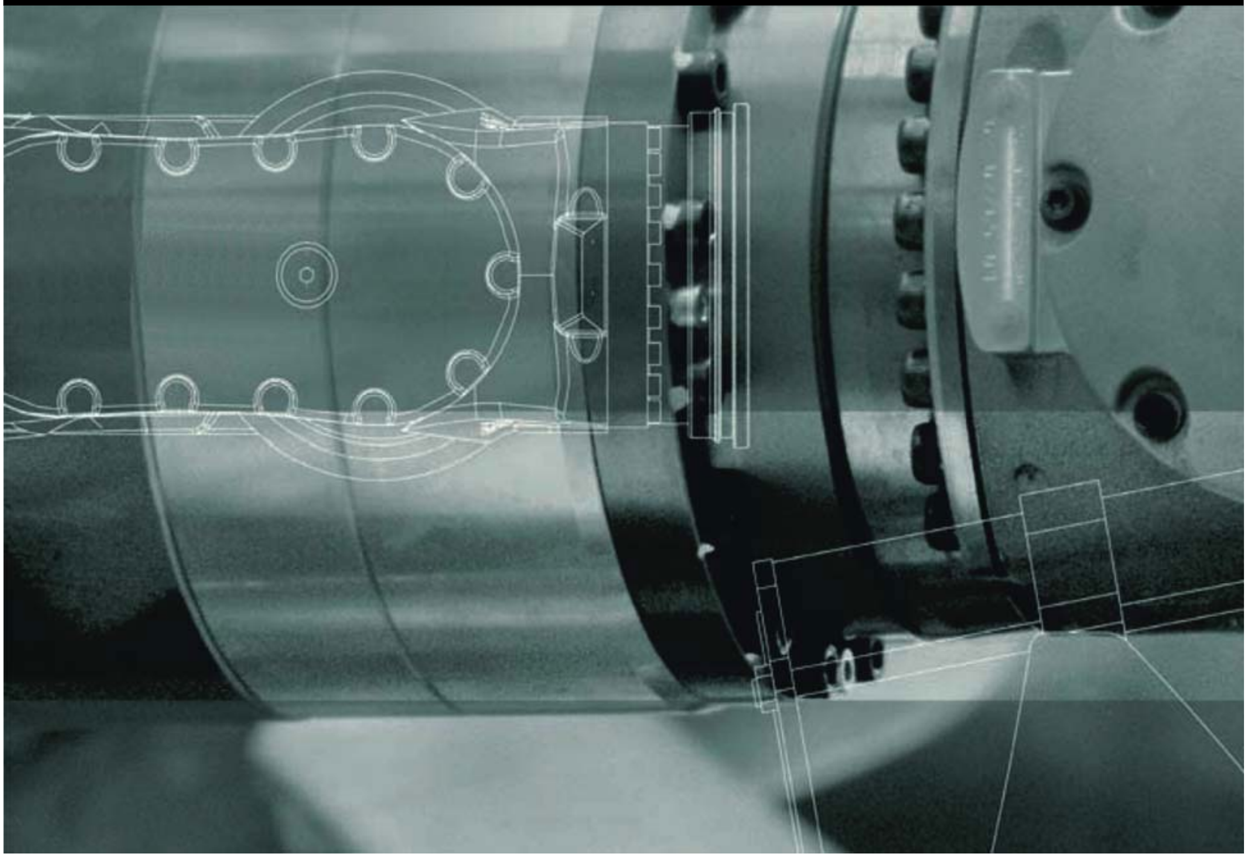
KUKA

KUKA System Technology

KUKA Roboter GmbH

KUKA.RobotSensorInterface 3.3

For KUKA System Software 8.3 and 8.4



Issued: 13.11.2014

Version: KST RSI 3.3 V2

KUKA Roboter GmbH
Zugspitzstraße 140
D-86165 Augsburg
Germany

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

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

Subject to technical alterations without an effect on the function.

