

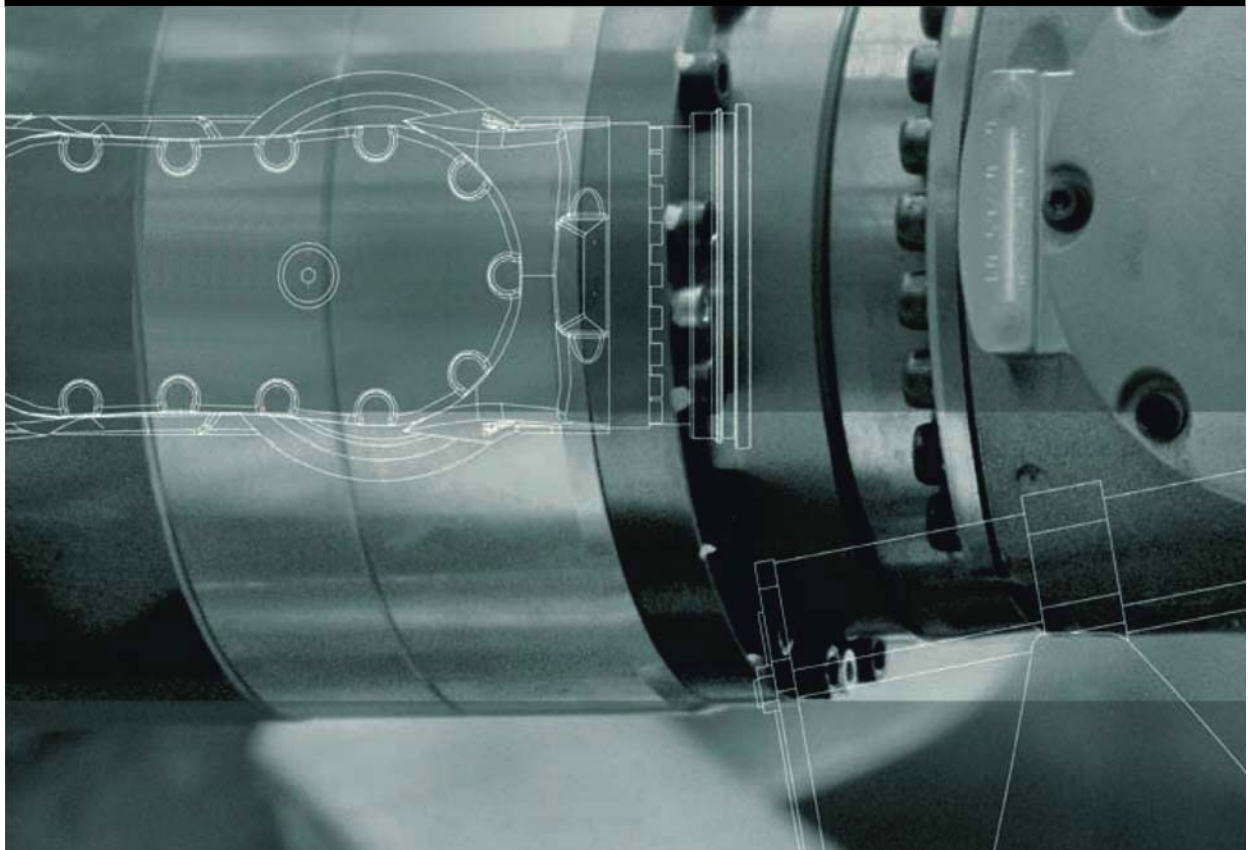
KUKA

KUKA System Technology

KUKA Roboter GmbH

CODESYS Library

For KUKA.PLC mxAutomation 2.1



Issued: 14.12.2015

Version: CODESYS Library mxAutomation 2.1 V2

© Copyright 2015
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 CODESYS Library mxAutomation 2.1 (PDF) en
Book structure:	CODESYS Library mxAutomation 2.1 V2.1
Version:	CODESYS Library mxAutomation 2.1 V2

Contents

1	Introduction	7
1.1	Target group	7
1.2	Industrial robot documentation	7
1.3	Representation of warnings and notes	7
1.4	Terms used	8
2	Product description	9
2.1	Overview	9
2.2	Intended use	9
3	Safety	11
4	Installation	13
4.1	System requirements	13
5	Configuration	15
5.1	Configuration in WorkVisual – overview	15
6	Programming	17
6.1	Instructions for programming	17
6.2	Overview of function blocks	17
6.3	Frequently used input/output signals in the function blocks	20
6.3.1	Input signals	20
6.3.2	Output signals	20
6.3.3	Signal sequence for execution of ExecuteCmd	21
6.4	Frequently used input/output signals in the MC function blocks	22
6.4.1	Input signals	22
6.4.2	Output signals	22
6.4.3	Signal sequence for execution of Execute	23
6.5	Structures for motion programming (STRUCT)	23
6.6	Integer variables	26
6.7	Data of a Cartesian workspace	28
6.8	Data of an axis-specific workspace	28
6.9	Programming tips for KUKA.PLC mxAutomation	29
6.9.1	Programming example (template MxA_CODESYS_Template_PN)	30
6.10	Administrative functions	31
6.10.1	Reading PLC-specific communication into a non-PLC-specific structure	31
6.10.2	Writing a non-PLC-specific structure into PLC-specific communication	32
6.10.3	Initializing the mxA interface	32
6.10.4	Setting the program override (POV)	33
6.10.5	Activating and reading Automatic External signals from the robot controller	34
6.10.6	Reading the current robot position	36
6.10.7	Reading the current axis position	37
6.10.8	Reading the current path velocity	38
6.10.9	Reading the current axis velocity	38
6.10.10	Reading the current robot acceleration	39
6.10.11	Reading a digital input	40
6.10.12	Reading digital inputs 1 to 8	40

6.10.13	Reading multiple digital inputs	41
6.10.14	Reading a digital output	42
6.10.15	Writing a digital output	42
6.10.16	Writing digital outputs 1 to 8	43
6.10.17	Reading an analog input	44
6.10.18	Reading an analog output	44
6.10.19	Writing an analog output	45
6.10.20	Wait statement (read digital input)	46
6.10.21	Selecting the tool, base and interpolation mode	46
6.10.22	Reading TOOL data	47
6.10.23	Writing TOOL data	48
6.10.24	Reading BASE data	49
6.10.25	Writing BASE data	50
6.10.26	Reading the load data	51
6.10.27	Writing load data	51
6.10.28	Reading the software limit switches of the robot axes	53
6.10.29	Reading the software limit switches of the external axes	53
6.10.30	Writing the software limit switches of the robot axes	54
6.10.31	Writing the software limit switches of the external axes	55
6.10.32	Declaring interrupts	56
6.10.33	Activating interrupts	57
6.10.34	Deactivating interrupts	58
6.10.35	Reading the state of an interrupt	59
6.10.36	Activating a path-related switching action (TRIGGER WHEN DISTANCE)	60
6.10.37	Activating a path-related switching action (TRIGGER WHEN PATH)	61
6.10.38	Canceling a program	63
6.10.39	Stopping the robot	64
6.10.40	Continuing a program	64
6.10.41	Automatically starting function blocks and signals	65
6.11	Functions for activating motions	66
6.11.1	Moving to a Cartesian position with a LIN motion	66
6.11.2	Moving to a Cartesian position with a LIN_REL motion	67
6.11.3	Moving to a Cartesian position with a PTP motion	69
6.11.4	Moving to a Cartesian position with a PTP_REL motion	70
6.11.5	Moving to an axis-specific position with a PTP motion	72
6.11.6	Moving to a Cartesian position with a CIRC motion	73
6.11.7	Moving to a Cartesian position with a CIRC_REL motion	75
6.11.8	Jogging to a relative end position	78
6.11.9	Jogging to a relative end position in the TOOL coordinate system	79
6.11.10	Jogging to an end position	80
6.12	Functions for activating motions (PLC OPEN-compliant)	83
6.12.1	Moving to a Cartesian position with a LIN motion	83
6.12.2	Moving to a Cartesian position with a LIN_REL motion	84
6.12.3	Moving to a Cartesian position with a PTP motion	86
6.12.4	Moving to a Cartesian position with a PTP_REL motion	87
6.12.5	Moving to an axis-specific position with a PTP motion	89
6.12.6	Moving to a Cartesian position with a CIRC motion	90
6.12.7	Moving to a Cartesian position with a CIRC_REL motion	93
6.13	Diagnostic functions	95

6.13.1	Reading the current state of the mxA interface	95
6.13.2	Reading error messages of the mxA interface	96
6.13.3	Resetting error messages of the mxA interface	96
6.13.4	Reading error messages of the robot controller	96
6.13.5	Reading diagnostic signals	97
6.13.6	Reading and acknowledging error states	99
6.14	Special functions	101
6.14.1	Reading system variables	101
6.14.2	Writing system variables	101
6.14.3	Calling a brake test	103
6.14.4	Calling a mastering test	104
6.14.5	Reading the safety controller signals	106
6.14.6	Reading the state of the TouchUp status keys	107
6.14.7	Teaching points	107
6.14.8	Modifying settings for the advance run	108
6.14.9	Reading values from KRC_SetAdvance	109
6.14.10	Calculating the Cartesian robot position from the axis angles	110
6.14.11	Calculating axis angles from the Cartesian robot position	111
6.14.12	Initializing a conveyor	112
6.14.13	Activating a conveyor	113
6.14.14	Tracking a workpiece on the conveyor	114
6.14.15	Picking up a workpiece from the conveyor	115
6.14.16	Activating interrupts for monitoring	117
6.14.17	Deactivating interrupts for monitoring	118
6.14.18	Activating a motion along a vector	119
6.14.19	Deactivating KRC_VectorMoveOn	120
6.14.20	Configuring Cartesian workspaces	121
6.14.21	Reading the configuration of Cartesian workspaces	122
6.14.22	Configuring axis-specific workspaces	123
6.14.23	Reading the configuration of axis-specific workspaces	124
6.14.24	Reading the status of the workspaces	124
7	Messages	127
7.1	Error messages of the mxA interface in the robot interpreter	127
7.2	Error messages of the mxA interface in the submit interpreter	131
7.3	Errors in the function block	137
7.4	ProConOS errors	140
8	KUKA Service	147
8.1	Requesting support	147
8.2	KUKA Customer Support	147
	Index	155

1 Introduction

1.1 Target group

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

- Knowledge of the robot controller system
- Advanced PLC programming skills
- Advanced knowledge of field bus interfaces



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 System Software
- Instructions for options and accessories
- Parts catalog on storage medium

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

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



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



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



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



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



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

This warning draws attention to procedures which serve to prevent or remedy emergencies or malfunctions:



Procedures marked with this warning **must** be followed exactly.

Notices

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



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

1.4 Terms used

Term	Description
Axis group	Depending on the machine data configuration, an axis group contains the following axes: <ul style="list-style-type: none"> ■ Robot axes A1 to A6 ■ External axes E1 to E6 (synchronous or asynchronous)
FIFO	Method used to process a data memory <ul style="list-style-type: none"> ■ First In First Out: the elements saved first are taken first from the memory.
KR C	KUKA Robot Controller
KRL	KUKA robot programming language (KUKA Robot Language)
KUKA smartHMI	User interface of the KUKA robot controller (KUKA smart Human-Machine Interface)
KUKA smartPAD	Hand-held operating and programming device for the KUKA industrial robot
mxA interface	KUKA.PLC mxAutomation CODESYS technology package on the robot controller
PROFINET	PROFINET is an Ethernet-based field bus.
Robot interpreter	The robot interpreter is a process that works synchronously in which the current robot program is executed.
BCO run	The robot is moved to the coordinates of the motion block in which the block pointer is situated. In this way, the robot position is made to match the coordinates of the current point.
PLC	Programmable Logic Controller
CODESYS V3.5 SP4	Development environment for CODESYS controllers
Submit interpreter	The Submit interpreter is a cyclical logic program that runs in parallel with the motion program on the robot controller.
WorkVisual	Engineering environment for KR C4-controlled robot cells

2 Product description

2.1 Overview

The CODESYS library contains function blocks for programming automation tasks with the CODESYS V3.5 SP4 software.

Components	<p>The following software components are included in the CODESYS library:</p> <ul style="list-style-type: none"> ■ Function blocks for CODESYS V3.5 SP4 (Library PLC\CODESYS folder) ■ Programming templates for CODESYS V3.5 SP4 (Template PLC\CODESYS folder) <ul style="list-style-type: none"> ■ For PROFINET: MxA_CODESYS_Template_PN ■ For EtherNet/IP: MxA_CODESYS_Template_EIP ■ For EtherCAT bridge: MxA_CODESYS_Template_ECat_Visu
Communication	For data exchange between the PLC and the robot controller, PROFINET, EtherNet/IP or EtherCAT bridge can be used.
WorkVisual	<p>The following software is required for configuring the field buses and mapping the field bus signals:</p> <ul style="list-style-type: none"> ■ WorkVisual 4.0

2.2 Intended use

The online part of KUKA.PLC mxAutomation may only be used on a KR C4 robot controller with the following software:

- KUKA System Software 8.3
- KUKA.PLC ProConOS 4.1
- KUKA.ProfiNet 3.1 or KR C4 EtherNet/IP 2.0 or EtherCAT bridge
- KUKA.ConveyorTech 6.0
- KUKA.VectorMove 1.0

The offline part of KUKA.PLC mxAutomation is intended for use with CODESYS V3.5 SP4 or higher.

Any other or additional use is considered misuse and is not allowed. The manufacturer cannot be held liable for any resulting damage. The risk lies entirely with the user.

Operation in accordance with the intended use also requires compliance with the start-up and configuration instructions in this documentation.

Misuse	<p>Any use or application deviating from the intended use is deemed to be misuse and is not allowed. This includes e.g.:</p> <ul style="list-style-type: none"> ■ Incorrect configuration (not in compliance with this documentation). This might result in the robot executing different actions from those planned by the PLC programmer. ■ Use in a programming environment other than CODESYS V3.5 SP4 or higher.
---------------	---

3 Safety

This documentation contains safety instructions which refer specifically to the software described here.

The fundamental safety information for the industrial robot can be found in the “Safety” chapter of the Operating and Programming Instructions for System Integrators or the Operating and Programming Instructions for End Users.



The “Safety” chapter in the operating and programming instructions of the KUKA System Software (KSS) must be observed. Death to persons, severe injuries or considerable damage to property may otherwise result.

4 Installation

4.1 System requirements

Hardware

Robot controller:

- KR C4
- Or KR C4 compact

External PLC:

- CODESYS controller



The minimum requirements on the external PLC are based on an average test program that generates about 1024 kB of code which is saved on the PLC and has to be executed. Larger, more complex applications may require more performance. It is advisable to contact KUKA Service in such cases.

Software

Robot controller:

- KUKA System Software 8.3

The following KRL resources must be free:

KRL resource	Number
I/Os	2 049 ... 4 080

- KUKA.PLC ProConOS 4.1
- Software for the field bus used:
 - KUKA.ProfiNet 3.1
 - Or KR C4 EtherNet/IP 2.0

Option with ConveyorTech:

- KUKA.ConveyorTech 6.0

Option with VectorMove:

- KUKA.VectorMove 1.0

Standard laptop/PC:

- WorkVisual 4.0
- CODESYS V3.5 SP4 or higher

5 Configuration

5.1 Configuration in WorkVisual – overview


Step	Description
1	Install the mxAutomation option package in WorkVisual.
2	Load the project from the robot controller.
3	Insert PROCONOS in the project.
4	Insert the mxAutomation option package into the project.
5	Insert the catalog element for the field bus used into the project. Note: WorkVisual regards the catalog element as a device from an option package. In this catalog element, the I/O mapping is already preconfigured. The start addresses of the inputs and outputs must not be modified, as mxAutomation will otherwise not work.
6	Transfer the project from WorkVisual to the robot controller.





Information about procedures in WorkVisual is contained in the WorkVisual documentation.

6 Programming

6.1 Instructions for programming

 Information about programming CODESYS is contained in the documentation of this software.

 Only the function blocks contained in the scope of supply of KU-KA.PLC mxAutomation may be used in an mxA robot program.

 The **MxA_CODESYS_Template_PN** template supplied on the USB stick (**Template** folder) contains all the function blocks and a programming example with the fundamental function blocks. It is advisable to use this template for creating an mxAutomation robot program.

6.2 Overview of function blocks

Administrative functions	
KRC_ReadAxisGroup	(>>> 6.10.1 "Reading PLC-specific communication into a non-PLC-specific structure" Page 31)
KRC_WriteAxisGroup	(>>> 6.10.2 "Writing a non-PLC-specific structure into PLC-specific communication" Page 32)
KRC_Initialize	(>>> 6.10.3 "Initializing the mxA interface" Page 32)
KRC_SetOverride	(>>> 6.10.4 "Setting the program override (POV)" Page 33)
KRC_AutomaticExternal	(>>> 6.10.5 "Activating and reading Automatic External signals from the robot controller" Page 34)
KRC_ReadActualPosition	(>>> 6.10.6 "Reading the current robot position" Page 36)
KRC_ReadActualAxisPosition	(>>> 6.10.7 "Reading the current axis position" Page 37)
KRC_ReadActualVelocity	(>>> 6.10.8 "Reading the current path velocity" Page 38)
KRC_ReadActualAxisVelocity	(>>> 6.10.9 "Reading the current axis velocity" Page 38)
KRC_ReadActualAcceleration	(>>> 6.10.10 "Reading the current robot acceleration" Page 39)
KRC_ReadDigitalInput	(>>> 6.10.11 "Reading a digital input" Page 40)
KRC_ReadDigitalInput1To8	(>>> 6.10.12 "Reading digital inputs 1 to 8" Page 40)
KRC_ReadDigitalInputArray	(>>> 6.10.13 "Reading multiple digital inputs" Page 41)
KRC_ReadDigitalOutput	(>>> 6.10.14 "Reading a digital output" Page 42)
KRC_WriteDigitalOutput	(>>> 6.10.15 "Writing a digital output" Page 42)
KRC_WriteDigitalOutput1To8	(>>> 6.10.16 "Writing digital outputs 1 to 8" Page 43)
KRC_ReadAnalogInput	(>>> 6.10.17 "Reading an analog input" Page 44)
KRC_ReadAnalogOutput	(>>> 6.10.18 "Reading an analog output" Page 44)
KRC_WriteAnalogOutput	(>>> 6.10.19 "Writing an analog output" Page 45)
KRC_SetCoordSys	(>>> 6.10.21 "Selecting the tool, base and interpolation mode" Page 46)
KRC_ReadToolData	(>>> 6.10.22 "Reading TOOL data" Page 47)
KRC_WriteToolData	(>>> 6.10.23 "Writing TOOL data" Page 48)
KRC_ReadBaseData	(>>> 6.10.24 "Reading BASE data" Page 49)
KRC_WriteBaseData	(>>> 6.10.25 "Writing BASE data" Page 50)
KRC_ReadLoadData	(>>> 6.10.26 "Reading the load data" Page 51)
KRC_WriteLoadData	(>>> 6.10.27 "Writing load data" Page 51)

Administrative functions	
KRC_ReadSoftEnd	(>>> 6.10.28 "Reading the software limit switches of the robot axes" Page 53)
KRC_ReadSoftEndExt	(>>> 6.10.29 "Reading the software limit switches of the external axes" Page 53)
KRC_WriteSoftEnd	(>>> 6.10.30 "Writing the software limit switches of the robot axes" Page 54)
KRC_WriteSoftEndExt	(>>> 6.10.31 "Writing the software limit switches of the external axes" Page 55)
KRC_AutoStart	(>>> 6.10.41 "Automatically starting function blocks and signals" Page 65)

Motion programming	
KRC_MoveLinearAbsolute	(>>> 6.11.1 "Moving to a Cartesian position with a LIN motion" Page 66)
KRC_MoveLinearRelative	(>>> 6.11.2 "Moving to a Cartesian position with a LIN_REL motion" Page 67)
KRC_MoveDirectAbsolute	(>>> 6.11.3 "Moving to a Cartesian position with a PTP motion" Page 69)
KRC_MoveDirectRelative	(>>> 6.11.4 "Moving to a Cartesian position with a PTP_REL motion" Page 70)
KRC_MoveAxisAbsolute	(>>> 6.11.5 "Moving to an axis-specific position with a PTP motion" Page 72)
KRC_MoveCircAbsolute	(>>> 6.11.6 "Moving to a Cartesian position with a CIRC motion" Page 73)
KRC_MoveCircRelative	(>>> 6.11.7 "Moving to a Cartesian position with a CIRC_REL motion" Page 75)
KRC_JogLinearRelative	(>>> 6.11.8 "Jogging to a relative end position" Page 78)
KRC_JogToolRelative	(>>> 6.11.9 "Jogging to a relative end position in the TOOL coordinate system" Page 79)
KRC_Jog	(>>> 6.11.10 "Jogging to an end position" Page 80)

Motion programming (PLC OPEN-compliant)	
MC_MoveLinearAbsolute	(>>> 6.12.1 "Moving to a Cartesian position with a LIN motion" Page 83)
MC_MoveLinearRelative	(>>> 6.12.2 "Moving to a Cartesian position with a LIN_REL motion" Page 84)
MC_MoveDirectAbsolute	(>>> 6.12.3 "Moving to a Cartesian position with a PTP motion" Page 86)
MC_MoveDirectRelative	(>>> 6.12.4 "Moving to a Cartesian position with a PTP_REL motion" Page 87)
MC_MoveAxisAbsolute	(>>> 6.12.5 "Moving to an axis-specific position with a PTP motion" Page 89)
MC_MoveCircularAbsolute	(>>> 6.12.6 "Moving to a Cartesian position with a CIRC motion" Page 90)
MC_MoveCircularRelative	(>>> 6.12.7 "Moving to a Cartesian position with a CIRC_REL motion" Page 93)

Program execution control	
KRC_Abort	(>>> 6.10.38 "Canceling a program" Page 63)
KRC_Interrupt	(>>> 6.10.39 "Stopping the robot" Page 64)

Program execution control	
KRC_Continue	(>>> 6.10.40 "Continuing a program" Page 64)
KRC_WaitForInput	(>>> 6.10.20 "Wait statement (read digital input)" Page 46)

Interrupt programming	
KRC_DeclareInterrupt	(>>> 6.10.32 "Declaring interrupts" Page 56)
KRC_ActivateInterrupt	(>>> 6.10.33 "Activating interrupts" Page 57)
KRC_DeactivateInterrupt	(>>> 6.10.34 "Deactivating interrupts" Page 58)
KRC_ReadInterruptState	(>>> 6.10.35 "Reading the state of an interrupt" Page 59)

Path-related switching actions (=Trigger)	
KRC_SetDistanceTrigger	(>>> 6.10.36 "Activating a path-related switching action (TRIGGER WHEN DISTANCE)" Page 60)
KRC_SetPathTrigger	(>>> 6.10.37 "Activating a path-related switching action (TRIGGER WHEN PATH)" Page 61)

Diagnostic functions	
KRC_ReadMXAStatus	(>>> 6.13.1 "Reading the current state of the mxA interface" Page 95)
KRC_ReadMXAError	(>>> 6.13.2 "Reading error messages of the mxA interface" Page 96)
KRC_ReadKRCErrors	(>>> 6.13.4 "Reading error messages of the robot controller" Page 96)
KRC_MessageReset	(>>> 6.13.3 "Resetting error messages of the mxA interface" Page 96)
KRC_Diag	(>>> 6.13.5 "Reading diagnostic signals" Page 97)
KRC_Error	(>>> 6.13.6 "Reading and acknowledging error states" Page 99)

Special functions (general)	
KRC_ReadSysVar	(>>> 6.14.1 "Reading system variables" Page 101)
KRC_WriteSysVar	(>>> 6.14.2 "Writing system variables" Page 101)
KRC_BrakeTest	(>>> 6.14.3 "Calling a brake test" Page 103)
KRC_MasRef	(>>> 6.14.4 "Calling a mastering test" Page 104)
KRC_ReadSafeOPStatus	(>>> 6.14.5 "Reading the safety controller signals" Page 106)
KRC_ReadTouchUPState	(>>> 6.14.6 "Reading the state of the TouchUp status keys" Page 107)
KRC_TouchUP	(>>> 6.14.7 "Teaching points" Page 107)
KRC_SetAdvance	(>>> 6.14.8 "Modifying settings for the advance run" Page 108)
KRC_GetAdvance	(>>> 6.14.9 "Reading values from KRC_SetAdvance" Page 109)
KRC_Forward	(>>> 6.14.10 "Calculating the Cartesian robot position from the axis angles" Page 110)
KRC_Inverse	(>>> 6.14.11 "Calculating axis angles from the Cartesian robot position" Page 111)

Special functions (conveyor)	
KRC_ConvIniOff	(>>> 6.14.12 "Initializing a conveyor" Page 112)
KRC_ConvOn	(>>> 6.14.13 "Activating a conveyor" Page 113)

Special functions (conveyor)	
KRC_ConvFollow	(>>> 6.14.14 "Tracking a workpiece on the conveyor" Page 114)
KRC_ConvSkip	(>>> 6.14.15 "Picking up a workpiece from the conveyor" Page 115)
KRC_ActivateConvInterrupt	(>>> 6.14.16 "Activating interrupts for monitoring" Page 117)
KRC_DeactivateConvInterrupt	(>>> 6.14.17 "Deactivating interrupts for monitoring" Page 118)



In order to be able to use these function blocks, the KUKA.Conveyor-Tech technology package must be installed on the robot controller.

Special functions (VectorMove)	
KRC_VectorMoveOn	(>>> 6.14.18 "Activating a motion along a vector" Page 119)
KRC_VectorMoveOff	(>>> 6.14.19 "Deactivating KRC_VectorMoveOn" Page 120)



In order to be able to use these function blocks, the KUKA.Vector-Move technology package must be installed on the robot controller.

Special functions (workspaces)	
KRC_WriteWorkspace	(>>> 6.14.20 "Configuring Cartesian workspaces" Page 121)
KRC_ReadWorkspace	(>>> 6.14.21 "Reading the configuration of Cartesian workspaces" Page 122)
KRC_WriteAxWorkspace	(>>> 6.14.22 "Configuring axis-specific workspaces" Page 123)
KRC_ReadAxWorkspace	(>>> 6.14.23 "Reading the configuration of axis-specific workspaces" Page 124)
KRC_ReadWorkstates	(>>> 6.14.24 "Reading the status of the workspaces" Page 124)

6.3 Frequently used input/output signals in the function blocks

6.3.1 Input signals

AxisGroupIdx This signal input is used to set the number of the axis group addressed by a function block.

