Manual

TC3 Robotics mxAutomation

TwinCAT 3

Version: 1.5
Date: 2018-11-26
Order No.: TF5120

BECKHOFF

Table of Contents

1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

Safety Instruction

⚠ DANGER

Familiarize yourself also when reading other documentations with the safety signs used therein and with the meaning of these safety signs.
Carefully attend to safety signs and notes on safety also within other documentations.

3 Introduction

3.1 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.

3.2 Industrial robot documentation

To be taken from the KUKA Documentation.

3.3 Terms used

Term	Description
Axis group	According to the machine data configuration an axis group contains the following axes: • The robot axes `A1` … `A6`, • the additional axes `E1` … `E6` (synchronous or asynchronous).
FIFO	Method used to process a data memory • **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	Option package KUKA.PLC mxAutomation on the robot control unit
NULLFRAME	Cartesian coordinate system in which all coordinates have the value zero
EtherCAT	EtherCAT 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
Submit interpreter	The Submit interpreter is a cyclical logic program that runs in parallel with the motion program on the robot controller.
TwinCAT 3	Runtime and Development Environment for Beckhoff Control Units
WorkVisual	Engineering environment for KR C4-controlled robot cells

4 Product description

4.1 Overview

The TwinCAT library contains function blocks for programming automation tasks with the TwinCAT 3 software.

Components

The following software components are included in the TwinCAT library:

- Function blocks for TwinCAT 3

Communication

For data exchange between the PLC and the robot controller the terminal EL6695-1001 by KUKA is provided.

WorkVisual

The following software is required for configuring the field buses and mapping the field bus signals:

- WorkVisual

4.2 Intended use

Use

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

Library Tc3_mxAutomation	Library Tc3_mxAutomationV3_0
• KUKA System Software 8.3	• KUKA System Software 8.5
• KUKA.PLC ProConOS 4.1	• KUKA.PLC ProConOS 5.0
• KUKA.EtherCAT Bridge	• KUKA.EtherCAT Bridge
• KUKA.ConveyorTech 6.0	• KUKA.ConveyorTech 7.0
• KUKA.VectorMove 1.0	• KUKA.VectorMove 2.0

The offline part of KUKA.PLC mxAutomation is intended for usage with TwinCAT 3.1.4020.14 or higher.

Using it for any other or additional purpose 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

Any use or application deviating from the intended use is deemed to be misuse and is not allowed. This includes e.g.:

- 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 TwinCAT 3.1.4020.14 or higher.

Also see about this

System requirements [▶ 12]

5 Safety

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

⚠ DANGER

Familiarize yourself also when reading other documentations with the safety signs used therein and with the meaning of these safety signs.
Carefully attend to safety signs and notes on safety also within other documentations.

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.

⚠ DANGER

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.

6 Installation

6.1 System requirements

Hardware

Robot controller:

- KR C4
- Or KR C4 compact

External PLC:

- Beckhoff TwinCAT 3 Control System

Software

Robot controller:

- KUKA System Software 8.3 or KUKA System Software 8.5

Table 1: The following KRL resources must be free:

KRL resource	Number
I/Os	2049 … 4080

Library Tc3_mxAutomation for KSS 8.3	Library Tc3_mxAutomationV3_0 for KSS 8.5
KUKA.PLC ProConOS 4-1 4.1 Option with ConveyorTech: • KUKA.ConveyorTech 6.0 Option with VectorMove: • KUKA.VectorMove 1.0 Standard-Laptop/-PC • WorkVisual 4.0	KUKA.PLC ProConOS 4-1 5.0 Option with ConveyorTech: • KUKA.ConveyorTech 7.0 Option with VectorMove: • KUKA.VectorMove 2.0 Standard-Laptop/-PC • WorkVisual 5.0

Industrial PC/ Embedded-PC:

- TwinCAT 3.1.4020.14 or higher

7 Programming

7.1 Frequently used input/output signals in the KRC function blocks

7.1.1 Input signals

AxisGroupIdx

This signal input is used to select the robot system. Up to 5 robot systems can be used.

A robot system is a grouping of axes to form an axis group.

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.

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.
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.

7.1.2 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.

The `Active` output cannot be used for approximate positioning because the next motion command is sent not until the previous command is executed. Approximate positioning is only possible if the `Busy` output of the previous function block is connected with the `ExecuteCmd` input of the succeeding function 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. 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: (>>> Messages [▶ 110])

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.

7.1.3 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.

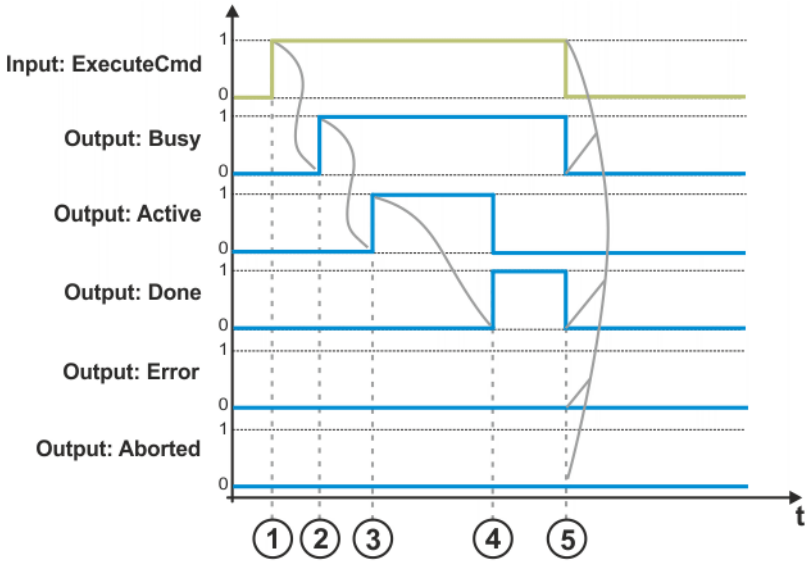


Fig. 1: Signal diagram – ExecuteCmd successful

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 funcion block will not be transferred to the robot in this case and the statement will therefore not be executed.

7.2 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 ComAcpt signal output must be used for linking function blocks.

7.2.1 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.

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.

Value	Name	Description
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.

7.2.2 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.

ComAcpt

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.

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: (>>> Messages [▶ 110])

CommandAborted

This signal output indicates that execution of a statement or motion has been aborted.

7.2.3 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.

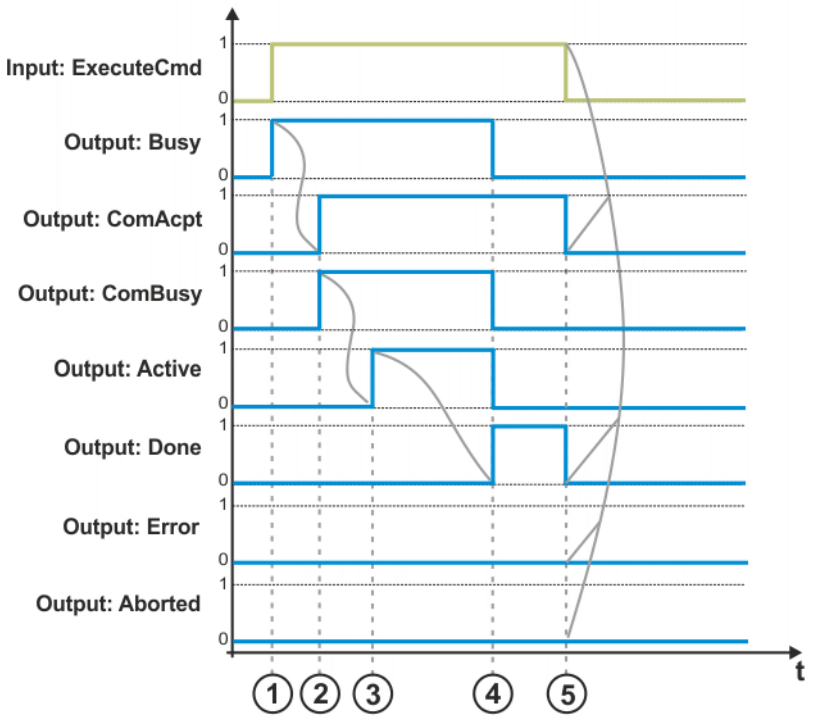


Fig. 2: Signal diagram – Execute successful

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 outputs ComAcpt and ComBusy 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.

7.3 Structures for motion programming (STRUCT)

Predefined data structures (STRUCT) can be used in the PLC library.

Overview

Structure	Description
APO	Approximation parameters for a Move motion command (>>> APO [▶ 18])
COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer in a Move or Jog motion command (>>> COORDSYS [▶ 19])
E6AXIS	Angular values or translation values of the axes in an axis group for a MoveAxis motion command (>>> E6AXIS [▶ 19])
E6POS	Cartesian coordinates of the end position for a Move or Jog motion command (>>> E6POS [▶ 19])

Structure	Description
FRAME	Space coordinates and orientation for the TOOL or BASE coordinate system (>>> FRAME [▶ 20])
POSITION	Contains all positions and the coordinate system to which a point refers. (>>> POSITION [▶ 20])
POSITION_ARRAY	Array of 100 positions of structure type POSITION (>>> POSITION_ ARRAY)

APO

Element	Type	Description
PTP_MODE	INT	Specifies whether and how the end point of a PTP motion is approximated. • **0**: Without approximate positioning (default) • **1**: 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**: PTP-CP approximation with distance parameter • **3**: PTP-CP approximation with orientation parameter • **4**: PTP-CP approximation with velocity parameter
CP_MODE	INT	Specifies whether and how the end point of a CP motion (LIN, CIRC) is approximated. • **0**: Without approximate positioning (default) • **1**: Approximate positioning with distance parameter • **2**: Approximate positioning with orientation parameter • **3**: Approximate positioning with velocity parameter
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.

COORDSYS

Element	Type	Description
Tool	INT	Number of the TOOL coordinate system • **-1**: coordinate system is not changed • **0**: NULLFRAME • **1 … 16**: TOOL_DATA[1 … 16] Default: **-1**
Base	INT	Number of the BASE coordinate system • **-1**: coordinate system is not changed • **0**: NULLFRAME • **1 … 32**: BASE_DATA[1 … 32] Default: **-1**
IPO_MODE	INT	Interpolation mode • **0**: The tool is mounted on the mounting flange. • **1**: The tool is a fixed tool. Default: **0**

E6AXIS

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

Element	Type	Description
X	REAL	Position on the X axis (unit: mm)
Y	REAL	Position on the Y axis (unit: mm)
Z	REAL	Position on the Z axis (unit: mm)
A	REAL	Rotation about Z axis • **-180° … +180°**
B	REAL	Rotation about Y axis • **-180° … +180°**
C	REAL	Rotation about X axis • **-180° … +180°**

Element	Type	Description	
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 the KUKA System Software (KSS).
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

Element	Type	Description
X	REAL	Position on the X axis (unit: mm)
Y	REAL	Position on the Y axis (unit: mm)
Z	REAL	Position on the Z axis (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°**

POSITION

Element	Description
COORDSYS	(>>> COORDSYS [▶ 19])
E6POS	(>>> E6POS [▶ 19])
E6AXIS	(>>> E6AXIS [▶ 19])

7.4 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 elements refer to the WORLD coordinate system and are defined in the variable BOX.

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

7.5 Data of an axis-specific workspace

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

These data are defined using the following elements in the variable AXBOX.

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/°	

7.6 Programming tips for TC3 mxAutomation

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_Initialize
- 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 procedure is that not for every function block an own data type is required.

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 or ComAcpt 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.
- The command `ExecuteCmd` should not be used as starting signal for a sequence of commands. This technique can cause that transmitting the subsequent commands takes valuable time.

Alternatively, performing a command sequence should be triggered with `KRC_WaitForInput`. By interconnecting the `Busy` and the `ExecuteCmd` signals it is ensured that performing the intended working steps can be started without delay as soon as the corresponding input signal comes in.

Program override

- If the `mxAutomation` robot program is started by a RESET at the `KRC_AutomaticExternal` function block, the program override has to be set to a value bigger than zero. Only in this case the robot interpreter is passed through in a loop.
- While the `mxAutomation` robot program runs, the program override can be set to zero value 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 after a motion statement a statement follows that evokes a preliminary run stop. Approximate positioning is only possible if there are 2 consecutive motion instructions.
- Approximate positioning is only possible if the motion statement is followed by a statement that is transferred in BUFFERED mode.

7.6.1 Structure of a PLC program

Each PLC program must contain the following function blocks:

1. KRC_ReadAxisGroup
2. KRC_Initialize
3. KRC_WriteAxisGroup

Any number of function blocks can be inserted between the function blocks KRC_Initialize and KRC_WriteAxisGroup, e.g. KRC_AutomaticExternal, MC_MoveDirectAbsolute, etc.

