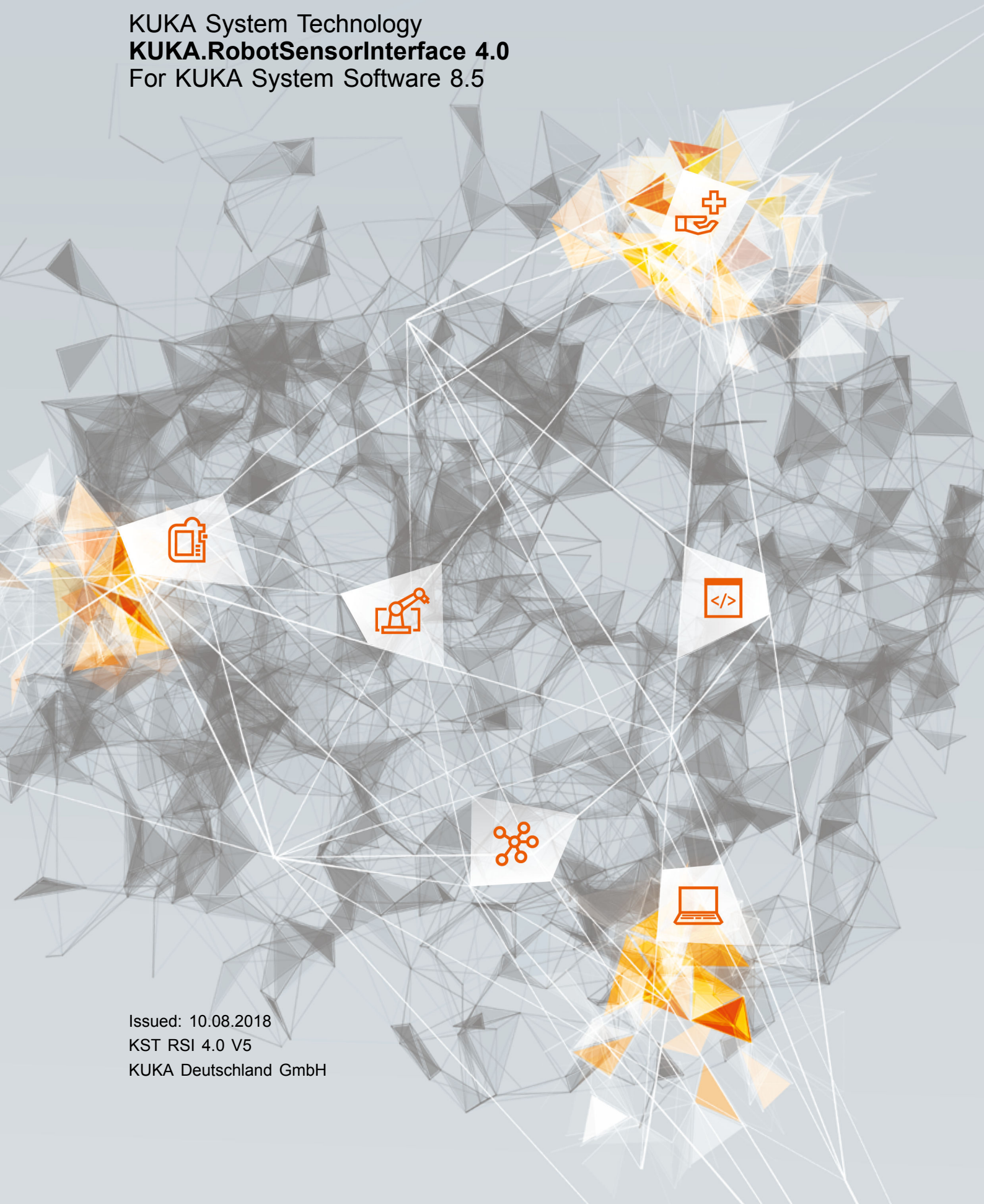
KUKA System Technology

KUKA.RobotSensorInterface 4.0

For KUKA System Software 8.5



Issued: 10.08.2018
KST RSI 4.0 V5
KUKA Deutschland GmbH

KUKA Deutschland GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

KIM-PS5-DOC

Translation of the original documentation

Publication:	Pub KST RSI 4.0 (PDF) en
	PB8012
Book structure:	KST RSI 4.0 V5.1
	BS7477
Version:	KST RSI 4.0 V5

Contents

1 Introduction

1.1 Target group

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

- Advanced KRL programming skills
- Advanced knowledge of the robot controller system
- Advanced knowledge of bus systems
- Basic knowledge of XML
- Basic knowledge of digital technology

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

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

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

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

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

DANGER

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

WARNING

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

CAUTION

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

NOTICE

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

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

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

SAFETY INSTRUCTION
The following procedure must be followed exactly!

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

Notices

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

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

1.4 Terms used

RSI terms

Term	Description
RSI	RobotSensorInterface Interface for communication between the industrial robot and a sensor system.
RSI container	An RSI container contains the RSI context created with RSIVisual and must be created in the KRL program.
RSI container ID	Identifier that is automatically assigned when the RSI container is created in the KRL program.
RSI context	The RSI context is the signal processing configured with RSIVisual. The RSI context consists of RSI objects and links between the RSI objects.
RSI monitor	Monitor for online visualization of RSI signals
RSI object	The RSI context is configured using RSI objects that are linked by means of object-specific signal inputs/outputs.
RSI object library	Library containing all RSI objects that are available for the creation of an RSI context
RSI object parameter	The RSI object parameters influence the functionality of an RSI object. The number of RSI object parameters is specific for each RSI object.
RSIVisual	Graphical editor for configuration of the signal processing (RSI context)

General terms

Term	Description
CCS	Correction Coordinate System Correction coordinate system in the TCP for Cartesian sensor correction.
Ethernet	Ethernet is a data network technology for local area networks (LANs). It allows data to be exchanged between the connected devices in the form of data frames.
KLI	KUKA Line Interface Line bus for the integration of the system in the customer network
KR C	KUKA Robot Controller
KUKA smartHMI	KUKA smart human-machine interface User interface of the KUKA System Software
Sensor mode	Signal processing mode • #IPO: signal processing at sensor cycle rate of 12 ms • #IPO_FAST: signal processing at sensor cycle rate of 4 ms
Sensor cycle rate	Cycle rate at which the signal processing is calculated. Depending on the mode, the sensor cycle rate is 12 ms (sensor mode #IPO) or 4 ms (sensor mode #IPO_FAST).
TTS	Tool-based technological system The TTS is a coordinate system that moves along the path with the robot. It is calculated every time a LIN or CIRC motion is executed. It is derived from the path tangent, the +X axis of the TOOL coordinate system and the resulting normal vector. The tool-based moving frame coordinate system is defined as follows: $\mathbf{X_{TTS}}$: path tangent $\mathbf{Y_{TTS}}$: normal vector to the plane derived from the path tangent and the +X axis of the TOOL coordinate system $\mathbf{Z_{TTS}}$: vector of the right-angled system derived from $\mathbf{X_{TTS}}$ and $\mathbf{Y_{TTS}}$ The path tangent and the +X axis of the TOOL coordinate system must not be parallel, otherwise the TTS cannot be calculated.
UDP	User Datagram Protocol Connectionless protocol for the data exchange between the devices of a network
IP	Internet Protocol

Term	Description
	The Internet Protocol is used to define subnetworks by means of physical MAC addresses.
XML	Extensible Markup Language Standard for creating machine-readable and human-readable documents in the form of a specified tree structure.

1.5 Trademarks

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

.NET Framework is a trademark of Microsoft Corporation.

Windows is a trademark of Microsoft Corporation.

1.6 Licenses

The KUKA license conditions and the license conditions of the open-source software used can be found in the following folders:

- Under .\LICENSE on the data storage medium with the installation files of the KUKA software
- Under D:\KUKA_OPT*Option package name*\LICENSE after installation on the robot controller
- In the license folder under the name of the option package in the **Options** catalog after installation in WorkVisual

Further information about open-source licenses can be requested from the following address: opensource@kuka.com

2 Product description

2.1 Overview of RobotSensorInterface

Functions

RobotSensorInterface is an add-on option package with the following functions:

- Data exchange between robot controller and sensor system.
- Cyclical signal processing and evaluation at the sensor cycle rate.
- Influence on the robot motion or program execution by processing sensor signals.
- Creation of RSI contexts in WorkVisual with the graphical editor RSI Visual
- Toolbox with over 75 RSI objects for RSIVisual
- Creation, exporting and importing of user-defined toolboxes
- Creation of function blocks for user-defined toolboxes
- Programming of function blocks in dedicated function block editor
- Use of function blocks in RSI contexts
- Online visualization of the RSI signals with the RSI monitor

RoboTeam

RobotSensorInterface can be operated together with RoboTeam, but certain limitations must be observed (>>> *2.4.4 "Sensor correction in RoboTeam operation" Page 20*).

Communication

The robot controller and sensor system can communicate with one another and exchange information in different ways:

- Via the I/O system of RSI

  The data and signals of the sensor system are read and written via the I/O system of RSI.

  The RSI inputs/outputs can be mapped to the signals of a field bus device or another I/O system.

Further information about bus management and I/O mapping can be found in the WorkVisual documentation.

Further information about bus configuration can be found in the bus system documentation.

- Via the I/O system of the robot controller

  The data and signals of the sensor system are read and written via the I/O system of the robot controller. RobotSensorInterface accesses the data and signals and processes them.

  The signals are mapped via a bus system to the I/O system of the robot controller.

- Via Ethernet
  - The robot controller communicates with the sensor system via a real-time-capable network connection. The data are transmitted via the Ethernet UDP/IP protocol. No fixed data frame is specified. The user must configure the data set in an XML file.

Characteristics:

- Cyclical data transmission from the robot controller to a sensor system parallel to program execution (e.g. position data, axis angles, operating mode, etc.)
- Cyclical data transmission from a sensor system to the robot controller parallel to program execution (e.g. sensor data)

Ethernet UDP/IP

The following must be taken into account for data exchange via Ethernet UDP/IP protocol:

- UDP is a connectionless network protocol for exchanging data packets. Data exchange via UDP is not reliable and not secure. It cannot be guaranteed, for example, that sent packets arrive reliably or in the order in which they were sent.
- Data exchange via Ethernet is appropriate for applications that do not react adversely to packets being lost or received in the incorrect order. If an application cannot tolerate this, the programmer must take appropriate measures (for example, checking in the program to determine whether all packets have been received and, if necessary, requesting the packets again).

2.2 Functional principle of signal processing

Description

Signal processing is established using RSI objects. An RSI object performs a specific function with its signal inputs and makes the result available at the signal outputs.

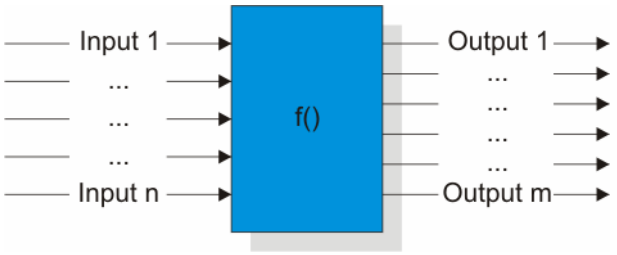


Fig. 2-1: Schematic structure of an RSI object

RobotSensorInterface provides the user with an extensive range of RSI objects in a library. The linking of the signal inputs and outputs of multiple RSI objects creates a signal flow. The overall signal flow is called the RSI context.

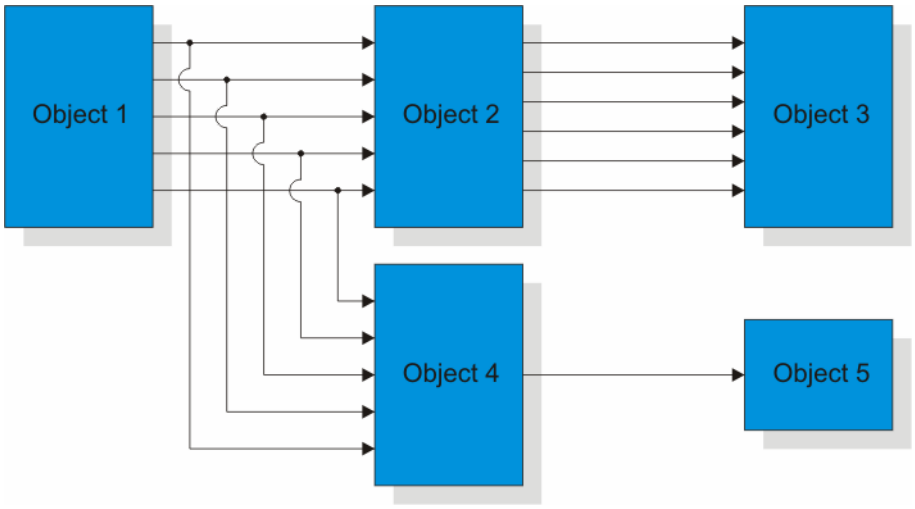


Fig. 2-2: Schematic structure of an RSI context

The RSI context is defined and saved with the graphical editor RSIVisual. In the KRL program, the RSI context can be loaded and the signal processing parallel to program execution can be activated and deactivated. The signal processing is calculated at the sensor cycle rate. Depending on the mode, the sensor cycle rate is 12 ms (sensor mode #IPO) or 4 ms (sensor mode #IPO_FAST).

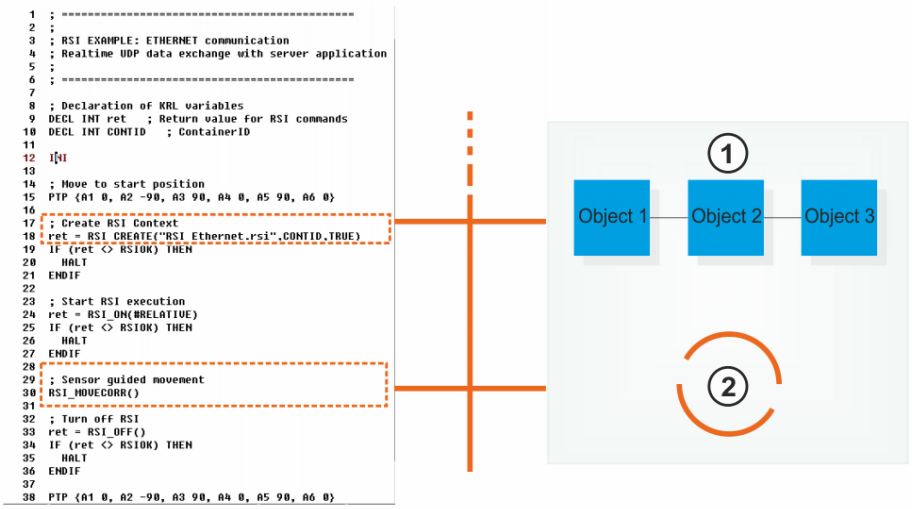


Fig. 2-3: Interaction between KRL program and signal processing

1 RSI context
2 Sensor cycle rate

2.3 Functional principle of data exchange

2.3.1 Data exchange via the I/O system of RSI

RSI provides a separate I/O system for data exchange. The I/O system of RSI is independent of the I/O system of the robot controller and can communicate with field bus devices as well as with other I/O systems.

RSI has 8192 bits for inputs/outputs which can be mapped to field bus devices or I/O systems. Access to the mapped signals is via the RSI context.

The following RSI objects are used for this:

- INPUT has read access to the digital RSI inputs. The output signals of a field bus device or another I/O system can be mapped to RSI inputs via WorkVisual.
- OUTPUT writes to the digital RSI outputs. The RSI outputs can be mapped to the inputs of a field bus device or another I/O system via WorkVisual.

The mapping of RSI outputs to RSI inputs is not permitted and is therefore disabled.

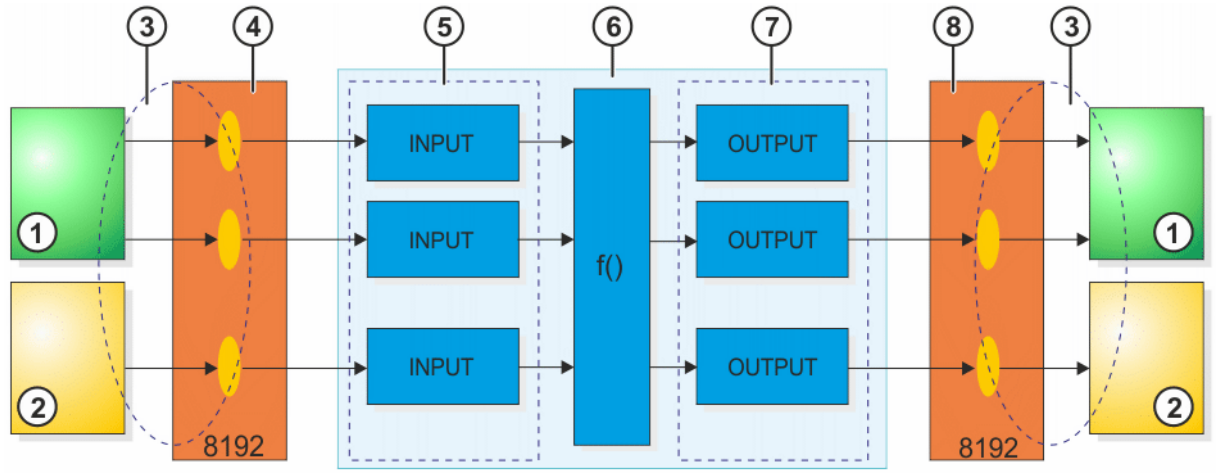


Fig. 2-4: Data exchange via the I/O system of RSI

1 Field bus device with its I/Os (e.g. sensor on PROFINET)
2 Field bus device with its I/Os (e.g. sensor on EtherCAT)
3 I/O mapping via WorkVisual
4 Inputs of the RSI I/O system
5 INPUT objects for accessing RSI inputs
6 RSI context for signal processing
7 OUTPUT objects for accessing RSI outputs
8 Outputs of the RSI I/O system

2.3.2 Data exchange via the I/O system of the robot controller

The data and signals of the sensor system are read via the I/O system ($IN, $ANIN) of the robot controller. The processed signals are returned to the sensor system via the I/O system ($OUT, $ANOUT). The signals are read and written at the sensor cycle rate.

The following RSI objects are used:

- ANIN and DIGIN have read access to the I/O system of the robot controller and transfer the data and signals from the sensor system to the signal processing.
- MAP2ANOUT and MAP2DIGOUT access the processed signals and write them to the I/O system of the robot controller.

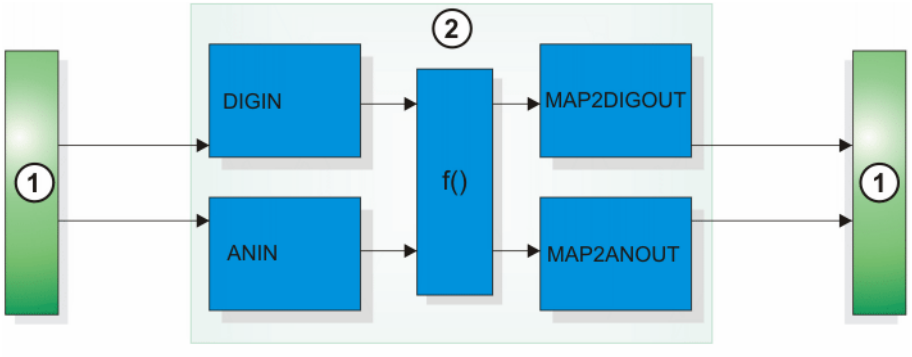


Fig. 2-5: Data exchange via the I/O system of the robot controller

1 I/O system of the robot controller
2 RSI context

2.3.3 Data exchange via Ethernet

Description

Data exchange via Ethernet is implemented using the RSI object ETHERNET.

Up to 64 inputs and outputs can be defined for ETHERNET. The signals at the inputs are sent to the sensor system. The data received from the sensor system are available at the outputs.

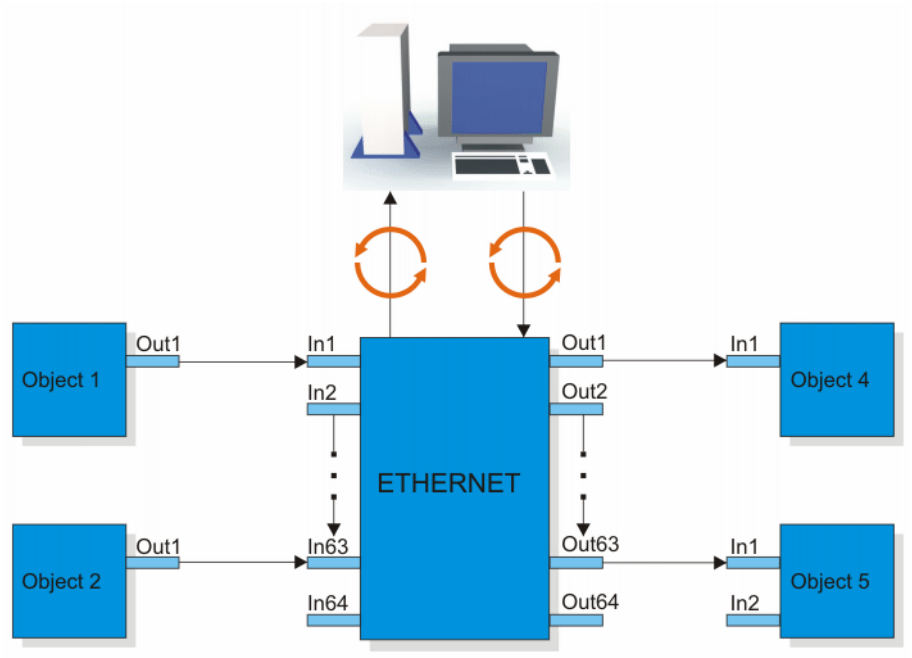


Fig. 2-6: Data exchange via Ethernet (functional principle)

When signal processing is activated, a channel is prepared for sending data to the sensor system via the UDP/IP protocol. The robot controller initiates the data exchange with a data packet and transfers further data packets to the sensor system at the sensor cycle rate. The sensor system must respond to the data packets received with a data packet of its own.

With signal processing activated, ETHERNET sends and receives a user-defined data set in XML format at the sensor cycle rate. This data set

must be configured in an XML file. The name of the XML file is specified in the ETHERNET object.