Translation of the original documentation

KIM-PS5-DOC

Publication:	Pub KST RSI 3.3 (PDF) en
Book structure:	KST RSI 3.3 V2.1
Version:	KST RSI 3.3 V2

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

Term	Description
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 (IPO mode) or 4 ms (IPO_FAST mode).
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: $\mathbf{X}_{\mathbf{TTS}}$: path tangent $\mathbf{Y}_{\mathbf{TTS}}$: normal vector to the plane derived from the path tangent and the +X axis of the TOOL coordinate system $\mathbf{Z}_{\mathbf{TTS}}$: vector of the right-angled system derived from $\mathbf{X}_{\mathbf{TTS}}$ and $\mathbf{Y}_{\mathbf{TTS}}$ 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 of the data exchange between the devices of a network
IP	Internet Protocol 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

.NET Framework is a trademark of Microsoft Corporation.

Visual Studio is a trademark of Microsoft Corporation.

Windows is a trademark of Microsoft Corporation.

2 Product description

2.1 RobotSensorInterface overview

Functions

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

- Data exchange between robot controller and sensor system.
- Data exchange via Ethernet or the I/O system of the robot controller.
- Cyclical signal processing and evaluation at the sensor cycle rate.
- Influence on the robot motion or program execution by processing sensor signals.
- Configuration of the signal flow (RSI context) with the graphical editor RSI Visual.
- Library with RSI objects for configuration of the signal flow (RSI context).
- Online visualization of the RSI signals (RSI monitor).

Communication

The robot controller can communicate with the sensor system via the I/O system or via Ethernet.

Data exchange via the I/O system:

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

> Signals are linked via a bus system to the I/O system of the robot controller:
>
> - General information about bus management and I/O mapping can be found in the WorkVisual documentation.
> - Detailed information about bus configuration can be found in the bus system documentation.

Data exchange 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)

> 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 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.

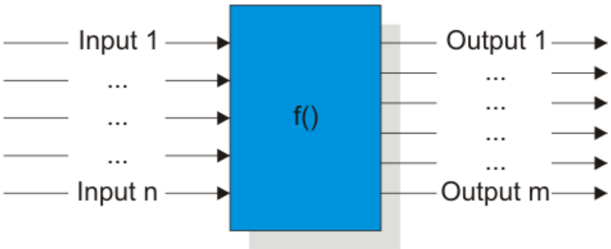


Fig. 2-1: Schematic structure of an RSI object

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.

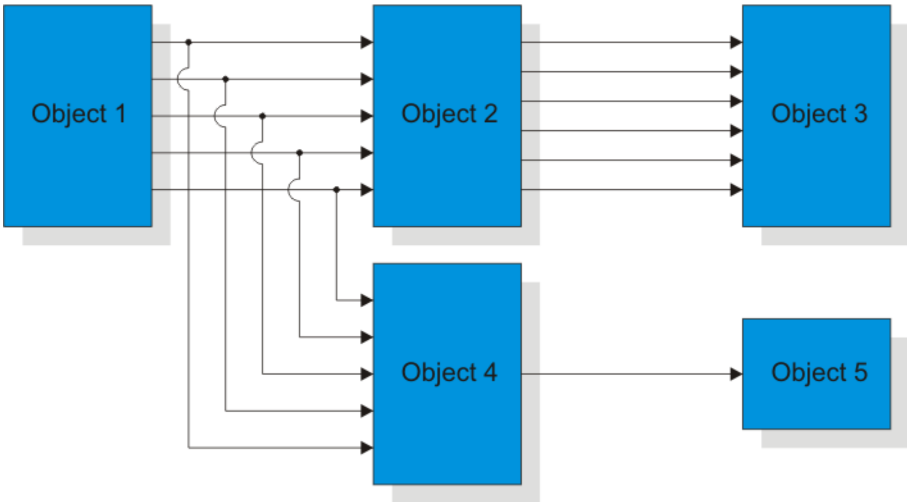


Fig. 2-2: Schematic structure of an RSI context

The RSI context is defined and saved with the graphical editor RSI Visual. 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 (IPO mode) or 4 ms (IPO_FAST mode).