7.6.2 Stopping the robot

There are 4 ways of stopping the robot. They differ in terms of how the motion is to be resumed:

- Stop and cancel program execution (input RESET on function block KRC_AutomaticExternal)
- Stop and delete the buffered statements (KRC_Abort)
- Stop and wait for a condition, then resume motion and execute the buffered statements (KRC_DeclareInterrupt)
- Stop and wait for the function block KRC_Continue, then resume motion and execute the buffered statements (KRC_Interrupt)

8 Function blocks

8.1 Overview of function blocks

Administrative functions	
KRC_ReadAxisGroup	(>>> Reading outputs of the robot system [▶ 84])
KRC_WriteAxisGroup	(>>> Writing robot system inputs [▶ 84])
KRC_Initialize	(>>> Initializing the mxA interface [▶ 85])
KRC_SetOverride	(>>> Setting the program override (POV) [▶ 86])
KRC_AutomaticExternal	(>>> Activating and reading Automatic External signals from the robot controller [▶ 87])
KRC_AutoStart	(>>> Setting KRC_AutomaticExternal inputs automatically [▶ 89])
KRC_ReadActualPosition	(>>> Reading the current robot position [▶ 89])
KRC_ReadActualAxisPosition	(>>> Reading the current axis position [▶ 90])
KRC_ReadActualVelocity	(>>> Reading the current path velocity [▶ 91])
KRC_ReadActualAxisVelocity	(>>> Reading the current axis velocity [▶ 92])
KRC_ReadActualAcceleration	(>>> Reading the current robot acceleration [▶ 93])
KRC_ReadDigitalInput	(>>> Reading a digital input [▶ 93])
KRC_ReadDigitalInput1To8	(>>> Reading digital inputs 1 to 8 [▶ 94])
KRC_ReadDigitalInputArray	(>>> Reading multiple digital inputs [▶ 95])
KRC_ReadDigitalOutput	(>>> Reading a digital output [▶ 95])
KRC_WriteDigitalOutput	(>>> Writing a digital output [▶ 96])
KRC_WriteDigitalOutput1To8	(>>> Writing digital outputs 1 to 8 [▶ 97])
KRC_ReadAnalogInput	(>>> Reading an analog input [▶ 98])
KRC_ReadAnalogOutput	(>>> Reading an analog output [▶ 98])
KRC_WriteAnalogOutput	(>>> Writing an analog output [▶ 99])
KRC_SetCoordSys	(>>> Selecting the tool, base and interpolation mode [▶ 100])
KRC_ReadToolData	(>>> Reading TOOL data [▶ 100])
KRC_WriteToolData	(>>> Writing TOOL data [▶ 101])
KRC_ReadBaseData	(>>> Reading BASE data [▶ 102])
KRC_WriteBaseData	(>>> Writing BASE data [▶ 103])
KRC_ReadLoadData	(>>> Reading the load data [▶ 104])
KRC_WriteLoadData	(>>> Writing load data [▶ 105])
KRC_ReadSoftEnd	(>>> Reading the software limit switches of the robot axes [▶ 106])
KRC_ReadSoftEndExt	(>>> Reading the software limit switches of the external axes [▶ 106])
KRC_WriteSoftEnd	(>>> Writing the software limit switches of the robot axes [▶ 107])
KRC_WriteSoftEndExt	(>>> Writing the software limit switches of the external axes [▶ 108])

Functions for motion programming	
KRC_Jog	(>>> Jogging to an end position [▶ 27])
KRC_JogToolRelative	(>>> Jogging to a relative end position in the TOOL coordinate system with a linear motion [▶ 29])
KRC_JogLinearRelative	(>>> Jogging to a relative end position with a linear motion [▶ 30])

Functions for motion programming (PLC OPEN-compliant)	
MC_MoveLinearAbsolute	(>>> Approaching an absolute Cartesian position with a linear motion [▶ 32])
MC_MoveLinearRelative	(>>> Approaching a relative Cartesian position with a linear motion [▶ 33])
MC_MoveDirectAbsolute	(>>> Approaching an absolute Cartesian position as quickly as possible [▶ 35])
MC_MoveDirectRelative	(>>> Approaching a relative Cartesian position as quickly as possible [▶ 36])
MC_MoveAxisAbsolute	(>>> Approaching an axis-specific position as quickly as possible [▶ 37])
MC_MoveCircularAbsolute	(>>> Approaching an absolute Cartesian position with a circular motion [▶ 39])
MC_MoveCircularRelative	(>>> Approaching a relative Cartesian position with a circular motion [▶ 41])

Functions for program execution control	
KRC_Abort	(>>> Canceling a program [▶ 43])
KRC_Interrupt	(>>> Pausing robot motion [▶ 44])
KRC_Continue	(>>> Continuing a program [▶ 44])
KRC_WaitForInput	(>>> Waiting for a digital input [▶ 45])

Functions for interrupt programming	
KRC_DeclareInterrupt	(>>> Declaring interrupts [▶ 46])
KRC_ActivateInterrupt	(>>> Activating interrupts [▶ 47])
KRC_DeactivateInterrupt	(>>> Deactivating interrupts [▶ 48])
KRC_ReadInterruptState	(>>> Reading the state of an interrupt [▶ 49])

Functions for path-related switching actions	
KRC_SetDistanceTrigger	(>>> Activating a switching action for path points [▶ 50])
KRC_SetPathTrigger	(>>> Activating a path-related switching action [▶ 51])

Diagnostic functions	
KRC_Error	(>>> Reading and acknowledging error states [▶ 53])
KRC_ReadMXAStatus	(>>> Reading the current state of the mxA interface [▶ 54])
KRC_ReadMXAError	(>>> Reading error messages of the mxA interface [▶ 55])
KRC_MessageReset	(>>> Resetting error messages of the mxA interface [▶ 55])
KRC_ReadKRCError	(>>> Reading error messages of the robot controller [▶ 56])
KRC_Diag	(>>> Reading diagnostic signals [▶ 57])

General special functions	
KRC_ReadSysVar	(>>> Reading system variables [▶ 58])
KRC_WriteSysVar	(>>> Writing system variables [▶ 59])
KRC_BrakeTest	(>>> Calling a brake test [▶ 61])
KRC_MasRef	(>>> Calling a mastering test [▶ 62])
KRC_ReadSafeOPStatus	(>>> Reading the safety controller signals [▶ 64])
KRC_ReadTouchUPState	(>>> Reading the state of the TouchUp status keys [▶ 65])
KRC_TouchUP	(>>> Teaching points [▶ 65])

General special functions	
KRC_SetAdvance	(>>> Modifying settings for the advance run [▶ 66])
KRC_GetAdvance	(>>> Reading settings for the advance run [▶ 67])
KRC_Forward	(>>> Calculating the Cartesian robot position from the axis angles [▶ 68])
KRC_Inverse	(>>> Calculating axis angles from the Cartesian robot position [▶ 69])

Special functions for Conveyor	
KRC_ConvIniOff	(>>> Initializing a conveyor [▶ 70])
KRC_ConvOn	(>>> Activating a conveyor [▶ 71])
KRC_ConvFollow	(>>> Tracking a workpiece on the conveyor [▶ 71])
KRC_ConvSkip	(>>> Picking up a workpiece from the conveyor [▶ 73])
KRC_ActivateConvInterrupt	(>>> Activating interrupts for monitoring [▶ 74])
KRC_DeactivateConvInterrupt	(>>> Deactivating interrupts for monitoring [▶ 75])

In order to be able to use these function blocks, KUKA.ConveyorTech must be installed on the robot controller.

Special functions for VectorMove	
KRC_VectorMoveOn	(>>> Activating a motion along a vector [▶ 76])
KRC_VectorMoveOff	(>>> Deactivating motion along a vector [▶ 77])

In order to be able to use these function blocks, KUKA.VectorMove must be installed on the robot controller.

Special functions for workspaces	
KRC_WriteWorkspace	(>>> Configuring Cartesian workspaces [▶ 78])
KRC_ReadWorkspace	(>>> Reading the configuration of Cartesian workspaces [▶ 79])
KRC_WriteAxWorkspace	(>>> Configuring axis-specific workspaces [▶ 80])
KRC_ReadAxWorkspace	(>>> Reading the configuration of axis-specific workspaces [▶ 81])
KRC_ReadWorkstates	(>>> Reading the status of the workspaces [▶ 82])

8.2 Functions for motion programming

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.

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 Busy output of the previous function block is connected to the ExecuteCmd input of the subsequent block.

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.

8.2.1 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.

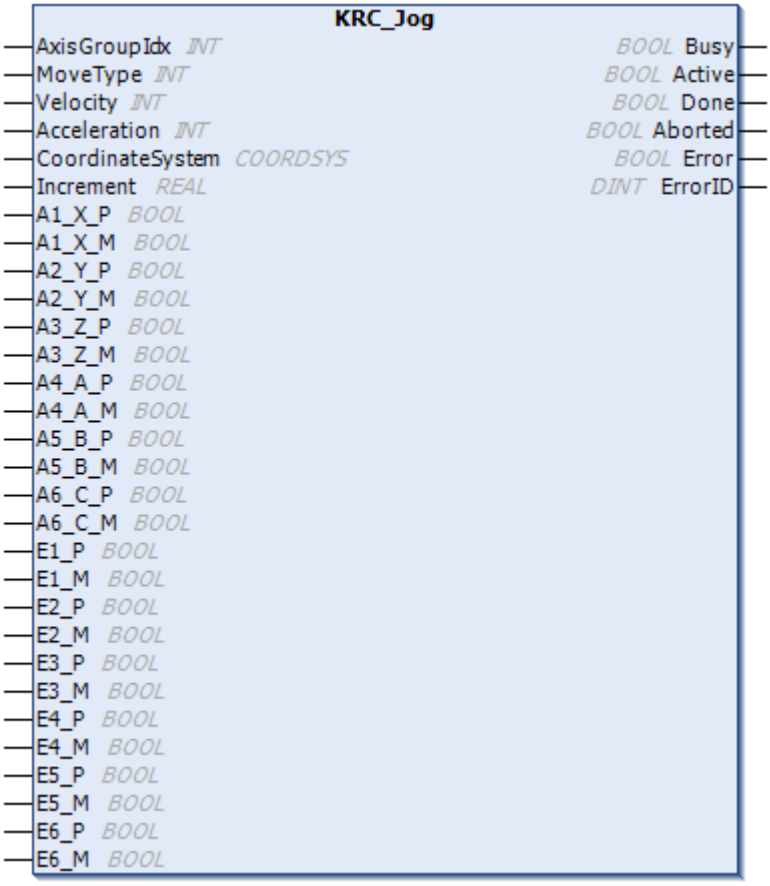


Fig. 3: Function block KRC_Jog

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
MoveType	INT	Motion type for Cartesian or axis-specific jogging • **0**: Axis-specific • **1**: Cartesian • **2**: Axis-specific, as spline motion • **3**: Cartesian, as spline motion
Velocity	INT	Velocity • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. In the case of MoveType 1 and 3, the maximum value refers to the value of DEF_VEL_CP of the robot system. Default value: **0%** (= velocity is not changed)

Parameter	Type	Description
Acceleration	INT	Acceleration • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. In the case of MoveType 1 and 3, the maximum value refers to the value of DEF_ACC_CP of the robot system. Default value: **0%** (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the coordinates of the end position refer. (>>> COORDSYS [▶ 19])
Increment	REAL	Incremental jogging The maximum distance for incremental jogging can be limited using this parameter. • **> 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 the case of MoveType 0 and 2, 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. 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. Depending on the MoveType, 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
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
ErrorID	DINT	Error number

8.2.2 Jogging to a relative end position in the TOOL coordinate system with a linear motion

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. 4: Function block KRC_JogToolRelative

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.
Position	E6POS	Distance from origin to current position (>>> E6POS [▶ 19]) 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 • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0%** (= velocity is not changed)

Parameter	Type	Description
Acceleration	INT	Acceleration • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: **0%** (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> COORDSYS [▶ 19])
OriType	INT	Orientation control of the TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 15])

Outputs

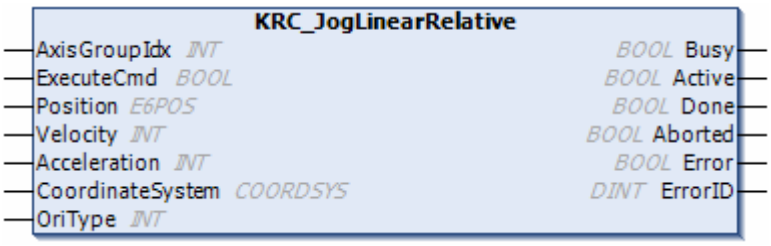
Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = motion is currently being executed
Done	BOOL	TRUE = motion has stopped
Aborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block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

8.2.3 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.



Function block KRC_JogLinearRelative

Inputs

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position >>> "E6POS". 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	>>> "COORDSYS".
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	>>> "OriType".

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

8.3 Functions for motion programming (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 (>>> Frequently used input/output signals in the MC function blocks).