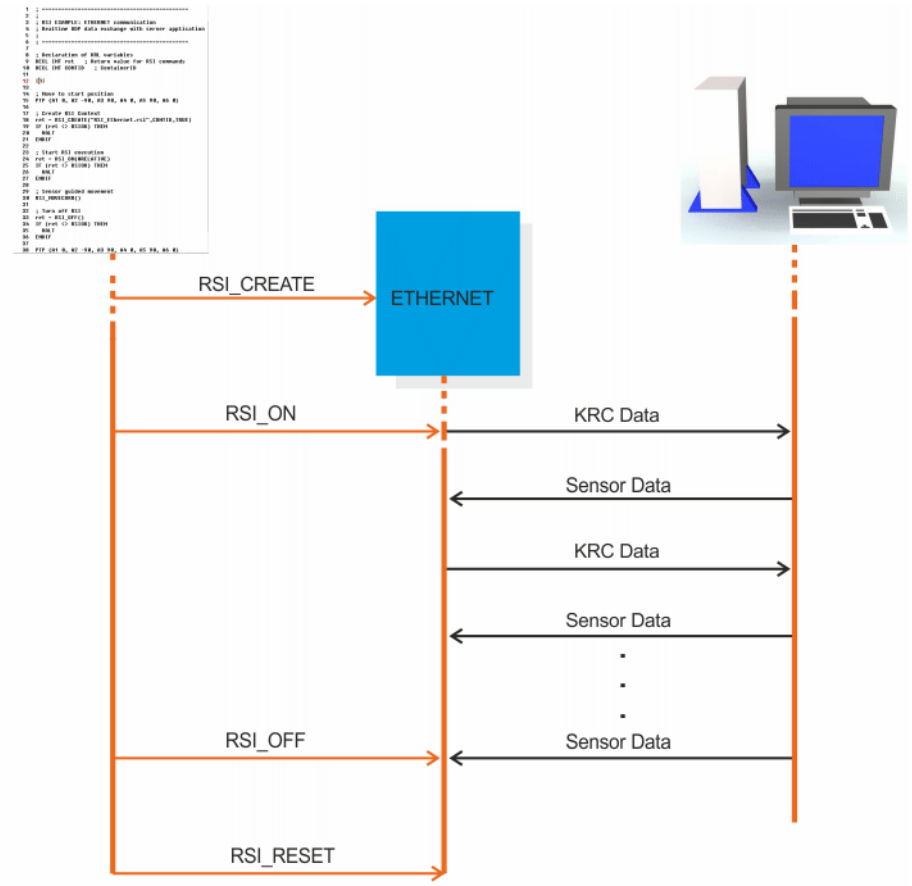


Fig. 2-7: Data exchange via Ethernet (sequence)

Real-time request

A data packet received by the sensor system must be answered within the sensor cycle rate. Packets that arrive too late are rejected.

When the maximum number of data packets for which a response has been sent too late has been exceeded, the robot stops. If signal processing is deactivated, data exchange also stops.

2.4 Functional principle of sensor correction

Description

RobotSensorInterface allows continual influence over the robot motion by means of sensor data. A correction value to the current setpoint position is calculated at the sensor cycle rate. The resulting deviation from the setpoint position is the sensor correction.

Sensor correction cannot be used for asynchronous axes.

Overview

Correction options	Description
Correction type	The correction types specify whether the sensor correction is to be Cartesian or axis-specific.

Correction options	Description
	(>>> *2.4.1 "Correction type" Page 16*)
Correction mode	The correction mode specifies whether the sensor correction is to be absolute or relative. (>>> *2.4.2 "Correction mode" Page 18*)
Correction method	The correction method specifies whether the sensor correction is purely sensor-guided or superposed on a planned path. (>>> *2.4.3 "Correction method" Page 19*)

NOTICE

Sensor corrections influence the robot motion directly. It is not the industrial robot that specifies the path, but the sensor. The user is responsible for ensuring that the correction specification signals of the sensor are prepared in such a way that no mechanical damage can occur to the robot system, e.g. as a result of vibrations.

2.4.1 Correction type

Description

The correction type specifies whether the sensor correction is to be Cartesian or axis-specific. Various different RSI objects are available for this:

- Axis angle correction
- Cartesian correction

Axis angle correction

A correction value can be applied on an axis-specific basis to robot axes A1 … A6 and external axes E1 … E6.

RSI objects used:

- AxisCorr (correction of robot axes)
- AxisCorrExt (correction of external axes)

The maximum permissible correction is limited in both directions.

Cartesian correction

A correction value (frame) can shift the robot position with a Cartesian motion. The correction frame is relative to the correction coordinate system (CCS) in the TCP.

The following reference coordinate systems are available for the orientation of the correction coordinate system:

- BASE coordinate system
- ROBROOT coordinate system
- TOOL coordinate system
- WORLD coordinate system
- Tool-based technological system (TTS)

RSI object used:

- PosCorr

The maximum permissible Cartesian correction is limited.

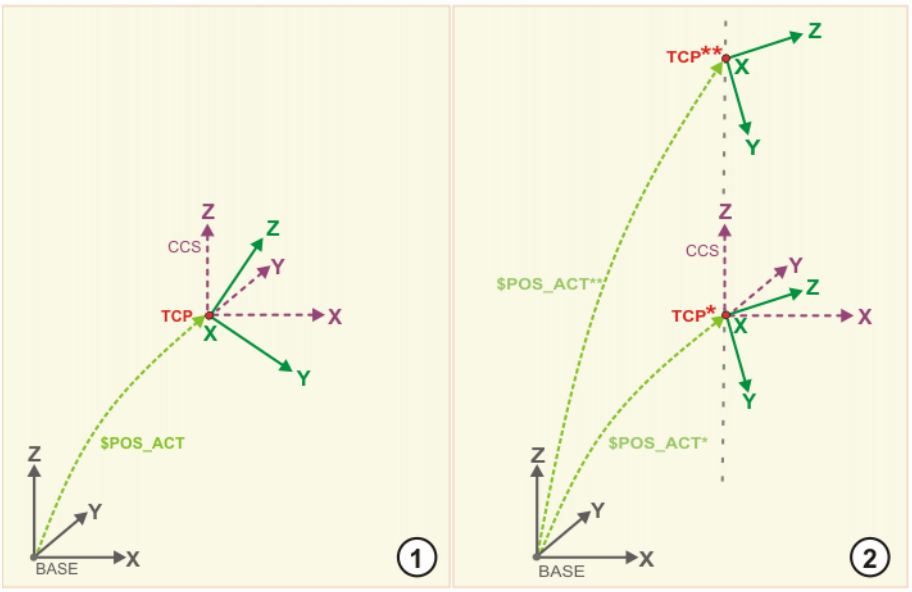


Fig. 2-8: Cartesian correction relative to BASE

Item	Description
1	Starting position for the Cartesian correction. • $POS_ACT: Cartesian robot position • CCS: correction coordinate system in the TCP with the orientation of BASE
2	Cartesian correction – correction coordinate system is the BASE coordinate system. • $POS_ACT*: Cartesian robot position rotated by the correction value – TCP*: the TCP is rotated about +B in the correction coordinate system. • $POS_ACT**: Cartesian robot position offset and rotated by the correction value – TCP**: the TCP is offset in the +Z direction and rotated about +B in the correction coordinate system.

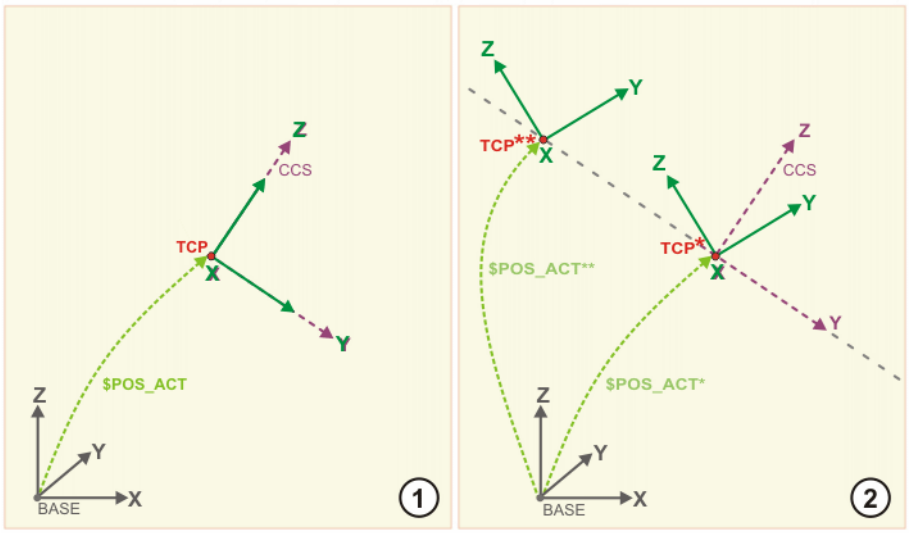


Fig. 2-9: Cartesian correction relative to TOOL

Item	Description
1	Starting position for the Cartesian correction. • $POS_ACT: Cartesian robot position • CCS: correction coordinate system in the TCP with the orientation of TOOL
2	Cartesian correction – correction coordinate system is the TOOL coordinate system • $POS_ACT*: Cartesian robot position rotated by the correction value – TCP*: the TCP is rotated about +C in the correction coordinate system. • $POS_ACT**: Cartesian robot position offset and rotated by the correction value – TCP**: TCP is offset in the -Y direction and rotated about +C in the correction coordinate system.

2.4.2 Correction mode

Description

The correction mode specifies whether the sensor correction is to be absolute or relative. It is selected using a parameter in the command RSI_ON().

Absolute correction

The new position results from the offset of the starting position by the current correction value.

Relative correction

The correction values are added together. The new position results from the offset of the starting position by the previous correction and the current correction value.

2.4.3 Correction method

Description

The correction method specifies whether the sensor correction is purely sensor-guided or superposed on a planned path. If the command RSI_ON() is followed by a motion command, the sensor correction is superposed on the programmed motion. If a purely sensor-guided sensor correction of the robot is to be carried out, the command RSI_MOVECORR() is to be used.

Superposed sensor correction

The correction values are applied to the control points of a programmed path. The path can be corrected on the basis of absolute or relative correction data.

If the signals are processed in IPO mode, the path can only be corrected using LIN and CIRC motions.

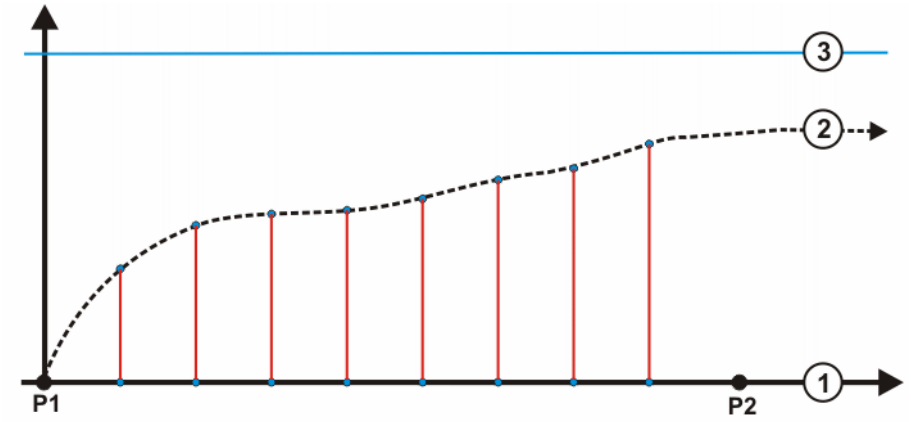


Fig. 2-10: Path correction based on absolute values

1	Programmed path
2	Corrected path
3	Maximum overall correction
Red	Absolute correction value

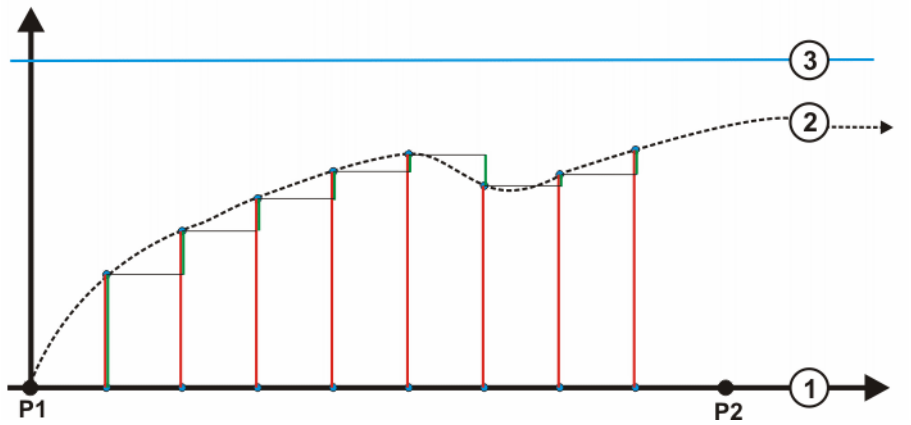


Fig. 2-11: Path correction based on relative values

1	Programmed path
2	Corrected path

3	Maximum overall correction
Red	Overall correction
Green	Relative correction value

Sensor-guided motion

A sensor-guided motion can be programmed using the command RSI_MOVECORR(). Moving away from a start point, the robot does not head for a defined end point, but is controlled purely by means of corrections on the basis of sensor data.

The sensor-guided motion can be executed on the basis of absolute or relative correction data.

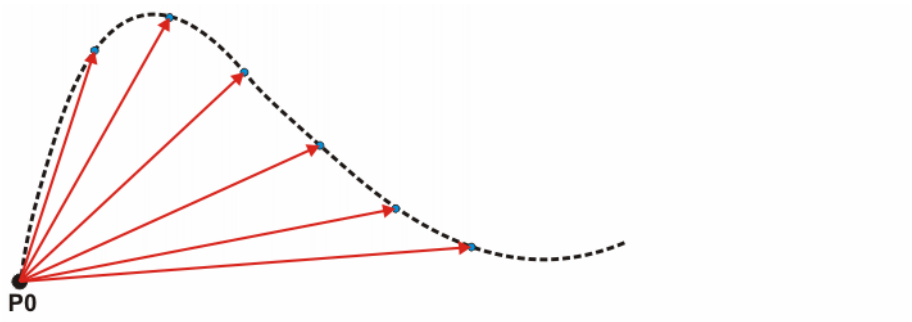


Fig. 2-12: Sensor-guided motion based on absolute values

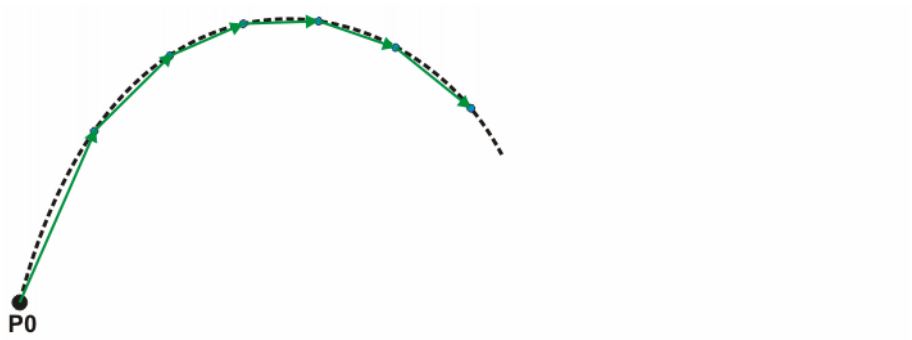


Fig. 2-13: Sensor-guided motion based on relative values

The motion characteristics of the robot during a sensor-guided motion differ between KR C2 and KR C4. Contact KUKA Support if the corresponding systems are to be converted from KR C2 to KR C4.

2.4.4 Sensor correction in RoboTeam operation

RobotSensorInterface can be operated together with RoboTeam, but with certain limitations.

- Sensor correction of a master kinematic system ($LK_Master[]) is possible in sensor modes #IPO and #IPO_FAST.
- Sensor correction of a slave kinematic system ($LK_Slave[]) is only possible in sensor mode #IPO.
- A sensor correction in sensor mode #IPO is only possible for CP motions. If a sensor correction for Spline and PTP commands is required, the sensor mode #IPO_FAST must be used.
- Sensor correction of the slave kinematic system is not possible in sensor mode #IPO_FAST. If an attempt is made to operate the slave kin-

ematic system in sensor mode #IPO_FAST, the following message is displayed:

  – *Active sensors not possible during {Incompatible functionality}*. In this case, the *{Incompatible functionality}* is the function *LK()*.

- Under no circumstances does the sensor correction of a slave kinematic system have any effect on the motion of the master kinematic system.

The sensor correction of the master kinematic system influences the motion of the slave kinematic system in accordance with the sensor mode that is active on the master controller.

	LK master without sensors	**LK master in sensor mode #IPO**	**LK master in sensor mode #IPO_FAST**
LK slave without sensors	(Normal RoboTeam application without signal processing by RSI)	• The LK slave follows the LK base (planned path) and the additional sensor correction of the LK master.	• The LK slave only follows the LK base (planned path) of the LK master. • The LK slave does not follow the additional sensor correction of the LK master.
LK slave in sensor mode #IPO	• The LK slave (planned path) follows the LK base of the LK master. • A sensor correction of the LK slave is not possible.	• The LK slave follows the LK base (planned path) and the additional sensor correction of the LK master. • A sensor correction of the LK slave is possible and is superposed on that of the LK master.	• The LK slave follows the LK base (planned path) of the LK master. • The LK slave does not follow the additional sensor correction of the LK master. • A sensor correction of the LK slave is not possible.

2.5 Intended use

Use

RobotSensorInterface is used as a universal interface for the implementation of industrial robotic applications in the field of industry and research. The objective is to expand the basic functionality of the system software in order to enable applications that require real-time processing of data from external systems and continual influence over the robot motion. RSI may be used for the following operations:

- Cyclical data communication between an external system and the signal processing of RSI during the program run time of the robot interpreter.

  The cyclical exchange is carried out at a defined interval.
- Preparation of data by the signal processing of RSI.
- Influence on the robot motion or program execution using the sensor data prepared by the signal processing.

The following points must be observed when using the option package:

- The option package must only be operated in compliance with the specified system requirements.
- Sensor corrections influence the robot motion directly. The industrial robot motion resulting from the usual path planning is modified depending on the measured sensor values during the motion.

  The user is responsible for ensuring that the correction specification signals of the sensor are prepared in such a way that no mechanical damage can occur to the robot system, e.g. as a result of vibrations.
- The option package can be used for applications in the field of human-robot collaboration (HRC).

  Special safety standards apply for HRC applications. The user is responsible for ensuring that these safety standards are met.
- Communication via Ethernet is not a complete substitute for a field bus system.

Operation in accordance with the intended use also requires compliance with the start-up and configuration instructions in this documentation and constant observance of the assembly and operating instructions for the cell components used.

Misuse

Any use or application deviating from the intended use is deemed to be misuse and is not allowed. The manufacturer cannot be held liable for any damage resulting from such use. The risk lies entirely with the user.

3 Safety

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

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

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

3.1 Safety instructions

Sensor-assisted operation

- Incorrect use of RobotSensorInterface can cause personal injury and material damage.
- In sensor-assisted operation, the robot may move unexpectedly in the following cases:
  - Incorrectly parameterized RSI objects
  - Hardware fault (e.g. incorrect cabling, break in the sensor cable or sensor malfunction)
- Unexpected movements may cause serious injuries and substantial material damage. The system integrator is obliged to minimize the risk of injury to himself/herself and other people, as well as the risk of material damage, by adopting suitable safety measures, e.g. by means of workspace limitation.
- At the start of signal processing with RobotSensorInterface, the safety controller generates an acknowledgement message in T1 or T2 mode:

  Caution – sensor correction is activated!!!

Workspace limitation

- The axis ranges of all robot axes are limited by means of adjustable software limit switches. These software limit switches must be set in such a way that the workspace of the robot is limited to the minimum range required for the process.
- The System Software allows the configuration of a maximum of 8 Cartesian and 8 axis-specific workspaces. The system integrator must configure the workspaces in such a way that they are limited to the minimum range required for the process. This reduces the risk of damage caused by unexpected movements in sensor-assisted operation to a minimum.

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

Sensor correction

RobotSensorInterface monitors and limits the maximum sensor correction. Each individual correction object can be monitored, as can the overall correction of all correction objects.

Object-specific sensor corrections are limited by default to max. ± 5 mm or 5°; the overall correction is limited to max. ± 6 mm or 6°.

If an object-specific correction is exceeded, signal processing continues and the correction is automatically limited to the maximum permissible correction value. If the permissible overall correction is exceeded, signal processing is stopped.

4 Installation

The option package can either be installed on the robot controller via the smartHMI or via WorkVisual.

4.1 System requirements

Hardware

- KR C4 robot controller
- For data exchange via Ethernet:
  - Processor-supported external system with real-time-capable operating system and real-time-capable network card with 100 Mbit in full duplex mode
  - Microprocessor-supported sensor with real-time-capable network card for use in sensor applications
  - Network cable for switch, hub or crossed network cable for direct connection
- For data exchange via an I/O system: bus system, e.g. PROFINET
- Laptop/PC for working with WorkVisual

Recommended robots

RobotSensorInterface should only be used in combination with KUKA 6-axis robots. The use of other robots may be planned only in consultation with KUKA Roboter GmbH.

(>>> *12 "KUKA Service" Page 106*)

	Sensor correction cannot be used for asynchronous axes.

Software

Robot controller:

- KUKA System Software 8.5

Laptop/PC:

- WorkVisual 5.0

KRL resources

For RSI corrections in IPO mode, the following KRL resources must be free:

KRL resource	Number
Function generator	1

Compatibility

RobotSensorInterface must not be installed on a robot controller together with the following option packages:

- KUKA.ConveyorTech
- KUKA.ServoGun TC
- KUKA.ServoGun FC
- KUKA.EqualizingTech

RoboTeam

RobotSensorInterface can be operated together with RoboTeam, but certain limitations must be observed (>>> *2.4.4 "Sensor correction in RoboTeam operation" Page 20*).

4.2 Installation via WorkVisual

4.2.1 Installing or updating RobotSensorInterface

Description

The option package is installed in WorkVisual and added to the project. During project deployment, the option package is automatically installed on the robot controller.

In the case of an update, the previous version of the option package in WorkVisual must first be uninstalled.

It is advisable to archive all relevant data before updating a software package.

Precondition

- User group “Expert”
- T1 or T2 mode
- No program is selected.
- Network connection between PC and robot controller
- The option package is available as a KOP file.

Procedure

1. Only for an update: Uninstall the previous version of the **RobotSensorInterface** option package in WorkVisual.
2. Install the **RobotSensorInterface** option package in WorkVisual.
3. Load the active project from the robot controller.
4. Insert the **RobotSensorInterface** option package into the project.
5. Configure the option package in WorkVisual as required.
   (>>> *5.2 "Mapping inputs/outputs of RSI in WorkVisual" Page 31*)
6. Deploy the project from WorkVisual to the robot controller and activate it.
7. The request for confirmation *Do you want to activate the project […]?* is displayed on the smartHMI. The active project is overwritten during activation. If no relevant project will be overwritten: Answer the query with **Yes**.
8. An overview with the changes and a request for confirmation are displayed on the smartHMI. Answer this with **Yes**. The option package is installed and the robot controller carries out a reboot.

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

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.2.2 Uninstalling RobotSensorInterface

It is advisable to archive all relevant data before uninstalling a software package.

Precondition

- User group “Expert”
- T1 or T2 mode
- No program is selected.
- Network connection between PC and robot controller

Procedure

1. Load the project from the robot controller.
2. Remove the **RobotSensorInterface** option package from the project. A window with modifications is displayed.
3. Deploy the project from WorkVisual to the robot controller and activate it.
4. Answer the request for confirmation *Do you want to activate the project […]?* on the smartHMI with **Yes**.
5. An overview with the changes and a request for confirmation are displayed on the smartHMI. Answer this with **Yes**. The option package is uninstalled and the robot controller carries out a reboot.