5 axis groups (robot and external axes) can be controlled by the PLC.

ExecuteCmd If this signal is set, mxAutomation transfers the associated function block to the robot. The function block is stored in a statement buffer by the robot, provided there is still sufficient space in the buffer. If the ExecuteCmd input is reset, mxAutomation deletes the function block from the buffer again unless execution of the statement has already begun.

6.3.2 Output signals

Busy This signal output indicates that the associated function block is currently being transferred to the robot's statement buffer or has already been transferred. It is reset when the ExecuteCmd input is reset.

Active This signal output indicates that the associated function block is currently being executed on the robot. It is reset when the ExecuteCmd input is reset.

Approximate positioning is not possible with the active output because the next motion instruction is not sent until the previous motion has been executed. Approximate positioning is only possible if the Busy output of the previous function block is connected to the ExecuteCmd input of the following block.

- Done** This signal output indicates that the associated function block has been successfully executed by the robot. It is reset when the ExecuteCmd input is reset.
- Error** This signal output indicates that an error has occurred during execution of the associated function block on the robot. In this case, the signal output ErrorID contains an error number. It is reset when the ExecuteCmd input is reset.
- ErrorID** This signal output contains an error number.
The errors and error causes corresponding to the error number are described here: (>>> 7 "Messages" Page 127)
- Aborted** This signal output is set either when the function block KRC_Abort is executed or when a statement is executed in the ABORTING mode. It is reset when the ExecuteCmd input is reset.

6.3.3 Signal sequence for execution of ExecuteCmd

Example The signal diagram applies in the following case:

- A statement has been transferred by means of ExecuteCmd and successfully executed.

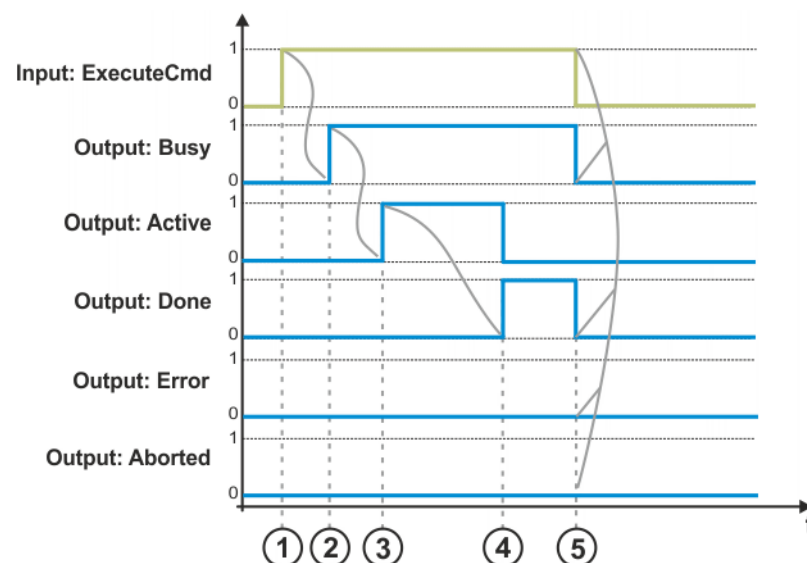


Fig. 6-1: Signal diagram – ExecuteCmd successful

Item	Description
1	The function block is transferred to the robot (= request to execute the statement).
2	The statement is transferred.
3	The statement is currently being executed.
4	The statement was completed successfully. Neither has an error occurred, nor has the statement been aborted, e.g. by KRC_Abort. The Error signal would be set instead of the Done signal in the case of an error, and the Aborted signal would be set instead of the Done signal if the statement is aborted.
5	If the ExecuteCmd input is reset, the outputs too are reset.

Variations

- ExecuteCmd is reset before Done is set. The statement will be executed in this case but the Done signal will not be set. This means there is no confirmation that the statement has been executed.
- ExecuteCmd is reset before Error or Aborted is set. The statement will be aborted in this case but neither the Error nor the Aborted signal will be set. This means there is no confirmation that the statement has been aborted.
- ExecuteCmd is reset before Active is set. In this case the function block will be deleted from the robot's statement buffer.
- ExecuteCmd is reset before Busy is set. The function block will not be transferred to the robot in this case and the statement will therefore not be executed.

6.4 Frequently used input/output signals in the MC function blocks

The MC function blocks differ from the KRC function blocks in that they correspond to the PLC OPEN standard or are closer to it. The behavior of the Busy signal output in particular is different for the MC function blocks. Here, the ComDone signal output must be used for linking function blocks.

6.4.1 Input signals

AxisGroupIdx	This signal input is used to set the number of the robot addressed by a function block.
Execute	If this signal is set, mxAutomation transfers the associated function block to the robot. The function block is stored in a statement buffer by the robot, provided there is still sufficient space in the buffer. If the Execute input is reset, mxAutomation deletes the function block from the buffer again unless execution of the statement has already begun.

6.4.2 Output signals

ComBusy	This signal output indicates that the associated function block has been sent from the PLC to the robot's statement buffer and has been correctly transferred.
ComDone	This signal output indicates that the associated function block has been sent from the PLC to the robot's statement buffer and has been correctly transferred. This signal output is identical to the Done signal output of the KRC function blocks. We recommend using this signal output for the approximation of motions.
Busy	This signal output indicates that execution of the associated function block has begun. However, it is possible that the function block has not yet been transferred to the robot's statement buffer. In this respect, this signal output differs from the Busy signal output of the KRC function blocks.
Active	This signal output indicates that the associated function block is currently being executed on the robot. It is reset when the Execute input is reset. Approximation is not possible with the Active output because the next motion instruction is not sent until the previous one is executed. Approximation is only possible if the ComDone output of the previous function block is connected to the Execute input of the subsequent block.
Error	This signal output indicates that an error has occurred during execution of the associated function block on the robot. In this case, the signal output ErrorID contains an error number. It is reset when the Execute input is reset.

ErrorID	This signal output contains an error number. The errors and error causes corresponding to the error number are described here: (>>> 7 "Messages" Page 127)
Command-Aborted	This signal output indicates that execution of a statement or motion has been aborted.

6.4.3 Signal sequence for execution of Execute

Example The signal diagram applies in the following case:

- A statement has been transferred by means of Execute and successfully executed.

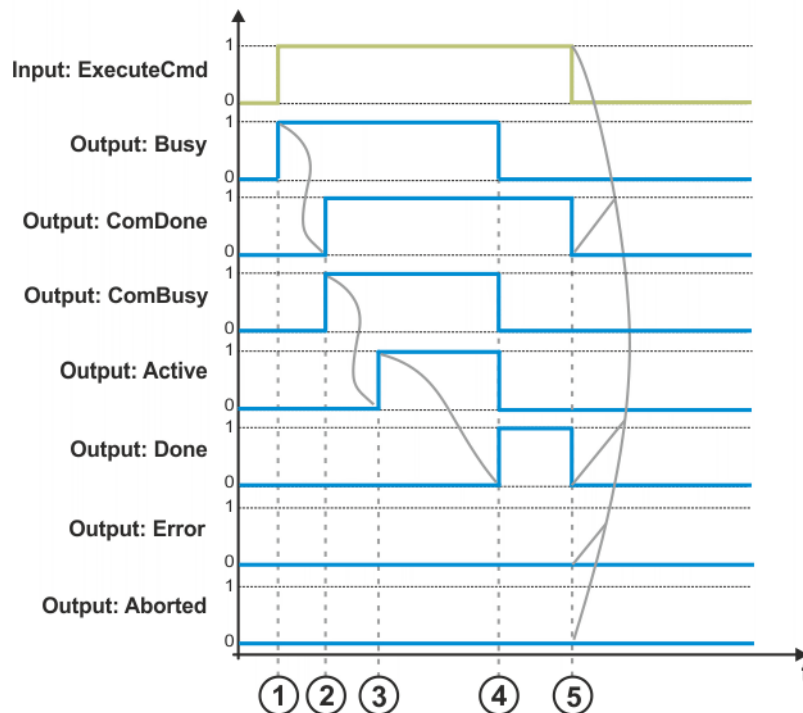


Fig. 6-2: Signal diagram – Execute successful

Item	Description
1	The function block is transferred to the robot (= request to execute the statement).
2	The statement has been transferred (= is located in the robot's statement buffer). The ComDone and ComBusy outputs are set.
3	The statement is currently being executed.
4	The statement was completed successfully. Neither has an error occurred, nor has the statement been aborted, e.g. by KRC_Abort. The Error signal would be set instead of the Done signal in the case of an error, and the Aborted signal would be set instead of the Done signal if the statement is aborted.
5	If the Execute input is reset, the outputs too are reset.

6.5 Structures for motion programming (STRUCT)

The data structures used for motion programming are described below.

APO

Approximation parameters for a KRC_Move motion command

Element	Type	Description
PTP_MODE	PTP_APO (INT)	Specifies whether and how the end point of a PTP motion is approximated. (>>> "PTP_APO (INT)" Page 24)
CP_MODE	CP_APO (INT)	Specifies whether and how the end point of a CP motion (LIN, CIRC) is approximated. (>>> "CP_APO (INT)" Page 24)
CPTP	INT	Approximation distance for PTP motions (= furthest distance before the end point at which approximate positioning can begin) ■ 1 ... 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
CDIS	REAL	Distance parameter (unit: mm) Approximation starts, at the earliest, when the distance to the end point falls below the value specified here.
CORI	REAL	Orientation parameter (unit: °) Approximation starts, at the earliest, when the dominant orientation angle (rotation or swiveling of the longitudinal axis of the tool) falls below the angle distance to the end point specified here.
CVEL	INT	Velocity parameter ■ 1 ... 100 % The approximation parameter specifies the percentage of the programmed velocity at which the approximate positioning process is started, at the earliest, in the deceleration phase towards the end point.

PTP_APO (INT)

Approximate positioning with PTP motions

Value	Name	Description
0	–	Without approximate positioning (default)
1	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, another approximate positioning parameter must also be specified.
2	C_PTP, C_DIS	PTP-CP approximation with distance parameter
3	C_PTP, C_ORI	PTP-CP approximation with orientation parameter
4	C_PTP, C_VEL	PTP-CP approximation with velocity parameter

CP_APO (INT)

Approximate positioning with CP motions (LIN, CIRC)

Value	Name	Description
0	–	Without approximate positioning (default)
1	C_DIS	Approximate positioning with distance parameter

Value	Name	Description
2	C_ORI	Approximate positioning with orientation parameter
3	C_VEL	Approximate positioning with velocity parameter

COORDSYS Coordinate system to which the Cartesian coordinates of the end position refer in a KRC_Move or KRC_Jog motion command

Element	Type	Description
Tool	INT	Number of the TOOL coordinate system <ul style="list-style-type: none"> ■ -1: coordinate system is not changed ■ 0: NULLFRAME ■ 1 ... 16: TOOL_DATA[1 ... 16] Default: -1
Base	INT	Number of the BASE coordinate system <ul style="list-style-type: none"> ■ -1: coordinate system is not changed ■ 0: NULLFRAME ■ 1 ... 32: BASE_DATA[1 ... 32] Default: -1
IPO_MODE	INT	Interpolation mode <ul style="list-style-type: none"> ■ 0: The tool is a fixed tool (#BASE). ■ 1: The tool is mounted on the mounting flange (#TCP). Default: 0

E6AXIS Angular values or translation values of the axes in an axis group for a KRC_MoveAxis motion command

Element	Type	Description
A1	REAL	Position of robot axis A1 (unit: mm or °)
A2	REAL	Position of robot axis A2 (unit: mm or °)
A3	REAL	Position of robot axis A3 (unit: mm or °)
A4	REAL	Position of robot axis A4 (unit: mm or °)
A5	REAL	Position of robot axis A5 (unit: mm or °)
A6	REAL	Position of robot axis A6 (unit: mm or °)
E1	REAL	Position of external axis E1 (optional), (unit: mm or °)
E2	REAL	Position of external axis E2 (optional), (unit: mm or °)
E3	REAL	Position of external axis E3 (optional), (unit: mm or °)
E4	REAL	Position of external axis E4 (optional), (unit: mm or °)
E5	REAL	Position of external axis E5 (optional), (unit: mm or °)
E6	REAL	Position of external axis E6 (optional), (unit: mm or °)

E6POS Cartesian coordinates of the end position for a KRC_Move or KRC_Jog motion command

Element	Type	Description
X	REAL	Offset in X direction (unit: mm)
Y	REAL	Offset in Y direction (unit: mm)
Z	REAL	Offset in Z direction (unit: mm)
A	REAL	Rotation about Z axis <ul style="list-style-type: none"> ■ -180° ... +180°

Element	Type	Description	
B	REAL	Rotation about Y axis ■ -180° ... +180°	
C	REAL	Rotation about X axis ■ -180° ... +180°	
S	INT	Status	The position (X, Y, Z) and orientation (A, B, C) values of the TCP are not sufficient to define the robot position unambiguously, as different axis positions are possible for the same TCP. Status and Turn serve to define an unambiguous position that can be achieved with different axis positions. Note: Further information about Status and Turn is contained in the "Operating and Programming Instructions for System Integrators".
T	INT	Turn	
E1	REAL	Position of external axis E1 (optional), (unit: mm or °)	
E2	REAL	Position of external axis E2 (optional), (unit: mm or °)	
E3	REAL	Position of external axis E3 (optional), (unit: mm or °)	
E4	REAL	Position of external axis E4 (optional), (unit: mm or °)	
E5	REAL	Position of external axis E5 (optional), (unit: mm or °)	
E6	REAL	Position of external axis E6 (optional), (unit: mm or °)	

FRAME

Space coordinates and orientation for the TOOL or BASE coordinate system

Element	Type	Description	
X	REAL	Offset in X direction (unit: mm)	
Y	REAL	Offset in Y direction (unit: mm)	
Z	REAL	Offset in Z direction (unit: mm)	
A	REAL	Orientation of the Z axis ■ -180° ... +180°	
B	REAL	Orientation of the Y axis ■ -180° ... +180°	
C	REAL	Orientation of the X axis ■ -180° ... +180°	

6.6 Integer variables

Some of the integer variables used in the function blocks are described below.

BufferMode

Mode in which a statement is executed on the robot controller

Value	Name	Description
0	DIRECT	The statement is executed directly by the Submit interpreter (Submit program). Note: This mode is not available for certain function blocks.

Value	Name	Description
1	ABORTING	The statement is executed immediately by the robot interpreter (main program). First, all active motions and buffered statements are aborted and the robot is braked to a standstill.
2	BUFFERED	The statement is buffered. Buffered statements are executed by the robot interpreter (main program) according to the FIFO principle.

QueueMode Mode in which a statement is executed on the robot controller

Value	Name	Description
0	DIRECT	The statement is executed directly by the submit interpreter (Submit program). Note: This mode is not available for certain function blocks.
1	ABORTING	The statement is executed immediately by the robot interpreter (main program). First, all active motions and buffered statements are aborted and the robot is braked to a standstill.
2	BUFFERED	The statement is buffered. Buffered statements are executed by the robot interpreter (main program) according to the FIFO principle.

CircType Orientation control for circular motion

Value	Name	Description
0	BASE	Base-related orientation control during a circular motion
1	PATH	Path-related orientation control during a circular motion

OriType Orientation control for the TCP

Value	Name	Description
0	VAR	The orientation of the TCP changes continuously during the motion.
1	CONSTANT	The orientation of the TCP remains constant during the motion.
2	JOINT	The orientation of the TCP changes continuously during the motion, but not uniformly. This is done by linear transformation (axis-specific motion) of the wrist axis angles. Note: This orientation type is not suitable if a specific orientation must be maintained exactly.

Status Current state of the mxA interface (function block KRC_ReadMXAStatus)

Value	Name	Description
0	Invalid	No function blocks can be processed. Frequent causes: <ul style="list-style-type: none"> ■ Submit interpreter stopped or deselected ■ I/O error due to incorrect bus configuration ■ Robot controller not started.
1	Error	An mxA error message is active. The error message must be reset with the function block KRC_MessageReset.
2	ProgramStopped	Robot interpreter is not active (main program has been stopped or deselected).
3	StandBy	Robot interpreter is active and waiting for statements, e.g. waiting for an input.
4	Executing	Robot interpreter is active (main program is being executed).
5	Aborting	Robot stopped and all statements aborted.

6.7 Data of a Cartesian workspace

The data of a Cartesian workspace which are used in certain function blocks are described below.

Origin and orientation

The origin and orientation of a Cartesian workspace are specified with the following elements. These are relative to the WORLD coordinate system.

Element	Data type	Unit	Minimum	Maximum
X	REAL	mm	-	-
Y	REAL	mm	-	-
Z	REAL	mm	-	-
A	REAL	°	-180	180
B	REAL	°	-180	180
C	REAL	°	-180	180

Dimensions

The dimensions of a Cartesian workspace are specified with the following elements.

Element	Data type	Unit
X1	REAL	mm
X2	REAL	mm
Y1	REAL	mm
Y2	REAL	mm
Z1	REAL	mm
Z2	REAL	mm

6.8 Data of an axis-specific workspace

The data of an axis-specific workspace which are used in certain function blocks are described below.


Robot axes

Element	Data type	Unit	Description
A1_N	REAL	mm/°	Lower limit for axis angle
A2_N	REAL	mm/°	
A3_N	REAL	mm/°	
A4_N	REAL	mm/°	
A5_N	REAL	mm/°	
A6_N	REAL	mm/°	
A1_P	REAL	mm/°	Upper limit for axis angle
A2_P	REAL	mm/°	
A3_P	REAL	mm/°	
A4_P	REAL	mm/°	
A5_P	REAL	mm/°	
A6_P	REAL	mm/°	

External axes

Element	Data type	Unit	Description
E1_N	REAL	mm/°	Lower limit for axis angle
E2_N	REAL	mm/°	
E3_N	REAL	mm/°	
E4_N	REAL	mm/°	
E5_N	REAL	mm/°	
E6_N	REAL	mm/°	
E1_P	REAL	mm/°	Upper limit for axis angle
E2_P	REAL	mm/°	
E3_P	REAL	mm/°	
E4_P	REAL	mm/°	
E5_P	REAL	mm/°	
E6_P	REAL	mm/°	

6.9 Programming tips for KUKA.PLC mxAutomation

 The **MxA_CODESYS_Template_PN** template supplied on the USB stick (**Template** folder) contains all the function blocks and a programming example with the fundamental function blocks. It is advisable to use this template for creating an mxAutomation robot program.

Instancing

The following function blocks may only be instanced once per robot. In the case of multiple instancing, the signals of the most recently called function block are output.

- KRC_ReadAxisGroup
- KRC_Intialize
- KRC_SetOverride
- KRC_AutomaticExternal
- KRC_AutoStart
- KRC_Diag
- KRC_WriteAxisGroup

All other function blocks used in the mxAutomation robot program can be created as a multi-instance call. The advantage of this is that not every function block requires a data block of its own.

- ExecuteCmd**
- As far as possible, an ExecuteCmd input should only ever be simultaneously set for a function block of the same robot.
 - After an ExecuteCmd input has been activated, do not reset it again until the function block has confirmed execution of the statement by means of the Done signal or indicated by means of the Error or Aborted signal that the statement has not been executed. If the ExecuteCmd input is reset beforehand, there is no confirmation that the statement has been executed.
 - By linking the Busy output of a function block to the ExecuteCmd input of the following block, it is possible to transfer a sequence of consecutive functions to the statement buffer and to execute them.
 - ExecuteCmd should not be used as a start signal for a sequence of statements. This might result in the transfer of the following statements taking up valuable time.
- Execution of a statement sequence should be triggered with KRC_WaitForInput instead. Linking the Busy and ExecuteCmd signals ensures that execution of the planned steps can be started without delay as soon as the corresponding input signal is received.
- Program override**
- If the mxAutomation robot program is started by a RESET at the function block KRC_AutomaticExternal, set the program override to a value greater than zero. Only then is the robot interpreter executed in a loop.
 - While the mxAutomation robot program is being executed, the program override can be set to zero without problems.
- 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.
 - Approximate positioning is only possible if the motion statement is followed by a statement that is transferred in BUFFERED mode.

6.9.1 Programming example (template MxA_CODESYS_Template_PN)

- Description** This example illustrates the basic structure of an mxAutomation robot program.
- Precondition** In order to be able to move the robot using function blocks, the following preconditions must be met:
- Submit interpreter is running.
 - Drives are activated.
 - Robot is in Automatic External mode.
 - mxAutomation robot program is selected and started.
- Procedure**
- Create a simple application program.
- Application program**
- The robot is to be moved to a position by means of a PTP motion. The following function blocks are required for this:
- Function block KRC_Initialize to initialize the mxA interface on the robot controller
- The versions of the PLC library and mxA interface must match. The mxA interface is initialized when the Done signal on the module is TRUE.
- Function block KRC_Error to read and reset mxA interface errors
- If an error is active, the motion enable of the robot is deactivated.

i The function block `KRC_Error` contains the function blocks `KRC_ReadmxAError` and `KRC_MessageReset`, thereby simplifying the programming. Furthermore, the current state of the mxA interface can be read with `KRC_Error`.

- Function blocks `KRC_AutomaticExternal` and `KRC_AutoStart` to activate the Automatic External interface (`KRC_AutoStart` automatically controls `KRC_AutomaticExternal`)

Important signals that must be mapped (`KRC_AutomaticExternal`):

- `ENABLE_EXT` (Robot may be moved in Automatic External mode.)
- `ENABLE_T1` (Robot may be moved in T1 mode.)
- `MOVE_ENABLE` (Grants the robot motion enable.)

Important signals that must be mapped (`KRC_AutoStart`):

- `AxisGroupIdx`
- `ExecuteReset`

The following signals are set automatically by `KRC_AutoStart`.

- `DRIVES_ON` (The drives of the robot are activated by means of a pulse.)
- `CONF_MESS` (Resets the error messages of the robot controller.)
- `RESET` (Selects the mxAutomation robot program and starts it.)

- Function block `KRC_SetOverride` to set the program override of the robot

i If the mxAutomation robot program is started by a `RESET` at the function block `KRC_AutomaticExternal`, the override must be greater than zero.

- Function block `KRC_MoveDirectAbsolute` to move to the desired position by means of a PTP motion

The motion has been executed successfully when the Done signal on the module is `TRUE`.

6.10 Administrative functions

6.10.1 Reading PLC-specific communication into a non-PLC-specific structure

Description The function block `KRC_ReadAxisGroup` translates the CODESYS-specific interface to the robot controller into the non-PLC-specific structure of an axis group.

i The function block `KRC_ReadAxisGroup` must always be called at the start of the program. Access to the axis group structure is only permissible between `KRC_ReadAxisGroup` and `KRC_WriteAxisGroup`.

i The function block may only be instanced once per axis group. In the case of multiple instancing, the signals of the most recently called function block are output.

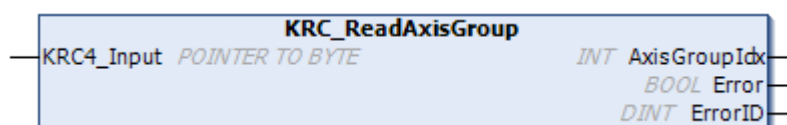


Fig. 6-3: Function block `KRC_ReadAxisGroup`

Inputs

Parameter	Type	Description
KRC4_Input	POINTER TO BYTE	Structure map of the input range of the robot controller
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.10.2 Writing a non-PLC-specific structure into PLC-specific communication

Description

The function block KRC_WriteAxisGroup translates the non-PLC-specific structure of an axis group into the CODESYS-specific interface to the robot controller.

i The function block KRC_WriteAxisGroup must always be called at the end of the program. Access to the axis group structure is only permissible between KRC_ReadAxisGroup and KRC_WriteAxisGroup.

i The function block may only be instantiated once per axis group. In the case of multiple instantiation, the signals of the most recently called function block are output.

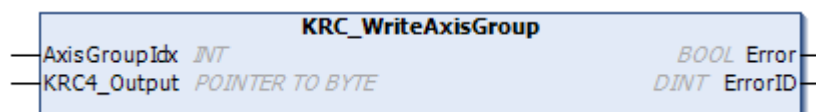


Fig. 6-4: Function block KRC_WriteAxisGroup

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
KRC4_Output	POINTER TO BYTE	Structure map of the output range of the robot controller

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.10.3 Initializing the mxA interface

Description

The function block KRC_Initialize initializes the mxA interface on the robot controller. Statements cannot be transferred until the interface has been initialized.

During initialization, the versions of the PLC library and mxA interface are compared and checked for compatibility. The 1st and 2nd digits of the version must match.

i The function block may only be instantiated once per axis group. In the case of multiple instancing, the signals of the most recently called function block are output.

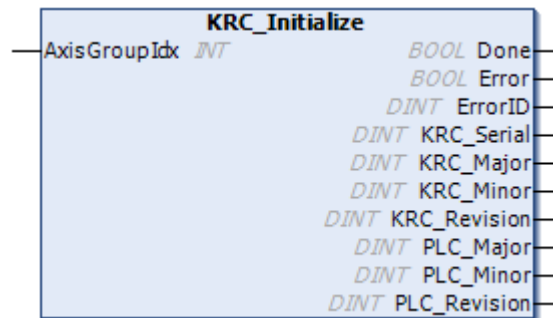


Fig. 6-5: Function block KRC_Initialize

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
KRC_Serial	DINT	Serial number of the robot controller
KRC_Major	DINT	Version identifier of the mxA interface (1st digit)
KRC_Minor	DINT	Version identifier of the mxA interface (2nd digit)
KRC_Revision	DINT	Version identifier of the mxA interface (3rd digit)
PLC_Major	DINT	Version identifier of the PLC library (1st digit)
PLC_Minor	DINT	Version identifier of the PLC library (2nd digit)
PLC_Revision	DINT	Version identifier of the PLC library (3rd digit)
Done	BOOL	TRUE = initialization successfully completed
Error	BOOL	TRUE = error in function block

6.10.4 Setting the program override (POV)

Description

The function block KRC_SetOverride sets the program override.

