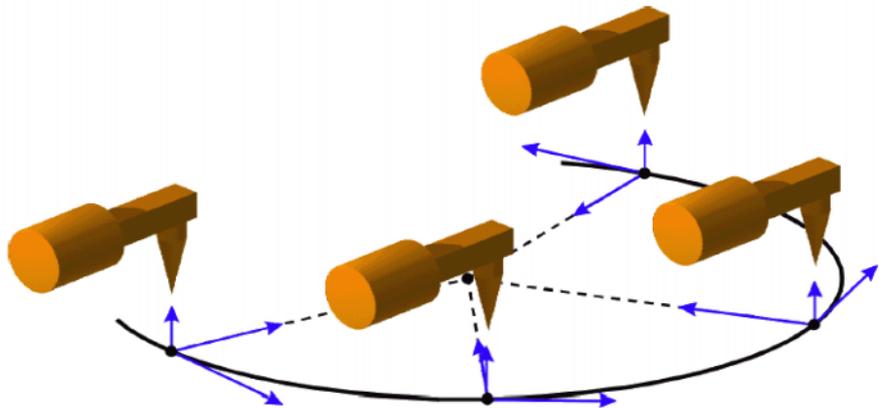


Fig. 8-11: Constant orientation, base-related

`$ORI_TYPE = #VAR, $CIRC_TYPE = #BASE:`

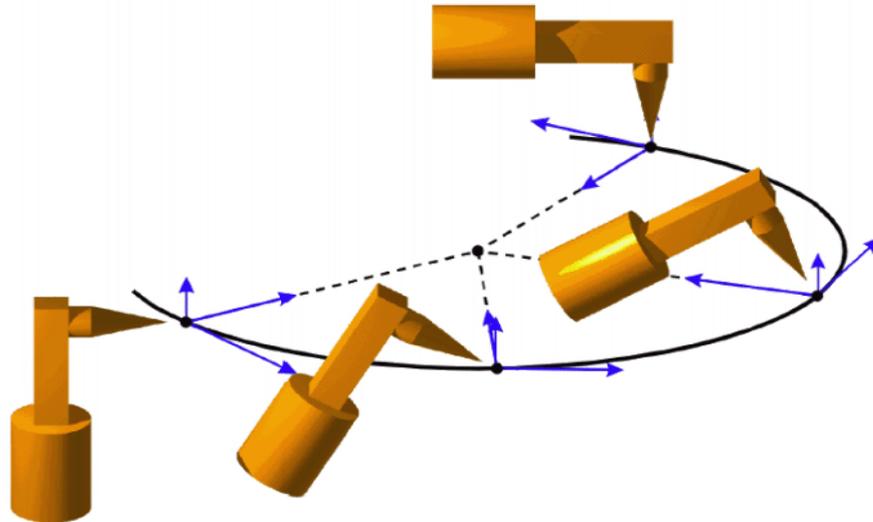


Fig. 8-12: Variable orientation, base-related

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.

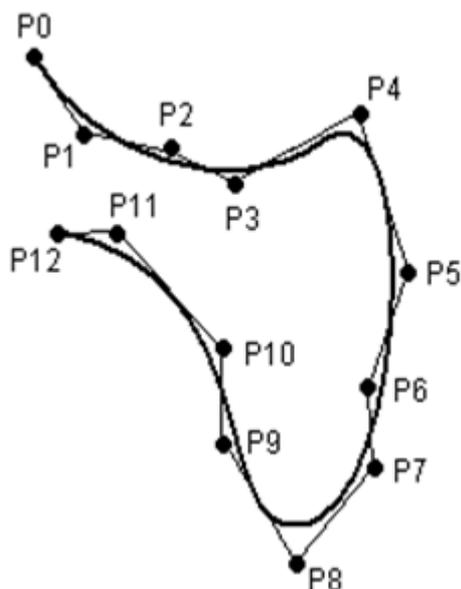


Fig. 8-13: Curved path with LIN

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 165)
- The path always remains the same, irrespective of the override setting, velocity or acceleration.
- Circles and tight radii are executed with great precision.

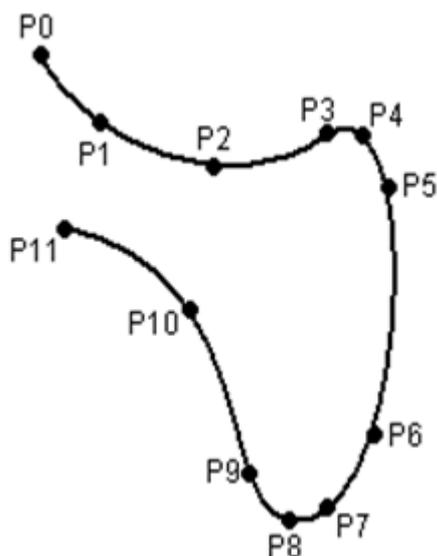


Fig. 8-14: Curved path with spline block

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.

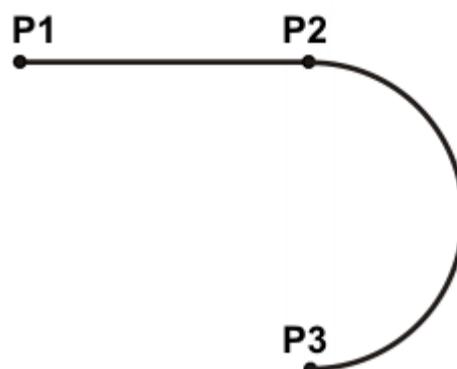


Fig. 8-15: Exact positioning at P2

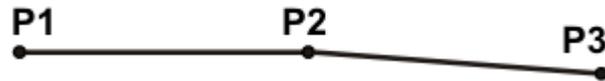


Fig. 8-16: Exact positioning at P2

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.

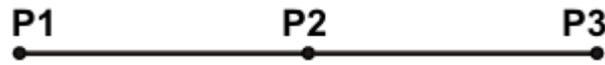


Fig. 8-17: P2 is executed without exact positioning.

- 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

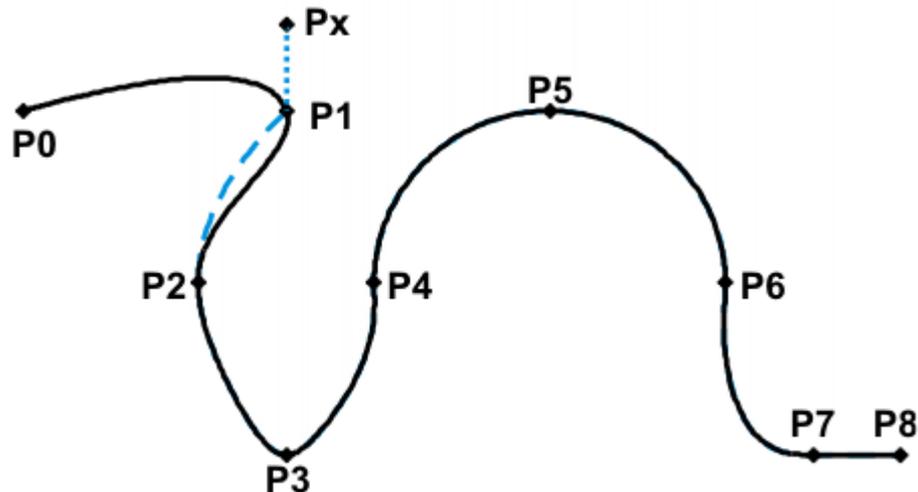


Fig. 8-18: Example: modified path in the case of block selection to P1

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 blocks	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
```

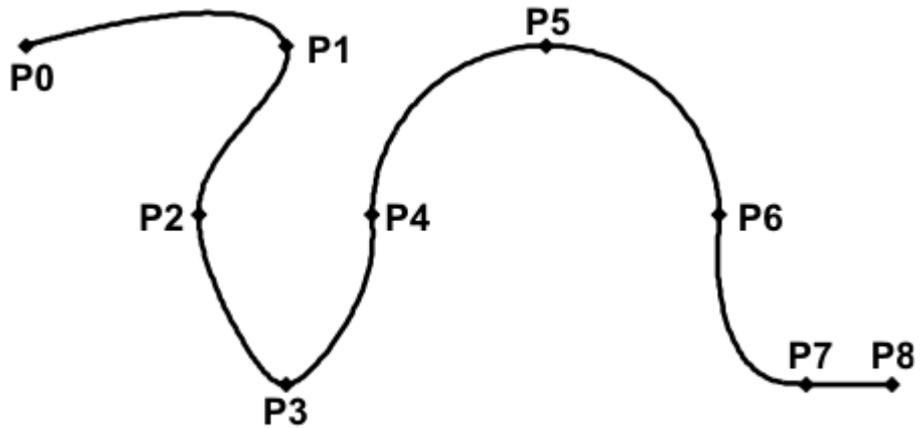


Fig. 8-19: Original path

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.

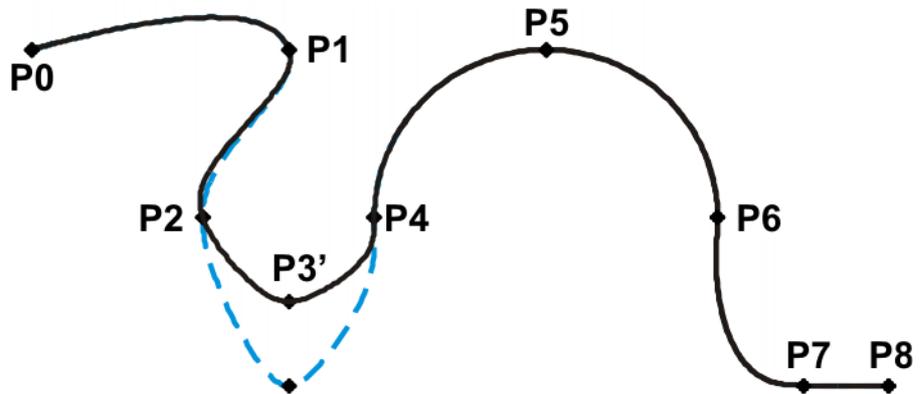


Fig. 8-20: Point has been offset

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
```

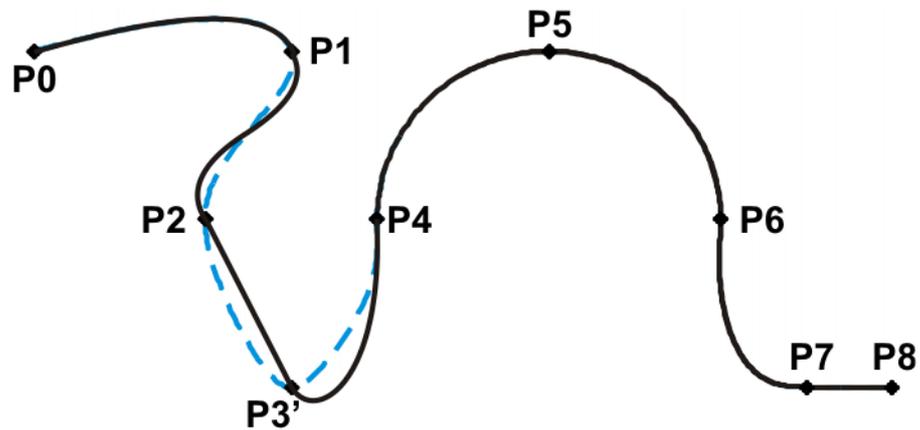


Fig. 8-21: Segment type has been changed

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

```

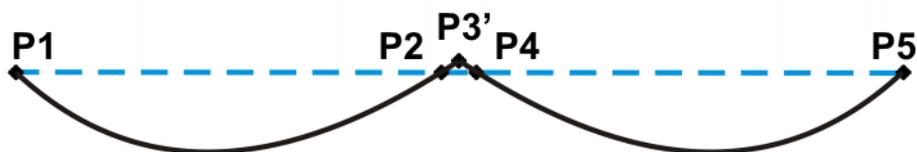


Fig. 8-23: Point has been offset

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 blocks 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

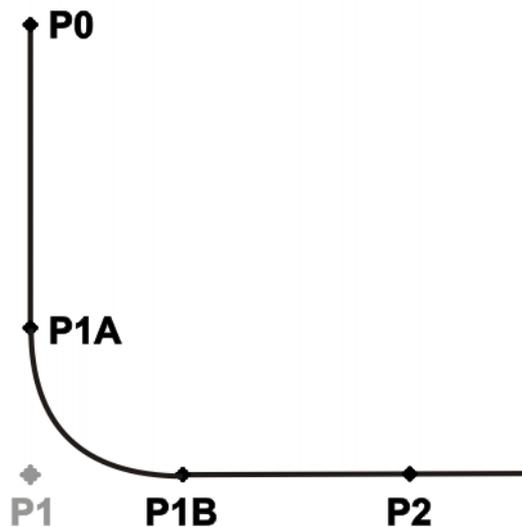


Fig. 8-24: Approximated motion - spline motion

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.

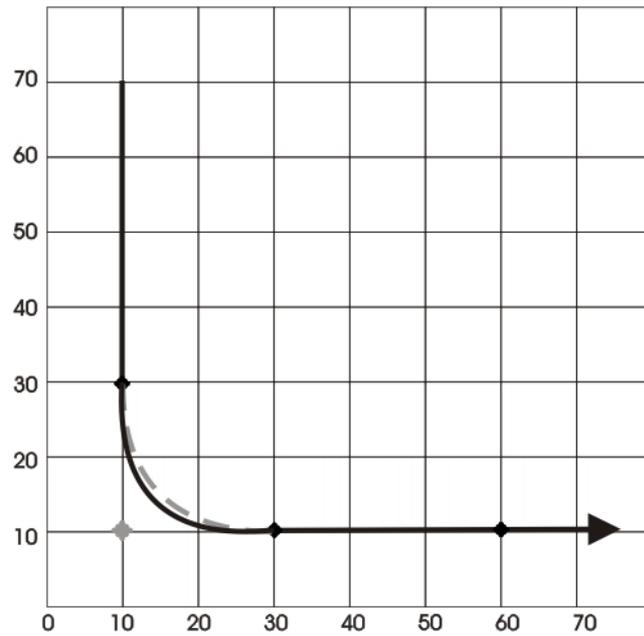


Fig. 8-25: Example: Approximated motion - spline motion 1

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
```

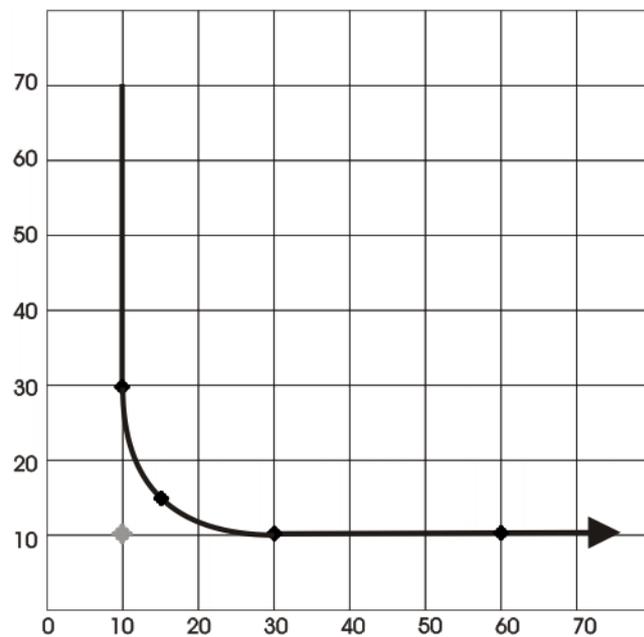


Fig. 8-26: Example: Approximated motion - spline motion 2

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.

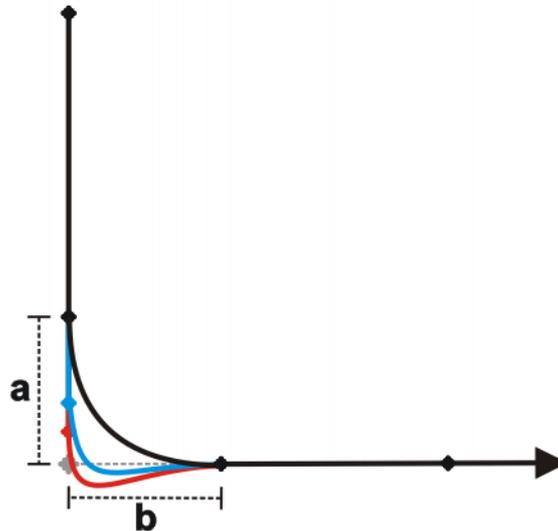


Fig. 8-27: SLIN-SPL-SLIN

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

8.7.6 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.

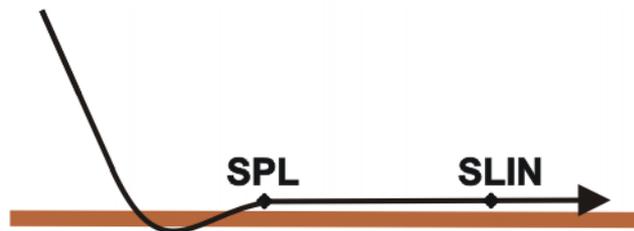


Fig. 8-28: Collision with work surface

In order to avoid a collision, observe the recommendations for the SLIN-SPL-SLIN transition.

(>>> 8.7.5.1 "SLIN-SPL-SLIN transition" Page 173)

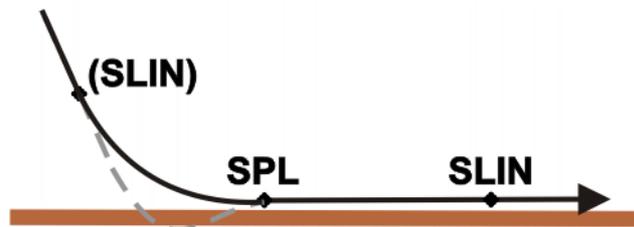


Fig. 8-29: Avoiding a collision with the work surface

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 via the system variable \$ORI_TYPE.

\$ORI_TYPE =	Description
#CONSTANT	<p>Constant orientation</p> <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>
#VAR	<p>Standard</p> <p>The orientation of the TCP changes continuously during the motion. At the end point, the TCP has the programmed orientation.</p>
#IGNORE	<p>Ignore Orientation</p> <p>This option is only available for spline segments (not for the spline block or for individual blocks).</p> <p>It is used if no specific orientation is required at a point.</p> <p>(>>> "#IGNORE" Page 175)</p>

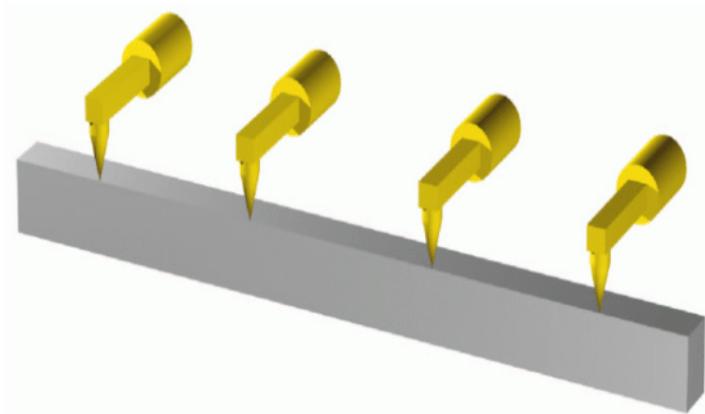


Fig. 8-30: Orientation control - Constant

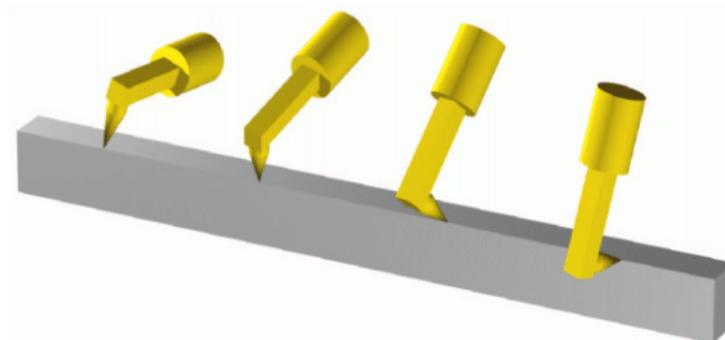


Fig. 8-31: Standard

#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 176)

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 177)

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 is defined via the system variable \$CIRC_TYPE.

\$CIRC_TYPE =	Description
#BASE	Base-related Base-related orientation control during the circular motion
#PATH	Path-related 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 161)

8.8.2 SCIRC: orientation behavior

Description 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.

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.

\$CIRC_MODE can only be written to by means of a SCIRC statement.
\$CIRC_MODE cannot be read.

Syntax

For auxiliary points:

```
$CIRC_MODE.AUX_PT.ORI = BehaviorAUX
```

For end points:

```
$CIRC_MODE.TARGET_PT.ORI = BehaviorEND
```

Explanation of the syntax

Element	Description
<i>BehaviorAUX</i>	Data type: ENUM <ul style="list-style-type: none"> ■ #INTERPOLATE: The programmed orientation is accepted in the auxiliary point. ■ #IGNORE: 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. ■ #CONSIDER (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.
<i>BehaviorEND</i>	Data type: ENUM <ul style="list-style-type: none"> ■ #INTERPOLATE: 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.) ■ #EXTRAPOLATE: 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.)

Restrictions

- If \$ORI_TYPE = #IGNORE for a SCIRC segment, \$CIRC_MODE is not evaluated.
- If a SCIRC segment is preceded by a SCIRC or SLIN segment with \$ORI_TYPE = #IGNORE, #CONSIDER cannot be used in this SCIRC segment.

For SCIRC with circular angle:

- #INTERPOLATE must not be set for the auxiliary point.
- If \$ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.
- If it is preceded by a spline segment with \$ORI_TYPE = #IGNORE, #EXTRAPOLATE must not be set for the end point.

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**.

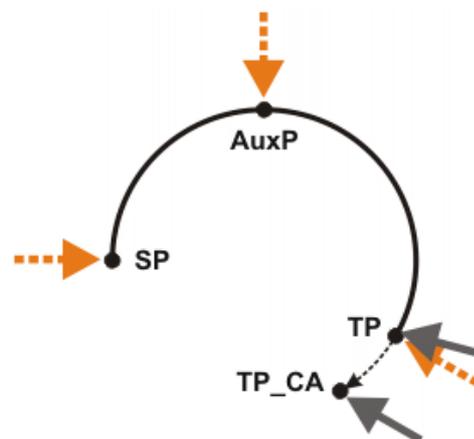


Fig. 8-32: #INTERPOLATE

SP	Start point
AuxP	Auxiliary point
TP	Programmed end point
TP_CA	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.

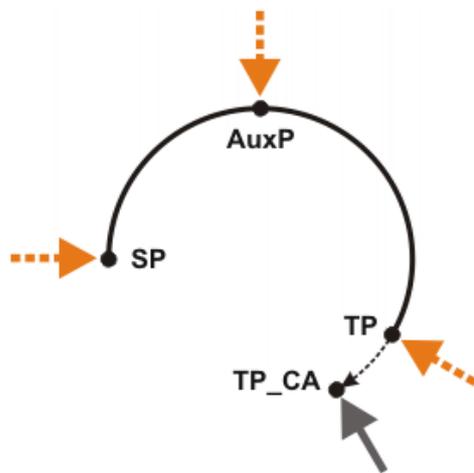


Fig. 8-33: #EXTRAPOLATE

8.9 Status and Turn

Status and Turn serve to define an unambiguous axis position in such cases where the same TCP position can be achieved with different axis positions.

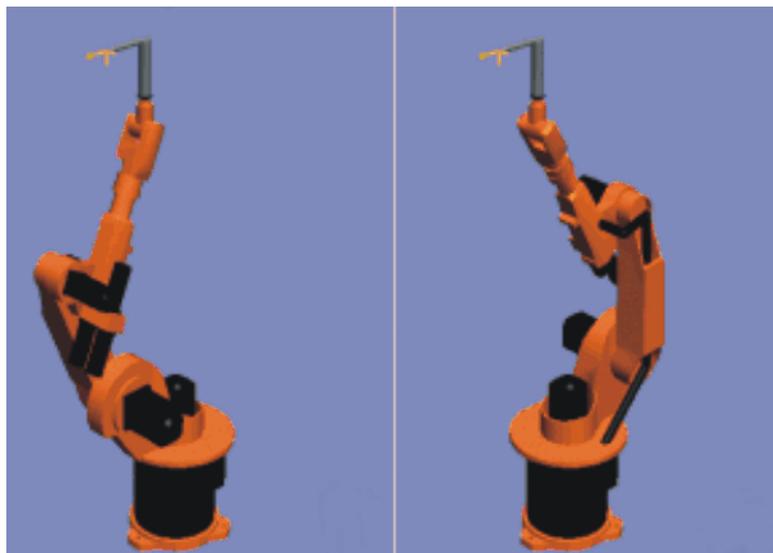


Fig. 8-34: Same TCP position, different axis position

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
```

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 a complete PTP instruction of type POS or E6POS in order to define an unambiguous starting position (or a complete PTP instruction of type AXIS or E6AXIS).