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

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.3 Installation via smartHMI

4.3.1 Installing or updating RobotSensorInterface

It is advisable to archive all relevant data before updating a software package.

In the case of an update, the existing configuration is automatically adopted. If this is not desired, the existing version must first be uninstalled.

Precondition

- User rights: Function group **General configuration**
- T1 or T2 mode
- No program is selected.
- USB stick with the option package

NOTICE

We recommend using a KUKA USB stick. Data may be lost if a stick from a different manufacturer is used.

Procedure

1. Connect the USB stick to the robot controller or smartPAD.
2. In the main menu, select **Start-up** > **Additional software**.
3. Press **New software**: The entry **RobotSensorInterface** must be displayed in the **Name** column and drive **E:** or **K:** in the **Path** column.
   If not, press **Refresh**.
4. If the specified entries are now displayed, continue with step 5.
   Otherwise, the path from which the software is to be installed must be configured first:
   a. Press the **Configure** button.
   b. Select a line in the **Installation paths for options** area.
      Note: If the line already contains a path, this path will be overwritten.
   c. Press **Path selection**. The available drives are displayed.
   d. If the stick is connected to the robot controller: Select **E:**.
      If the stick is connected to the smartPAD: **K:** instead of **E:**
   e. Press **Save**. The **Installation paths for options** area is displayed again. It now contains the new path.
   f. Mark the line with the new path and press **Save** again.
5. Activate the check box next to **RobotSensorInterface** and press **Install**. Confirm the installation query with **OK**.
6. The request for confirmation *Do you want to activate the project […]?* is displayed. The active project is overwritten during activation. If no relevant project will be overwritten: Answer the query with **Yes**.
7. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is installed and the robot controller carries out a reboot.
8. Remove the stick.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

4.3.2 Uninstalling RobotSensorInterface

It is advisable to archive all relevant data before uninstalling a software package.

Precondition

- User rights: Function group **General configuration**
- T1 or T2 mode
- No program is selected.

Procedure

1. In the main menu, select **Start-up** > **Additional software**.
2. Activate the check box next to **RobotSensorInterface** and press **Uninstall**. Answer the request for confirmation with **Yes**.
3. Answer the request for confirmation *Do you want to activate the project […]?* with **Yes**.
4. An overview with the changes and a request for confirmation are displayed. Answer this with **Yes**. The option package is uninstalled and the robot controller carries out a reboot.

LOG file

A LOG file is created under C:\KRC\ROBOTER\LOG.

5 Configuration

5.1 Configuring Ethernet via KLI network configuration

Description

A network connection must be established via the KLI of the robot controller in order to exchange data via Ethernet. For this, a separate Ethernet sensor network is required which is independent of other KLI subnetworks.

The following Ethernet interfaces are available as options at the customer interface of the robot controller, depending on the specification:

- Interface X66 (1 slot)
- Interface X67.1-3 (3 slots)

Further information on the Ethernet interfaces can be found in the operating or assembly instructions for the robot controller.

Precondition

- User rights: Function group **General configuration**
- Network connection via the KLI
- T1 or T2 mode
- No program is selected.

Procedure

1. In the main menu, select **Start-up** > **Network configuration**. The **Network configuration** window opens.
2. Press **Advanced...**. The window for advanced network configuration opens.
3. Press **Add interface**. A new entry is created automatically in the **Configured interfaces:** area.
4. Select the newly created entry and enter the network name (e.g. Ethernet sensor network) in the **Interface designation:** box.
5. Select the **Mixed IP address** type in the **Address type:** box.

   By selecting the **Mixed IP address** type, the receiving tasks required are created automatically:

   - **Receiving task**:
     - **Reception filter: Target subnet**
   - **Real-time receiving Task**:
     - **Reception filter: UDP**
6. In the boxes underneath, enter the IP address of the robot controller and the subnet mask.

The IP address of the sensor (default: 192.168.1.1) and the robot controller must be located in the same network segment. In other words, only the last of the 4 sections of the addresses may differ.

The IP address range 192.168.0.x is blocked for the configuration of the network connection. The IP address entered must be in a separate subnet. The address must not be in the address range of another KLI subnet.
Example:
- IP address: 192.168.1.2
- Subnet mask: 255.255.255.0

7. Press **Save**.
8. Reboot the robot controller so that the change takes effect.

Further information about KLI network configuration can be found in the Operating and Programming Instructions for System Integrators for the system software.

5.2 Mapping inputs/outputs of RSI in WorkVisual

Description

The RSI inputs/outputs are mapped in the WorkVisual mapping editor to the inputs/outputs of a field bus device or to the inputs/outputs of another I/O system.

The following procedure describes the mapping of an RSI input/output to the input/output of a field bus device. When mapping to another I/O system, the procedure differs slightly. Further information about I/O mapping can be found in the WorkVisual documentation.

Precondition

- The robot controller has been set as the active controller.
- The bus configuration in WorkVisual corresponds to the real bus configuration.
- The field bus devices are configured.
- The **I/O Mapping** window is open.

Procedure

1. On the **RSI** tab in the left-hand half of the **I/O Mapping** window, select the RSI area that is to be mapped, e.g. **Digital inputs**.
   The KSI signals are displayed in the bottom area of the **I/O Mapping** window.
2. Select the desired device on the **Field buses** tab in the right-hand half of the **I/O Mapping** window.
   The signals of the device are displayed in the bottom area of the **I/O Mapping** window.
3. Drag the RSI signal onto the input or output of the device. (Or alternatively, drag the input or output of the device onto the RSI signal.)
   The “**Signal grouping**” dialog opens.
4. Confirm the dialog with **Yes**. The signal is grouped and then mapped.

The data types of the signals in WorkVisual have no influence on the data types used in the RSI context. They are purely for grouping purposes. The data types of the field bus devices must be assigned in the RSI context.

Example

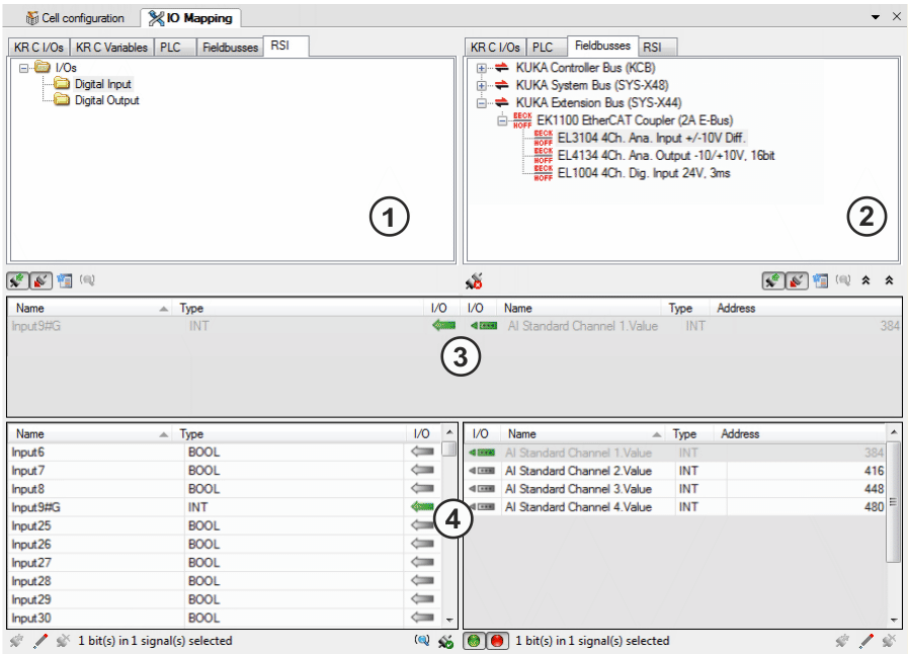


Fig. 5-1: "I/O Mapping" window

Item	Description
1	Display of the input/output types of RSI The area to be mapped is selected here. The signals of the selected area are shown in the display lower down.
2	Display of the field bus devices The field bus device to be mapped is selected here. The signals of the selected field bus device are shown in the display lower down.
3	Display of the mapped signals
4	Display of all signals The inputs/outputs are mapped here.

In this example, the analog input of an EtherCAT device has been mapped to a digital RSI input. When using this signal in the RSI context, the following parameters must be assigned to the **Input** object.

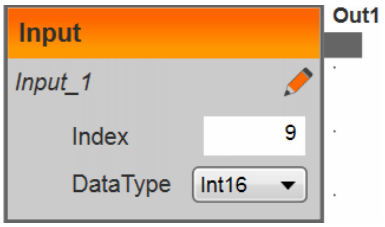


Fig. 5-2: Input object parameters

Parameter	Description
Index	Number of the first bit of the grouped signal
DataType	Data type corresponding to the bit width of the grouped signal

5.3 Modifying global variables

Description

The following global variables can be modified by the user:

Variable	Description
RSIERRMSG	Acknowledgement message when errors occur upon executing RSI commands TRUE = errors are displayed on the smartHMI with an acknowledgement message. FALSE = no acknowledgement message. For error treatment, the return values of the RSI commands must be evaluated in the KRL program. Default: TRUE
RSITECHIDX	Function generator for RSI corrections in IPO mode Default value: **1** The maximum number of function generators is defined in the machine data ($TECH_MAX).

Configuration file

Directory	KRC:\R1\TP\RSI
File	RSI_USER.DAT

```
DEFDAT RSI_USER PUBLIC
...
RSI global Variables:
GLOBAL BOOL RSIERRMSG=TRUE
...
; Flag for writing context information
GLOBAL INT RSITECHIDX=1
; Tech Channel used for RSI corrections
ENDDAT
```

Precondition

- User group Expert

Procedure

1. Select the file in the Navigator and press **Open**. The file is displayed in the editor.
2. Modify variables as required.
3. Save and close the file.

6 Programming

6.1 Overview of RSI commands

RobotSensorInterface provides functions for programming the signal processing. Each of these functions, with the exception of RSI_MOVECORR(), has a return value. The return value can be polled and evaluated in the KRL program.

Constants are declared as error codes in the data list RSI.DAT in the directory KRC:\R1\TP\RSI. To check whether an RSI command has been executed correctly, the constants specified in the function descriptions can be used.

Function	Description
RSI_CREATE()	(>>> *6.1.2 "RSI_CREATE()" Page 35*)
RSI_DELETE()	(>>> *6.1.3 "RSI_DELETE()" Page 35*)
RSI_ON()	(>>> *6.1.4 "RSI_ON()" Page 36*)
RSI_OFF()	(>>> *6.1.5 "RSI_OFF()" Page 37*)
RSI_MOVECORR()	(>>> *6.1.6 "RSI_MOVECORR()" Page 37*)
RSI_GETPUBLICPAR()	(>>> *6.1.7 "RSI_GETPUBLICPAR()" Page 38*)
RSI_SETPUBLICPAR()	(>>> *6.1.8 "RSI_SETPUBLICPAR()" Page 38*)
RSI_RESET()	(>>> *6.1.9 "RSI_RESET()" Page 39*)
RSI_CHECKID()	(>>> *6.1.10 "RSI_CHECKID()" Page 39*)
RSI_ENABLE()	(>>> *6.1.11 "RSI_ENABLE()" Page 40*)
RSI_DISABLE()	(>>> *6.1.12 "RSI_DISABLE()" Page 40*)

6.1.1 Symbols and fonts

The following symbols and fonts are used in the syntax descriptions:

Syntax element	Representation
KRL code	• Courier font • Upper-case letters Examples: `GLOBAL`; `ANIN ON`; `OFFSET`
Elements that must be replaced by program-specific entries	• Italics • Upper/lower-case letters Examples: *Distance*; *Time*; *Format*
Optional elements	• In angle brackets Example: <`STEP` *Increment*>
Elements that are mutually exclusive	• Separated by the "|" symbol Example: `IN` |`OUT`

6.1.2 RSI_CREATE()

Description

RSI_CREATE() creates an RSI container and loads an RSI context into the container. The created container can be accessed using the container ID.

The container created with RSI_CREATE() is activated by default. If the container is deactivated (element *Status:IN*), it must be reactivated with RSI_ON() before the signal processing is activated. RSI_ENABLE() activates a deactivated container.

Syntax

RET=RSI_CREATE(*File:IN*<, *ContainerID:OUT*><, *Status:IN*>)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIFILENOTFOUND: File not found with the signal configuration • RSIINVFILE: Invalid file, e.g. invalid file format or error in the configuration • RSINOMEMORY: No free RSI memory available • RSIINVOBJTYPE: Unknown object in the RSI context • RSIEXTLIBNOTFOUND: External RSI object library not found • RSINOTLINKED: RSI object with missing input signal • RSILNKCIRCLE: Error in the signal link
File:IN	Type: CHAR array File name of the RSI context (without .rsix file extension)
ContainerID:OUT	Type: INT ID of the RSI container
Status:IN	Type: BOOL True = activates the RSI container False = deactivates the RSI container Default: TRUE

6.1.3 RSI_DELETE()

Description

RSI_DELETE() deletes an RSI container and the RSI objects it contains.

Syntax

RET=RSI_DELETE(*ContainerID:IN*)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIINVOBJID: Invalid container ID
ContainerID:IN	Type: INT ID of the RSI container

6.1.4 RSI_ON()

Description

RSI_ON() activates the signal processing and defines the correction mode and sensor mode.

The signal processing is carried out by default in sensor mode #IPO_FAST. In this case, the reference coordinate system for the sensor correction must be configured in the RSI object POSCORR. If the signal processing is activated in IPO mode, the reference coordinate system must be defined with RSI_ON().

If the signals are processed in IPO mode, the path can only be corrected using LIN and CIRC motions.

Syntax

RET=RSI_ON(<*Correction mode:IN*><, *Sensor mode:IN*><, *Coordinate system:IN*>)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIALREADYON: Signal processing is already activated.
Correction mode:IN	Type: ENUM Correction mode: • #ABSOLUTE: Absolute correction • #RELATIVE: Relative correction Default: #ABSOLUTE
Sensor mode:IN	Type: ENUM Signal processing mode: • #IPO_FAST: 4 ms • #IPO: 12 ms with filtering ($FILTER) Default: #IPO_FAST
Coordinate system:IN	Type: ENUM

Element	Description
	Reference coordinate system for the sensor correction (only relevant if sensor mode = #IPO) • #BASE • #TCP • #TTS • #WORLD Default: #BASE

6.1.5 RSI_OFF()

Description

RSI_OFF() deactivates the signal processing.

Syntax

RET=RSI_OFF()

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSINOTRUNNING: No signal processing running

6.1.6 RSI_MOVECORR()

Description

RSI_MOVECORR() activates the sensor-guided motion. The robot is controlled purely by means of corrections on the basis of sensor data, i.e. with the correction values of the RSI objects POSCORR or AXISCORR.

A sensor-guided motion can be terminated by means of the RSI object STOP.

Syntax

RSI_MOVECORR(<*Stop mode:IN*>)

Explanation of the syntax

Element	Description
Stop mode	Type: ENUM Behavior after termination of the motion: • #RSIBRAKE: Robot resumes motion directly from the stopping point. • #RSIBRAKERET: Robot returns to the point on the path at which the stop signal was received. Default: RSIBRAKE

6.1.7 RSI_GETPUBLICPAR()

Description

The parameter value of an RSI object can be read with RSI_GETPUBLICPAR().

Precondition

- RSI object parameter is enabled.
  (>>> *7.3.5 "Enabling RSI object parameters" Page 56*)

Syntax

RET=RSI_GETPUBLICPAR(*ContainerID:IN*, *Object:IN*, *Parameter:IN*, *Value:OUT*)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIINVCONT: Invalid container ID • RSIINPARAMID: Invalid RSI object or parameter name, or RSI object parameter is not enabled.
ContainerID:IN	Type: INT ID of the RSI container
Object:IN	Type: CHAR array Name of the RSI object
Parameter:IN	Type: CHAR array Name of the RSI object parameter
Value:OUT	Type: REAL Value of the RSI object parameter

To access a function block parameter via KRL, a special object name must be used.
(>>> *7.5.5 "Access to function block parameters via KRL" Page 66*)

6.1.8 RSI_SETPUBLICPAR()

Description

A new value can be assigned to the parameter of an RSI object with RSI_SETPUBLICPAR().

Precondition

- RSI object parameter is enabled.
  (>>> *7.3.5 "Enabling RSI object parameters" Page 56*)

Syntax

RET=RSI_SETPUBLICPAR(*ContainerID:IN*, *Object:IN*, *Parameter:IN*, *Value:IN*)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIINVCONT: Invalid container ID • RSIINPARAMID: Invalid RSI object or parameter name, or RSI object parameter is not enabled. • RSIINPARAM: Invalid RSI object parameter value
ContainerID:IN	Type: INT ID of the RSI container
Object:IN	Type: CHAR array Name of the RSI object
Parameter:IN	Type: CHAR array Name of the RSI object parameter
Value:IN	Type: REAL New value of the RSI object parameter

To access a function block parameter via KRL, a special object name must be used.
(>>> *7.5.5 "Access to function block parameters via KRL" Page 66*)

6.1.9 RSI_RESET()

Description

RSI_RESET() resets signal processing and deletes all RSI objects.

Syntax

RET=RSI_RESET()

Explanation of the syntax

Element	Description
RET	Type: INT Return value: • RSIOK: Function executed successfully

6.1.10 RSI_CHECKID()

Description

RSI_CHECKID() can be used to check whether a valid RSI container ID is being used.

Syntax

RET=RSI_CHECKID(*ContainerID:IN*)

Explanation of the syntax

Element	Description
RET	Type: BOOL Return values: • TRUE = RSI container available for this ID • FALSE = No RSI container available for this ID
ContainerID:IN	Type: INT ID of the RSI container

6.1.11 RSI_ENABLE()

Description

RSI_ENABLE() activates a deactivated RSI container.

Syntax

RET=RSI_ENABLE(*ContainerID:IN*)

Explanation of the syntax

Element	Description
RET	Type: INT Return values: • RSIOK: Function executed successfully • RSIINVOBJID: Invalid container ID
ContainerID:IN	Type: INT ID of the RSI container

6.1.12 RSI_DISABLE()

Description

RSI_DISABLE() deactivates an RSI container.

A deactivated container must be activated again with RSI_ON() before signal processing is enabled. RSI_ENABLE() activates a deactivated container.

Syntax

RET=RSI_DISABLE(*ContainerID:IN*)

Explanation of the syntax

Element	Description
RET	Type: INT

Element	Description
	Return values: • RSIOK: Function executed successfully • RSIINVOBJID: Invalid container ID
ContainerID:IN	Type: INT ID of the RSI container

6.2 Response of the RSI commands RSI_ENABLE() and RSI_DISABLE()

Description

A specified RSI context can be paused using the commands RSI_ENABLE(ContId) and RSI_DISABLE(ContId). The current values within the context are maintained following the activation of RSI_DISABLE(). After the pause, the RSI context continues as usual.

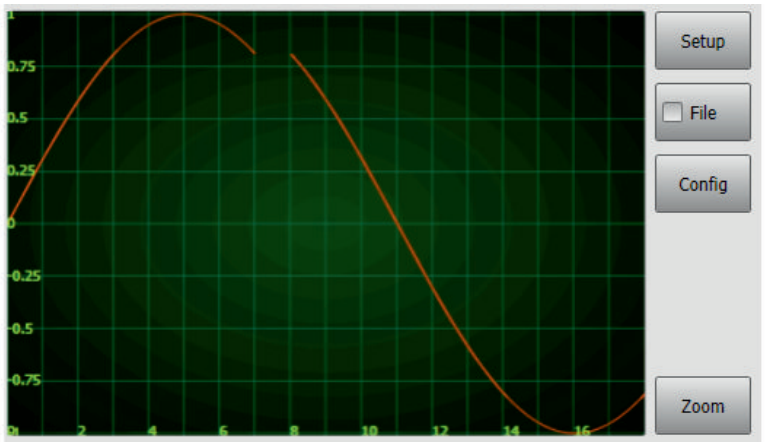


Fig. 6-1: Paused RSI context

Example

Pausing the RSI context via RSI_DISABLE() when generating a sine signal by means of a SOURCE object.

```
DEF RSI_DISABLEENABLE()
DECL INT ret,cont
ret=RSI_CREATE("Signals", cont)
ret=RSI_ON()
wait sec 7
ret=RSI_DISABLE(cont)
wait sec 1
ret=RSI_ENABLE(cont)
wait sec 10
ret=RSI_OFF()
END
```

During the pause, sensors may lose their values. This can lead to an unexpected response when restarting the context with RSI_ENABLE(ContId).

6.3 Response of the RSI commands RSI_ON() and RSI_OFF()

Description

The entire RSI context is restarted with the commands RSI_OFF and RSI_ON. No single RSI context can be addressed with these commands, but rather all contexts created via RSI_CREATE() are addressed jointly.

Fig. 6-2: Restarting the context

Example

Resetting the RSI context via RSI_OFF() when generating a sine signal by means of a SOURCE object.

```
DEF RSI_ONOFF()
DECL INT ret,cont
ret=RSI_CREATE("Signals",cont)
ret=RSI_ON()
wait sec 7
ret=RSI_OFF()
wait sec 1
ret=RSI_ON()
Wait sec 10
ret=RSI_OFF()
END
```

6.4 Overview: Programming signal processing

Step	Description
1	Configure the RSI context with RSIVisual.
2	Transfer the RSI context to the robot controller. Target directory: C:\KRC\Roboter\Config\User\Common\SensorInterface
3	Integrate the RSI context into a KRL program. (>>> *6.4.1 "Integrating an RSI context into a KRL program" Page 43*)
4	Subsequently modify the RSI object parameters in KRL if necessary. (>>> *6.4.2 "Access to RSI object parameters via KRL" Page 43*)

6.4.1 Integrating an RSI context into a KRL program

Description

The signal processing must be initialized, activated and then deactivated again in the KRL program.

Structure of a signal processing program:

```
1  DEF signal_processing()
2
3  DECL INT ret
4
3  INI
...
6  ret=RSI_CREATE("myContext")
7  ret=RSI_ON()
...
10 movements
...
15 ret=RSI_OFF()
...
20 END
```

Line	Description
3	Declaration of the KRL variables (here only the variable “ret” for the return value)
6	RSI_CREATE() initializes the signal processing. The RSI context is loaded into an RSI container.
7	RSI_ON() activates the signal processing.
10	Motion instructions or RSI_MOVECORR() for a sensor-guided motion
15	RSI_OFF() deactivates the signal processing.

6.4.2 Access to RSI object parameters via KRL

Description

RSI object parameters can be subsequently read or set in the KRL program by means of the following functions:

- RSI_GETPUBLICPAR()
  Function for reading the object parameter value
- RSI_SETPUBLICPAR()
  Function for setting the object parameter value

Precondition

- RSI object parameter is enabled.
  (>>> *7.3.5 "Enabling RSI object parameters" Page 56*)
- KRL program has been started.
- RSI context has been loaded with RSI_CREATE().

Example

(>>> *8.4 "RSI_CircleCorr example: Sensor-guided circular motion" Page 75*)

6.5 Configuring an XML file for the Ethernet connection

Description

RobotSensorInterface uses the XML format to exchange data via Ethernet. A configuration file must be defined for the Ethernet connection in the directory C:\KRC\ROBOTER\Config\User\Common\SensorInterface.