8.3.1 Approaching an absolute Cartesian position with a linear 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.

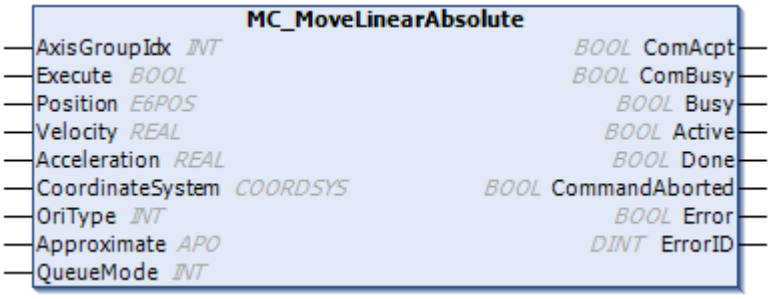


Fig. 5: Function block MC_MoveLinearAbsolute

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 [▶ 19]) 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	REAL	Velocity for the path motion • **0 … 2 m/s** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0 m/s** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • 0 … 2.3 m/s^2 The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: 0 m/s^2 (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> COORDSYS [▶ 19]) **Note**: In the case of a linear motion, the Cartesian coordinates always refer to the BASE coordinate system.
OriType	INT	Orientation control of the TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 15])

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

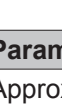
Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.2 Approaching a relative Cartesian position with a linear motion

Description

The function block MC_MoveLinearRelative is used to execute a linear motion to a relative Cartesian end position. The parameter “Position” contains the path from the current position to the end position.



This statement always refers to the current position of the robot. If the motion is interrupted and then re-executed, the robot executes the entire motion again, starting from the position at which it was interrupted.

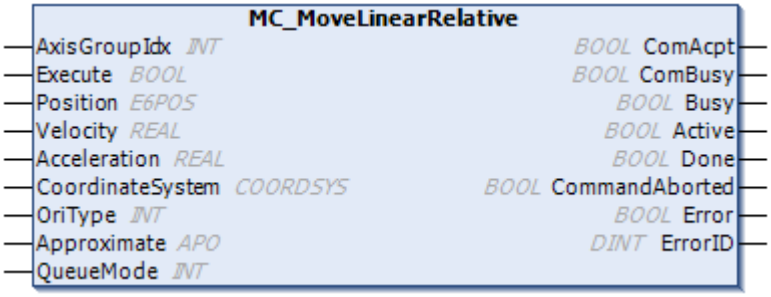


Fig. 6: Function block MC_MoveLinearRelative

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.

Parameter	Type	Description
Position	E6POS	Distance from origin to current position (>>> E6POS [▶ 19]) 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	REAL	Velocity for the path motion • **0 … 2 m/s** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0 m/s** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • 0 … 2.3 m/s^2 The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: 0 m/s^2 (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> COORDSYS [▶ 19])
OriType	INT	Orientation control of the TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 15])
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.3 Approaching an absolute Cartesian position as quickly as possible

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.

The robot moves as quickly as possible to the end position. The fastest path is generally not the shortest path and is thus not a straight line. This corresponds to a PTP motion on the robot system.

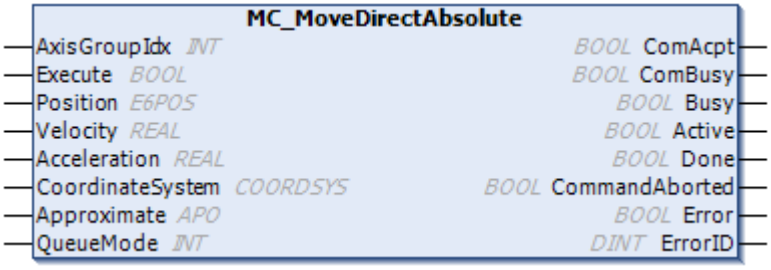


Fig. 7: Function block MC_MoveDirectAbsolute

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 [▶ 19]) 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	REAL	Velocity for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0%**
Acceleration	REAL	Acceleration for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: **0%** (= velocity is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> COORDSYS [▶ 19]) **Note**: In the case of a point-to-point motion, the Cartesian coordinates always refer to the BASE coordinate system.
Approximate	APO	Approximation parameter (>>> APO [▶ 18])

Parameter	Type	Description
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.4 Approaching a relative Cartesian position as quickly as possible

Description

The function block MC_MoveDirectRelative executes a point-to-point motion to a relative Cartesian end position. The parameter “Position” contains the path from the current position to the end position. This corresponds to a PTP_REL motion on the robot system.

This statement always refers to the current position of the robot. If the motion is interrupted and then re-executed, the robot executes the entire motion again, starting from the position at which it was interrupted.

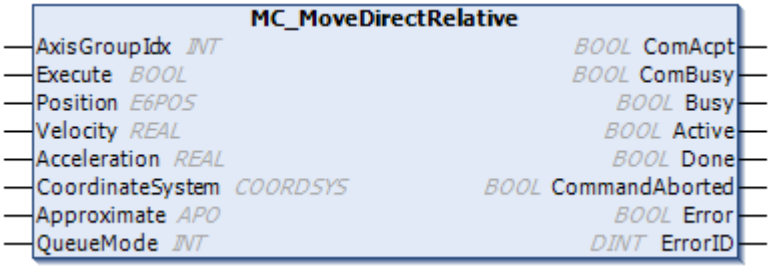


Fig. 8: Function block MC_MoveDirectRelative

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.

Parameter	Type	Description
Position	E6POS	Coordinates of the Cartesian end position (>>> E6POS [▶ 19]) 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	REAL	Velocity for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0%** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: **0%** (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the end position refer (>>> COORDSYS [▶ 19]) **Note** : At a PTP_REL-motion the Cartesian Coordinates always refer to the BASE-Coordinate System.
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.5 Approaching an axis-specific position as quickly as possible

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.

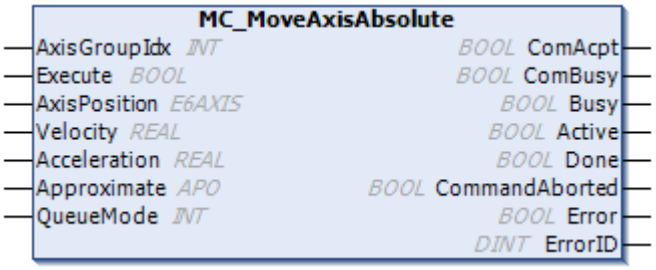


Fig. 9: Function block MC_MoveAxisAbsolute

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.
AxisPosition	E6AXIS	Axis-specific end position (>>> E6AXIS [▶ 19]) The data structure E6Axis contains the angle values or translation values for all axes of the axis group in the end position.
Velocity	REAL	Velocity for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0%** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • **0 … 100%** The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: **0%** (= acceleration is not changed)
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.6 Approaching an absolute Cartesian position with a circular 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.

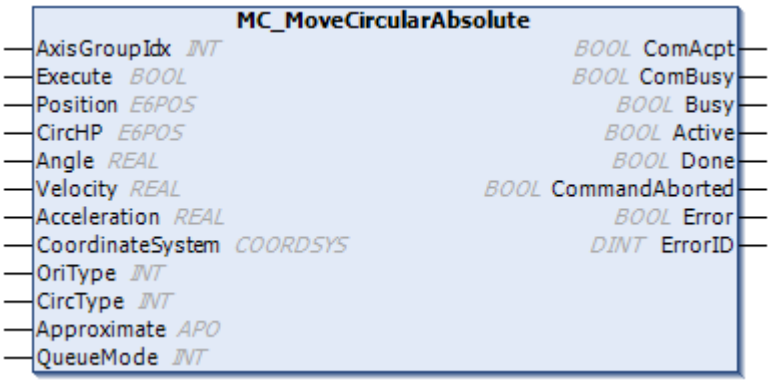


Fig. 10: Function block MC_MoveCircularAbsolute

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 [▶ 19]) 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 [▶ 19]) 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	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 ±360° 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	REAL	Velocity for the path motion • **0 … 2 m/s** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0 m/s** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • 0 … 2.3 m/s^2 The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: 0 m/s^2 (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer (>>> COORDSYS [▶ 19]) **Note**: In the case of a circular motion, the Cartesian coordinates always refer to the BASE coordinate system.
OriType	INT	Orientation control of the TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 15])
CircType	INT	Orientation control during the circular motion • **0**: BASE • **1**: PATH (>>> CircType [▶ 15])
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.3.7 Approaching a relative Cartesian position with a circular 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.

This statement always refers to the current position of the robot. If the motion is interrupted and then re-executed, the robot executes the entire motion again, starting from the position at which it was interrupted.

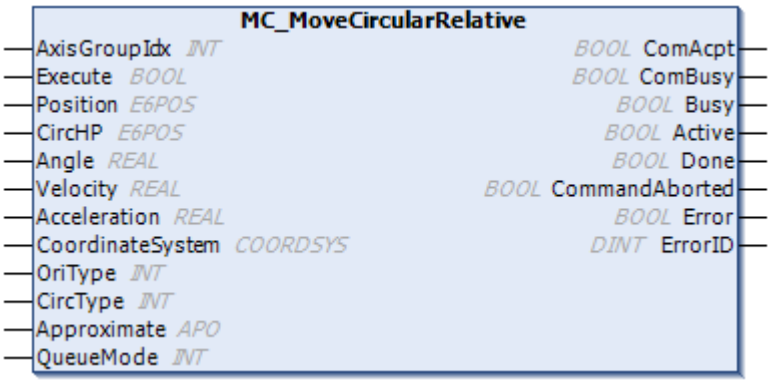


Fig. 11: Function block MC_MoveCircularRelative

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.

Parameter	Type	Description
Position	E6POS	Distance from origin to current position (>>> E6POS [▶ 19]) 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 (relative to the current position) (>>> E6POS [▶ 19]) 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 ±360° 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	REAL	Velocity for the path motion • **0 … 2 m/s** The maximum value is dependent on the robot type and refers to the value of DEF_VEL_CP of the robot system. Default: **0 m/s** (= velocity is not changed)
Acceleration	REAL	Acceleration for the path motion • 0 … 2.3 m/s^2 The maximum value is dependent on the robot type and refers to the value of DEF_ACC_CP of the robot system. Default: 0 m/s^2 (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the auxiliary or end position refer (>>> COORDSYS [▶ 19])
OriType	INT	Orientation control of the TCP • **0**: VAR • **1**: CONSTANT • **2**: JOINT (>>> OriType [▶ 15])
CircType	INT	Orientation control during the circular motion • **0**: BASE • **1**: PATH (>>> CircType [▶ 15])

Parameter	Type	Description
Approximate	APO	Approximation parameter (>>> APO [▶ 18])
QueueMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> QueueMode [▶ 15])

Outputs

Parameter	Type	Description
ComAcpt	BOOL	TRUE = statement was completely transferred and confirmed by the robot controller.
ComBusy	BOOL	TRUE = statement was transferred and confirmed by the robot controller, but has not yet been executed completely.
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
CommandAborted	BOOL	TRUE = statement/motion has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.4 Functions for program execution control

8.4.1 Canceling a program

Description

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

Statements and motions of function blocks without BufferMode or QueueMode that are executed cyclically are not canceled.

KRC_Abort is not processed if the function block KRC_Interrupt is active. In this case, the program must first be resumed with KRC_Continue before it can be aborted with KRC_Abort.

Fig. 12: Function block KRC_Abort

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
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.4.2 Pausing robot motion

Description

The function block KRC_Interrupt is used to stop the robot. It brakes either gently (BRAKE) or as quickly as possible (BRAKE F) from high velocities.

> 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. 13: Function block KRC_Interrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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 as quickly as possible FALSE = robot stops gently

Outputs

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

8.4.3 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.

If KRC_Continue is used together with KRC_Interrupt, the input Execute for KRC_Interrupt must have the value FALSE before KRC_Continue can be executed.

Fig. 14: 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.

Outputs

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

8.4.4 Waiting for a digital input

Description

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

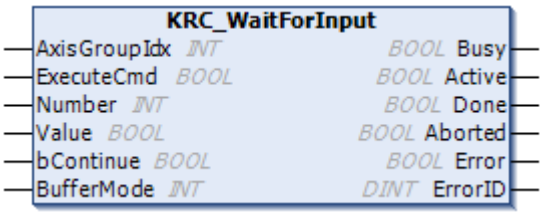


Fig. 15: Function block KRC_WaitForInput

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 (corresponds to $IN[1 … 2048] on the robot controller) • **1 … 2048**
Value	BOOL	Value of the digital input
bContinue	BOOL	TRUE = poll input in advance run **Note**: The robot controller executes programs with an advance run and a main run. Further information about the advance run and main run is contained in the operating and programming instructions for the KUKA System Software (KSS).
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
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
ErrorID	DINT	Error number

8.5 Functions for interrupt programming

8.5.1 Declaring interrupts

Description

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

An interrupt can be used to stop the robot during the motion. Depending on how the Reaction parameter is configured, the robot brakes gently from high velocities (BRAKE) or as quickly as possible (BRAKE F). The remaining program sequence can be determined using a robot input or a PLC function block.

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.

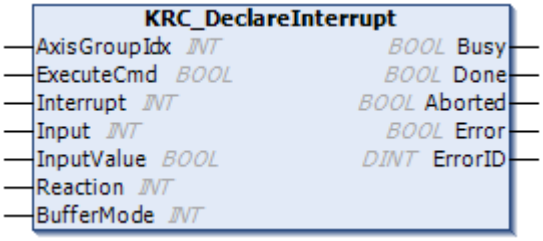


Fig. 16: Function block KRC_DeclareInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Interrupt	INT	Number of the interrupt • **1 … 8** **Note**: Number 1 is the interrupt with the highest priority.
Input	INT	Number of the digital input to which the interrupt is declared • **1 … 2048** **Note**: It must be ensured that no inputs are used that are already assigned by the system. Example: $IN[1025] is always TRUE.