Program override is the velocity of the robot during program execution. The program override is specified as a percentage of the programmed velocity. The override setting is transferred to the robot during every PLC cycle. If the override setting is changed, this change is detected by the robot and applied.

The override is only applied in Automatic External mode so that the override can be set via the smartPAD in the test modes T1 and T2, e.g. for teaching.

i The function block may only be instantiated once per axis group. In the case of multiple instancing, the signals of the most recently called function block are output.



Fig. 6-6: Function block KRC_SetOverride

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
Override	INT	Set the program override. <ul style="list-style-type: none"> 0 ... 100 %

Outputs

Parameter	Type	Description
ActualOverride	INT	Current override setting <ul style="list-style-type: none"> 0 ... 100%
ErrorID	DINT	Error number
Valid	BOOL	TRUE = data are valid
Error	BOOL	TRUE = error in function block

6.10.5 Activating and reading Automatic External signals from the robot controller

Description

The function block KRC_AutomaticExternal activates the Automatic External interface and reads the interface signals.

i The function block may only be instanced once per axis group. In the case of multiple instancing, the signals of the most recently called function block are output.

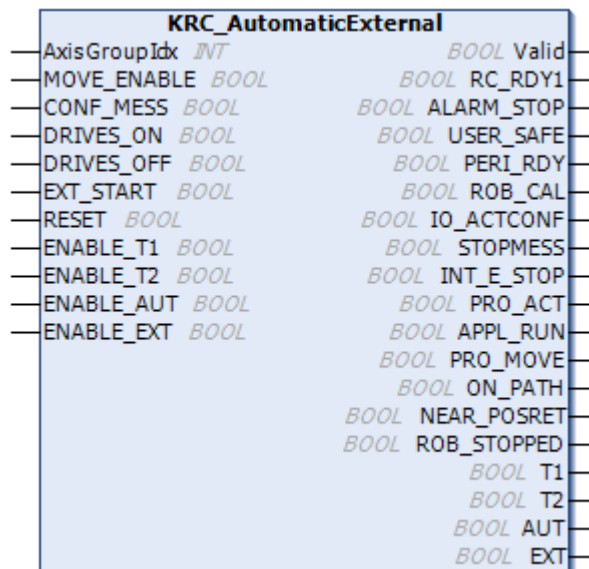


Fig. 6-7: Function block KRC_AutomaticExternal

Inputs

Parameter	Type	Signal name (KRL)	Description
AxisGroupIdx	INT	—	Index of axis group ■ 1 ... 5
MOVE_ENABLE	BOOL	\$MOVE_ENABLE	TRUE = motion enable for the robot Note: This system variable is monitored by the robot controller in all operating modes.
CONF_MESS	BOOL	\$CONF_MESS	TRUE = acknowledgement of error messages
DRIVES_ON	BOOL	\$DRIVES_ON	TRUE = activation of the robot drives
DRIVES_OFF	BOOL	\$DRIVES_OFF	TRUE = deactivation of the robot drives
EXT_START	BOOL	\$EXT_START	TRUE = start or continuation of robot program execution
RESET	BOOL	—	Selects the mxAutomation robot program in the case of a rising edge of the signal and starts it. First, all buffered statements are aborted.
ENABLE_T1	BOOL	—	TRUE = enabling of T1 mode The signal \$MOVE_ENABLE is suppressed in the absence of enabling. The robot cannot be moved.
ENABLE_T2	BOOL	—	TRUE = enabling of T2 mode The signal \$MOVE_ENABLE is suppressed in the absence of enabling. The robot cannot be moved.
ENABLE_AUT	BOOL	—	TRUE = enabling of Automatic mode The signal \$MOVE_ENABLE is suppressed in the absence of enabling. The robot cannot be moved.
ENABLE_EXT	BOOL	—	TRUE = enabling of Automatic External mode The signal \$MOVE_ENABLE is suppressed in the absence of enabling. The robot cannot be moved.

Outputs

Parameter	Type	Signal name (KRL)	Description
Valid	BOOL	—	TRUE = data are valid
RC_RDY1	BOOL	\$RC_RDY1	TRUE = robot controller ready for program start
ALARM_STOP	BOOL	\$ALARM_STOP	FALSE = robot stop by EMERGENCY STOP
USER_SAFE	BOOL	\$USER_SAF	FALSE = operator safety violated
PERI_RDY	BOOL	\$PERI_RDY	TRUE = robot drives activated
ROB_CAL	BOOL	\$ROB_CAL	TRUE = robot axes mastered
IO_ACTCONF	BOOL	\$IO_ACTCONF	TRUE = Automatic External interface active

Parameter	Type	Signal name (KRL)	Description
STOPMESS	BOOL	\$STOPMESS	TRUE = safety circuit interrupted (robot fault)
INT_E_STOP	BOOL	Int. E-Stop	TRUE = external EMERGENCY STOP FALSE = EMERGENCY STOP device pressed on the smartPAD
PRO_ACT	BOOL	\$PRO_ACT	TRUE = process active at robot level
APPL_RUN	BOOL	APPL_RUN	TRUE = robot program running
PRO_MOVE	BOOL	\$PRO_MOVE	TRUE = synchronous robot motion active
ON_PATH	BOOL	\$ON_PATH	TRUE = robot on programmed path
NEAR_POSRET	BOOL	\$NEAR_POSRET	TRUE = robot near most recently saved position on the programmed path (after leaving path)
ROB_STOPPED	BOOL	\$ROB_STOPPED	TRUE = robot is at standstill
T1	BOOL	\$T1	TRUE = operating mode T1 selected
T2	BOOL	\$T2	TRUE = operating mode T2 selected
AUT	BOOL	\$AUT	TRUE = operating mode Automatic selected
EXT	BOOL	\$EXT	TRUE = operating mode Automatic External selected

6.10.6 Reading the current robot position

Description

The function block KRC_ReadActualPosition reads the current Cartesian actual position of the robot \$POS_ACT. This is updated cyclically.

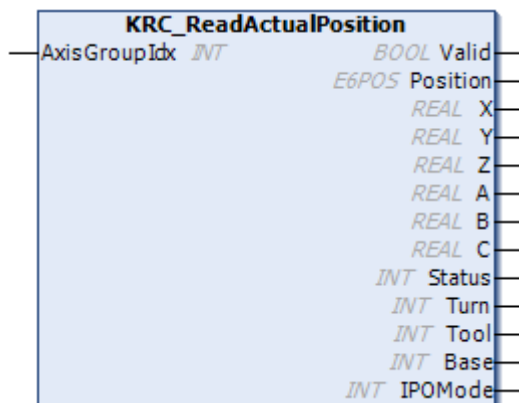


Fig. 6-8: Function block KRC_ReadActualPosition

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
Position	E6POS	Current Cartesian actual position \$POS_ACT The data structure E6POS contains all components of the Cartesian actual position (= position of the TCP relative to the origin of the BASE coordinate system).
X, Y, Z	REAL	Current actual position in the X, Y, Z directions
A, B, C	REAL	Orientation A, B, C in the current actual position
Status	INT	Status of the current actual position
Turn	INT	Turn of the current actual position
Tool	INT	Number of the currently used TOOL coordinate system \$ACT_TOOL
Base	INT	Number of the currently used BASE coordinate system \$ACT_BASE
IPOMode	INT	Current interpolation mode in the main run \$IPO_MODE_C

6.10.7 Reading the current axis position

Description

The function block KRC_ReadActualAxisPosition reads the current axis-specific robot position \$AXIS_ACT. This is updated cyclically.



Fig. 6-9: Function block KRC_ReadActualAxisPosition

Inputs


Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
AxisPosition	E6AXIS	Current axis-specific robot position \$AXIS_ACT The data structure E6AXIS contains all the axis positions of the axis group.
A1 ... A6	REAL	Current position of robot axes A1 to A6
E1 ... E6	REAL	Current position of external axes E1 to E6

6.10.8 Reading the current path velocity

Description The function block KRC_ReadActualVelocity reads the current actual velocity at the TCP of the robot \$VEL_ACT.

 The current path velocity can only be read for CP motions in program mode.

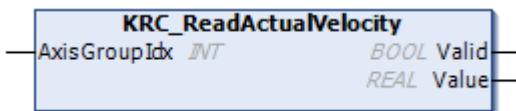


Fig. 6-10: Function block KRC_ReadActualVelocity

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
Value	REAL	Current path velocity \$VEL_ACT (unit: m/s)

6.10.9 Reading the current axis velocity

Description The function block KRC_ReadActualAxisVelocity reads the current axis-specific velocity of the robot \$VEL_AXIS_ACT.

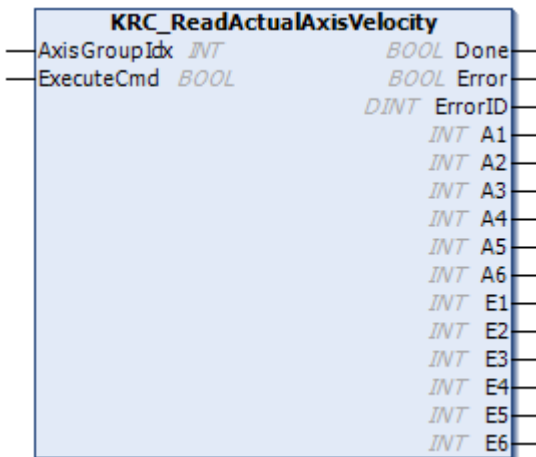


Fig. 6-11: Function block KRC_ReadActualAxisVelocity

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.


Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
A1 ... A6	INT	Current motor speed (-100% ... +100%) of A1 ... A6 in the main run, relative to the maximum motor speed (\$VEL_AXIS_MA) Note: The actual resulting speed of the robot axis is dependent on the gear ratio.
E1 ... E6	INT	Current motor speed (-100% ... +100%) of E1 ... E6 in the main run, relative to the maximum motor speed (\$VEL_AXIS_MA) Note: The actual resulting speed of the external axis is dependent on the gear ratio.
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.10 Reading the current robot acceleration

Description

The function block KRC_ReadActualAcceleration reads the current Cartesian acceleration at the TCP of the robot \$ACC_CAR_ACT.

 The current Cartesian acceleration about angles A, B, C is not evaluated.

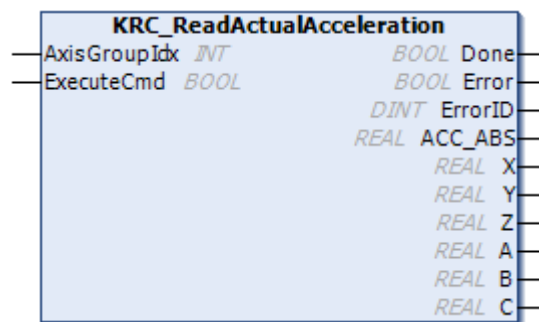


Fig. 6-12: Function block KRC_ReadActualAcceleration

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
ACC_ABS	REAL	Current Cartesian acceleration relative to the absolute value of the overall acceleration $\sqrt{X^2 + Y^2 + Z^2}$ (unit: m/s^2)
X, Y, Z	REAL	Current Cartesian acceleration in the X, Y, Z direction (unit: m/s^2)

Parameter	Type	Description
A, B, C	REAL	Current Cartesian acceleration about angles A, B, C 0 m/s ² (not calculated)
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.11 Reading a digital input

Description

The function block KRC_ReadDigitalInput polls and reads a digital input of the robot controller. This function is executed in the Submit interpreter.



Fig. 6-13: Function block KRC_ReadDigitalInput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the digital input ■ 1 ... 2 048

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Value	BOOL	Value of the digital input
Error	BOOL	TRUE = error in function block

6.10.12 Reading digital inputs 1 to 8

Description

The function block KRC_ReadDigitalInput1To8 polls and reads the digital inputs 1 to 8 of the robot controller. This function is executed in the Submit interpreter. The input values can be read continuously without triggering a blockade.



Fig. 6-14: Function block KRC_ReadDigitalInput1To8

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
IN1 ... IN8	BOOL	Actual value of the digital input \$IN[1] ... \$IN[8]
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.10.13 Reading multiple digital inputs

Description

The function block KRC_ReadDigitalInputArray polls and reads multiple digital inputs of the robot controller. This function is executed in the Submit interpreter.

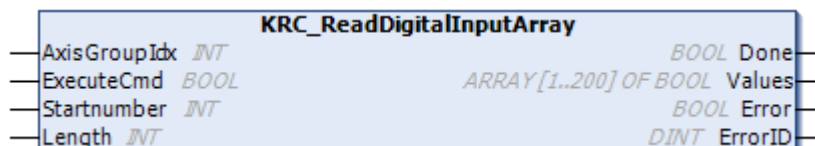


Fig. 6-15: Function block KRC_ReadDigitalInputArray

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Startnumber	INT	Number of the first digital output that is polled ■ 1 ... 2 048
Length	INT	Number of inputs that are polled ■ 1 ... 2 00 Note: If the number of inputs to be read exceeds the 1 ... 2048 range, no error message is generated. Inputs outside of this range are not read.

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Values	BOOL[200]	Values of the digital inputs
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.10.14 Reading a digital output

Description

The function block KRC_ReadDigitalOutput polls and reads a digital output of the robot controller. This function is executed in the Submit interpreter.



Fig. 6-16: Function block KRC_ReadDigitalOutput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the digital output ■ 1 ... 2 048

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Value	BOOL	Value of the digital output
Error	BOOL	TRUE = error in function block

6.10.15 Writing a digital output

Description

The function block KRC_WriteDigitalOutput writes a digital output or a pulse output on the robot controller. This function is executed in the Submit interpreter.

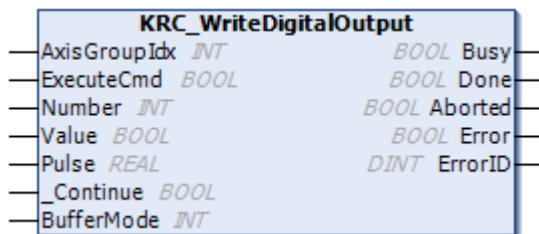


Fig. 6-17: Function block KRC_WriteDigitalOutput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the digital output <ul style="list-style-type: none"> 1 ... 2 048 Note: It must be ensured that no outputs are used that are already assigned by the system. Example: \$OUT[1025] is always TRUE.
Value	BOOL	Value of the digital output
Pulse	REAL	Length of the pulse <ul style="list-style-type: none"> 0.0 s No pulse active 0.1 ... 3.0 s Pulse interval = 0.1 s; pulse durations outside this range of values trigger a program stop.
_Continue	BOOL	TRUE = output written in advance run
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.16 Writing digital outputs 1 to 8

Description

The function block KRC_WriteDigitalOutput1To8 writes the digital outputs 1 to 8 on the robot controller. This function is executed in the Submit interpreter. The output values can be written continuously without triggering a blockade.

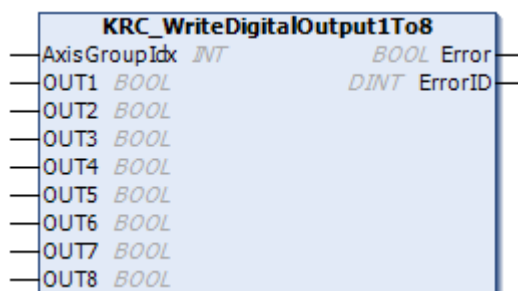


Fig. 6-18: Function block KRC_WriteDigitalOutput1To8

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
OUT1 ... OUT8	BOOL	Setpoint value of the output \$OUT[1] ... \$OUT[8]

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.10.17 Reading an analog input

Description

The function block KRC_ReadAnalogInput polls and reads an analog input of the robot controller. This function is executed in the Submit interpreter.

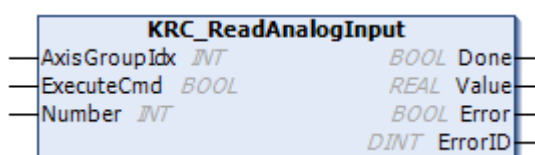


Fig. 6-19: Function block KRC_ReadAnalogInput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the analog input <ul style="list-style-type: none"> ■ 1 ... 32

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Value	REAL	Value of the analog input
Error	BOOL	TRUE = error in function block

6.10.18 Reading an analog output

Description

The function block KRC_ReadAnalogOutput polls and reads an analog output of the robot controller. This function is executed in the Submit interpreter.



Fig. 6-20: Function block KRC_ReadAnalogOutput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the analog output ■ 1 ... 32

Outputs

Parameter	Type	Description
Value	REAL	Value of the analog output
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.19 Writing an analog output

Description

The function block KRC_WriteAnalogOutput polls and writes an analog output of the robot controller. This function is executed in the Submit interpreter.

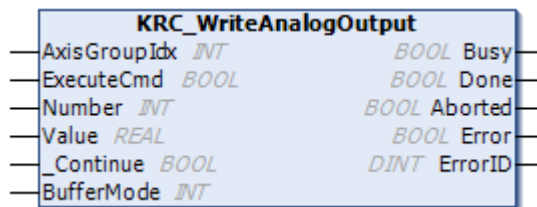


Fig. 6-21: Function block KRC_WriteAnalogOutput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the analog output ■ 1 ... 32
Value	REAL	Value of the analog output
_Continue	BOOL	TRUE = output written in advance run
BufferMode	INT	Mode in which the statement is executed ■ 0: DIRECT ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.20 Wait statement (read digital input)

Description

The function block KRC_WaitForInput stops the program until a digital input takes the defined value. Program execution is then resumed.



Fig. 6-22: Function block KRC_WaitForInput

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Number	INT	Number of the digital input <ul style="list-style-type: none"> 1 ... 2 048
Value	BOOL	Setpoint value of the digital input
iContinue	BOOL	TRUE = poll input in advance run
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = statement is currently being executed (robot is waiting for an input)
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement has been aborted
Error	BOOL	TRUE = error in function block

6.10.21 Selecting the tool, base and interpolation mode

Description

The function block KRC_SetCoordSys can be used to set the tool, base and interpolation mode without having to execute a motion at the same time. This

function is required, for example, to read the current position in different coordinate systems.



Fig. 6-23: Function block KRC_SetCoordSys

Inputs

Parameter	Type	Description
CoordinateSystem	COORDSYS	Coordinate system to which the specified values refer (>>> "COORDSYS" Page 25)
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.22 Reading TOOL data

Description

The function block KRC_ReadToolData reads the TOOL data of the robot.



Fig. 6-24: Function block KRC_ReadToolData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5

Parameter	Type	Description
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
ToolNo	INT	Number of the TOOL coordinate system <ul style="list-style-type: none"> 1 ... 16: TOOL_DATA[1 ... 16]

Outputs

Parameter	Type	Description
ToolData	FRAME	The data structure FRAME contains the following TOOL data: <ul style="list-style-type: none"> 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 (>>> "FRAME" Page 26)
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.23 Writing TOOL data

Description

The function block KRC_WriteToolData writes the TOOL data of the robot.



Fig. 6-25: Function block KRC_WriteToolData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
ToolData	FRAME	The data structure FRAME contains the following TOOL data: <ul style="list-style-type: none"> 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 (>>> "FRAME" Page 26)

Parameter	Type	Description
ToolNo	INT	Number of the TOOL coordinate system <ul style="list-style-type: none"> 1 ... 16: TOOL_DATA[1 ... 16]
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.24 Reading BASE data

Description

The function block KRC_ReadBaseData reads the BASE data of the robot.

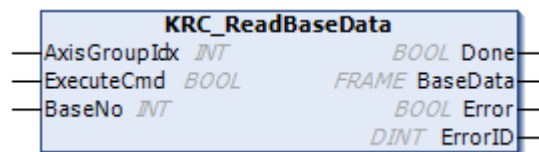


Fig. 6-26: Function block KRC_ReadBaseData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
BaseNo	INT	Number of the BASE coordinate system <ul style="list-style-type: none"> 1 ... 32: BASE_DATA[1 ... 32]

Outputs

Parameter	Type	Description
BaseData	FRAME	The data structure FRAME contains the following BASE data: <ul style="list-style-type: none"> X, Y, Z: Origin of the BASE coordinate system relative to the WORLD coordinate system A, B, C: Orientation of the BASE coordinate system relative to the WORLD coordinate system (>>> "FRAME" Page 26)
ErrorID	DINT	Error number

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.25 Writing BASE data

Description

The function block KRC_WriteBaseData writes the BASE data of the robot.



Fig. 6-27: Function block KRC_WriteBaseData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
BaseData	FRAME	The data structure FRAME contains the following BASE data: <ul style="list-style-type: none"> X, Y, Z: Origin of the BASE coordinate system relative to the WORLD coordinate system A, B, C: Orientation of the BASE coordinate system relative to the WORLD coordinate system (>>> "FRAME" Page 26)
BaseNo	INT	Number of the BASE coordinate system <ul style="list-style-type: none"> 1 ... 32: BASE_DATA[1 ... 32]
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.26 Reading the load data

Description The function block KRC_ReadLoadData reads the load data of the robot (payload data or supplementary load data). This function is executed in the Submit interpreter.

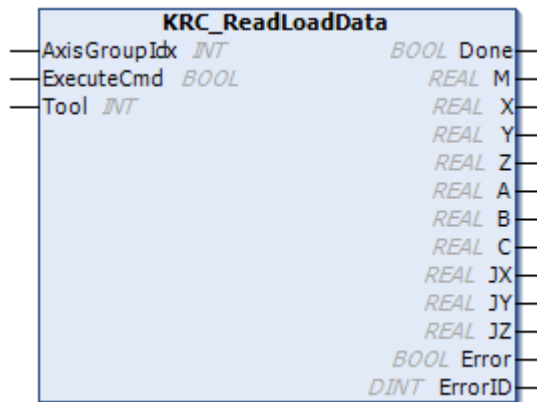


Fig. 6-28: Function block KRC_ReadLoadData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Tool	INT	Number of the TOOL coordinate system for reading the payload data or number for reading the supplementary load data <ul style="list-style-type: none"> ■ 1 ... 16: TOOL_DATA[1 ... 16] ■ -1: Supplementary load A1 ■ -2: Supplementary load A2 ■ -3: Supplementary load A3

Outputs

Parameter	Type	Description
M	REAL	Mass
X, Y, Z	REAL	Position of the center of gravity relative to the flange
A, B, C	REAL	Orientation of the principal inertia axes relative to the flange
JX, JY, JZ	REAL	Mass moments of inertia
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.27 Writing load data

Description The function block KRC_WriteLoadData writes the load data of the robot (payload data or supplementary load data).

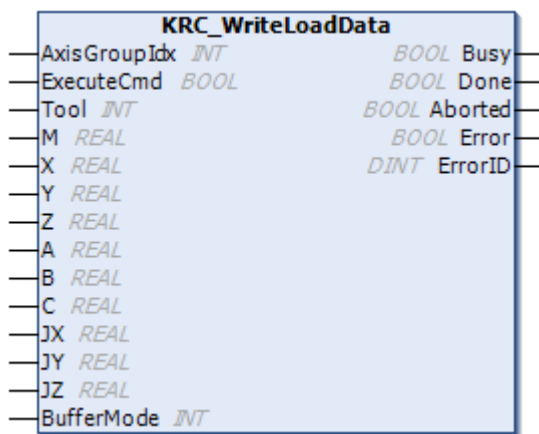


Fig. 6-29: Function block KRC_WriteLoadData

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Tool	INT	Number of the TOOL coordinate system for writing the payload data or number for writing the supplementary load data <ul style="list-style-type: none"> 1 ... 16: TOOL_DATA[1 ... 16] -1: Supplementary load A1 -2: Supplementary load A2 -3: Supplementary load A3
M	REAL	Mass
X, Y, Z	REAL	Position of the center of gravity relative to the flange
A, B, C	REAL	Orientation of the principal inertia axes relative to the flange
JX, JY, JZ	REAL	Mass moments of inertia
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.28 Reading the software limit switches of the robot axes

Description The function block KRC_ReadSoftEnd reads the software limit switches of the robot axes. This function is executed in the Submit interpreter.



Fig. 6-30: Function block KRC_ReadSoftEnd

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.

Outputs

Parameter	Type	Description
A1_Min ... A6_Min	REAL	Negative software limit switch of axis A1 ... A6
A1_Max ... A6_Max	REAL	Positive software limit switch of axis A1 ... A6
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.29 Reading the software limit switches of the external axes

Description The function block KRC_ReadSoftEndExt reads the software limit switches of the external axes. This function is executed in the Submit interpreter.

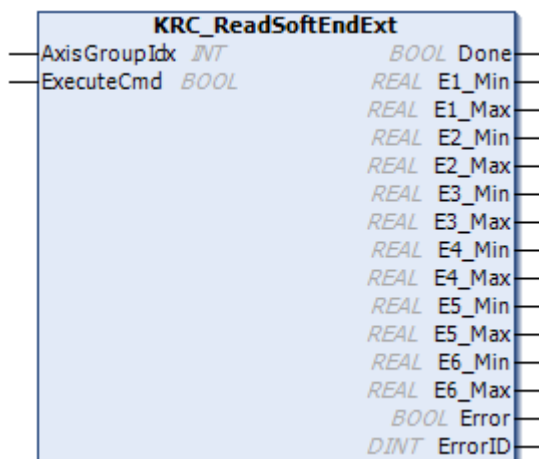


Fig. 6-31: Function block KRC_ReadSoftEndExt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.

Outputs

Parameter	Type	Description
E1_Min ... E6_Min	REAL	Negative software limit switch of axis E1 ... E6
E1_Max ... E6_Max	REAL	Positive software limit switch of axis E1 ... E6
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.30 Writing the software limit switches of the robot axes

Description

The function block KRC_WriteSoftEnd writes the software limit switches of the robot axes.