The name of the configuration file must be specified later in the Ethernet object of the RSI context and be read during initialization of the signal processing in the KRL program.

Overview

Structure of the configuration file:

```
<ROOT>
  <CONFIG></CONFIG>
  <SEND>
    <ELEMENTS></ELEMENTS>
  </SEND>
  <RECEIVE>
    <ELEMENTS></ELEMENTS>
  </RECEIVE>
</ROOT>
```

Section	Description
<CONFIG … </CONFIG>	Configuration of the connection parameters between the sensor system and the interface (>>> *6.5.1 "XML structure for connection properties" Page 44*)
<SEND> … </SEND>	Configuration of the transmission structure (>>> *6.5.2 "XML structure for data transmission" Page 45*)
<RECEIVE> … </RECEIVE>	Configuration of the reception structure (>>> *6.5.3 "XML structure for data reception" Page 47*)

WorkVisual

A template for configuration of the Ethernet connection is available in WorkVisual.

(>>> *6.5.7 "Configuring the Ethernet connection in WorkVisual" Page 52*)

6.5.1 XML structure for connection properties

Description

Elements of the XML structure:

Element	Description
IP_NUMBER	IP address of the sensor system
PORT	Port number of the sensor system • **1 … 65,534**

Element	Description
SENTYPE	Identifier of the sensor system (name freely definable) The robot controller checks the identifier for every data packet it receives.
ONLYSEND	Direction of data exchange • TRUE = the robot controller sends data and expects no return data from the sensor system. • FALSE = the robot controller sends and receives data. Default: FALSE

Example

```
<CONFIG>
<IP_NUMBER>172.1.10.5</IP_NUMBER>
<PORT>49152</PORT>
<SENTYPE>ImFree</SENTYPE>
<ONLYSEND>FALSE</ONLYSEND>
</CONFIG>
```

6.5.2 XML structure for data transmission

Description

The signals from the RSI context that arrive at the inputs of the ETHERNET object and are sent to the sensor system are defined here.

The ETHERNET object also has a read function that can be used to read system information from the robot controller and send it to the sensor system. The read function is activated using keywords.

From the configured XML structure, RobotSensorInterface automatically creates the XML document that the robot controller transmits.

Signal inputs

Definition of the signal inputs in the XML structure:

Attribute	Description
TAG	Name of the element The XML structure for data transmission is defined here (XML schema). (>>> *6.5.4 "Configuration according to XML schema" Page 49*)
TYPE	Data type of the element • BOOL • DOUBLE • LONG
INDX	Number of the ETHERNET object input • **1 … 64** **Note**: The object inputs must be numbered consecutively.

Example of signal inputs

- Configured XML structure for data transmission:

```
<SEND>
<ELEMENTS>
<ELEMENT TAG="Out.01" TYPE="BOOL" INDX="1" />
<ELEMENT TAG="Out.02" TYPE="BOOL" INDX="2" />
<ELEMENT TAG="Out.03" TYPE="BOOL" INDX="3" />
<ELEMENT TAG="Out.04" TYPE="BOOL" INDX="4" />
<ELEMENT TAG="Out.05" TYPE="BOOL" INDX="5" />
<ELEMENT TAG="FTC.Fx" TYPE="DOUBLE" INDX="6" />
<ELEMENT TAG="FTC.Fy" TYPE="DOUBLE" INDX="7" />
<ELEMENT TAG="FTC.Fz" TYPE="DOUBLE" INDX="8" />
<ELEMENT TAG="FTC.Mx" TYPE="DOUBLE" INDX="9" />
<ELEMENT TAG="FTC.My" TYPE="DOUBLE" INDX="10" />
<ELEMENT TAG="FTC.Mz" TYPE="DOUBLE" INDX="11" />
<ELEMENT TAG="Override" TYPE="LONG" INDX="12" />
</ELEMENTS>
</SEND>
```

- XML document transmitted by the robot controller:

```
<Rob TYPE="KUKA">
<Out 01="0" 02="1" 03="1" 04="0" 05="0" />
<FTC Fx="1.234" Fy="54.75" Fz="345.7
Mx="2346.6" My="12.0" Mz="3546" />
<Override>90</Override>
<IPOC>123645634563</IPOC>
</Rob>
```

The keyword IPOC sends a time stamp and is generated automatically.

Read function

Activation of the read function in the XML structure:

Attribute	Description
TAG	Name of the element A keyword specifies which system information is read. (>>> *6.5.5 "Keywords – reading data" Page 49*)
TYPE	Data type of the element • DOUBLE • LONG
INDX	Keyword for reading the system information • INTERNAL

Example of read function

(>>> *"Example of read function" Page 50*)

6.5.3 XML structure for data reception

Description

The signals received by the sensor system at the outputs of the ETHERNET object and forwarded to the robot controller in the RSI context are defined here.

The ETHERNET object also has a write function that can be used to write information to the robot controller or generate messages on the smartHMI. The write function is activated using keywords.

From the configured XML structure, RobotSensorInterface automatically creates the XML document that the robot controller is expecting.

Signal outputs

Definition of the signal outputs in the XML structure:

Attribute	Description
TAG	Name of the element The XML structure for data reception is defined here (XML schema). (>>> *6.5.4 "Configuration according to XML schema" Page 49*)
TYPE	Data type of the element • BOOL • DOUBLE • LONG
INDX	Number of the ETHERNET object output • **1 … 64** **Note**: The object outputs must be numbered consecutively.
HOLDON	Behavior of the object output with regard to data packets that arrive too late • **0**: The output is reset. • **1**: The most recent valid value to arrive remains at the output.

Example of signal outputs

- Configured XML structure for data reception:

```
<RECEIVE>
<ELEMENTS>
<ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" HOL-
DON="1" />
<ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" HOL-
DON="1" />
<ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" HOL-
DON="1" />
<ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" HOL-
DON="1" />
<ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" HOL-
DON="1" />
<ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" HOL-
DON="1" />
<ELEMENT TAG="AK.A1" TYPE="DOUBLE" INDX="7" HOLDON="0" />
```

```
<ELEMENT TAG="AK.A2" TYPE="DOUBLE" INDX="8" HOLDON="0" />
<ELEMENT TAG="AK.A3" TYPE="DOUBLE" INDX="9" HOLDON="0" />
<ELEMENT TAG="AK.A4" TYPE="DOUBLE" INDX="10" HOL-
DON="0" />
<ELEMENT TAG="AK.A5" TYPE="DOUBLE" INDX="11" HOL-
DON="0" />
<ELEMENT TAG="AK.A6" TYPE="DOUBLE" INDX="12" HOL-
DON="0" />
<ELEMENT TAG="EK.E1" TYPE="DOUBLE" INDX="13" HOL-
DON="0" />
<ELEMENT TAG="EK.E2" TYPE="DOUBLE" INDX="14" HOL-
DON="0" />
<ELEMENT TAG="EK.E3" TYPE="DOUBLE" INDX="15" HOL-
DON="0" />
<ELEMENT TAG="EK.E4" TYPE="DOUBLE" INDX="16" HOL-
DON="0" />
<ELEMENT TAG="EK.E5" TYPE="DOUBLE" INDX="17" HOL-
DON="0" />
<ELEMENT TAG="EK.E6" TYPE="DOUBLE" INDX="18" HOL-
DON="0" />
<ELEMENT TAG="DiO" TYPE="LONG" INDX="19" HOLDON="1" />
</ELEMENTS>
</RECEIVE>
```

- XML document received by the sensor system:

```
<Sen Type="ImFree">
<RKorr X="4" Y="7" Z="32" A="6" B="" C="6" />
<AK A1="2" A2="54" A3="35" A4="76" A5="567" A6="785" />
<EK E1="67" E2="67" E3="678" E4="3" E5="3" E6="7" />
<DiO>123</DiO>
<IPOC>123645634563</IPOC>
</Sen>
```

The time stamp with the keyword IPOC is checked. The data packet is only valid if the time stamp corresponds to the time stamp sent previously.

Write function

Activation of the write function in the XML structure:

Attribute	Description
TAG	Name of the element A keyword specifies which information is written to the robot controller or whether a message is generated on the smartHMI.
TYPE	Data type of the element • DOUBLE • STRING
INDX	Keyword for writing the information • INTERNAL
HOLDON	Behavior of the object output with regard to data packets that arrive too late • **0**: The output is reset. • **1**: The most recent valid value to arrive remains at the output.

Example of write function

(>>> *"Example of write function" Page 51*)

6.5.4 Configuration according to XML schema

Description

From the configured XML structure, RobotSensorInterface automatically creates the XML documents for the data exchange.

The following notations are to be distinguished according to XML schema:

- Element notation
- Attribute notation

Element notation

- TAGs in the configured XML structure:

```
...
<ELEMENTS>
<ELEMENT TAG="Out1" ... />
<ELEMENT TAG="Out2" ... />
<ELEMENT TAG="Out3" ... />
</ELEMENTS>
...
```

- TAGs in the created XML document:

```
...
<Out1>...</Out1>
<Out2>...</Out2>
<Out3>...</Out3>
...
```

Attribute notation

- TAGs in the configured XML structure:

```
...
<ELEMENTS>
<ELEMENT TAG="Out.01" ... />
<ELEMENT TAG="Out.02" ... />
<ELEMENT TAG="Out.03" ... />
</ELEMENTS>
...
```

- TAG with attributes in the created XML document:

```
...
<Out 01="..." 02="..." 03="..." />
...
```

6.5.5 Keywords – reading data

Keywords are sequences of letters having a fixed meaning. They must not be used in the XML structure in any way other than with this meaning. No distinction is made between uppercase and lowercase letters. A keyword remains valid irrespective of the way in which it is written.

Keywords

The following robot controller information can be read using keywords in the TAG attribute:

Information	Keyword	Data type
Cartesian actual position	DEF_RIst	DOUBLE
Cartesian setpoint position	DEF_RSol	DOUBLE
Axis-specific actual position of robot axes A1 to A6	DEF_AIPos	DOUBLE
Axis-specific setpoint position of robot axes A1 to A6	DEF_ASPos	DOUBLE
Axis-specific actual position of external axes E1 to E6	DEF_EIPos	DOUBLE
Axis-specific setpoint position of external axes E1 to E6	DEF_ESPos	DOUBLE
Motor currents of robot axes A1 to A6	DEF_MACur	DOUBLE
Motor currents of external axes E1 to E6	DEF_MECur	DOUBLE
Number of late data packets	DEF_Delay	LONG
Technology parameters in the main run (function generators 1 to 6)	DEF_Tech.C1 ... DEF_Tech.C6	DOUBLE
Technology parameters in the advance run (function generators 1 to 6)	DEF_Tech.T1 … DEF_Tech.T6	DOUBLE

Example of read function

- Configured XML structure for data transmission:

```
<SEND>
<ELEMENTS>
<ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" />
<ELEMENT TAG="DEF_AIPos" TYPE="DOUBLE" INDX="INTERNAL" />
<ELEMENT TAG="DEF_MACur" TYPE="DOUBLE" INDX="INTERNAL" />
<ELEMENT TAG="DEF_Delay" TYPE="LONG" INDX="INTERNAL" />
<ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTER-
NAL" />
</ELEMENTS>
</SEND>
```

- XML document transmitted by the robot controller:

```
<Rob TYPE="KUKA">
<RIst X="0.0" Y="0.0" Z="0.0" A="0.0" B="0.0" C="0.0" />
<AIPos A1="0.0" A2="0.0" A3="0.0" A4="0.0" A5="0.0"
A6="0.0" />
<MACur A1="1.0" A2="1.0" A3="1.0" A4="1.0" A5="1.0"
A6="1.0" />
<Delay D="" />
<Tech T11="0.0" T12="0.0" T13="0.0" T14="0.0" T15="0.0"
T16="0.0"
T17="0.0" T18="0.0" T19="0.0" T110="0.0" />
<IPOC>123645634563</IPOC>
```

```
</Rob>
```

The keyword IPOC sends a time stamp and is generated automatically.

6.5.6 Keywords – writing data

Keywords are sequences of letters having a fixed meaning. They must not be used in the XML structure in any way other than with this meaning. No distinction is made between uppercase and lowercase letters. A keyword remains valid irrespective of the way in which it is written.

Keywords

The following information can be written to the robot controller using keywords in the TAG attribute:

Information	Keyword	Data type
Technology parameters in the main run (function generators 1 to 6)	DEF_Tech.C1 ... DEF_Tech.C6	DOUBLE
Technology parameters in the advance run (function generators 1 to 6)	DEF_Tech.T1 ... DEF_Tech.T6	DOUBLE

Keyword in the TAG attribute for generating messages on the smartHMI:

Information	Keyword	Data type
Notification or error message	DEF_EStr	STRING

Message types

The following message types may occur in the XML document written to and transmitted by the sensor system:

- **<EStr> xxx </EStr>**: Notification message
- **<EStr>Error: xxx </EStr>**: Acknowledgement message (robot stop)
- **<EStr/>**: No message if the tag is blank

Example of write function

- Configured XML structure for data reception:

```
<RECEIVE>
<ELEMENTS>
<ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" />
<ELEMENT TAG="DEF_Tech.T2" TYPE="DOUBLE" INDX="INTERNAL"
HOLDON="0" />
</ELEMENTS>
</RECEIVE>
```

- XML document received by the robot system:

```
<Sen Type="ImFree">
<EStr>Message!</EStr>
<Tech T21="0.0" T22="0.0" T23="0.0" T24="0.0" T25="0.0"
T26="0.0"
T27="0.0" T28="0.0" T29="0.0" T210="0.0" />
<IPOC>123645634563</IPOC>
```

```
</Sen>
```

The time stamp with the keyword IPOC is checked. The data packet is only valid if the time stamp corresponds to the time stamp sent previously.

6.5.7 Configuring the Ethernet connection in WorkVisual

Description

The configuration file for the Ethernet connection can be created, edited and transferred to the robot controller in WorkVisual.

Precondition

- RobotSensorInterface is installed in WorkVisual.
- The active project has been loaded from the robot controller, and the project contains the **Templates** catalog.

  OR:

  For working online with KRC Explorer: The workspace **Programming and diagnosis** has been used to establish a connection to the robot controller.

For transferring the project to the robot controller:

- User group “Expert”
- T1 or T2 mode

Procedure

1. Create the configuration file under C:\KRC\USER\COMMON\SensorInterface:
   a. Right-click and select **Add…** from the context menu.
   b. Select and add the **RSI Ethernet Configuration** template.
   c. Enter a name for the configuration and confirm it with **OK**.
2. Open the newly created configuration file and configure the Ethernet connection as required.
3. Save the configuration file.
4. Transfer the project back to the robot controller.

   OR:

   Update the data on the robot controller via the workspace **Programming and diagnosis**.

7 Expert programming with RSIVisual

7.1 Overview of RSIVisual

RSIVisual is a graphical editor with the following functions:

- Creation of RSI contexts in WorkVisual
- Editing of RSI contexts on the robot controller using the **Programming and diagnosis** workspace of WorkVisual

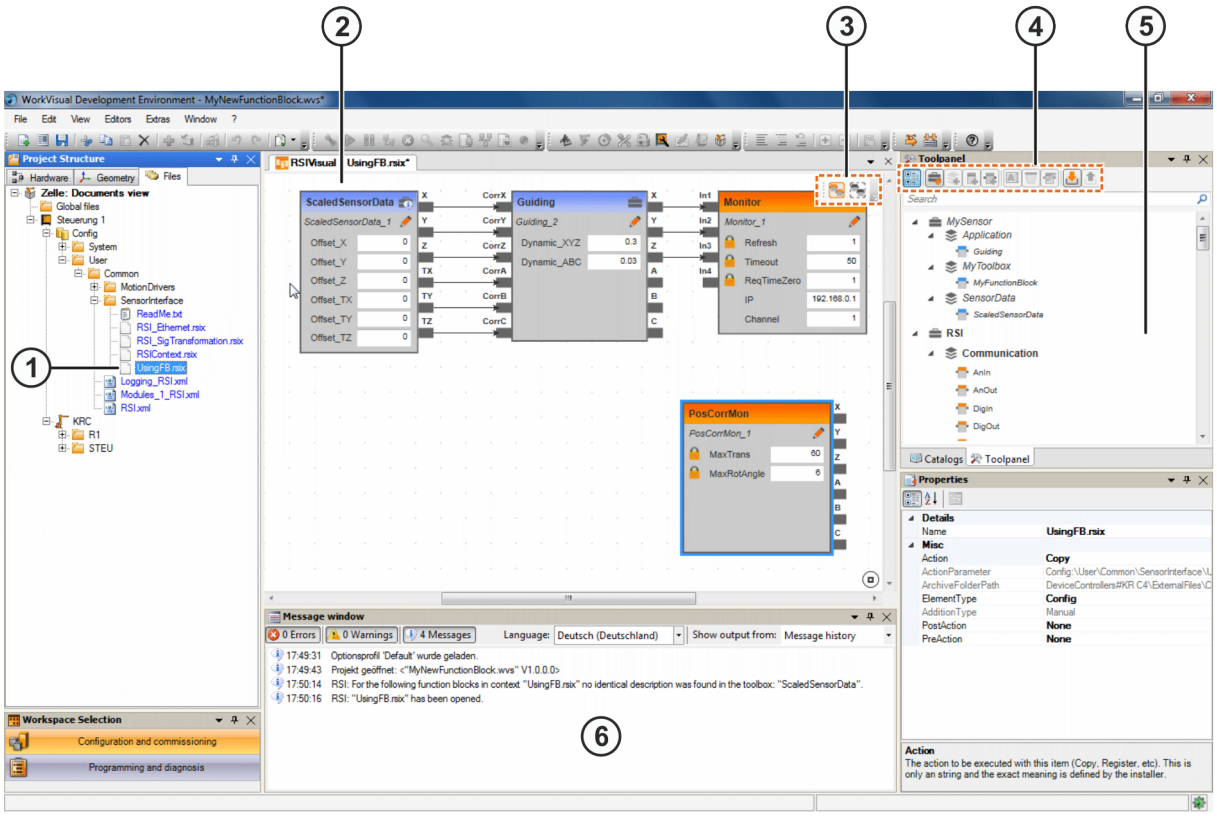


Fig. 7-1: Overview of RSIVisual

Item	Description
1	RSI context (>>> *7.2 "Creating an RSI context" Page 54*)
2	RSIVisual editor with open RSI context (>>> *7.3 "Opening an RSI context with RSIVisual" Page 54*)
3	Buttons for grouping RSI objects (>>> *7.3.7 "Grouping RSI objects" Page 57*)
4	Button bar in toolbox The functionalities provided by the toolbox can be executed using the buttons. (>>> *7.4.1 "Button bar in toolbox" Page 58*)
5	"**Toolpanel**" window When an RSI context is opened in RSIVisual, the available RSI objects are displayed in the toolbox. (>>> *7.4 "Displaying the Toolpanel window" Page 58*)

Item	Description
	The RSI objects can be filtered via the search function in the toolbox by entering part of the object name. Filtering begins when the first letter is entered. No distinction is made between upper-case and lower-case letters.
6	Message window Errors while creating an RSI context are displayed.

7.2 Creating an RSI context

Description

The RSI contexts must be created in the following target directory:

- …\Config\User\Common\SensorInterface

Precondition

- RobotSensorInterface is installed in WorkVisual.
- The active project has been loaded from the robot controller, and the project contains the **Templates** catalog.

  OR:

  For working online with KRC Explorer: The workspace **Programming and diagnosis** has been used to establish a connection to the robot controller.

Procedure

1. Select the **Files** tab in the **Project structure** window.
2. Right-click on the **SensorInterface** folder and select **Add...** from the context menu.
3. Select and add the **RSIContext** template.
4. Enter a file name and confirm with **OK**.

7.3 Opening an RSI context with RSIVisual

Precondition

- RSI context is created.

Procedure

- Double-click or right-click on the file with the RSI context and select **RSIVisual** from the context menu.

Alternative procedure

1. Select the file with the RSI context.
2. Select the menu sequence **Editors** > **Option packages** > **RSIVisual**.

7.3.1 Inserting an RSI object in the context

Description

An RSI object inserted into the context is automatically assigned a generic object name. Object names are comprised as follows:

- Name of the object type plus underscore and a consecutive number

For example, if 3 objects of the **Input** type are inserted into the context, they are automatically assigned the following object names:

- Input_1, Input_2, Input_3

Precondition

- RSI context is open in the editor.
- The **Toolpanel** window is open.

Procedure

- Search for the RSI object in the toolbox and drag it into the editor.

7.3.2 Modifying the name of an RSI object in the context

Description

The name of an RSI object inserted into the context can be modified. The object name that can be edited is marked with a pencil icon.

The new object name must be unique, i.e. it must not already be present in the context. If a name is entered which is not unique, a red border appears around the entry and the context cannot be saved. A corresponding error message is displayed in the message window.

7.3.3 Editing RSI object parameters in the context

Description

The RSI objects must be used and parameterized according to the object descriptions in the help file.

If an invalid value is entered for an object parameter, a red border appears around the entry and the context cannot be saved. A corresponding error message is displayed in the message window.

7.3.4 Connecting inputs and outputs of RSI objects

Description

The inputs and outputs of an RSI object can be connected in both directions – from input to output or vice versa.

The inputs marked in red are mandatory inputs and must be connected to outputs. It is possible to save the RSI context without all the mandatory inputs being connected, however, a corresponding warning is displayed in the message window.

If there are unconnected mandatory inputs present in an RSI context, the context cannot be executed on the robot controller.

Procedure

1. With the left mouse button held down, click on the output and move it to the input.

   OR:

   With the left mouse button held down, click on the input and move it to the output.

2. Release the mouse button when the connection is displayed as desired. When a mandatory input has been connected to an output, its color changes from red to gray.

7.3.5 Enabling RSI object parameters

Description

If the parameter of an RSI object is enabled, its value can be read in the KRL program and reset.

(>>> *6.4.2 "Access to RSI object parameters via KRL" Page 43*)

Precondition

- The parameter is indicated by a padlock icon.
- The padlock is closed.

Procedure

- Click on the padlock icon. The padlock opens.
- To lock the parameter again, click on the padlock icon once again. The padlock closes.

Example

The **Amp** parameter is enabled in the RSI object Sine_1. The padlock is open. The amplitude value for the sine signal can be read and written in KRL.

Fig. 7-2: Amp parameter enabled

7.3.6 Displaying RSI object information

Description

Information can be displayed about the RSI objects in the editor and about individual object elements, e.g. object parameters or inputs/outputs. It is also possible to display brief object descriptions of the RSI objects in the toolbox.

Precondition

- RSI context is open in the editor.
- The **Toolpanel** window is open.

Procedure

- Move the mouse pointer over an RSI object or object element in order to view information about it.

7.3.7 Grouping RSI objects

Description

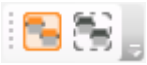
RSI objects can be combined in groups within an RSI context. This serves to increase the clarity of large RSI contexts. In addition, the groups can be saved in a user-defined toolbox and reused in future RSI contexts.

Groups are indicated by a gray frame and are automatically assigned a generic group name: Group_1, Group_2, etc.