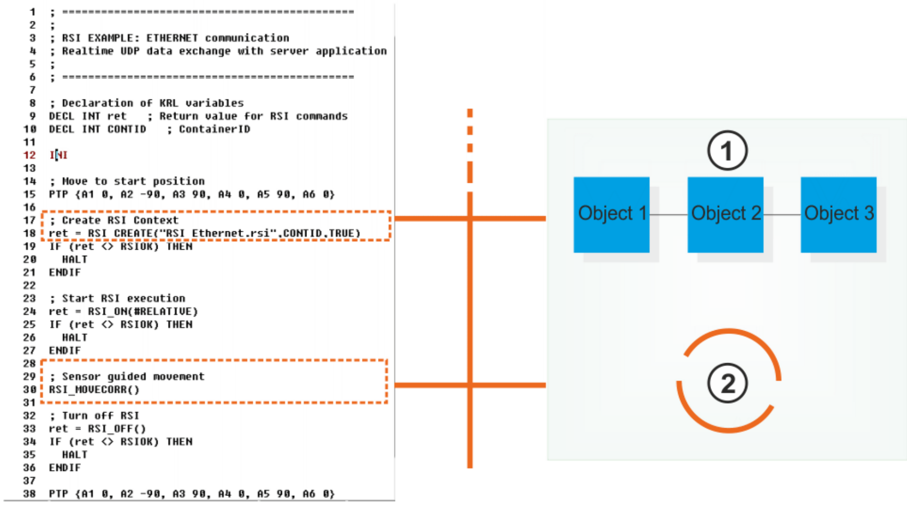


Fig. 2-3: Interaction between KRL program and signal processing

1 RSI context 2 Sensor cycle rate

2.3 Functional principle of data exchange

2.3.1 Data exchange via the I/O system

Description

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 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.

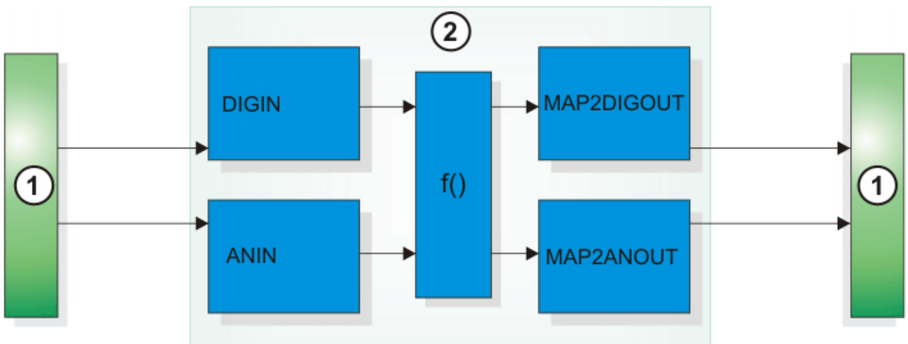


Fig. 2-4: Data exchange via the I/O system

1 I/O system 2 RSI context

2.3.2 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.