```
DEFDAT MAINPROGRAM ()
DECL POS XPOINT1={X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 27}
DECL FDAT FPOINT1 ...
...
ENDDDAT
```

Status and Turn can be omitted in the following instructions. The robot retains the previous Status value and selects the Turn value that results in the shortest 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.

Position	Value of bit 0
Overhead area If the x-value of the intersection of the wrist axes, relative to the A 1 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 A 1 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.

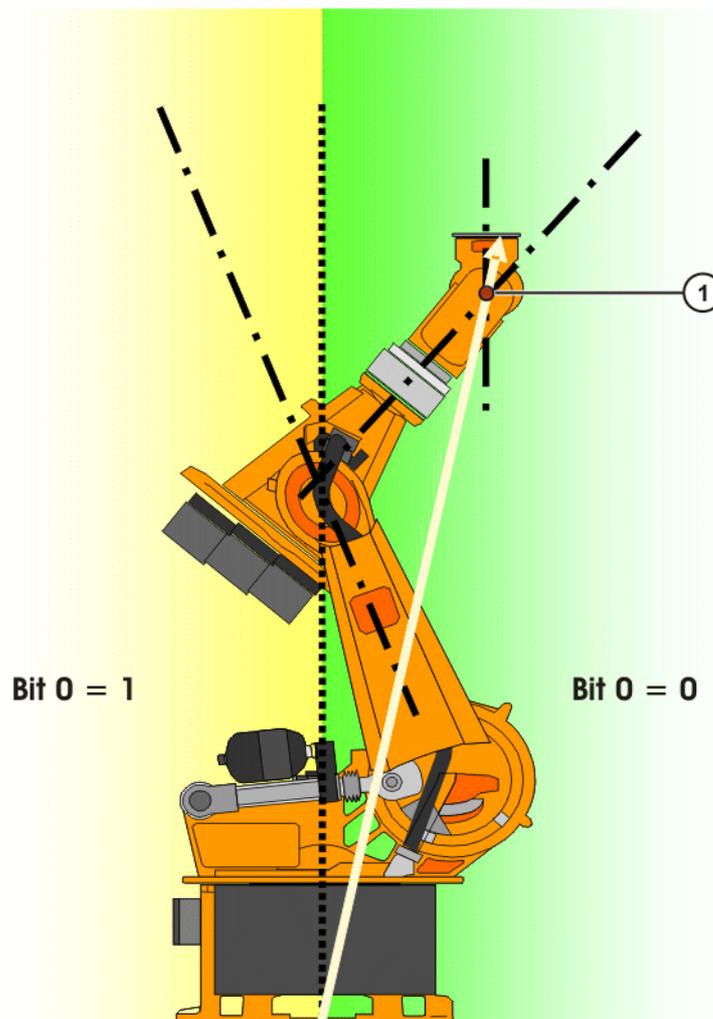


Fig. 8-35: Status bit 0

- 1 The intersection of the wrist axes (A 4, A 5, A 6) 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 of bit 1
$A\ 3 \geq 0$	Bit 1 = 1
$A\ 3 < 0$	Bit 1 = 0

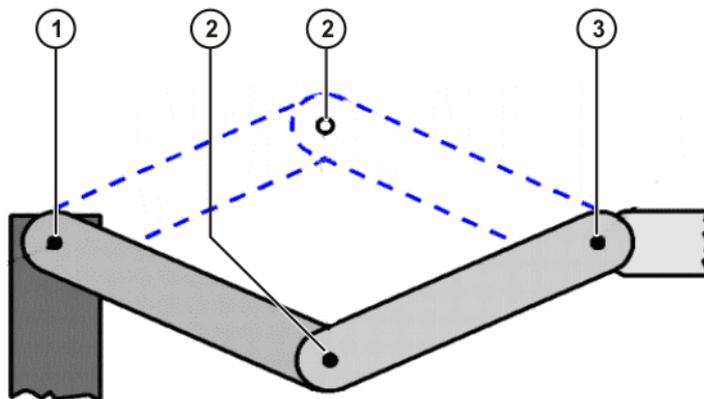


Fig. 8-36: Status bit 1

- 1 Axis 2
- 2 Axis 3
- 3 Axis 5

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.

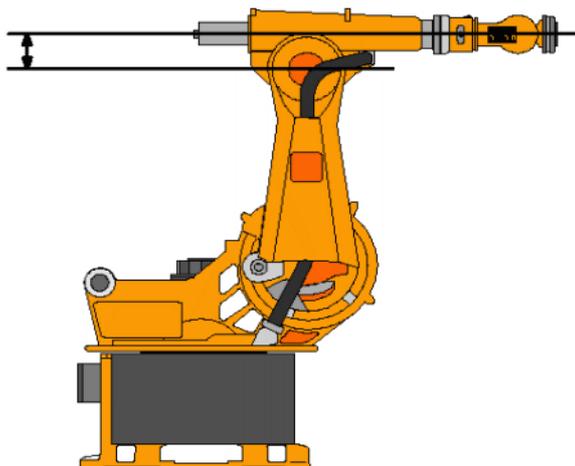


Fig. 8-37: Offset between A 3 and A 4: example KR 30

Bit 2

Bit 2 specifies the configuration of the wrist. This is dependent on the position of axis 5.

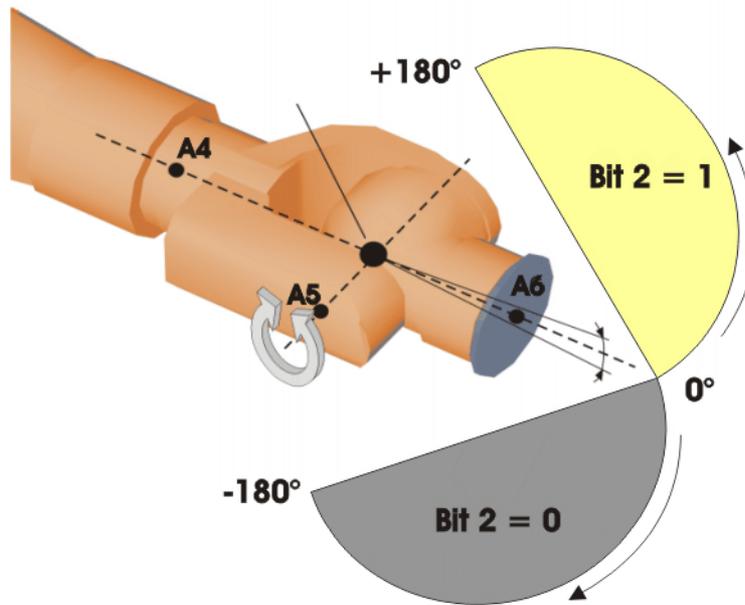


Fig. 8-38: Status bit 2

8.9.2 Turn

Description

The Turn specification makes it possible to move axes through angles greater than +180° or less than -180° 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 x = 0: angle of axis x+1 ≥ 0°

Bit x = 1: angle of axis x+1 < 0°

Value	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A6 ≥ 0°	A5 ≥ 0°	A4 ≥ 0°	A3 ≥ 0°	A2 ≥ 0°	A1 ≥ 0°
1	A6 < 0°	A5 < 0°	A4 < 0°	A3 < 0°	A2 < 0°	A1 < 0°

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
A 1	negative
A 2	negative
A 3	positive
A 4	positive
A 5	negative
A 6	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 VSS 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.

9.1 Programming motions

9.1.1 Programming a PTP motion



Caution!

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

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 softkey **Commands** > **Standard motion** > **PTP**.
4. Set the parameters in the inline form.
 - (>>> 9.1.2 "Inline form for PTP motions" Page 185)
5. Save instruction with **Cmd Ok**.

9.1.2 Inline form for PTP motions

Fig. 9-1: Inline form for PTP motions

Item	Description
1	Type of motion <ul style="list-style-type: none"> ■ PTP, LIN, CIR, KLIN, KCIR
2	Velocity <ul style="list-style-type: none"> ■ 1 ... 100 %

Item	Description
3	<p>Furthest distance before the end point at which approximate positioning can begin:</p> <ul style="list-style-type: none"> ■ 0 ... 100 % ■ 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 ■ 0 %: the motion stops exactly at the end point.
4	<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"> ■ 1 ... 100 %
5	<p>Tool number</p> <ul style="list-style-type: none"> ■ 1 ... 32
6	<p>Base number</p> <ul style="list-style-type: none"> ■ 1 ... 32
7	<p>PLC trigger</p> <p>Point in time before reaching the end point at which PLC instructions are triggered</p> <ul style="list-style-type: none"> ■ 0 ... 100 [1/100 s] <p>(>>> 9.4 "Programming PLC instructions" Page 201)</p>

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord.	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.1.3 Programming a LIN motion



Caution!

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

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 softkey **Commands > Standard motion > LIN**.
4. Set the parameters in the inline form.
(>>> 9.1.4 "Inline form for LIN motions" Page 187)
5. Save the instruction by pressing the **Cmd Ok** softkey.

9.1.4 Inline form for LIN motions

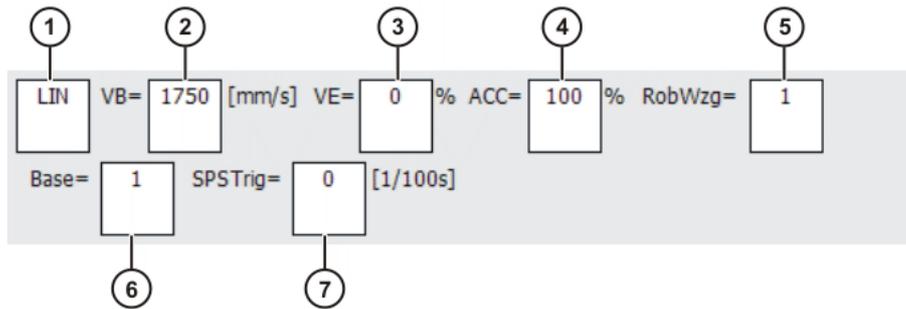


Fig. 9-2: Inline form for LIN motions

Item	Description
1	Type of motion <ul style="list-style-type: none"> ■ PTP, LIN, CIR, KLIN, KCIR
2	Velocity <ul style="list-style-type: none"> ■ 1 ... 2,000 mm/s
3	Furthest distance before the end point at which approximate positioning can begin: <ul style="list-style-type: none"> ■ 0 ... 100 % ■ Maximum distance 100%: half the distance between the start point and the end point relative to the contour of the LIN motion without approximate positioning ■ 0 %: the motion stops exactly at the end point.
4	Acceleration Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. <ul style="list-style-type: none"> ■ 1 ... 100 %
5	Tool number <ul style="list-style-type: none"> ■ 1 ... 32
6	Base number <ul style="list-style-type: none"> ■ 1 ... 32
7	PLC trigger Point in time before reaching the end point at which PLC instructions are triggered <ul style="list-style-type: none"> ■ 0 ... 100 [1/100 s] (>>> 9.4 "Programming PLC instructions" Page 201)

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord.	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.1.5 Programming a CIR motion



Caution!

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Program is selected.
- Operating mode T1 or T2.

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 softkey **Commands > Standard motion > CIR**.
4. Set the parameters in the inline form.
 - (>>> 9.1.6 "Inline form for CIR motions" Page 188)
5. Press **Coord Aux**.
6. Move the TCP to the position that is to be taught as the end point.
7. Save instruction with **Cmd Ok**.

9.1.6 Inline form for CIR motions

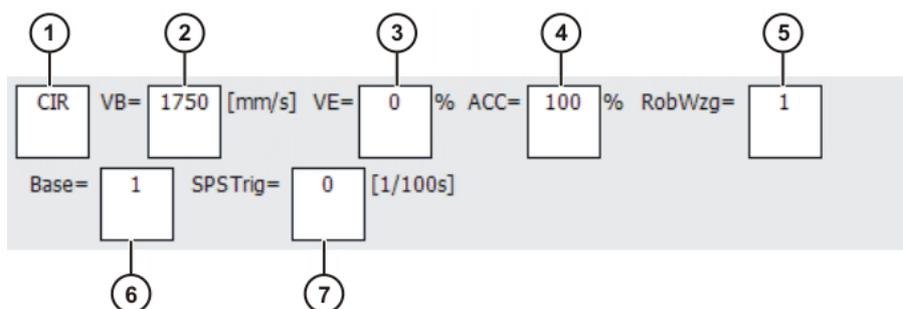


Fig. 9-3: Inline form for CIR motions

Item	Description
1	Type of motion <ul style="list-style-type: none"> ■ PTP, LIN, CIR, KLIN, KCIR
2	Velocity <ul style="list-style-type: none"> ■ 1 ... 2,000 mm/s
3	Furthest distance before the end point at which approximate positioning can begin: <ul style="list-style-type: none"> ■ 0 ... 100 % ■ Maximum distance 100%: half the distance between the start point and the end point relative to the contour of the CIR motion without approximate positioning ■ 0 %: the motion stops exactly at the end point.
4	Acceleration <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"> ■ 1 ... 100 %

Item	Description
5	Tool number <ul style="list-style-type: none"> ■ 1 ... 32
6	Base number <ul style="list-style-type: none"> ■ 1 ... 32
7	PLC trigger Point in time before reaching the end point at which PLC instructions are triggered <ul style="list-style-type: none"> ■ 0 ... 100 [1/100 s] (>>> 9.4 "Programming PLC instructions" Page 201)

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord AUX	Saves the current position of the TCP as the auxiliary point.
Coord END	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.2 Programming application-specific motions

Overview

The following application-specific motions can be programmed:

- Linear motions (KLIN)
 (>>> 9.2.1 "Programming a KLIN motion" Page 190)
- Circular motions (KCIR)
 (>>> 9.2.3 "Programming a KCIR motion" Page 191)
- Sensor-monitored linear motions (search run)
 (>>> 9.2.5 "Programming a sensor-monitored LIN motion" Page 193)

KLIN, KCIR

Because of the high accelerations that are possible, KLIN and KCIR motions are only suitable for adhesive bonding and arc welding applications.

In the case of KLIN and KCIR motions, the approximate positioning radius is specified precisely in mm. The robot always attempts to maintain the programmed path; the TCP does not deviate from the path. If the programmed velocity is too high for maintaining the path, it is automatically reduced.



In the case of LIN and CIR motions, the path in the approximate positioning range cannot be specified exactly. The robot always attempts to maintain the programmed velocity; the TCP deviates from the path.

Search run

Sensor-monitored linear motion is suitable for tasks such as palletizing. The robot guides the TCP along a LIN motion to an end point. This end point must always be situated behind a stack of pallets that are to be processed.

2 sensors on the gripper control the linear motion:

- A signal from the far sensor causes the programmed velocity to be reduced if a defined distance from the pallet stack has been reached.

- A signal from the near sensor causes the linear motion to be terminated if the pallet stack has been reached. At this point, the configured gripper functions are carried out.

The linear motion is repeated as often as the near sensor sends a signal. When it no longer sends a signal, the motion to the next point is executed.

9.2.1 Programming a KLIN motion



Caution!

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

Precondition

- Applications: Adhesive bonding or arc welding
- A program is selected.
- Operating mode T1 or T2

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 softkey **Commands > Technology motion > KLIN**.
4. Set the parameters in the inline form.
 - (>>> 9.2.2 "Inline form for KLIN motions" Page 190)
5. Save instruction with **Cmd Ok**.

9.2.2 Inline form for KLIN motions

Fig. 9-4: Inline form for KLIN motions

Item	Description
1	Type of motion <ul style="list-style-type: none"> ■ PTP, LIN, CIR, KLIN, KCIR
2	Velocity <ul style="list-style-type: none"> ■ 1 ... 2,000 mm/s
3	Furthest distance before the end point at which approximate positioning can begin: <ul style="list-style-type: none"> ■ 0 ... 300 mm ■ Maximum distance 300 mm: half the distance between the start point and the end point relative to the contour of the KLIN motion without approximate positioning. If a value is entered that is too high, the maximum value is automatically used. ■ 0 mm: the motion stops exactly at the end point.

Item	Description
4	Acceleration <ul style="list-style-type: none"> ■ 1 ... 100% Maximum value 100%: 10 m/s ²
5	Tool number <ul style="list-style-type: none"> ■ 1 ... 32
6	Base number <ul style="list-style-type: none"> ■ 1 ... 32
7	PLC trigger Point in time before reaching the end point at which PLC instructions are triggered <ul style="list-style-type: none"> ■ 0 ... 100 [1/100 s] (>>> 9.4 "Programming PLC instructions" Page 201)

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord.	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.2.3 Programming a KCIR motion



Caution!

When programming motions, it must be ensured that the energy supply system is not wound up or damaged during program execution.

- Precondition**
- Applications: Adhesive bonding or arc welding
 - A program is selected.
 - Operating mode T1 or T2

- 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 softkey **Commands > Technology motion > KCIRC**.
 4. Set the parameters in the inline form.
 (>>> 9.2.4 "Inline form for KCIR motions" Page 192)
 5. Press **Coord Aux**.
 6. Move the TCP to the position that is to be taught as the end point.
 7. Save instruction with **Cmd Ok**.

9.2.4 Inline form for KCIR motions

Fig. 9-5: Inline form for KCIR motions

Item	Description
1	Type of motion <ul style="list-style-type: none"> ■ PTP, LIN, CIR, KLIN, KCIR
2	Velocity <ul style="list-style-type: none"> ■ 1 ... 2,000 mm/s
3	Furthest distance before the end point at which approximate positioning can begin: <ul style="list-style-type: none"> ■ 0 ... 300 mm ■ Maximum distance 300 mm: half the distance between the start point and the end point relative to the contour of the KCIR motion without approximate positioning. If a value is entered that is too high, the maximum value is automatically used. ■ 0 mm: the motion stops exactly at the end point.
4	Acceleration <ul style="list-style-type: none"> ■ 1 ... 100 % Maximum value 100%: 10 m/s ²
5	Tool number <ul style="list-style-type: none"> ■ 1 ... 32
6	Base number <ul style="list-style-type: none"> ■ 1 ... 32
7	PLC trigger Point in time before reaching the end point at which PLC instructions are triggered <ul style="list-style-type: none"> ■ 0 ... 100 [1/100 s] (>>> 9.4 "Programming PLC instructions" Page 201)

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord AUX	Saves the current position of the TCP as the auxiliary point.
Coord END	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.2.5 Programming a sensor-monitored LIN motion

- Precondition**
- Program is selected.
 - Operating mode T1 or T2.
- 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 softkey **Commands > Technology motion > SUCHLAUF**.
 4. Set the parameters in the inline form.
 - (>>> 9.2.6 "Inline form "LIN SUCHEN"" Page 193)
 5. Save instruction with **Cmd Ok**.

9.2.6 Inline form "LIN SUCHEN"

The screenshot shows the 'LIN SUCHEN' inline form with the following fields and values:

- 1**: VB= 1750 [mm/s]
- 2**: RobWzg= 1
- 3**: Base= 1
- 4**: Fern= E
- 5**: 1 (input field)
- 6**: Vred= 10 %
- 7**: Nah= E
- 8**: 1 (input field)

Fig. 9-6: Inline form "LIN SUCHEN"

Item	Description
1	Velocity <ul style="list-style-type: none"> ■ 1 ... 2,000 mm/s
2	Tool number <ul style="list-style-type: none"> ■ 1 ... 32
3	Base number <ul style="list-style-type: none"> ■ 1 ... 32
4	Signal from far sensor <ul style="list-style-type: none"> ■ E: input of far sensor ■ M: cyclical flag of far sensor
5	Number of input or cyclical flag (far sensor) <ul style="list-style-type: none"> ■ E: 1 ... 4,096 ■ M: 1 ... 200
6	Reduction of the velocity <ul style="list-style-type: none"> ■ 1 ... 100 % Maximum value 100%: Velocity is not reduced.
7	Signal from near sensor <ul style="list-style-type: none"> ■ E: input of near sensor ■ M: cyclical flag of near sensor
8	Number of input or cyclical flag (near sensor) <ul style="list-style-type: none"> ■ E: 1 ... 4,096 ■ M: 1 ... 200



The evaluation cycle of cyclical flags depends on the number of cyclical flags and their definition and cannot be determined exactly.

The following softkeys are available:

Softkey	Description
Command Abort	Closes the inline form without saving the motion instruction in the program.
Coord.	Saves the current position of the TCP as the end point.
Cmd OK	Closes the inline form. The settings in the inline form are saved. The current position of the TCP is saved as the end point.

9.3 Modifying programmed motions

9.3.1 Modifying motion parameters

- Precondition**
- Program is selected.
 - Operating mode T1 or T2.

- Procedure**
1. Position the cursor in the line containing the instruction that is to be changed.
 2. Press the **Change** softkey. The inline form for this instruction is opened.
 3. Modify parameters.
 4. Save changes by pressing the **Cmd Ok** softkey.

9.3.2 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**
- Program is selected.
 - Operating mode T1 or T2.

- 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 the **Change** softkey. The inline form for this instruction is opened.
 4. For PTP, LIN and KLIN motions: press the **Coord** softkey to accept the current position of the TCP as the new end point.
For CIR and KCIR motions:
 - Press the **Coord Aux** softkey to accept the current position of the TCP as the new auxiliary point.
 - Press the **Coord End** softkey 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 the **Cmd Ok** softkey.

9.3.3 Block function

Overview The block function can be used to select and modify multiple consecutive motion blocks simultaneously.

- Modify the motion parameters of a path section.
(>>> 9.3.3.1 "Modifying blocks of motion parameters" Page 195)
- Offset the points in a path section by a defined value.
(>>> 9.3.3.2 "Transforming blocks of coordinates" Page 195)



The block function can only be used for motion instructions.

9.3.3.1 Modifying blocks of motion parameters

- Precondition**
- Program is selected.
 - Operating mode T1 or T2.

- Procedure**
1. Select the motion instructions to be modified. (Only consecutive motion instructions can be modified as a block.)
 2. Press the **Change** softkey. The inline form of the first selected motion block opens.
 3. Modify parameters.
 4. Save and apply changes for all selected motion blocks by pressing the **Cmd Ok** softkey.

When the changes are applied, the system checks whether the parameters for all motion blocks can be applied, e.g. it is not possible to apply the PTP parameter velocity to the LIN motion block.

9.3.3.2 Transforming blocks of coordinates

- Precondition**
- Program is selected.
 - Operating mode T1 or T2.

- Procedure**
1. Select the motion instructions to be modified. (Only consecutive motion instructions can be modified as a block.)
 2. Select the softkey **Program > Marked Region**. Select transformation type.
The corresponding window is opened.
(>>> 9.3.3.3 "'Mirroring" window" Page 198)
(>>> 9.3.3.4 "'Point transformation - axis-specific" window" Page 199)
(>>> 9.3.3.5 "'Cartesian point transformation" window" Page 200)
 3. Enter values for the transformation and press the **Calculate** softkey.

Overview The following transformation types are available:

- Transform - Cartesian Base
- Transform - Cartesian TCP
- Transform - Cartesian World
- Transform - Axis Specific
- Mirroring

Transform - Base **Transform - Cartesian Base:**

The transformation refers to the current BASE coordinate system.

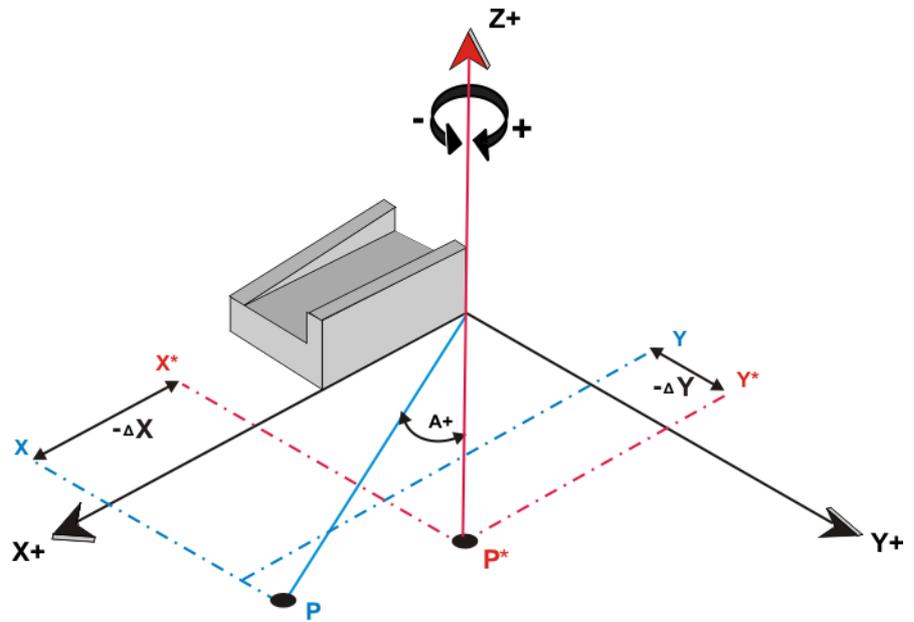


Fig. 9-7: Transform - Cartesian Base

Point P is offset by ΔX and ΔY in the negative direction. The new position of the point is P^* .

Transform - TCP

Transform - Cartesian TCP:

The transformation refers to the current TOOL coordinate system.

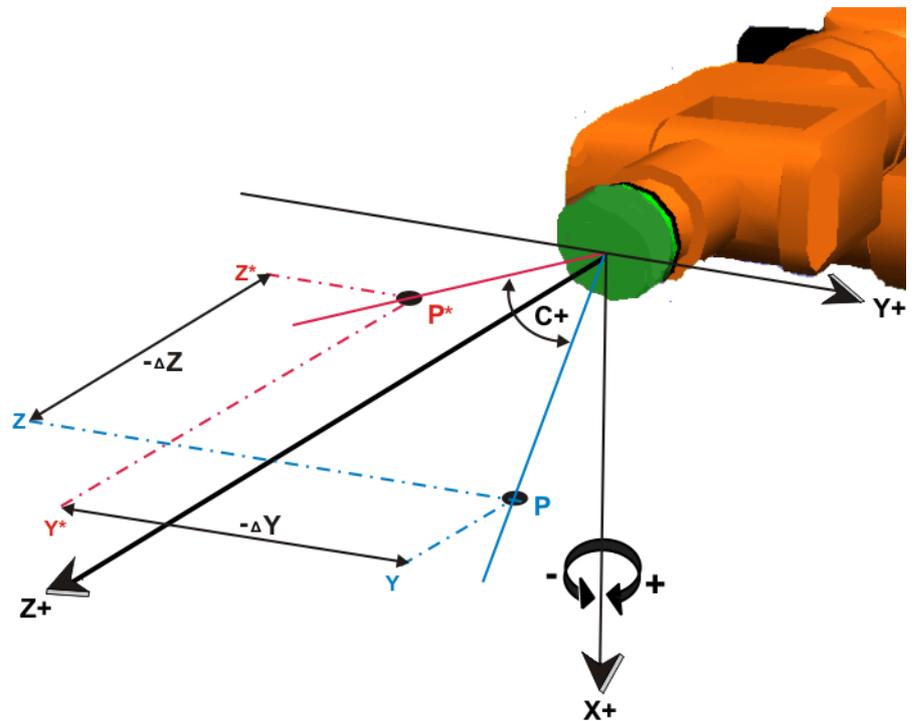


Fig. 9-8: Transform - Cartesian TCP

Point P is offset by ΔZ and ΔY in the negative direction. The new position of the point is P^* .

Transform - World

Transform - Cartesian World:

The transformation is relative to the WORLD coordinate system.

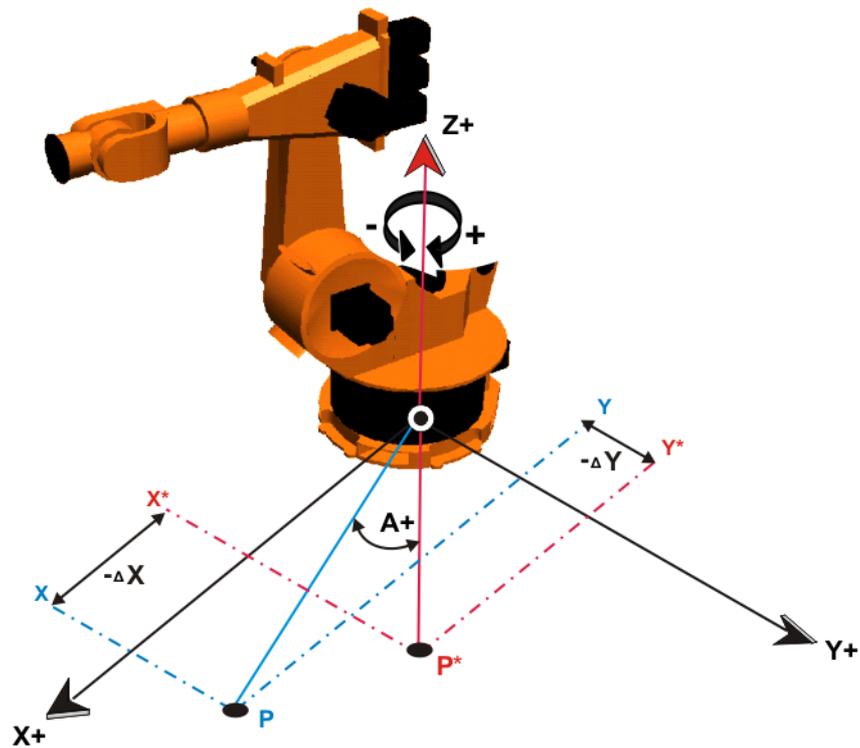


Fig. 9-9: Transform - Cartesian World

Point P is offset by ΔX and ΔY in the negative direction. The new position of the point is P^* .

Transform - Axis Specific

Transform - Axis Specific:

The transformation is axis-specific.

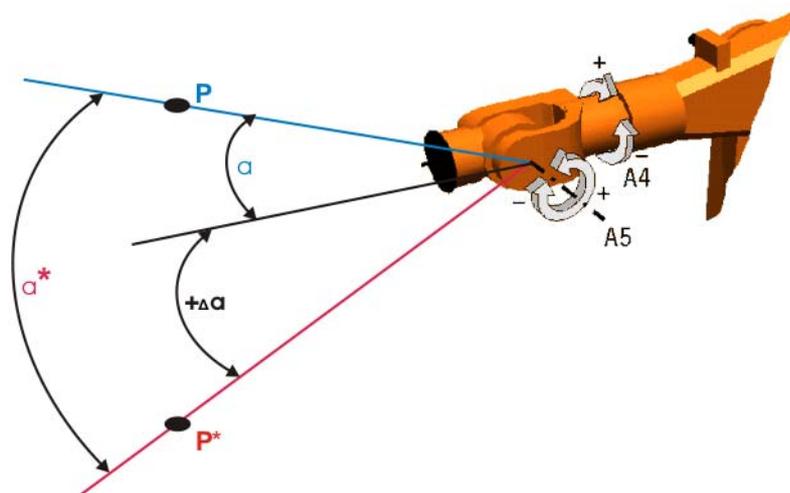


Fig. 9-10: Transform - Axis Specific

Axis A5 is rotated by the angle $\Delta\alpha$. The new position of point P is P^* .

Mirroring

Mirroring:

Mirroring in the XY plane of the ROBROOT coordinate system.

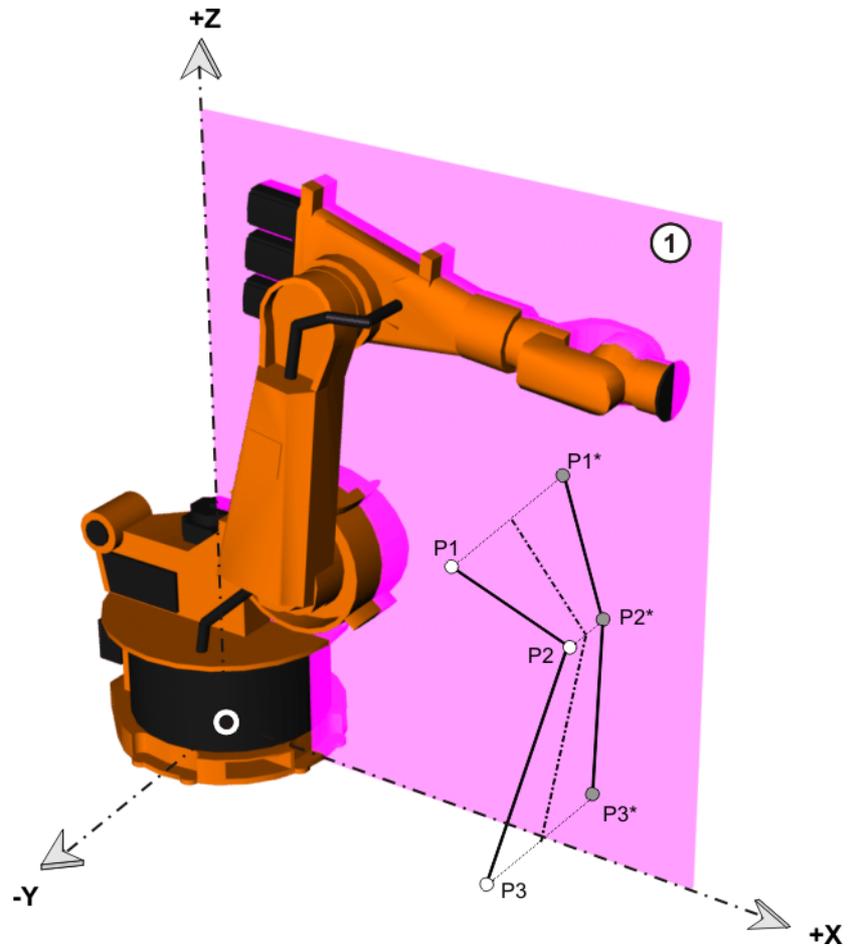


Fig. 9-11: Mirroring

Points P1, P2 and P3 are mirrored in the XY plane (1). The new positions of the points are P1*, P2* and P3*.

9.3.3.3 “Mirroring” window

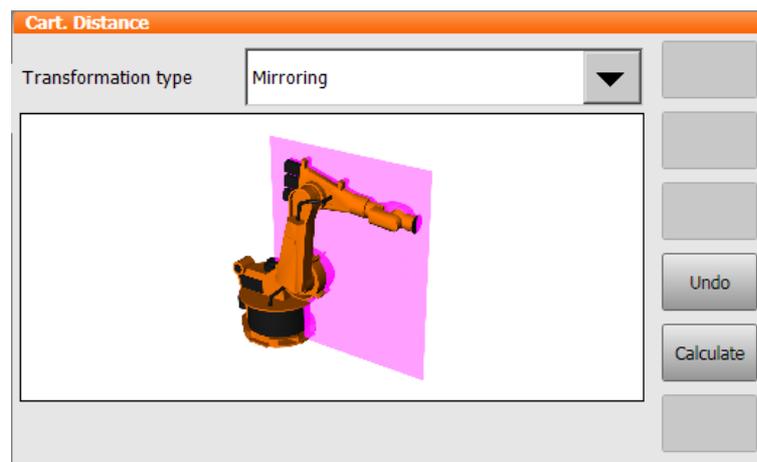


Fig. 9-12: Mirroring

No values need to be entered in this window. Pressing the **Calculate** softkey mirrors the point coordinates in the XZ plane of the ROBROOT coordinate system.



Following mirroring of the coordinates, the tool used must also be mirrored in the XZ plane.

The following softkeys are available:

Softkey	Description
Calculate	Mirrors the coordinates of the selected motion points in the XZ plane, converts the coordinates to axis angles and applies the new values.
Undo	Undoes the mirroring and restores the old point data.



Only selected points with a complete E6POS definition are copied. This includes, for example, all those that were generated via inline forms during programming. Points without a complete E6POS definition are ignored during the point offset.

9.3.3.4 "Point transformation - axis-specific" window

Fig. 9-13: Point transformation - axis-specific

Item	Description
1	Selection of the transformation type (>>> 9.3.3.2 "Transforming blocks of coordinates" Page 195)
2	Rotation group: input boxes for the position offset of axes A1 ... A6 <ul style="list-style-type: none"> Range of values: Dependent on the configuration of the axis-specific workspaces <p>The E1 .. E6 softkey switches to the External axes group: input boxes for the position offset of axes E1 ... E6</p> <p>Note: Values can only be entered for configured axes.</p>

The following softkeys are available:

Softkey	Description
E1 .. E6 / A1 .. A6	Toggles between the Rotation and External axes groups.

Softkey	Description
Undo	Undoes the transformation and restores the old point data.
Calculate	Calculates the point transformation and applies it to all selected motion points. If the transformation would cause a point to be situated outside the configured workspace, the point is not transformed.



Only selected points with a complete E6POS definition are copied. This includes, for example, all those that were generated via inline forms during programming. Points without a complete E6POS definition are ignored during the point offset.

9.3.3.5 “Cartesian point transformation” window

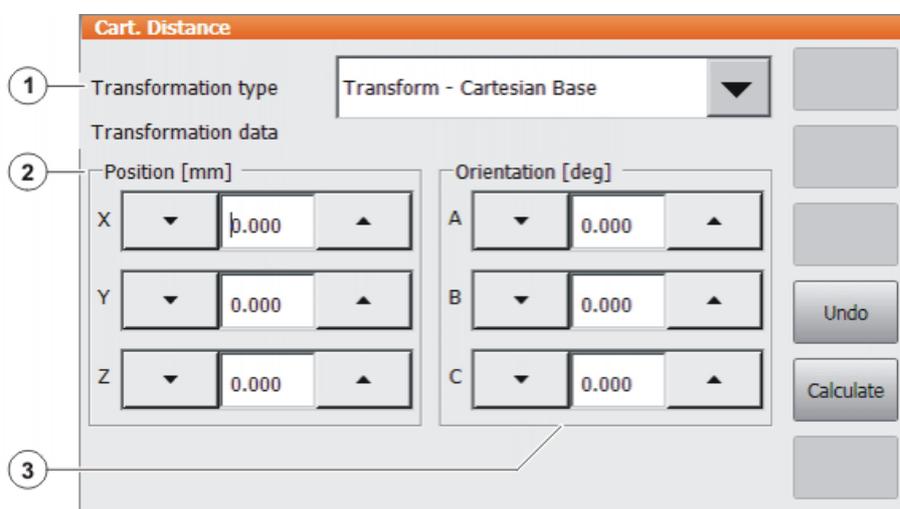


Fig. 9-14: Point transformation - Cartesian

Item	Description
1	Selection of the transformation type (>>> 9.3.3.2 "Transforming blocks of coordinates" Page 195)
2	Position group: input boxes for the point transformation in the X, Y, Z direction <ul style="list-style-type: none"> Range of values: Dependent on the configuration of the Cartesian workspaces
3	Orientation group: input boxes for the transformation of the A, B, C orientation <ul style="list-style-type: none"> Range of values: Dependent on the configuration of the Cartesian workspaces

The following softkeys are available:

Softkey	Description
Undo	Undoes the transformation and restores the old point data.
Calculate	Calculates the point transformation and applies it to all selected motion points. If the transformation would cause a point to be situated outside the configured workspace, the point is not transformed.



Only selected points with a complete E6POS definition are copied. This includes, for example, all those that were generated via inline forms during programming. Points without a complete E6POS definition are ignored during the point offset.

9.4 Programming PLC instructions

Description Motions can be linked to PLC instructions in VSS. PLC instructions are used to monitor the inputs or outputs of the robot controller for specific programmed conditions.

The inline forms for programming PLC instructions are available if the Point PLC of the motion is open.

Precondition

- Program is selected.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in the line containing the motion that is to be linked to one or more PLC instructions.
2. Open the Point PLC by pressing the **PLC Open** softkey.
3. Program PLC instructions.
4. Close the Point PLC by pressing the **PLC Close** softkey.

9.4.1 Boolean operands

Operand	Number	Description	Range of values
E	1 ... 4096	Input	TRUE, FALSE
A	1 ... 4096	Output	TRUE, FALSE
M	1 ... 200	Cyclical flag (Merker)	TRUE, FALSE
F	1 ... 999	Flag	TRUE, FALSE
T	1 ... 20	Timer flag	TRUE, FALSE
S	1 ... 32	Sensor flag	TRUE, FALSE

9.4.2 Arithmetic operands

Operand	Number	Description	Range of values
num	-	Number	-99,999 ... +99,999
i	1 ... 22	Integer counter	-9,999 ... +9,999
bin	1 ... 20	Binary output	-9,999 ... +9,999
binin	1 ... 20	Binary input	-9,999 ... +9,999
t	1 ... 20	Timer	-99,999 ... +99,999 [1/10 s]

Operand	Number	Description	Range of values
ana	1 ... 16	Analog output	-9,999 ... +9,999 mV
anain	1 ... 16	Analog input	-9,999 ... +9,999 mV
p	1 ... 256	Process parameter	-99,999 ... +99,999

9.4.3 Operators

Operator	Description	Function
(Open brackets	Operator
)	Close brackets	Operator
+	Or	Boolean operator
&	And	Boolean operator
+	Plus	Arithmetic operator
-	Minus	Arithmetic operator
*	Times	Arithmetic operator
/	Divided by	Arithmetic operator
<	Less than	Relational operator
>	Greater than	Relational operator
<=	Less than or equal to	Relational operator
>=	Greater than or equal to	Relational operator
=	Equal to	Relational operator
!	Not equal to	Relational operator
ON	Switch on	Boolean constant
OFF	Switch off	Boolean constant

9.4.4 Priority of the operators

The priority of the operators specifies the order in which the operators are evaluated within a statement.

Priority	Operator
1	!
2	*, /
3	+, -
4	&
5	+
6	=, <, >, <=, >=

9.4.5 Inserting a new operator or operand

Description If required, in the case of a Boolean or arithmetic operation, multiple operators or operands (max. 11) can be linked in an inline form.

Precondition ■ Program is selected.

- Operating mode T1 or T2.

Procedure

1. Select the operator box after which an additional operator or operand is to be inserted.
2. Press the **New OP** softkey. The additional operator or operand is displayed in the inline form.

The operator or operand can be deleted again by pressing the **Delete OP** softkey. For this, the corresponding operator box must be selected beforehand.

9.4.6 Setting an output, cyclical flag (Merker) or flag**Precondition**

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands > SPS ==> > A/M/F=.**
3. Set the parameters in the inline form.
 - (>>> 9.4.7 "Inline form "A/M/F"" Page 203)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.7 Inline form "A/M/F"**Description**