The group name can be changed. The new group name must be unique, i.e. it must not already exist in the context. If a name is entered which is not unique, a red border appears around the entry and the context cannot be saved. A corresponding error message is displayed in the message window.

Buttons

The following buttons are available for grouping RSI objects:

Button	Name/description
	**Grouping selected elements** The button is active (orange) if an RSI object is selected.
	**Ungroup** The button is active (orange) if a group is selected.

Precondition

- RSI context is open in the editor.

Procedure

Create group:

1. Select RSI objects in the editor which are to be grouped.
2. Click on the **Grouping selected elements** button.
   Alternatively: Right-click in the editor and select **Grouping selected elements** from the context menu.

Ungroup:

1. Select a group or groups in the editor which are to be disbanded.
2. Click on the **Ungroup** button.
   Alternatively: Right-click in the editor and select **Ungroup** from the context menu.

Example

A group has been formed from a timer function. The button to the left of the group name can be clicked to expand and collapse the group.

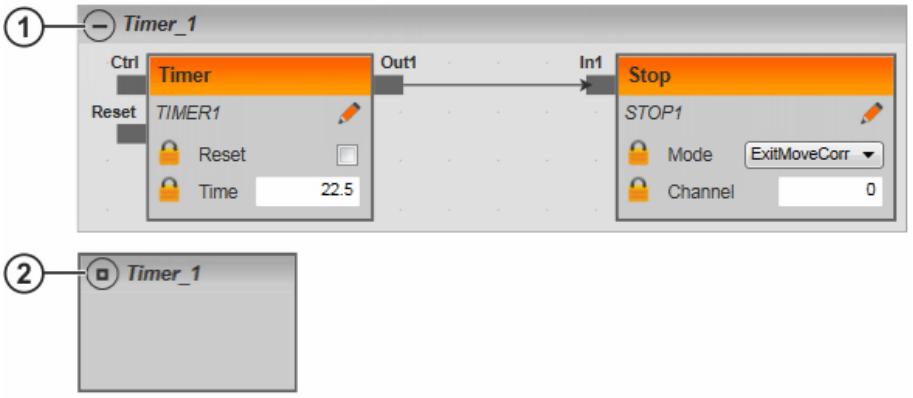


Fig. 7-3: Example of a group in an RSI context

1 Collapse 2 Expand

7.3.8 Additional editing functions

Description

Further functions for editing RSI contexts can be called using the **Edit** menu in WorkVisual:

Cut, **Copy**, **Delete**

Precondition:

- The RSI object is selected.

Paste

Precondition:

- The RSI object has been copied.

To execute these functions, the usual key combinations can also be used, e.g. CTRL+C to copy a selected object.

7.4 Displaying the Toolpanel window

Description

As soon as an RSI context has been opened in RSIVisual, the RSI objects are available in the toolbox and can be dragged into the editor.

If the toolbox is not yet displayed, then the window must be opened.

Precondition

- RSI context is open in the editor.

Procedure

- Select the menu sequence **Window** > **Toolpanel**.

7.4.1 Button bar in toolbox

The edit functions provided by the toolbox in WorkVisual can be executed using the buttons.

Button	Name/description
	**Display categorized toolbox.** Changes the display of the RSI objects in the **Toolpanel** window: • Display according to categories • Display in alphabetical order
	**Create new user-defined toolbox** Opens a dialog for creating a new user-defined toolbox.
	**Create new category** Opens a dialog for creating a new category in the selected user-defined toolbox.
	**Create a new function block** Opens a dialog for creating a new function block in the selected category of a user-defined toolbox.
	**Rename** Opens a dialog for renaming the selected element: user-defined toolbox, category, function block or group
	**Remove** Removes the selected element: user-defined toolbox, category, function block or group
	**Edit function block** Opens the function block selected in a user-defined toolbox for editing in the editor.
	**Import user-defined toolbox** Imports a user-defined toolbox that has been saved as an xml file.
	**Export** Exports the selected user-defined toolbox and saves it as an xml file.

7.4.2 Creating a user-defined toolbox

Description

Function blocks as well as groups that have been created in RSI contexts can be managed and made available in a user-defined toolbox.

Each toolbox contains at least one category. Additional categories can be added.

Precondition

- The **Toolpanel** window is open.

Procedure

1. Click on the **Create new user-defined toolbox** button in the toolbox.
2. Accept the default name with **OK** or enter a new name for the toolbox and confirm it with **OK**.

   The new toolbox is created in the **Toolpanel** window. It already contains one first category.

3. To add a further category, select the newly created toolbox and click on the **Create new category** button.
   Alternatively: Right-click on the newly created toolbox and select **Add category** from the context menu.
4. Accept the default name with **OK** or enter a new name for the category and confirm it with **OK**.

7.4.3 Creating a category for a user-defined toolbox

Description

Various categories can be created for a user-defined toolbox. A category can contain groups and/or function blocks.

Precondition

- The **Toolpanel** window is open.

Procedure

1. Select the desired user-defined toolbox and click on the **Create new category** button.
   Alternatively: Right-click on the user-defined toolbox and select **Add category** from the context menu.
2. Accept the default name with **OK** or enter a new name for the category and confirm it with **OK**.

7.4.4 Adding a group in a user-defined toolbox

Description

The groups created in an RSI context can be added to any of the categories available in a user-defined toolbox.

Precondition

- RSI context is open in the editor.
- The **Toolpanel** window is open.

Procedure

1. Right-click on the desired category and select **Add group...** from the context menu.
   The “**Add group**” dialog opens. The groups available in the current RSI context are displayed.
2. Select the desired group and confirm the selection with **OK**. The group is added to the selected category.

7.4.5 Creating a function block in a user-defined toolbox

Description

The function blocks created in a user-defined toolbox can be used in RSI contexts.

Precondition

- The **Toolpanel** window is open.

Procedure

1. Select the desired category in the user-defined toolbox and click on the **Create a new function block** button.
   Alternatively: Right-click on the category and select **Add function block** from the context menu.
2. Accept the default name with **OK** or enter a new name for the function block and confirm it with **OK**.

7.4.6 Exporting a user-defined toolbox

Description

A user-defined toolbox can be exported from WorkVisual and saved as an xml file.

Precondition

- The **Toolpanel** window is open.

Procedure

1. Select the user-defined toolbox which is to be exported and click on the **Export** button.
   Alternatively: Right-click on the user-defined toolbox and select **Export...** from the context menu.
2. The **Save as** window is opened. Select a storage location for the toolbox.
3. Enter a new name for the toolbox in the **File name** box if necessary.
4. Click on the **Save** button.

7.4.7 Importing a user-defined toolbox

Description

A user-defined toolbox saved as an xml file can be imported into the toolbox of WorkVisual.

Precondition

- The **Toolpanel** window is open.

Procedure

1. Click on the **Import user-defined toolbox** button in the toolbox. A window opens.
2. Navigate to the directory containing the file which is to be imported.
3. Select the file and click on **Open**.
4. If a user-defined toolbox with the same name already exists, a dialog opens to allow a new name to be entered for the toolbox to be imported.
   Enter a new name and confirm with **OK**.

7.4.8 Using a user-defined toolbox in the option package editor

Description

A user-defined toolbox can be added to a user-specific option package in the option package editor. If the user-specific option package is installed in WorkVisual, the user-defined toolbox is available in the WorkVisual toolbox.

Precondition

- OptionPackageEditor is installed.

Procedure

1. Export user-defined toolbox from WorkVisual.
   (>>> *7.4.6 "Exporting a user-defined toolbox" Page 61*)
2. Start the option package editor.
3. Open a user-specific option package in the option package editor or create a new one.

> Further information about creating and editing option packages can be found in the **OptionPackageEditor** documentation.

4. In the structure of the user-specific option package in the **Option package structure** window, insert a new directory with the name **RsiToolbox** under **WorkVisual** > **Plugins**.
5. Insert the user-defined toolbox (xml file) in the new directory.

Example

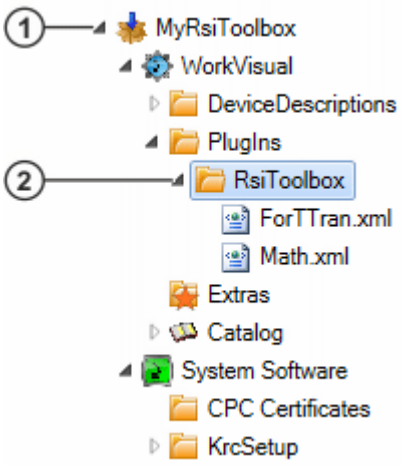


Fig. 7-4: Toolbox in the option package editor

1 User-specific option package in the **Option package structure** window
2 Directory with 2 user-defined toolboxes

7.5 Function block programming

A function block represents a reusable functional unit of an RSI context. While groups are primarily used for enhancing the clarity of a context, a function block is intended to be used like a standard RSI element, without the user receiving or requiring information about the contents of the function block.

7.5.1 Opening a function block for editing in the editor

Description

A newly created function block has no content to start with. A dedicated function block editor is available for editing purposes. The name of the function block and the name of the user-defined toolbox in which the function block was created are displayed in the editor.

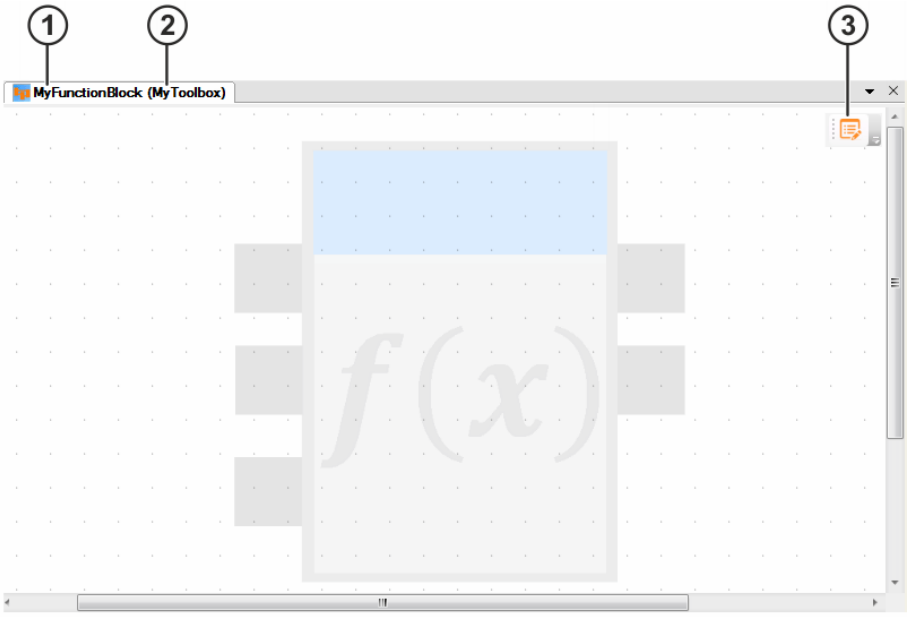


Fig. 7-5: Function block editor

1 Name of the function block
2 Name of the user-defined toolbox
3 **Edit the properties of the function block.** button

Buttons

Button	Name/description
	**Edit the properties of the function block.** Opens the **Function block properties** window.

Precondition

- The **Toolpanel** window is open.

Procedure

- Select the desired function block in the user-defined toolbox and click on the **Edit function block** button.
  Alternatively: Right-click on the function block and select **Edit function block** from the context menu.

7.5.2 Programming a function block

Description

A function block can be programmed like an RSI context. All RSI objects from the toolbox are available for this.

The grouping of RSI objects is not possible in a function block. Nor can the groups created in a user-defined toolbox or other function blocks be used in a function block.

Precondition

- Function block is open in the editor.

Procedure

1. Program the desired function using the RSI objects.
2. Edit the function block properties as required.
3. Save the function block.

7.5.3 Defining inputs and outputs of function blocks

Description

The inputs and outputs of a function block can be defined using special connecting elements.

An input of an RSI object can be linked to a ConnectorIn connecting element in the function block editor, and an output of an RSI object can be linked to a ConnectorOut connecting element.

When using the function block in an RSI context, an input of the RSI object linked to ConnectorIn is shown as an input of the function block. An output of the RSI object linked to ConnectorOut is shown as an output of the function block. The function block can thus be connected to other RSI objects or function blocks in the RSI context.

The connecting elements can only be used in the function block editor.

Connecting elements

The connecting elements can be found in the toolbox under the **Editor** element.

RSI object	Name/description
ConnectorIn ConnectorIn_1 	**Connecting element for inputs in the function block** On insertion into the function block, a name and a number are assigned automatically for the link object: • The name can be changed. This is shown as the name of the function block input when the function block is used. • The number corresponds to the sequence in the display of function block inputs, beginning from the top. The connecting element must be connected to the input of an RSI object in the function block editor. **Note:** All inputs included in a function block must be mapped when used in an RSI context.
ConnectorOut ConnectorOut_1 	**Connecting element for outputs in the function block** On insertion into the function block, a name and a number are assigned automatically for the link object: • The name can be changed. This is shown as the name of the function block output when the function block is used. • The number corresponds to the sequence in the display of function block outputs, beginning from the top.

RSI object	Name/description
	The connecting element must be connected to the output of an RSI object in the function block editor.

7.5.4 Configuring function block parameters

Description

The parameters of RSI objects in function blocks can be configured as function block parameters in the function block editor. The configuration is saved in the toolbox with the function block.

The function block parameter can be set when using the function block in the RSI context. The set function block parameter is saved in the RSI context.

If a function block parameter is assigned to multiple RSI object parameters, the following must be observed:

- The object parameters must be of the same data type.
- The first object parameter assigned to the function block parameter defines the type for further object parameters.

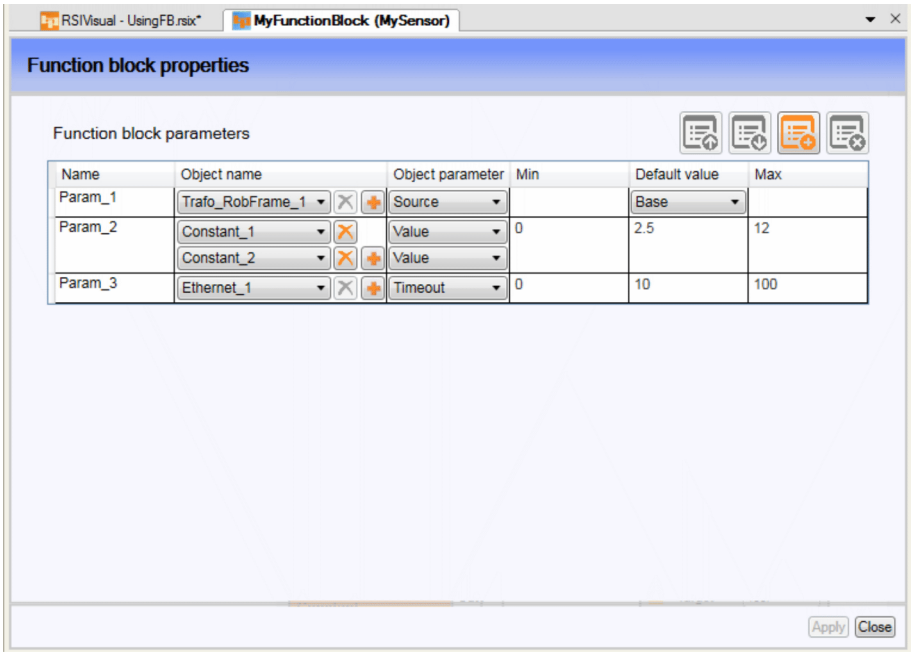


Fig. 7-6: Function block properties

Parameter	Description
Name	Name of the function block parameter A name for the function block parameter can be defined here.
Object name	Name of the RSI object whose parameter is to be written with the function block parameter All parameterizable RSI objects used in the function block are available for selection.
Object parameter	Parameter of the selected RSI object which is to be written with the function block parameter All parameters of the selected RSI object are available for selection.

Parameter	Description
Min	Minimum value of the function block parameter
Default value	Default value of the function block parameter This parameter value is displayed by default when the function block is used in an RSI context.
Max	Maximum value of the function block parameter

Buttons

The following buttons are available for creating the function block parameters:

Button	Name/description
	**Move the selected parameter up.** Moves the selected function block parameter up one line.
	**Move the selected parameter down.** Moves the selected function block parameter down one line.
	**Add a new parameter.** Inserts a new function block parameter.
	**Delete the selected parameter.** Deletes the selected function block parameter.
	**Add parameter** Adds a new object parameter to the function block parameter.
	**Delete parameter** Deletes an object parameter. The button is only active if more than 1 object parameter is assigned to the function block parameter.

Precondition

- Function block is open in the editor.

Procedure

1. Click on the **Edit the properties of the function block.** button in the function block editor. The **Function block properties** window opens.
2. Create or edit the function block parameter.
3. Apply changes and close the window.

7.5.5 Access to function block parameters via KRL

Description

Function block parameters cannot be directly accessed from KRL. However, it is possible to enable those specific parameters of RSI objects within the function block which have also been configured as function block parameters.

(>>> *7.3.5 "Enabling RSI object parameters" Page 56*)

The RSI_GETPUBLICPAR() and RSI_SETPUBLICPAR() functions can be used to access these enabled parameters via a special object name. This special object name is composed of 2 parts which are arranged consecutively with a dot in between:

- Special object name = *Part_1*.*Part_2*
  - *Part_1*: Name of the function block used in the RSI context
  - *Part_2*: Object name of the RSI object within the function block

Precondition

- RSI object parameter is enabled in the function block.
- Function block is used in an RSI context.

Procedure

- Create KRL method for the function block parameter.

Example

Enabled RSI object parameter in the function block **Fb_Param**:

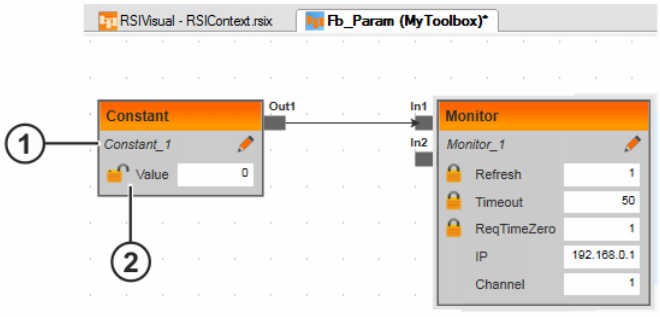


Fig. 7-7: RSI object parameter in the function block Fb_Param

1 RSI object with the name **Contstant_1**
2 RSI object parameter **Value** is enabled

Function block **Fb_Param** in the RSI context:

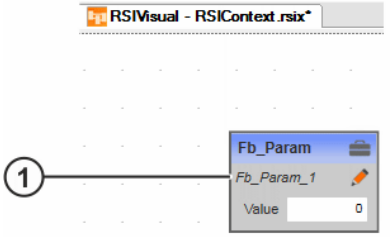


Fig. 7-8: Function block Fb_Param in the RSI context

1 Function block with the name **Fb_Param_1**

KRL method for setting the function block parameter:

```
1 DEF FbParam ( )
2
3 INT Ret
4 INT ContId
5
6 Ret = RSI_Create("RSIContext", ContId)
7 IF (Ret <> RSIOK) THEN
8  ;error handling
```

```
 9  halt
10 ENDIF
11
12 Ret = SetMyFbParam(ContId, 5.67)
13 IF (Ret <> RSIOK) THEN
14  ;error handling
15  halt
16 ENDIF
17
18 ;do rest
19
20 DEFFCT INT SetMyFbParam(ContId :in, Value :in)
21
22 INT ContID
23 REAL Value
24
25 RETURN RSI_SetPublicPar(ContId, "Fb_Param_1.Con-
   stant_1", "Value", Value)
26
27 ENDFCT
```

Line	Description
6	RSI_Create(…) initializes signal processing. The RSI context is loaded into an RSI container.
25	RSI_SetPublicPar(…) assigns a new value to the function block parameter used in the RSI context.

7.6 Help file

The help file with the RSI object descriptions is located in the **Options** catalog in the **Documentation** folder of the RobotSensorInterface option.

8 Examples

Overview

The scope of supply for RobotSensorInterface comprises a server program and various example files:

- Under .DOC\Examples\... on the data storage medium with the installation files of the option
- In the folder **Documentation\Examples**... in the **Options** catalog after installation in WorkVisual

Server program and example files for establishing Ethernet communication with the robot controller:

Folder	Files
…\Ethernet\Server	Server program • TestServer.exe
…\Ethernet	RSI context with KRL program • RSI_Ethernet.rsix • RSI_Ethernet.src
	XML file for Ethernet connection • RSI_EthernetConfig.xml

Further example files:

Folder	Files
…\CircleCorr	RSI context with KRL program • RSI_CircleCorr.rsix • RSI_CircleCorr.src
…\DistanceCtrl	RSI context with KRL program • RSI_DistanceCtrl.rsix • RSI_DistanceCtrl.src
…\Transformations	RSI context with KRL program • RSI_SigTransformation.rsix • RSI_SigTransformation.src

8.1 Integrating the server program and example files

Description

The example files must be copied to the robot controller, e.g.:

- Directly via USB stick
- Via project in WorkVisual
- Via KRC Explorer in WorkVisual (working online)

Target directories

Target directory for KRL programs:

- C:\KRC\ROBOTER\KRC\R1\Program

Target directory for RSI contexts and XML file for Ethernet connection:

- C:\KRC\ROBOTER\Config\User\Common\SensorInterface

Precondition

External system for server program:

- Windows operating system with .NET Framework 3.5 or higher installed

Robot controller:

- User group “Expert”
- T1 or T2 mode

For copying/transferring example files:

- USB stick with the example files

  OR:
- If WorkVisual is used: Network connection to the real robot controller

Procedure

1. Copy the server program onto an external system.
2. Copy the example files via the USB stick into the corresponding target directory on the robot controller or via WorkVisual.

   (>>> *8.1.1 "Transferring files to the controller via a project in WorkVisual" Page 70*)

   (>>> *8.1.2 "Transferring files from WorkVisual to the controller via KRC Explorer" Page 70*)
3. Start the server program on the external system.
4. Press the menu button. The **Server Properties** window is opened.
5. The available IP addresses of the server PC are displayed under Available Network Interfaces.
6. Set the IP address for the connection between robot and PC in the XML file for the Ethernet connection.

8.1.1 Transferring files to the controller via a project in WorkVisual

Procedure

1. Load the active project from the robot controller in WorkVisual.
2. Insert example files into the target directory in the project by dragging them from the **Options** catalog.
3. Transfer the project back to the robot controller.

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

8.1.2 Transferring files from WorkVisual to the controller via KRC Explorer

Procedure

1. Load the active project from the robot controller in WorkVisual.
2. Switch to the workspace **Programming and diagnosis** and establish a connection to the robot controller.
3. Display the **Catalogs** window via the WorkVisual menu.
4. Insert example files into the target directory on the robot controller by dragging them from the **Options** catalog.

5. Update the data on the robot controller.

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

8.2 Server program user interface

The server program enables the connection between an external system and the robot controller to be tested by establishing stable communication with the robot controller.