Parameter	Type	Description
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 • **0**: Brake fast and wait for the EXT_START parameter of the function block KRC_AutomaticExternal • **1**: Brake normally and wait for the EXT_START parameter of the function block KRC_AutomaticExternal • **2**: Brake fast and wait for the Input parameter • **3**: Brake normally and wait for the Input parameter • **4**: Brake fast and wait for the edge of the function block KRC_Continue • **5**: Brake normally and wait for the edge of the function block KRC_Continue • **6**: Brake fast and wait for the edge of the function block KRC_Continue and the Input parameter • **7**: Brake normally and wait for the edge of the function block KRC_Continue and the Input parameter
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

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

8.5.2 Activating interrupts

Description

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

The function block KRC_ReadInterruptState can be used to monitor and check whether an interrupt is active.

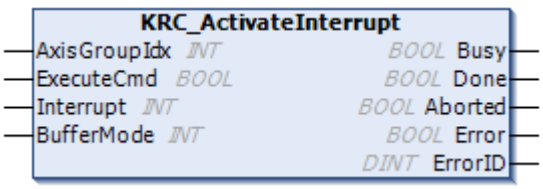


Fig. 17: Function block KRC_ActivateInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Interrupt	INT	Number of the interrupt • **1 … 8**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

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

8.5.3 Deactivating interrupts

Description

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

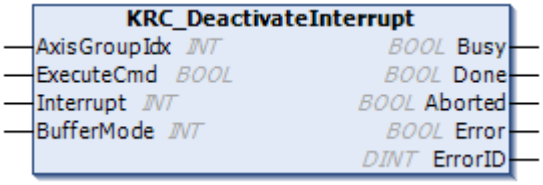


Fig. 18: Function block KRC_DeactivateInterrupt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Interrupt	INT	Number of the interrupt • **1 … 8**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.5.4 Reading the state of an interrupt

Description

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

Fig. 19: Function block KRC_ReadInterruptState

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
Interrupt	INT	Number of the interrupt • **1 … 8**

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
Value	INT	State of specified interrupt • **0**: Interrupt has not been declared. • **1**: Interrupt has been declared. • **2**: Interrupt has been declared and activated. • **3**: Interrupt has been declared and activated and has now been deactivated again (see Status 1). • **4**: Interrupt has been triggered and is active. • **5**: Interrupt has been triggered and the main program has already been resumed with the function block KRC_Continue.
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.6 Functions for path-related switching actions

8.6.1 Activating a switching action for path points

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. 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.

Further information on triggers, on offsetting the switching point and on the offset limits can be found in the operating and programming instructions for the KUKA System Software (KSS).

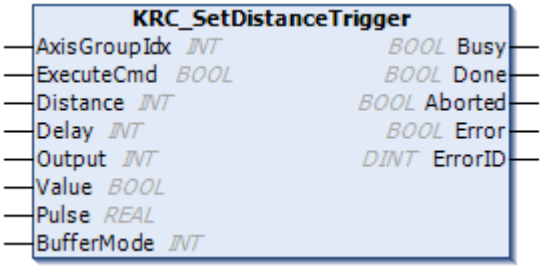


Fig. 20: Function block KRC_SetDistanceTrigger

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
ExecuteCmd	BOOL	The statement is buffered in the case of a rising edge of the signal.
Distance	INT	Switching point of the trigger • **0**: Switching action at the start point • **1**: Switching action at the end point
Delay	INT	Statement delay • **Delay = 0 ms**: no delay 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 the KUKA System Software (KSS).
Output	INT	Number of the digital output that can be set by the switching action • **1 … 2048** **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	TRUE = activate output FALSE = deactivate output

Parameter	Type	Description
Pulse	REAL	Length of the pulse • **0.0 s** No pulse active • **0.1 … 3.0 s** Pulse interval = 0.1 s
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed **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 has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.6.2 Activating a path-related switching action

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. 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.

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.

Further information on triggers, on offsetting the switching point and on the offset limits can be found in the operating and programming instructions for the KUKA System Software (KSS).

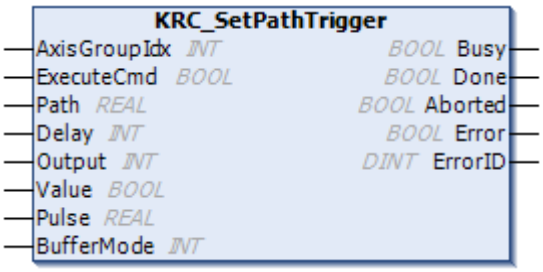


Fig. 21: Function block KRC_SetPathTrigger

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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. • **Path = 0.0 mm**: no offset • **Path > 0.0 mm**: shifts the statement towards the end of the motion. • **Path < 0.0 mm**: shifts the statement towards the start of the motion.
Delay	INT	Statement delay • **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 the KUKA System Software (KSS).
Output	INT	Number of the digital output that can be set by the switching action • **1 … 2048** **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	TRUE = activate output FALSE = deactivate output
Pulse	REAL	Length of the pulse • **0.0 s** No pulse active • **0.1 … 3.0 s** Pulse interval = 0.1 s
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Done	BOOL	TRUE = statement has been processed **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 has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.7 Diagnostic functions

8.7.1 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.

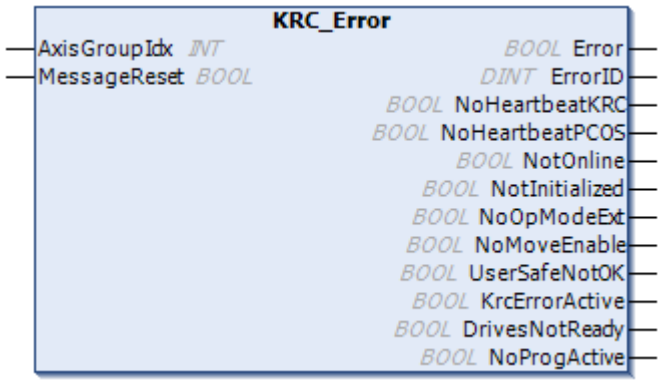


Fig. 22: Function block KRC_Error

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
MessageReset	BOOL	Acknowledges error messages of the mxA interface and the robot controller 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

Parameter	Type	Description
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.
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.

8.7.2 Reading the current state of the mxA interface

Description

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

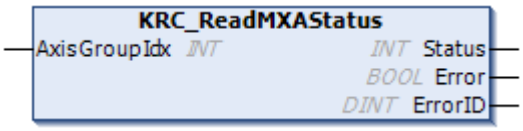


Fig. 23: Function block KRC_ReadMXAStatus

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Outputs

Parameter	Type	Description
Status	INT	Current state of the mxA interface (>>> Status [▶ 54])
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

Status

Current state of the mxA interface (function block KRC_ReadMXAStatus)

Value	Name	Description
0	Invalid	No function blocks can be processed. Frequent causes: • 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.

Value	Name	Description
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.

8.7.3 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.

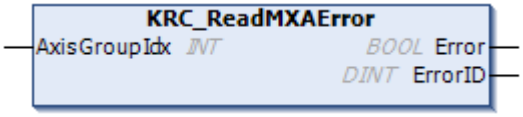


Fig. 24: Function block KRC_ReadMXAError

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Outputs

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

8.7.4 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.

Messages can only be reset if the robot is stationary.

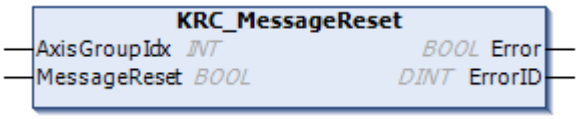


Fig. 25: Function block KRC_MessageReset

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Parameter	Type	Description
MessageReset	BOOL	TRUE = reset message

Outputs

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

8.7.5 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.

Messages can only be reset if the robot is stationary.

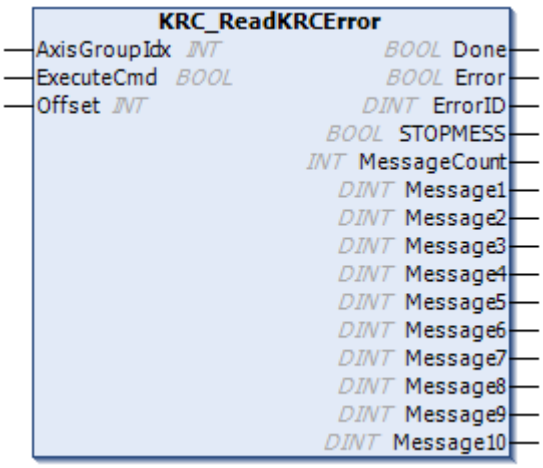


Fig. 26: Function block KRC_ReadKRCError

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.
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
Done	BOOL	TRUE = data are valid
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number
STOPMESS	BOOL	TRUE = safety circuit is interrupted (robot fault)
MessageCount	INT	Number of messages in the message buffer

Parameter	Type	Description
Message1 … Message10	DINT	The numbers of up to 10 messages in the message buffer can be output.

8.7.6 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.

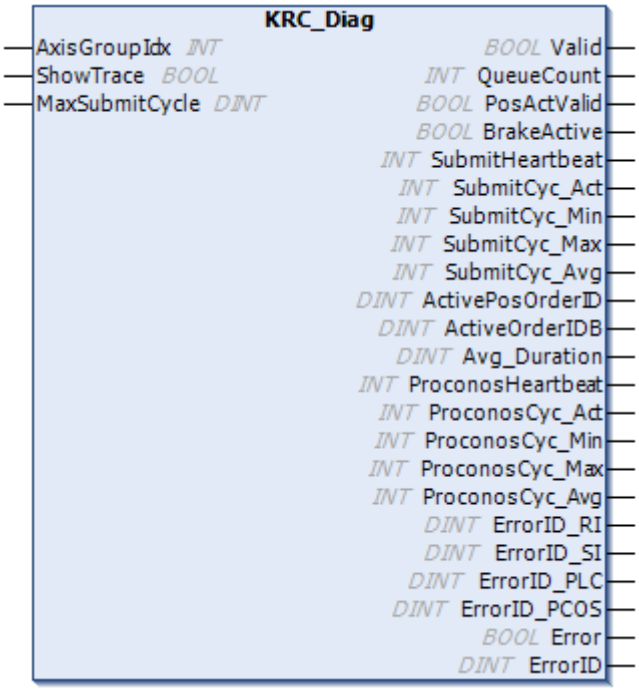


Fig. 27: Function block KRC_Diag

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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
Valid	BOOL	TRUE = data are valid
QueueCount	INT	Number of buffered statements • **1 … 90**
PosActValid	BOOL	TRUE = position data are valid (BCO)
BrakeActive	BOOL	TRUE = robot is stopped by means of a BRAKE statement
SubmitHeartbeat	INT	Heartbeat signal of the submit interpreter (counter is incremented by 1 every Submit cycle) • **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
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
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
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
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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.8 General special functions

8.8.1 Reading system variables

Description

The function block KRC_ReadSysVar can be used to read system variables.

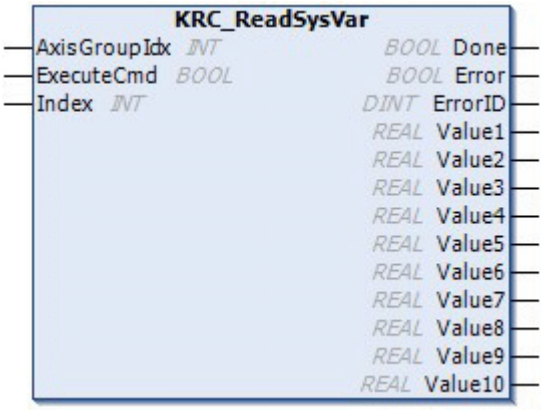


Fig. 28: Function block KRC_ReadSysVar

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.
Index	INT	Index of the system variable • **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
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number
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.

8.8.2 Writing system variables

Description

The function block KRC_WriteSysVar can be used to write system variables.

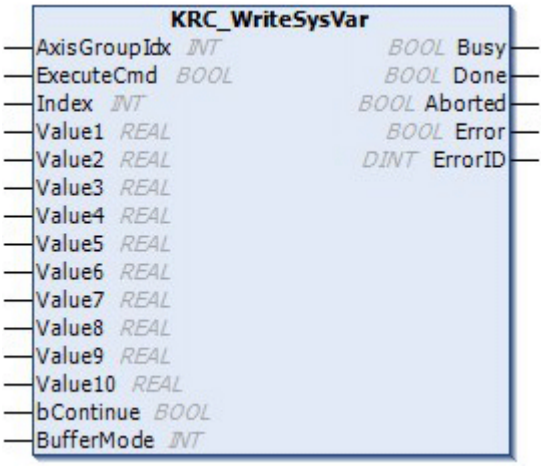


Fig. 29: Function block KRC_WriteSysVar

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.
Index	INT	Index of the system variable • **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.
bContinue	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 • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.8.3 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 the KUKA System Software (KSS).