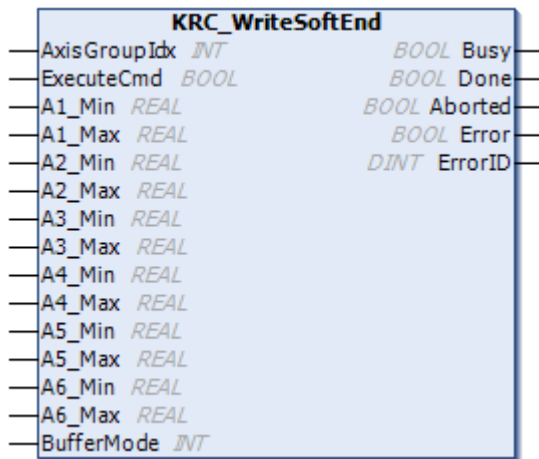


Fig. 6-32: Function block KRC_WriteSoftEnd

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
A1_Min ... A6_Min	REAL	Negative software limit switch of axis A1 ... A6 (Unit: mm or °)
A1_Max ... A6_Max	REAL	Positive software limit switch of axis A1 ... A6 (Unit: mm or °)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 0: DIRECT ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.31 Writing the software limit switches of the external axes

Description

The function block KRC_WriteSoftEndExt writes the software limit switches of the external axes.

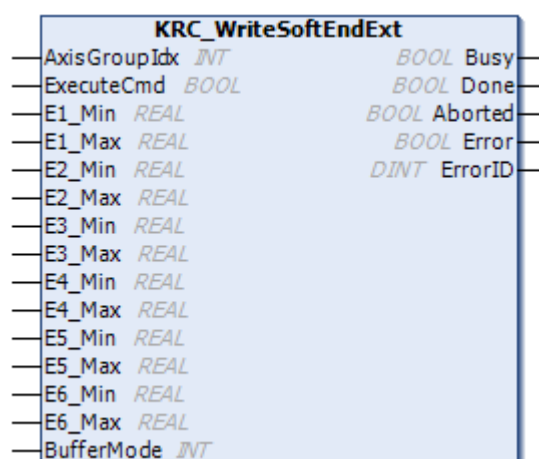


Fig. 6-33: Function block KRC_WriteSoftEndExt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
E1_Min ... E6_Min	REAL	Negative software limit switch of axis E1 ... E6 (Unit: mm or °)
E1_Max ... E6_Max	REAL	Positive software limit switch of axis E1 ... E6 (Unit: mm or °)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 0: DIRECT ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.32 Declaring interrupts

Description

The function block KRC_DeclareInterrupt declares an interrupt to a digital input. There are 8 predefined interrupts available for this.

Syntax

```
GLOBAL INTERRUPT DECL 90+Interrupt WHEN $IN[Input] == InputValue
DO Subprogram
```



Fig. 6-34: Function block KRC_DeclareInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.

Parameter	Type	Description
Interrupt	INT	Number of the interrupt <ul style="list-style-type: none"> ■ 1 ... 8
Input	INT	Number of the digital input to which the interrupt is declared <ul style="list-style-type: none"> ■ 1 ... 2 048 Note: It must be ensured that no inputs are used that are already assigned by the system. Example: \$IN[1025] is always TRUE.
InputValue	BOOL	TRUE = statement is executed in the case of a rising edge of the signal. FALSE = statement is executed in the case of a falling edge of the signal.
Reaction	INT	Reaction to the interrupt <ul style="list-style-type: none"> ■ 0: BRAKE F + HALT ■ 1: BRAKE + HALT ■ 2: BRAKE F + WAIT FOR \$IN[Input]<>InputValue ■ 3: BRAKE + WAIT FOR \$IN[Input]<>InputValue ■ 4: BRAKE F + WAIT FOR KRC_Continue ■ 5: BRAKE + WAIT FOR KRC_Continue ■ 6: BRAKE F + WAIT FOR \$IN[Input]<>InputValue AND KRC_Continue ■ 7: BRAKE + WAIT FOR \$IN[Input]<>InputValue AND KRC_Continue
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed in the advance run Note: The statement can no longer be aborted. Exception: Program is deselected or reset.
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.33 Activating interrupts

Description The function block KRC_ActivateInterrupt activates a previously declared interrupt. There are 8 predefined interrupts available for this.

An interrupt cannot be processed until the interrupt has been activated by the main run of the robot interpreter. The function block KRC_WaitForInput can be used to monitor and check whether an interrupt is active.



Fig. 6-35: Function block KRC_ActivateInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Interrupt	INT	Number of the interrupt <ul style="list-style-type: none"> ■ 1 ... 8
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed in the advance run Note: The signal does not indicate whether the interrupt has really been triggered or activated by the main run.
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.34 Deactivating interrupts

Description

The function block KRC_DeactivateInterrupt deactivates a previously declared interrupt. There are 8 predefined interrupts available for this.

Syntax

```
INTERRUPT OFF 90+Interrupt
```



Fig. 6-36: Function block KRC_DeactivateInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Interrupt	INT	Number of the interrupt <ul style="list-style-type: none"> ■ 1 ... 8
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.35 Reading the state of an interrupt

Description

The function block KRC_ReadInterruptState reads the state of an interrupt in the advance run. This is updated cyclically.

A distinction can be made between the state of the interrupt in the advance run and the state in the main run.



Fig. 6-37: Function block KRC_ReadInterruptState

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Interrupt	INT	Number of the interrupt <ul style="list-style-type: none"> ■ 1 ... 8

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Value	INT	State of specified interrupt <ul style="list-style-type: none"> ■ 0: Interrupt has not been declared. ■ 1: Interrupt has been declared, but not activated. ■ 2: Interrupt has been declared and activated in the advance run. ■ 3: Interrupt has been declared and deactivated while still in the advance run. ■ 4: Interrupt has been triggered and is active. ■ 5: Interrupt has been triggered and the main program has already been resumed with KRC_Continue.
Error	BOOL	TRUE = error in function block
Valid	BOOL	TRUE = data are valid

6.10.36 Activating a path-related switching action (TRIGGER WHEN DISTANCE)

Description

The function block KRC_SetDistanceTrigger triggers a path-related switching action in the case of PTP or LIN motions.

The Trigger triggers a defined statement. The statement refers to the start point or end point of the motion block in which the Trigger is situated in the program. The statement is executed parallel to the robot motion.

The statement can be shifted in time. It is then not triggered exactly at the start or end point, but brought forward or delayed.

i 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.

i Further information on triggers, on offsetting the switching point and on the offset limits can be found in the Operating and Programming Instructions for System Integrators.

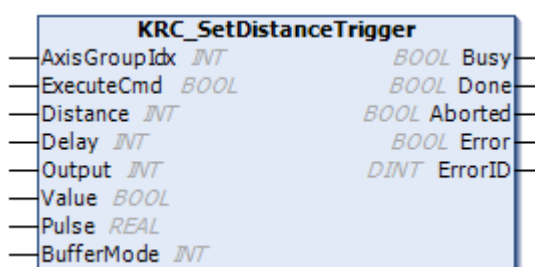


Fig. 6-38: Function block KRC_SetDistanceTrigger

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.

Parameter	Type	Description
Distance	INT	Switching point of the trigger <ul style="list-style-type: none"> ■ 0: Switching action at the start point ■ 1: Switching action at the end point
Delay	INT	Statement delay <ul style="list-style-type: none"> ■ Delay = 0 ms: no delay <p>The statement cannot be shifted freely in time. The shifts that are available depend on the value selected for Distance. Further information about this is contained in the Operating and Programming Instructions for System Integrators.</p>
Output	INT	Number of the digital output <ul style="list-style-type: none"> ■ 1 ... 2 048 <p>Note: It must be ensured that no outputs are used that are already assigned by the system. Example: \$OUT[1025] is always TRUE.</p>
Value	BOOL	TRUE = activate output FALSE = deactivate output
Pulse	REAL	Length of the pulse <ul style="list-style-type: none"> ■ 0.0 s No pulse active ■ 0.1 ... 3.0 s Pulse interval = 0.1 s; pulse durations outside this range of values trigger a program stop.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED <p>(>>> "BufferMode" Page 26)</p>

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed in the advance run <p>Note: The statement can no longer be aborted. Exception: Program is deselected or reset. The signal does not indicate whether the switching action has really been triggered.</p>
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.37 Activating a path-related switching action (TRIGGER WHEN PATH)

Description

The function block KRC_SetPathTrigger triggers a path-related switching action in the case of CP motions.

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. 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 end point, but beforehand or afterwards.

i Path triggers can only be activated before CP motions. If the subsequent motion is not a CP motion, the robot controller issues an error message.

i 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.

i Further information on triggers, on offsetting the switching point and on the offset limits can be found in the Operating and Programming Instructions for System Integrators.

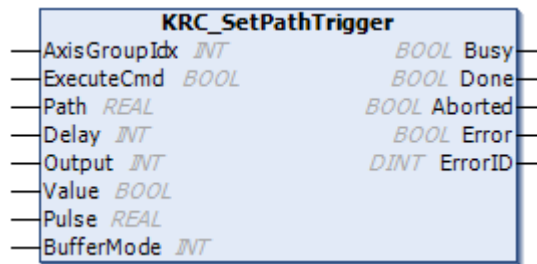


Fig. 6-39: Function block KRC_SetPathTrigger

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Path	REAL	Statement offset 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, Path is the distance to the position on the approximate positioning arc closest to the end point. <ul style="list-style-type: none"> Delay = 0.0 mm: no offset Delay > 0.0 mm: shifts the statement towards the end of the motion. Delay < 0.0 mm: shifts the statement towards the start of the motion.
Delay	INT	Statement delay <ul style="list-style-type: none"> Delay = 0 ms: no delay The statement cannot be shifted freely in time. The offsets that are possible depend on the value selected for Path . Further information about this is contained in the Operating and Programming Instructions for System Integrators.
Output	INT	Number of the digital output <ul style="list-style-type: none"> 1 ... 2 048 Note: It must be ensured that no outputs are used that are already assigned by the system. Example: \$OUT[1025] is always TRUE.

Parameter	Type	Description
Value	BOOL	TRUE = activate output FALSE = deactivate output
Pulse	REAL	Length of the pulse <ul style="list-style-type: none"> ■ 0.0 s No pulse active ■ 0.1 ... 3.0 s Pulse interval = 0.1 s; pulse durations outside this range of values trigger a program stop.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed in the advance run Note: The statement can no longer be aborted. Exception: Program is deselected or reset. The signal does not indicate whether the switching action has really been triggered.
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.10.38 Canceling a program

Description The function block KRC_Abort cancels all active and buffered statements and motions.



Fig. 6-40: Function block KRC_Abort

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.10.39 Stopping the robot

Description

The function block KRC_Interrupt triggers an interrupt with a BRAKE or BRAKE F statement.

i If a BRAKE statement is active, no more statements are processed via the mxA interface.
 The function block KRC_Abort is also no longer processed. KRC_Abort cannot cancel the program until it has been resumed with KRC_Continue, i.e. the BRAKE statement is no longer active. While the BRAKE statement is active, the program can only be canceled by means of a RESET of the function block KRC_AutomaticExternal.

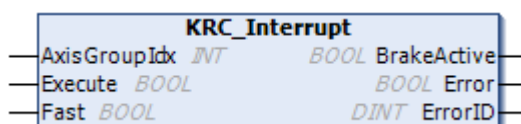


Fig. 6-41: Function block KRC_Interrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	The statement is executed in the case of a rising edge of the signal. The robot program is interrupted for as long as the input Execute is set to TRUE.
Fast	BOOL	TRUE = robot stops with a STOP 1 (BRAKE F statement). FALSE = robot stops with a STOP 2 (BRAKE statement).

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
BrakeActive	BOOL	TRUE = statement is active and robot is waiting for enabling
Error	BOOL	TRUE = error in function block

6.10.40 Continuing a program

Description

The function block KRC_Continue can be used to resume execution of a program that has been stopped by means of an interrupt.



Fig. 6-42: Function block KRC_Continue

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
Enable	BOOL	The statement is executed in the case of a rising edge of the signal. In the case of an interrupt triggered by a BRAKE or BRAKE F statement, the program can only be resumed when the interrupt is no longer active and the input Enable is set to TRUE.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.10.41 Automatically starting function blocks and signals

Description

The function block KRC_AutoStart controls the existing signals and function blocks in a typical sequence of statements. The signals for activating the Automatic External interface are checked prior to starting. If one or more signals is missing, corresponding error numbers are displayed.



The drives are activated with an edge. For this reason, the DRIVES_ON input on the function block KRC_AutomaticExternal should not be permanently activated.



Fig. 6-43: Function block KRC_AutoStart

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteReset	BOOL	Selects the mxAutomation robot program in the case of a rising edge of the signal and starts it. The program is reset beforehand and all buffered statements are aborted.

Outputs

Parameter	Type	Description
Busy	BOOL	The sequence is active but not yet completed.
Done	BOOL	The sequence is completed.
DispActive	BOOL	TRUE = robot program is active
ResetValid	BOOL	TRUE = conditions for a RESET at the function block KRC_AutomaticExternal are met
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.11 Functions for activating motions

i Motion instructions can only be executed in ABORTING or BUFFERED mode. If a motion is to be approximated, the following motion must be transferred in BUFFERED mode.

i Further information about the basics of motion programming – motion types, orientation control, and approximate positioning – is contained in the operating and programming instructions for the KUKA System Software.

6.11.1 Moving to a Cartesian position with a LIN motion

Description

The function block KRC_MoveLinearAbsolute executes a linear motion to a Cartesian end position. The coordinates of the end position are absolute.



Fig. 6-44: Function block KRC_MoveLinearAbsolute

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a LIN motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.

Parameter	Type	Description
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.2 Moving to a Cartesian position with a LIN_REL motion

Description

The function block KRC_MoveLinearRelative executes a linear motion to a Cartesian end position. The coordinates of the end position 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.

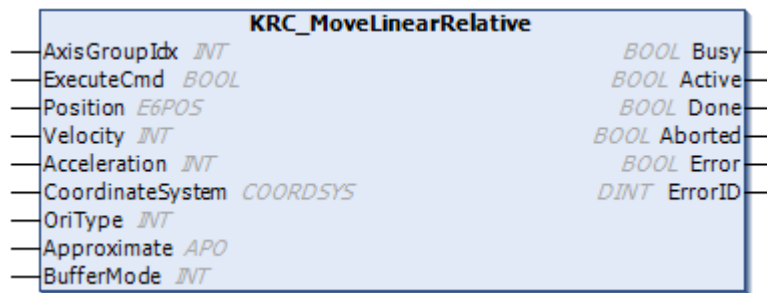


Fig. 6-45: Function block KRC_MoveLinearRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a LIN_REL motion, the Cartesian coordinates can refer to the BASE or TOOL coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)

Parameter	Type	Description
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.3 Moving to a Cartesian position with a PTP motion

Description

The function block `KRC_MoveDirectAbsolute` executes a point-to-point motion to a Cartesian end position. The coordinates of the end position are absolute.

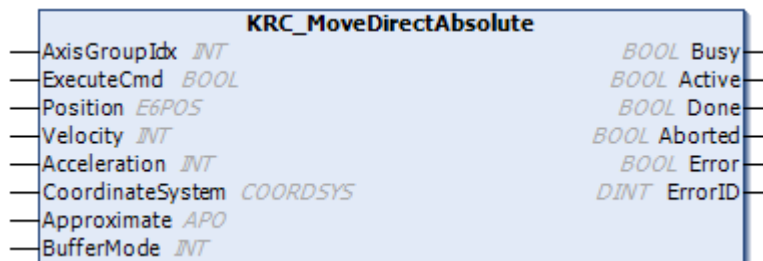


Fig. 6-46: Function block `KRC_MoveDirectAbsolute`

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a PTP motion, the Cartesian coordinates always refer to the BASE coordinate system.

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0 %
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.4 Moving to a Cartesian position with a PTP_REL motion

Description

The function block KRC_MoveDirectRelative executes a point-to-point motion to a Cartesian end position. The coordinates of the end position 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.



Fig. 6-47: Function block KRC_MoveDirectRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a PTP_REL motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
BufferMode	INT	Mode in which the statement is executed ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.5 Moving to an axis-specific position with a PTP motion

Description

The function block KRC_MoveAxisAbsolute executes a point-to-point motion to an axis-specific end position. The axis positions are absolute.



Fig. 6-48: Function block KRC_MoveAxisAbsolute

Inputs

Parameter	Type	Description
AxisPosition	E6AXIS	Axis-specific end position (>>> "E6AXIS" Page 25) The data structure E6Axis contains the angle values or translation values for all axes of the axis group in the end position.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)

Parameter	Type	Description
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.6 Moving to a Cartesian position with a CIRC motion

Description

The function block KRC_MoveCircAbsolute executes a circular motion to a Cartesian end position. In order for the robot controller to be able to calculate the circular motion, an auxiliary position must be specified in addition to the end position.

The coordinates of the auxiliary position and end position are absolute. The auxiliary position cannot be approximated. The motion always stops exactly at this point.

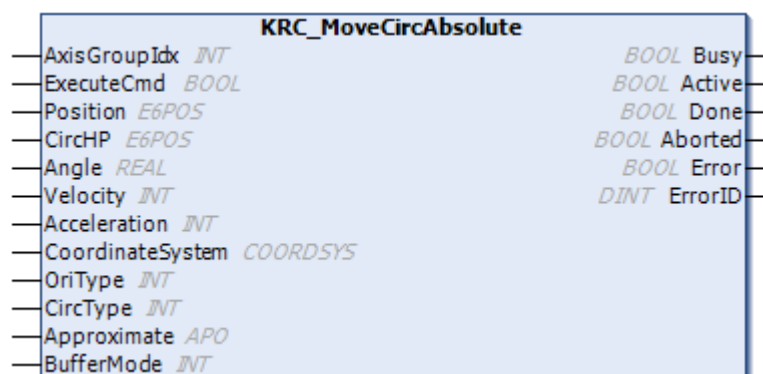


Fig. 6-49: Function block KRC_MoveCircAbsolute

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CircHP	E6POS	Coordinates of the Cartesian auxiliary position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the auxiliary position (= position of the TCP relative to the origin of the BASE coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer (>>> "COORDSYS" Page 25) Note: In the case of a CIRC motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Angle	REAL	Circular angle (= overall angle of the circular motion) The circular angle 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. The circular angle is not limited, i.e. a circular angle greater than $\pm 360^\circ$ can be specified: ■ > 0.0°: In the case of a positive angle, the motion is carried out from the start point via CircHP towards Position . ■ < 0.0°: In the case of a negative angle, the motion is carried out from the start point via Position towards CircHP . ■ = 0.0°: The circular angle is ignored. End position is Position . The radius of the circle is calculated on the basis of the start position, CircHP and Position . Default: 0.0°
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)

Parameter	Type	Description
Acceleration	INT	Acceleration <ul style="list-style-type: none"> 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> 0: VAR 1: CONSTANT 2: JOINT (>>> "OriType" Page 27)
CircType	INT	Orientation control during the circular motion <ul style="list-style-type: none"> 0: BASE 1: PATH (>>> "CircType" Page 27)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.7 Moving to a Cartesian position with a CIRC_REL motion

Description

The function block KRC_MoveCircRelative executes a circular motion to a Cartesian end position. In order for the robot controller to be able to calculate the circular motion, an auxiliary position must be specified in addition to the end position.

The coordinates of the auxiliary position and end position are relative to the current position (= start position of the circular motion). The auxiliary position cannot be approximated. The motion always stops exactly at this point.



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.



Fig. 6-50: Function block KRC_MoveCircRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CircHP	E6POS	Coordinates of the Cartesian auxiliary position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the auxiliary position (= position of the TCP relative to the origin of the BASE coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer (>>> "COORDSYS" Page 25) Note: In the case of a CIRC_REL motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.

Parameter	Type	Description
Angle	REAL	<p>Circular angle (= overall angle of the circular motion)</p> <p>The circular angle 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>The circular angle is not limited, i.e. a circular angle greater than $\pm 360^\circ$ can be specified:</p> <ul style="list-style-type: none"> ■ > 0.0°: In the case of a positive angle, the motion is carried out from the start point via CircHP towards Position. ■ < 0.0°: In the case of a negative angle, the motion is carried out from the start point via Position towards CircHP. ■ = 0.0°: The circular angle is ignored. End position is Position. The radius of the circle is calculated on the basis of the start position, CircHP and Position. <p>Default: 0.0°</p>
Velocity	INT	<p>Velocity</p> <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default: 0% (= velocity is not changed)</p>
Acceleration	INT	<p>Acceleration</p> <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default: 0% (= acceleration is not changed)</p>
OriType	INT	<p>Orientation control of the TCP</p> <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT <p>(>>> "OriType" Page 27)</p>
CircType	INT	<p>Orientation control during the circular motion</p> <ul style="list-style-type: none"> ■ 0: BASE ■ 1: PATH <p>(>>> "CircType" Page 27)</p>
BufferMode	INT	<p>Mode in which the statement is executed</p> <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED <p>(>>> "BufferMode" Page 26)</p>

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred

Parameter	Type	Description
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.8 Jogging to a relative end position

Description

The function block KRC_JogLinearRelative can be used to move to a Cartesian end position with a linear motion. The coordinates of the end position are relative to the current position. Status and Turn of the end position are ignored, i.e. the axis positions at the end position are not unambiguously defined.

The function is always executed in ABORTING mode, i.e. all active motions and buffered statements are canceled, the robot is braked and the linear motion is then executed.

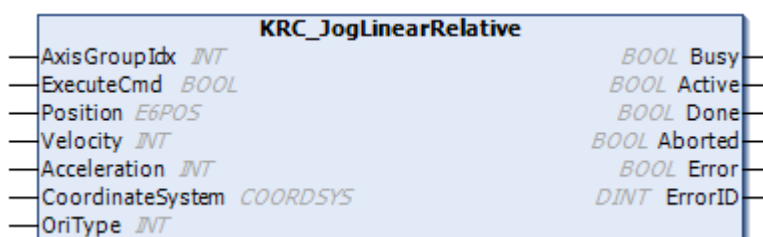


Fig. 6-51: Function block KRC_JogLinearRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)

Parameter	Type	Description
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.9 Jogging to a relative end position in the TOOL coordinate system

Description

The function block KRC_JogToolRelative can be used to move to a Cartesian end position in the TOOL coordinate system with a linear motion. The coordinates of the end position are relative to the current position. Status and Turn of the end position are ignored, i.e. the axis positions at the end position are not unambiguously defined.

The function is always executed in ABORTING mode, i.e. all active motions and buffered statements are canceled, the robot is braked and the linear motion is then executed.

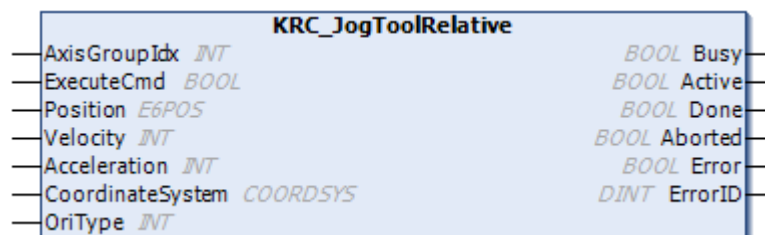


Fig. 6-52: Function block KRC_JogToolRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
OriType	INT	Orientation control of the TCP ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.11.10 Jogging to an end position

Description

The function block KRC_Jog can be used to move to an end position with a linear motion or a point-to-point motion.

The function is always executed in ABORTING mode, i.e. all active motions and buffered statements are canceled, the robot is braked and the motion is then executed.

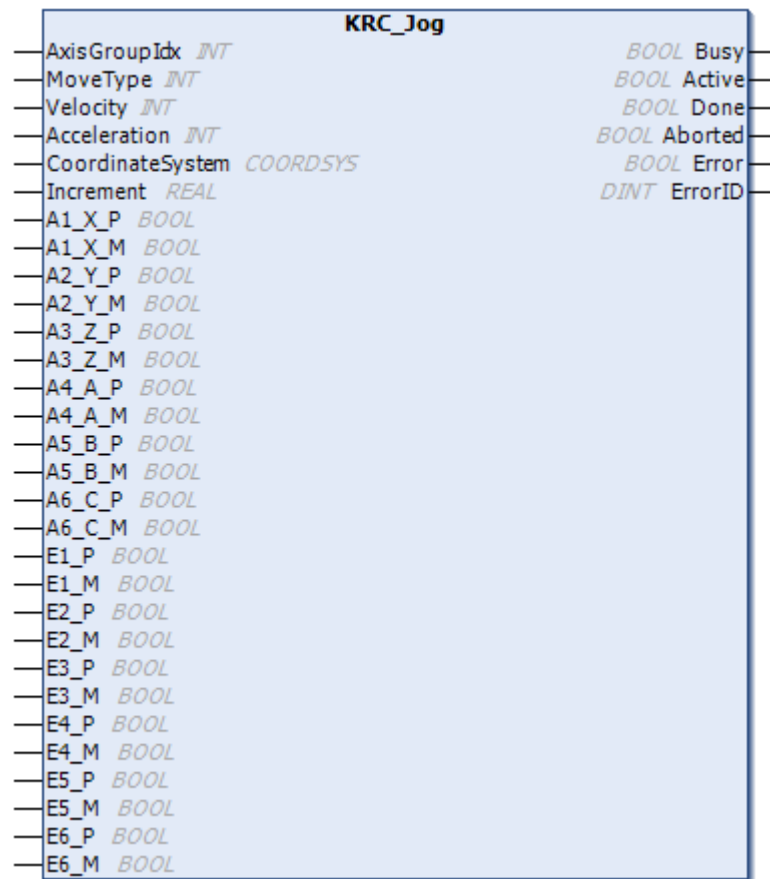


Fig. 6-53: Function block KRC_Jog

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
MoveType	INT	Motion type for Cartesian or axis-specific jogging <ul style="list-style-type: none"> ■ 0: Motion type PTP (axis-specific) ■ 1: Motion type LIN (Cartesian) ■ 2: Motion type SPTP (axis-specific) ■ 3: Motion type SLIN (Cartesian)
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default value: 0% (= velocity is not changed)</p>
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default value: 0% (= acceleration is not changed)</p>


Parameter	Type	Description
CoordinateSystem	COORDSYS	Coordinate system to which the coordinates of the end position refer. (>>> "COORDSYS" Page 25)
Increment	REAL	Incremental jogging <ul style="list-style-type: none"> ■ >0.0: The robot moves no more than the specified distance. For axis-specific jogging, the maximum distance is automatically limited to the software limit switches. For Cartesian jogging in the A, B or C direction, the maximum distance is limited to 90°. If the input signal is reset before the robot has reached the end position, the robot stops immediately. ■ ≤0.0: For Cartesian jogging in the X, Y or Z direction, the maximum distance is limited to 100000 mm. The robot motion restarts each time the input signals are changed.
A1_X_P	BOOL	Motion instruction The motion is started at a rising edge of the signal and stopped at a falling edge of the signal. In motion type PTP, the robot axes can be moved as far as 0.1 mm before the software limit switches. For this, the software limit switch values are read once when the PLC is started. During operation, the software limit switches can be updated via the input UpdateSoftEnd . Several robot axes can be moved simultaneously. The TCP can be moved along the axes of several coordinate systems. If the input signals are changed, the motion is stopped and then resumed with the modified configuration. If the positive and negative motion directions are activated simultaneously, an error number is displayed. The inputs for axes A1 ... A6 have a dual assignment with the coordinates, e.g. A1 with X, A2 with Y, etc.. The inputs that end in "P" (e.g. A2_Y_P) move in the positive direction. The inputs that end in "M" (e.g. A3_Z_M) move in the negative direction.
A1_X_M	BOOL	
A2_Y_P	BOOL	
A2_Y_M	BOOL	
A3_Z_P	BOOL	
A3_Z_M	BOOL	
A4_A_P	BOOL	
A4_A_M	BOOL	
A5_B_P	BOOL	
A5_B_M	BOOL	
A6_C_P	BOOL	
A6_C_M	BOOL	
E1_P	BOOL	
E1_M	BOOL	
E2_P	BOOL	
E2_M	BOOL	
E3_P	BOOL	
E3_M	BOOL	
E4_P	BOOL	
E4_M	BOOL	
E5_P	BOOL	
E5_M	BOOL	
E6_P	BOOL	
E6_M	BOOL	

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.12 Functions for activating motions (PLC OPEN-compliant)

The MC function blocks described below differ from the KRC function blocks in that they correspond to the PLC OPEN standard or are closer to it.