The instruction can be used to set an output, cyclical flag or flag and link the setting of the signal to a condition (Boolean operand). If required, several operands (maximum 11) can be linked.

Cyclical flags are evaluated cyclically. Outputs and flags are only evaluated at the time of assignment.

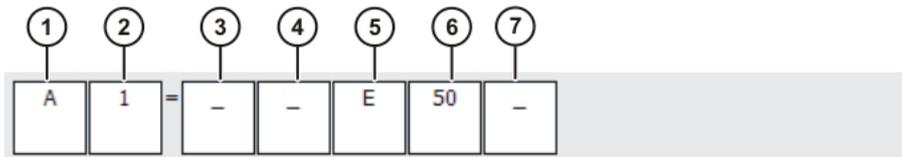


Fig. 9-15: Inline form "A/M/F"

Item	Description
1	Signal <ul style="list-style-type: none"> ■ A: output ■ M: Merker (cyclical flag) ■ F: flag
2	Number of the signal <ul style="list-style-type: none"> ■ A: 1 ... 4,096 ■ M: 1 ... 200 ■ F: 1 ... 999
3	Operator <ul style="list-style-type: none"> ■ →, (
4	Operator. This box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ →, !

Item	Description
5	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
6	Number of the operand. This box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
7	Operator <ul style="list-style-type: none"> ■ -,)

Example



Fig. 9-16: Inline form “A/M/F” – setting an output (example)

Output 3 is set if condition 1 or condition 2 is met:

- Condition 1: input 3 and output 7 are TRUE.
- Condition 2: input 3 and output 11 are TRUE.

9.4.8 Setting an integer counter or binary output

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands > SPS ==> > i/bin=**.
3. Set the parameters in the inline form.
(>>> 9.4.9 "Inline form “i/bin”" Page 204)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.9 Inline form “i/bin”

Description

The instruction can be used to assign a value to an integer counter or binary output and link the setting of the signal to a condition (Boolean operand). If required, several arithmetic operands (maximum 11) can be linked to the value assignment.

A binary output can be used to assign binary values to a number of defined outputs, e.g. to send program numbers to other devices or controllers. Integer counters are used, for example, to count weld spots.

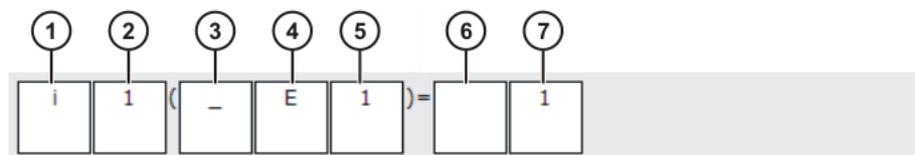


Fig. 9-17: Inline form “i/bin”

Item	Description
1	Signal <ul style="list-style-type: none"> ■ i: integer counter ■ bin: binary output
2	Number of the signal <ul style="list-style-type: none"> ■ i: 1 ... 22 ■ bin: 1 ... 20
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ _, !
4	Boolean operand (condition) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
7	Value that is assigned to the counter or output <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)

Example

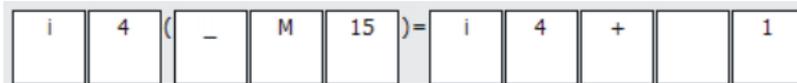


Fig. 9-18: Inline form "i/bin" – setting a counter (example)

The value 1 is added to integer counter 4 if the following condition is met: Merker (cyclical flag) 15 is TRUE.

9.4.10 Starting a timer

- Precondition**
- Program is selected.
 - Point PLC is open.
 - Operating mode T1 or T2.

- Procedure**
1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
 2. Select the softkey **Commands** > **SPS ==>** > **t=**.
 3. Set the parameters in the inline form.
(>>> 9.4.11 "Inline form "t= (Start)"" Page 205)
 4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.11 Inline form "t= (Start)"

Description The instruction can be used to assign a start value to a timer and link the start of the timer to a condition (Boolean operand). If required, several arithmetic operands (maximum 11) can be linked to the assignment of the start value.

The condition for stopping the timer is programmed using the inline form "t=STOP".

(>>> 9.4.13 "Inline form "t=STOP"" Page 207)

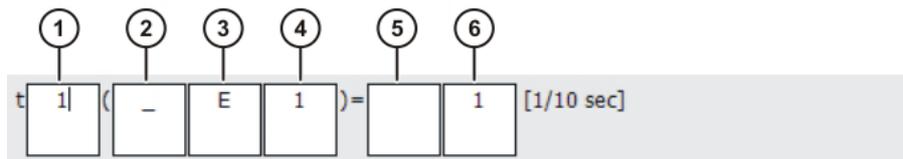


Fig. 9-19: Inline form "t= (Start)"

Item	Description
1	Number of the timer <ul style="list-style-type: none"> 1 ... 20
2	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> _, !
3	Boolean operand (condition) <ul style="list-style-type: none"> ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	Arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
6	Start value of the timer. A negative start value can be assigned to the timer. If the value of the timer is > 0, the timer flag is set. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)

Example



Fig. 9-20: Inline form "t= (Start)" – example

Timer 5 is started with 30 s if the following condition is met: Flag 25 is FALSE.

9.4.12 Stopping a timer

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

- In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
- Select the softkey **Commands** > **SPS ==>** > **t=STOP**.
- Set the parameters in the inline form.
 (>>> 9.4.13 "Inline form "t=STOP"" Page 207)
- Save the instruction by pressing the **Cmd Ok** softkey.

9.4.13 Inline form "t=STOP"

Description The instruction can be used to stop the timer and link the stopping of the timer to a condition (Boolean operand). If required, several operands (maximum 11) can be linked.

The assignment of a start value and the condition for starting the timer are programmed using the inline form "t= (Start)".

(>>> 9.4.11 "Inline form "t= (Start)" Page 205)

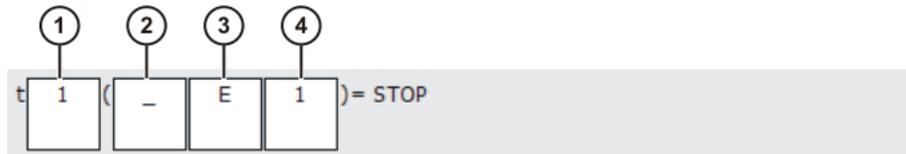


Fig. 9-21: Inline form "t=STOP"

Item	Description
1	Number of the timer <ul style="list-style-type: none"> ■ 1 ... 30
2	Operator. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
3	Boolean operand (condition) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)

Example (>>> Fig. 9-21)

Timer 1 is stopped if the following condition is met: input 1 is TRUE.

9.4.14 Programming an arithmetic comparison

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands** > **SPS ==>** > **Compare**.
3. Set the parameters in the inline form.
 (>>> 9.4.15 "Inline form "Compare" Page 207)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.15 Inline form "Compare"

Description The instruction can be used to program an arithmetic comparison. The result of the comparison is stored in a flag and can be polled.

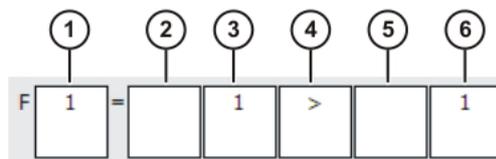


Fig. 9-22: Inline form "Compare"

Item	Description
1	Number of the flag <ul style="list-style-type: none"> 1 ... 999
2	1st arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
3	Number of the operand <ul style="list-style-type: none"> Range of values: Dependent on the 1st operand (>>> 9.4.2 "Arithmetic operands" Page 201)
4	Relational operator <ul style="list-style-type: none"> >, <, =, !, >=, <=
5	2nd arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
6	Number of the operand <ul style="list-style-type: none"> Range of values: Dependent on the 2nd operand (>>> 9.4.2 "Arithmetic operands" Page 201)

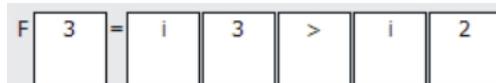
Example

Fig. 9-23: Inline form "Compare" (example)

Flag 3 is set if the following condition is met: integer counter 3 is greater than integer counter 2.

9.4.16 Setting a pulse output**Description**

Pulse outputs are used, for example, in stud welding, to feed the next stud while the robot is moving.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.



The instruction cannot be used in the **MakroSPS** macro.

Procedure

- In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
- Select the softkey **Commands** > **SPS ==>** > **Pulse**.
- Set the parameters in the inline form.
 (>>> 9.4.17 "Inline form "Pulse"" Page 209)
- Save the instruction by pressing the **Cmd Ok** softkey.

9.4.17 Inline form "Pulse"

Description The instruction sets a pulse of a defined length while the robot is moving. The pulse is held at a defined level or logic state for the specified length of time.

The instruction for setting the pulse is not executed if the corresponding level is already set. Edge changes are not permissible.

The pulse can be set synchronously or asynchronously to the robot motion and retriggered.

Fig. 9-24: Inline form "Pulse"

Item	Description
1	Number of the pulse output <ul style="list-style-type: none"> 1 ... 4,096
2	Operator. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> _, !
3	Boolean operand (level) <ul style="list-style-type: none"> ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	Length of the pulse <ul style="list-style-type: none"> 1 ... 300 [1/10 s]

9.4.18 Programming a FB ONL motion condition

Description Motion conditions are used to be able to stop robots under specific circumstances, e.g. in the event of faults or interlocks.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

- Position the cursor in any line in the Point PLC.
- Select the softkey **Commands > Wait/FB > FB ONL**.
- Set the parameters in the inline form.
 (>>> 9.4.19 "Inline form "FB ONL"" Page 210)
- Save the instruction by pressing the **Cmd Ok** softkey.

The FB ONL motion condition is inserted at the start of the Point PLC without a line number and is activated asynchronously in relation to the PLC trigger on the way to the end point.

9.4.19 Inline form "FB ONL"

Description

The instruction can be used to program a condition (Boolean operand) for robot motion. If required, several operands (maximum 11) can be linked. The robot is stopped if the robot motion condition is no longer met.

The FB ONL motion condition is activated asynchronously in relation to the PLC trigger on the way to the end point. It remains active until a new FB ONL or FB PSPS motion condition is programmed.

(>>> 9.4.21 "Inline form "FB PSPS"" Page 211)



The FB ONL motion condition is reset at the end of the program.



Fig. 9-25: Inline form "FB ONL"

Item	Description
1	Operator <ul style="list-style-type: none"> ■ <code>_ , (</code>
2	Operator The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ <code>_ , !</code>
3	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	Operator <ul style="list-style-type: none"> ■ <code>_ ,)</code>

Example

(>>> Fig. 9-25)

The robot moves as long as the following condition is met: input 100 is TRUE. The robot is stopped if the input is FALSE.

9.4.20 Programming a FB PSPS motion condition

Description

Motion conditions are used to be able to stop robots under specific circumstances, e.g. in the event of faults or interlocks.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands > Wait/FB > FB PSPS**.

3. Set the parameters in the inline form.
 - (>>> 9.4.21 "Inline form "FB PSPS"" Page 211)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.21 Inline form "FB PSPS"

Description

The instruction can be used to program a condition (Boolean operand) for robot motion. If required, several operands (maximum 11) can be linked. The robot is stopped if the robot motion condition is no longer met.

The FB PSPS motion condition is activated either at the end point, or before the end point at a time defined with the PLC trigger. It remains active until a new FB PSPS or FB ONL motion condition is programmed.

The FB PSPS motion condition overwrites a previously programmed FB ONL motion condition.

(>>> 9.4.19 "Inline form "FB ONL"" Page 210)

Fig. 9-26: Inline form "FB PSPS"

Item	Description
1	Operator <ul style="list-style-type: none"> ■ $_ , ($
2	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ $_ , !$
3	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	Operator <ul style="list-style-type: none"> ■ $_ ,)$

Example

(>>> Fig. 9-26)

The robot moves as long as the following condition is met: input 200 is TRUE. The robot is stopped if the input is FALSE.

9.4.22 Programming a signal-dependent wait function

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands** > **Wait/FB** > **Wait onl/bis**.

3. Set the parameters in the inline form.
 - (>>> 9.4.23 "Inline form "WARTE ONL/bis"" Page 212)
4. Save the instruction by pressing the **Cmd Ok** softkey.

The **WAIT ONL** instruction is inserted at the start of the Point PLC without a line number and is executed asynchronously in relation to the PLC trigger on the way to the end point. The **WAIT bis** instruction is inserted into the Point PLC with a line number and executed at the end point.

9.4.23 Inline form "WARTE ONL/bis"

Description

The instruction is used to program a signal-dependent wait function. If required, several signals or operands (maximum 11) can be linked.

If the programmed condition is not fulfilled, the robot is stopped at the end point. The robot motion is resumed as soon as the condition is met.

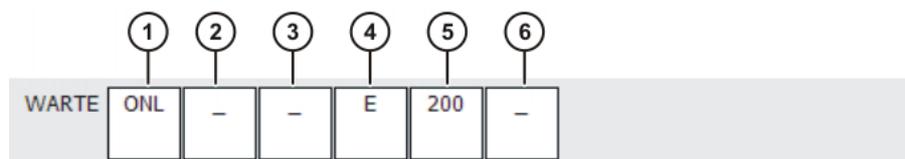


Fig. 9-27: Inline form "WARTE ONL/bis"

Item	Description
1	Type of wait function <ul style="list-style-type: none"> ■ ONL: The programmed condition is checked in the advance run: If the condition is met, the robot motion is resumed without stopping at the end point. If the condition is not fulfilled, the robot stops at the end point. ■ to: The robot is stopped at the end point of the motion. The programmed condition is checked there. <p>Note: If the wait function calls for the robot to stop at the end point, the motion cannot be approximated.</p>
2	Operator <ul style="list-style-type: none"> ■ _, (
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ _, !
4	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> ■ _,)

Example 1

Wait function WAIT ONL

WARTE	ONL	-	-	E	1	-
-------	-----	---	---	---	---	---

Fig. 9-28: Inline form "WARTE ONL" (example)

If the check in the advance run reveals that the condition "input 1 is TRUE" is met, the wait function is not executed. The wait function is also ignored if the input is set to FALSE on the way to the end point.

If the check in the advance run reveals that the condition "input 1 is TRUE" is not met, the robot stops at the end point and waits until the input is TRUE.

Example 2

Wait function WAIT bis

WARTE	bis	-	-	E	1	-
-------	-----	---	---	---	---	---

Fig. 9-29: Inline form "WARTE bis" (example)

The robot stops at the end point and waits until the following condition is met: input 1 is TRUE.

9.4.24 Programming a wait time

- Precondition**
- Program is selected.
 - Point PLC is open.
 - Operating mode T1 or T2.

- Procedure**
1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
 2. Select the softkey **Commands > Wait/FB > Wait Time**.
 3. Set the parameters in the inline form.
 - (>>> 9.4.25 "Inline form "WARTE ZEIT"" Page 213)
 4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.25 Inline form "WARTE ZEIT"

Description The instruction can be used to program a wait time and link it to a condition. If required, several arithmetic operands (maximum 11) can be linked to the definition of the wait time.

The robot motion is stopped for the programmed wait time if the condition is met. If the condition is no longer met, the wait time is canceled. In this case, the robot motion is resumed even if the programmed wait time has not yet elapsed.

WARTE	(-	-	-	E	1	-)	ZEIT	-	1

[1/10Sek]

Fig. 9-30: Inline form "WARTE ZEIT"

Item	Description
1	Operator <ul style="list-style-type: none"> ■ -, !
2	Operator <ul style="list-style-type: none"> ■ -, (
3	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
4	Boolean operand (condition) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> ■ -,)
7	Arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
8	Wait time in 1/10 s <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)

Example

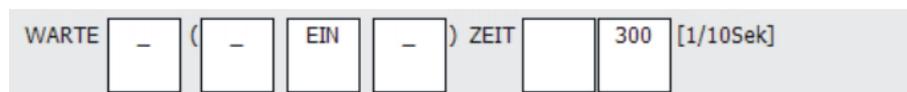


Fig. 9-31: Inline form "WARTE ZEIT" (example)

The robot is stopped for 30 s. The wait time is not dependent on a condition.

9.4.26 Switching interlocking on/off

Description Interlocking is used to avoid collisions in the case of overlapping workspaces.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands > Wait/FB > Interlocking**.
3. Set the parameters in the inline form.
(>>> 9.4.27 "Inline form "VERR."" Page 215)
4. Save the instruction by pressing the **Cmd Ok** softkey.

The instruction for switching on interlocking is inserted at the start of the Point PLC without a line number and is executed asynchronously in relation to the PLC trigger on the way to the end point. The instruction for switching off interlocking is inserted into the Point PLC with a line number. It is executed either at the end point, or before the end point at a time defined with the PLC trigger.

9.4.27 Inline form "VERR."

Description

The instruction can be used to switch the interlocking on and off.

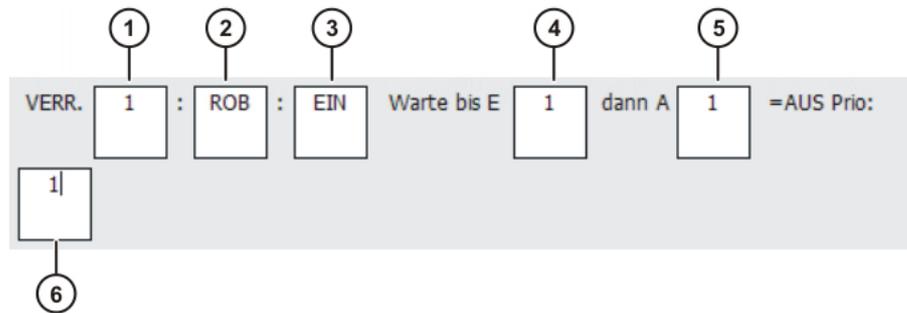


Fig. 9-32: Inline form "VERR."

Item	Description
1	Number of the interlock area, i.e. of the workspace shared by 2 robots <ul style="list-style-type: none"> 1 ... 8
2	Name of robot <ul style="list-style-type: none"> Freely selectable; max. 8 characters
3	Switches interlocking on/off. <ul style="list-style-type: none"> EIN (ON): Interlocking is switched on. AUS (OFF): Interlocking is switched off.
4	Number of the input used to transfer a signal from the other robot The input box is only available if EIN is selected. <ul style="list-style-type: none"> 1 ... 4,096
5	Number of the output used to transfer a signal to the other robot The input box is only available if EIN is selected. <ul style="list-style-type: none"> 1 ... 4,096
6	Priority of the robot <ul style="list-style-type: none"> 1: for the robot that is to have precedence 2: for the robot that is to be interlocked while the other robot is moving in the interlock area The input box is only available if EIN is selected.



The same interlock numbers must not be nested.

Example

The workspaces of the robots Rob 1 and Rob 2 overlap in interlock area 1. To prevent the robots simultaneously moving into the shared workspace, the interlocking must be activated in both robots for area 1.

For this purpose, an input and an output must be defined for each robot; these are used for exchanging signals between the robot controllers. In the example, these are input 1 and output 2 for Rob 2.



Fig. 9-33: Inline form “VERR.” (example)

If input 1 of Rob 2 is TRUE, Rob 2 is interlocked, as this robot has been assigned priority 2. At the same time, the value FALSE is transferred via output 1 to Rob 1.

9.4.28 Switching an Interbus segment or Interbus device on/off

Description Every tool is configured in such a way that it is connected to the controller via predefined Interbus segments or Interbus devices. Before a tool is changed, the associated segment or device must be switched off to prevent bus errors.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
2. Select the menu softkey **Commands > Wait/FB > I-Bus alternative**.
3. Set the parameters in the inline form.
(>>> 9.4.29 "Inline form “IBG”" Page 216)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.29 Inline form “IBG”

Description The instruction can be used to switch an Interbus segment or an Interbus device on and off.

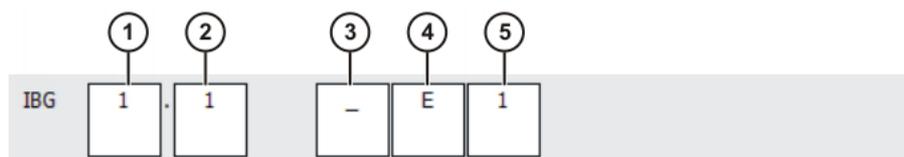


Fig. 9-34: Inline form “IBG”

Item	Description
1	Number of the Interbus segment <ul style="list-style-type: none"> ■ 1 ... 256
2	Number of the Interbus device <ul style="list-style-type: none"> ■ 1 ... 512
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !

Item	Description
4	Boolean operand <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S ■ EIN (ON): switch on ■ AUS (OFF): switch off
5	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)

9.4.30 Setting an analog output

Description Setting an analog output allows analog voltages to be output via the robot controller.

Depending on the specific application, the following analog outputs can be set:

- **ana=KONST**: Analog output for a static voltage, e.g. for adhesive dispensing
- **ana=Vprop**: Analog output for a velocity-proportional voltage, e.g. for parameterization during arc welding
- **ana=KST+P**: Analog output for a voltage that is dependent on the position in a weave motion, e.g. for controlling pressure stages of spot-welding guns

The robot controller can manage 16 analog inputs and 16 analog outputs. The inputs/outputs are implemented in the control PC by means of KUKA field bus cards. 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.

- Precondition**
- Program is selected.
 - Point PLC is open.
 - Operating mode T1 or T2.

- Procedure**
1. In the Point PLC, position the cursor in the line after which the instruction is to be inserted.
 2. Select the softkey **Commands > ANA/BS/Weave > ana const:** or **ana vprop:** or **ana kst+p.**
 3. Set the parameters in the inline form.
 - (>>> 9.4.31 "Inline form "ana=KONST"" Page 217)
 - (>>> 9.4.32 "Inline form "ana=Vprop"" Page 218)
 - (>>> 9.4.33 "Inline form "ana=KST+P"" Page 221)
 4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.31 Inline form "ana=KONST"

Description The instruction sets an analog output for a static voltage. The setting of the output can be linked to a condition (Boolean operand). If required, several arithmetic operands (maximum 11) can be linked to the determination of the voltage.

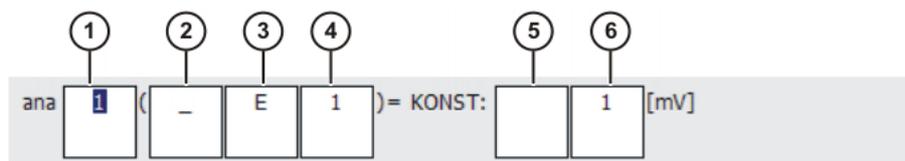


Fig. 9-35: Inline form “ana=KONST”

Item	Description
1	Number of the analog output <ul style="list-style-type: none"> 1 ... 16
2	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> -, !
3	Boolean operand (condition) <ul style="list-style-type: none"> ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	Arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
6	Voltage that is output at the analog output <ul style="list-style-type: none"> -10,000 ... +10,000 mV

Example

Fig. 9-36: Inline form “ana=KONST” (example)

10,000 mV are output at analog output 3 if the following condition is met: Merker (cyclical flag) 21 is FALSE.

9.4.32 Inline form “ana=Vprop”**Description**

The instruction sets an analog output for a dynamic voltage. The setting of the output can be linked to a condition (Boolean operand). If required, several arithmetic operands (maximum 11) can be linked to the determination of the voltage.

The output voltage varies dynamically and is dependent on the following values:

- Velocity
- Offset



A velocity-proportional analog motion is only possible in the case of CP motions (LIN, KLIN, CIR, KCIR).

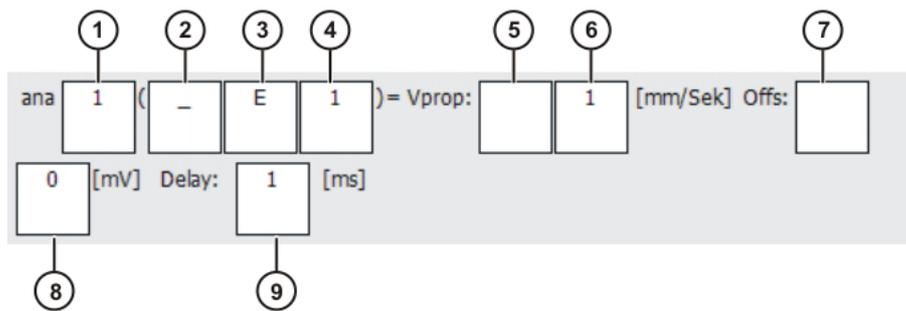


Fig. 9-37: Inline form "ana=Vprop"

Item	Description
1	Number of the analog output. The Delay softkey, which can be used to open the input box of the same name (9), is only available if the cursor is located in this input box. <ul style="list-style-type: none"> ■ 1 ... 4
2	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
3	Boolean operand <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	1st arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
6	Velocity at which the output voltage would reach the maximum value of 10,000 mV with offset = 0. <ul style="list-style-type: none"> ■ -9,999 ... +9,999 mm/s
7	2nd arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
8	Offset: Value by which the output voltage is increased or decreased <ul style="list-style-type: none"> ■ -10,000 ... +10,000 mV
9	Time by which the output signal is delayed (+) or brought forward (-). The input box is only available if the softkey Delay has been pressed. <ul style="list-style-type: none"> ■ -500 ... +500 ms



Analog outputs with a delay activated cannot be approximated.

Example 1

In this example, the maximum possible voltage of 10,000 mV is output at a velocity of 500 mm/s.

ana 4 (! E 16) = Vprop: 500 [mm/Sek] Offs:
 0 [mV]

Fig. 9-38: Inline form “ana=Vprop” – analog output without offset

The analog output is activated if the following condition is met: input 16 is FALSE.

This analog output is active until it is deactivated by a static analog output (ana=KONST or ana=KST+P).

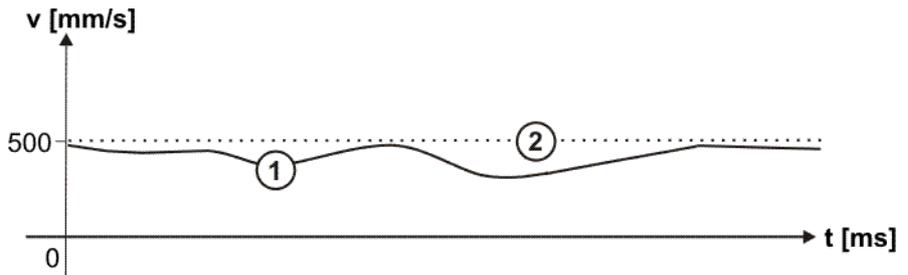


Fig. 9-39: Diagram of analog output without offset

- 1 Current robot velocity
- 2 Robot velocity at which the maximum voltage is output

Example 2

In this example, an analog voltage of 5,000 mV is required for optimal adhesive application with a robot velocity of 400 mm/s. Tests show that the adhesive controller only reaches an internal analog value of 0 V with an offset of -70 mV.

ana 4 (! E 16) = Vprop: 800 [mm/Sek] Offs:
 -70 [mV]

Fig. 9-40: Inline form “ana=Vprop” – analog output with offset

The required 5,000 mV are output at the analog output at a robot velocity of 400 mm/s. To reach the maximum possible voltage of 10,000 mV, the robot would thus have to be moved at a velocity of 800 mm/s. This velocity must be entered in the inline form.

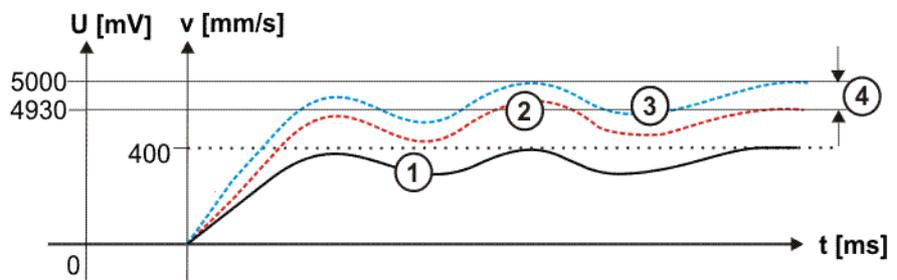


Fig. 9-41: Diagram of analog output with offset

- 1 Current robot velocity
- 2 Voltage at the analog output: 4,930 mV

- 3 Required voltage at the analog output: 5,000 mV
- 4 Offset: -70 mV

Example 3

ana () = Vprop: [mm/Sek] Offs: [mV]

Delay: [ms]

Fig. 9-42: Inline form "Ana vprop": (analog output with delay)

In this example, the adhesive controller starts 85 ms before the robot with an offset of -70 mV.

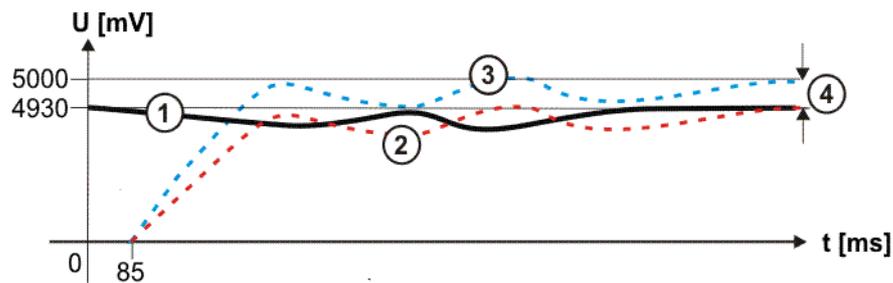


Fig. 9-43: Diagram of analog output with delay

- 1 Dispensing controller