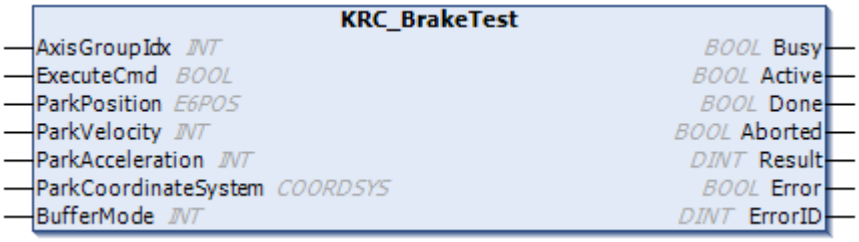


Fig. 30: Function block KRC_BrakeTest

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.
ParkPosition	E6POS	Coordinates of the Cartesian parking position (>>> E6POS [▶ 19]) The data structure E6POS contains all components of the parking position (= position of the TCP relative to the origin of the selected coordinate system).

Parameter	Type	Description
ParkVelocity	INT	Velocity • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. Default: **0%** (= velocity is not changed)
ParkAcceleration	INT	Acceleration • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. Default: **0%** (= acceleration is not changed)
ParkCoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the parking position refer (>>> COORDSYS [▶ 19])
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
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
Result	DINT	Result of the brake test • **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)
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.8.4 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 KUKA.SafeOperation documentation.

Fig. 31: Function block KRC_MasRef

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.
Position	E6POS	Coordinates of the Cartesian reference position (>>> E6POS [▶ 19]) The data structure E6POS contains all components of the reference position (= position of the TCP relative to the origin of the selected coordinate system).
Velocity	INT	Velocity • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. Default: **0%** (= velocity is not changed)
Acceleration	INT	Acceleration • **0 … 100%** Refers to the maximum value specified in the machine data. The maximum value is dependent on the robot type. Default: **0%** (= acceleration is not changed)
CoordinateSystem	COORDSYS	Coordinate system to which the Cartesian coordinates of the reference position refer (>>> COORDSYS [▶ 19])
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
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

Parameter	Type	Description
ErrorID	DINT	Error number
MasRefRequest	BOOL	TRUE = mastering test has been requested internally by the robot controller

8.8.5 Reading the safety controller signals

Description

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

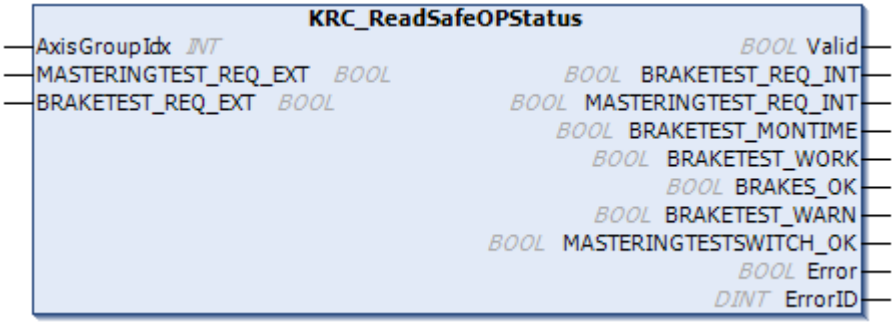


Fig. 32: Function block KRC_ReadSafeOPStatus

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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
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.
MASTERINGTESTSWITCH_OK	BOOL	TRUE = reference switch is OK.
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.8.6 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.

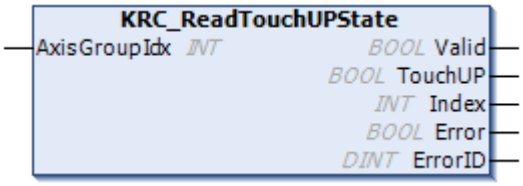


Fig. 33: Function block KRC_ReadTouchUPState

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Outputs

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

8.8.7 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 saved by the function block.

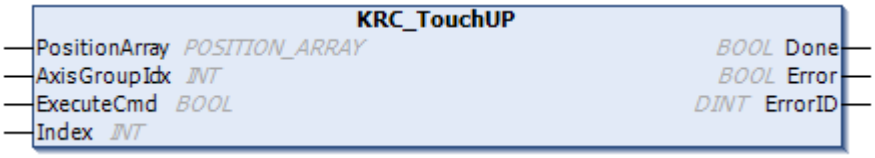


Fig. 34: Function block KRC_TouchUP

Inputs

Parameter	Type	Description
PositionArray	POSITION_ ARRAY	Array with the taught points and the corresponding coordinate systems **Note**: The array can be used directly in the PLC. (>>> POSITION_ ARRAY)

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
ExecuteCmd	BOOL	TRUE = the point is taught
Index	INT	Number under which the taught point is saved in the PLC • **1 … 100**

Outputs

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

8.8.8 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.

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

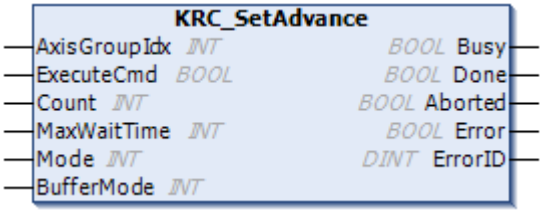


Fig. 35: Function block KRC_SetAdvance

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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 • **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. • **1 … 32 767 ms** Default value: 300 ms

Parameter	Type	Description
Mode	INT	Wait mode • **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 • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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

8.8.9 Reading settings for the advance run

Description

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

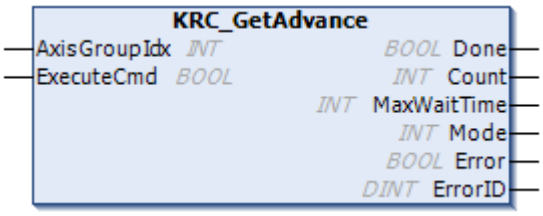


Fig. 36: Function block KRC_GetAdvance

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
Done	BOOL	TRUE = statement has been executed

Parameter	Type	Description
Count	INT	Number of functions to be transferred by the PLC before the first robot motion • **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. • **1 … 32 767 ms** Default value: 300 ms
Mode	INT	Wait mode • **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

8.8.10 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.

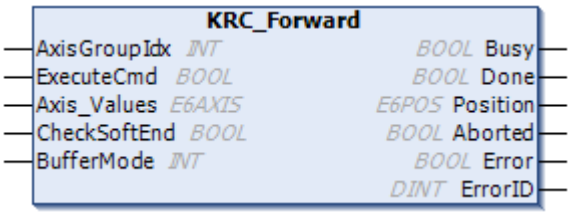


Fig. 37: Function block KRC_Forward

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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 (>>> E6AXIS [▶ 19]) The data structure E6AXIS contains the angle values or translation values for all axes of the axis group in this position.
CheckSoftEnd	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 • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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 has been aborted
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.8.11 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.

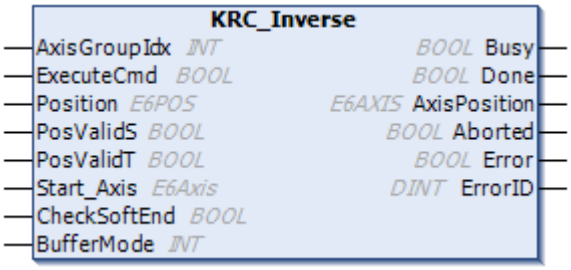


Fig. 38: Function block KRC_Inverse

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **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.
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.
CheckSoftEnd	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 • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

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

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
AxisPosition	E6AXIS	Axis angles that have been calculated from the specified Cartesian robot position (>>> E6AXIS [▶ 19]) 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

8.9 Special functions for Conveyor

8.9.1 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. 39: Function block KRC_ConvIniOff

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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.9.2 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. 40: Function block KRC_ConvOn

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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.9.3 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. 41: Function block KRC_ConvFollow

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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
StartDistance	REAL	Distance traveled by the workpiece while the robot waits before starting to track the workpiece on the conveyor. • 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. • In the case of a linear conveyor: Specification in millimeters • In the case of a circular conveyor: Specification in degrees **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.
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = statement is currently being executed
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

8.9.4 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. 42: Function block KRC_ConvSkip

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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
PieceNumber	INT	The number entered specifies which workpieces are to be picked up. Examples: • **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. • 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. • In the case of a linear conveyor: Specification in millimeters • In the case of a circular conveyor: Specification in degrees **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.
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = statement is currently being executed
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

8.9.5 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. 43: Function block KRC_ActivateConvInterrupt

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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = statement is currently being executed
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

8.9.6 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. 44: 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.
ConveyorNumber	INT	Number of the conveyor • **1 … 3**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

Outputs

Parameter	Type	Description
Busy	BOOL	TRUE = statement is currently being transferred or has already been transferred
Active	BOOL	TRUE = statement is currently being executed
Done	BOOL	TRUE = statement has been executed

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

8.10 Special functions for VectorMove

8.10.1 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.

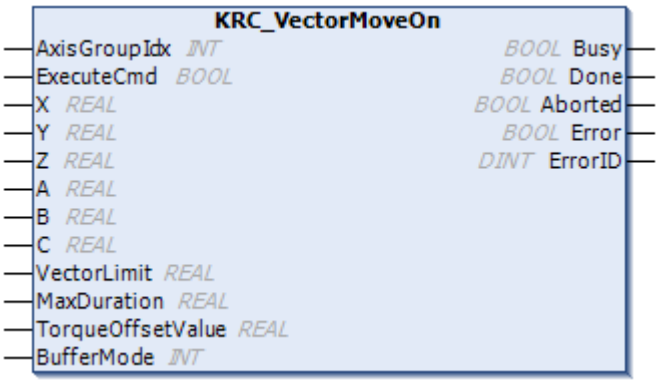


Fig. 45: Function block KRC_VectorMoveOn

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.
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 called is the root point of the vector. Limits: • Translational motion (X, Y, Z): max. 200 mm in the positive and negative directions • Rotational motion (A, B, C): max. 30° in the positive and negative directions
Y	REAL	
Z	REAL	
A	REAL	
B	REAL	
C	REAL	
VectorLimit	REAL	Permissible vector length limit If this value is exceeded, the robot is stopped with ramp-down braking. • **0 … 30%** Default: **10%**
MaxDuration	REAL	Length of time after which VectorMove is deactivated if an error has occurred • **0 … 10000 s** Default: **100 s**

Parameter	Type	Description
TorqueOffsetValue	REAL	Defines the resistance torque of the robot (unit: Nm) 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. • **1 … 200 Nm** Default: **1 Nm**
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.10.2 Deactivating motion along a vector

Description

The function block KRC_VectorMoveOff is used to deactivate the motion along a vector.

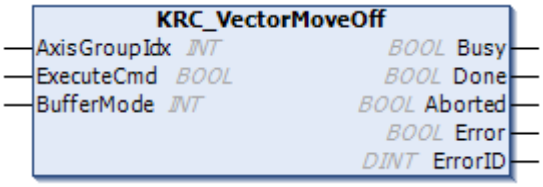


Fig. 46: Function block KRC_VectorMoveOff

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.
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.11 Special functions for workspaces

8.11.1 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.

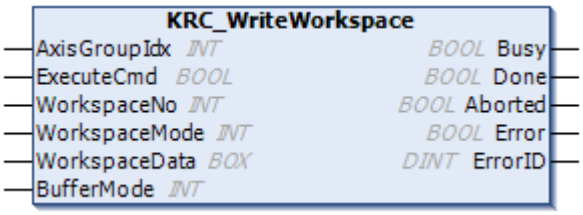


Fig. 47: Function block KRC_WriteWorkspace

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.
WorkspaceNo	INT	Number of workspace • **1 … 8**
WorkspaceMode	INT	Mode for workspaces • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP Only for Tc3_mxAutomationV3_0, additionally: • **5** : #TCP_INSIDE • **6** : #TCP_OUTSIDE • **7** : #TCP_INSIDE_STOP • **8** : #TCP_OUTSIDE_STOP **Note**: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).

Parameter	Type	Description
WorkspaceData	BOX	Data of the workspace (>>> Data of a Cartesian workspace [▶ 20])
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.11.2 Reading the configuration of Cartesian workspaces

Description

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

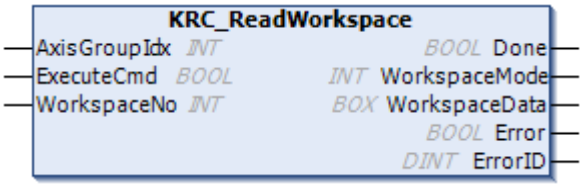


Fig. 48: Function block KRC_ReadWorkspace

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.
WorkspaceNo	INT	Number of workspace • **1 … 8**

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed

Parameter	Type	Description
WorkspaceMode	INT	Mode for workspaces • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP Only for Tc3_mxAutomationV3_0, additionally: • **5** : #TCP_INSIDE • **6** : #TCP_OUTSIDE • **7** : #TCP_INSIDE_STOP • **8** : #TCP_OUTSIDE_STOP **Note:** Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).
WorkspaceData	BOX	Data of the workspace (>>> Data of a Cartesian workspace [▶ 20])
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.11.3 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.