For this purpose, the received data are evaluated and the current time stamp of the packet is copied to the XML document that is to be sent. The XML document can be transmitted with correction data or zero values.

The server program has the following functions:

- Sending and receiving of data at the sensor cycle rate
- Free Cartesian motion correction using operator control elements
- Displaying the data received
- Displaying the data sent

The test server program and the Windows operating system are not real-time capable. No conclusions can be drawn as to the time response or the stability of the overall system.

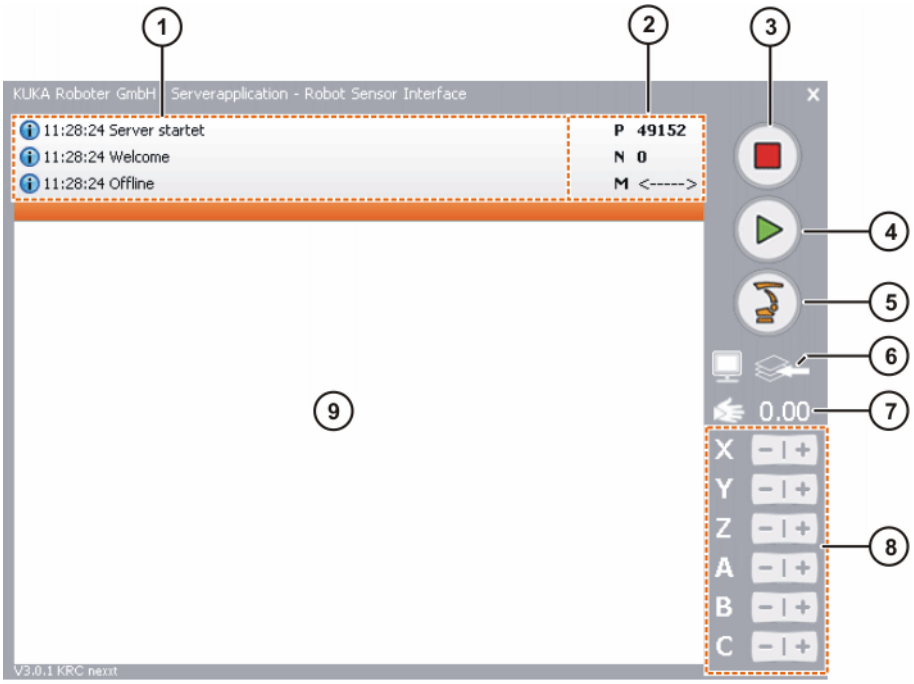


Fig. 8-1: Server program

Item	Description
1	Message window
2	Display of the communication parameters set • **P**: Port number • **N**: Network card index • **M**: Communication mode

Item	Description
	– **<---->**: The server can receive and transmit data. – **<-----**: The server can only receive data.
3	Stop button Communication with the robot controller is terminated and the server is reset.
4	Start button Data exchange between the server program and robot controller is evaluated. The first incoming connection request is linked and used as a communication adapter.
5	Menu button for setting the communication parameters (>>> *8.2.1 "Setting communication parameters in the server program" Page 72*)
6	Display options • Arrow pointing to the left: the received data are displayed. (Default) • Arrow pointing to the right: the sent data are displayed.
7	Hand icon A slider control can be used to set the increment for motion correction per sensor cycle. • **0.00 … 3.33**
8	Buttons for incremental motion correction per sensor cycle. The increment is set using the hand icon.
9	Display window The sent or received data are displayed, depending on the display option set. The displayed data are refreshed at the sensor cycle rate.

8.2.1 Setting communication parameters in the server program

Procedure

1. Click on the menu button in the server program.
   The **Server Properties** window is opened.
2. Set the communication parameters.
3. Close the window.

Description

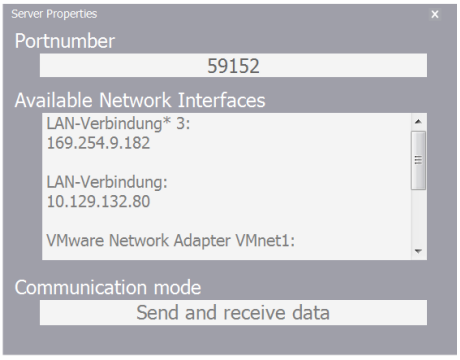


Fig. 8-2: Server Properties window

Element	Description
Portnumber	Enter the port number of the socket connection. The external system awaits the connection request from the robot controller at this port. A free number that is not assigned a standard service must be selected. Default value: **59152** **Note:** When selecting the port, it must be ensured that it is not being used by other services, e.g. by the operating system. Otherwise, no connection can be established via this port.
Available Network Interfaces	Displays the available IP addresses defined on the PC used.
Communication mode	Select communication mode. • **Send and receive data**: The server can receive and transmit data. • **Only receive data**: The server can only receive data. Default value: **Send and receive data**

8.3 RSI_Ethernet example: Cartesian correction via Ethernet

The robot controller receives Cartesian correction data from a sensor and sends them to the robot. The robot is controlled purely by means of corrections on the basis of relative correction values. The reference coordinate system for applying the corrections is the BASE coordinate system.

Program

```
 1 DEF RSI_Ethernet( )
 2 ; ===========================================
 3 ;
 4 ; RSI EXAMPLE: ETHERNET communication
 5 ; Realtime UDP data exchange with server application
 6 ;
 7 ; ===========================================
 8
 9 ; Declaration of KRL variables
10 DECL INT ret ; Return value for RSI commands
```

```
11 DECL INT CONTID ; ContainerID
12
13 INI
14
15 ; Move to start position
16 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
17
18 ; Create RSI Context
19 ret = RSI_CREATE("RSI_Ethernet",CONTID,TRUE)
20 IF (ret <> RSIOK) THEN
21   HALT
22 ENDIF
23
24 ; Start RSI execution
25 ret = RSI_ON(#RELATIVE)
26 IF (ret <> RSIOK) THEN
27   HALT
28 ENDIF
29
30 ; Sensor guided movement
31 RSI_MOVECORR()
32
33 ; Turn off RSI
34 ret = RSI_OFF()
35 IF (ret <> RSIOK) THEN
36   HALT
37 ENDIF
38
39 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
40
41 END
```

Line	Description
16	Start position of the sensor-guided motion
19	RSI_CREATE() loads the RSI context into an RSI container.
25	RSI_ON() activates the signal processing. • Correction mode: Relative correction
31	RSI_MOVECORR() activates the sensor-guided motion.
34	RSI_OFF() deactivates the signal processing.
39	Return to start position

RSI context

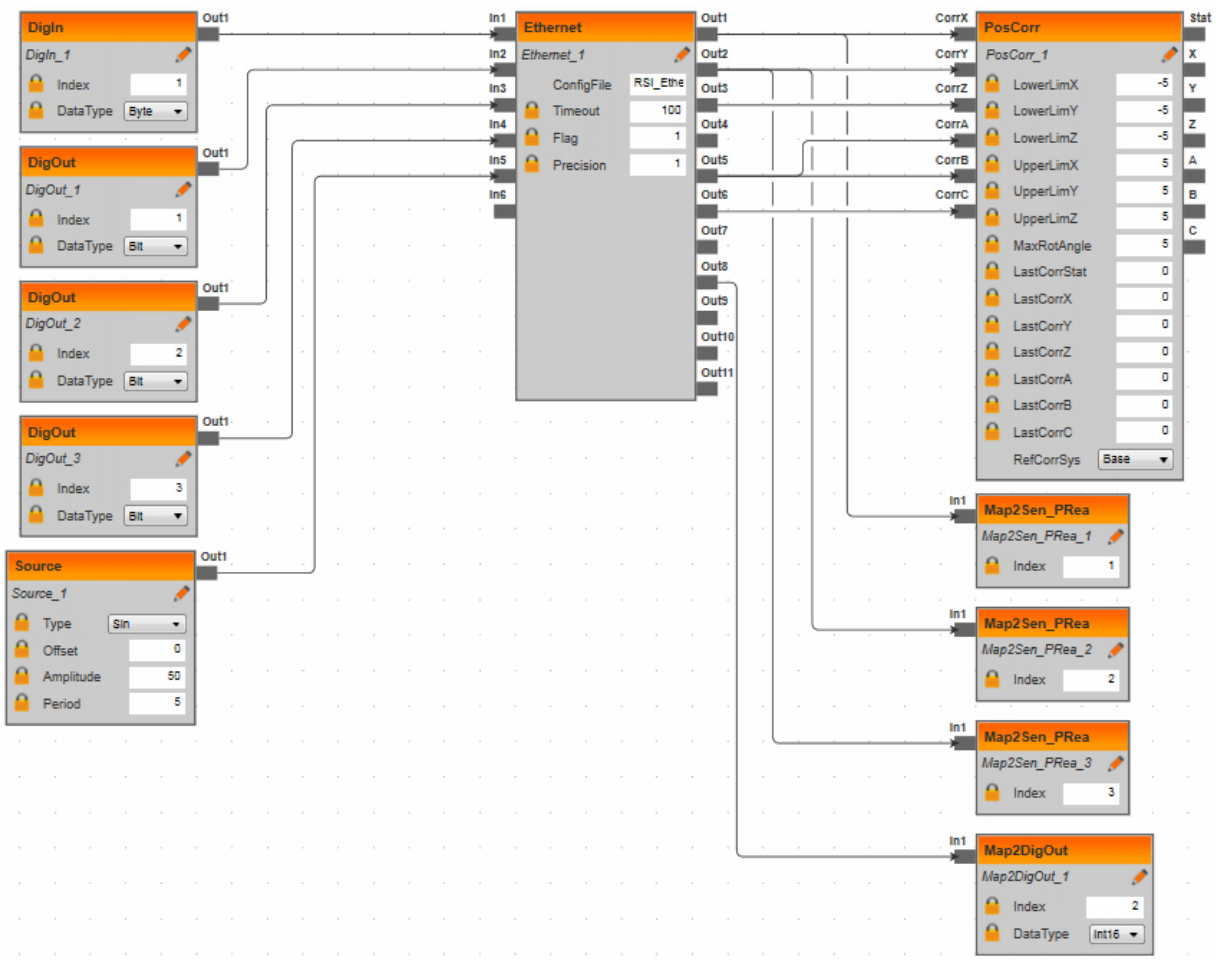


Fig. 8-3: RSI_Ethernet example: Cartesian correction via Ethernet

RSI object	Description
Digin_1	Loads sensor data via 8 digital inputs and transfers them to the Ethernet interface (input 1).
DigOut_1 … Dig-Out_3	Loads robot data via 3 digital outputs and transfers them to the Ethernet interface (inputs 2 to 4).
Source_1	Supplies a periodical sinusoidal signal with an amplitude of 50 every 5 s.
Ethernet_1	Sends the signals arriving at inputs 2 to 5 to the sensor system and receives sensor data back via input 1. The sensor data are available at outputs 1 to 6 for further processing.
PosCorr_1	Loads the sensor data that are available at outputs 1 to 6 of the Ethernet interface and determines the Cartesian correction data.
Map2Sen_PRea_1 … Map2Sen_PRea_1	Writes the Cartesian correction data to system variable $SEN_PREA.
Map2DigOut_1	Accesses the processed signals and sets 16 digital outputs.

8.4 RSI_CircleCorr example: Sensor-guided circular motion

A sensor-guided circular motion is configured. For this purpose, a sinusoidal signal is generated. This signal is loaded into the correction object POSCORR as a sine function in the Z direction and again, with a delay,

as a sine function in the Y direction. Following the first execution of the signal processing, the amplitude of the signal is subsequently modified in the KRL program. When the signal processing is started again with half the amplitude, a smaller circular motion is obtained.

The robot is controlled purely by means of corrections on the basis of the absolute correction values in the Y and Z directions. The reference coordinate system for applying the corrections is the BASE coordinate system. After a defined time, the sensor-guided motion is aborted by means of a timer.

Program

```
 1 DEF RSI_CircleCorr( )
 2 ; ===========================================
 3
 4 ; RSI EXAMPLE: Lissajous circle
 5 ; Create a cirle movement with two sine corrections
 6 ;
 7 ; ===========================================
 8
 9 ; Declaration of KRL variables
10 DECL INT ret ; Return value for RSI commands
11 DECL INT CONTID ; ContainerID
12
13 INI
14
15 ; Move to start position
16 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
17
18 ; Base in actual position
19 $BASE.X=$POS_ACT.X
20 $BASE.Y=$POS_ACT.Y
21 $BASE.Z=$POS_ACT.z
22
23 ; Create RSI Context
24 ret=RSI_CREATE("RSI_CircleCorr",CONTID)
25 IF (ret <> RSIOK) THEN
26   HALT
27 ENDIF
28
29 ; Start RSI execution
30 ret=RSI_ON(#ABSOLUTE)
31 IF (ret <> RSIOK) THEN
32   HALT
33 ENDIF
34
35 ; Sensor guided movement
36 RSI_MOVECORR()
37
38 ; Turn off RSI
39 ret=RSI_OFF()
40 IF (ret <> RSIOK) THEN
41   HALT
42 ENDIF
43
44 ; Modify RSI parameter
45 ret=RSI_GETPUBLICPAR(CONTID,"SOURCE1","Amplitude", fVar)
   ...
49 ret=RSI_SETPUBLICPAR(CONTID,"SOURCE1","Amplitude", fVar/2)
```

```
   ...
54 ; Start RSI execution
55 ret=RSI_ON(#ABSOLUTE)
   ...
60 ; Sensor guided movement
61 RSI_MOVECORR()
62
63 ; Turn off RSI
64 ret=RSI_OFF()
   ...
69 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
70
71 END
```

Line	Description
16	Start point of the sensor-guided motion
19 … 21	Current robot position relative to the base
24	RSI_CREATE() loads the RSI context into an RSI container.
30	RSI_ON() activates the signal processing. • Correction mode: Absolute correction
36	RSI_MOVECORR() activates the sensor-guided motion.
39	RSI_OFF() deactivates the signal processing.
45	RSI_GETPUBLICPAR() reads the currently set amplitude of the signal (SOURCE1).
49	RSI_SETPUBLICPAR() assigns a new value to the amplitude of the signal (SOURCE1). The amplitude is halved.
55	RSI_ON() activates the signal processing. • Correction mode: Absolute correction
61	RSI_MOVECORR() activates the sensor-guided motion.
64	RSI_OFF() deactivates the signal processing.

RSI context

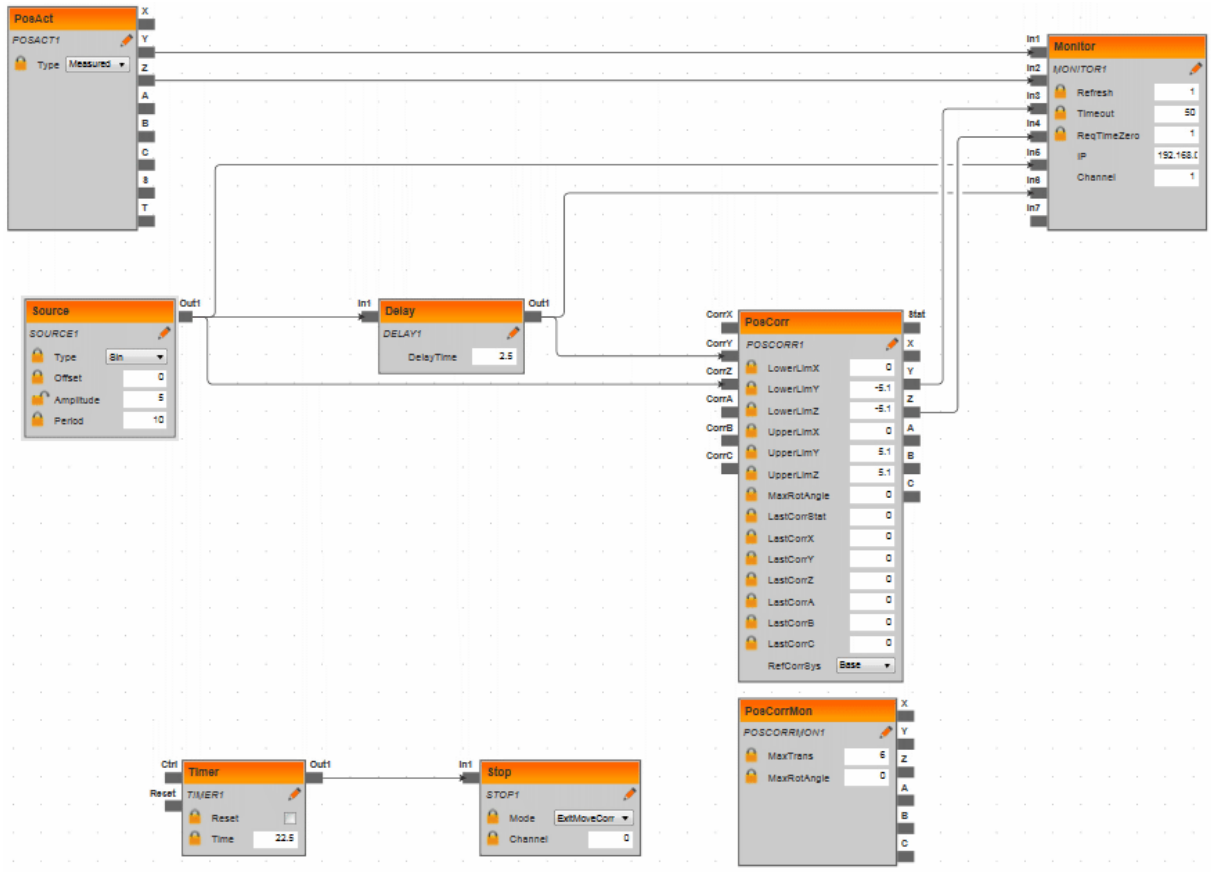


Fig. 8-4: RSI_CircleCorr example: Sensor-guided circular motion

RSI object	Description
TIMER1	Once the time set in the timer has elapsed, the sensor-guided motion is aborted.
STOP1	
POSCORRMON1	Limits the maximum overall Cartesian correction. • Maximum translational deflection in X, Y, Z: 6 mm
SOURCE1	Supplies a periodical sinusoidal signal with an amplitude of 5.0 every 10 s.
DELAY1	The signal is delayed by 2.5 s.
POSCORR1	Loads the sinusoidal correction value in the Z direction and, delayed, the sinusoidal correction value in the Y direction.
POSACT1	Loads the Cartesian actual position of the robot in the Y and Z directions. • Reference coordinate system for applying corrections: BASE
MONITOR1	The following signals are linked to the MONITOR object and can be displayed on the robot controller using the RSI monitor: • Cartesian actual position of the robot in the Y and Z directions [mm]

8.5 RSI_DistanceCtrl example: Path correction for distance control

A defined distance from a workpiece is to be maintained. When signal processing is activated, a sensor measures the distance to the workpiece

and moves 100 mm in the Y direction with a LIN motion. Parallel to this, a relative correction value is determined and the path of the LIN motion in the Z direction is corrected.

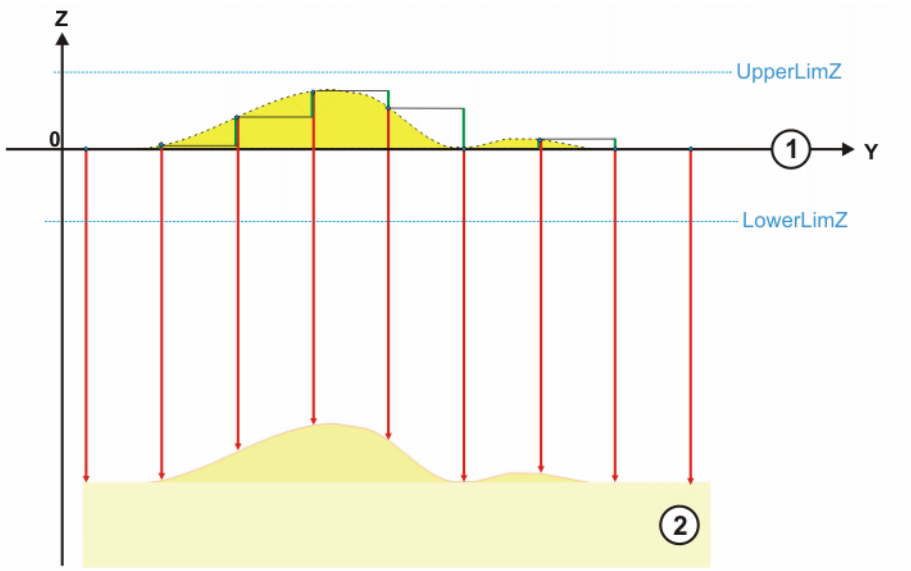


Fig. 8-5: Path correction for distance control

1 Workpiece 2 Programmed path

Program

```
 1 DEF RSI_DistanceCtrl( )
 2 ; ===========================================
 3 ;
 4 ; RSI EXAMPLE: Distance Crtl
 5 ; Move on a LIN path with superimposed corrections
 6 ; Deviation from programmed path is controlled with
 7 ; a analog input $ANIN[1]
 8 ;
 9 ; ===========================================
10
11 ; Declaration of KRL variables
12 DECL INT ret ; Return value for RSI commands
13 DECL INT CONTID ; ContainerID
14
15 INI
16
17 ; Move to start position
18 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
19 $BASE=$POS_ACT
20
21 ; Create signal processing
22 ret=RSI_CREATE("RSI_DistanceCtrl")
23 IF (ret <> RSIOK) THEN
24   HALT
25 ENDIF
26
27 ; Start signal processing in relative correction mode
28 ret=RSI_ON(#RELATIVE)
29 IF (ret <> RSIOK) THEN
30   HALT
31 ENDIF
32
```

```
33 LIN_REL {Y 100
34
35 ; Turn off RSI
36 ret=RSI_OFF()
37 IF (ret <> RSIOK) THEN
38   HALT
39 ENDIF
40
41 END
```

Line	Description
18	Start point of the sensor correction
19	Position of the BASE coordinate system in the current TCP
22	RSI_CREATE() loads the RSI context into an RSI container.
28	RSI_ON() activates the signal processing. • Correction mode: Relative correction
33	Relative LIN motion in Y direction (100 mm)
36	RSI_OFF() deactivates the signal processing.