- 2 Voltage at the analog output: 4,930 mV
- 3 Required voltage at the analog output: 5,000 mV
- 4 Offset: -70 mV

9.4.33 Inline form "ana=KST+P"

Description

The instruction is used if the CP motion of the robot is superposed with a weave motion. The instruction can be used to set an analog output for the voltage that is dependent on the position on the weave amplitude during weaving.

The setting of the output can be linked to a condition (Boolean operand). If required, several arithmetic operands (maximum 11) can be linked to the determination of the voltage.

ana () = KST+P: [mV] Pendel:

[mV]

Fig. 9-44: Inline form "ana=KST+P"

Item	Description
1	Number of the analog output <ul style="list-style-type: none"> 1 ... 4
2	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> -, !
3	Boolean operand (condition) <ul style="list-style-type: none"> ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
4	Number of the operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
5	1st arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
6	Static voltage superposed with a weave pattern <ul style="list-style-type: none"> -10,000 ... +10,000 mV
7	2nd arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
8	Amplitude of the voltage <ul style="list-style-type: none"> -10,000 ... +10,000 mV

Example

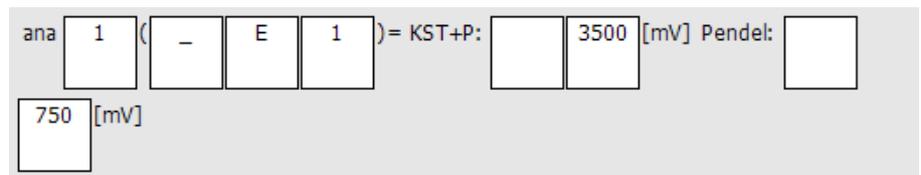


Fig. 9-45: Inline form “ana=KST+P” (example)

The analog output is activated if the following condition is met: input 1 is TRUE.

The output voltage is between 2,750 mV and 4,250 mV and depends on the current position in the weave pattern.

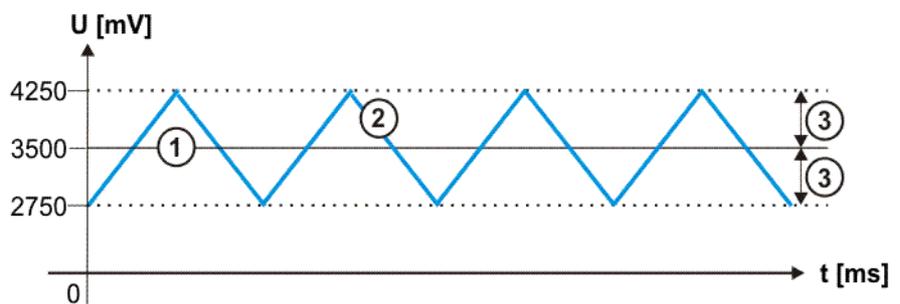


Fig. 9-46: Diagram of analog output with the “Triangle” weave pattern

- 1 Static voltage
- 2 Voltage at the analog output: “Triangle” weave pattern
- 3 Amplitude of the voltage or weave pattern

9.4.34 Activating/deactivating weaving

Description In mechanical weaving, a weave motion is superposed onto the path motion, e.g. in order to weld a weave seam.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

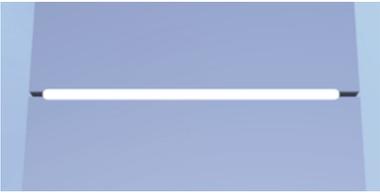
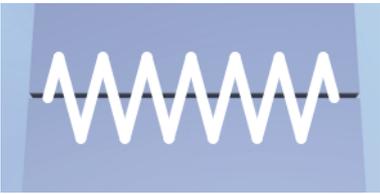
1. Position the cursor in any line in the Point PLC.
2. Select the softkey **Commands** > **ANA/BS/Weave** > **Weave**.
3. Set the parameters in the inline form.
(>>> 9.4.36 "Inline form "Weave"" Page 224)
4. Save the instruction by pressing the **Cmd Ok** softkey.

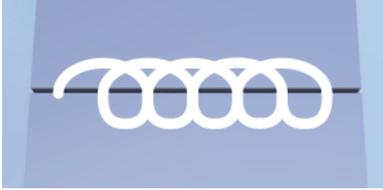
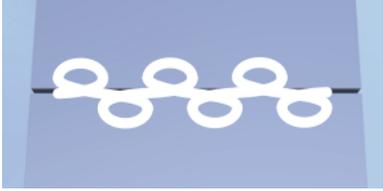
The weave instruction is inserted without a line number at the end of the Point PLC and is not carried out until all numbered PLC instructions have been executed.

9.4.35 Weave patterns

There are 7 weave patterns, of which 5 are predefined and 2 can be defined freely.

- The weave pattern is constantly repeated when weaving is carried out.
- The shape of the weave pattern is dependent on the weld velocity. The higher the weld velocity, the more the weave pattern will be approximated.
- The shape of the weave pattern is also dependent on the values defined by the user for weave length and amplitude.

Name	Weave pattern
No weaving	 <p data-bbox="794 1402 1190 1440">No deflection of the welding torch</p>
Triangle	 <p data-bbox="794 1659 1313 1697">Deflection of the welding torch in 1 direction</p>
Trapezoid	 <p data-bbox="794 1917 1313 1953">Deflection of the welding torch in 1 direction</p>

Name	Weave pattern
Asymmetric trapezoid	 <p data-bbox="906 421 1426 454">Deflection of the welding torch in 1 direction</p>
Spiral	 <p data-bbox="906 683 1394 739">Deflection of the welding torch in 2 directions</p>
Double 8	 <p data-bbox="906 974 1394 1030">Deflection of the welding torch in 2 directions</p>
UserDef1	<p data-bbox="906 1048 1276 1081">Freely-definable weave pattern</p> <p data-bbox="906 1093 1385 1149">Deflection of the welding torch in up to 3 directions</p>
UserDef2	<p data-bbox="906 1167 1276 1200">Freely-definable weave pattern</p> <p data-bbox="906 1211 1385 1267">Deflection of the welding torch in up to 3 directions</p>

The following parameters are relevant for the weave motion:

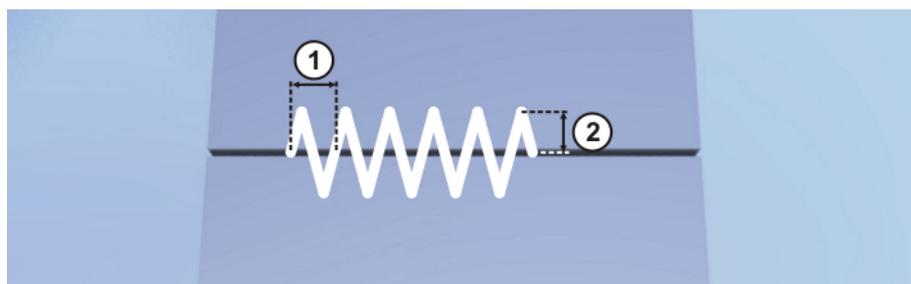


Fig. 9-47: Parameters of a weave pattern

- 1 Weave length (= 1 oscillation; length of the path from the start point to the end point of the pattern)
- 2 Amplitude

9.4.36 Inline form “Weave”

Description

The instruction can be used to superpose a weave motion onto the path motion.

Fig. 9-48: Inline form "Weave"

Item	Description
1	Activate/deactivate weaving. <ul style="list-style-type: none"> ■ EIN (ON): switch on ■ AUS (OFF): switch off
2	Weave pattern number <ul style="list-style-type: none"> ■ 1: Triangle ■ 2: Trapezoid ■ 3: Asymmetric trapezoid ■ 4: Spiral ■ 5: Double 8 ■ 6,7: Freely definable Precondition: User group "Expert"
3	Weave amplitude. Distance between the taught path and the maximum lateral deflection of the weave. <ul style="list-style-type: none"> ■ 1 ... 30 mm
4	Weave length. Distance between the start and end points of one weave pattern oscillation. <ul style="list-style-type: none"> ■ 1 ... 50 mm
5	Weave angle. Angle by which the weave plane is rotated. <ul style="list-style-type: none"> ■ -90 ... 0°

9.4.37 Programming a path-related switching action

Description

Path-related switching actions can be triggered relative to the end point of a motion block. They are executed parallel to the robot motion.

- If the end point is an exact positioning point, a switching action is triggered exactly at the end point.
- If the end point is approximated, a switching action is triggered in the middle of the approximate positioning arc.

The motion block can be a LIN, CIR, KLIN, KCIR 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.

A switching point can be shifted in time and space. It is then not triggered exactly at the end point or in the middle of the approximate positioning arc, but either brought forward or delayed. A switching action cannot be shifted freely.

The types of shift that are possible depend on whether the end point is an exact positioning point or an approximate positioning point.

4 types of motion block are possible:

Start point	End point
Exact positioning point Approximation distance: 0%	Exact positioning point Approximation distance: 0%
Exact positioning point Approximation distance: 0%	Approximate positioning point Approximation distance: 100%
Approximate positioning point Approximation distance: 100%	Approximate positioning point Approximation distance: 100%
Approximate positioning point Approximation distance: 100%	Exact positioning point Approximation distance: 0%

Switching range

End point is an exact positioning point:

In this case, the switching action cannot be shifted beyond the end point. It can only be triggered earlier, i.e. in the negative direction, towards the start point.

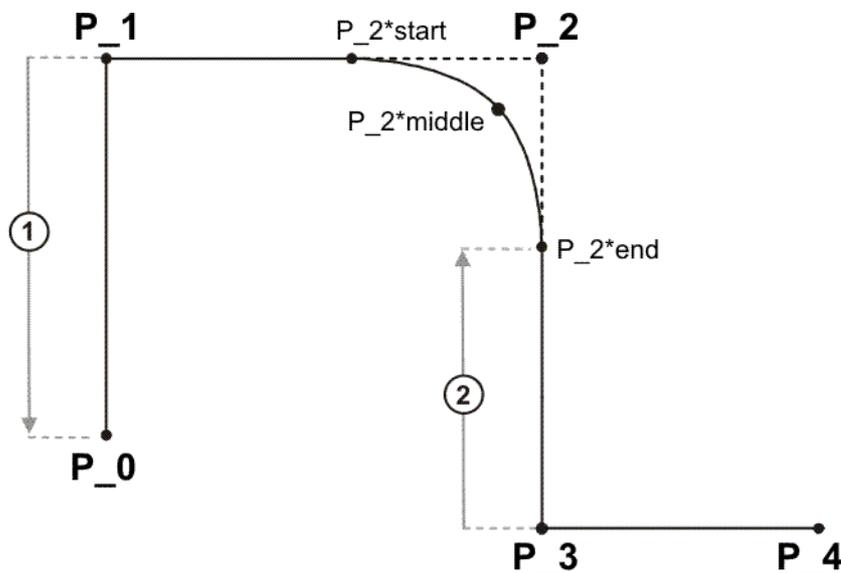


Fig. 9-49: End point is an exact positioning point

The switching limit of the shift depends on whether the start point is an exact positioning point or an approximate positioning point.

Item	Description
1	If the start point is an exact positioning point, the switching action can be shifted, at most, as far as the start point.
2	If the start point is approximated, the switching action can be shifted, at most, as far as the end of the approximate positioning arc.

End point is approximated:

In this case, switching action can be triggered earlier or with a delay, i.e. shifted in the negative direction towards the start point or in the positive direction towards the next end point.

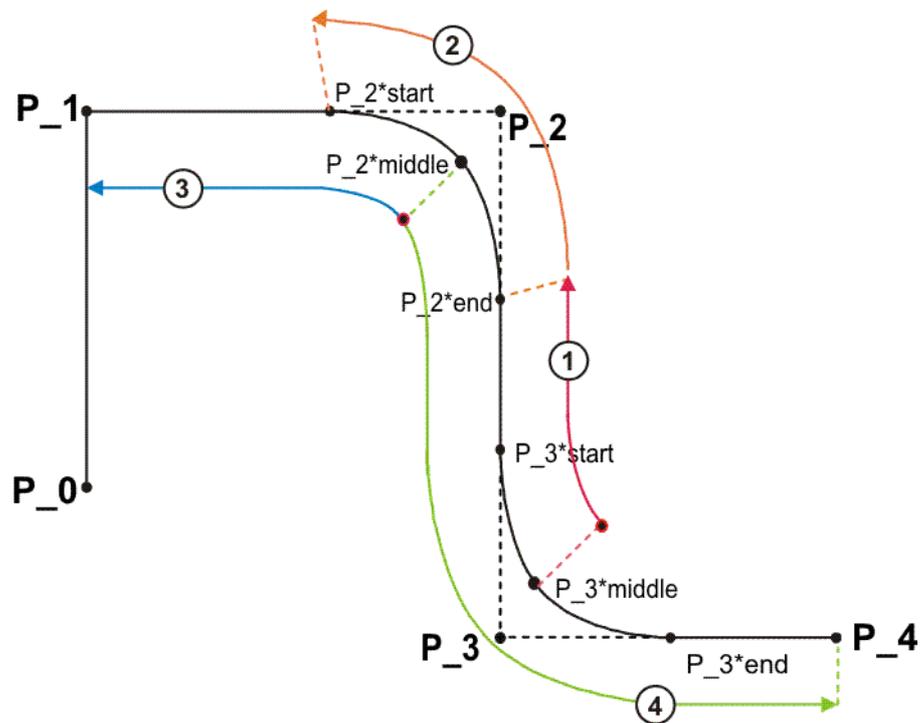


Fig. 9-50: End point is approximated

The switching limit of the shift depends on whether the start point or next end point is an exact positioning point or an approximate positioning point.

Item	Description
1	Shift towards the start point: If the start point is an approximated LIN or CIR point, the switching action can be shifted, at most, as far as the start of its approximate positioning arc.
1 ... 2	Shift towards the start point: If the start point is an approximated PTP point, the switching action can be shifted, at most, as far as the end of its approximate positioning arc.
3	Shift towards the start point: If the start point is an exact positioning point, the switching action can be shifted, at most, as far as the start point.
4	Shift towards the next end point: The switching action can be shifted, at most, as far as the next exact positioning point (skipping all approximate positioning points). The switching range here is P_2*middle to P_4. If point P_3 were not approximated, the switching range would be P_2*middle to P_3.



If the values specified for the shift in space or time are too great, the controller automatically switches at the switching limit.

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. Position the cursor in any line in the Point PLC.

2. Select the softkey **Commands > ANA/BS/Weave > BS A/F=** or **BS bin/ana=**.
3. Set the parameters in the inline form.
 - (>>> 9.4.38 "Inline form "BS A/F"" Page 228)
 - (>>> 9.4.39 "Inline form "BS bin/ana"" Page 229)
4. Save the instruction by pressing the **Cmd Ok** softkey.

A path-related switching action is inserted at the start of the Point PLC without a line number and is executed asynchronously in relation to the PLC trigger on the way to the end point.

9.4.38 Inline form "BS A/F"

Description

The instruction can be used to set an output or flag and link the setting of the signal to a condition. If the condition is met, the value of a Boolean operand can be assigned to the output or flag and the defined switching point is triggered. If required, several Boolean operands (maximum 11) can be linked to the value assignment.

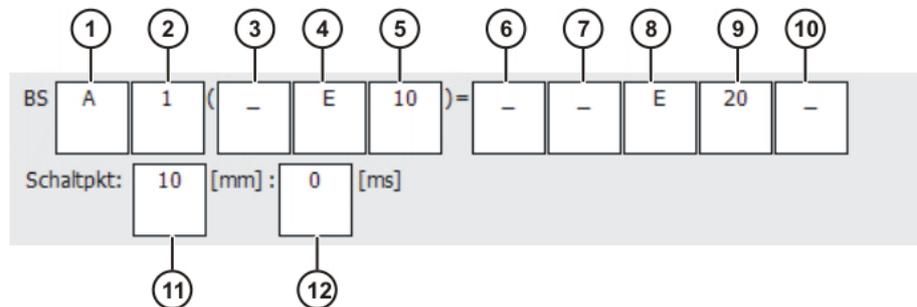


Fig. 9-51: Inline form "BS A/F"

Item	Description
1	Signal <ul style="list-style-type: none"> ■ A: output ■ F: flag
2	Number of the output or flag <ul style="list-style-type: none"> ■ A: 1 ... 4,096 ■ F: 1 ... 999
3	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
4	1st Boolean operand (condition) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the 1st operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> ■ -, (
7	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !

Item	Description
8	2nd Boolean operand (value assignment) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
9	Number of the 2nd operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
10	Operator <ul style="list-style-type: none"> ■ _,)
11	Distance from the switching point to the end point <ul style="list-style-type: none"> ■ -9,999...+9,999 mm <p>Positive value: shifts the statement towards the end of the motion. Negative value: shifts the statement towards the start of the motion.</p> <p>Note: In the case of a PTP motion, the value 0 must be entered. The switching point must be situated at the taught end point.</p>
12	Switching action delay <ul style="list-style-type: none"> ■ -9,999 ... +9,999 ms <p>Positive value: shifts the statement towards the end of the motion. Negative value: shifts the statement towards the start of the motion.</p> <p>Note: The time specification is absolute. The switching point varies according to the velocity of the robot.</p>

Example

The screenshot shows the inline form "BS A/F" with the following fields and values:

- BS: A
- Output: 10
- Condition: EIN
- Operator: =
- Switching point: -
- Distance: -
- Delay: E
- Value: 5
- Label: Schaltpkt:
- Distance input: -150 [mm]
- Delay input: -100 [ms]

Fig. 9-52: Inline form "BS A/F" (example)

The value from input 5 is assigned to output 10. The switching action is triggered 100 ms before a distance of 150 mm from the end point is reached.

9.4.39 Inline form "BS bin/ana"

Description

The instruction can be used to set a binary or analog output and link the setting of the signal to a condition. If the condition is met, a value can be set at the output at a defined switching point. If required, several arithmetic operands (maximum 11) can be linked to the determination of the value.

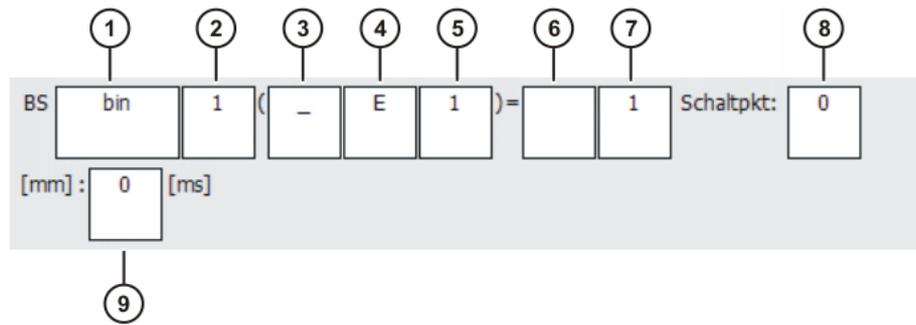


Fig. 9-53: Inline form “BS bin/ana”

Item	Description
1	Signal <ul style="list-style-type: none"> ■ bin: A binary value is set at the output. ■ ana: The output voltage is constant. ■ ana_vprop: The output voltage is dependent on the velocity. ■ ana_offs: The output voltage is dependent on the offset.
2	Number of the output <ul style="list-style-type: none"> ■ bin: 1 ... 20 ■ ana, ana_vprop, ana_offs: 1 ... 16
3	Operator. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
4	Boolean operand (condition) <ul style="list-style-type: none"> ■ ?, EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the Boolean operands ?, EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
7	Value assignment <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)

Item	Description
8	Distance from the switching point to the end point ■ -9,999...+9,999 mm Positive value: shifts the statement towards the end of the motion. Negative value: shifts the statement towards the start of the motion. Note: In the case of a PTP motion, the value 0 must be entered. The switching point must be situated at the taught end point.
9	Switching action delay ■ -9,999 ... +9,999 ms Positive value: shifts the statement towards the end of the motion. Negative value: shifts the statement towards the start of the motion. Note: The time specification is absolute. The switching point varies according to the velocity of the robot.

Example

The screenshot shows the inline form for 'BS bin/ana'. The form is structured as follows:

BS [ana] [3] ([] [] [] [] [] []) = [] [250] [mV] Schaltpkt:

[120] [mm] : [] [-80] [ms]

Fig. 9-54: Inline form "BS bin/ana" (example)

A voltage of 250 mV is output at analog output 3 if the following condition is met: input 1 is TRUE. The switching action is triggered 80 ms before a distance of 120 mm after the end point is reached.

9.4.40 Modifying a switching point

Precondition ■ Program is selected.

Procedure

1. Select the program run mode **GO**.
2. Execute the program with reduced override until the position is reached at which the switching action is to be triggered.
3. Go to the corresponding box in the inline form.
4. Enter the distance from the switching point to the end point.
5. Press the **Teach** softkey. The switching point is updated and the inline form is closed.

9.4.41 Calling a macro

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

1. In the Point PLC, position the cursor in the line after which the call is to be inserted.
2. Select the softkey **Commands > MAKRO/UP/Zange > MAKRO**.
3. Set the parameters in the inline form.
(>>> 9.4.42 "Inline form "SPSMAKRO"" Page 232)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.42 Inline form "SPSMAKRO"

Description

The instruction can be used to call a macro of type **MakroSPS** and link the macro call to a condition (Boolean operand). If required, several operands (maximum 11) can be linked.

(>>> 9.6.3 "MakroSPS" Page 240)



If a macro contains instructions that trigger an advance run stop, it cannot be approximated.

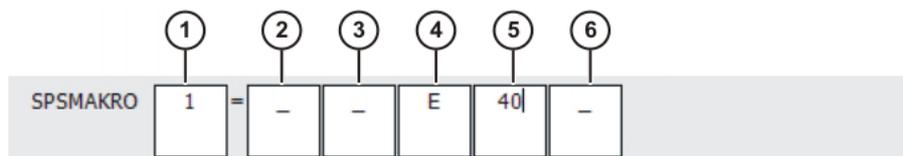


Fig. 9-55: Inline form "SPSMAKRO"

Item	Description
1	Number of the macro <ul style="list-style-type: none"> 0 ... 999
2	Operator <ul style="list-style-type: none"> _, (
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> _, !
4	Boolean operand (condition) <ul style="list-style-type: none"> EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> _,)

Example

(>>> Fig. 9-55)

The macro MakroSPS 1 is started if the following condition is met: input 40 is TRUE.

9.4.43 Calling a subprogram

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

- Position the cursor in any line in the Point PLC.
- Select the softkey **Commands > MAKRO/UP/Zange > UP**.
- Set the parameters in the inline form.
 (>>> 9.4.44 "Inline form "UP"" Page 233)
- Save the instruction by pressing the **Cmd Ok** softkey.

The subprogram call is inserted without a line number at the end of the Point PLC and is not carried out until all numbered PLC instructions have been executed.

9.4.44 Inline form "UP"

Description The instruction can be used to call a subprogram and link the subprogram call to a condition (Boolean operand). If required, several operands (maximum 11) can be linked.

There is no limit on the number of times that subprograms can be called.

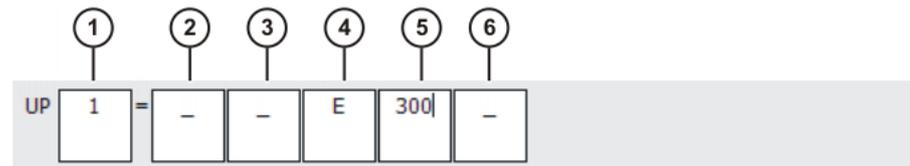


Fig. 9-56: Inline form "UP"

Item	Description
1	Subprogram number <ul style="list-style-type: none"> 0 ... 999
2	Operator <ul style="list-style-type: none"> _, (
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> _, !
4	Boolean operand (condition) <ul style="list-style-type: none"> EIN (ON), AUS (OFF), E, A, M, F, T, S
5	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values. Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> _,)

Example (>>> Fig. 9-56)

Subprogram 1 is started if the following condition is met: input 300 is TRUE.

9.4.45 Programming program loops

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.

Procedure

- In the Point PLC, position the cursor in the line after which the call is to be inserted.
- Select the softkey **Commands** > **MAKRO/UP/Zange** > **MAKRO/UP-Loop**.
- Set the parameters in the inline form.
 (>>> 9.4.46 "Inline form "REPEAT MAKRO/UP"" Page 234)
- Save the instruction by pressing the **Cmd Ok** softkey.

Subprogram loops are inserted without a line number at the end of the Point PLC and are not carried out until all numbered PLC instructions have been executed. MAKRO loops are inserted into the Point PLC with a line number. They are executed either at the end point, or before the end point at a time defined with the PLC trigger.

9.4.46 Inline form “REPEAT MAKRO/UP”

Description

The instruction can be used to call a subprogram or macro repeatedly. Before each new call, the system checks whether the termination condition defined in the inline form has been met. If so, the loop is terminated.

Fig. 9-57: Inline form “REPEAT MAKRO/UP”

Item	Description
1	Program type <ul style="list-style-type: none"> ■ UP: subprogram ■ MAKRO: macro
2	Number of the subprogram or macro <ul style="list-style-type: none"> ■ UP: 1 ... 999 ■ MAKRO: 0 ... 999
3	Arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
4	Maximum number of loops <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)
5	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
6	Boolean operand (termination condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
7	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)

Example

Fig. 9-58: Inline form “REPEAT UP” (example)

The subprogram is executed 5 to 12 times. The loop is terminated if the following termination condition is met: input 1 is TRUE.

9.4.47 Calling a gun function

- Precondition**
- Program is selected.
 - Point PLC is open.
 - Operating mode T1 or T2.

- Procedure**
1. In the Point PLC, position the cursor in the line after which the call is to be inserted.
 2. Select the softkey **Commands > MAKRO/UP/Zange > Zange =**.
 3. Set the parameters in the inline form.
 - (>>> 9.4.48 "Inline form "ZANGE"" Page 235)
 4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.48 Inline form "ZANGE"

Description The instruction can be used to open or close a gun and link these gun functions to a condition (Boolean operand). If required, several operands (maximum 11) can be linked.

Fig. 9-59: Inline form "ZANGE"

Item	Description
1	Number of the gun <ul style="list-style-type: none"> ■ 0 ... 16
2	Gun function <ul style="list-style-type: none"> ■ AUF: open gun. ■ ZU: close gun.
3	Operator <ul style="list-style-type: none"> ■ _, (
4	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ _, !
5	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
6	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
7	Operator <ul style="list-style-type: none"> ■ _,)

Example

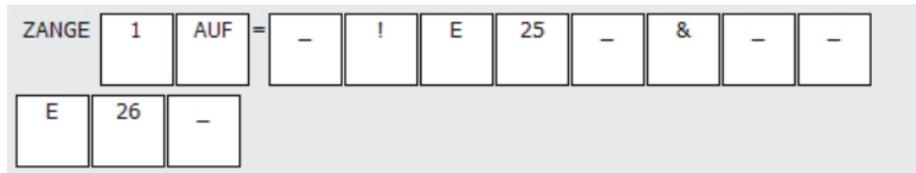


Fig. 9-60: Inline form “ZANGE” (example)

Gun 1 is opened if the following condition is met: input 25 is FALSE and input 26 is TRUE.

9.4.49 Calling VW_USER

Description

Expert users can program functions in KRL in the VW_USER module. For this purpose, several local subprograms are available, which can be used as required. The expert configures an inline form that enables the user to transfer parameters to the VW_USER module.

(>>> 11 "KRL function call with parameter transfer (=USER)" Page 307)

Precondition

- Program is selected.
- Point PLC is open.
- Operating mode T1 or T2.



VW_USER calls in macros are not permissible.

Procedure

1. In the Point PLC, position the cursor in the line after which the call is to be inserted.
2. Select the softkey **Commands > USER**.
3. Set the parameters in the inline form.
(>>> 9.4.50 "Inline form “VW User”" Page 236)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.4.50 Inline form “VW User”

Description

The instruction can be used to transfer up to 7 parameters to the VW_User module.



The structure of the inline form depends on the configuration.
(>>> 11.3 "Configuring the inline form “VW User”" Page 308)

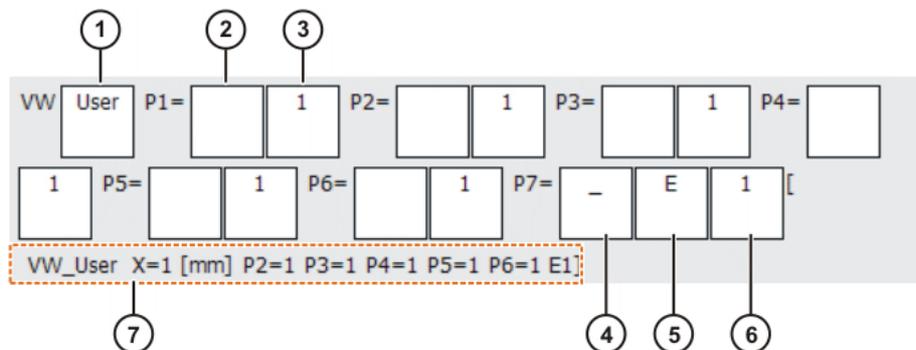


Fig. 9-61: Inline form “VW User”

Item	Description
1	Name of the KRL program <ul style="list-style-type: none"> ■ Range of values: Dependent on the configuration
2	Arithmetic operand <ul style="list-style-type: none"> ■ num, i, bin, t, ana, anain, binin, p
3	Value of the parameter that is transferred <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)
4	Operator. The input box is not available if the Boolean operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ -, !
5	Boolean operand <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
6	Number of the operand. The input box is not available if the Boolean operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
7	Comment <p>The comment is automatically completed from the configuration and appears as program text in the Point PLC.</p>

9.5 Programming jump labels

Description Labels can be set in programs. It is possible to jump to these labels from other positions within the program. Loops are programmed in this way, for example. Jump labels can be used between motion blocks and in the Point PLC of a motion block.

Precondition

- Program has been opened.
- Point PLC has been opened (only if using jump labels in the Point PLC).
- Operating mode T1 or T2

Procedure

1. Position the cursor in the line containing the motion block or the line in the Point PLC after which the label is to be inserted.
2. Select the softkey **Commands > Jump/Label > Label**.
3. Set the parameters in the inline form.
(>>> 9.5.1 "Inline form "Label/SPSLabel"" Page 237)
4. Position the cursor in the line containing the motion block or the line in the Point PLC after which the jump is to be inserted.
5. Select the softkey **Commands > Jump/Label > Jump**.
6. Set the parameters in the inline form.
(>>> 9.5.2 "Inline form "GOTO Label/GOTO SPSLabel"" Page 238)
7. Save the instruction by pressing the **Cmd Ok** softkey.

9.5.1 Inline form "Label/SPSLabel"

Description The instruction can be used to insert a label into a Folge, subprogram or macro.



A label may only be used once.

When a label is inserted between 2 motion blocks, the following inline form is opened:

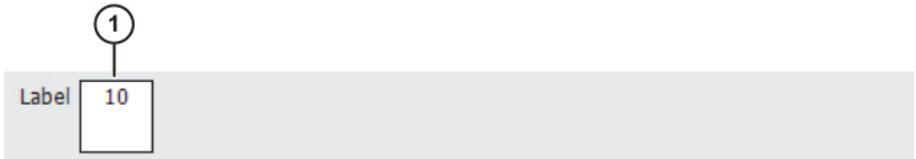


Fig. 9-62: Inline form “Label”

When a label is inserted into a Point PLC, the following inline form is opened:

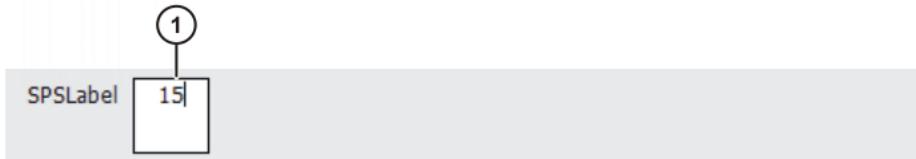


Fig. 9-63: Inline form “SPSLabel”

Item	Description
1	Number of the label ■ 1 ... 32,000

Example

(>>> 9.5.2 "Inline form “GOTO Label/GOTO SPSLabel”" Page 238)

9.5.2 Inline form “GOTO Label/GOTO SPSLabel”

Description

The jump instruction can be inserted into a Folge or subprogram and linked to a condition (Boolean operand).



Multiple jump instructions in a program can refer to the same label.

When a jump instruction is inserted between 2 motion blocks, the following inline form is opened:

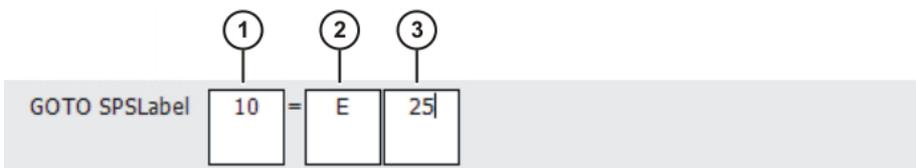


Fig. 9-64: Inline form “GOTO Label”

When a jump instruction is inserted into a Point PLC, the following inline form is opened:



Fig. 9-65: Inline form “GOTO SPSLabel”

Item	Description
1	Number of the label to which the jump is to be carried out. <ul style="list-style-type: none"> 1 ... 32,000
2	Boolean operand (condition) <ul style="list-style-type: none"> EIN (ON), AUS (OFF), E, A, M, F, T, S
3	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)

Example 1

Jump label between motion commands