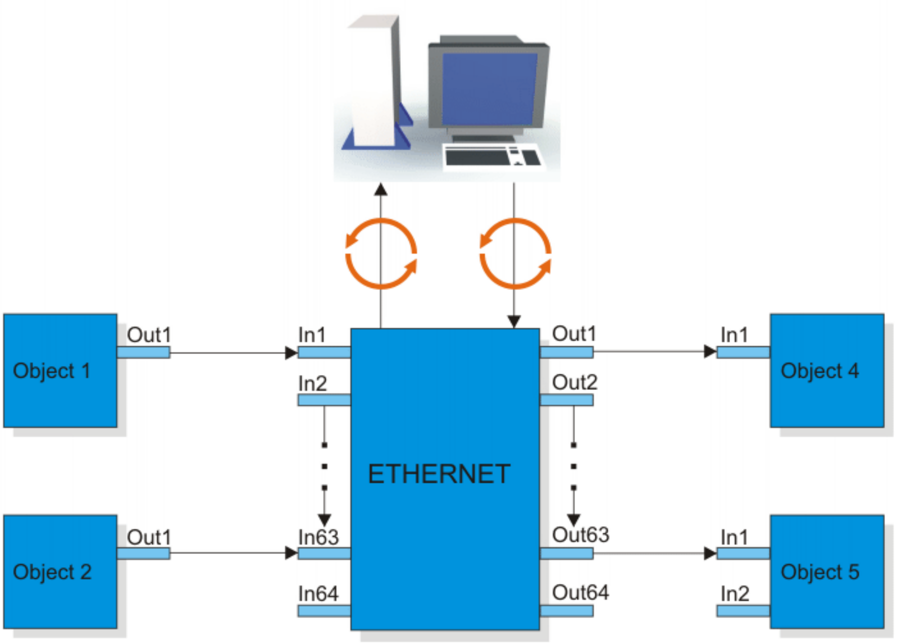


Fig. 2-5: Data exchange via Ethernet (functional principle)

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.

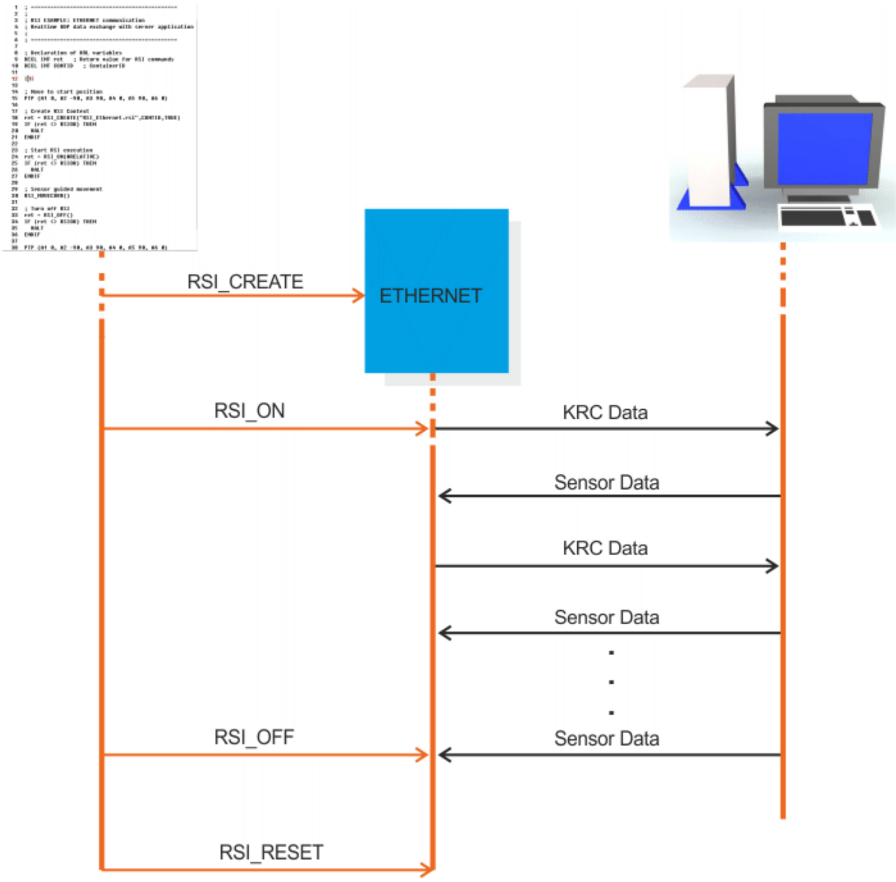


Fig. 2-6: Data exchange via Ethernet (sequence)

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

> Sensor correction cannot be used for asynchronous axes.

Overview

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 following correction types can be configured:

- Motion with superposed sensor correction:
  - Axis angle correction, absolute or relative
  - Cartesian correction, absolute or relative
- Sensor-guided motion:
  - Axis angle correction, absolute or relative
  - Cartesian correction, absolute or relative

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.