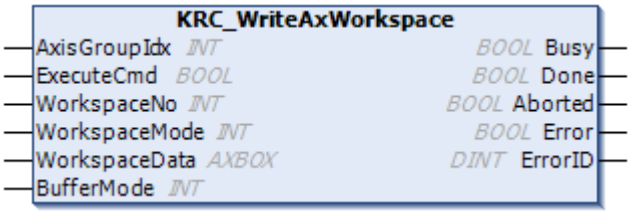


Fig. 49: Function block KRC_WriteAxWorkspace

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.
WorkspaceNo	INT	Number of workspace • **1 … 8**

Parameter	Type	Description
WorkspaceMode	INT	Mode for workspaces • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP Only for Tc3_mxAutomationV3_0, additionally: • **5** : #TCP_INSIDE • **6** : #TCP_OUTSIDE • **7** : #TCP_INSIDE_STOP • **8** : #TCP_OUTSIDE_STOP **Note**: Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).
WorkspaceData	AXBOX	Data of the workspace (>>> Data of an axis-specific workspace [▶ 21])
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.11.4 Reading the configuration of axis-specific workspaces

Description

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

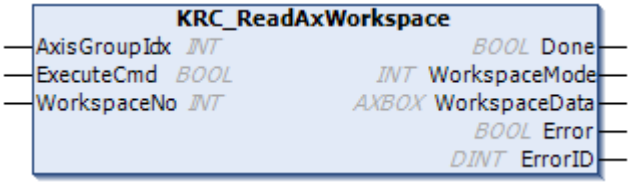


Fig. 50: Function block KRC_ReadAxWorkspace

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.
WorkspaceNo	INT	Number of workspace • **1 … 8**

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
WorkspaceMode	INT	Mode for workspaces • **0**: #OFF • **1**: #INSIDE • **2**: #OUTSIDE • **3**: #INSIDE_STOP • **4**: #OUTSIDE_STOP Only for Tc3_mxAutomationV3_0, additionally: • **5** : #TCP_INSIDE • **6** : #TCP_OUTSIDE • **7** : #TCP_INSIDE_STOP • **8** : #TCP_OUTSIDE_STOP **Note:** Further information about the mode for workspaces is contained in the operating and programming instructions for the KUKA System Software (KSS).
WorkspaceData	AXBOX	Data of the workspace (>>> Data of an axis-specific workspace [▶ 21])
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.11.5 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.

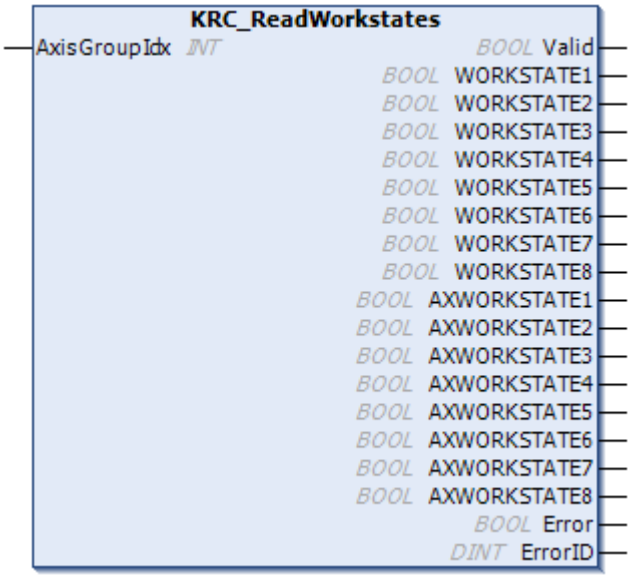


Fig. 51: Function block KRC_ReadWorkstates

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

8.12 Administrative functions

8.12.1 Reading outputs of the robot system

Description

The function block KRC_ReadAxisGroup is used to assign the parameter KRC4_Input to the parameter AxisGroupIdx. The following function blocks always refer to the same robot system using the parameter AxisGroupIdx.

The function block KRC_ReadAxisGroup must always be called at the start of the program. Access to the parameter AxisGroupIdx is only permissible between the function blocks KRC_ReadAxis-Group and KRC_WriteAxisGroup.

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. 52: Function block KRC_ReadAxisGroup

Inputs

Parameter	Type	Description
KRC4_Input	POINTER TO BYTE	Defines the start index of the output 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

8.12.2 Writing robot system inputs

Description

The function block KRC_WriteAxisGroup uses the parameter AxisGroupIdx to write the inputs to the range defined by the parameter KRC4_Output.

The function block KRC_WriteAxisGroup must always be called at the end of the program. Access to AxisGroupIdx is only permissible between the function blocks KRC_ReadAxisGroup and KRC_WriteAxisGroup.

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.

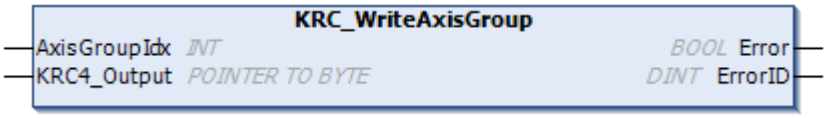


Fig. 53: Function block KRC_WriteAxisGroup

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
KRC4_Output	POINTER TO BYTE	Defines the start index of the input range of the robot controller

Outputs

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

8.12.3 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.

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.

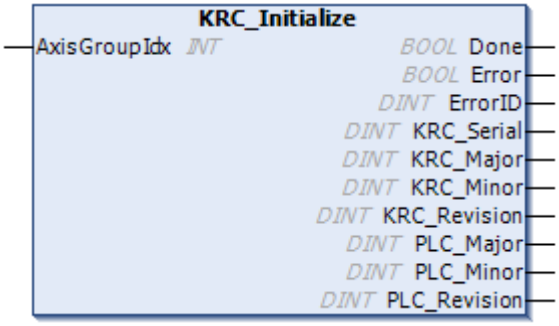


Fig. 54: Function block KRC_Initialize

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = initialization successfully completed
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

Parameter	Type	Description
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)

8.12.4 Setting the program override (POV)

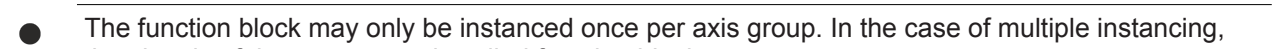
Description

The function block KRC_SetOverride sets the program override.

Program override is the velocity of the robot during program execution. It 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.

The program override refers to all motions of the robot system.



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. 55: Function block KRC_SetOverride

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
Override	INT	Program override • **0 … 100%**

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
ActualOverride	INT	Current override setting • **0 … 100%**
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.5 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.

The function block KRC_AutoStart can be used to simplify use of KRC_AutomaticExternal.

(>>> Setting KRC_AutomaticExternal inputs automatically [▶ 89])

The function block requires the Automatic External mode of the robot system. Further information about this functionality is contained in the operating and programming instructions for the KUKA System Software (KSS).

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.

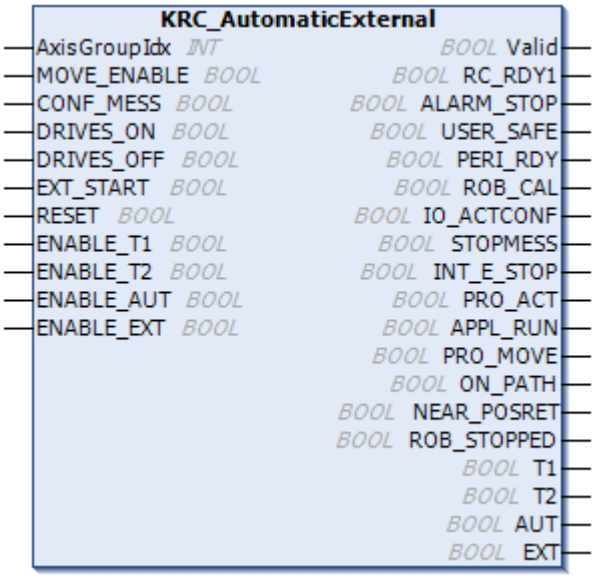


Fig. 56: Function block KRC_AutomaticExternal

Inputs

Parameter	Type	Signal name (robot controller)	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.

Parameter	Type	Signal name (robot controller)	Description
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 (robot controller)	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
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

8.12.6 Setting KRC_AutomaticExternal inputs automatically

Description

The function block KRC_AutoStart automatically sets the existing signals of the KRC_AutomaticExternal function block in the correct sequence. This function block can be used to activate the Automatic External interface of the robot system without the need for in-depth knowledge of the individual steps.

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 following inputs must additionally be set at the function block KRC_AutomaticExternal:

- MOVE_ENABLE
- ENABLE_T1
- ENABLE_T2
- ENABLE_AUT
- ENABLE_EXT
- DRIVES_OFF

All other inputs are set by the function block KRC_AutoStart.

Fig. 57: 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

8.12.7 Reading the current robot position

Description

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

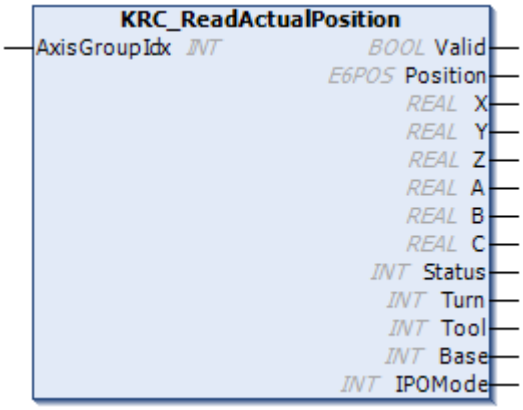


Fig. 58: Function block KRC_ReadActualPosition

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 ($POS_ACT on the robot controller) The data structure E6POS contains all components of the Cartesian actual position (= position of the TCP relative to the origin of the BASE coordinate system).
X, Y, Z	REAL	Current actual position in the X, Y, Z directions
A, B, C	REAL	Orientation A, B, C in the current actual position
Status	INT	Status of the current actual position
Turn	INT	Turn of the current actual position
Tool	INT	Number of the currently used TOOL coordinate system ($ACT_TOOL_C on the robot controller)
Base	INT	Number of the currently used BASE coordinate system ($ACT_BASE_C on the robot controller)
IPOMode	INT	Current interpolation mode ($IPO_MODE_C on the robot controller)

8.12.8 Reading the current axis position

Description

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

Fig. 59: 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 on the robot controller) 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

8.12.9 Reading the current path velocity

Description

The function block KRC_ReadActualVelocity reads the current actual velocity at the TCP of the robot. This is updated cyclically.

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

Fig. 60: Function block KRC_ReadActualVelocity

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Outputs

Parameter	Type	Description
Valid	BOOL	TRUE = data are valid
Value	REAL	Current path velocity ($VEL_ACT on the robot controller) Unit: m/s

8.12.10 Reading the current axis velocity

Description

The function block KRC_ReadActualAxisVelocity reads the current axis-specific velocity of the robot.

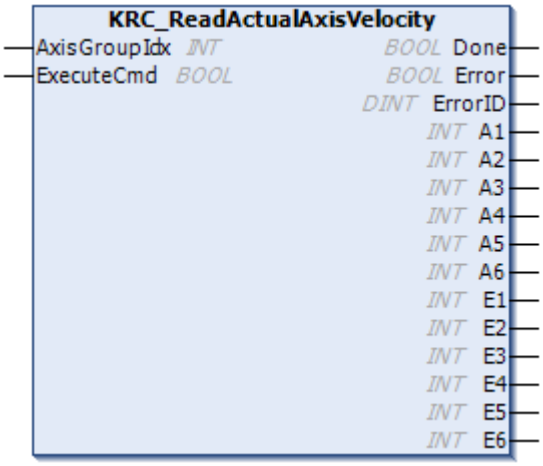


Fig. 61: Function block KRC_ReadActualAxisVelocity

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.

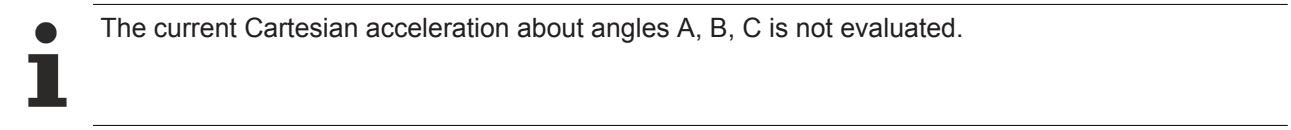
Outputs

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

8.12.11 Reading the current robot acceleration

Description

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



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

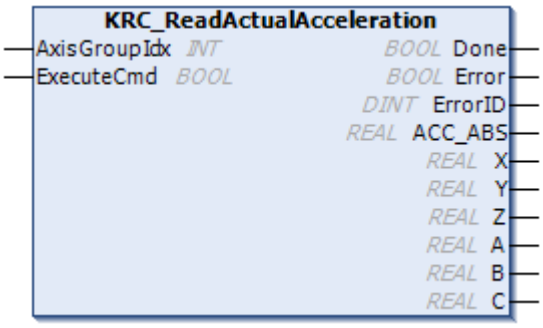


Fig. 62: Function block KRC_ReadActualAcceleration

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
Done	BOOL	TRUE = statement has been executed
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number
ACC_ABS	REAL	Current Cartesian acceleration relative to the absolute value of the overall acceleration ($ACC_CAR_ACT on the robot controller) Unit: m/s^2
X, Y, Z	REAL	Current Cartesian acceleration in the X, Y, Z directions Unit: m/s^2
A, B, C	REAL	Current Cartesian acceleration about angles A, B, C 0 m/s^2 (not calculated)

8.12.12 Reading a digital input

Description

The function block KRC_ReadDigitalInput polls and reads a digital input of the robot controller.