 For information on the frequently used signals in the MC function blocks, see (>>> 6.4 "Frequently used input/output signals in the MC function blocks" Page 22).

6.12.1 Moving to a Cartesian position with a LIN motion

Description The function block MC_MoveLinearAbsolute is used to execute a linear motion to a Cartesian end position. The coordinates of the end position are absolute.

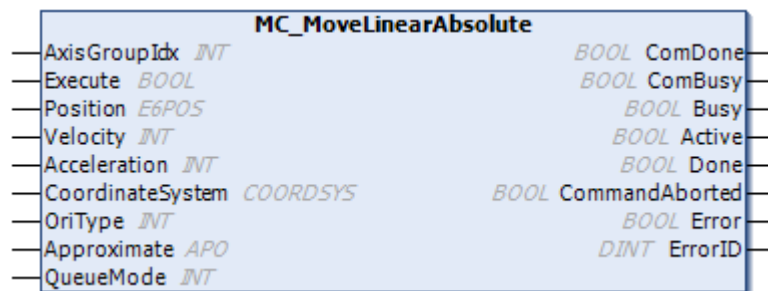


Fig. 6-54: Function block MC_MoveLinearAbsolute

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)

Parameter	Type	Description
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a LIN motion, the Cartesian coordinates always refer to the BASE coordinate system.
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)


Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.2 Moving to a Cartesian position with a LIN_REL motion

Description

The function block MC_MoveLinearRelative is used to execute a linear motion to a Cartesian end position. The coordinates of the end position 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.
---	--

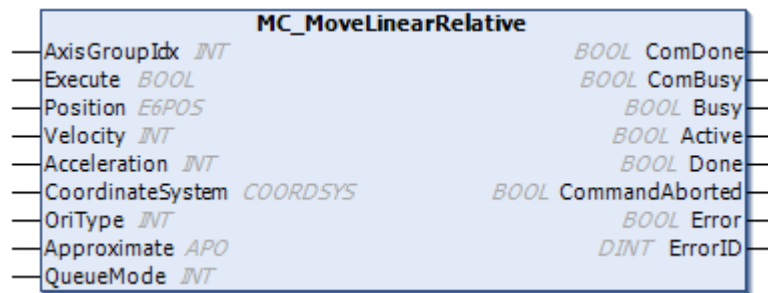


Fig. 6-55: Function block MC_MoveLinearRelative

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a LIN_REL motion, the Cartesian coordinates can refer to the BASE or TOOL coordinate system.
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT (>>> "OriType" Page 27)

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.3 Moving to a Cartesian position with a PTP motion

Description

The function block MC_MoveDirectAbsolute executes a point-to-point motion to a Cartesian end position. The coordinates of the end position are absolute.

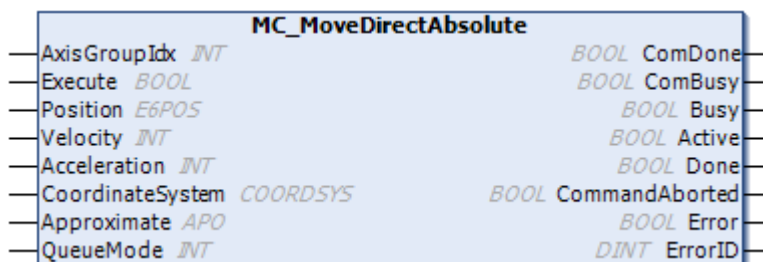


Fig. 6-56: Function block MC_MoveDirectAbsolute

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).

Parameter	Type	Description
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0 %
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a PTP motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.4 Moving to a Cartesian position with a PTP_REL motion

Description

The function block MC_MoveDirectRelative executes a point-to-point motion to a Cartesian end position. The coordinates of the end position are relative to the current position.

i 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.

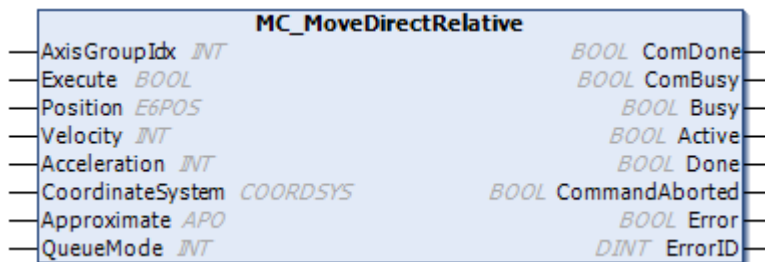


Fig. 6-57: Function block MC_MoveDirectRelative

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the selected coordinate system).
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> "COORDSYS" Page 25) Note: In the case of a PTP_REL motion, the Cartesian coordinates always refer to the BASE coordinate system.

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.5 Moving to an axis-specific position with a PTP motion

Description

The function block MC_MoveAxisAbsolute is used to execute a point-to-point motion to an axis-specific end position. The axis positions are absolute.

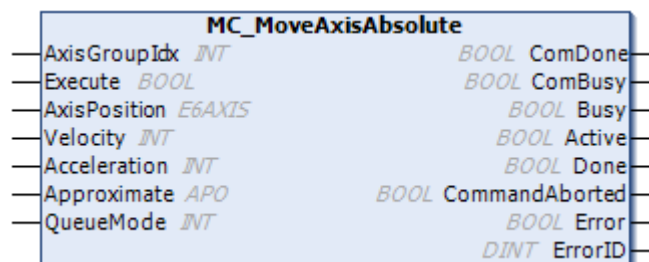


Fig. 6-58: Function block MC_MoveAxisAbsolute

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
AxisPosition	E6AXIS	Axis-specific end position (>>> "E6AXIS" Page 25) The data structure E6Axis contains the angle values or translation values for all axes of the axis group in the end position.

Parameter	Type	Description
Velocity	INT	Velocity <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.6 Moving to a Cartesian position with a CIRC motion

Description

The function block MC_MoveCircularAbsolute executes a circular motion to a Cartesian end position. In order for the robot controller to be able to calculate the circular motion, an auxiliary position must be specified in addition to the end position.

The coordinates of the auxiliary position and end position are absolute. The auxiliary position cannot be approximated. The motion always stops exactly at this point.

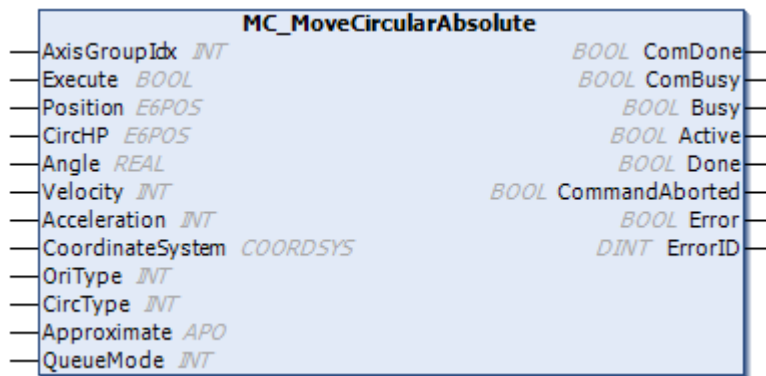


Fig. 6-59: Function block MC_MoveCircularAbsolute

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CircHP	E6POS	Coordinates of the Cartesian auxiliary position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the auxiliary position (= position of the TCP relative to the origin of the BASE coordinate system).
Angle	REAL	Circular angle (= overall angle of the circular motion) The circular angle 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. The circular angle is not limited, i.e. a circular angle greater than $\pm 360^\circ$ can be specified: <ul style="list-style-type: none"> ■ > 0.0°: In the case of a positive angle, the motion is carried out from the start point via CircHP towards Position. ■ < 0.0°: In the case of a negative angle, the motion is carried out from the start point via Position towards CircHP. ■ = 0.0°: The circular angle is ignored. End position is Position. The radius of the circle is calculated on the basis of the start position, CircHP and Position. Default: 0.0°

Parameter	Type	Description
Velocity	INT	Velocity <ul style="list-style-type: none"> 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration <ul style="list-style-type: none"> 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer (>>> "COORDSYS" Page 25) Note: In the case of a CIRC motion, the Cartesian coordinates always refer to the BASE coordinate system.
OriType	INT	Orientation control of the TCP <ul style="list-style-type: none"> 0: VAR 1: CONSTANT 2: JOINT (>>> "OriType" Page 27)
CircType	INT	Orientation control during the circular motion <ul style="list-style-type: none"> 0: BASE 1: PATH (>>> "CircType" Page 27)
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 1: ABORTING 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted

Parameter	Type	Description
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.12.7 Moving to a Cartesian position with a CIRC_REL motion

Description

The function block MC_MoveCircularRelative executes a circular motion to a Cartesian end position. In order for the robot controller to be able to calculate the circular motion, an auxiliary position must be specified in addition to the end position.

The coordinates of the auxiliary position and end position are relative to the current position (= start position of the circular motion). The auxiliary position cannot be approximated. The motion always stops exactly at this point.



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.

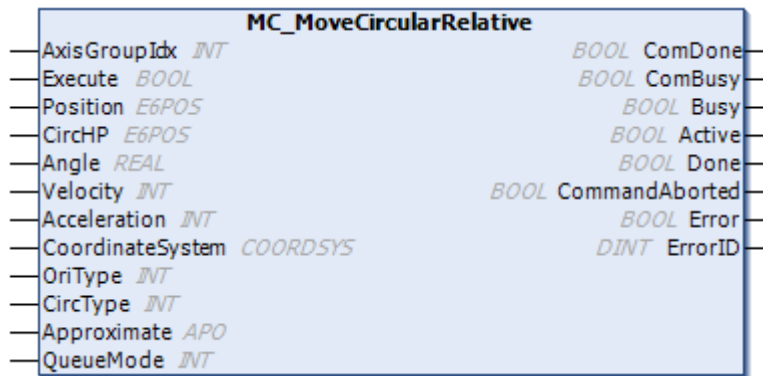


Fig. 6-60: Function block MC_MoveCircularRelative

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
Execute	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Position	E6POS	Coordinates of the Cartesian end position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the end position (= position of the TCP relative to the origin of the BASE coordinate system).
CircHP	E6POS	Coordinates of the Cartesian auxiliary position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the auxiliary position (= position of the TCP relative to the origin of the BASE coordinate system).

Parameter	Type	Description
Angle	REAL	<p>Circular angle (= overall angle of the circular motion)</p> <p>The circular angle 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>The circular angle is not limited, i.e. a circular angle greater than $\pm 360^\circ$ can be specified:</p> <ul style="list-style-type: none"> ■ > 0.0°: In the case of a positive angle, the motion is carried out from the start point via CircHP towards Position. ■ < 0.0°: In the case of a negative angle, the motion is carried out from the start point via Position towards CircHP. ■ = 0.0°: The circular angle is ignored. End position is Position. The radius of the circle is calculated on the basis of the start position, CircHP and Position. <p>Default: 0.0°</p>
Velocity	INT	<p>Velocity</p> <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default: 0% (= velocity is not changed)</p>
Acceleration	INT	<p>Acceleration</p> <ul style="list-style-type: none"> ■ 0 ... 100 % <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> <p>Default: 0% (= acceleration is not changed)</p>
CoordinateSystem	COORDSYS	<p>Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer</p> <p>(>>> "COORDSYS" Page 25)</p> <p>Note: In the case of a CIRC_REL motion, the Cartesian coordinates always refer to the BASE coordinate system.</p>
OriType	INT	<p>Orientation control of the TCP</p> <ul style="list-style-type: none"> ■ 0: VAR ■ 1: CONSTANT ■ 2: JOINT <p>(>>> "OriType" Page 27)</p>
CircType	INT	<p>Orientation control during the circular motion</p> <ul style="list-style-type: none"> ■ 0: BASE ■ 1: PATH <p>(>>> "CircType" Page 27)</p>

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> "APO" Page 24)
QueueMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "QueueMode" Page 27)

Outputs

Parameter	Type	Description
ComDone	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement is currently being transferred and has not yet been confirmed by the robot controller.
Busy	BOOL	TRUE = function block has not yet been executed completely.
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Command-Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.13 Diagnostic functions

6.13.1 Reading the current state of the mxA interface

Description The function block KRC_ReadMXAStatus reads the current state of the mxA interface.

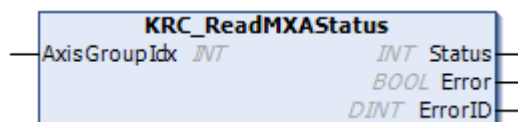


Fig. 6-61: Function block KRC_ReadMXAStatus

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5

Outputs

Parameter	Type	Description
Status	INT	Current state of the mxA interface (>>> "Status" Page 27)
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.13.2 Reading error messages of the mxA interface

Description The function block KRC_ReadMXAError is used to read the current error state of an axis group. Only error messages generated by the mxA interface are displayed.

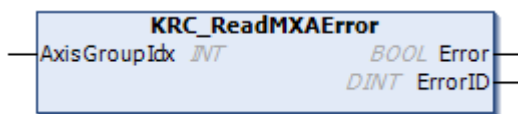


Fig. 6-62: Function block KRC_ReadMXAError

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.13.3 Resetting error messages of the mxA interface

Description The function block KRC_MessageReset resets the current error state of an axis group. Only error messages generated by the mxA interface are reset.

Error messages from the robot controller are reset by means of the function block KRC_AutomaticExternal (input CONF_MESS).

Messages can only be reset if the robot is stationary.

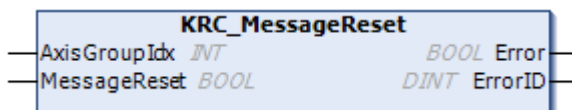


Fig. 6-63: Function block KRC_MessageReset

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
MessageReset	BOOL	TRUE = reset message


Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Error	BOOL	TRUE = error in function block

6.13.4 Reading error messages of the robot controller

Description The function block KRC_ReadKRCError reads the current error state of the robot controller. Only error messages generated by the robot controller are displayed.

Error messages from the robot controller are reset by means of the function block `KRC_AutomaticExternal` (input `CONF_MESS`).

 Messages can only be reset if the robot is stationary.

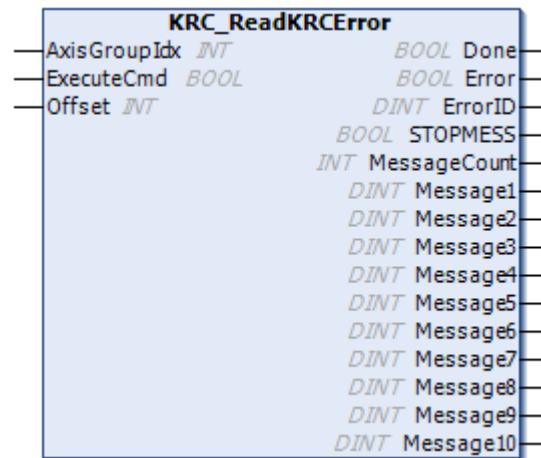


Fig. 6-64: Function block `KRC_ReadKRCError`

Inputs


Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Offset	INT	If there are more than 10 messages in the message buffer, the desired start index of the message buffer can be selected using the offset. Example: If there are 15 messages in the message buffer, the offset must be 6 in order to read messages 6 to 15.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
MessageCount	INT	Number of messages in the message buffer
Message1 ... Message10	DINT	The numbers of up to 10 messages in the message buffer can be output.
Done	BOOL	TRUE = data are valid
Error	BOOL	TRUE = error in function block
STOPMESS	BOOL	TRUE = safety circuit is interrupted (robot fault)

6.13.5 Reading diagnostic signals

Description The function block `KRC_Diag` reads the diagnostic signals of the robot controller.

 The function block may only be instanced once per axis group. In the case of multiple instancing, the signals of the most recently called function block are output.

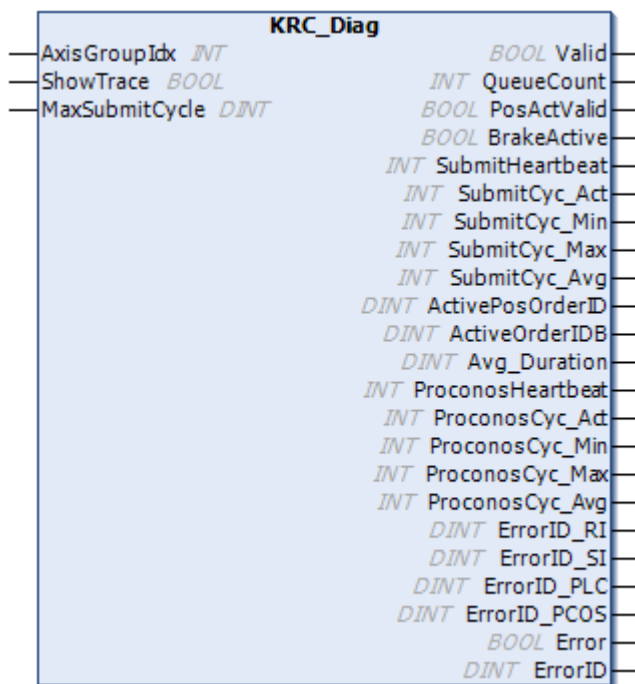


Fig. 6-65: Function block KRC_Diag

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ShowTrace	BOOL	TRUE = activate display of the active function blocks in the message window of the KUKA smartHMI. FALSE = deactivate display of the active function blocks in the message window of the KUKA smartHMI. Note: Only activate the display for test and diagnostic purposes. If the display is active, approximate positioning is no longer possible and the cycle time of the submit interpreter is adversely affected.
MaxSubmitCycle	INT	Maximum cycle time of the submit interpreter Default: 1 000 ms Note: If the maximum cycle time is exceeded, the \$MOVE_ENABLE signal for motion enable is reset.

Outputs

Parameter	Type	Description
QueueCount	INT	Number of buffered statements <ul style="list-style-type: none"> 1 ... 90
SubmitHeartbeat	INT	Heartbeat signal of the submit interpreter (counter is incremented by 1 every Submit cycle) <ul style="list-style-type: none"> 1 ... 245
SubmitCyc_Act	REAL	Current cycle time of the submit interpreter; unit: ms Mean value over 1,000 ms = 1/number of cycles per second

Parameter	Type	Description
SubmitCyc_Min	REAL	Shortest cycle time of the submit interpreter since the last broken connection; unit: ms
SubmitCyc_Max	REAL	Longest cycle time of the submit interpreter since the last broken connection; unit: ms
SubmitCyc_Avg	INT	Mean value of the cycle time of the submit interpreter during the calculation period Avg_Duration ; unit: ms
Avg_Duration	DINT	Duration of the current calculation period for the mean value of the cycle time; unit: ms The calculation period is restarted after a break in the connection to the submit interpreter or, at the latest, after 60 minutes.
ProconosHeartbeat	INT	Life sign from ProConOS (counter is incremented by 1 every ProConOS cycle)
ProconosCyc_Act	INT	Current cycle time of ProConOS; unit: ms Mean value over 1,000 ms = 1/number of cycles per second
ProconosCyc_Min	INT	Shortest cycle time of ProConOS since the last broken connection; unit: ms
ProconosCyc_Max	INT	Longest cycle time of ProConOS since the last broken connection; unit: ms
ProconosCyc_Avg	INT	Mean value of the cycle time of ProConOS during the calculation period Avg_Duration ; unit: ms
ActivePosOrderID	DINT	Order ID of the KRC_Move motion command that is currently being executed
ActiveOrderIDB	DINT	Order ID of the current KRC_Move motion command in the advance run
ErrorID_RI	DINT	Robot interpreter error number
ErrorID_SI	DINT	Submit interpreter error number
ErrorID_PLC	DINT	PLC error number
ErrorID_PCOS	DINT	ProConOS error number
ErrorID	DINT	Error number
Valid	BOOL	TRUE = data are valid
PosActValid	BOOL	TRUE = position data are valid (BCO)
BrakeActive	BOOL	TRUE = robot is stopped by means of a BRAKE statement
Error	BOOL	TRUE = error in function block

6.13.6 Reading and acknowledging error states

Description

The function block KRC_Error collectively reads and acknowledges the current error state of the mxA interface, the error state of the robot controller and the error state of the function blocks.

If more than one error has occurred in the function block at the same time, only the error number of the most recent error is displayed. Errors in a function block cause the motion enable to be canceled.

If more than one error has occurred at the same time, these errors are displayed with the following priority ranking:

1. Errors of the mxA interface in the robot interpreter
2. Errors of the mxA interface in the submit interpreter
3. ProConOS errors
4. Errors in the PLC
5. Errors in a function block of the local PLC
6. Errors in the robot controller

The function block KRC_Error contains all diagnostic function blocks, which means that this block displays all important diagnostic data.

i The \$STOPMESS signal can only be reset if there is no error and the drives are switched on. The drives must therefore be activated with the DRIVES_ON signal via the function block KRC_Error. The drives are activated with an edge. For this reason, the DRIVES_ON input on the function block KRC_AutomaticExternal should not be permanently activated.

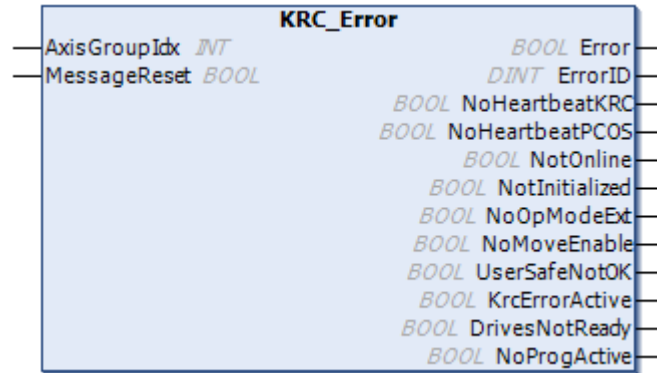


Fig. 6-66: Function block KRC_Error

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
MessageReset	BOOL	Resets error messages of the mxA interface and the robot controller via the function block KRC_AutomaticExternal (input CONF_MESS). Activates the drives. TRUE = reset message Note: The messages can only be reset if the robot is stationary.

Outputs

Parameter	Type	Description
Error	BOOL	TRUE = error in function block
ErrorID	INT	Error number
NoHeartbeatKRC	BOOL	The submit interpreter is not sending a life sign
NoHeartbeatPCOS	BOOL	ProConOS is not sending a life sign
NotOnline	BOOL	No connection to robot controller
NotInitialized	BOOL	No statements can be executed, as the connection has not been initialized.
NoOpModeExt	BOOL	Robot is not in Automatic External mode
NoMoveEnable	BOOL	No motion enable present.
UserSafeNotOK	BOOL	The operator safety is violated. The \$USER_SAF signal of the Automatic External interface is not active.
KrcErrorActive	BOOL	Error messages of the robot controller are active. The \$STOPMESS signal of the Automatic External interface is active.

Parameter	Type	Description
DrivesNotReady	BOOL	The drives are not ready. The \$PERI_RDY signal of the Automatic External interface is not active.
NoProgActive	BOOL	The robot program is not active. The \$PRO_ACT signal of the Automatic External interface is not active.

6.14 Special functions

6.14.1 Reading system variables

Description The function block KRC_ReadSysVar reads a system variable. This function is executed in the Submit interpreter.



Fig. 6-67: Function block KRC_ReadSysVar

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Index	INT	Index of the system variable <ul style="list-style-type: none"> ■ 1: \$ADVANCE



So far, only the system variable \$ADVANCE can be read. If required for the customer-specific application, the list of readable system variables can be expanded by KUKA.

Outputs

Parameter	Type	Description
Value1 ... Value10	REAL	Value of the system variable If the system variable is a structure type, up to 10 components of the structure can be read.
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.14.2 Writing system variables

Description The function block KRC_WriteSysVar writes a system variable.