```

1      1 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
2      Warte auf Folgenstart
3      2 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
4      LABEL 10
5      3 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
6      4 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
7      GOTO Label 10 = E 25
8      5 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
9      6 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]

```

Line	Description
4	Label 10
7	Jump to label 10 if the following condition is met: input 25 is TRUE.

Example 2

Jump label in a Point PLC

```

1      1 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]
2      Warte auf Folgenstart
3      2 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s] FP
4      FB ONL = EIN
5      1: SPSLABEL 5
6      2: SPSMAKRO99 = !F100
7      3: SPSMAKRO81 = EIN
8      4: SPSMAKRO82 = EIN
9      5: SPSMAKRO95 = EIN
10     6: GOTO SPSLabel 5 = E 16
11     7: bin1 (EIN) = 3
12     8: SPSMAKRO3 = EIN
13     9: SPSMAKRO31 = EIN
14     3 PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTRIG=0 [1/100s]

```

Line	Description
5	Label 5
10	Jump to label 5 if the following condition is met: input 16 is TRUE.

9.6 Programming instructions in macros**9.6.1 Macros in VSS****Overview**

This following files are located by default in the directory C:\KRC\Robot-er\KRC\R1\Makros:

- makrosaw.src (MakroSAW)
(>>> 9.6.2 "MakroSAW" Page 240)

- makrosp.src (MakroSPS)
(>>> 9.6.3 "MakroSPS" Page 240)
- makrostep.src (MakroStep)
(>>> 9.6.6 "MakroStep" Page 241)
- makrotrigger.src (MakroTrigger)
(>>> 9.6.9 "MakroTrigger" Page 243)

9.6.2 MakroSAW

Description

The **MakroSAW** macro is executed for every block selection, e.g. after the user has interrupted program execution to modify it.

The outputs that are reset in the event of a block selection, e.g. the interlock outputs to other robots, can be defined in this macro.

Outputs and flags can be set in the open macro. A maximum of 30 instructions is permissible.

9.6.3 MakroSPS

Description

The **MakroSPS** macro is polled cyclically during program execution. If an error occurs, defined actions can be triggered by means of the macro. For example, if the weld controller signals an error, an input is set. The macro then sets an output that the connected PLC evaluates.

The following PLC instructions can be used for programming in the open macro:

Statement	Description
A/M/F	(>>> 9.4.7 "Inline form "A/M/F"" Page 203) Note: Merker (cyclical flags) cannot be set.
i/bin	(>>> 9.4.9 "Inline form "i/bin"" Page 204)
t=	(>>> 9.4.11 "Inline form "t= (Start)"" Page 205)
t=STOP	(>>> 9.4.13 "Inline form "t=STOP"" Page 207)
Compare	(>>> 9.4.15 "Inline form "Compare"" Page 207)
Position Flag	(>>> 9.6.5 "Inline form "Position Flag"" Page 240)

9.6.4 Setting a position-dependent flag

Precondition

- MakroSPS is open.
- Operating mode T1 or T2

Procedure

1. Position the cursor in the line after which the instruction is to be inserted.
2. Select the softkey **Commands** > **SPS ==>** > **Position Flag**.
3. Set the parameters in the inline form.
(>>> 9.6.5 "Inline form "Position Flag"" Page 240)
4. Save the instruction by pressing the **Cmd Ok** softkey.

9.6.5 Inline form "Position Flag"

Description

The instruction sets a flag once an axis has reached a defined position.

Fig. 9-66: Inline form "Position Flag"

Item	Description
1	Number of the flag <ul style="list-style-type: none"> 1 ... 999
2	Axis <ul style="list-style-type: none"> AXIS_1 ... AXIS_6, EXAX_1 ... EXAX_6
3	Relational operator <ul style="list-style-type: none"> >, <, =, !, >=, <=
4	Arithmetic operand <ul style="list-style-type: none"> num, i, bin, t, ana, anain, binin, p
5	Axis angle in degrees <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.2 "Arithmetic operands" Page 201)

Example (>>> Fig. 9-66)

Flag 1 is set if the axis angle of axis A1 is greater than 20°.

9.6.6 MakroStep

Description Sequences can be programmed in the **MakroStep** macro. A step in a sequence is executed within a cycle of the Submit interpreter. The next step is executed in the following cycle, and so on until the complete sequence has been executed.

Precondition

- MakroStep is open.
- Operating mode T1 or T2

Procedure

- Position the cursor in the line after which the step is to be inserted.
- Select the softkey **Commands** > **Step / Transition** > **Step**.
- Enter the step number in the inline form.
 (>>> 9.6.7 "Inline form "SCHRITT"" Page 242)
- Save the instruction by pressing the **Cmd Ok** softkey.
- To program the step, open the Point PLC by pressing the **PLC Open** softkey.

The following PLC instructions can be used and opened by means of a softkey:

Softkey	Description
A/M/F	(>>> 9.4.7 "Inline form "A/M/F"" Page 203) Note: Merker (cyclical flags) cannot be set.
i/bin	(>>> 9.4.9 "Inline form "i/bin"" Page 204)
t=	(>>> 9.4.11 "Inline form "t= (Start)"" Page 205)
t=STOP	(>>> 9.4.13 "Inline form "t=STOP"" Page 207)
Compare	(>>> 9.4.15 "Inline form "Compare"" Page 207)

6. Press the **Transition** softkey to program the transition to the next step.
7. Set the parameters in the inline form.
 - (>>> 9.6.8 "Inline form "Schritt" (transition)" Page 242)
8. Save the instruction by pressing the **Cmd Ok** softkey.
9. Close the Point PLC by pressing the **PLC Close** softkey.

9.6.7 Inline form "SCHRITT"

Description The instruction inserts a step from a sequence into the **MakroStep** macro.

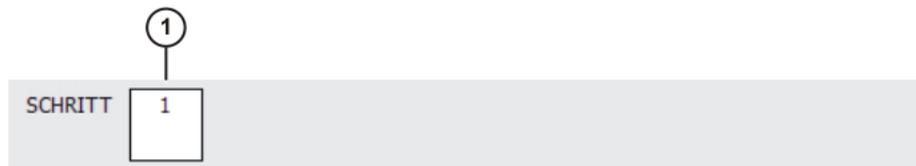


Fig. 9-67: Inline form "SCHRITT"

Item	Description
1	Number of the step. Steps with identical step numbers may be programmed. All steps with the same step number are executed within a cycle of the Submit interpreter. <ul style="list-style-type: none"> ■ 1 ... 32,000

Example (>>> 9.6.8 "Inline form "Schritt" (transition)" Page 242)

9.6.8 Inline form "Schritt" (transition)

Description The instruction is used to program the condition (Boolean operand) that must be met in order to switch to the next step in the sequence. If required, several operands (maximum 11) can be linked.



If, during program execution, multiple transition conditions are met in one step, it is always the last transition that is carried out.

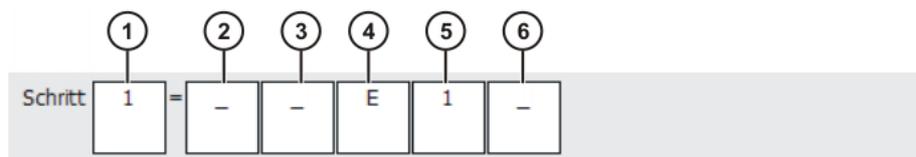


Fig. 9-68: Inline form "Schritt" (transition)

Item	Description
1	Number of the step to which switching is to be carried out <ul style="list-style-type: none"> ■ 1 ... 32,000
2	Operator <ul style="list-style-type: none"> ■ →, (
3	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ →, !
4	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S

Item	Description
5	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)
6	Operator <ul style="list-style-type: none"> →,)

Example

```

1      Makro Schrittkette
2      1  SCHRITT 1
3      A1 = EIN
4      Schritt2 = E1
5      2  SCHRITT 2
6      A2 = EIN
7      Schritt3 = E1
8      3  SCHRITT 3
9      4  SCHRITT 4

```

Line	Description
2, 5	Steps with open Point PLC
3, 7	PLC instructions: <ul style="list-style-type: none"> Step 1: output 1 is set. Step 2: output 2 is set.
4, 7	Transition condition to the next step: input 1 is TRUE.
8	Step with closed Point PLC

9.6.9 MakroTrigger

Description

The **MakroTrigger** macro contains 8 predefined triggers for which trigger instructions can be programmed. All statements of a trigger are executed within a cycle of the Submit interpreter. Very fast reaction times can be implemented with the trigger function, e.g. if an error message is received at an input.

All triggers are switched off by default. In order to switch on a trigger, a switch-on condition must be programmed.



Triggers 31...38 are reserved for the predefined triggers. These triggers cannot be deleted and no additional triggers can be inserted.

Precondition

- MakroTrigger is open.
- Operating mode T1 or T2

Procedure

- Position the cursor in the line containing the trigger that is to be programmed.
- Open the inline form **Trigger** by pressing the **Change** softkey.
- Program the switch-on condition in the inline form.
(>>> 9.6.10 "Inline form "Trigger"" Page 244)
- Save by pressing the **Cmd Ok** softkey.
- Program trigger instructions.

The following PLC instructions can be used and opened by means of a softkey:

Softkey	Description
A/M/F	(>>> 9.4.7 "Inline form "A/M/F"" Page 203)
i/bin	(>>> 9.4.9 "Inline form "i/bin"" Page 204)

Softkey	Description
t=	(>>> 9.4.11 "Inline form "t= (Start)"" Page 205)
t=STOP	(>>> 9.4.13 "Inline form "t=STOP"" Page 207)
Compare	(>>> 9.4.15 "Inline form "Compare"" Page 207)



In the case of an upgrade, switch-on conditions and trigger instructions that have already been programmed are transferred to the SRC file of the macro.

9.6.10 Inline form "Trigger"

Description

The instruction is used to program the condition (Boolean operand) that must be met in order to switch on the trigger and execute the trigger instructions.

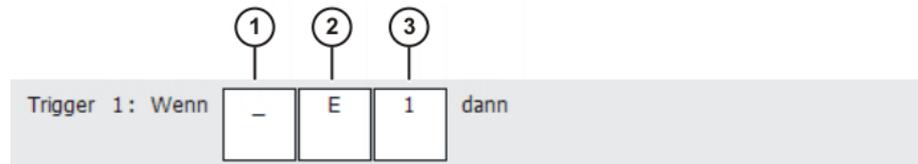


Fig. 9-69: Inline form "Trigger"

Item	Description
1	Operator. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ _, !
2	Boolean operand (condition) <ul style="list-style-type: none"> ■ EIN (ON), AUS (OFF), E, A, M, F, T, S
3	Number of the operand. The input box is not available if the operands EIN (ON) and AUS (OFF) are selected. <ul style="list-style-type: none"> ■ Range of values: Dependent on the operand (>>> 9.4.1 "Boolean operands" Page 201)

Example

```

1      1  Trigger 1: Wenn E1 dann
2          A5 = EIN
3          t1 ( EIN )= 500 [1/10Sek]
4      2  Trigger 2: Wenn AUS dann
5      3  Trigger 3: Wenn AUS dann
6          ...
7      8  Trigger 8: Wenn AUS dann
    
```

Line	Description
1	Switch-on condition: input 1 is TRUE.
2 ... 3	Trigger instructions that are executed when the switch-on condition is met: <ul style="list-style-type: none"> ■ Output 5 is set. ■ Timer 1 is started immediately with the start value 50 s.
4 ... 7	Deactivated triggers

10 Programming for user group "Expert" (KRL syntax)



KRL syntax should always be used in the VW_USER module, e.g. for programming functions or subprograms.

10.1 Overview of KRL syntax

Variables and declarations	
DECL	(>>> 10.4.1 "DECL" Page 251)
ENUM	(>>> 10.4.2 "ENUM" Page 252)
STRUC	(>>> 10.4.3 "STRUC" Page 254)
Motion programming	
PTP	(>>> 10.5.1 "PTP" Page 255)
PTP_REL	(>>> 10.5.2 "PTP_REL" Page 256)
LIN	(>>> 10.5.3 "LIN" Page 257)
LIN_REL	(>>> 10.5.4 "LIN_REL" Page 257)
CIRC	(>>> 10.5.5 "CIRC" Page 259)
CIRC_REL	(>>> 10.5.6 "CIRC_REL" Page 260)
SPLINE ... ENDSPLINE	(>>> 10.6.1 "SPLINE ... ENDSPLINE" Page 261)
SLIN	(>>> 10.6.2 "SLIN" Page 262)
SCIRC	(>>> 10.6.3 "SCIRC" Page 263)
SPL	(>>> 10.6.4 "SPL" Page 265)
TIME_BLOCK	(>>> 10.6.5 "TIME_BLOCK" Page 265)
Program execution control	
CONTINUE	(>>> 10.7.1 "CONTINUE" Page 269)
EXIT	(>>> 10.7.2 "EXIT" Page 270)
FOR ... TO ... ENDFOR	(>>> 10.7.3 "FOR ... TO ... ENDFOR" Page 270)
GOTO	(>>> 10.7.4 "GOTO" Page 271)
HALT	(>>> 10.7.5 "HALT" Page 272)
IF ... THEN ... ENDIF	(>>> 10.7.6 "IF ... THEN ... ENDIF" Page 272)
LOOP ... ENDLOOP	(>>> 10.7.7 "LOOP ... ENDLOOP" Page 273)
REPEAT ... UNTIL	(>>> 10.7.8 "REPEAT ... UNTIL" Page 273)
SWITCH ... CASE ... ENDSWITCH	(>>> 10.7.9 "SWITCH ... CASE ... ENDSWITCH" Page 274)
WAIT ... FOR	(>>> 10.7.10 "WAIT FOR" Page 275)
WAIT ... SEC	(>>> 10.7.11 "WAIT SEC" Page 275)
WHILE ... ENDWHILE	(>>> 10.7.12 "WHILE ... ENDWHILE" Page 276)
Inputs/outputs	
ANIN	(>>> 10.8.1 "ANIN" Page 276)
ANOUT	(>>> 10.8.2 "ANOUT" Page 277)
PULSE	(>>> 10.8.3 "PULSE" Page 279)
SIGNAL	(>>> 10.8.4 "SIGNAL" Page 282)
Subprograms and functions	
RETURN	(>>> 10.9.1 "RETURN" Page 283)

Interrupt programming	
BRAKE	(>>> 10.10.1 "BRAKE" Page 284)
INTERRUPT	(>>> 10.10.3 "INTERRUPT" Page 286)
INTER- RUPT ... DECL ... WHEN ... DO	(>>> 10.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Page 284)
RESUME	(>>> 10.10.4 "RESUME" Page 287)
Path-related switching actions (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 10.11.1 "TRIGGER WHEN DISTANCE" Page 289)
TRIGGER WHEN PATH	(>>> 10.11.2 "TRIGGER WHEN PATH" Page 292)
TRIGGER WHEN PATH (for spline)	(>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)
Communication	
(>>> 10.12 "Communication" Page 299)	
System functions	
VARSTATE()	(>>> 10.13.1 "VARSTATE()" Page 299)
Manipulating string variables	
(>>> 10.14 "Editing string variables" Page 301)	

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"> ■ Courier font ■ Upper-case letters Examples: GLOBAL; ANIN ON; OFFSET
Elements that must be replaced by program-specific entries	<ul style="list-style-type: none"> ■ Italics ■ Upper/lower-case letters Examples: <i>Distance</i> ; <i>Time</i> ; <i>Format</i>
Optional elements	<ul style="list-style-type: none"> ■ In angle brackets Example: <STEP <i>Increment</i> >
Elements that are mutually exclusive	<ul style="list-style-type: none"> ■ Separated by the " " symbol Example: IN OUT

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.
(>>> 10.3.5 "Areas of validity" Page 249)
- **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.

(>>> 11 "KRL function call with parameter transfer (=USER)" Page 307)



All global subprogram and function calls must begin with the prefix "VW_". The corresponding variables and user-defined data types must be declared in the section USER GLOBALS in \$CONFIG.DAT.

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 <ul style="list-style-type: none"> ■ $-2^{31}-1 \dots 2^{31}-1$ Examples: 1; 32; 345
Real	REAL	Floating-point number <ul style="list-style-type: none"> ■ $+1.1E-38 \dots +3.4E+38$ Examples: 1.43; 38.50; 300.25
Boolean	BOOL	Logic state <ul style="list-style-type: none"> ■ TRUE ■ FALSE
Character	CHAR	1 character <ul style="list-style-type: none"> ■ 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.

```
STRUC 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.

```
STRUC 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.