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.

If RobotSensorInterface is used in the RoboTeam, it must be ensured that Cartesian sensor corrections from the master robot are not passed on to the slave robots.

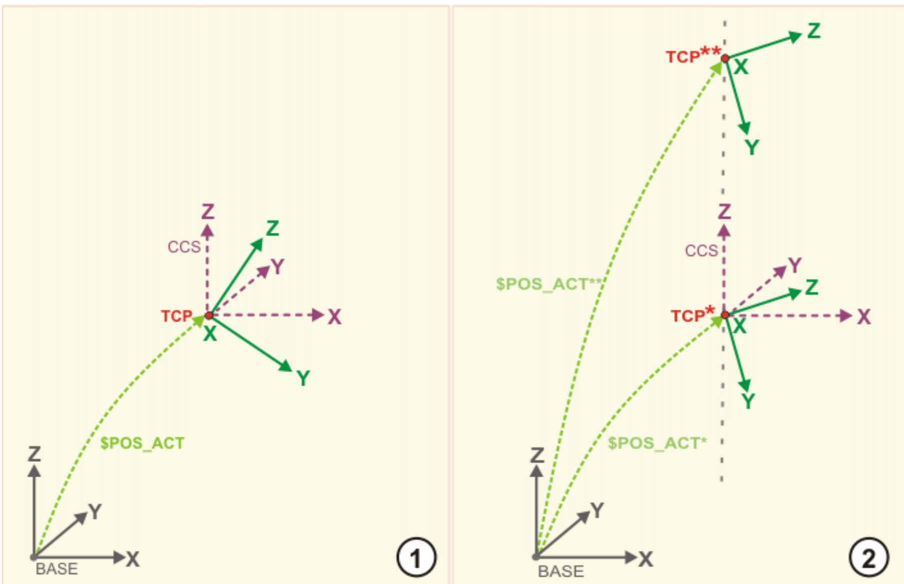


Fig. 2-7: Cartesian correction relative to BASE

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.

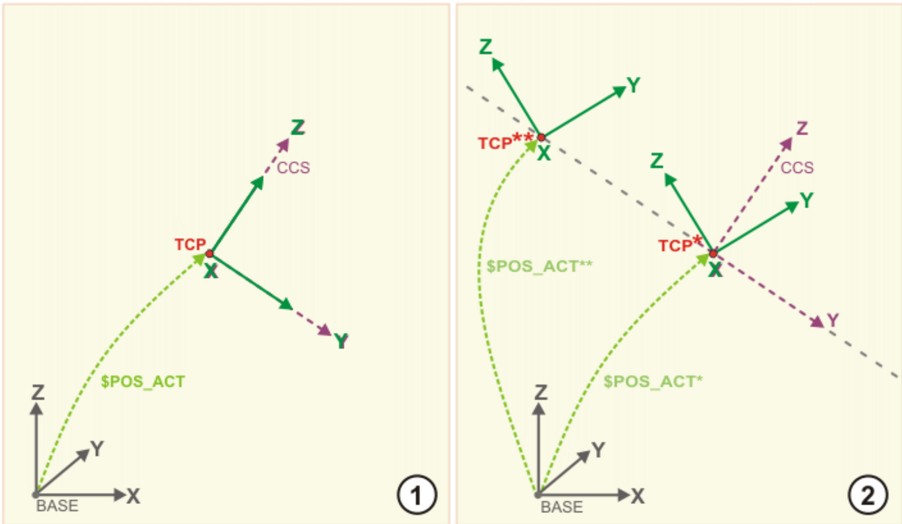


Fig. 2-8: Cartesian correction relative to TOOL

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**: the TCP is offset in the -Y direction and rotated about +C in the correction coordinate system.

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.

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.