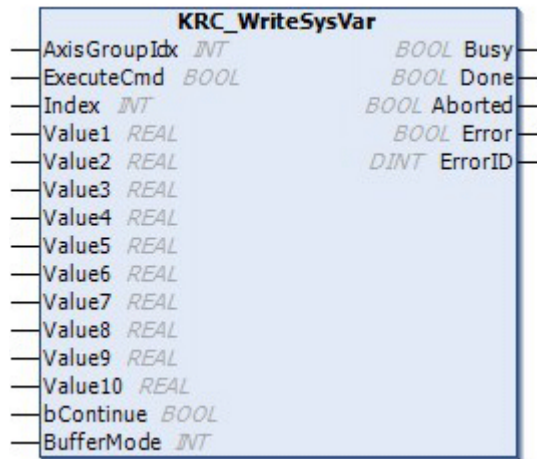


Fig. 6-68: Function block KRC_WriteSysVar

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.
Index	INT	Index of the system variable <ul style="list-style-type: none"> 1: \$ADVANCE
Value1 ... Value10	REAL	Value of the system variable If the system variable is a structure type, up to 10 components of the structure can be written.
Continue	BOOL	TRUE = write to system variable without advance run stop Note: Only possible for specific system variables.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)


i So far, only the system variable \$ADVANCE can be written. If required for the customer-specific application, the list of writable system variables can be expanded by KUKA.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.3 Calling a brake test

Description The function block `KRC_BrakeTest` calls the program for the brake test. The brake test is started at the position at which the robot is located when the program is called.


 The brake test must be performed with a program override of 100% (function block `KRC_SetOverride`).


During the brake test, all brakes are checked to see whether the wear limit has been reached. For this purpose, the robot accelerates to a defined velocity limit. Once the robot has reached the velocity, the brake is applied and the result for this braking operation is displayed.

If the brake test is successful, the robot is located back at the start position at the end of the measurement.

If the brake test fails, i.e. a brake has been identified as being defective, the robot moves directly to a parking position. The coordinates of the parking position must be specified in the function block.

Parking position The parking position must be selected in a position where no persons are endangered if the robot sags because of the defective brake. The transport position, for example, can be selected as the parking position.

 Further information about the transport position is contained in the robot operating or assembly instructions.

 Detailed information about the brake test is contained in the Operating and Programming Instructions for System Integrators.

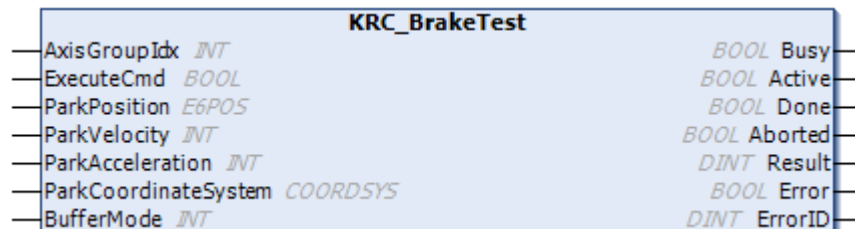


Fig. 6-69: Function block `KRC_BrakeTest`

Inputs

Parameter	Type	Description
ParkPosition	E6POS	Coordinates of the Cartesian parking position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the parking position (= position of the TCP relative to the origin of the selected coordinate system).
ParkCoordinate-System	COORDSYS	Coordinate system to which the Cartesian coordinates of the parking position refer (>>> "COORDSYS" Page 25)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.

Parameter	Type	Description
ParkVelocity	INT	Velocity <ul style="list-style-type: none"> 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
ParkAcceleration	INT	Acceleration <ul style="list-style-type: none"> 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
Result	DINT	Result of the brake test <ul style="list-style-type: none"> 0: brake test failed (brake identified as defective or no connection to robot controller) 1: brake test successful (no brake defective, but at least one brake has reached the wear limit) 2: brake test successful (no brake defective or reached wear limit)
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block

6.14.4 Calling a mastering test

Description

The function block KRC_MasRef is used to execute the mastering test.

After the function block has been called, the robot moves in a linear direction from the current position to the reference position. Once the robot has reached the reference position, the current axis values are compared with the axis values which have been saved in KUKA.SafeOperation. The robot then moves back to the start position (= position before the function block was called).



The reference position is defined in the function block with the input parameter **Position** and corresponds to the reference position defined with KUKA.SafeOperation.

If the deviation between the current position and the reference position is too great, the mastering test has failed.


 Detailed information about the mastering test can be found in the documentation **KUKA.SafeOperation**.



Fig. 6-70: Function block KRC_MasRef

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian reference position (>>> "E6POS" Page 25) The data structure E6POS contains all components of the reference position (= position of the TCP relative to the origin of the selected coordinate system).
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the reference position refer (>>> "COORDSYS" Page 25)
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Velocity	INT	Velocity ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= velocity is not changed)
Acceleration	INT	Acceleration ■ 0 ... 100 % Refers to the maximum value specified in the machine data. The maximum value depends on the robot type and the selected operating mode. Default: 0% (= acceleration is not changed)
BufferMode	INT	Mode in which the statement is executed ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
MasRefRequest	BOOL	TRUE = mastering test has been requested internally by the robot controller.

6.14.5 Reading the safety controller signals

Description

The function block KRC_ReadSafeOPStatus reads signals of the safety controller. (Only relevant if KUKA.SafeOperation is installed.)

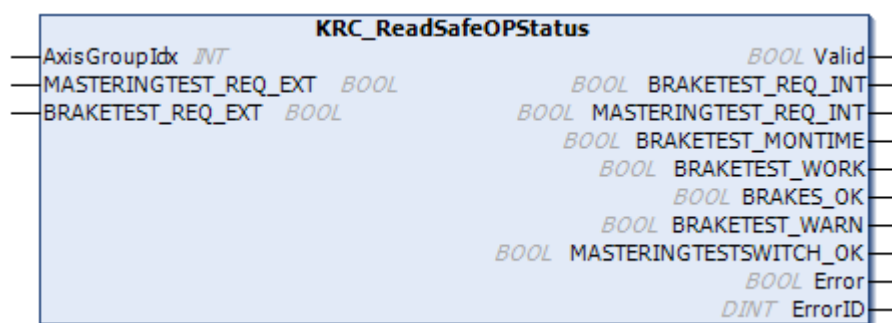


Fig. 6-71: Function block KRC_ReadSafeOPStatus

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
MASTERINGTEST_REQ_EXT	BOOL	TRUE = mastering test requested by the PLC.
BRAKETEST_REQ_EXT	BOOL	TRUE = brake test requested by the PLC.

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Valid	BOOL	TRUE = data are valid
BRAKETEST_REQ_INT	BOOL	TRUE = brake test requested by the safety controller.
MASTERINGTEST_REQ_INT	BOOL	TRUE = mastering test requested by the safety controller.
BRAKETEST_MONTIME	BOOL	TRUE = robot was stopped due to elapsed brake test monitoring time.
BRAKETEST_WORK	BOOL	TRUE = brake test is currently being performed
BRAKES_OK	BOOL	Edge TRUE --> FALSE : A brake has been identified as defective.
BRAKETEST_WARN	BOOL	Edge FALSE --> TRUE : At least 1 brake has been detected as having reached the wear limit.

Parameter	Type	Description
MASTERINGTESTSWITCH_OK	BOOL	TRUE = reference switch is OK.
Error	BOOL	TRUE = error in function block

6.14.6 Reading the state of the TouchUp status keys

Description The function block KRC_ReadTouchUPState reads the current state of the TouchUp status keys on the smartPAD. In order to be able to teach points using the status keys on the smartPAD, the function block must be linked to the function block KRC_TouchUP.

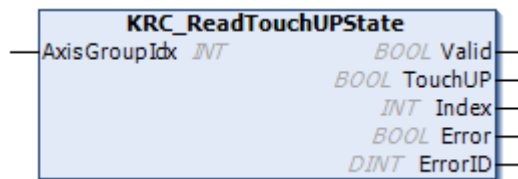


Fig. 6-72: Function block KRC_ReadTouchUPState

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
Index	INT	Number selected using the status key on the smartPAD to teach a position ■ 1 ... 100
ErrorID	DINT	Error number
Valid	BOOL	TRUE = data are valid
TouchUP	BOOL	State of the TouchUp status key on the smart-PAD TRUE = TouchUp status key has been pressed.
Error	BOOL	TRUE = error in function block

6.14.7 Teaching points

Description The function block KRC_TouchUP can be used to teach a point directly in the PLC. Tool, base and interpolation mode of this point are automatically set by the function block.

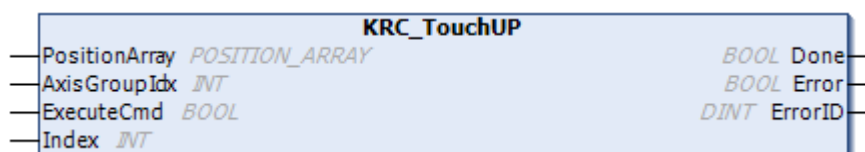


Fig. 6-73: Function block KRC_TouchUP

Memory

Parameter	Type	Description
PositionArray	Array[100]	List of the taught positions

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	TRUE = the point is taught.
Index	INT	Number under which the taught point is saved in the PLC <ul style="list-style-type: none"> 1 ... 100

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block

6.14.8 Modifying settings for the advance run

Description

The settings for the advance run are modified using the function block KRC_SetAdvance.

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 is required, for example, in order to be able to calculate approximate positioning motions.

i If the program execution is reset, the set values are reset to the default values.



Fig. 6-74: Function block KRC_SetAdvance

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Count	INT	Number of functions to be transferred before the first robot motion <ul style="list-style-type: none"> 1 ... 50 Default value: 2

Parameter	Type	Description
MaxWaitTime	INT	Maximum wait time before the beginning of program execution if the set number of functions is not reached in the parameter count. <ul style="list-style-type: none"> 1 ... 60 000 ms Default value: 300 ms
Mode	INT	Wait mode <ul style="list-style-type: none"> 0: The currently set mode is not changed. 1: If the first instruction is an approximated motion instruction, the system waits for further instructions. 2: The system always waits for the number of set functions or for the maximum wait time to elapse. Default value: 1
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement has been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.9 Reading values from KRC_SetAdvance

Description

Function block KRC_GetAdvance reads the values that have been set in the function block KRC_SetAdvance.

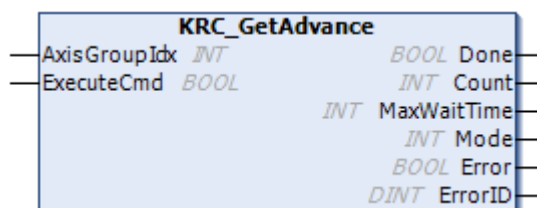


Fig. 6-75: Function block KRC_GetAdvance

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	The statement is executed in the case of a rising edge of the signal.

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Count	INT	Number of functions to be transferred before the first robot motion <ul style="list-style-type: none"> 1 ... 50 Default value: 2
MaxWaitTime	INT	Maximum wait time before the beginning of program execution if the set number of functions is not reached in the parameter count. <ul style="list-style-type: none"> 1 ... 60 000 ms Default value: 300 ms
Mode	INT	Wait mode <ul style="list-style-type: none"> 0: The currently set mode is not changed. 1: If the first instruction is an approximated motion instruction, the system waits for further instructions. 2: The system always waits for the number of set functions or for the maximum wait time to elapse. Default value: 1
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.10 Calculating the Cartesian robot position from the axis angles

Description

The function block KRC_Forward uses specified axis angles to calculate the Cartesian robot position. The function block can only be executed by the robot interpreter.

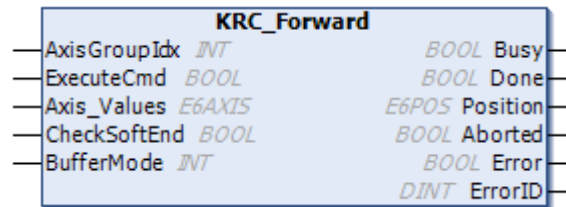


Fig. 6-76: Function block KRC_Forward

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the statement at a rising edge of the signal.
Axis_Values	E6AXIS	Axis-specific values that are to be converted to Cartesian coordinates The data structure E6AXIS contains the angle values or translation values for all axes of the axis group in this position.

Parameter	Type	Description
Check-SoftEnd	BOOL	Checks whether the specified axis angles lie within the software limit switches. If not, an error number is displayed.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Position	E6POS	Cartesian robot position calculated from the specified axis angles
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.11 Calculating axis angles from the Cartesian robot position

Description

The function block KRC_Inverse uses a specified Cartesian robot position to calculate the axis angles. The function block can only be executed by the robot interpreter.

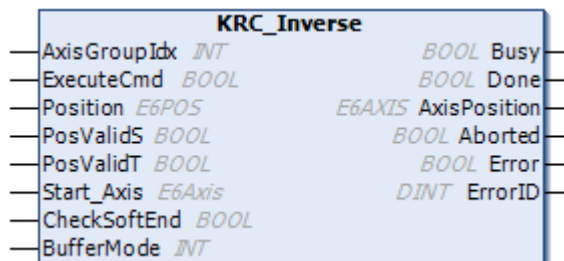


Fig. 6-77: Function block KRC_Inverse

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the statement at a rising edge of the signal.
Position	E6POS	Cartesian robot position
PosValidS	BOOL	TRUE = The Status value contained in the Position parameter is valid. FALSE = The Status value is unknown.
PosValidT	BOOL	TRUE = The Turn value contained in the Position parameter is valid. FALSE = The Turn value is unknown.

Parameter	Type	Description
Start_Axis	E6Axis	Axis-specific values at the start point of the motion The start point is the axis-specific position from which the robot moves to the position that is to be calculated.
Check-SoftEnd	BOOL	Checks whether the values from the Start_Axis parameter lie within the software limit switches. If not, an error number is displayed.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
AxisPosition	E6AXIS	Axis angles that have been calculated from the specified Cartesian robot position The data structure E6AXIS contains all the axis positions of the axis group.
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.12 Initializing a conveyor

Description

The function block KRC_ConvIniOff is used to initialize a conveyor. The AMI is set to the state #INITIALIZED and the conveyor distance to 0.



Fig. 6-78: Function block KRC_ConvIniOff

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.

Parameter	Type	Description
Conveyor-Number	INT	Number of the conveyor <ul style="list-style-type: none"> ■ 1 ... 3
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.13 Activating a conveyor

Description

The function block KRC_ConvOn activates the AMI, i.e. sets it to the #ACTIVE state. If the AMI is activated, the synchronization signals at the input of interface X33 (Fast Measurement) are evaluated.

The conveyor offset can be detected out in the background leaving the robot controller free to perform other tasks. This allows the robot to carry out on-the-fly tracking of a part on the conveyor.



Fig. 6-79: Function block KRC_ConvOn

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Conveyor-Number	INT	Number of the conveyor <ul style="list-style-type: none"> ■ 1 ... 3
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs


Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.14 Tracking a workpiece on the conveyor

Description

The function block KRC_ConvFollow enables the robot to follow a workpiece on the conveyor. KRC_ConvFollow can be used to define a range on the conveyor in which the robot starts to track the workpiece.

If the workpiece has already exceeded the maximum conveyor distance (input **MaxDistance**) when the function block is called, the output **MaxDistanceReached** is set.


This function block can only be executed if the AMI has been activated using KRC_ConvOn.

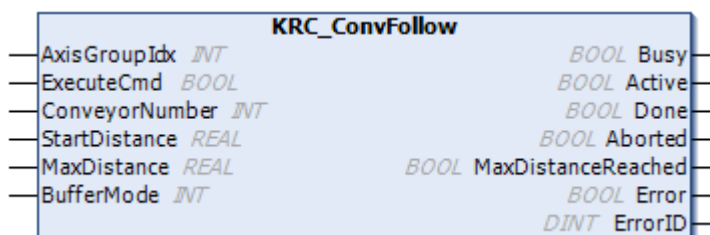


Fig. 6-80: Function block KRC_ConvFollow

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Conveyor-Number	INT	Number of the conveyor <ul style="list-style-type: none"> ■ 1 ... 3
StartDistance	REAL	Distance traveled by the workpiece while the robot waits before starting to track the workpiece on the conveyor. <ul style="list-style-type: none"> ■ In the case of a linear conveyor: Specification in millimeters ■ In the case of a circular conveyor: Specification in degrees

Parameter	Type	Description
MaxDistance	REAL	<p>Maximum distance traveled by the workpiece before the robot starts to synchronize itself with the workpiece.</p> <ul style="list-style-type: none"> ■ In the case of a linear conveyor: Specification in millimeters ■ In the case of a circular conveyor: Specification in degrees <p>Note: This input is not monitored during synchronized motions of the conveyor. The distance covered by the workpiece is monitored by an interrupt. The corresponding settings are made in WorkVisual.</p>
BufferMode	INT	<p>Mode in which the statement is executed</p> <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED <p>(>>> "BufferMode" Page 26)</p>

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement has been aborted
MaxDistanceReached	BOOL	TRUE = the maximum distance traveled by the workpiece (input MaxDistance) was already exceeded at the time of execution. The statement was not executed. Execution of the program is stopped (WAIT FOR FALSE) and is waiting for the program to be aborted.
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.15 Picking up a workpiece from the conveyor

Description

The function block KRC_ConvSkip is used to determine which workpieces are to be picked up, e.g. every second workpiece, every third workpiece, etc. A total of up to 1024 workpieces can be monitored in the background.

If the workpiece has already exceeded the maximum conveyor distance (input **MaxDistance**) when the function block is called, the output **MaxDistanceReached** is set.



This function block can only be executed if the AMI has been activated using KRC_ConvOn.



Fig. 6-81: Function block KRC_ConvSkip

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Conveyor-Number	INT	Number of the conveyor <ul style="list-style-type: none"> 1 ... 3
PieceNumber	INT	The number entered specifies which workpieces are to be picked up. Examples: <ul style="list-style-type: none"> 1: Every workpiece is picked up. 3: Every 3rd workpiece is picked up. 5: Every 5th workpiece is picked up.
StartDistance	REAL	Distance traveled by the workpiece while the robot waits before starting to track the workpiece on the conveyor. <ul style="list-style-type: none"> In the case of a linear conveyor: Specification in millimeters In the case of a circular conveyor: Specification in degrees
MaxDistance	REAL	Maximum distance traveled by the workpiece before the robot starts to synchronize itself with the workpiece. <ul style="list-style-type: none"> In the case of a linear conveyor: Specification in millimeters In the case of a circular conveyor: Specification in degrees <p>Note: This input is not monitored during synchronized motions of the conveyor. The distance covered by the workpiece is monitored by an interrupt. The corresponding settings are made in WorkVisual.</p>
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement has been aborted
MaxDistanceReached	BOOL	TRUE = the maximum distance traveled by the workpiece (input MaxDistance) was already exceeded at the time of execution. The statement was not executed. Execution of the program is stopped (WAIT FOR FALSE) and is waiting for the program to be aborted.
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.16 Activating interrupts for monitoring

Description

The function block KRC_ActivateConvInterrupt activates the following interrupts:

- Alarm distance monitoring
- Maximum distance monitoring
- \$STOPMESS error monitoring

An interrupt cannot be processed until the interrupt has been activated by the main run of the robot interpreter.

The monitoring functions are activated by the function blocks KRC_ConvFollow and KRC_ConvSkip insofar as these have been successfully synchronized with a workpiece. Calling this function block is only necessary if the monitoring function is to be ended and reactivated.



Fig. 6-82: Function block KRC_ActivateConvInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Conveyor-Number	INT	Number of the conveyor <ul style="list-style-type: none"> ■ 1 ... 3
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.17 Deactivating interrupts for monitoring

Description

The function block KRC_DeactivateConvInterrupt deactivates the following interrupts:

- Alarm distance monitoring
- Maximum distance monitoring
- \$STOPMESS error monitoring


 It is advisable to call this function block if leaving the conveyor area, or if monitoring is not desired.



Fig. 6-83: Function block KRC_DeactivateConvInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Conveyor-Number	INT	Number of the conveyor ■ 1 ... 3
BufferMode	INT	Mode in which the statement is executed ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.18 Activating a motion along a vector

Description The function block KRC_VectorMoveOn is used to move a robot along a defined vector in Cartesian space. Here, the robot is moved by an external force.

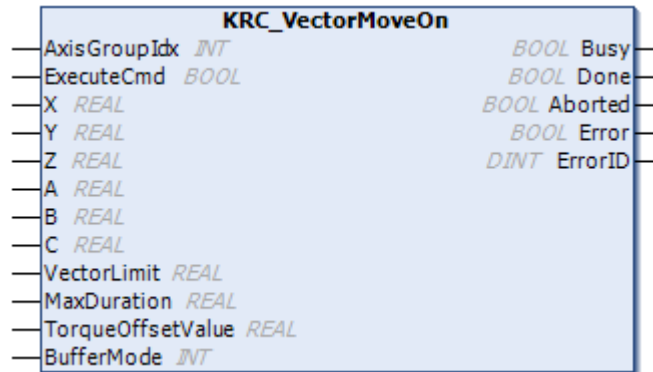


Fig. 6-84: Function block KRC_VectorMoveOn

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
X	REAL	Defines the direction of the vector The vector must be specified in relation to the TCP of the TOOL coordinate system. The last taught point before the function block is the root point of the vector.
Y	REAL	
Z	REAL	
A	REAL	
B	REAL	
C	REAL	Limits: <ul style="list-style-type: none"> ■ Translational motion (X, Y, Z): max. 200 mm in the positive and negative direction ■ Rotational motion (A, B, C): max. 30° in the positive and negative direction
VectorLimit	REAL	Permissible vector length limit; unit: % If this value is exceeded, the robot is brought to a halt with ramp-down braking.
MaxDuration	REAL	Length of time after which VectorMove is deactivated if an error has occurred

Parameter	Type	Description
TorqueOffset-Value	REAL	Defines the resistance torque of the robot The resistance torque is the torque with which the robot acts against the external force. It has the effect that the robot only begins to move when a specific amount of force is exerted. The resistance torque can be defined in addition to the holding torque. The holding torque depends on the robot position, type, size and additional load.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.19 Deactivating KRC_VectorMoveOn

Description

The function block KRC_VectorMoveOff is used to deactivate the function block KRC_VectorMoveOn.



Fig. 6-85: Function block KRC_VectorMoveOff

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.20 Configuring Cartesian workspaces

Description

The function block KRC_WriteWorkspace is used to configure Cartesian (= cubic) workspaces for the robot. Workspaces serve to protect the system. A maximum of 8 Cartesian workspaces can be configured at any one time. The workspaces may overlap.

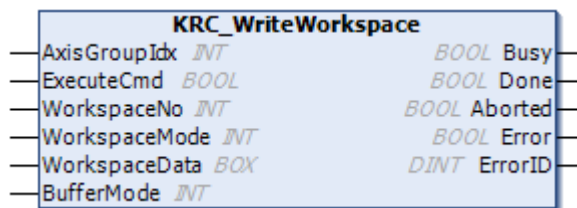


Fig. 6-86: Function block KRC_WriteWorkspace

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> ■ 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
WorkspaceNo	INT	Number of workspace <ul style="list-style-type: none"> ■ 1 ... 8
Workspace-Mode	INT	Mode for workspaces <ul style="list-style-type: none"> ■ 0: #OFF ■ 1: #INSIDE ■ 2: #OUTSIDE ■ 3: #INSIDE_STOP ■ 4: #OUTSIDE_STOP <p>Note: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).</p>
Workspace-Data	BOX	Data of the workspace (>>> 6.7 "Data of a Cartesian workspace" Page 28)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> ■ 0: DIRECT ■ 1: ABORTING ■ 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.21 Reading the configuration of Cartesian workspaces

Description

The function block KRC_ReadWorkspace reads the configuration of the Cartesian workspaces for the robot.

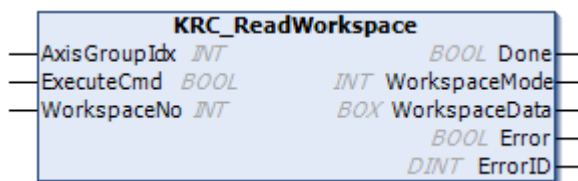


Fig. 6-87: Function block KRC_ReadWorkspace

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
WorkspaceNo	INT	Number of workspace <ul style="list-style-type: none"> 1 ... 8

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Workspace-Mode	INT	Mode for workspaces <ul style="list-style-type: none"> 0: #OFF 1: #INSIDE 2: #OUTSIDE 3: #INSIDE_STOP 4: #OUTSIDE_STOP <p>Note: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).</p>
Workspace-Data	BOX	Data of the workspace (>>> 6.7 "Data of a Cartesian workspace" Page 28)
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.22 Configuring axis-specific workspaces

Description The function block KRC_WriteAxWorkspace is used to configure axis-specific workspaces for the robot. These serve to protect the system. A maximum of 8 axis-specific workspaces can be configured at any one time. The workspaces may overlap.

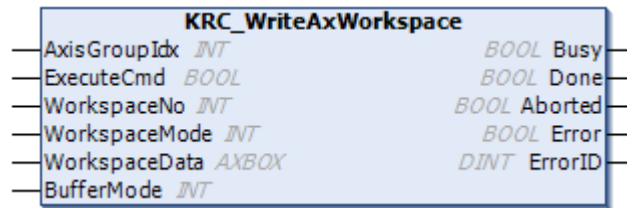


Fig. 6-88: Function block KRC_WriteAxWorkspace

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Work-spaceNo	INT	Number of workspace <ul style="list-style-type: none"> 1 ... 8
Work-space-Mode	INT	Mode for workspaces <ul style="list-style-type: none"> 0: #OFF 1: #INSIDE 2: #OUTSIDE 3: #INSIDE_STOP 4: #OUTSIDE_STOP <p>Note: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).</p>
Work-space-Data	AXBOX	Data of the workspace (>>> 6.8 "Data of an axis-specific workspace" Page 28)
BufferMode	INT	Mode in which the statement is executed <ul style="list-style-type: none"> 0: DIRECT 1: ABORTING 2: BUFFERED (>>> "BufferMode" Page 26)

Outputs

Parameter	Type	Description
ErrorID	DINT	Error number
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Aborted	BOOL	TRUE = statement was aborted before it was processed in the advance run
Error	BOOL	TRUE = error in function block

6.14.23 Reading the configuration of axis-specific workspaces

Description The function block KRC_ReadAxWorkspace reads the configuration of the axis-specific workspaces for the robot.