```
STRUC 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 ENDDEF containing the declaration of the variables.
Constant	Valid in the module to which the data list in which the constant was declared belongs.

Data object	Area of validity
User-defined data type	If the data type has been defined in a DAT file: valid in the SRC file that belongs to the DAT file. If the data type has been defined in an SRC file: valid at, or below, the program level in which it was declared.
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.



Für Variablen und benutzerdefinierte Datentypen kann GLOBAL nur verwendet werden, wenn sie in einer Datenliste vereinbart sind.

Variables and user-defined data types are 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:

```
<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 249)
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 249) Note: Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.
NameEnum-Type	Name of the new enumeration type. Recommendation: For user-defined data types, assign names ending in _TYPE, to distinguish them from variable names.
Constant	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 249) Note: Data types defined using the keyword GLOBAL must not be used in \$CONFIG.DAT.
<i>NameStructureType</i>	Name of the new structure type. The names of user-defined data types should end in <code>_TYPE</code> , to distinguish them from variable names.
<i>Data type</i>	TYPE: Any data type Structure types are also permissible as data types.
<i>Component</i>	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.
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.
<i>CP approximation</i>	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"> ■ C_DIS Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.

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>
<i>C_PTP</i>	<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"> ■ C_DIS Distance parameter (default): Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.

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>	<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 LIN (and CIRC) motions.</p> <p>The coordinates refer to the BASE coordinate system.</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"> ■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.

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"> ■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.
#BASE, #TOOL	<ul style="list-style-type: none"> ■ #BASE Default setting. The coordinates of the end point refer to the BASE coordinate system. ■ #TOOL The coordinates of the end point refer to the TOOL coordinate system. <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"> ■ 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.
<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"> ■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.

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 > 360° can be programmed.</p> <ul style="list-style-type: none"> ■ 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.
<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"> ■ C_DIS Distance parameter: Approximation starts, at the earliest, when the distance to the end point falls below the value of \$APO.CDIS. ■ C_ORI Orientation parameter: Approximation starts, at the earliest, when the dominant orientation angle falls below the value of \$APO.CORI. ■ 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.

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 SPLINE ... ENDSPLINE

Description

A spline block may contain the following:

- Spline segments (only limited by the memory capacity.)
- PATH trigger
- 1 time block
(>>> 10.6.5 "TIME_BLOCK" Page 265)
- Comments
- Blank lines

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.

A spline block does not trigger an advance run stop.

The following components must be specified for the first point in a spline block:

- X, Y, Z
- A, B, C
- E1 to E6 (if present)

This also applies if the first point of the auxiliary point is a SCIRC segment.

Syntax

```
SPLINE <WITH SysVarSpline = Value1 <,Value2 , ..., ValueN> >
  Spline segment1
  <Trigger>
  ...
  <Spline segmentN-1>
  <Trigger>>
  <Spline segmentN>
ENDSPLINE <C_DIS>
```

Explanation of the syntax

Element	Description
SysVar-Spline	The following system variables may be used: \$BASE, \$TOOL, \$IPO_MODE, \$LOAD, \$VEL, \$VEL_AXIS, \$VEL_EXTAX, \$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$APO, \$JERK, \$ORI_TYPE, \$CIRC_TYPE
Value	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. A value can also be assigned to the system variables \$VEL, \$VEL_EXTAX, \$ACC_EXTAX and \$JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)
C_DIS	Approximate the end point of the spline block (= last point in the block). \$APO is used to define the earliest point at which the approximate positioning can begin.

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
```

10.6.2 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 \$BASE, \$TOOL, \$IPO_MODE or \$LOAD.

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"> ■ 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. ■ The following applies for individual blocks: 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.
SysVarSlin	<p>The following system variables may be used:</p> <p>\$VEL, \$VEL_AXIS, \$VEL_EXTAX, \$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$JERK, \$ORI_TYPE</p> <ul style="list-style-type: none"> ■ The following may also be used for SLIN in the spline block: \$EX_AX_IGNORE ■ The following may also be used for individual SLIN blocks: \$BASE, \$TOOL, \$IPO_MODE, \$LOAD, \$APO
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 \$VEL, \$VEL_EXTAX, \$ACC_EXTAX and \$JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger.</p> <p>(>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)</p>
C_DIS	<p>Only possible for individual SLIN block: Approximate the end point.</p> <p>\$APO is used to define the earliest point at which the approximate positioning can begin.</p>

10.6.3 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 \$BASE, \$TOOL, \$IPO_MODE or \$LOAD.

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	Type: POS, E6POS, FRAME The coordinates refer to the BASE coordinate system. <ul style="list-style-type: none"> ■ 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. ■ The following applies for individual blocks: 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.
Circular angle	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. Unit: degrees. There is no limit; in particular, a circular angle greater than 360° can be programmed. <ul style="list-style-type: none"> ■ 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.
SysVarScirc	The following system variables may be used: \$VEL, \$VEL_AXIS, \$VEL_EXTAX, \$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$JERK, \$ORI_TYPE, \$CIRC_TYPE, \$CIRC_MODE <ul style="list-style-type: none"> ■ The following may also be used for SCIRC in the spline block: \$EX_AX_IGNORE ■ The following may also be used for individual SCIRC blocks: \$BASE, \$TOOL, \$IPO_MODE, \$LOAD, \$APO
Value	Value assignment to the system variable. In the case of SCIRC segments: The assignment applies only for this segment. It has no effect on subsequent segments. A value can also be assigned to the system variables \$VEL, \$VEL_EXTAX, \$ACC_EXTAX and \$JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)
C_DIS	Only possible for individual SCIRC block: Approximate the end point. \$APO is used to define the earliest point at which the approximate positioning can begin.

Example

```
SCIRC P2, P3 WITH $CIRC_TYPE=#PATH
```

10.6.4 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: \$VEL, \$VEL_AXIS, \$VEL_EXTAX, \$ACC, \$ACC_AXIS, \$ACC_EXTAX, \$JERK, \$ORI_TYPE, \$CIRC_TYPE, \$EX_AX_IGNORE
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 \$VEL, \$VEL_EXTAX, \$ACC_EXTAX and \$JERK via a function. The same restrictions apply to these functions as to functions in the spline trigger. (>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)

Example

```
SPL P4 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
```

10.6.5 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"> ■ From the point before TIME_BLOCK PART = Component[x-1] to the point before TIME_BLOCK PART = Component[x] ■ 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] <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"> ■ From the point before TIME_BLOCK START to the point before TIME_BLOCK END <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.

(>>> 10.11.3 "TRIGGER WHEN PATH (for SPLINE)" Page 295)

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.

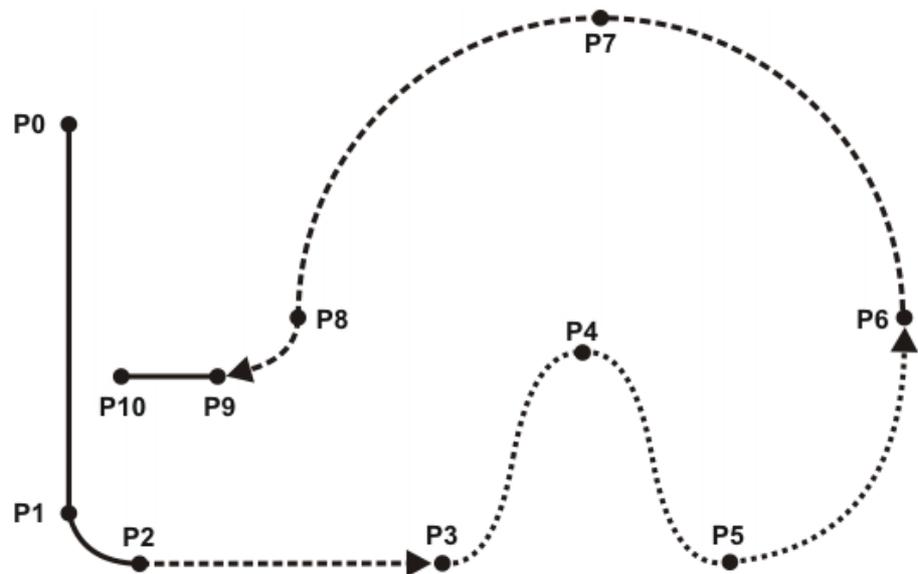


Fig. 10-1: Example TIME_BLOCK

Block selection

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

10.7 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

```

**Caution!**

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"> ■ Positive value: the loop is ended if <i>Counter</i> is greater than <i>End</i>. ■ Negative value: the loop is ended if <i>Counter</i> is less than <i>End</i>. 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 *Marker*

...

Marker:

Explanation of the syntax

Element	Description
<i>Marker</i>	Position to which a jump is made. At the destination position, <i>Marker</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"> ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

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>	Type: BOOL Possible: <ul style="list-style-type: none"> ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

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"> ■ If the condition is already TRUE when WAIT is called, program execution is not halted. ■ If the condition is FALSE, program execution is stopped until the condition is TRUE.

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"> ■ Variable of type BOOL ■ Function of type BOOL ■ Logic operation, e.g. a comparison, with a result of type BOOL

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
```



- 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).
- The robot controller has 32 analog inputs (\$ANIN[1] ... \$ANIN[32]).

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

Description

Cyclical writing (every 12 ms) to an analog output.

ANOUT triggers an advance run stop.

Syntax

Starting cyclical writing:

```
ANOUT ON Signal name = Factor * Control element <±Offset> <DELAY = ±Time> <MINIMUM = Minimum_Value> <MAXIMUM = Maximum value>
```

Ending cyclical writing:

```
ANOUT OFF Signal name
```



- 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).
- The robot controller has 32 analog outputs (\$ANOUT[1] ... \$ANOUT[32]).

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"> OUT[No] Signal variable
<i>Level</i>	Type: BOOL Logical expression: <ul style="list-style-type: none"> TRUE represents a positive pulse (high). FALSE represents a negative pulse (low).
<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:

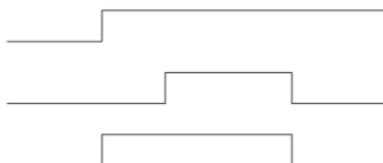


Fig. 10-2: \$OUT+PULSE, example 1

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:

```

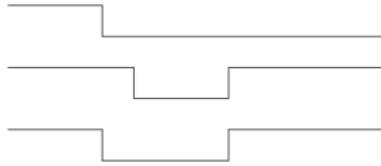


Fig. 10-3: \$OUT+PULSE, example 2

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:

```

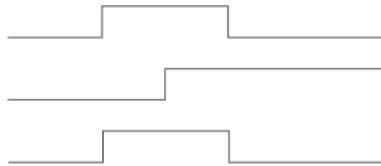


Fig. 10-4: PULSE+\$OUT, example 1

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:

```

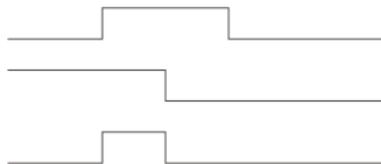


Fig. 10-5: PULSE+\$OUT, example 2

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:

```

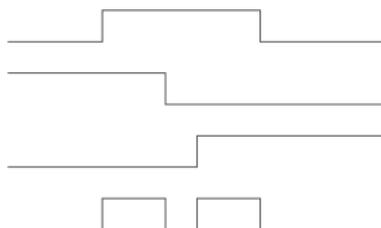


Fig. 10-6: PULSE+\$OUT, example 3

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):

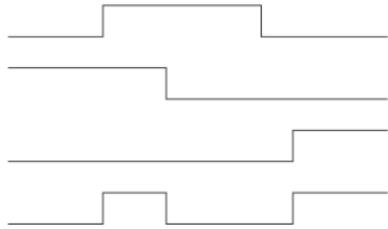


Fig. 10-7: PULSE+\$OUT, example 4

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:
```

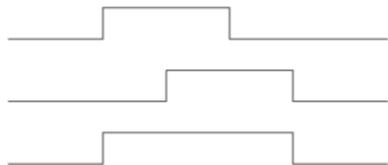


Fig. 10-8: PULSE+PULSE, example 1

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:
```

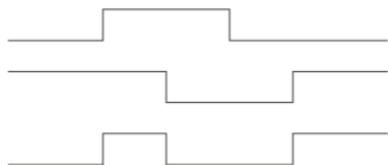


Fig. 10-9: PULSE+PULSE, example 2

```
PULSE ($OUT [50] , TRUE , 3)
PULSE ($OUT [50] , FALSE , 1)
Actual pulse characteristic at output 50:
```

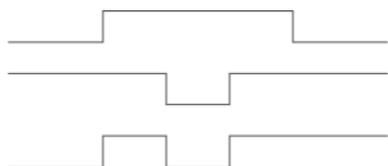


Fig. 10-10: PULSE+PULSE, example 3

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:
```

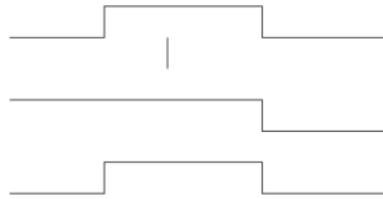


Fig. 10-11: PULSE+END, example

**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:
```

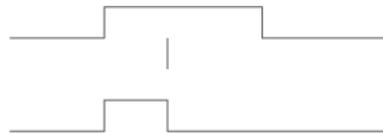


Fig. 10-12: PULSE+RESET, example

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"> ■ \$IN[x] ■ \$OUT[x] ■ \$ANIN[x] ■ \$ANOUT[x]

Element	Description
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

Brakes the robot motion.



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 286)

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 286)

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. Note: Priorities 3 and 40 to 80 are reserved for use by the system. They must not 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"> ■ a Boolean constant ■ a Boolean variable ■ a signal name ■ a comparison ■ a simple logic operation: NOT, OR, AND or EXOR
<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.



Für Variablen und benutzerdefinierte Datentypen kann GLOBAL nur verwendet werden, wenn sie in einer Datenliste vereinbart sind.

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 284)

Syntax

INTERRUPT *Action* <*Number*>

Explanation of the syntax

Element	Description
<i>Action</i>	<ul style="list-style-type: none"> ■ ON: Activates an interrupt. ■ OFF: Deactivates an interrupt. ■ DISABLE: Disables an activated interrupt. ■ ENABLE: Enables a disabled interrupt.
<i>Number</i>	<p>Type: INT</p> <p>Number (= priority) of the interrupt to which the <i>Action</i> is to refer.</p> <p><i>Number</i> 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.



Warning!

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	<p>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.</p> <p>The statement can be shifted in time. It is then not triggered exactly at the start or end point, but brought forward or delayed.</p>
Syntax	<pre>TRIGGER WHEN DISTANCE=<i>Position</i> DELAY=<i>Time</i> DO <i>Statement</i> <PRIO=<i>Priority</i>></pre>

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"> ■ 0: 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. ■ 1: 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.
<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"> ■ Position = 0 (start point) <p>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, at most, as far as the end point. If the end point is approximated, the statement can be delayed, at most, as far as the start of the approximate positioning arc.</p> ■ Position = 1 (end point) <p>In this case, a distinction must be made as to whether the end point is an exact positioning point or an approximate positioning point.</p> <ul style="list-style-type: none"> ■ Exact positioning point: 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, at most, as far as the start point. If the start point is approximated, the statement can be brought forward, at most, as far as the end of the approximate positioning arc. ■ Approximate positioning point: 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, at most, as far as the start or end of the approximate positioning arc of the end point.

Element	Description
<i>Statement</i>	Possible: <ul style="list-style-type: none"> ■ Assignment of a value to a variable Note: There must be no runtime variable on the left-hand side of the assignment. ■ OUT statement ■ PULSE statement ■ Subprogram call. In this case, <i>Priority</i> must be specified.
<i>Priority</i>	Type: INT; variable or constant Priority of the trigger. Only relevant if <i>Statement</i> calls a subprogram, and then obligatory. 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: <code>PRIO = -1</code> . 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.



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
<code>\$DIST_NEXT</code>	REAL	Distance to next point. Value is negative or 0. This system variable can be used for teaching triggers.
<code>\$DISTANCE</code>	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

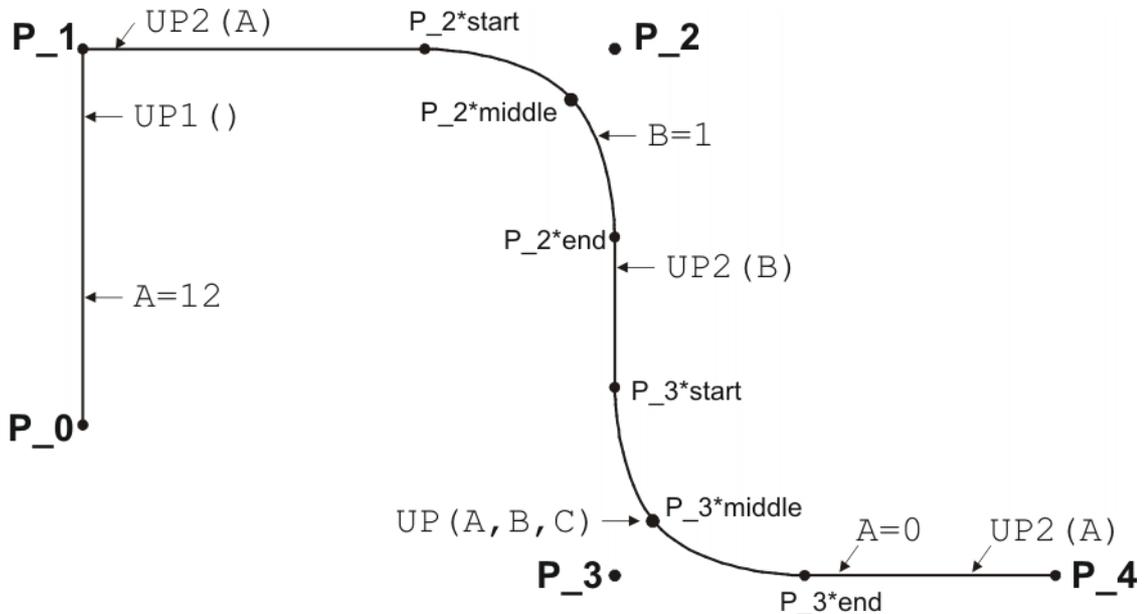


Fig. 10-13: Example of TRIGGER WHEN DISTANCE

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"> ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion. <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"> ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion. <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"> ■ Assignment of a value to a variable Note: There must be no runtime variable on the left-hand side of the assignment. ■ OUT statement ■ PULSE statement ■ subprogram call. In this case, <i>Priority</i> must be specified.
<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: <code>PRIO = -1</code>.</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.

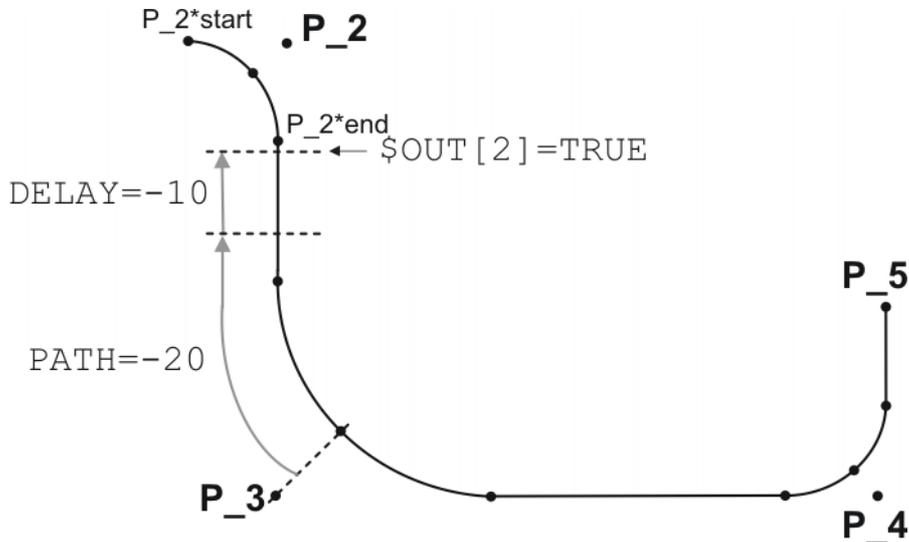


Fig. 10-14: Example of TRIGGER WHEN PATH

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)

Description

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.

- In the negative direction, the statement can be shifted, at most, as far as the first point before the spline block.
- In the positive direction, the statement can be shifted as far as the last point of the spline block if this is an exact positioning point. If this point is approximated, the statement can be shifted, at most, as far as the next exact positioning point.

Syntax

TRIGGER WHEN PATH = *Distance* <ONSTART> DELAY = *Time* DO *Statement*
<PRIO = *Priority*>

Functions

PATH and DELAY can call functions. The following restrictions apply here:

- The KRL program with the function must have the attribute **Hidden**.
- The function must be globally valid.
- The functions may only contain the following statements:

- Write to user-defined variable
- IF statement
- Assignment
- Read system variable

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"> ■ Positive value: shifts the statement towards the end of the motion. ■ Negative value: shifts the statement towards the start of the motion.
ONSTART	<p>the PATH value refers to the start point.</p> <p>Without ONSTART: the PATH value refers to the end point.</p> <p>(>>> 10.11.3.1 "Spline: trigger point in the case of approximate positioning" Page 297)</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"> ■ Positive value: shifts the statement towards the end of the motion. Maximum: 1,000 ms ■ Negative value: shifts the statement towards the start of the motion.
<i>Statement</i>	<p>Possible:</p> <ul style="list-style-type: none"> ■ Assignment of a value to a variable <ul style="list-style-type: none"> Note: There must be no runtime variable on the left-hand side of the assignment. ■ OUT statement ■ PULSE statement ■ subprogram call. In this case, <i>Priority</i> must be specified.
<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: <code>PRIO = -1</code>.</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.

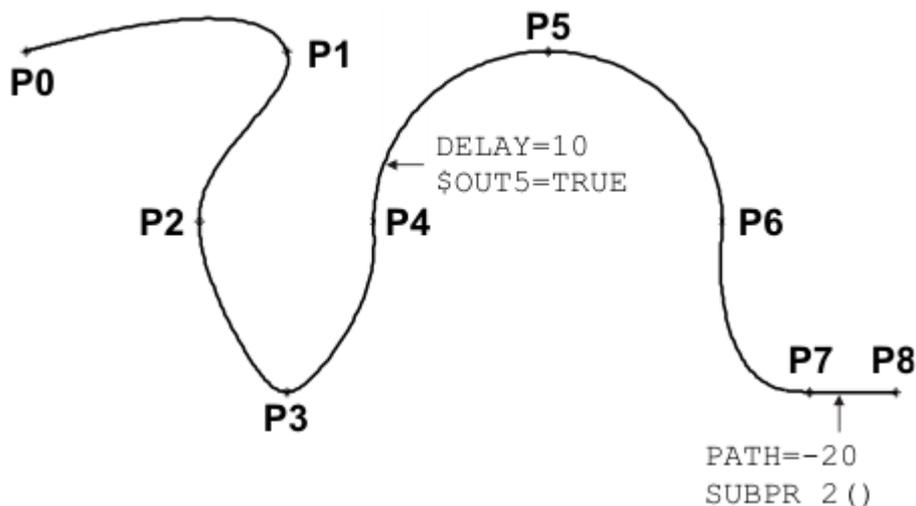


Fig. 10-15: Example of TRIGGER WHEN PATH (for spline)

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 $P_{\text{StartApprox}} \rightarrow P3'$ is the same as $P_{\text{StartApprox}} \rightarrow P3$.

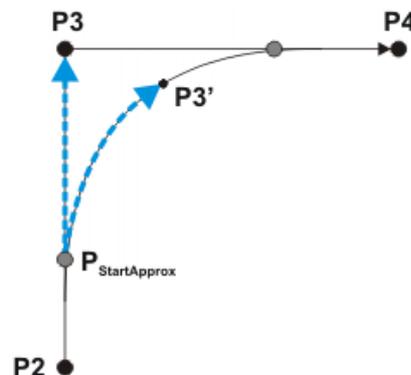


Fig. 10-16: Trigger 1: Trigger point in the case of approximate positioning

P3	Trigger point in the case of exact positioning
P3'	Trigger point in the case of approximate positioning
$P_{\text{StartApprox}}$	Start of the approximate positioning arc
$P_{\text{EndApprox}}$	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 $P_{\text{EndApprox}} \rightarrow P3'$ is the same as $P_{\text{EndApprox}} \rightarrow P3$.

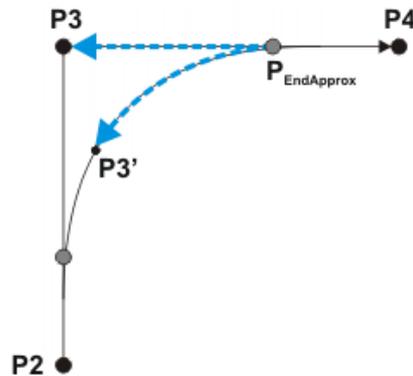


Fig. 10-17: Trigger 2: Trigger point in the case of approximate positioning

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.

10.14 Editing string variables

Overview

Various functions are available for editing string variables. The functions can be used in SRC files, in SUB files and in the variable display.

String variable length in the declaration

(>>> 10.14.1 "String variable length in the declaration" Page 301)

String variable length after initialization

(>>> 10.14.2 "String variable length after initialization" Page 302)

Deleting the contents of a string variable

(>>> 10.14.3 "Deleting the contents of a string variable" Page 302)

Extending a string variable

(>>> 10.14.4 "Extending a string variable" Page 303)

Searching a string variable

(>>> 10.14.5 "Searching a string variable" Page 303)

Comparing the contents of string variables

(>>> 10.14.6 "Comparing the contents of string variables" Page 304)

Copying a string variable

(>>> 10.14.7 "Copying a string variable" Page 305)

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 = 7

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"> ■ The contents of the string variable have been deleted: TRUE ■ The contents of the string variable have not been deleted: FALSE
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 <code>StrDest[]</code> and <code>StrToAdd[]</code> If the sum is longer than the previously defined length of <code>StrDest[]</code> , 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 <code>StrDest[]</code> 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.

Element	Description
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"> ■ #CASE_SENS: Upper and lower case are taken into consideration. ■ #NOT_CASE_SENS: Upper and lower case are ignored.

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"> ■ The character strings match: TRUE ■ The character strings do not match: FALSE
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"> ■ #CASE_SENS: Upper and lower case are taken into consideration. ■ #NOT_CASE_SENS: Upper and lower case are ignored.

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"> ■ The string variable was copied successfully: TRUE ■ The string variable was not copied: FALSE
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 KRL function call with parameter transfer (=USER)

Overview

Local subprograms for programming functions are available in the VW_USER module in the directory C:\KRC\ROBOTER\KRC\R1. The local subprogram determines the point in time at which the function is executed.

The module VW_USER is only available to the user group "Expert".

The inline form "VW User" can be used to transfer up to 7 parameters from the motion program to a function. If these 7 parameters are not sufficient, the inline form "VW User" can be configured to allow the transfer of additional parameters.

Step	Description
1	Program function. (>>> 11.1 "VW_USR_R" Page 307) (>>> 11.2 "VW_USR_S" Page 307)
2	Configure inline form "VW User". (>>> 11.3 "Configuring the inline form "VW User"" Page 308)
3	Insert inline form "VW User" into Folge or subprogram. (>>> 9.4.49 "Calling VW_USER" Page 236)

11.1 VW_USR_R

The file VW_USR_R is integrated into the robot interpreter.

The following local subprograms are available:

Subprogram	Description
USER_INIT	Function is executed when a Folge is initialized.
USER_ADV	Function is executed in the advance run, before the point in whose Point PLC the inline form "VW User" has been inserted.
USER_TRIG	Function is executed in the point in whose Point PLC the inline form "VW User" has been inserted. If PLC trigger has been programmed, the function call is delayed or brought forward.
USER_MAIN	Function is executed in the main run, after the point in whose Point PLC the inline form "VW User" has been inserted.



If the inline form "VW User" has been inserted into a Point PLC, the robot interpreter calls the file VW_USR_R three times: in the advance run before the point, in the main run after the point and in the point itself, or before the point is reached if offset by the PLC trigger.

11.2 VW_USR_S

The file VW_USR_S is integrated into the Submit interpreter.

The following local subprograms are available:

Subprogram	Description
USER_INIT	Function is executed when a Submit interpreter is initialized.
USER_SAW	Function is executed when block coincidence is lost.
USER_RESET	Function is executed when the program is reset.
USER_CANCEL	Function is executed when the program is deselected.
USER_LOOP	Function is executed in the Submit interpreter cycle.



Caution!

Functions in the file VW_USR_S must not contain instructions that stop the Submit interpreter, e.g. HALT or WAIT.

11.3 Configuring the inline form “VW User”

Description

The inline form “VW User” for transferring up to 7 parameters is predefined in the file VW_USER.INI, in the directory C:\KRC\ROBOTER\INIT.

The file VW_USER.INI is only available to the user group “Expert”.

If these 7 parameters are not sufficient, additional parameters can be defined in main groups and subgroups in the file VW_USER.INI. Each main group can consist of up to 7 parameters or 10 subgroups. Each subgroup consists of up to 7 parameters.