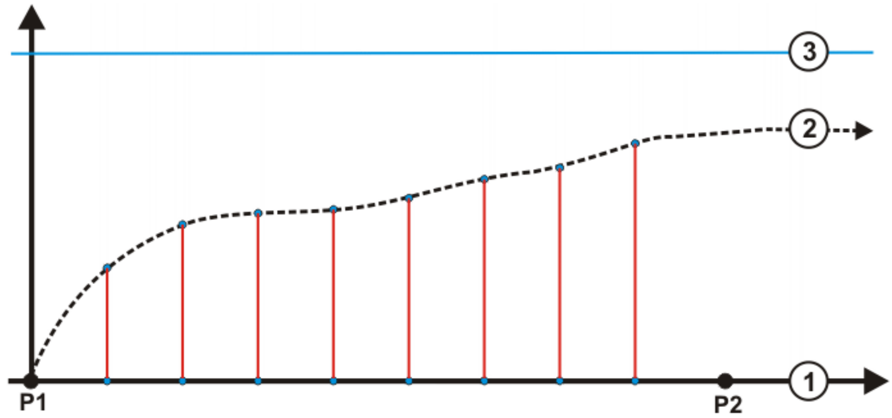


Fig. 2-9: Path correction based on absolute values

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

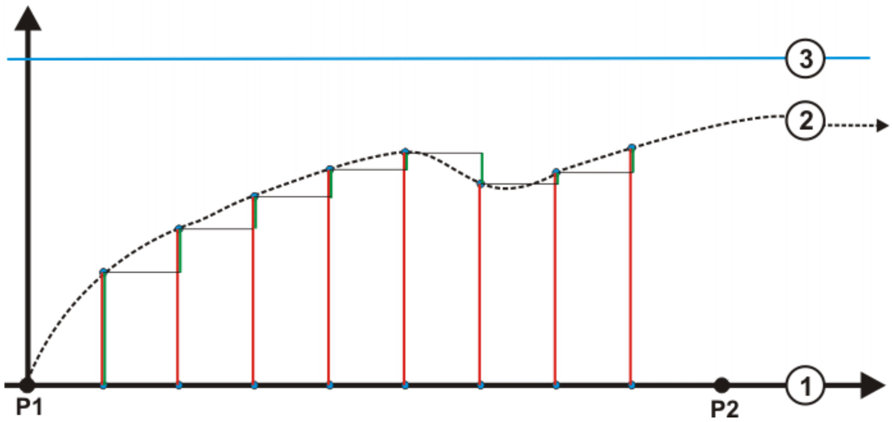


Fig. 2-10: Path correction based on relative values

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.

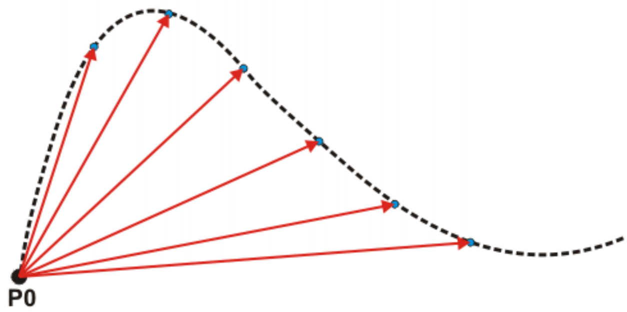


Fig. 2-11: Sensor-guided motion based on absolute values

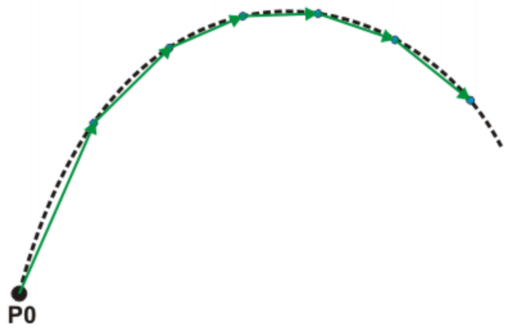


Fig. 2-12: Sensor-guided motion based on relative values

> 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.