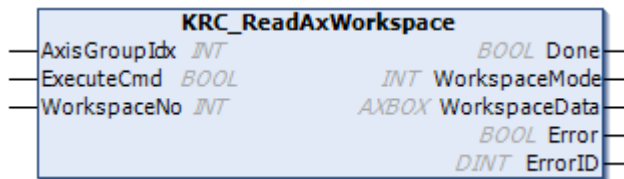


Fig. 6-89: Function block KRC_ReadAxWorkspace

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group <ul style="list-style-type: none"> 1 ... 5
ExecuteCmd	BOOL	Starts/buffers the motion in the case of a rising edge of the signal.
Work-spaceNo	INT	Number of workspace <ul style="list-style-type: none"> 1 ... 8

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
Workspace-Mode	INT	Mode for workspaces <ul style="list-style-type: none"> 0: #OFF 1: #INSIDE 2: #OUTSIDE 3: #INSIDE_STOP 4: #OUTSIDE_STOP <p>Note: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).</p>
Workspace-Data	AXBOX	Data of the workspace (>>> 6.8 "Data of an axis-specific workspace" Page 28)
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

6.14.24 Reading the status of the workspaces

Description The function block KRC_ReadWorkstates reads the current status of the workspaces. The status of the workspaces is updated cyclically.



Fig. 6-90: Function block KRC_ReadWorkstates

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group ■ 1 ... 5

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
WORKSTATE1	BOOL	Status of the workspaces
WORKSTATE2	BOOL	
WORKSTATE3	BOOL	
WORKSTATE4	BOOL	
WORKSTATE5	BOOL	
WORKSTATE6	BOOL	
WORKSTATE7	BOOL	
WORKSTATE8	BOOL	
AXWORKSTATE1	BOOL	
AXWORKSTATE2	BOOL	
AXWORKSTATE3	BOOL	
AXWORKSTATE4	BOOL	
AXWORKSTATE5	BOOL	
AXWORKSTATE6	BOOL	
AXWORKSTATE7	BOOL	
AXWORKSTATE8	BOOL	
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

7 Messages

7.1 Error messages of the mxA interface in the robot interpreter

No.	Message text	Cause	Remedy
0	—	—	—
1	INTERNAL ERROR	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
2	ASSERT FAILED	Internal exceptional error	
3	OVERFLOW STATUS RETURN QUEUE (MAIN)	There are more than 100 checkback signals relating to status changes waiting to be transferred from the robot controller to the PLC. The transmission rate is considerably lower than the processing speed.	Reduce the number of statements to be buffered simultaneously.
4	OVERFLOW STATUS RETURN QUEUE (TRIGGER)		If this is not possible, contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
5	INVALID COMMAND QUEUE INDEX	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
6	INVALID COMMAND STATE	Internal exceptional error	
7	INVALID COMMAND ID	Internal exceptional error	
8	INVALID MOVE TYPE	Internal exceptional error	
9	OVERFLOW TRIGGER FIFO	Internal exceptional error	
10	UNDERFLOW TRIGGER FIFO	Internal exceptional error	
11	INVALID TRIGGER FIFO INDEX	Internal exceptional error	
12	EXECUTION OF T_AFTER MISSING	Internal exceptional error	
13	EXECUTION OF T_START MISSING	Internal exceptional error	
14	INVALID ADVANCE_ACT	Internal exceptional error	
16	TIMEOUT HEARTBEAT FROM PLC	Connection to PLC interrupted:	Restore connection, then acknowledge error:
		PLC program stopped	Restart the PLC program.
		Connecting cable defective or not correctly connected	Exchange connecting cable or connect it correctly.
17	INVALID ORDERID (INVERSE)	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
30	INVALID PTP APO	An invalid approximation parameter has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
31	INVALID CP APO	An invalid approximation parameter has been transferred for a CP motion (LIN, CIRC).	

No.	Message text	Cause	Remedy
32	INVALID BASE NUMBER	An invalid number has been programmed in the function block KRC_ReadBaseData or KRC_WriteBaseData for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter BaseNo). ■ 1 ... 32
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Base). (>>> "COORDSYS" Page 25)
33	INVALID TOOL NUMBER	An invalid number has been programmed in the function block KRC_ReadToolData or KRC_WriteToolData for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter ToolNo). ■ 1 ... 16
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Tool). (>>> "COORDSYS" Page 25)
34	INVALID VELOCITY	An invalid value has been programmed in a function block for the velocity.	Program a valid value (parameter Velocity): ■ 0 ... 100 %
35	INVALID ACCELERATION	An invalid value has been programmed in a function block for the acceleration.	Program a valid value (parameter Acceleration): ■ 0 ... 100 %
36	INVALID C_PTP	An invalid approximation distance has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
37	INVALID C_DIS	An invalid distance parameter has been transferred for an approximated motion.	
38	INVALID C_VEL	An invalid velocity parameter has been transferred for an approximated motion.	
39	INVALID C_ORI	An invalid orientation parameter has been transferred for an approximated motion.	
40	INVALID ORI_TYPE	An invalid value has been programmed in a KRC_Move or KRC_Jog function block for the orientation control of the TCP.	Program a valid value (parameter OriType). (>>> "OriType" Page 27)
41	POSITION DATA NOT INITIALIZED	No end position transferred when calling a KRC_Move function block.	Define at least 1 element of the end position (parameter Position). (>>> "E6POS" Page 25)

No.	Message text	Cause	Remedy
42	AXISPOSITION DATA NOT INITIAL- IZED	No axis position transferred when calling a KRC_MoveAxis function block.	Define at least 1 axis position (parameter AxisPosition). (>>> "E6AXIS" Page 25)
43	INVALID TRIGGER DISTANCE	An invalid value has been programmed in a KRC_SetDistanceTrigger function block for the switching point of the trigger.	Program a valid value (parameter Distance): <ul style="list-style-type: none"> ■ 0: Switching action at the start point ■ 1: Switching action at the end point
44	INVALID TRIGGER IO	An invalid output has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block.	Program a valid value (parameter Output): <ul style="list-style-type: none"> ■ 1 ... 2 048
45	INVALID TRIGGER PULSE	An invalid value has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block for the length of the pulse.	Program a valid value (parameter Pulse): <ul style="list-style-type: none"> ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active)
46	INVALID CIRC_HP	No auxiliary position transferred when calling a KRC_MoveCirc function block.	Define at least 1 element of the auxiliary position (parameter CircHP). (>>> "E6AXIS" Page 25)
47	INVALID INTER- RUPT IO	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): <ul style="list-style-type: none"> ■ 1 ... 2 048
48	INVALID INTER- RUPT PRIORITY	An invalid number was transferred when calling a KRC_...Interrupt function block.	Program a valid value (parameter Interrupt): <ul style="list-style-type: none"> ■ 1 ... 8
49	INTERRUPT NOT DECLARED	Interrupt has not been declared.	Declare interrupt. (>>> 6.10.32 "Declaring interrupts" Page 56)
50	INVALID INTER- RUPT ACTION	The interrupt reaction programmed when the the interrupt was declared is invalid.	Program a valid reaction (parameter Reaction). (>>> 6.10.32 "Declaring interrupts" Page 56)
51	INVALID IO NUM- BER	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): <ul style="list-style-type: none"> ■ 1 ... 2 048
52	INVALID PULSE DURATION	An invalid value has been programmed in the function block KRC_WriteDigitalOutput for the length of the pulse.	Program a valid value (parameter Pulse): <ul style="list-style-type: none"> ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active)
53	INVALID BUFFER_MODE	An invalid BufferMode has been programmed in a function block, e.g. DIRECT mode is not available for certain function blocks.	Program a valid BufferMode . (>>> "BufferMode" Page 26)

No.	Message text	Cause	Remedy
54	INVALID TOOL NUMBER FOR LOAD_DATA	An invalid number has been programmed in the function block KRC_ReadLoadData or KRC_WriteLoadData for reading or writing the load data or supplementary load data.	Program a valid value (parameter Tool). (>>> 6.10.26 "Reading the load data" Page 51) (>>> 6.10.27 "Writing load data" Page 51)
55	INVALID ANALOG IO NUMBER	An invalid value has been programmed in a function block for the analog input or output.	Program a valid value (parameter Number): ■ 1 ... 32
56	INVALID IPO_MODE	An invalid value has been programmed in a function block for the interpolation mode, e.g. in a KRC_Move function block.	Program a valid value (parameter CoordinateSystem – COORDSYS.IPO_MODE). (>>> "COORDSYS" Page 25)
57	INVALID CIRC_TYPE	An invalid value has been programmed in a KRC_MoveCirc function block for the orientation control during the circular motion.	Program a valid value (parameter CircType). (>>> "CircType" Page 27)
58	INVALID FRAME DATA	Invalid TOOL or BASE data have been programmed in a KRC_WriteToolData or KRC_WriteBaseData function block.	Program valid data (parameter ToolData or BaseData). (>>> 6.10.23 "Writing TOOL data" Page 48) (>>> 6.10.25 "Writing BASE data" Page 50)
59	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> 6.10.27 "Writing load data" Page 51)
60	INVALID SOFT_END (REVERSED)	Error writing the software limit switches: positive software limit switch < negative software limit switch (function block KRC_WriteSoftEnd or KRC_WriteSoftEndEx)	Program lower values for the negative software limit switch than for the positive software limit switch.
61	INVALID INTERRUPT STATE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
62	INVALID SYS VAR INDEX	An index for which no system variable is stored has been transferred in a KRC_ReadSysVar or KRC_WriteSysVar function block.	Program a valid value (parameter Index).
63	INVALID SYS VAR VALUE	An invalid value has been programmed in a KRC_WriteSysVar function block for the system variable.	Program a valid value (parameter Value1 ... Value10).
64	SYS VAR NOT WRITEABLE	An error occurred when writing a system variable. The specified system variable does not exist or may not be written in the current operating state.	

No.	Message text	Cause	Remedy
65	INVALID REAL VALUE	The programmed Real value is invalid.	Program a valid value: <ul style="list-style-type: none"> ■ -2,147,483,500 ... ■ +2,147,483,500
66	ERROR SETTING OUTPUT	Error writing a digital output. The output may already be assigned by the system.	Use a different digital output (parameter Number): <ul style="list-style-type: none"> ■ 1 ... 2 048
67	ERROR SETTING SOFTEND	Error writing the software limit switches: One possible error, for example, is writing a rotational axis with a value outside the range +/-360°.	Program valid values for the software limit switches (see machine data).
68	INVALID TECH FUNCTION INDEX	A TechFunctionID for which no technology function is stored has been transferred in a KRC_TechFunction function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
69	INVALID TECH FUNCTION PARAMETER	An invalid value has been programmed in a KRC_TechFunction function block for a parameter.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
70	INVALID PARAMETER VALUE	An invalid value has been programmed in the called function block for one or more parameters.	Program valid values for the parameters.

7.2 Error messages of the mxA interface in the submit interpreter

No.	Message text	Cause	Remedy
401	INTERNAL ERROR	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
402	ASSERT FAILED	Internal exceptional error	
403	INVALID COMMAND ID	Internal exceptional error	
404	INVALID COMMAND STATE	Internal exceptional error	
405	OVERFLOW COMMAND QUEUE	Internal exceptional error	
406	INVALID COMMAND QUEUE INDEX	Internal exceptional error	
407	INVALID COMMAND (PRE) QUEUE INDEX	Internal exceptional error	
408	INVALID WRITE_Q_IDX AND WRITE_PRE_Q_IDX	Internal exceptional error	

No.	Message text	Cause	Remedy
409	OVERFLOW STATUS RETURN QUEUE (SUBMIT)	There are more than 100 checkback signals relating to status changes waiting to be transferred from the robot controller to the PLC. The transmission rate is considerably lower than the processing speed.	Reduce the number of statements to be buffered simultaneously. If this is not possible, contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
410	INVALID FIELD-BUS TELEGRAMM LENGTH	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
411	TIMEOUT ABORT_REQUEST	Internal exceptional error	
412	INVALID CHECKSUM PLC -> KRC	The checksum for data transmission from the PLC to the robot controller is invalid:	
		Error during start-up: <ul style="list-style-type: none">■ PROFINET configuration in WorkVisual or CODESYS faulty	Check configuration in WorkVisual and CODESYS and configure PROFINET correctly.
		Error during operation: <ul style="list-style-type: none">■ Bit error during data transfer	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
413	INVALID MOVE TYPE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
414	TIMEOUT HEARTBEAT FROM PLC	Connection to PLC interrupted:	Restore connection, then acknowledge error:
		PLC program stopped	Restart the PLC program.
		Submit interpreter deselected or stopped	Restart submit interpreter.
		Connecting cable defective or not correctly connected	Exchange connecting cable or connect it correctly.
416	SYS VAR NOT INITIALIZED	An error occurred when reading a system variable. The specified system variable does not exist or may not be read in the current operating state. Example: \$POS_ACT cannot be accessed until a BCO run has been carried out.	

No.	Message text	Cause	Remedy
417	UNDERFLOW OF NIBBLE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
418	OVERFLOW OF NIBBLE	Internal exceptional error	
419	UNDERFLOW OF BYTE	Internal exceptional error	
420	OVERFLOW OF BYTE	Internal exceptional error	
421	UNDERFLOW OF INT16	Internal exceptional error	
422	OVERFLOW OF INT16	Internal exceptional error	
423	UNDERFLOW OF INT32	Internal exceptional error	
424	OVERFLOW OF INT32	Internal exceptional error	
425	UNDERFLOW OF REAL	Internal exceptional error	
426	OVERFLOW OF REAL	Internal exceptional error	
430	INVALID PTP APO	An invalid approximation parameter has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
431	INVALID CP APO	An invalid approximation parameter has been transferred for a CP motion (LIN, CIRC).	
432	INVALID BASE NUMBER	An invalid number has been programmed in the function block KRC_ReadBaseData or KRC_WriteBaseData for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter BaseNo). ■ 1 ... 32
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Base). (>>> "COORDSYS" Page 25)
433	INVALID TOOL NUMBER	An invalid number has been programmed in the function block KRC_ReadToolData or KRC_WriteToolData for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter ToolNo). ■ 1 ... 16
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Tool). (>>> "COORDSYS" Page 25)

No.	Message text	Cause	Remedy
434	INVALID VELOCITY	An invalid value has been programmed in a function block for the velocity.	Program a valid value (parameter Velocity): <ul style="list-style-type: none"> ■ 0 ... 100 %
435	INVALID ACCELERATION	An invalid value has been programmed in a function block for the acceleration.	Program a valid value (parameter Acceleration): <ul style="list-style-type: none"> ■ 0 ... 100 %
436	INVALID C_PTP	An invalid approximation distance has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
437	INVALID C_DIS	An invalid distance parameter has been transferred for an approximated motion.	
438	INVALID C_VEL	An invalid velocity parameter has been transferred for an approximated motion.	
439	INVALID C_ORI	An invalid orientation parameter has been transferred for an approximated motion.	
440	INVALID ORI_TYPE	An invalid value has been programmed in a KRC_Move or KRC_Jog function block for the orientation control of the TCP.	Program a valid value (parameter OriType). (>>> "OriType" Page 27)
441	POSITION DATA NOT INITIALIZED	No end position transferred when calling a KRC_Move function block.	Define at least 1 element of the end position (parameter Position). (>>> "E6POS" Page 25)
442	AXISPOSITION DATA NOT INITIALIZED	No axis position transferred when calling a KRC_MoveAxis function block.	Define at least 1 axis position (parameter AxisPosition). (>>> "E6AXIS" Page 25)
443	INVALID TRIGGER DISTANCE	An invalid value has been programmed in the KRC_SetDistanceTrigger function block for the switching point of the trigger.	Program a valid value (parameter Distance): <ul style="list-style-type: none"> ■ 0: Switching action at the start point ■ 1: Switching action at the end point
444	INVALID TRIGGER IO	An invalid output has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block.	Program a valid value (parameter Output): <ul style="list-style-type: none"> ■ 1 ... 2 048
445	INVALID TRIGGER PULSE	An invalid value has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block for the length of the pulse.	Program a valid value (parameter Pulse): <ul style="list-style-type: none"> ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active)
446	INVALID CIRC_HP	No auxiliary position transferred when calling a KRC_MoveCirc function block.	Define at least 1 element of the auxiliary position (parameter CircHP). (>>> "E6AXIS" Page 25)

No.	Message text	Cause	Remedy
447	INVALID INTERRUPT IO	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): ■ 1 ... 2 048
448	INVALID INTERRUPT NUMBER/PRIORITY	An invalid number was transferred when calling a KRC_...Interrupt function block.	Program a valid value (parameter Interrupt): ■ 1 ... 8
449	INTERRUPT NOT DECLARED	Interrupt has not been declared.	Declare interrupt. (>>> 6.10.32 "Declaring interrupts" Page 56)
450	INVALID INTERRUPT ACTION	The interrupt reaction programmed when the the interrupt was declared is invalid.	Program a valid reaction (parameter Reaction). (>>> 6.10.32 "Declaring interrupts" Page 56)
451	INVALID IO NUMBER	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): ■ 1 ... 2 048
452	INVALID PULSE DURATION	An invalid value has been programmed in the function block KRC_WriteDigitalOutput for the length of the pulse.	Program a valid value (parameter Pulse): ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active) (>>> 6.10.15 "Writing a digital output" Page 42)
453	INVALID BUFFER_MODE	An invalid BufferMode has been programmed in a function block, e.g. DIRECT mode is not available for certain function blocks.	Program a valid BufferMode . (>>> "BufferMode" Page 26)
454	INVALID TOOL NUMBER FOR LOAD_DATA	An invalid number has been programmed in the function block KRC_ReadLoadData or KRC_WriteLoadData for reading or writing the load data or supplementary load data.	Program a valid value (parameter Tool). (>>> 6.10.26 "Reading the load data" Page 51) (>>> 6.10.27 "Writing load data" Page 51)
455	INVALID ANALOG IO NUMBER	An invalid number has been programmed for the analog input or output in a function block.	Program a valid number (parameter Number): ■ 1 ... 32
456	INVALID IPO_MODE	An invalid value has been programmed in a function block for the interpolation mode, e.g. in a KRC_Move function block.	Program a valid value (parameter CoordinateSystem – COORDSYS.IPO_MODE). (>>> "COORDSYS" Page 25)
457	INVALID CIRC_TYPE	An invalid value has been programmed in a KRC_MoveCirc function block for the orientation control during the circular motion.	Program a valid value (parameter CircType). (>>> "CircType" Page 27)

No.	Message text	Cause	Remedy
458	INVALID FRAME DATA	Invalid TOOL or BASE data have been programmed in a KRC_WriteToolData or KRC_WriteBaseData function block.	Program valid data (parameter ToolData or BaseData). (>>> 6.10.23 "Writing TOOL data" Page 48) (>>> 6.10.25 "Writing BASE data" Page 50)
459	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> 6.10.27 "Writing load data" Page 51)
460	INVALID SOFT_END (REVERSED)	Error writing the software limit switches: Positive software limit switch < negative software limit switch	Program lower values for the negative software limit switch than for the positive software limit switch.
461	INVALID INTERRUPT STATE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
462	INVALID SYS VAR INDEX	An index for which no system variable is stored has been transferred in a KRC_ReadSysVar or KRC_WriteSysVar function block.	Program a valid value (parameter Index).
463	INVALID SYS VAR VALUE	An invalid value has been programmed in a KRC_WriteSysVar function block for the system variable.	Program a valid value (parameter Value1 ... Value10).
464	SYS VAR NOT WRITEABLE	An error occurred when writing a system variable. The specified system variable does not exist or may not be written in the current operating state.	
465	INVALID REAL VALUE	The programmed Real value is invalid.	Program a valid value: ■ -2,147,483,500 ... +2,147,483,500
466	ERROR SETTING OUTPUT	Error writing an output. The output may already be assigned by the system.	Use a different digital output (parameter Number): ■ 1 ... 2 048
467	ERROR SETTING SOFTEND	An error occurred when writing a software limit switch. One possible error, for example, is writing a rotational axis with a value outside the range +/-360°.	Program valid values for the software limit switches (see machine data).
468	INVALID TECH FUNCTION INDEX	A TechFunctionID for which no technology function is stored has been transferred in a KRC_TechFunction function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)

No.	Message text	Cause	Remedy
469	INVALID TECH FUNCTION PARAMETER	An invalid value has been programmed in a KRC_TechFunction function block for a parameter.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
470	INVALID PARAMETER VALUE	An invalid value has been programmed in the called function block for one or more parameters.	Program valid values for the parameters.

7.3 Errors in the function block

No.	Message text	Cause	Remedy
501	INTERNAL ERROR	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
502	INVALID BUFFER_MODE	BufferMode 0 : DIRECT is not permissible for this function block.	Program the correct mode: <ul style="list-style-type: none"> ■ 1: ABORTING ■ 2: BUFFERED
503	INVALID MXA VERSION	The software versions of the mxA interface and PLC library are not compatible.	Install compatible software versions on the robot controller and PLC. (>>> 6.10.3 "Initializing the mxA interface" Page 32)
504	INVALID OVERRIDE	Invalid override value in the function block KRC_SetOverride	Program a valid value (parameter Override). <ul style="list-style-type: none"> ■ 0 ... 100 %
505	MAX GROUP REF IDX REACHED	The axis group index specified in the function block KRC_ReadAxisGroup is already assigned.	Only instance the KRC_ReadAxisGroup function block once in a program.
506	INVALID GROUPPREFIX	The axis group index specified in the function block is invalid.	Specify a valid index for the axis group (parameter AxisGroupIdx).
507	INVALID FB ORDER	The order in which the function blocks were called is invalid.	Program the function blocks in the correct order.
508	CONNECTION NOT INITIALIZED	No statements can be transferred, as the mxA interface has not been initialized.	Initialize the mxA interface. (>>> 6.10.3 "Initializing the mxA interface" Page 32)
509 510	NO CONNECTION TO KRC TIMEOUT HEARTBEAT FROM KRC	Connection to robot controller interrupted:	Restore connection, then acknowledge error:
		Robot controller is switched off	Reboot the robot controller.
		Submit interpreter deselected or stopped	Restart submit interpreter.
		Bus error or I/O configuration faulty	Check I/O configuration.
		Connecting cable defective or not correctly connected	Exchange connecting cable or connect it correctly.
		Maximum cycle time of the submit interpreter is too short (only for message no. 510)	Increase the value for MaxSubmitCycle in the function block KRC_DIAG.

No.	Message text	Cause	Remedy
511	TIMEOUT CMD INTERFACE BLOCKED	The ExecuteCmd input was reset before the Busy signal was set.	Acknowledge the message and in future do not reset the ExecuteCmd input until the Done, Error or Aborted signal has been set.
512	INVALID CHECKSUM KRC -> PLC	The checksum for data transmission from the robot controller to the PLC is invalid.	Check configuration in WorkVisual and CODESYS and configure PROFINET correctly.
		Error during start-up: <ul style="list-style-type: none"> ■ PROFINET configuration in WorkVisual or CODESYS faulty 	
		Error during operation: <ul style="list-style-type: none"> ■ Bit error during data transfer 	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
513	INVALID POSITION INDEX	An invalid number for the position to be taught was transferred in the function block KRC_TouchUP.	Program a valid value (parameter Index): <ul style="list-style-type: none"> ■ 1 ... 100
514	POS_ACT INVALID	The current position cannot be taught, as the position data are invalid (no BCO).	Establish BCO with a RESET at the function block KRC_AutomaticExternal.
517	INVALID COMMAND SIZE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
518	KRC STOPMESS ACTIVE	Group error which prevents motion enable	Check how the error was triggered and eliminate the error. <ul style="list-style-type: none"> ■ Analyze the messages in the message window of the KUKA smartHMI. ■ Read the current error state of the robot controller with the function block KRC_ReadKRCError.
519	INVALID ABSOLUTE VELOCITY	An invalid value has been programmed for the parameter AbsoluteVelocity in a KRC_Move function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
520	VELOCITY CONFLICT	More than one value has been programmed for the velocity in a KRC_Move function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
521	INVALID PARAMETER COUNT	An invalid value has been programmed for the parameter ParameterCount in a KRC_TechFunction function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
522	INVALID PARAMETER USAGE	The parameter ParameterCount has been incorrectly configured in the KRC_TechFunction function block.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
523	INVALID OPERATION MODE	The robot is in the incorrect operating mode.	Select Automatic External mode.
524	USER_SAF SIGNAL NOT ACTIVE	The operator safety is violated.	Close the safeguard and acknowledge the closed state.

No.	Message text	Cause	Remedy
525	ALARM_STOP SIGNAL NOT ACTIVE	The safety configuration is incorrect and an EMERGENCY STOP has been triggered.	Check and modify the safety configuration of the system (robot controller and PLC).
		No connection to the EMERGENCY STOP of the system	Check the EMERGENCY STOP of the system and reestablish the connection.
		The inputs and outputs of the Automatic External interface are incorrectly configured.	<ol style="list-style-type: none"> In the main menu, select Display > Inputs/outputs > Automatic External. Check and modify the configuration of the inputs and outputs.
526	APPL_RUN SIGNAL ACTIVE	RESET cannot be carried out because a robot program is running.	<ol style="list-style-type: none"> Wait until the robot program has been executed. Execute the statement again.
527	TIMEOUT MESSAGE CONFIRM	The message cannot be acknowledged by the PLC.	Acknowledge the message on the robot controller.
528	TIMEOUT MXA MESSAGE CONFIRM	An error cannot be acknowledged in the function block KRC_AutoStart.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
529	TIMEOUT SWITCHING DRIVES ON	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
530	TIMEOUT PROGRAM SELECTION	Internal exceptional error	
531	TIMEOUT PROGRAM START	Internal exceptional error	
532	MOVE_ENABLE SIGNAL NOT ACTIVE	The robot does not have motion enable	Issue motion enable with the parameter MOVE_ENABLE .
533	INVALID AXIS_VALUES	In the function block KRC_Forward, not all axis angles required for execution are defined.	Define the missing axis angles in the function block KRC_Forward.
534	INVALID \$BASE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
535	INVALID \$TOOL	Internal exceptional error	
536	INVALID SOFTEND	<p>Error in the function block KRC_Forward:</p> <p>The specified axis angles lie outside of the software limit switches.</p>	<p>Enter axis angles that lie within the software limit switches (parameter Axis_Values).</p> <p>or: Modify the software limit switches.</p>
537	ERR MATH TRAFO	<p>Error in the function block KRC_Forward:</p> <p>The robot cannot reach the specified axis angles.</p>	Enter axis angles that the robot can reach (parameter Axis_Values).