Fig. 63: 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 (corresponds to $IN[1 … 2048] on the robot controller) • **1 … 2048**

Outputs

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

8.12.13 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.

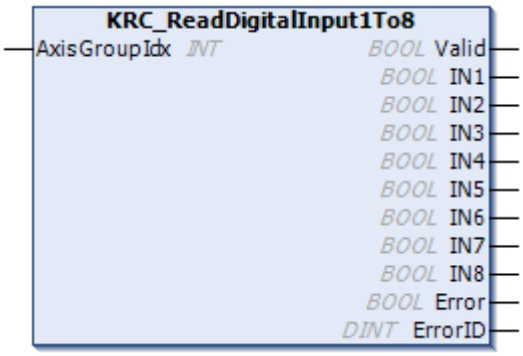


Fig. 64: Function block KRC_ReadDigitalInput1To8

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]

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

8.12.14 Reading multiple digital inputs

Description

The function block KRC_ReadDigitalInputArray polls and reads multiple digital inputs of the robot controller.

Fig. 65: 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 1st digital input polled (corresponds to $IN[1 … 2048] on the robot controller) • **1 … 2048**
Length	INT	Number of inputs that are polled • **1 … 200** **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

8.12.15 Reading a digital output

Description

The function block KRC_ReadDigitalOutput polls and reads a digital output of the robot controller.

Fig. 66: 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 (corresponds to $OUT[1 … 2048] on the robot controller) • **1 … 2048**

Outputs

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

8.12.16 Writing a digital output

Description

The function block KRC_WriteDigitalOutput writes a digital output or a pulse output on the robot controller.

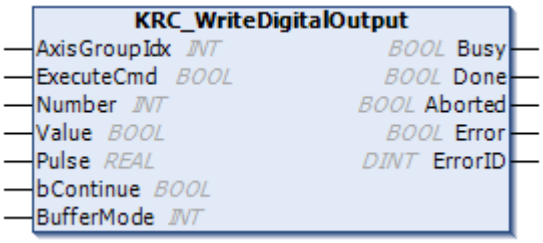


Fig. 67: Function block KRC_WriteDigitalOutput

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 (corresponds to $OUT[1 … 2048] on the robot controller) • **1 … 2048** **Note**: It must be ensured that no outputs are used that are already assigned by the robot system. Example: $OUT[1025] is always TRUE.
Value	BOOL	Value of the digital output
Pulse	REAL	Length of the pulse • **0.0 s** No pulse active • **0.1 … 3.0 s** Pulse interval = 0.1 s

Parameter	Type	Description
bContinue	BOOL	TRUE = output written in advance run **Note**: The robot controller executes programs with an advance run and a main run. Further information about the advance run and main run is contained in the operating and programming instructions for the KUKA System Software (KSS).
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.17 Writing digital outputs 1 to 8

Description

The function block KRC_WriteDigitalOutput1To8 writes the digital outputs 1 to 8 on the robot controller. The outputs are written cyclically.

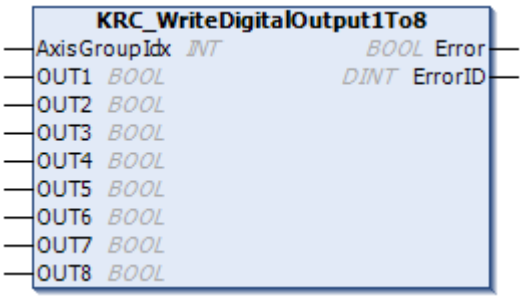


Fig. 68: Function block KRC_WriteDigitalOutput1To8

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**
OUT1 … OUT8	BOOL	Setpoint value of the output $OUT[1] … $OUT[8]

Outputs

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

8.12.18 Reading an analog input

Description

The function block KRC_ReadAnalogInput polls and reads an analog input of the robot controller.

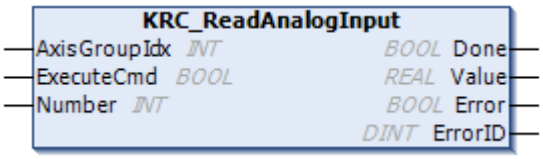


Fig. 69: Function block KRC_ReadAnalogInput

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 input ($ANIN[1 … 32] on the robot controller) • **1 … 32**

Outputs

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

8.12.19 Reading an analog output

Description

The function block KRC_ReadAnalogOutput polls and reads an analog output of the robot controller.

Fig. 70: 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 ($ANOUT[1 … 32] on the robot controller) • **1 … 32**

Outputs

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

8.12.20 Writing an analog output

Description

The function block KRC_WriteAnalogOutput polls and writes an analog output of the robot controller.

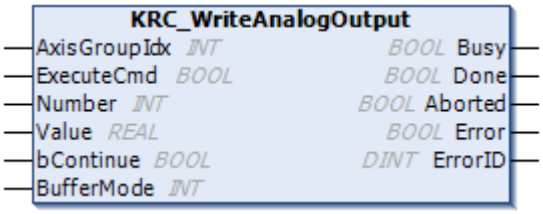


Fig. 71: Function block KRC_WriteAnalogOutput

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 ($ANOUT[1 … 32] on the robot controller) • **1 … 32**
Value	REAL	Value of the analog output
bContinue	BOOL	TRUE = output written in advance run **Note**: The robot controller executes programs with an advance run and a main run. Further information about the advance run and main run is contained in the operating and programming instructions for the KUKA System Software (KSS).
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.21 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.

Further information about tool and base in the robot system is contained in the operating and programming instructions for the KUKA System Software (KSS).

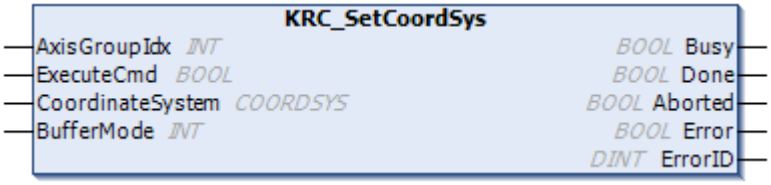


Fig. 72: Function block KRC_SetCoordSys

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.
CoordinateSystem	COORDSYS	Coordinate system to which the specified values refer (>>> COORDSYS [▶ 19])
BufferMode	INT	Mode in which the statement is executed • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.22 Reading TOOL data

Description

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

Fig. 73: Function block KRC_ReadToolData

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.
ToolNo	INT	Number of the TOOL coordinate system • **1 … 16**: TOOL_DATA[1 … 16]

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed
ToolData	FRAME	The data structure FRAME contains the following TOOL data: • 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 [▶ 20])
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.23 Writing TOOL data

Description

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

Fig. 74: Function block KRC_WriteToolData

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.

Parameter	Type	Description
ToolData	FRAME	The data structure FRAME contains the following TOOL data: • 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 [▶ 20])
ToolNo	INT	Number of the TOOL coordinate system • **1 … 16**: TOOL_DATA[1 … 16]
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.24 Reading BASE data

Description

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

Fig. 75: Function block KRC_ReadBaseData

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.
BaseNo	INT	Number of the BASE coordinate system • **1 … 32**: BASE_DATA[1 … 32]

Outputs

Parameter	Type	Description
Done	BOOL	TRUE = statement has been executed

Parameter	Type	Description
BaseData	FRAME	The data structure FRAME contains the following BASE data: • 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 [▶ 20])
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.25 Writing BASE data

Description

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

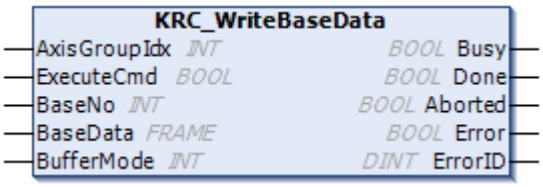


Fig. 76: Function block KRC_WriteBaseData

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.
BaseNo	INT	Number of the BASE coordinate system • **1 … 32**: BASE_DATA[1 … 32]
BaseData	FRAME	The data structure FRAME contains the following BASE data: • 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 [▶ 20])
BufferMode	INT	Mode in which the statement is executed • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block

Parameter	Type	Description
ErrorID	DINT	Error number

8.12.26 Reading the load data

Description

The function block KRC_ReadLoadData reads the load data of the robot (payload data or supplementary load data). Each tool of the robot controller has its own tool load data. The supplementary load data are valid for the entire robot.

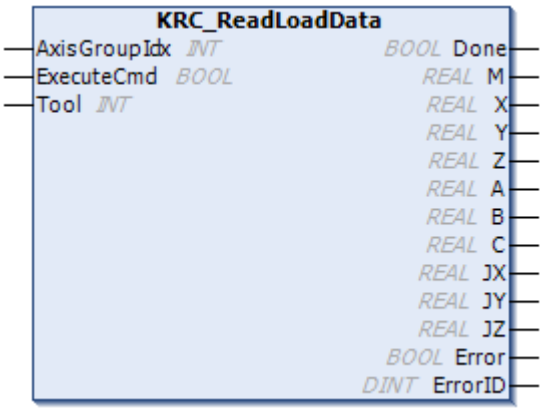


Fig. 77: Function block KRC_ReadLoadData

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.
Tool	INT	Number of the TOOL coordinate system for reading the payload data or number for reading the supplementary load data • **1 … 16**: TOOL_DATA[1 … 16] • **-1**: Supplementary load A1 • **-2**: Supplementary load A2 • **-3**: Supplementary load A3

Outputs

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

8.12.27 Writing load data

Description

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

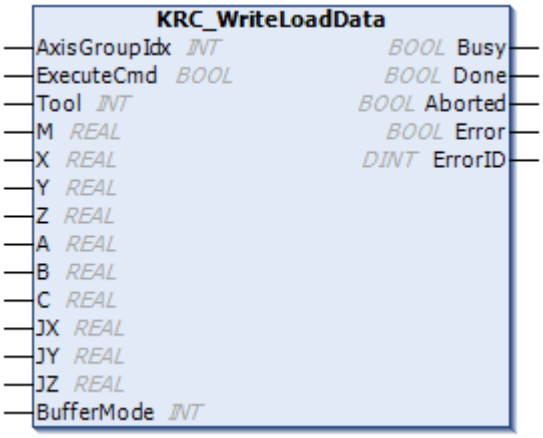


Fig. 78: Function block KRC_WriteLoadData

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.
Tool	INT	Number of the TOOL coordinate system for writing the payload data or number for writing the supplementary load data • **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 • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.28 Reading the software limit switches of the robot axes

Description

The function block KRC_ReadSoftEnd reads the software limit switches of the robot axes.

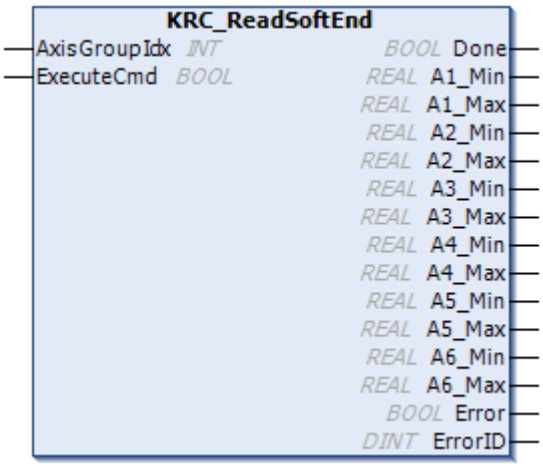


Fig. 79: Function block KRC_ReadSoftEnd

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

8.12.29 Reading the software limit switches of the external axes

Description

The function block KRC_ReadSoftEndExt reads the software limit switches of the external axes.