Syntax

```
[Hauptgruppe0]
Titel=String
< [Untergruppe1] >
Name=String
P [1]
Visible=Bool
<num1=String>
<ana5=String>
<num*n=String*n>|<num*n=String=*n [Einheit] >
P [2]
...
P [7]
Visible=Bool
<TRUE=String>
<FALSE=String>
< [GruppenEnde] >
[GruppenEnde]
```

Explanation of the syntax

Element	Description
[Hauptgruppe0] ... [Hauptgruppe9]	Definition of the main groups (max. 10) Note: Main groups must not be nested.
Titel=	Name of the main group The name is used in the comment in the inline form and in the program text in the Point PLC.
[Untergruppe1] ... [Untergruppe10]	Definition of the subgroups (max. 10) Note: Subgroups must not be nested.
Name=	Name of the subgroup The name is used in the comment in the inline form and in the program text in the Point PLC.
P [1] ... P [7]	Definition of the parameters of a main group or subgroup <ul style="list-style-type: none"> ■ P[1] ... P[6]: Parameter of type INTEGER The arithmetic operands are available for selection in the inline form. ■ P[7]: Parameter of type BOOL The Boolean operands are available for selection in the inline form.
visible=	Defines whether the parameter is shown in the inline form. <ul style="list-style-type: none"> ■ TRUE: The parameter is shown. ■ FALSE: The parameter is not shown.
num1= ... num99 999=	A constant character string can be assigned to the value of a parameter. Entering this value in the inline form generates this character string in the comment and in the program text in the Point PLC. The permissible values depend on the operand type. Examples: <ul style="list-style-type: none"> ■ Num1=Value-10 ■ Num2=Value-15 ■ Ana3=Analog_3 ■ EIN=SensorOn
i1= ... i22=	
bin1= ... bin20=	
binin1= ... binin20=	
t1= ... t20=	
ana1= ... ana16=	
anain1= ... anain16=	
p1= ... p256=	
EIN= , AUS=	
num*n= ... *n	

Element	Description
num*n= ... =*n [...]	A variable character string in the form of an assignment can be assigned to the value of a parameter of type "num". Example: <ul style="list-style-type: none"> num*n=X=*n [mm] Entering the number 1 in the inline form generates the character string "X=1[mm]" in the comment and in the program text in the Point PLC; entering the number 2 generates the character string "X=2[mm]", etc.
[GruppenEnde]	Each main group and each subgroup must be closed with this keyword.

Example 1

Predefined VW_USER.INI

```
[Hauptgruppe0]
Titel=VW
Name=User
[P1]
visible=TRUE
num*n=X=*n [mm]
il=Timer_1
bin1=Binary_1
anal=Analog_1
[P2]
visible=TRUE
[P3]
visible=TRUE
[P4]
visible=TRUE
[P5]
visible=TRUE
[P6]
visible=TRUE
[P7]
visible=TRUE
[GruppenEnde]
```

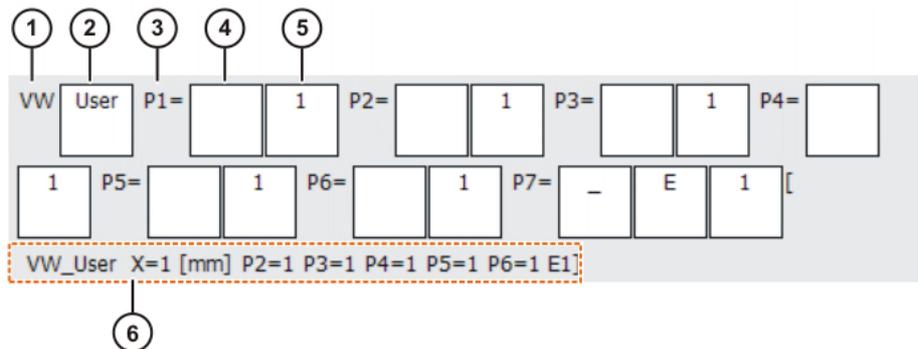


Fig. 11-1: Inline form "VW User" (default)

Item	Description
1	Title or name of the main group
2	Name of the subgroup
3	Parameter P1
4	Selection box with the operands of parameter P1 The empty box corresponds to the operand "num".

Item	Description
5	Input box for the value of parameter P1
6	The comment is formed from the following elements: <ul style="list-style-type: none"> ■ Title or name of the main group ■ Name of the subgroup ■ Parameters that are defined with <code>visible=TRUE</code>

Example 2

Modified VW_USER.INI with one main group and 2 subgroups

```
[Hauptgruppe0]
Titel=VW_USER
[Untergruppe1]
Name=LIN_REL
[P1]
visible=TRUE
num*n=X=*n [mm]
binin1=binin1
[P2]
visible=TRUE
num*n=Y=*n [mm]
[P3]
visible=TRUE
num*n=Z=*n [mm]
[P4]
visible=FALSE
[P5]
visible=FALSE
[P6]
visible=FALSE
[P7]
visible=TRUE
TRUE=APO
FALSE=NO APO
[GruppenEnde]
[Untergruppe2]
Name=Vision
[P1]
visible=TRUE
num1=Sample1
num2=Sample2
num3=Sample3
[P2]
visible=TRUE
num*n=Focus=*n [mm]
[P3]
visible=FALSE
[P4]
visible=FALSE
[P5]
visible=FALSE
[P6]
visible=FALSE
[P7]
visible=TRUE
TRUE=Adjusted
FALSE=Standard
[GruppenEnde]
[GruppenEnde]
```

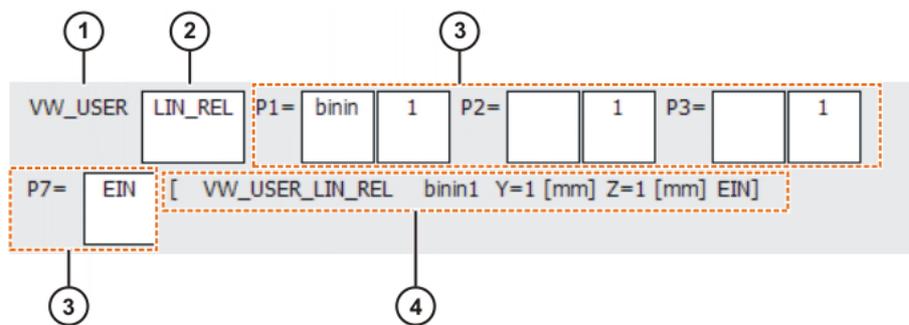


Fig. 11-2: Inline form “VW User” (example)

Item	Description
1	Name of the main group
2	Selection box with the names of the subgroups <ul style="list-style-type: none"> ■ LIN_REL ■ Vision
3	Selection and input boxes of parameters P1 ... P3 and P7 Parameters P4 ... P6 are not shown in the inline form, as they are defined with <code>visible=FALSE</code> in <code>VW_USER.INI</code> .
4	Comment

11.4 Example programs

11.4.1 Transferring parameters for LIN_REL motion to a function

Function

```

1 DEF USER_MAIN (PARAR1 :IN,PAR2 :IN, PAR3 :IN,PAR4 :IN,PAR5 :IN,
2   PAR6 :IN,PAR7 :IN)
3   ;Aufruf im Hauptlauf
4   INT PAR1,PAR2,PAR3,PAR4,PAR5,PAR6
5   BOOL PAR7
6   DECL E6POS P_REL
7   P_REL={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0,S 'H0',T 'H0',E1 0.0,
8     E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
9   P_REL.X=PAR1
10  P_REL.Y=PAR2
11  P_REL.Z=PAR3
12  P_REL.A=PAR4
13  P_REL.B=PAR5
14  P_REL.C=PAR6
15  IF PAR7==TRUE THEN
16  LIN_REL P_REL C_VEL
17  ELSE
18  LIN_REL P_REL
19  ENDIF

```

Description

Line	Description
1	Subprogram USER_MAIN in the file <code>VW_USR_R.SRC</code> The function is executed in the main run, after the point in whose Point PLC the inline form “VW User” has been inserted.
5	Declaration of the variable <code>P_REL</code> of structure type <code>E6POS</code>
6	<code>P_REL</code> defines the end point of the <code>LIN_REL</code> motion. The value <code>P_REL</code> is returned to the motion program.
7	Parameter P1 of VW User: <code>LIN_REL</code> motion in X in [mm]
8	Parameter P2 of VW User: <code>LIN_REL</code> motion in Y in [mm]
9	Parameter P3 of VW User: <code>LIN_REL</code> motion in Z in [mm]

Line	Description
10	Parameter P4 of VW User: LIN_REL motion in A in [°]
11	Parameter P5 of VW User: LIN_REL motion in B in [°]
12	Parameter P6 of VW User: LIN_REL motion in C in [°]
13 ... 17	<p>Conditional branch to define the approximate positioning characteristics with parameter P7 of VW User</p> <ul style="list-style-type: none"> ■ P7 = TRUE: LIN_REL motion to end point P_REL with approximate positioning (C_VEL) ■ P7 = FALSE: LIN_REL motion to end point P_REL without approximate positioning

VW_USER.INI

```
[Hauptgruppe0]
Titel = VW
Name = User
[P1]
Visible = TRUE
Num*n=X=*n [mm]
[P2]
Visible = TRUE
Num*n=Y=*n [mm]
[P3]
Visible = TRUE
Num*n=Z=*n [mm]
[P4]
Visible = TRUE
Num*n=A=*n [Grad]
[P5]
Visible = TRUE
Num*n=B=*n [Grad]
[P6]
Visible = TRUE
Num*n=C=*n [Grad]
[P7]
Visible = TRUE
TRUE = APO
FALSE = No APO
[GruppenEnde]
```

11.4.2 Transferring parameters to a function for executing a pattern

Function

```

1  DEF USER_MAIN (PARA1 :IN,PAR2 :IN, PAR3 :IN,PAR4 :IN,PAR5 :IN,
   PAR6 :IN,PAR7 :IN)
2  ;Aufruf im Hauptlauf
3  INT PARA1,PAR2,PAR3,PAR4,PAR5,PAR6
4  BOOL PAR7
5  DECL POS P_FIG
6  P_FIG={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0,S 'H0',T 'H0'}
7
8  SWITCH PAR6
9
10 CASE 1
11 ;Rectangle
12 P_FIG.X=PAR1
13 P_FIG.Y=0
14 LIN_REL P_FIG
15 P_FIG.X=0
16 P_FIG.Y=PAR2
17 LIN_REL P_FIG
18 P_FIG.X=-PAR1
19 P_FIG.Y=0
20 LIN_REL P_FIG
21 P_FIG.X=0
22 P_FIG.Y=-PAR2
23 LIN_REL P_FIG
24
25 CASE 2
26 ;Triangle
27 P_FIG.X=PAR1*0.5
28 P_FIG.Y=PAR2
29 LIN_REL P_FIG
30 P_FIG.X=PAR1
31 P_FIG.Y=0
32 LIN_REL P_FIG
33 P_FIG.X=-PAR1*0.5
34 P_FIG.Y=PAR2
35 LIN_REL P_FIG
36
37 ENDSWITCH
38 END

```

Description

Line	Description
1	Subprogram USER_MAIN in the file VW_USR_R.SRC The function is executed in the main run, after the point in whose Point PLC the inline form "VW User" has been inserted.
5	Declaration of the variable P_FIG of structure type POS
6	P_FIG defines the end point of the LIN_REL motion. The value P_FIG is returned to the motion program.
8	SWITCH statement for defining the pattern to be executed with parameter P6 of VW User <ul style="list-style-type: none"> ■ P6 = 1: rectangle (CASE 1) ■ P6 = 2: triangle (CASE 2)

Line	Description
12 ... 23	<p>Statements for executing the rectangle pattern</p> <ul style="list-style-type: none"> ■ Parameter P1 of VW User: side length X in [mm] ■ Parameter P2 of VW User: side length Y in [mm] <p>Parameters P1 and P2 are used to define the 4 end points P_FIG that are addressed in sequence with LIN_REL.</p>
27 ... 35	<p>Statements for executing the triangle pattern (equilateral)</p> <ul style="list-style-type: none"> ■ Parameter P1 of VW User: side length X in [mm] ■ Parameter P2 of VW User: height Y in [mm] <p>Parameters P1 and P2 are used to define the 3 end points P_FIG that are addressed in sequence with LIN_REL.</p>

VW_USER.INI

```
[Hauptgruppe0]
Titel = VW
Name = User
[P1]
Visible = TRUE
num*n=X=*n [mm]
[P2]
Visible = TRUE
num*n=Y=*n [mm]
[P3]
Visible = FALSE
[P4]
Visible = FALSE
[P5]
Visible = FALSE
[P6]
Visible = TRUE
num1=Rectangle
num2=Triangle
[P7]
Visible = FALSE
[GruppenEnde]
```


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 ■ Select the menu sequence **Diagnosis > Logbook** and select the desired log.

Overview The following logs are available:

Log	Description
Custom	The log events selected by the user are displayed.
All Logs	All log events are displayed.
Setup Entries	Log events of type "Setup" are displayed, i.e. all log events involving mastering or calibration.

The following tabs are available:

- **Log** for displaying the log
(>>> 12.1.2 "'Log" tab" Page 318)
- **Filter** for selecting the log events to be displayed in the custom log
(>>> 12.1.3 "'Filter" tab" Page 319)

12.1.2 "Log" tab

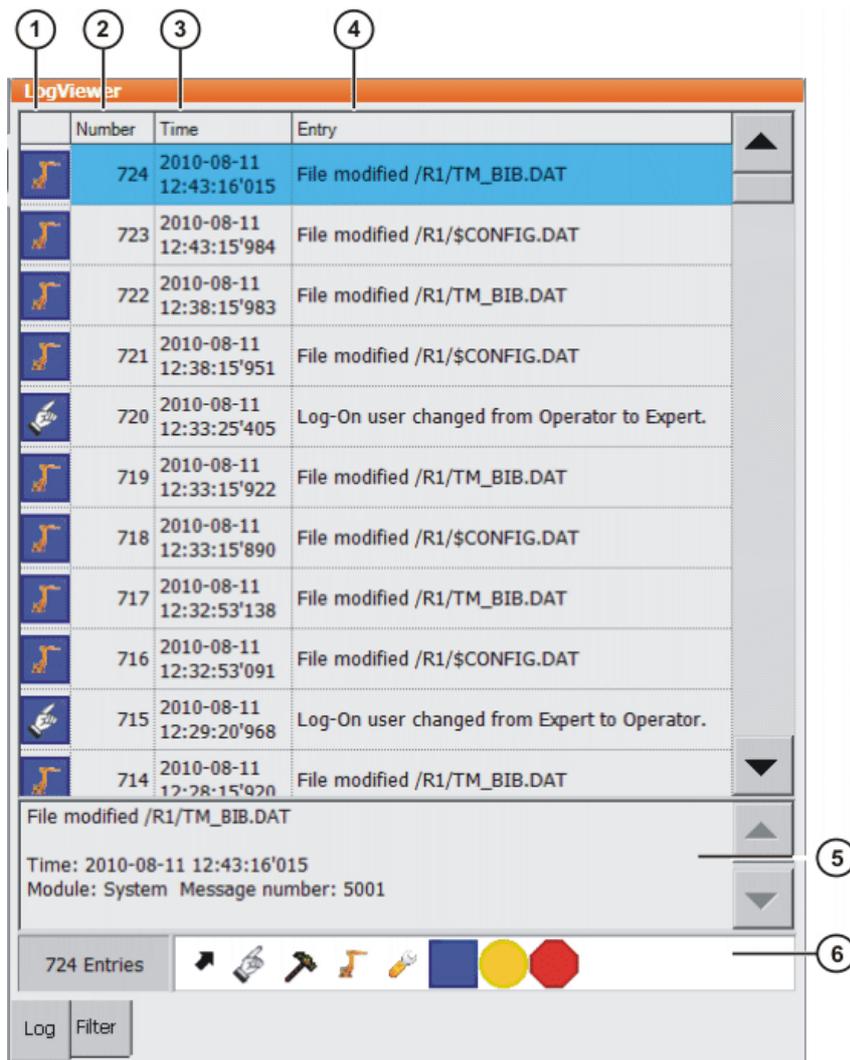


Fig. 12-1: Logbook, Log tab

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 Filter 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 softkeys are available:

Softkey	Description
Export	Exports the log data as a text file. (>>> 12.1.4 "Configuring the logbook" Page 319)
Refresh	Refreshes the log display.

12.1.3 "Filter" tab

The **Filter** tab is only displayed if the log **User-defined** has been selected.

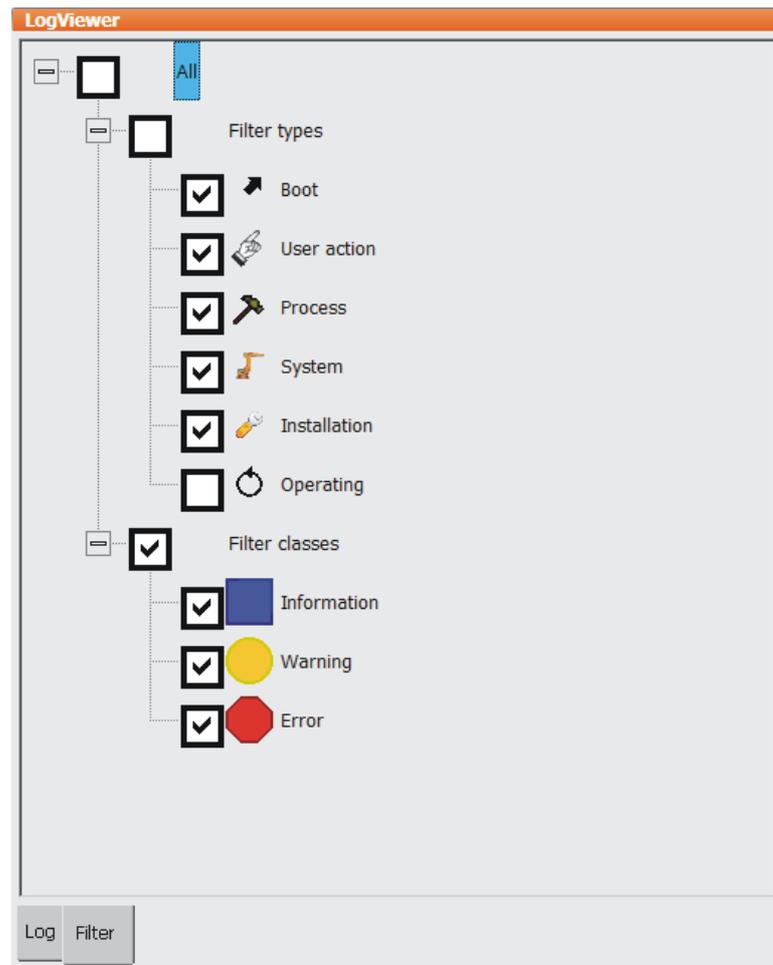


Fig. 12-2: Logbook, Filter tab

12.1.4 Configuring the logbook

Precondition ■ Expert user group

Procedure

1. Select the menu sequence **Diagnosis > Logbook > Configuration**. A window opens.
2. Make the desired settings.
3. Press **OK** to save the configuration and close the window.

Description

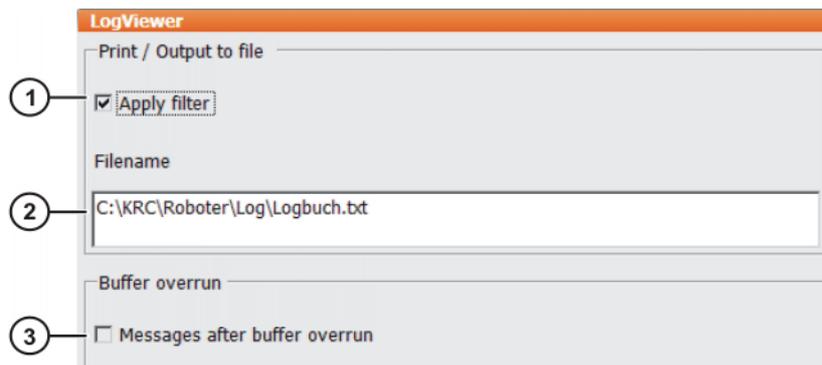


Fig. 12-3: Logbook configuration window

Item	Description
1	<ul style="list-style-type: none"> ■ Check box active: the log events selected with the filter are saved in the text file. ■ Check box not active: all log events are saved in the text file.
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"> ■ Check box active: log data deleted because of a buffer overflow are indicated in gray in the text file. ■ Check box not active: log data deleted because of a buffer overflow are not indicated in the text file.

12.2 Displaying a wait condition

This function is used if program execution is not resumed because the conditions of a wait statement programmed in the Point PLC are not met.

Precondition

- A message has been generated in the message window indicating that the wait condition is being analyzed. The button **Analyze** is displayed.

Procedure

- Press **Analyze** in the message window.
A window opens. It indicates which operands of a wait condition have not yet been fulfilled.



Caution!

If program execution is resumed following the simulation, the robot controller may behave differently from how it was programmed. This may result in collisions.

Description

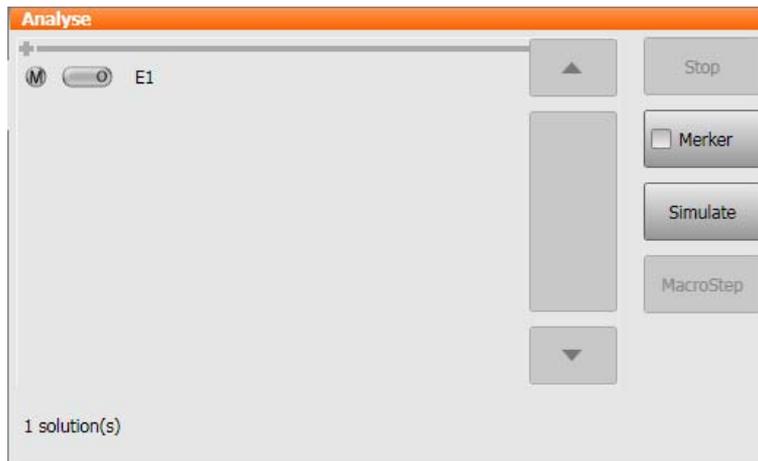


Fig. 12-4: Displaying a wait condition

The following softkeys are available:

Softkey	Description
Merker	<ul style="list-style-type: none"> ■ Check box active: Merker (cyclical flags) are displayed. ■ Check box not active: Merker (cyclical flags) are not displayed.
Simulate	<p>The signal combination at the inputs/outputs defined in the wait statement can be simulated and program execution resumed by pressing Simulate.</p> <p>Precondition: Operating mode T1 or T2.</p>

12.3 Displaying the caller stack

This function displays the data for the process pointer (\$PRO_IP).

Precondition

- User group "Expert"
- Program is selected.

Procedure

- Select the menu sequence **Diagnosis > Caller Stack**.

Description

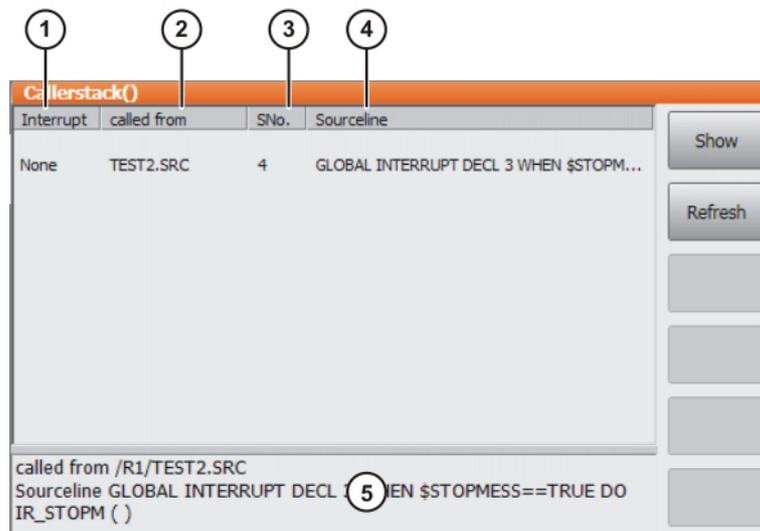


Fig. 12-5: Caller Stack window

Item	Description
1	<ul style="list-style-type: none"> ■ None: Call not initiated by interrupt ■ [No.]: Call initiated by interrupt with the number [No.]
2	This file contains the call.
3	<p>The program line with this number contains the call.</p> <p>Preconditions in the program for the correct line to be determined using the number:</p> <ul style="list-style-type: none"> ■ Detail view (ASCII mode) is activated. ■ All Point PLCs are open.
4	Source line
5	Detailed information about the entry selected in the list

12.4 Displaying the reference list

The reference list displays all inputs/outputs, timers, flags, process parameters, interlocks, macro calls and subprogram calls used in Folgen, subprograms and macros.

Procedure

- Select the menu sequence **Diagnosis > Reference list**.

The reference list is generated and displayed in the **VWReferenceList** window. At the same time, a text file is generated for the reference list and automatically saved in the directory C:\KRC\ROBOTER\INIT\RefListe.txt.

Description

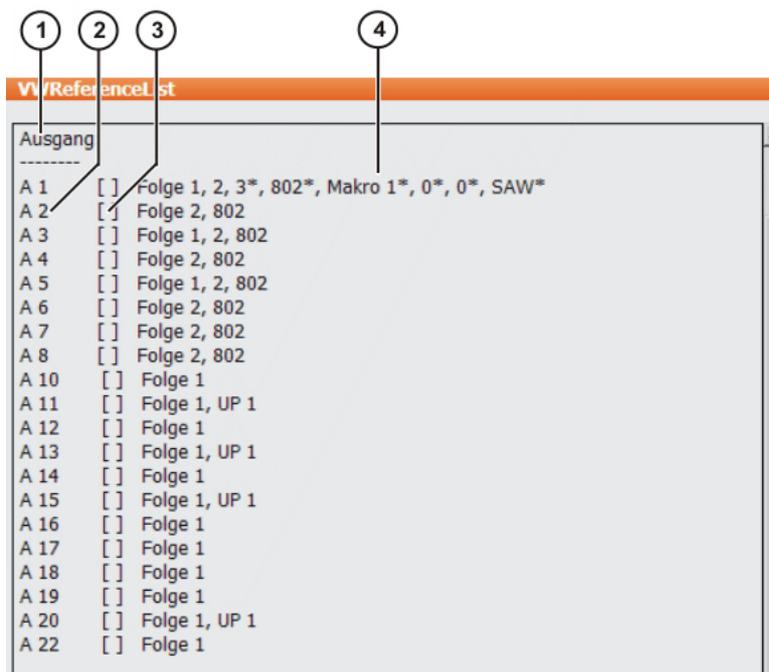


Fig. 12-6: VWReferenceList window

Item	Description
1	Ausgang (output) section of the reference list
2	Element of the reference list and element number
3	Long text name (shown in square brackets if one exists)
4	<p>Reference or use of the element</p> <p>If references are labeled with "*", the corresponding element is changed, e.g. setting of an output.</p>

The following softkeys are available:

Softkey	Description
Generate	Refreshes the displayed reference list.
Print	Prints the reference list as a text file.
Save...	Opens a window for selection of the target directory to which the reference list is saved as a text file.

The following elements are displayed in the reference list:

Element	Section
A	Output
ana	Analog output
anain	Analog input
bin	Binary output
binin	Binary input
E	Input
F	Flag
Folge	Cell
I	Counter
M	Merker
Makro	Macro
p	Process parameter
S	Sensor flag
t	Timer
T	Timer flag
UP	UP
VERR	Interlock

12.5 Interlock diagnosis

This function is used to check robot interlocks in order to avoid collisions.

12.5.1 Configuring interlock diagnosis

Procedure

1. Select the menu sequence **Diagnosis > Interlock diagnosis > Interlock configuration**.
2. Select the line with the interlock number that is to be edited.
3. Set the interlock type with **Type**.
4. Press **Inputs**, enter the number of the input and save by pressing Enter.
5. Press **Outputs**, enter the number of the output and save by pressing Enter.
6. Only in the case of the interlock type "Makro": press **Makro**, enter the number of the macro and save by pressing Enter.

Description

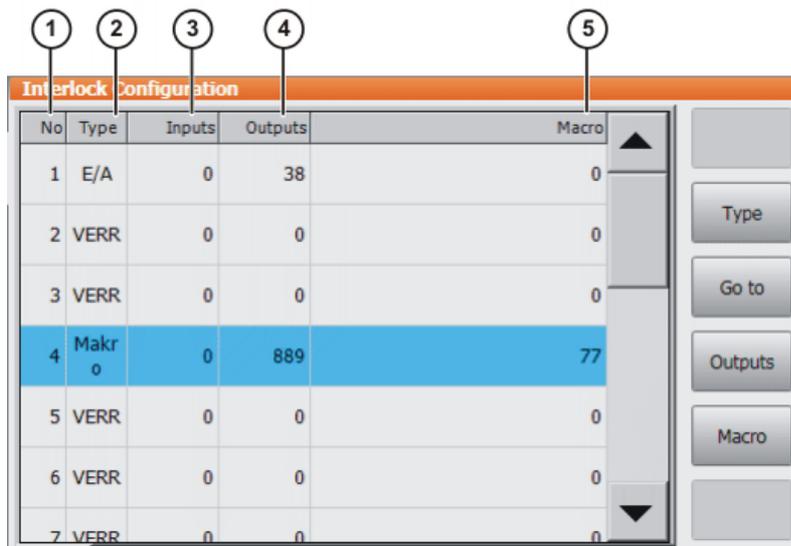


Fig. 12-7: Interlock configuration

Item	Description
1	Interlock number <ul style="list-style-type: none"> 1 ... 16
2	Interlock type <ul style="list-style-type: none"> I/O: I/O interlock VERR: interlock command, inline form "VERR" Makro: interlock macros
3	Number of the input <ul style="list-style-type: none"> 1 ... 4,096
4	Number of the output <ul style="list-style-type: none"> 1 ... 4,096
5	Number of the macro <ul style="list-style-type: none"> 1 ... 999

The following softkeys are available:

Softkey	Description
Type	Toggles between the interlock types.
Inputs	The selected cell is made available for editing.
Output	The selected cell is made available for editing.
Macro	The selected cell is made available for editing.