No.	Message text	Cause	Remedy
538	INVALID AXIS_VALUES	Error in the function block KRC_Inverse: <ul style="list-style-type: none"> ■ The Cartesian robot position has not been fully specified. ■ The axis-specific values at the start point of the motion have not been fully specified. 	<ul style="list-style-type: none"> ■ Fully specify the Cartesian robot position (parameter Position). ■ Fully specify the axis-specific values at the start point of the motion (parameter Start_Axis).
539	INVALID \$BASE	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
540	INVALID \$TOOL	Internal exceptional error	
541	INVALID SOFTEND	Error in the function block KRC_Inverse: The specified axis-specific values at the start point of the motion lie outside of the software limit switches.	Enter values that lie within the software limit switches (parameter Start_Axis). or: Modify the software limit switches.
542	ERR MATH TRAFO	Error in the function block KRC_Inverse: The robot cannot reach the specified axis-specific values at the start point of the motion.	Enter values that the robot can reach (parameter Start_Axis).
543	INVALID EXECUTE	During a linked motion capable of being approximated, the Execute input was reset before the ComDone signal was set by the function block.	Acknowledge the message and in the future do not reset the Execute input until the ComDone signal has been set by the function block.
544	INVALID DEV_VEL_CP	Initialization of the mxA interface on the robot controller has not yet been completed or has an error.	Check whether the Done output on the function block KRC_Initialize is active.

7.4 ProConOS errors

No.	Message text	Cause	Remedy
701	INTERNAL ERROR	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
702	ASSERT FAILED	Internal exceptional error	
703	INVALID COMMAND ID	Internal exceptional error	
704	INVALID HEADER DATA	Internal exceptional error	
709	ERROR READING SOFTPLC	Internal exceptional error	
710	ERROR FROM KRC SUBMIT	Internal exceptional error	
712	INVALID CHECKSUM PLC -> KRC	The checksum for data transmission from the PLC to the robot controller is invalid.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)

No.	Message text	Cause	Remedy
713	INVALID MOVE TYPE	An invalid value has been programmed in a KRC_Move function block for the parameter MoveType.	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
730	INVALID PTP APO	An invalid approximation parameter has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
731	INVALID CP APO	An invalid approximation parameter has been transferred for a CP motion (LIN, CIRC).	
732	INVALID BASE NUMBER	An invalid number has been programmed in the function block KRC_ReadBaseData or KRC_WriteBaseData for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter BaseNo). ■ 1 ... 32
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the BASE coordinate system.	Specify the number of the BASE coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Base). (>>> "COORDSYS" Page 25)
733	INVALID TOOL NUMBER	An invalid number has been programmed in the function block KRC_ReadToolData or KRC_WriteToolData for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter ToolNo). ■ 1 ... 16
		An invalid number has been programmed in a KRC_Move or KRC_Jog function block for the TOOL coordinate system.	Specify the number of the TOOL coordinate system that is currently being used in the robot controller (parameter CoordinateSystem – COORDSYS.Tool). (>>> "COORDSYS" Page 25)
734	INVALID VELOCITY	An invalid value has been programmed in a function block for the velocity.	Program a valid value (parameter Velocity): ■ 0 ... 100 %
735	INVALID ACCELERATION	An invalid value has been programmed in a function block for the acceleration.	Program a valid value (parameter Acceleration): ■ 0 ... 100 %

No.	Message text	Cause	Remedy
736	INVALID C_PTP	An invalid approximation distance has been transferred for a PTP motion.	Program a valid value (parameter Approximate). (>>> "APO" Page 24)
737	INVALID C_DIS	An invalid distance parameter has been transferred for an approximated motion.	
738	INVALID C_VEL	An invalid velocity parameter has been transferred for an approximated motion.	
739	INVALID C_ORI	An invalid orientation parameter has been transferred for an approximated motion.	
740	INVALID ORI_TYPE	An invalid value has been programmed in a KRC_Move or KRC_Jog function block for the orientation control of the TCP.	Program a valid value (parameter OriType). (>>> "OriType" Page 27)
741	POSITION DATA NOT INITIALIZED	No end position transferred when calling a KRC_Move function block.	Define at least 1 element of the end position (parameter Position). (>>> "E6POS" Page 25)
742	AXISPOSITION DATA NOT INITIALIZED	No axis position transferred when calling a KRC_MoveAxis function block.	Define at least 1 axis position (parameter AxisPosition). (>>> "E6AXIS" Page 25)
743	INVALID TRIGGER DISTANCE	An invalid value has been programmed in a KRC_SetDistanceTrigger function block for the switching point of the trigger.	Program a valid value (parameter Distance): <ul style="list-style-type: none"> ■ 0: Switching action at the start point ■ 1: Switching action at the end point
744	INVALID TRIGGER IO	An invalid output has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block.	Program a valid value (parameter Output): <ul style="list-style-type: none"> ■ 1 ... 2 048
745	INVALID TRIGGER PULSE	An invalid value has been programmed in a KRC_SetDistanceTrigger or KRC_SetPathTrigger function block for the length of the pulse.	Program a valid value (parameter Pulse): <ul style="list-style-type: none"> ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active)
746	INVALID CIRC_HP	No auxiliary position transferred when calling a KRC_MoveCirc function block.	Define at least 1 element of the auxiliary position (parameter CircHP). (>>> "E6AXIS" Page 25)
747	INVALID INTERRUPT IO	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): <ul style="list-style-type: none"> ■ 1 ... 2 048
748	INVALID INTERRUPT PRIORITY	An invalid number was transferred when calling a KRC_...Interrupt function block.	Program a valid value (parameter Interrupt): <ul style="list-style-type: none"> ■ 1 ... 8

No.	Message text	Cause	Remedy
750	INVALID INTERRUPT ACTION	The interrupt reaction programmed when the interrupt was declared is invalid.	Program a valid reaction (parameter Reaction). (>>> 6.10.32 "Declaring interrupts" Page 56)
751	INVALID IO NUMBER	The number of the digital input to which the interrupt is declared is invalid (function block KRC_DeclareInterrupt).	Program a valid value (parameter Input): ■ 1 ... 2 048
752	INVALID PULSE DURATION	An invalid value has been programmed in the function block KRC_WriteDigitalOutput for the length of the pulse.	Program a valid value (parameter Pulse): ■ 0.1 ... 3.0 s ■ 0.0 s (No pulse active)
753	INVALID BUFFER_MODE	An invalid BufferMode has been programmed in a function block, e.g. DIRECT mode is not available for certain function blocks.	Program a valid BufferMode . (>>> "BufferMode" Page 26)
754	INVALID TOOL NUMBER FOR LOAD_DATA	An invalid number has been programmed in the function block KRC_ReadLoadData or KRC_WriteLoadData for reading or writing the load data or supplementary load data.	Program a valid value (parameter Tool). (>>> 6.10.26 "Reading the load data" Page 51) (>>> 6.10.27 "Writing load data" Page 51)
755	INVALID ANALOG IO NUMBER	An invalid value has been programmed in a function block for the analog input or output.	Program a valid value (parameter Number): ■ 1 ... 32
756	INVALID IPO_MODE	An invalid value has been programmed in a function block for the interpolation mode, e.g. in a KRC_Move function block.	Program a valid value (parameter CoordinateSystem – COORDSYS.IPO_MODE). (>>> "COORDSYS" Page 25)
757	INVALID CIRC_TYPE	An invalid value has been programmed in a KRC_MoveCirc function block for the orientation control during the circular motion.	Program a valid value (parameter CircType). (>>> "CircType" Page 27)
758	INVALID FRAME DATA	Invalid TOOL or BASE data have been programmed in a KRC_WriteToolData or KRC_WriteBaseData function block.	Program valid data (parameter ToolData or BaseData). (>>> 6.10.23 "Writing TOOL data" Page 48) (>>> 6.10.25 "Writing BASE data" Page 50)
759	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> 6.10.27 "Writing load data" Page 51)
760	INVALID SOFT_END (REVERSED)	Error writing the software limit switches: positive software limit switch < negative software limit switch (function block KRC_WriteSoftEnd or KRC_WriteSoftEndEx)	Program lower values for the negative software limit switch than for the positive software limit switch.

No.	Message text	Cause	Remedy
765	INVALID REAL VALUE	The programmed Real value is invalid.	<ul style="list-style-type: none"> ■ -2,147,483,500 ... +2,147,483,500
770	INVALID PARAMETER VALUE	An invalid value has been programmed in the called function block for one or more parameters.	Program valid values for the parameters.
771	INVALID ADVANCE COUNT	In the function block KRC_SetAdvance, an invalid value has been programmed for the number of functions which are to be transferred prior to the first robot motion.	Program a valid value (parameter Count): <ul style="list-style-type: none"> ■ 1 ... 50
772	INVALID MAX-WAITTIME	In the function block KRC_SetAdvance, an invalid value has been programmed for the maximum wait time before the start of program execution if the set number of functions in the parameter Count is not reached.	Program a valid value (parameter MaxWaitTime): <ul style="list-style-type: none"> ■ 1 ... 60 000 ms
773	INVALID ADVANCE MODE	In the function block KRC_SetAdvance, an invalid value has been programmed for Wait mode.	Program a valid value (parameter Mode): <ul style="list-style-type: none"> ■ 0 ... 2
774	INVALID DI START-NUMBER	In the function block KRC_ReadDigitalInputArray, an invalid value has been programmed for the number of the first digital input that is called.	Program a valid value (parameter Startnumber): <ul style="list-style-type: none"> ■ 1 ... 2 048
775	INVALID DI LENGTH	In the function block KRC_ReadDigitalInputArray, an invalid value has been programmed for the number of inputs that are polled.	Program a valid value (parameter Length): <ul style="list-style-type: none"> ■ 1 ... 2 00
776	INVALID CONVEYOR NUMBER	In the called function block, an invalid number has been programmed for the number of the conveyor.	Program a valid value (parameter ConveyorNumber): <ul style="list-style-type: none"> ■ 1 ... 3
777	INVALID CONVEYOR STARTDISTANCE	In the function block KRC_ConvFollow or KRC_ConvSkip, an invalid value has been programmed for the distance traveled by the workpiece while the robot waits before starting to track the workpiece on the conveyor.	Program a valid value (parameter StartDistance): <ul style="list-style-type: none"> ■ In the case of a linear conveyor: Specification in millimeters ■ In the case of a circular conveyor: Specification in degrees
778	INVALID CONVEYOR MAXDISTANCE	In the function block KRC_ConvFollow or KRC_ConvSkip, an invalid value has been programmed for the maximum distance traveled by the workpiece before the robot starts to synchronize itself with the workpiece.	Program a valid value (parameter MaxDistance): <ul style="list-style-type: none"> ■ In the case of a linear conveyor: Specification in millimeters ■ In the case of a circular conveyor: Specification in degrees

No.	Message text	Cause	Remedy
779	INVALID CONVEYOR PIECE-NUMBER	In the function block KRC_ConvSkip, an invalid value has been programmed for the number specifying which workpieces are to be picked up.	Program a valid value (parameter PieceNumber). Examples: <ul style="list-style-type: none"> ■ 1: Every workpiece is picked up. ■ 3: Every 3rd workpiece is picked up.
780	INVALID WORK-SPACENO	In the called function block, an invalid number has been programmed for the number of the workspace.	Program a valid value (parameter WorkspaceNo): <ul style="list-style-type: none"> ■ 1 ... 8
781	INVALID WORK-SPACEMODE	In the called function block, an invalid number has been programmed for the mode for workspaces.	Program a valid value (parameter WorkspaceMode): <ul style="list-style-type: none"> ■ 0 ... 4
782	INVALID WORK-SPACEPART	Internal exceptional error	Contact KUKA Roboter GmbH. (>>> 8 "KUKA Service" Page 147)
801	STOPMESS ACTIVE	Group error which prevents motion enable	Check how the error was triggered and eliminate the error. <ul style="list-style-type: none"> ■ Analyze the messages in the message window of the KUKA smartHMI. ■ Read the current error state of the robot controller with the function block KRC_ReadKRCErrors.

8 KUKA Service

8.1 Requesting support

Introduction	This documentation provides information on operation and operator control, and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.
Information	<p>The following information is required for processing a support request:</p> <ul style="list-style-type: none"> ■ Description of the problem, including information about the duration and frequency of the fault ■ As comprehensive information as possible about the hardware and software components of the overall system <p>The following list gives an indication of the information which is relevant in many cases:</p> <ul style="list-style-type: none"> ■ Model and serial number of the kinematic system, e.g. the manipulator ■ Model and serial number of the controller ■ Model and serial number of the energy supply system ■ Designation and version of the system software ■ Designations and versions of other software components or modifications ■ Diagnostic package KrcDiag: Additionally for KUKA Sunrise: Existing projects including applications For versions of KUKA System Software older than V8: Archive of the software (KrcDiag is not yet available here.) ■ Application used ■ External axes used

8.2 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	<p>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</p>
Australia	<p>KUKA Robotics Australia Pty Ltd 45 Fennell Street Port Melbourne VIC 3207 Australia Tel. +61 3 9939 9656 info@kuka-robotics.com.au www.kuka-robotics.com.au</p>

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. Travessa Claudio Armando, nº 171 Bloco 5 - Galpões 51/52 Bairro Assunção CEP 09861-7630 São Bernardo do Campo - SP Brazil Tel. +55 11 4942-8299 Fax +55 11 2201-7883 info@kuka-roboter.com.br www.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 Robotics China Co., Ltd. No. 889 Kungang Road Xiaokunshan Town Songjiang District 201614 Shanghai P. R. China Tel. +86 21 5707 2688 Fax +86 21 5707 2603 info@kuka-robotics.cn www.kuka-robotics.com
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 India Pvt. Ltd.
Office Number-7, German Centre,
Level 12, Building No. - 9B
DLF Cyber City Phase III
122 002 Gurgaon
Haryana
India
Tel. +91 124 4635774
Fax +91 124 4635773
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.
YBP Technical Center
134 Godo-cho, Hodogaya-ku
Yokohama, Kanagawa
240 0005
Japan
Tel. +81 45 744 7691
Fax +81 45 744 7696
info@kuka.co.jp

Canada KUKA Robotics Canada Ltd.
6710 Maritz Drive - Unit 4
Mississauga
L5W 0A1
Ontario
Canada
Tel. +1 905 670-8600
Fax +1 905 670-8604
info@kukarobotics.com
www.kuka-robotics.com/canada

Korea	KUKA Robotics Korea Co. Ltd. RIT Center 306, Gyeonggi Technopark 1271-11 Sa 3-dong, Sangnok-gu Ansan City, Gyeonggi Do 426-901 Korea Tel. +82 31 501-1451 Fax +82 31 501-1461 info@kukakorea.com
Malaysia	KUKA Robot Automation (M) Sdn Bhd South East Asia Regional Office No. 7, Jalan TPP 6/6 Taman Perindustrian Puchong 47100 Puchong Selangor Malaysia Tel. +60 (03) 8063-1792 Fax +60 (03) 8060-7386 info@kuka.com.my
Mexico	KUKA de México S. de R.L. de C.V. Progreso #8 Col. Centro Industrial Puente de Vigas Tlalnepantla de Baz 54020 Estado de México Mexico Tel. +52 55 5203-8407 Fax +52 55 5203-8148 info@kuka.com.mx www.kuka-robotics.com/mexico
Norway	KUKA Sveiseanlegg + Roboter Sentrumsvegen 5 2867 Hov Norway Tel. +47 61 18 91 30 Fax +47 61 18 62 00 info@kuka.no
Austria	KUKA Roboter CEE GmbH Gruberstraße 2-4 4020 Linz Austria Tel. +43 7 32 78 47 52 Fax +43 7 32 79 38 80 office@kuka-roboter.at www.kuka.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 Robots IBÉRICA, S.A.
Rua do Alto da Guerra n° 50
Armazém 04
2910 011 Setúbal
Portugal
Tel. +351 265 729 780
Fax +351 265 729 782
info.portugal@kukapt.com
www.kuka.com

Russia KUKA Robotics RUS
Werbnaja ul. 8A
107143 Moskau
Russia
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
info@kuka-robotics.ru
www.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
Industriestr. 9
5432 Neuenhof
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@kukarob.es www.kuka.es
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. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 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.
Fö út 140
2335 Taksony
Hungary
Tel. +36 24 501609
Fax +36 24 477031
info@kuka-robotics.hu

USA KUKA Robotics Corporation
51870 Shelby Parkway
Shelby Township
48315-1787
Michigan
USA
Tel. +1 866 873-5852
Fax +1 866 329-5852
info@kukarobotics.com
www.kukarobotics.com

UK KUKA Robotics UK Ltd
Great Western Street
Wednesbury West Midlands
WS10 7LL
UK
Tel. +44 121 505 9970
Fax +44 121 505 6589
service@kuka-robotics.co.uk
www.kuka-robotics.co.uk

Index

Symbols

\$ACC_CAR_ACT 39
 \$ACT_BASE 37
 \$ACT_TOOL 37
 \$AXIS_ACT 37
 \$IPO_MODE_C 37
 \$POS_ACT 36, 37
 \$VEL_ACT 38
 \$VEL_AXIS_ACT 38

A

Aborted, signal output 21
 Active, signal output 20
 Active, signal output (PLC OPEN) 22
 Advance run, modifying settings 108
 APO (STRUCT) 24
 Approximate positioning 30
 Approximate positioning, CP motion 24
 Approximate positioning, PTP motion 24
 Automatic External, signals 34
 Axis angles, calculating 111
 Axis group 8
 Axis position, reading 37
 Axis velocity, reading 38
 AxisGroupIdx, signal input 20
 AxisGroupIdx, signal input (PLC OPEN) 22

B

BASE data, reading 49
 BASE data, writing 50
 Base, selecting 46
 BCO 8
 Brake test, calling 103
 BufferMode (variable) 26
 Busy, signal output 20
 Busy, signal output (PLC OPEN) 22

C

Cartesian robot position, calculating 110
 CIRC motion 73, 90
 CIRC_REL motion 75, 93
 CircType (variable) 27
 Circular motion 73, 75, 90, 93
 Circular motion, orientation control 27
 CODESYS V3.5 SP4 8
 ComBusy, signal output (PLC OPEN) 22
 ComDone, signal output (PLC OPEN) 22
 CommandAborted, signal output (PLC OPEN) 23
 Communication 9
 Components 9
 Configuration 15
 Conveyor, activating 113
 Conveyor, initializing 112
 COORDSYS (STRUCT) 25
 CP motion, approximate positioning 24
 CP_APO (INT) 24

D

Diagnostic signals, reading 97
 Documentation, industrial robot 7
 Done, signal output 21

E

E6AXIS (STRUCT) 25
 E6POS (STRUCT) 25
 Error states, acknowledging 99
 Error states, reading 99
 Error, signal output 21
 Error, signal output (PLC OPEN) 22
 ErrorID, signal output 21
 ErrorID, signal output (PLC OPEN) 23
 Errors, reading 96
 Errors, resetting 96
 Example, programming 30
 Execute, signal input (PLC OPEN) 22
 ExecuteCmd, signal input 20, 30

F

FIFO 8
 FRAME (STRUCT) 26
 Function blocks, overview 17
 Function blocks, starting automatically 65

H

Hardware 13

I

I/Os 13
 Input signals 20
 Input signals (PLC OPEN) 22
 Inputs 13
 Inputs 1 to 8, digital 40
 Inputs, analog 44
 Inputs, digital 40
 Installation 13
 Instancing 29
 Intended use 9
 Interpolation mode, selecting 46
 Interrupts for monitoring, activating 117
 Interrupts for monitoring, deactivating 118
 Interrupts, activating 57
 Interrupts, deactivating 58
 Interrupts, declaring 56
 Introduction 7

J

Jogging 78, 79, 80

K

Knowledge, required 7
 KR C 8
 KRC_Abort 63
 KRC_ActivateConvInterrupt 117
 KRC_ActivateInterrupt 57
 KRC_AutomaticExternal 34

- KRC_AutoStart 65
 - KRC_BrakeTest 103
 - KRC_Continue 64
 - KRC_ConvFollow 114
 - KRC_ConvIniOff 112
 - KRC_ConvOn 113
 - KRC_ConvSkip 115
 - KRC_DeactivateConvInterrupt 118
 - KRC_DeactivateInterrupt 58
 - KRC_DeclareInterrupt 56
 - KRC_Diag 97
 - KRC_Error 99
 - KRC_Forward 110
 - KRC_GetAdvance 109
 - KRC_Initialize 32
 - KRC_Interrupt 64
 - KRC_Inverse 111
 - KRC_Jog 80
 - KRC_JogLinearRelative 78
 - KRC_JogToolRelative 79
 - KRC_MasRef 104
 - KRC_MessageReset 96
 - KRC_MoveAxisAbsolute 72
 - KRC_MoveCircAbsolute 73
 - KRC_MoveCircRelative 75
 - KRC_MoveDirectAbsolute 69
 - KRC_MoveDirectRelative 70
 - KRC_MoveLinearAbsolute 66
 - KRC_MoveLinearRelative 67
 - KRC_ReadActualAcceleration 39
 - KRC_ReadActualAxisPosition 37
 - KRC_ReadActualAxisVelocity 38
 - KRC_ReadActualPosition 36
 - KRC_ReadActualVelocity 38
 - KRC_ReadAnalogInput 44
 - KRC_ReadAnalogOutput 44
 - KRC_ReadAxisGroup 31
 - KRC_ReadAxWorkspace 124
 - KRC_ReadBaseData 49
 - KRC_ReadDigitalInput 40
 - KRC_ReadDigitalInput1To8 40
 - KRC_ReadDigitalInputArray 41
 - KRC_ReadDigitalOutput 42
 - KRC_ReadInterruptState 59
 - KRC_ReadKRCError 96
 - KRC_ReadLoadData 51
 - KRC_ReadMXAError 96
 - KRC_ReadMXAStatus 95
 - KRC_ReadSafeOPStatus 106
 - KRC_ReadSoftEnd 53
 - KRC_ReadSoftEndExt 53
 - KRC_ReadSysVar 101
 - KRC_ReadToolData 47
 - KRC_ReadTouchUPState 107
 - KRC_ReadWorkspace 122
 - KRC_ReadWorkstates 124
 - KRC_SetAdvance 108
 - KRC_SetCoordSys 46
 - KRC_SetDistanceTrigger 60
 - KRC_SetOverride 33
 - KRC_SetPathTrigger 61
 - KRC_TouchUP 107
 - KRC_VectorMoveOff 120
 - KRC_VectorMoveOn 119
 - KRC_VectorMoveOn, deactivating 120
 - KRC_WaitForInput 46
 - KRC_WriteAnalogOutput 45
 - KRC_WriteAxisGroup 32
 - KRC_WriteAxWorkspace 123
 - KRC_WriteBaseData 50
 - KRC_WriteDigitalOutput 42
 - KRC_WriteDigitalOutput1To8 43
 - KRC_WriteLoadData 51
 - KRC_WriteSoftEnd 54
 - KRC_WriteSoftEndExt 55
 - KRC_WriteSysVar 101
 - KRC_WriteToolData 48
 - KRC_WriteWorkspace 121
 - KRL 8
 - KRL resources 13
 - KUKA Customer Support 147
 - KUKA smartHMI 8
 - KUKA smartPAD 8
- L**
- LIN motion 66, 83
 - LIN_REL motion 67, 84
 - Linear motion 66, 67, 78, 79, 80, 83, 84
 - Load data, reading 51
 - Load data, writing 51
- M**
- Mastering test, calling 104
 - MC_MoveAxisAbsolute 89
 - MC_MoveCircularAbsolute 90
 - MC_MoveCircularRelative 93
 - MC_MoveDirectAbsolute 86
 - MC_MoveDirectRelative 87
 - MC_MoveLinearAbsolute 83
 - MC_MoveLinearRelative 84
 - Messages 127
 - Motion along vector, activating 119
 - Multi-instance call 29
 - Multiple inputs, digital 41
 - mxA interface 8
 - mxA interface, error messages 96
 - mxA interface, initializing 32
- O**
- Orientation control, circular motion 27
 - Orientation control, TCP 27
 - OriType (variable) 27
 - Output signals 20
 - Output signals (PLC OPEN) 22
 - Outputs 13
 - Outputs 1 to 8, digital 43
 - Outputs, analog 44, 45
 - Outputs, digital 42
 - Overview 9
 - Overview, function blocks 17

P

Path velocity, reading 38
Path-related switching action 60, 61
PLC 8
Point-to-point motion 69, 70, 72, 80, 86, 87, 89
Points, teaching 107
Product description 9
PROFINET 8
Program override 30
Program override, setting 33
Program, canceling 63
Program, continuing 64
Programming 17
Programming tips 29
Programming, example 30
Programming, instructions 17
PTP motion 69, 72, 86, 89
PTP motion, approximate positioning 24
PTP_APO (INT) 24
PTP_REL motion 70, 87

Q

QueueMode (variable) 27

R

Robot acceleration, reading 39
Robot interpreter 8
Robot position, reading 36
Robot, stopping 64

S

Safety 11
Safety controller, reading signals 106
Safety instructions 7
Service, KUKA Roboter GmbH 147
Signal sequence, Execute PLC (PLC OPEN) 23
Signal sequence, ExecuteCmd 21
Signals, frequently used 20
Signals, frequently used (PLC OPEN) 22
smarHMI 8
smartPAD 8
Software 13
Software limit switches, reading 53
Software limit switches, writing 54, 55
State of an interrupt, reading 59
State of mxA interface, reading 95
Status, mxA interface 27
Structures (STRUCT) 23
Submit interpreter 8
Support request 147
System requirements 13
System variables, reading 101
System variables, writing 101

T

Target group 7
TCP, orientation control 27
Template, program 9, 17, 29, 30
Terms used 8
Terms, used 8
TOOL data, reading 47

TOOL data, writing 48
Tool, selecting 46
TouchUp status keys 107
Training 7
TRIGGER 60, 61

U

Use, intended 9

V

Values, reading 109

W

Wait statement 46
Warnings 7
Workpiece, picking up 115
Workpiece, tracking 114
Workspaces, configuring (axes) 123
Workspaces, configuring (Cartesian) 121
Workspaces, reading configuration (axes) 124
Workspaces, reading configuration (Cartesian) 122
Workspaces, reading status 124
WorkVisual 8, 9