RSI context

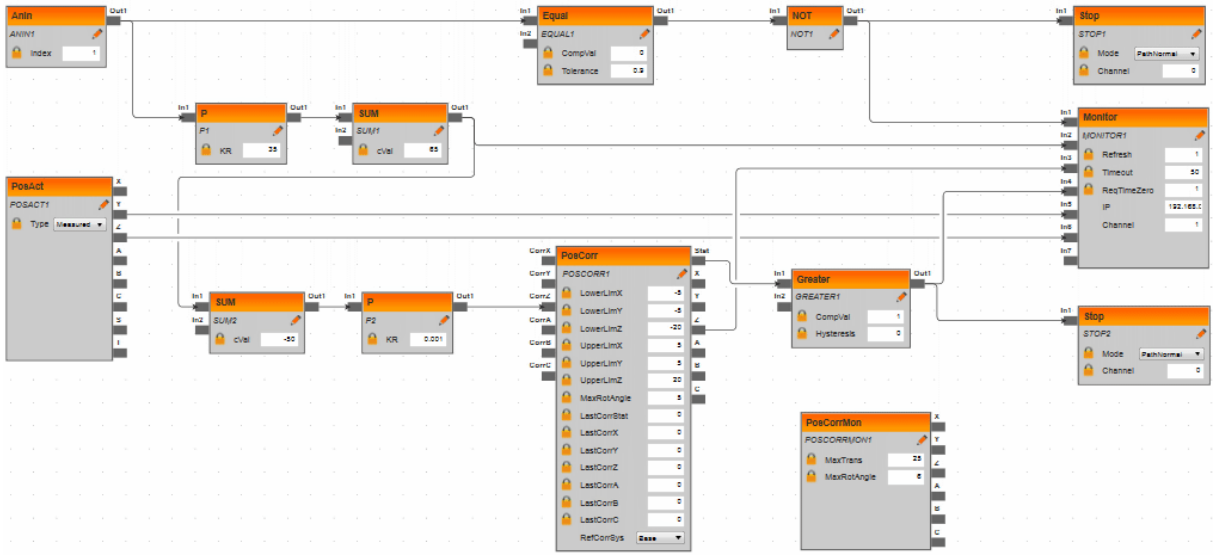


Fig. 8-6: RSI_DistanceCtrl example: Path correction for distance control

RSI object	Description
ANIN1	Loads the sensor signal via an analog input.
EQUAL1	EQUAL is used to check whether the sensor signal is within a tolerance limit. If this is not the case (NOT), the robot is stopped on the programmed path.
NOT1	
STOP1	
P1	P1 is used to convert the sensor signal, e.g. 5 V gives a distance of 10 cm (= actual distance). The actual distance (SUM1) is added to the command distance (SUM2). The result is the correction value in the Z direction in cm. P2 is used to convert the correction value to mm.
SUM1	
SUM2	
P2	
POSCORR1	Loads the calculated correction value in the Z direction that is present as a signal at the output of object P2.

RSI object	Description
	• Reference coordinate system for applying corrections: BASE
POSCORRMON1	Limits the maximum overall Cartesian correction. • Maximum translational deflection in X, Y, Z: 25 mm • Maximum rotational deflection of the angle of rotation: 6°
GREATER1 STOP2	The correction status present at the output “Stat” of the correction object POSCORR is checked. If the correction status >1, i.e. the permissible correction has been exceeded and automatically limited to the maximum correction ±20 mm, the robot is stopped on the programmed path.
POSACT1	Loads the current Cartesian actual position of the robot in the Y and Z directions.
MONITOR1	The following signals are linked to the MONITOR object and can be displayed on the robot controller using the RSI monitor: • Loaded analog sensor signal [V] • Calculated actual distance [cm] • Correction value in the Z direction [mm] • Correction limit (true, false) • Cartesian actual position of the robot in the Y and Z directions [mm]

8.6 RSI_SigTransformation example: Transformation into a new coordinate system

Here, the programming of a transformation of position data acquired by a sensor is described.

In addition to the tool, a sensor is mounted on the mounting flange of the robot. This sensor, e.g. a camera, acquires the position of a workpiece. The sensor data must be transformed from the sensor coordinate system to the BASE coordinate system of the robot controller.

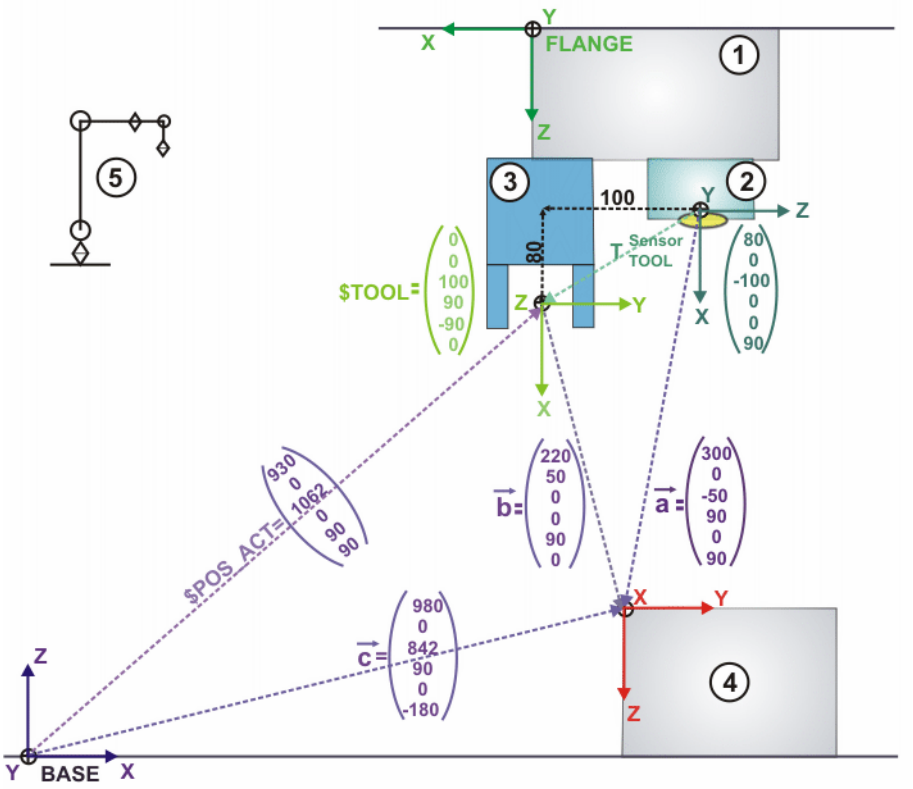


Fig. 8-7: Example of a transformation

1	Mounting flange	4	Workpiece
2	Sensor	5	Robot position
3	Tool		

The sensor acquires the position and orientation of a workpiece in the sensor coordinate system (vector a). In the RSI object TRAFO_USERFRAME, the offset and rotation of the sensor are specified relative to the tool (T_Sensor/Tool). TRAFO_USERFRAME transforms the sensor data to the TOOL coordinate system (vector b). To receive the sensor data in the BASE coordinate system, the RSI object TRAFO_ROBFRAME is used. TRAFO_ROBFRAME transforms the tool coordinates to the BASE coordinate system (vector c).

The sample program can be used to check the numeric example shown in the figure. A KR 16 must be set for this. If the signals linked with the MONITOR object are displayed on the robot controller with RSI monitor, the position and orientation of the workpiece are given in the BASE coordinate system of the robot controller (vector c).

Program

```
 1 DEF RSI_SigTransformation( )
 2 ; ===========================================
 3 ;
 4 ; RSI EXAMPLE: Transformation of coordinates
 5 ; Simulate a sensorsignal in relationship to
 6 ; a flange mounted sensor. Transform the SIGNAL
 7 ; to $BASE coordinates. Show the transformed
 8 ; position in RSIMONITOR
 9 ; ===========================================
10
11 ; Declaration of KRL variables
```

```
12 DECL INT CONTID ; ContainerID
13
14 INI
15
16 ; Move to start position
17 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
18 $TOOL = {X 0, Y 0, Z 100, A 90 ,B -90, C 0}
19 $BASE = $NULLFRAME
20
21 ; Create signal processing
22 IF (RSI_CREATE("RSI_SigTransformation") <> RSIOK) THEN
23   HALT
24 ENDIF
25
26 ; Start signal processing
27 IF (RSI_ON() <> RSIOK) THEN
28   HALT
29 ENDIF
30
31 wait sec 0.012
32
33 ; Turn off RSI
34 IF (RSI_OFF() <> RSIOK) THEN
35   HALT
36 ENDIF
37
38 END
```

Line	Description
17	Start position of the transformation
18	Position of the TOOL coordinate system
19	Position of the BASE coordinate system (NULLFRAME)
22	RSI_CREATE() loads the RSI context into an RSI container.
27	RSI_ON() activates the signal processing. • Correction mode: Relative correction
31	The transformation data are calculated during the wait time.
34	RSI_OFF() deactivates the signal processing.

RSI context

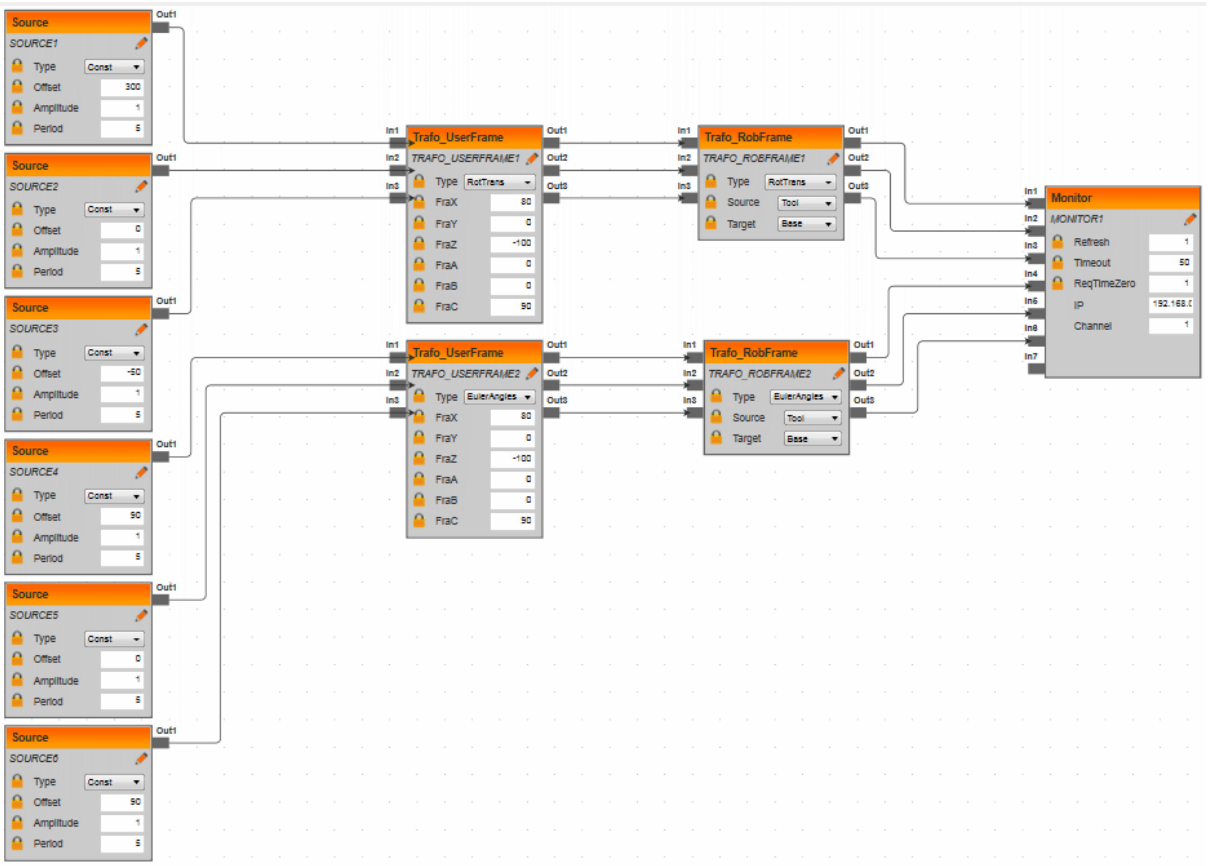


Fig. 8-8: RSI_SigTransformation example: Transformation into a new coordinate system

RSI object	Description
SOURCE1 … SOURCE3	Provide the position of the workpiece in the sensor coordinate system (vector a) and transfer the data to TRAFO_USERFRAME1.
TRAFO_ USERFRAME1	Transforms the position data of the workpiece in the sensor coordinate system to the TOOL coordinate system of the robot controller (vector b). The data are available at the outputs of the object.
TRAFO_ ROBFRAME1	Transforms the position data of the workpiece in the TOOL coordinate system to the BASE coordinate system of the robot controller (vector c). The data are available at the outputs of the object.
SOURCE4 … SOURCE6	Provide the orientation of the workpiece in the sensor coordinate system (vector a) and transfer the data to TRAFO_USERFRAME2.
TRAFO_ USERFRAME2	Transforms the orientation angles of the workpiece in the sensor coordinate system to the TOOL coordinate system of the robot controller (vector b). The data are available at the outputs of the object.
TRAFO_ ROBFRAME2	Transforms the orientation angles of the workpiece in the TOOL coordinate system to the BASE coordinate system of the robot controller (vector c). The data are available at the outputs of the object.
MONITOR1	The following signals are linked to the MONITOR object and can be displayed on the robot controller using the RSI monitor: • Result of the transformation (vector c): position and orientation of the workpiece in the BASE coordinate system of the robot controller

9 Diagnosis

9.1 Overview of RSI monitor

Call

- Select **Display** > **RSI monitor** in the main menu.

Description

The RSI monitor can display and record up to 24 signals from the RSI context. The RSI object MONITOR is used for this in the RSI context. In the RSI context, the signals to be displayed must be linked to the inputs of the MONITOR object.

Fig. 9-1: Overview of the graphical user interface

The following buttons are available:

Button	Description
Setup	The signal properties for the signal recording can be defined.
File	The recorded signal diagram can be saved in a file or a file can be loaded.
Config	The channel number of the RSI object MONITOR can be set. (This is relevant if multiple MONITOR objects are used in the RSI context.) • **1 … 8** Default: **1** This button is not available in the user group “User”.
Zoom	The displayed time frame can be increased or decreased in size using a slide controller. The visible time frame can be shifted by dragging it horizontally in the monitor display window.

9.1.1 Setting signal properties

Description

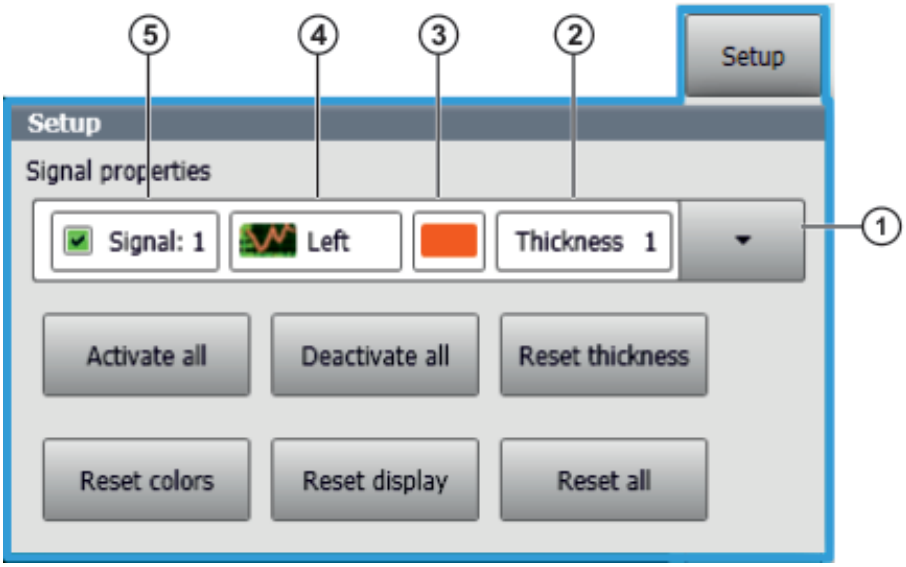


Fig. 9-2: RSI monitor – signal properties

Item	Description
1	List box with the signals 1 … 24
2	Line thickness of the signal • **Line thickness1 … 4** Default: **Line thickness1**
3	Signal color of the signal
4	Ordinate on which the signal is based • **Left**: Display relative to left-hand ordinate • **Right**: Display relative to right-hand ordinate Default: **Left** The scaling of the ordinate is automatically based on the largest signal assigned to it.
5	The check box must be activated to display a signal in the monitor. Default: Check box for **Signal 1 … 6** activated.

The following buttons are available:

Button	Description
Activate all	Activates all signals.
Deactivate all	Deactivates all signals.
Reset thickness	Resets the line thicknesses of the signals to the preset line thickness 1.
Reset colors	Resets the signal colors to the preset colors.
Reset monitor	Resets the time window to the preset size (cancel zoom).
Reset all?	Resets all signal properties to the preset values.

9.1.2 Displaying a signal diagram

Description

Every MONITOR object uses its own channel to the RSI monitor. If multiple MONITOR objects are used in the RSI context, the channel number of the desired MONITOR object must be set for the signal recording. RSI monitor only displays the signals received via the set channel.

Precondition

- IP address in the RSI object MONITOR: 192.168.0.1

Procedure

1. Call RSI monitor and press **Setup**.
2. Set signal properties for the recording.
3. If required, switch to the user group “Expert” and press **Config** to set the channel number of the MONITOR object.
4. Select and execute the program.
   The recording starts when the signal processing is activated and ends when the signal processing is deactivated.

A signal trace is not deleted in the RSI monitor until a new MONITOR object is created in the KRL program. When the program is reset or the signal processing is deleted, the signal trace is retained in the RSI monitor.

9.1.3 Saving a signal trace

Procedure

1. Activate the **File** check box.
2. Enter a file name for the trace in the **Save file** box and press **Save**.
   The trace is saved as a DAT file in the directory C:\KRC\ROBOTER\LOG\SensorInterface\MONITOR.

9.1.4 Loading a signal trace into the monitor

Procedure

1. Activate the **File** check box.
2. Select the desired file in the **Load file** box and press **Load**.
   All traces saved in the directory C:\KRC\ROBOTER\LOG\SensorInterface\MONITOR are available for selection.

9.2 Displaying diagnostic data

Precondition

- User rights: Function group **Diagnostic functions**

Procedure

1. In the main menu, select **Diagnosis** > **Diagnostic monitor**.
2. Select the **RSI diagnosis** module in the **Module** box.

Description

RSI diagnostic data:

Name	Description
Status	Status of the signal processing • **Running (IPO)**: Signal processing in sensor mode #IPO • **Running (IPO_FAST)**: Signal processing in sensor mode #IPO_FAST • **Inactive**: No signal processing
Sensor cycle	Cycle time of the signal processing
Counter	Number of calculation cycles since the start of signal processing
Execution time	Time required for calculation of the current RSI context
Execution time (min)	Minimum time for calculation of the current RSI context
Execution time (max)	Maximum time for calculation of the current RSI context
Execution time (mean)	Average time for calculation of the current RSI context
Object counter	Number of created RSI objects
Memory	Total memory available for RSI (bytes)
Used memory	Memory used (bytes)
Communication cycles	Successful communication cycles since the start of signal processing
Total loss	Number of packet losses since the start of signal processing
Connection quality	Quality of the signal processing • **0 … 100%** 100% = all packets have arrived successfully. 0% = no packet has arrived successfully.
Maximum contiguous loss	Largest contiguous loss of packets since the start of signal processing

9.3 Error log (LOG file)

As standard, the error messages of RSI are logged in a LOG file under C:\KRC\ROBOTER\LOG\SensorInterface.

The LOG level can be modified so that notification messages are also logged.

9.3.1 Configuring the LOG level

The LOG level can be modified in the file C:\KRC\Roboter\Config\User\Common\Logging_RSI.xml.

Precondition

- User group “Expert”
- Operating mode T1 or T2.
- No program is selected.

Procedure

1. Open the file.
2. Modify the LOG level in this line:

```
<Class Name="RSILogger1" LogLevel="error" />
```

3. Save the change.

Description

LogLevel	Description
error	Error messages of the interface are logged.
info	Error messages and notification messages of the interface are logged.

10 Messages

10.1 Information about the messages

The “Messages” chapter contains selected messages. It does not cover all the messages displayed in the message window.

10.2 System messages from module: CrossMeld (KSS)

10.2.1 KSS29000

Message code	KSS29000
Message text	{Type} Permissible overall correction exceeded: RSI is stopped
Message type	Acknowledgement message 
Effect	Maximum braking Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	**Cause:** Permissible correction by RSI correction object too high (>>> Page 90) **Solution:** Further limit permissible correction in RSI correction object (>>> Page 91) **Cause:** Permissible overall correction in RSI context too low (>>> Page 91) **Solution:** Increase permissible overall correction (>>> Page 91)

Cause: Permissible correction by RSI correction object too high

Description

The maximum permissible Cartesian or axis-specific overall correction is defined with the RSI objects POSCORRMON or AXISCORRMON. The correction specifications of the sensor transferred to the RSI context have exceeded the maximum permissible overall correction.

This may be because excessively high corrections are permitted in the RSI correction objects:

- AXISCORR
- POSCORR
- AXISCORREXT

Checking instructions

1. Open the RSI context with RSIVisual and check what values the correction is limited to in the RSI objects AXISCORR, POSCORR and AXISCORREXT (LowerLim/UpperLim).
2. Check whether the application permits further limitation of the values set in the RSI objects.

Information about the individual RSI objects and their object parameters can be found in the RSIVisual help files.

Solution: Further limit permissible correction in RSI correction object

Description

Further limit the permissible correction in the RSI correction object if the application allows, so that the maximum correction is no longer exceeded.

Precondition

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

Procedure

1. Open the RSI context with RSIVisual and modify as required.
2. Save the RSI context and transfer it to the robot controller.

Cause: Permissible overall correction in RSI context too low

Description

The maximum permissible Cartesian or axis-specific overall correction is defined with the RSI objects POSCORRMON or AXISCORRMON. The correction specifications of the sensor transferred to the RSI context have exceeded the maximum permissible overall correction.

Checking instructions

1. Open the RSI context with RSIVisual and check what values are set for the maximum permissible overall correction in the RSI objects POSCORRMON or AXISCORRMON.
2. Check whether the values set in the RSI objects are sufficient for the application.

Information about the individual RSI objects and their object parameters can be found in the RSIVisual help files.

Solution: Increase permissible overall correction

Description

Increase the permissible overall correction in the RSI object to the maximum extent allowed by the application.

Precondition

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

Procedure

1. Open the RSI context with RSIVisual and modify as required.
2. Save the RSI context and transfer it to the robot controller.

10.2.2 KSS29001

Message code	KSS29001
Message text	{Type} Correction outside the permissible range: {Value}
Message type	Notification message 
Effect	No braking reaction No interlock of motions or commands
Possible cause(s)	**Cause:** Correction specification of sensor exceeds permissible range (>>> Page 92) **Solution:** System information: no operator action required (>>> Page 92)

Cause: Correction specification of sensor exceeds permissible range

Description

The commanded correction specification of the sensor exceeds the defined permissible range for the correction object (POSCORR, AXISCORR or AXISCORREXT). The correction is limited to the permissible range.

Solution: System information: no operator action required

Description

This message provides system information to the operator and requires no action.

10.2.3 KSS29002

Message code	KSS29002
Message text	Signal flow ({Mode}): Object {ObjName} returns error {ErrorCode}
Message type	Acknowledgement message 
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	**Cause:** RSI object configured incorrectly (>>> Page 93) **Solution:** Modify the RSI context (>>> Page 93)

Cause: RSI object configured incorrectly

Description

The RSI object specified in the message results in an error and could not be processed correctly.

Checking instructions

1. Open the RSI context with RSIVisual.
2. Check the RSI object that triggered the error, and the input data active for the RSI object.

Information about the individual RSI objects and their object parameters can be found in the RSIVisual help files.

Solution: Modify the RSI context

Precondition

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

Procedure

1. Open the RSI context with RSIVisual and modify as required.
2. Save the RSI context and transfer it to the robot controller.