12.5.2 Carrying out interlock diagnosis

Precondition

- Interlock diagnosis has been configured.

Procedure

- Select the menu sequence **Diagnosis > Interlock diagnosis > Interlock check**.

Description

The following checks are carried out:

- Syntax check of the programmed robot interlocks
- Check of the inputs/outputs reserved for the robot interlock

The result of the interlock check is saved in a log file. Default path: C:\KRC\ROBOTER\LOG\VERRCHECK.LOG.

If programming errors are found, a window is opened in which the result of the interlock check is displayed.

12.6 Displaying diagnostic data about the kernel system

Description

The menu item **Diagnostic Monitor** makes it possible to display a wide range of diagnostic data concerning numerous software modules of the kernel system.

Examples:

- Module **Kcp3 Driver** (= drive for the smartPAD)
- Network driver

The data displayed depend on the selected module. The display includes states, fault counters, message counters, etc.

Procedure

1. Select the menu sequence **Diagnosis > Diagnostic Monitor**.
2. Select a software module in the **Module** box.
Diagnostic data are displayed for the selected module.

13 Installation

The robot controller is supplied with a Windows operating system and an operational version of the VW System Software (VSS). Therefore, no installation is required when starting up the industrial robot for the first time.

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 333)

13.1 Overview of the software components

Overview The following software components are used:

- VW System Software 8.1
- Windows XP embedded 3.0

13.2 Installing Windows and VSS (from image)

- Precondition**
- KUKA.Restore USB 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.Restore USB stick can be used. Installation from a different stick is not possible.

Procedure

1. Connect the KUKA.Restore 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 **Continue**.
5. Information about the installation and copyright are displayed. Confirm with **Continue**.
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 **Continue**.
7. The system suggests a robot type. Confirm with **Continue**.
Or: If the type suggested does not correspond to the type used, select **Browse** with the TAB key. In the MaDa directory, select the control cabinet used and the corresponding type. Confirm with **OK**. Then confirm with **Continue**.
8. Select which options are to be installed. Confirm with **Continue**.
Options:
 - **ASCII interface**
 - **User group 15**
 - **ProConOS**
 - **Profinet KRC-Nexxt**
9. A summary of the setup settings is displayed. Confirm with **Continue**. The VSS 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.

13.3 Installing additional software (via KUKA smartHMI)

This function can be used to install additional software. New programs and updates can be installed. The software is installed from the KUKA USB stick. Alternatively, it can also be installed via a network path.

The system checks whether the additional software is relevant for the VSS. 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 333)

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 VSS update from the network. This function cannot be used, however, to install the VSS update.

(>>> 13.4 "VSS update" Page 329)

Preparation

- Copy software from CD to KUKA USB stick



Caution!

The only USB stick that may be used is the KUKA USB stick. Data may be lost or modified if any other USB stick is used.

Procedure

1. Select the menu sequence **Setup > Install Additional Software**.
2. Press **New Software**. If a software package on the USB storage medium is not yet displayed, press **Refresh**.
3. Select the software to be installed and press **Install**. Answer the request for confirmation with **Yes**. The files are copied onto the hard drive.
4. If another additional software package is to be installed, repeat step 3.
5. Depending on the specific additional software, it may be necessary to reboot the controller. In this case, a corresponding prompt will be displayed. Confirm with **OK** and restart the robot controller. Installation is resumed and completed.

Description

The following softkeys are available:

Softkey	Description
New Software	All programs available for installation are displayed.
Back	Additional software already installed is displayed.
Refresh	Refreshes the display, e.g. after the USB stick has been connected
Install	Shows additional softkeys: <ul style="list-style-type: none"> ■ Yes: The selected software is installed. If it is necessary to reboot the controller, this is indicated by a message. ■ No: The software is not installed.

Softkey	Description
Configuration	<p>This softkey is only displayed if the New Software softkey has been pressed.</p> <p>Paths for the installation of additional software or VSS updates can be selected and saved here.</p> <p>Shows additional softkeys:</p> <ul style="list-style-type: none"> ■ Browse: A new path can be selected. ■ Save: Saves the displayed paths.
Uninstall	<p>Shows additional softkeys:</p> <ul style="list-style-type: none"> ■ Yes: The selected software is uninstalled. ■ No: The software is not uninstalled.

13.4 VSS update

Description

This function can be used to install VSS updates, e.g. from VSS 8.1.0 to VSS 8.1.1.

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 updates of additional software.
(>>> 13.3 "Installing additional software (via KUKA smartHMI)" Page 328)



Warning!

If machine data are reloaded after an update, the version of the machine data must correspond exactly to the VSS version. 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 VSS update:

- From the KUKA USB stick
(>>> 13.4.1 "Update from a USB storage medium" Page 329)
- From the network
(>>> 13.4.2 "Update from the network" Page 330)

13.4.1 Update from a USB storage medium



Caution!

The only USB stick that may be used is the KUKA USB stick. Data may be lost or modified if any other USB stick is used.

Procedure

1. Select the menu sequence **Setup > Software Update > Automatic**.
2. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing **Yes**.
3. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.
4. Wait until the computer has shut down completely. Then switch the controller back on.

Once the update has been completed, the computer is automatically shut down and rebooted.

**Caution!**

In the case of installation from a USB medium: the medium must not be removed until the LED on the USB medium is no longer lit. Otherwise, the medium could be damaged.

13.4.2 Update from the network

Description

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.

Installation is started on completion of the copying operation.

Preparation

Configure the network path from which the update installation is to be carried out:

1. Select the menu sequence **Setup > Install Additional Software**.
2. Press **New Software**.
3. Press **Configuration**.
4. Select the **Installationpath for KRC Update from network** box. Press **Browse**.
5. Select the desired network path. Press **Save**.
6. The selected path is displayed in the **Installationpath for KRC Update from network** box. Press **Save**.
7. Close the window.



It is only necessary to configure the network path once. It remains saved for subsequent updates.

Procedure

1. Select the menu sequence **Setup > Software Update > Net**.
2. A request for confirmation is displayed, asking if the update should be carried out. Confirm by pressing **Yes**.
Depending on the network utilization, the procedure may take up to 15 min.
3. A message is displayed, indicating that a cold start will be forced next time the system is booted. Switch the controller off.
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 Error messages of the positionally accurate robot model

No.	Message	Description
272	No robot number programmed	<p>Cause: The file ROBOTINFO.XML is missing on the RDC, e.g. after exchange of the RDC memory chip.</p> <p>Remedy: Create the file ROBOTINFO.XML on the RDC again.</p>
276	Wrong machine data for this robot type	<p>Cause: The machine data loaded in the robot controller do not correspond to the robot type. The robot must not be moved.</p> <p>Remedy: Load the correct machine data.</p>
384	Not enough memory: Cannot load precision robot model	<p>Cause: It was not possible to create the positionally accurate robot model due to insufficient memory capacity. Positioning accuracy is not possible.</p> <p>Remedy: Increase the memory capacity.</p>
417	Absolute accuracy model not active	<p>Cause: The positionally accurate robot model was deactivated.</p> <p>Remedy: Contact KUKA Roboter GmbH. (>>> 15 "KUKA Service" Page 333)</p>
1446	Value assignment inadmissible	<p>Cause:</p> <ul style="list-style-type: none"> ■ A system variable was assigned an inadmissible value, e.g. \$SPEED.ORI1 ≤0.0 or >\$SPEED_MA.ORI1. ■ An inadmissible value was assigned to \$BASE or \$TOOL. <p>Remedy: Check the system variables and assign a correct value. Acknowledge the message.</p>
2972	Checksum error File %1	<p>Cause: Error in the file indicated: The parameters are not applied.</p> <p>Remedy: Install the correct file.</p>
2973	Precision Robot Model Parameterfile wrong Serial-number	<p>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.</p> <p>Remedy:</p> <ul style="list-style-type: none"> ■ Use the correct XPid file. ■ Use the correct serial number.
2974	Precision robot: Model parameter not consistent with machine data	<p>Cause: The machine data have been modified. The positionally accurate robot model cannot be initialized.</p> <p>Remedy: Recalibrate the robot with KUKA.XROB RCS.</p>

No.	Message	Description
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.



Faults leading to production downtime should be reported to the local KUKA subsidiary within one hour of their occurrence.

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 VW System Software
- Optional software or modifications
- Archive of the software
- 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

Australia Marand Precision Engineering Pty. Ltd. (Agency)
153 Keys Road
Moorabbin
Victoria 31 89
Australia
Tel. +61 3 8552-0600
Fax +61 3 8552-0605
robotics@marand.com.au

Belgium	KUKA Automatisering + Robots N.V. Centrum Zuid 1031 3530 Houthalen Belgium Tel. +32 11 516160 Fax +32 11 526794 info@kuka.be www.kuka.be
Brazil	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 info@kuka-roboter.com.br
Chile	Robotec S.A. (Agency) Santiago de Chile Chile Tel. +56 2 331-5951 Fax +56 2 331-5952 robotec@robotec.cl www.robotec.cl
China	KUKA Flexible Manufacturing Equipment (Shanghai) Co., Ltd. Shanghai Qingpu Industrial Zone No. 502 Tianying Rd. 201712 Shanghai P.R. China Tel. +86 21 5922-8652 Fax +86 21 5922-8538 Franz.Poeckl@kuka-sha.com.cn www.kuka.cn
Germany	KUKA Roboter GmbH Zugspitzstr. 140 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 info@kuka-roboter.de www.kuka-roboter.de

France	KUKA Automatismes + Robotique SAS Techvallée 6, Avenue du Parc 91140 Villebon S/Yvette France Tel. +33 1 6931660-0 Fax +33 1 6931660-1 commercial@kuka.fr www.kuka.fr
India	KUKA Robotics, Private Limited 621 Galleria Towers DLF Phase IV 122 002 Gurgaon Haryana India Tel. +91 124 4148574 info@kuka.in www.kuka.in
Italy	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 kuka@kuka.it www.kuka.it
Japan	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 info@kuka.co.jp
Korea	KUKA Robot Automation Korea Co. Ltd. 4 Ba 806 Sihwa Ind. Complex Sung-Gok Dong, Ansan City Kyunggi Do 425-110 Korea Tel. +82 31 496-9937 or -9938 Fax +82 31 496-9939 info@kukakorea.com

Malaysia	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 info@kuka.com.my
Mexico	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 info@kuka.com.mx
Norway	KUKA Sveiseanlegg + Roboter Bryggeveien 9 2821 Gjøvik Norway Tel. +47 61 133422 Fax +47 61 186200 geir.ulsrud@kuka.no
Austria	KUKA Roboter Austria GmbH Vertriebsbüro Österreich Regensburger Strasse 9/1 4020 Linz Austria Tel. +43 732 784752 Fax +43 732 793880 office@kuka-roboter.at www.kuka-roboter.at
Poland	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 ServicePL@kuka-roboter.de

Portugal 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
kuka@mail.telepac.pt

Russia OOO KUKA Robotics Rus
Webnaja ul. 8A
107143 Moskau
Russia
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
kuka-robotics.ru

Sweden KUKA Svetsanläggningar + Robotar AB
A. Odhners gata 15
421 30 Västra Frölunda
Sweden
Tel. +46 31 7266-200
Fax +46 31 7266-201
info@kuka.se

Switzerland KUKA Roboter Schweiz AG
Riedstr. 7
8953 Dietikon
Switzerland
Tel. +41 44 74490-90
Fax +41 44 74490-91
info@kuka-roboter.ch
www.kuka-roboter.ch

Spain 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
Comercial@kuka-e.com
www.kuka-e.com

South Africa	Jendamark Automation LTD (Agency) 76a York Road North End 6000 Port Elizabeth South Africa Tel. +27 41 391 4700 Fax +27 41 373 3869 www.jendamark.co.za
Taiwan	KUKA Robot Automation Taiwan Co. Ltd. 136, Section 2, Huanjung E. Road Jungli City, Taoyuan Taiwan 320 Tel. +886 3 4371902 Fax +886 3 2830023 info@kuka.com.tw www.kuka.com.tw
Thailand	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 atika@ji-net.com www.kuka-roboter.de
Czech Republic	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 support@kuka.cz
Hungary	KUKA Robotics Hungaria Kft. Fő út 140 2335 Taksony Hungary Tel. +36 24 501609 Fax +36 24 477031 info@kuka-robotics.hu

USA KUKA Robotics Corp.
22500 Key Drive
Clinton Township
48036
Michigan
USA
Tel. +1 866 8735852
Fax +1 586 5692087
info@kukarobotics.com
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

Index

Symbole

_TYP 253
 _TYPE 254
 #BSTEP 146
 #CSTEP 147
 #IGNORE 175
 #ISTEP 146
 #MSTEP 146
 #PSTEP 147
 \$ 247
 \$ADVANCE 147
 \$CIRC_MODE 177, 264
 \$CIRC_TYPE 176
 \$COOLDOWN_TIME 126
 \$ORI_TYPE 160, 174
 \$PAL_MODE 84
 \$PATHTIME 269
 \$PRO_IP 321
 \$PRO_MODE 147
 \$ROBRUNTIME 79
 \$TOOL_DIRECTION 95
 \$WARMUP_CURR_LIMIT 126
 \$WARMUP_MIN_FAC 127
 \$WARMUP_RED_VEL 126
 \$WARMUP_SLEW_RATE 127
 \$WARMUP_TIME 126

Zahlen

2004/108/EC 39
 2006/42/EC 39
 3-point method 103
 89/336/EEC 39
 95/16/EC 39
 97/23/EC 39

A

A/M/F (menu item) 203
 ABC 2-Point method 101
 ABC World method 99
 Accessories 15, 17
 Actual position 65
 Advance run 147
 Advance run stop 269
 Amplitude 224
 Ana konst (menu item) 217
 ana kst+p (menu item) 217
 Ana vprop (menu item) 217
 Analog inputs 276
 Analog outputs 277
 ANIN 276
 ANOUT 277
 Applied norms and regulations 39
 Approximate positioning 159, 186, 187, 188, 190, 192
 Archive (menu item) 154
 ARCHIVE.ZIP 154, 155
 Archiving 154
 Areas of validity 249

ASCII Mode (menu item) 145
 AUT and EXT Consistency 137
 Automatic External 129
 Automatic External mode, starting 150
 Automatic mode 35
 Auxiliary point 158, 259, 260
 Axis range 19
 Axis range limitation 28
 Axis range monitoring 28

B

Backward motion 149
 Base calibration 102
 BASE coordinate system 53, 102
 Block function 194
 Block selection 149, 151, 152, 166
 BRAKE 284
 Brake defect 30
 Braking distance 19
 Braking the robot motion 284
 Branch, conditional 272
 BS A/F= (menu item) 228
 BS bin/ana= (menu item) 228

C

Calibration 95
 Calibration tolerances, defining 127
 Calibration, base 102
 Calibration, external TCP 105
 Calibration, fixed tool 105
 Calibration, linear unit 109
 Calibration, tool 95
 Calibration, workpiece 105
 Caller stack 321
 Caller Stack (menu item) 321
 CASE 274
 CAST_FROM 299
 CAST_TO 299
 CCLOSE 299
 CE mark 18
 CELL.SRC 129, 150
 CHANNEL 299
 CIOCTL 299
 CIR motion 188
 CIR, motion type 158
 CIRC 259
 CIRC_REL 260
 Circular angle 259, 261
 Circular motion 259, 260
 Cleaning work 36
 Cold start 49
 Comment (menu item) 152
 Compare (menu item) 207
 Comparison, data from kernel system and hard drive 137
 Conditional branch 272
 Configuration 115
 Configure (menu item) 118

- Connecting cables 15, 17
 - Connection manager 42
 - Constants 251
 - CONTINUE 269
 - Continuous Path 157
 - Coordinate system for jog keys 46
 - Coordinate system for Space Mouse 45
 - Coordinate systems 53
 - Coordinate systems, angles 55
 - Coordinate systems, orientation 55
 - COPEN 299
 - Copy 153
 - Counterbalancing system 36
 - CP motions 157
 - CREAD 299
 - Creating a new program 142
 - Cut 153
 - CWRITE 299
 - Cyclical flags, displaying 68
- D**
- Danger zone 19
 - DAT 246
 - Data list 246
 - Data types 248
 - Data, restoring 155
 - DECL 251
 - Declaration of conformity 18
 - Declaration of incorporation 17, 18
 - Decommissioning 37
 - DEFAULT 274
 - Deleting mastering 94
 - Detail view (ASCII mode), activating 145
 - Diagnosis 317
 - Diagnostic Monitor (menu item) 325
 - Dial gauge 93
 - Directory structure 141
 - Disabling the robot controller 52
 - Display (menu item) 77
 - Displaying a variable, single 75, 76
 - Displaying the logbook 317
 - Displaying variables, in overview 77
 - Displaying, robot controller information 78
 - Displaying, robot information 78
 - Disposal 37
 - DISTANCE 289
 - Documentation, industrial robot 13
 - Dry run 151
- E**
- EC declaration of conformity 18
 - Edit (softkey) 46
 - Editing a program 152
 - Electronic Mastering Device 88
 - ELSE 272
 - EMC Directive 18, 39
 - EMD 88
 - EMERGENCY STOP 42
 - EMERGENCY STOP button 25, 33
 - EMERGENCY STOP device 25, 26, 30
 - EMERGENCY STOP, external 26, 33
 - EMERGENCY STOP, local 33
 - 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 27
 - Enabling switch 43
 - Enabling switches 26
 - ENDFOR 270
 - ENDIF 272
 - Endless loop 273
 - ENDLOOP 273
 - ENDSPLINE 261
 - ENDSWITCH 274
 - ENDWHILE 276
 - ENUM 252
 - Enumeration type 252
 - Error messages, positionally accurate robot 331
 - ERSYSROOT coordinate system 110
 - Event planner (menu item) 136
 - Example programs 312
 - EXIT 270, 273
 - Exiting, VSS 49
 - External axes 17, 19, 66, 79
- F**
- F 284
 - FALSE 283
 - Faults 31
 - FB ONL (menu item) 209
 - FB PSPS (menu item) 210
 - File list 141
 - File, renaming 143
 - Filter 142
 - Find 153
 - First mastering 88
 - Fixed tool, calibration 105
 - Flags, displaying 68
 - FLANGE coordinate system 54, 96
 - Fonts 246
 - FOR 275
 - FOR ... TO ... ENDFOR 270
 - Function test 32
 - Functions 247
- G**
- General safety measures 30
 - Global 250
 - GLOBAL (interrupt declaration) 285
 - GOTO 271
 - Guard interlock 24
- H**
- HALT 272

Hardware, options 81
 Hazardous substances 37
 Header 141
 Hibernate 49
 HOV 59

I

I-Bus alternative (menu item) 216
 i/bin= (menu item) 204
 Identification plate 43
 IF ... THEN ... ENDIF 272
 Increment 64
 Incremental jogging 63
 Indirect method 104
 Industrial robot 15, 17
 Info (menu item) 78
 Inline forms 185
 Inputs/outputs, analog 69
 Inputs/outputs, Automatic External 74, 130
 Inputs/outputs, binary 71, 116
 Inputs/outputs, digital 66
 Inputs/outputs, gun 73, 115
 Installation 327
 Intended use 17
 Interlock 209, 210
 Interlock check (menu item) 324
 Interlock configuration (menu item) 323
 Interlock diagnosis 323
 Interlocking (menu item) 214
 INTERRUPT 284, 286
 Interrupt 284
 Interrupt program 284
 Introduction 13

J

Jog keys 42, 55, 60
 Jog mode 27, 30
 Jog mode "Jog keys" 57
 Jog mode "Space Mouse" 58
 Jog mode, activating 59
 Jog override 59
 Jogging, axis-specific 55, 60
 Jogging, Cartesian 55, 60, 63
 Jogging, external axes 64
 Jogging, robot 55
 Jump 271
 Jump (menu item) 237
 Jump labels 237

K

KCIR motion 191
 KCIR, description 189
 KCP 19, 31, 41
 Keyboard 42
 Keyboard key 42
 Keyboard, external 31
 Keypad 47
 Keywords 247
 Kinematics group 46, 57
 KLIN motion 190
 KLIN, description 189

KRL syntax 245
 KUKA Control Panel 41
 KUKA Customer Support 78, 333
 KUKA smartHMI 45
 KUKA smartPAD 19, 41
 KUKA.Load 111
 KUKA.LoadDataDetermination 112

L

Label (menu item) 237
 Labeling 29
 Language 51
 Liability 17
 LIN 257
 LIN motion 186
 LIN motion, sensor-monitored 193
 LIN_REL 257
 LIN, motion type 158
 Linear motion 257
 Linear unit 17, 110
 Linear unit, calibrating 109
 Linebreak (menu item) 146
 Load data 111
 Logbook 317
 Logbook, configuring 319
 Logic Consistency 137, 138
 LOOP ... ENDLOOP 273
 Loss of mastering 88, 92
 Low Voltage Directive 18

M

Machine data 33, 34, 79, 81, 329
 Machinery Directive 18, 39
 Maintenance 36
 MAKRO (menu item) 231
 MAKRO/UP-Loop (menu item) 233
 MakroSAW 149, 240
 MakroSPS 240
 MakroStep 241
 MakroTrigger 243
 Manipulator 15, 17, 19, 22
 Manual mode 34
 Mastering 85
 Mastering marks 87
 Mastering methods 86
 Measurement Points (menu item) 78
 Mechanical axis range limitation 28
 Mechanical end stops 28
 Menu, calling 48
 Messages 331
 Mode selection 23, 24
 Modifying a variable 75
 Modifying coordinates 194
 Modifying motion parameters 194, 195
 Modifying variables 77
 Module 246
 Monitoring, velocity 27
 Motion programming, basic principles 157
 Motion types 157
 Mouse, external 31

N

Name, control PC 79
Name, robot 79
Names 247
Navigator 141
Non-rejecting loop 273
Numeric entry, external TCP 107
Numeric input, base 104
Numeric input, linear unit 111
Numeric input, tool 102

O

Offset 88, 91
Online optimizing 137, 139
Operand, inserting 202
Operating hours 79
Operating hours meter 79
Operating mode, changing 53
Operation 41
Operator 19, 20, 52
Operator safety 23, 24, 30
Operator safety acknowledgement 82
Operator, inserting 202
Options 15, 16, 17
Orientation control, LIN, CIR 160
Orientation control, SPLINE 174
Overload 30
Override 59, 147
Override (menu item) 65
Overview of the industrial robot 15

P

Palletizing robots 84, 97, 102
Panic position 26
Paste 153
PATH 292, 295
Payload data 112
Payload data (menu item) 112
Performance Level 23
Peripheral contactor 82
Personnel 20
Plant integrator 19
PLC instructions, displaying 145
Point-to-point 157
Point-to-point motion 255, 256
Position Flag (menu item) 240
Positionally accurate robot, checking activation 84
Positionally accurate robot, error messages 331
Positioner 17
POV 147
Pre-mastering position 87
Pressure Equipment Directive 36, 39
Preventive maintenance work 36
Printing a program 154
Priority 285, 291, 294, 296
Process parameters 73
Product description 15
ProfiNet 82
Program lines, deleting 153
Program management 141

Program override 147
Program run mode, selecting 146
Program run modes 146
Program, creating 142
Program, selecting 143
Program, starting 148
Programmer 52
Programming, Expert 245
Programming, inline forms 185
Programming, KRL syntax 245
Programming, User 185
Protective equipment 27
PTP 255
PTP motion 185
PTP_REL 256
PTP, motion type 157
PULSE 279
Pulse 279
Pulse (menu item) 208

R

Rating plate 81
Reaction distance 19
Recommissioning 32, 81
Reference list, displaying 322
Rejecting loop 276
Release device 28
Renaming a file 143
Renaming the base 109
Renaming the tool 109
Repair 36
REPEAT ... UNTIL 273
Replace 153
Resetting a program 150
Restart, Windows 51
RESUME 287
RETURN 283
Robot controller 15, 17
Robot data (menu item) 79
ROBROOT coordinate system 53
ROBROOT kinematic system 110

S

S_Archive (menu item) 156
Safeguards, external 29
Safety 17
Safety controller 23
Safety functions, overview 23
Safety instructions 13
Safety measures 30
Safety STOP 0 19
Safety STOP 1 19
Safety STOP 2 19
Safety stop, external 27
Safety zone 19, 21, 22
Safety, general 17
SCIRC 263
Search run, description 189
SEC 275
Selecting the base 59
Selecting the tool 59

- Serial number 79
 - Service life 79
 - Service, KUKA Roboter 333
 - Shutdown (menu item) 49
 - SIGNAL 282
 - Signal diagrams 134
 - Simulation 35
 - Single (menu item) 75, 76
 - Single point of control 37
 - Singularities 182
 - SLIN 262
 - smarHMI 16, 45
 - smartPAD 19, 41
 - Software 15, 17
 - Software components 15, 327
 - Software limit switch 95
 - Software limit switches 27, 30
 - Space Mouse 42, 55, 60, 62, 63
 - SPL 265
 - SPLINE 295
 - SPLINE ... ENDSPLINE 261
 - Spline, motion type 163
 - SPOC 37
 - SRC 246
 - SREAD 299
 - Standard archive 155
 - Start backwards key 42
 - Start key 42, 43
 - Start type, VSS 49
 - Start-up 32, 81
 - Start-up mode 34
 - Starting a program, backwards 149
 - Starting a program, manual 148
 - Starting the VSS 48
 - Status 179
 - Status bar 45, 46, 141
 - STEP 270
 - Step (menu item) 241
 - STOP 0 18, 19
 - STOP 1 18, 19
 - STOP 2 18, 19
 - Stop category 0 19
 - Stop category 1 19
 - Stop category 2 19
 - STOP key 42
 - Stop reactions 22
 - Stopping a program 149, 151
 - Stopping distance 19, 22
 - Storage 37
 - Storage capacities 79
 - String variable length after initialization 302
 - String variable length in the declaration 301
 - String variable, deleting contents 302
 - String variables 301
 - String variables, comparing contents 304
 - String variables, copying 305
 - String variables, extending 303
 - String variables, searching 303
 - STRUC 254
 - Structure type 254
 - Submit interpreter 46
 - Submit interpreter, starting 129
 - Submit interpreter, stopping 128
 - Subprograms 247
 - SUCHLAUF (menu item) 193
 - Supplementary load data (menu item) 113
 - Support request 333
 - SWITCH ... CASE ... ENDSWITCH 274
 - Switching on the robot controller 48
 - SWRITE 299
 - Symbols 246
 - System integrator 18, 19, 20
- T**
- t= (menu item) 205
 - t=STOP (menu item) 206
 - T1 19
 - T1 and T2 Consistency 137
 - T2 19
 - TCP 96
 - TCP, external 105
 - Teach pendant 15, 17
 - Teaching 194
 - Technology keys 42
 - Technology motion, programming 189
 - Technology packages 79
 - Terms used, safety 18
 - Time block 265
 - TIME_BLOCK 265
 - Timer, starting 205
 - Timer, stopping 206
 - Too change 216
 - Tool calibration 95
 - Tool Center Point 96
 - TOOL coordinate system 53, 95
 - Tool direction 95
 - Touch screen 41, 47
 - Trademarks 13
 - Training 13
 - Transforming coordinates 195
 - Transport position 32
 - Transportation 32
 - TRIGGER 289, 292
 - TRIGGER, for spline 295
 - Turn 179
 - Turn-tilt table 17
 - Type, robot 79
 - Type, robot controller 79
- U**
- Unmastering 94
 - UNTIL 273
 - UP (menu item) 232
 - Update, VSS 329
 - Use, contrary to intended use 17
 - Use, improper 17
 - User 19, 20
 - USER (menu item) 236
 - User group 51
 - User interface 45

V

Variable overview, configuring 117
VARSTATE() 299
Velocity 59, 147
Velocity monitoring 27
Version, kernel system 79
Version, operating system 79
Version, robot controller 79
Version, user interface 79
Voltage 70, 217
VW_USR_R 307
VW_USR_S 307

W

WAIT 275
Wait condition, displaying 320
WAIT FOR 275
Wait function, signal-dependent 211
Wait onl/bis (menu item) 211
WAIT SEC 275
Wait time 275
Wait Time (menu item) 213
Warm-up 124
Warnings 13
Weave (menu item) 223
Weave length 224
Weave patterns 223
Weave patterns, height 224
Weld velocity 223
WHILE ... ENDWHILE 276
Windows, restart 51
Working range limitation 28
Workspace 19, 21, 22
Workspace monitoring, bypassing 65
Workspaces, axis-specific 119
Workspaces, Cartesian 119
Workspaces, cubic 119
WORLD coordinate system 53

X

XYZ 4-Point method 97
XYZ Reference method 98

Z

Zange = (menu item) 235