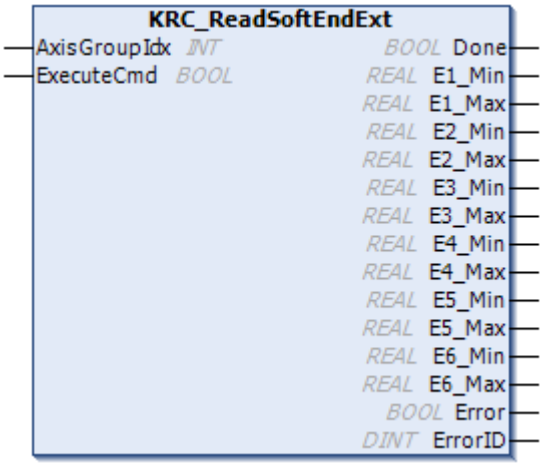


Fig. 80: Function block KRC_ReadSoftEndExt

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

8.12.30 Writing the software limit switches of the robot axes

Description

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

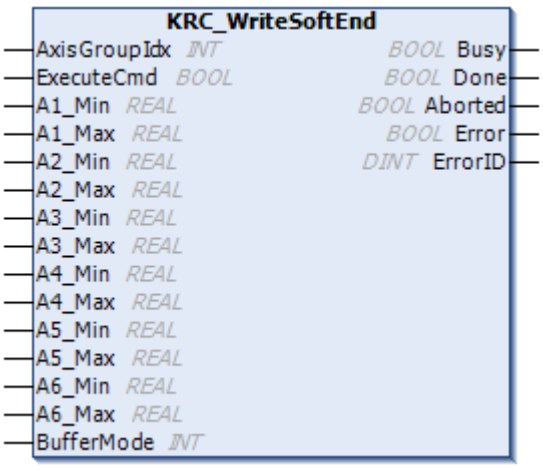


Fig. 81: Function block KRC_WriteSoftEnd

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.
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 • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

8.12.31 Writing the software limit switches of the external axes

Description

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

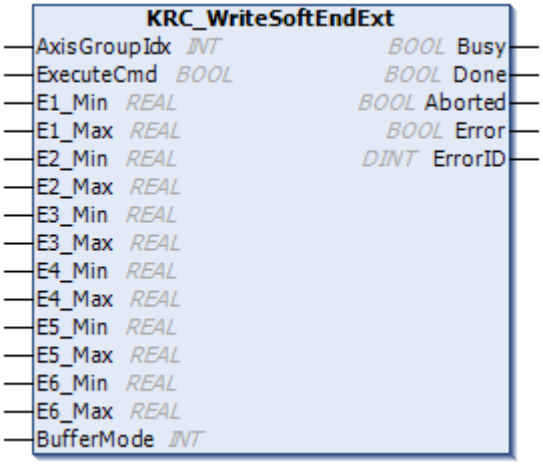


Fig. 82: Function block KRC_WriteSoftEndExt

Inputs

Parameter	Type	Description
AxisGroupIdx	INT	Index of axis group • **1 … 5**

Parameter	Type	Description
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 • **0**: DIRECT • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])

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
Error	BOOL	TRUE = error in function block
ErrorID	DINT	Error number

9 Messages

9.1 Error messages of the mxA interface in the robot interpreter

No.	Message text	Cause	Remedy
0	—	—	—
1	INTERNAL ERROR	Internal exceptional error	Contact service.
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. If this is not possible, contact service.
4	OVERFLOW STATUS RETURN QUEUE (TRIGGER)		
5	INVALID COMMAND QUEUE INDEX	Internal exceptional error	Contact service.
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 service.
30	INVALID PTP APO	An invalid approximation parameter has been transferred for a PTP motion.	Program a valid value (parameter **Approximate**). (>>> APO [▶ 18])
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 [▶ 19])
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 [▶ 19])
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 [▶ 18])
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 [▶ 15])
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 [▶ 19])
42	AXISPOSITION DATA NOT INITIALIZED	No axis position transferred when calling a KRC_MoveAxis function block.	Define at least 1 axis position (parameter **AxisPosition**). (>>> E6AXIS [▶ 19])

No.	Message text	Cause	Remedy
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**): • **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**): • **1 … 2048**
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**): • **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 [▶ 19])
47	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 … 2048**
48	INVALID INTERRUPT PRIORITY	An invalid number was transferred when calling a KRC_…Interrupt function block.	Program a valid value (parameter **Interrupt**): • **1 … 8**
49	INTERRUPT NOT DECLARED	Interrupt has not been declared.	Declare interrupt. (>>> Declaring interrupts [▶ 46])
50	INVALID INTERRUPT ACTION	The interrupt reaction programmed when the the interrupt was declared is invalid.	Program a valid reaction (parameter **Reaction**). (>>> Declaring interrupts [▶ 46])
51	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 … 2048**
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**): • **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. 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 [▶ 13])
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**). (>>> Reading the load data [▶ 104]) (>>> Writing load data [▶ 105])
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**

No.	Message text	Cause	Remedy
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 [▶ 19])
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 [▶ 15])
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**). (>>> Writing TOOL data [▶ 101]) (>>> Writing BASE data [▶ 103])
59	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> Writing load data [▶ 105])
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 service.
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**). (>>> Reading system variables [▶ 58]) (>>> Writing system variables [▶ 59])
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**). (>>> Writing system variables [▶ 59])
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.	
65	INVALID REAL VALUE	The programmed Real value is invalid.	Program a valid value: • **-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**): • **1 … 2048**
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 service.

No.	Message text	Cause	Remedy
69	INVALID TECH FUNCTION PARAMETER	An invalid value has been programmed in a KRC_TechFunction function block for a parameter.	Contact service.
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.

9.2 Error messages of the mxA interface in the submit interpreter

No.	Message text	Cause	Remedy
401	INTERNAL ERROR	Internal exceptional error	Contact service.
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	
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 service.
410	INVALID FIELDBUS TELEGRAMM LENGTH	Internal exceptional error	Contact service.
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: • EtherCAT configuration in WorkVisual or TwinCAT faulty	Check configuration in WorkVisual and TwinCAT and configure EtherCAT correctly.
		Error during operation: • Bit error during data transfer	Contact service.
413	INVALID MOVE TYPE	Internal exceptional error	Contact service.
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.

No.	Message text	Cause	Remedy
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.	
417	UNDERFLOW OF NIBBLE	Internal exceptional error	Contact service.
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 [▶ 18])
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 [▶ 19])
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 [▶ 19])
434	INVALID VELOCITY	An invalid value has been programmed in a function block for the velocity.	Program a valid value (parameter **Velocity**): • **0 … 100%**

No.	Message text	Cause	Remedy
435	INVALID ACCELERATION	An invalid value has been programmed in a function block for the acceleration.	Program a valid value (parameter **Acceleration**): • **0 … 100%**
436	INVALID C_PTP	An invalid approximation distance has been transferred for a PTP motion.	Program a valid value (parameter **Approximate**). (>>> APO [▶ 18])
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 [▶ 15])
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 [▶ 19])
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 [▶ 19])
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**): • **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**): • **1 … 2048**
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**): • **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 [▶ 19])
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 … 2048**
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. (>>> Declaring interrupts [▶ 46])

No.	Message text	Cause	Remedy
450	INVALID INTERRUPT ACTION	The interrupt reaction programmed when the the interrupt was declared is invalid.	Program a valid reaction (parameter **Reaction**). (>>> Declaring interrupts [▶ 46])
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 … 2048**
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) (>>> Writing a digital output [▶ 96])
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 [▶ 13])
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**). (>>> Reading the load data [▶ 104]) (>>> Writing load data [▶ 105])
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 [▶ 19])
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 [▶ 15])
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**). (>>> Writing TOOL data [▶ 101]) (>>> Writing BASE data [▶ 103])
459	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> Writing load data [▶ 105])
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 service.
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**). (>>> Reading system variables [▶ 58]) (>>> Writing system variables [▶ 59])

No.	Message text	Cause	Remedy
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**). (>>> Writing system variables [▶ 59])
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 … 2048**
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 service.
469	INVALID TECH FUNCTION PARAMETER	An invalid value has been programmed in a KRC_TechFunction function block for a parameter.	Contact service.
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.

9.3 Errors in the function block

No.	Message text	Cause	Remedy
501	INTERNAL ERROR	Internal exceptional error	Contact service.
502	INVALID BUFFER_MODE	BufferMode **0**: DIRECT is not permissible for this function block.	Program the correct mode: • **1**: ABORTING • **2**: BUFFERED (>>> BufferMode [▶ 13])
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. (>>> Initializing the mxA interface [▶ 85])
504	INVALID OVERRIDE	Invalid override value in the function block KRC_SetOverride	Program a valid value (parameter **Override**). • **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 GROUPREFIDX	The axis group index specified in the function block is invalid.	Specify a valid index for the axis group (parameter **AxisGroupIdx**).

No.	Message text	Cause	Remedy
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. (>>> Initializing the mxA interface [▶ 85])
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.
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.	
		Error during start-up: • EtherCAT configuration in WorkVisual or TwinCAT faulty	Check configuration in WorkVisual and TwinCAT and configure EtherCAT correctly.
		Error during operation: • Bit error during data transfer	Contact service.
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**): • **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 service.
518	KRC STOPMESS ACTIVE	Group error which prevents motion enable	Check how the error was triggered and eliminate the error. • 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 service.
520	VELOCITY CONFLICT	More than one value has been programmed for the velocity in a KRC_Move function block.	Contact service.

No.	Message text	Cause	Remedy
521	INVALID PARAMETER COUNT	An invalid value has been programmed in a KRC_TechFunction function block for the parameter **ParameterCount**.	Contact service.
522	INVALID PARAMETER USAGE	The parameter **ParameterCount** has been incorrectly configured in the KRC_TechFunction function block.	Contact service.
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.
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.	1. In the main menu, select Display > Inputs/outputs > Automatic External. 2. 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.	1. Wait until the robot program has been executed. 2. 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 service.
529	TIMEOUT SWITCHING DRIVES ON	Internal exceptional error	Contact service.
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 service.
535	INVALID $TOOL	Internal exceptional error	
536	INVALID SOFTEND	Error in the function block KRC_Forward: The specified axis angles lie outside of the software limit switches.	Enter axis angles that lie within the software limit switches (parameter **Axis_Values**). or: Modify the software limit switches.
537	ERR MATH TRAFO	Error in the function block KRC_Forward: The robot cannot reach the specified axis angles.	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: • 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.	• 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 service.
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 ComAcpt signal was set by the function block.	Acknowledge the message and in the future do not reset the Execute input until the ComAcpt 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.

9.4 ProConOS errors

No.	Message text	Cause	Remedy
701	INTERNAL ERROR	Internal exceptional error	Contact service.
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 service.
713	INVALID MOVE TYPE	An invalid value has been programmed in a KRC_Move function block for the parameter MoveType.	Contact service.
730	INVALID PTP APO	An invalid approximation parameter has been transferred for a PTP motion.	Program a valid value (parameter **Approximate**). (>>> APO [▶ 18])
731	INVALID CP APO	An invalid approximation parameter has been transferred for a CP motion (LIN, CIRC).	

No.	Message text	Cause	Remedy
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 [▶ 19])
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 [▶ 19])
734	INVALID VELOCITY	An invalid value has been programmed in a function block for the velocity.	Program a valid value (parameter **Velocity**): • **0 … 100%** • **0 … 2 m/s** (only with the MC_MoveLinear and MC_MoveCircular blocks)
735	INVALID ACCELERATION	An invalid value has been programmed in a function block for the acceleration.	Program a valid value (parameter **Acceleration**): • **0 … 100%** • 0 … 2.3 m/s^2 (only with the MC_MoveLinear and MC_MoveCircular blocks)
736	INVALID C_PTP	An invalid approximation distance has been transferred for a PTP motion.	Program a valid value (parameter **Approximate**). (>>> APO [▶ 18])
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 [▶ 15])
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 [▶ 19])

No.	Message text	Cause	Remedy
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 [▶ 19])
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**): • **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**): • **1 … 2048**
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**): • **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 [▶ 19])
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**): • **1 … 2048**
748	INVALID INTERRUPT PRIORITY	An invalid number was transferred when calling a KRC_…Interrupt function block.	Program a valid value (parameter **Interrupt**): • **1 … 8**
750	INVALID INTERRUPT ACTION	The interrupt reaction programmed when the the interrupt was declared is invalid.	Program a valid reaction (parameter **Reaction**). (>>> Declaring interrupts [▶ 46])
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 … 2048**
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 [▶ 13])
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**). (>>> Reading the load data [▶ 104]) (>>> Writing load data [▶ 105])
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**

No.	Message text	Cause	Remedy
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 [▶ 19])
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 [▶ 15])
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**). (>>> Writing TOOL data [▶ 101]) (>>> Writing BASE data [▶ 103])
759	INVALID LOAD DATA	Invalid load data have been programmed in a KRC_WriteLoadData function block.	Program valid data. (>>> Writing load data [▶ 105])
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.
765	INVALID REAL VALUE	The programmed Real value is invalid.	• **-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**): • **1 … 50**
772	INVALID MAXWAITTIME	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**): • **1 … 32 767 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**): • **0 … 2**
774	INVALID DI STARTNUMBER	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**): • **1 … 2048**
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**): • **1 … 200**
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**): • **1 … 3**

No.	Message text	Cause	Remedy
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**): • 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**): • In the case of a linear conveyor: Specification in millimeters • In the case of a circular conveyor: Specification in degrees
779	INVALID CONVEYOR PIECENUMBER	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: • **1**: Every workpiece is picked up. • **3**: Every 3rd workpiece is picked up.
780	INVALID WORKSPACENO	In the called function block, an invalid number has been programmed for the number of the workspace.	Program a valid value (parameter **WorkspaceNo**): • **1 … 8**
781	INVALID WORKSPACEMODE	In the called function block, an invalid number has been programmed for the mode for workspaces.	Program a valid value (parameter **WorkspaceMode**): • **0 … 4**
782	INVALID WORKSPACEPART	Internal exceptional error	Contact service.
801	STOPMESS ACTIVE	Group error which prevents motion enable	Check how the error was triggered and eliminate the error. • 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.