10.2.4 KSS29004

Message code	KSS29004
Message text	Internal RSI error
Message type	Status message
	
Effect	Path-maintaining EMERGENCY STOP Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	**Cause:** Internal error when executing RSI (>>> Page 93) **Solution:** Reset the program or send RSI LOG file to KUKA Service (>>> Page 94)

Cause: Internal error when executing RSI

Description

An internal error occurred while RSI was running.

All RSI error messages are logged in a LOG file.

LOG file

Directory	C:\KRC\ROBOTER\LOG\SensorInterface\
File	RSIyyyy-MM-dd_HH-mm-ss.log

	Log files are saved with the date and time specified.

Solution: Reset the program or send RSI LOG file to KUKA Service

Description

If the error persists after the program has been reset, contact the responsible KUKA Customer Support team. Customer Support requires the corresponding RSI LOG file.

Precondition

- Program is selected.

Procedure

- Select the menu sequence **Edit** > **Reset program**.

Alternative procedure

- In the status bar, touch the **Robot interpreter** status indicator. A window opens.
  Select **Reset program**.

10.2.5 KSS29006

Message code	KSS29006
Message text	RSI: Signal calculation timeout {CalcTime} usec
Message type	Acknowledgement message 
Effect	Ramp stop Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	**Cause:** RSI context is too extensive (>>> Page 94) **Solution:** Delete objects that are not required from the RSI context (>>> Page 95) **Cause:** RSI context started twice (>>> Page 95) **Solution:** End RSI context first before a restart (>>> Page 95)

Cause: RSI context is too extensive

Description

The RSI context is too extensive and takes too much computing time. The signal calculation could not be completed within the sensor cycle rate.

Checking instructions

- Open the RSI context with RSIVisual and check whether the RSI context contains RSI objects that are not required.

Solution: Delete objects that are not required from the RSI context

Description

RSI objects that are not required can be deleted from the RSI context.

Precondition

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

Procedure

1. Open the RSI context with RSIVisual and modify as required.
2. Save the RSI context and transfer it to the robot controller.

Cause: RSI context started twice

Description

The RSI context is started with the RSI functions `RSI_CREATE()` and `RSI_ON()`, even though it has already been started and not terminated in the meantime.

The procedure for checking whether the RSI context has been started twice is as follows:

Checking instructions

- Check whether `RSI_CREATE()` and `RSI_ON()` are called twice in the program code without `RSI_RESET()` and `RSI_OFF()` being called in between.

Solution: End RSI context first before a restart

Description

Before a restart of the RSI context with the RSI function `RSI_CREATE()` and `RSI_ON()`, first end the RSI context with `RSI_RESET()` and `RSI_OFF()`.

Procedure

1. Select the program in the Navigator and press **Open**. The program is displayed in the editor.
2. Search for the corresponding position in the program and edit it.
3. Close the file and respond to the request for confirmation asking whether the changes should be saved by pressing **Yes**.

10.2.6 KSS29007

Message code	KSS29007
Message text	Error updating RSI signal {Signal name}
Message type	Acknowledgement message

Effect	Maximum braking Input of active commands (robot motions, program start) is blocked.
Possible cause(s)	**Cause:** Connection to the field bus interrupted (>>> Page 96) **Solution:** Restore field bus connection and reconfigure I/O driver (>>> Page 96)

Cause: Connection to the field bus interrupted

Description

The connection to the field bus has been interrupted.

Checking instructions

- Check whether the field bus is connected correctly and supplied with power and that no connecting cable is defective.

The procedure for checking whether a connecting cable is defective is as follows:

Checking instructions

1. Check whether the connectors are correctly connected. Particular attention must be paid to:
   - Pins pushed in
   - Corrosion
   - Scorched contacts
   - Connector insert pushed back
   - Socket pushed back
   - Connector on correct slot
2. Check whether the cable is mechanically damaged. Causes of squashed cables or wires can include the following:
   - Cable straps too tight
   - Clips too tight
   - Trapped when closing a cover
   - Bend radius too tight
3. Check whether the cable still conducts electricity. Particular attention must be paid to:
   - Cross-connection of individual wires
   - Short-circuit of individual wires with the ground conductor
   - Correct wiring in accordance with circuit diagram

Solution: Restore field bus connection and reconfigure I/O driver

Description

Once the field bus connection has been restored, the I/O driver must be reconfigured.

Precondition

- User group “Expert”
- Operating mode T1 or T2.

- No program is selected.

Procedure

1. In the main menu, select **Configuration** > **Inputs/outputs** > **I/O drivers**.
2. Press the **Reconfigure** button.
   It makes no difference whether the **State** or **Configuration** tab is selected.
3. Answer the request for confirmation *Do you really want to reconfigure all I/O drivers?* with **Yes**.
   The message *Reconfiguration in progress ...* is displayed. When the message disappears, reconfiguration is completed.

10.2.7 KSS29008

Message code	KSS29008
Message text	RSI I/Os configuration file not found.
Message type	Status message 
Effect	No braking reaction No interlock of motions or commands
Possible cause(s)	**Cause:** RSI I/O configuration file not found (>>> Page 97) **Solution:** Reload WorkVisual project (>>> Page 98)

Cause: RSI I/O configuration file not found

Description

The configuration file with the RSI I/Os could not be found, e.g. because it has been deleted manually or moved to a different directory.

When the WorkVisual project with the mapped RSI I/Os is transferred to the robot controller, the configuration file is created automatically.

Configuration file

Directory	C:\KRC\ROBOTER\Config\User\Common\
File	RSIIO.xml

The procedure for checking whether the file is present is as follows:

Precondition

- User group Expert

Checking instructions

1. In the Navigator, navigate to the directory in which the file should be located.
2. Check whether the file is present in the directory.

Solution: Reload WorkVisual project

Description

The WorkVisual project must be transferred to the robot controller again.

Precondition

- User group “Expert”
- T1 or T2 mode
- Network connection to the robot controller

Procedure

1. Load the active project from the robot controller.
2. Transfer the project back from WorkVisual to the robot controller and activate it.

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

10.2.8 KSS29009

Message code	KSS29009
Message text	Configuration file of the RSI I/Os is invalid.
Message type	Status message
	
Effect	No braking reaction
	No interlock of motions or commands
Possible cause(s)	**Cause:** RSI I/O configuration file invalid (>>> Page 98) **Solution:** Reload WorkVisual project (>>> Page 98)

Cause: RSI I/O configuration file invalid

Description

The configuration file with the RSI I/Os is invalid, e.g. because the structure of the file has been changed in an inadmissible manner.

When the WorkVisual project with the mapped RSI I/Os is transferred to the robot controller, the configuration file is created automatically.

Configuration file

Directory	C:\KRC\ROBOTER\Config\User\Common\
File	RSIIO.xml

Solution: Reload WorkVisual project

Description

The WorkVisual project must be transferred to the robot controller again.

Precondition

- User group “Expert”
- T1 or T2 mode
- Network connection to the robot controller

Procedure

1. Load the active project from the robot controller.
2. Transfer the project back from WorkVisual to the robot controller and activate it.

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

11 Appendix

11.1 Increasing the memory

The memory may be increased only in consultation with KUKA Roboter GmbH. (>>> *12 "KUKA Service" Page 106*)

Description

If the available memory is insufficient, it is recommended to first check the programming method in KRL as well as the configured signal processing in the RSI context.

Precondition

- Windows interface

Procedure

1. Open the file C:\KRC\ROBOTER\Config\User\Common\RSI.XML.
2. Enter the desired memory capacity in bytes in the <MemSize> element in the <Interface> section.

```
<Interface>
<MemSize>500000</MemSize>
</Interface>
```

3. Save the change and close the file.

11.2 RSI object library

11.2.1 RSI objects for communication

Type	Description
AnIn	Returns the value of an analog input $ANIN.
AnOut	Returns the value of an analog output $ANOUT.
DigIn	Returns the value of a range of digital inputs $IN.
DigOut	Returns the value of a range of digital outputs $OUT
Ethernet	UDP Ethernet communication in XML data format Up to 64 inputs/outputs can be defined in the configuration file. Signals at the inputs are sent to the communication partner. The data received from the communication partner are available at the outputs.
Input	Returns the value of an RSI input or of a range of RSI inputs.
Map2AnOut	Describes an analog output $ANOUT.
Map2DigOut	Describes a digital output $OUT or a range of digital outputs.
Map2Sen_PInt	Changes the value of the system variable $SEN_PINT.
Map2Sen_PRea	Changes the value of the system variable $SEN_PREA.
Monitor	RSI monitor Visualization of up to 24 RSI signals
Output	Describes an RSI output or a range of RSI outputs.

Type	Description
ResetDigOut	Resets a digital output $OUT at a positive edge. The reset output is maintained even at a negative edge.
Sen_PInt	Returns the value of the system variable $SEN_PINT.
Sen_PRea	Returns the value of the system variable $SEN_PREA.
SetDigOut	Sets a digital output $OUT at a positive edge. The set output is maintained even at a negative edge.

11.2.2 RSI objects for logical operations

Type	Description
AND	AND operation Up to 10 input signals can be connected.
BAND	Binary AND operation Combines signal input 1 with a constant value. If a number of signal inputs are linked, these are combined with each other. Up to 10 input signals can be connected.
BCOMPL	Binary complement
BOR	Binary OR operation Combines signal input 1 with a constant value. If a number of signal inputs are linked, these are combined with each other. Up to 10 input signals can be connected.
NOT	Logical negation
OR	OR operation Up to 10 input signals can be connected.
RS FlipFlop	RS flip-flop Reset-dominant flip-flop
SignalSwitch	Switches between 2 signal paths via control signal.
SR FlipFlop	SR flip-flop Set-dominant flip-flop
XOR	Exclusive ORing Up to 10 input signals can be connected.

11.2.3 RSI objects for mathematical operations

Type	Description
ABS	Absolute value function
ACOS	Arc cosine function
ASIN	Arc sine function
ATAN	Arc tangent function
ATAN2	Arc tangent of the quotient of inputs 1 and 2 The quadrant of the result is calculated from the signs of the input signals.

Type	Description
CEIL	Smallest integer greater than or equal to input signal
COS	Cosine function
EXP	Exponential function
FLOOR	Greatest integer greater than or equal to input signal
Limit	Limits the signal to values within a lower and upper limit (LowerLimit, UpperLimit).
LOG	Logarithm function
MinMax	Returns the current smallest and largest signal across all input signals. Up to 10 input signals can be connected.
MULTI	Multiplication of signals
NORM	Absolute value of the vector comprising up to 10 signal inputs
POW	Power function
ROUND	Rounding function
SIN	Sine function
SUB	Subtraction of signals Up to 5 input signals can be connected. A constant value can be subtracted by means of the **Value** parameter.
SUM	Addition of signals Up to 5 input signals can be connected. A constant value can be added by means of the **Value** parameter.
TAN	Tangent function

11.2.4 RSI objects for mathematical comparisons

Type	Description
Equal	Comparison for equality Comparison of signal input 1 with a constant value or comparison of signal inputs 1 and 2 with each other.
Greater	Comparison for greater-than relation Comparison of signal input 1 with a constant value or comparison of signal inputs 1 and 2 with each other.
Less	Comparison for less-than relation Comparison of signal input 1 with a constant value or comparison of signal inputs 1 and 2 with each other.

11.2.5 RSI objects for coordinate transformation

Type	Description
Trafo_RobFrame	Transforms a vector consisting of inputs 1 - 3 from one robot reference coordinate system to another.
Trafo_UserFrame	Transforms a vector consisting of inputs 1 - 3 into a new reference coordinate system with a defined translational and rotational offset.

11.2.6 RSI objects for motion correction

Type	Description
AxisCorr	Axis-specific correction with limitation, robot axes A1 to A6
AxisCorrExt	Axis-specific correction with limitation, external axes E1 to E6
Map2OV_PRO	Changes the program override ($OV_PRO).
PosCorr	Cartesian correction with limitation
Stop	Stops a motion at a positive signal edge. A purely sensor-guided motion with RSI_MOVECORR can be terminated with ExitMoveCorr mode.

11.2.7 RSI objects for correction monitoring

Type	Description
AxisCorrMon	Limitation for the overall axis-specific correction If it is exceeded, the robot program must be reset. The outputs of the object return the current overall correction.
PosCorrMon	Limitation for the overall Cartesian correction If it is exceeded, the robot program must be reset. The outputs of the object return the current overall correction.

11.2.8 RSI objects for reading robot data

Type	Description
AxisAct	Returns the current axis angles of robot axes A1 to A6.
AxisActExt	Returns the current positions of external axes E1 to E6.
GearTorque	Returns the gear torques of robot axes A1 to A6.
GearTorqueExt	Returns the gear torques of external axes E1 to E6.
MotorCurrent	Returns the motor currents of robot axes A1 to A6.
MotorCurrentExt	Returns the motor currents of external axes E1 to E6.
OV_PRO	Returns the current program override $OV_PRO.
PosAct	Returns the current Cartesian robot position.
Status	Returns robot controller status information, e.g. current status of submit or robot interpreter, current operating mode, etc.

11.2.9 RSI objects for signal processing

Type	Description
Constant	Constant Sets a constant value
Cosine	Signal generator for a cosine wave
D	Differentiation object y(k) = B0 * (x(k) - x(k-1)) B0 = KD / <sensor cycle>

Type	Description
Delay	Delays the input signal by a defined time.
GenCtrl	Generic signal processing object up to the 8th order y(z) = B0*u(z) + B1*u(z-1) + B2*u(z-2) +...+ B8*u(z-8) - A1*y(z-1) - A2*y(z-2) -...- A8*y(z-8)
I	Integration object (trapezoid algorithm) y(k) = B0 * (x(k) + x(k-1)) + y(k-1) B0 = <sensor cycle> / (2 * TI)
IIRFilter	IIR FILTER
P	Signal gain
PD	Proportional differential object y(k) = B0 * x(k) + B1 *x(k-1) B0 = KR * (1 + (TV / <sensor cycle>)) B1 = -KR * (TV / <sensor cycle>)
PID	PID object y(k) = y(k-1) + B0 * x(k) + B1 * x(k-1) + B2 * x(k-2) B0 = KR * (1 + TV / <sensor cycle>) B1 = - KR * (1 - <sensor cycle> / TN + 2 * TV / <sensor cycle>) B2 = KR * TV / <sensor cycle>
PT1	1st-order delay object y(k) = - A0 * y(k-1) + B0 * x(k) A0 = -exp(-<sensor cycle> / T1) B0 = KR * (1 - exp(-<sensor cycle> / T1)
PT2	2nd-order delay object y(k) = - A0 * y(k-1) - A1 * y(k-2) + B0 * x(k) + B1 *x(k-1) Case 1: T1 != T2 • Z1 = exp(-<sensor cycle> / T1) • Z2 = exp(-<sensor cycle> / T2) • A0 = -Z1 - Z2 A1 = Z1 * Z2 • B0 = (KP / (T1 - T2)) / (T1 * (1 - Z1) - T2 * (1 - Z2)) • B1 = (KP / (T1 - T2)) / (T2 * Z1 *(1 - Z2) - T1 * Z2 *(1 - Z1)) Case 2: T1 == T2 • Z0 = exp(-<sensor cycle> / T1) • B0 = KP * (1 - Z0 * ((<sensor cycle> / T1) + 1)) • B1 = KP * Z0 * (Z0 + (<sensor cycle> / T1) - 1)
Rectangle	Signal generator for a rectangular wave
Sawtooth	Signal generator for a sawtooth wave
Sine	Signal generator for a sine wave
Source	Signal generator Generates a defined signal curve, e.g. for a constant signal, a sine or cosine signal, etc.
Timer	On expiry of the set time, a positive edge is set at the **Out1** signal output.

Type	Description
Triangle	Signal generator for a triangular wave

12 KUKA Service

12.1 Requesting support

Introduction

This documentation provides information on operation and operator control, and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Information

The following information is required for processing a support request:

- Description of the problem, including information about the duration and frequency of the fault
- As comprehensive information as possible about the hardware and software components of the overall system

  The following list gives an indication of the information which is relevant in many cases:

  – Model and serial number of the kinematic system, e.g. the manipulator
  – Model and serial number of the controller
  – Model and serial number of the energy supply system
  – Designation and version of the system software
  – Designations and versions of other software components or modifications
  – Diagnostic package KRCDiag

    Additionally for KUKA Sunrise: Existing projects including applications

    For versions of KUKA System Software older than V8: Archive of the software (KRCDiag is not yet available here.)
  – Application used
  – External axes used

12.2 KUKA Customer Support

Availability

KUKA Customer Support is available in many countries. Please do not hesitate to contact us if you have any questions.

Argentina

Ruben Costantini S.A. (Agency)
Luis Angel Huergo 13 20
Parque Industrial
2400 San Francisco (CBA)
Argentina
Tel. +54 3564 421033
Fax +54 3564 428877
ventas@costantini-sa.com

Australia

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

Belgium

KUKA Automatisering + Robots N.V.
Centrum Zuid 1031
3530 Houthalen
Belgium
Tel. +32 11 516160
Fax +32 11 526794
info@kuka.be
www.kuka.be

Brazil

KUKA Roboter do Brasil Ltda.
Travessa Claudio Armando, nº 171
Bloco 5 - Galpões 51/52
Bairro Assunção
CEP 09861-7630 São Bernardo do Campo - SP
Brazil
Tel. +55 11 4942-8299
Fax +55 11 2201-7883
info@kuka-roboter.com.br
www.kuka-roboter.com.br

Chile

Robotec S.A. (Agency)
Santiago de Chile
Chile
Tel. +56 2 331-5951
Fax +56 2 331-5952
robotec@robotec.cl
www.robotec.cl

China

KUKA Robotics China Co., Ltd.
No. 889 Kungang Road
Xiaokunshan Town
Songjiang District
201614 Shanghai
P. R. China
Tel. +86 21 5707 2688
Fax +86 21 5707 2603
info@kuka-robotics.cn
www.kuka-robotics.com

Germany

KUKA Deutschland GmbH
Zugspitzstr. 140
86165 Augsburg
Germany
Tel. +49 821 797-1926
Fax +49 821 797-41 1926
Hotline.robotics.de@kuka.com
www.kuka.com

France

KUKA Automatisme + Robotique SAS
Techvallée
6, Avenue du Parc
91140 Villebon S/Yvette
France
Tel. +33 1 6931660-0
Fax +33 1 6931660-1
commercial@kuka.fr
www.kuka.fr

India

KUKA India Pvt. Ltd.
Office Number-7, German Centre,
Level 12, Building No. - 9B
DLF Cyber City Phase III
122 002 Gurgaon
Haryana
India
Tel. +91 124 4635774
Fax +91 124 4635773
info@kuka.in
www.kuka.in

Italy

KUKA Roboter Italia S.p.A.
Via Pavia 9/a - int.6
10098 Rivoli (TO)
Italy
Tel. +39 011 959-5013
Fax +39 011 959-5141
kuka@kuka.it
www.kuka.it

Japan

KUKA Japan K.K.
YBP Technical Center
134 Godo-cho, Hodogaya-ku
Yokohama, Kanagawa
240 0005
Japan
Tel. +81 45 744 7531
Fax +81 45 744 7541
info@kuka.co.jp

Canada

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

Korea

KUKA Robotics Korea Co. Ltd.
RIT Center 306, Gyeonggi Technopark
1271-11 Sa 3-dong, Sangnok-gu
Ansan City, Gyeonggi Do
426-901
Korea
Tel. +82 31 501-1451
Fax +82 31 501-1461
info@kukakorea.com

Malaysia

KUKA Robot Automation (M) Sdn Bhd
South East Asia Regional Office
No. 7, Jalan TPP 6/6
Taman Perindustrian Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 (03) 8063-1792
Fax +60 (03) 8060-7386
info@kuka.com.my

Mexico

KUKA de México S. de R.L. de C.V.
Progreso #8
Col. Centro Industrial Puente de Vigas
Tlalnepantla de Baz
54020 Estado de México
Mexico
Tel. +52 55 5203-8407
Fax +52 55 5203-8148
info@kuka.com.mx
www.kuka-robotics.com/mexico

Norway

KUKA Sveiseanlegg + Roboter
Sentrumsvegen 5
2867 Hov
Norway
Tel. +47 61 18 91 30
Fax +47 61 18 62 00
info@kuka.no

Austria

KUKA CEE GmbH
Gruberstraße 2-4
4020 Linz
Austria
Tel. +43 732 784 752 0
Fax +43 732 793 880
KUKAAustriaOffice@kuka.com
www.kuka.at

Poland

KUKA Roboter CEE GmbH Poland
Spółka z ograniczoną odpowiedzialnością
Oddział w Polsce
Ul. Porcelanowa 10
40-246 Katowice
Poland
Tel. +48 327 30 32 13 or -14
Fax +48 327 30 32 26
ServicePL@kuka-roboter.de

Portugal

KUKA Robots IBÉRICA, S.A.
Rua do Alto da Guerra n° 50
Armazém 04
2910 011 Setúbal
Portugal
Tel. +351 265 729 780
Fax +351 265 729 782
info.portugal@kukapt.com
www.kuka.com

Russia

KUKA Russia OOO
1-y Nagatinskiy pr-d, 2
117105 Moskau
Russia
Tel. +7 495 665-6241
support.robotics.ru@kuka.com

Sweden

KUKA Svetsanläggningar + Robotar AB
A. Odhners gata 15
421 30 Västra Frölunda
Sweden
Tel. +46 31 7266-200
Fax +46 31 7266-201
info@kuka.se

Switzerland

KUKA Roboter CEE GmbH
Linz, Zweigniederlassung Schweiz
Heinrich Wehrli-Strasse 27
5033 Buchs
Switzerland
Tel. +41 62 837 43 20
info@kuka-roboter.ch

Slovakia

KUKA CEE GmbH
organizačná zložka
Bojnická 3
831 04 Bratislava
Slovakia
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Spain

KUKA Iberia, S.A.U.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spain
Tel. +34 93 8142-353
comercial@kukarob.es

South Africa

Jendamark Automation LTD (Agency)
76a York Road
North End
6000 Port Elizabeth
South Africa
Tel. +27 41 391 4700
Fax +27 41 373 3869
www.jendamark.co.za

Taiwan

KUKA Automation Taiwan Co. Ltd.
1F, No. 298 Yangguang ST.,
Nei Hu Dist., Taipei City, Taiwan 114
Taiwan
Tel. +886 2 8978 1188
Fax +886 2 8797 5118
info@kuka.com.tw

Thailand

KUKA (Thailand) Co. Ltd.
No 22/11-12 H-Cape Biz Sector Onnut
Sukhaphiban 2 road, Prawet
Bangkok 10250
Thailand
Tel. +66 (0) 90-940-8950
HelpdeskTH@kuka.com

Czech Republic

KUKA Roboter CEE GmbH
organizační složka
Pražská 239
25066 Zdiby
Czech Republic
Tel. +420 226 212 273
support.robotics.cz@kuka.com

Hungary

KUKA HUNGÁRIA Kft.
Fö út 140
2335 Taksony
Hungary
Tel. +36 24 501609
Fax +36 24 477031
info@kuka-robotics.hu

USA

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

UK

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

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

X