3 Safety

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

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

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

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

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 the I/O system: bus system, e.g. Profinet
- External PC for signal flow configuration with RSI Visual

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 75)

Software

Robot controller:

- KUKA System Software 8.3 or 8.4

External PC:

- Windows operating system with .Net Framework 3.5 including Service Pack 1

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 technology packages:
  - KUKA.ConveyorTech
  - KUKA.ServoGun TC
  - KUKA.ServoGun FC
  - KUKA.EqualizingTech
- If RobotSensorInterface and KUKA.RoboTeam are installed on the same robot controller, it must be ensured that Cartesian sensor corrections from the master robot are not passed on to the slave robots.

4.2 Installing or updating RobotSensorInterface

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

Precondition

- Software on KUKA.USBData stick
- No program is selected.
- T1 or T2 operating mode
- “Expert” user group

NOTICE Only the KUKA.USB data stick may be used. Data may be lost or modified if any other USB stick is used.

Procedure

1. Plug in USB stick.
2. Select **Start-up** > **Install additional software** in the main menu.
3. Press **New software**. If a software package that is on the USB stick is not displayed, press **Refresh**.
4. Select the entry **RSI** and press **Install**. Reply to the request for confirmation with **Yes**. The files are copied onto the hard drive.
5. Repeat step 4 if another software package is to be installed from this stick.
6. Remove USB stick.
7. It may be necessary to reboot the controller, depending on the additional software. In this case, a corresponding prompt is displayed. Confirm with **OK** and reboot the robot controller. Installation is resumed and completed.

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

4.3 Uninstalling RobotSensorInterface

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

Precondition

- "Expert" user group

Procedure

1. Select **Start-up** > **Install additional software** in the main menu. All additional programs installed are displayed.
2. Select the entry **RSI** and press **Uninstall**. Reply to the request for confirmation with **Yes**. Uninstallation is prepared.
3. Reboot the robot controller. Uninstallation is resumed and completed.

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

4.4 Installing RSI Visual on an external PC

Preparation

- Copy the **RSIVisual** folder to the external PC:
  - From KUKA.USB data stick
  - Or from the directory D:\KUKA_OPT\RSI on the robot controller if the software is pre-installed

Precondition

- Local administrator rights

Procedure

1. Start the program **setup.exe** in the folder **RSIVisual**.
2. An installation wizard for RSI Visual opens. Follow the instructions in the installation wizard.
3. RSI Visual is installed by default in the folder C:\Program Files\KUKA Roboter GmbH\RSIVisual.

   If desired, select a different directory.
4. Once installation is completed, click on **Close** to close the installation wizard.

4.5 Uninstalling RSI Visual

Precondition

- Local administrator rights

Procedure

1. In the Windows Start menu, select **Settings** > **Control Panel** > **Software**, and delete the entry **RSIVisual**.
2. In the directory C:\Program Files\KUKA Roboter GmbH, delete the folder **RSIVisual**.

5 Configuration

5.1 Network connection via the KLI of the robot controller

Description

A network connection must be established via the KLI of the robot controller in order to exchange data via Ethernet. For this, RSI requires its own Ethernet sensor network 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.

5.2 Configuring the Ethernet sensor network

Precondition

- “Expert” user group
- Network connection via the KLI of the robot controller

Procedure

1. In the main menu, select **Start-up** > **Service** > **Minimize HMI**.
2. Select **All Programs** > **RSI-Network** in the Windows Start menu.

   The **Network Setup** window appears. The network connections already set up are displayed in the tree structure under **Other Installed Interfaces**.
3. Select the entry **New** under **RSI Ethernet** in the tree structure and press **Edit**.
4. Enter the IP address and confirm with **OK**.

> The IP address range 192.168.0.x is blocked for the configuration of the network connection. The entered IP address must be in a separate subnet. The address must not be in the address range of another KLI subnet.

5. Reboot the robot controller with a cold restart.

5.3 Modifying global variables in RSI.DAT

Global variables are defined in the file KRC:\R1\TP\RSI\RSI.DAT. Only the variables described here can be modified.

Precondition

- “Expert” user group

Description

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

ENDDAT
```

Variable	Description
RSIERRMSG	TRUE = errors during execution of RSI commands 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).

6 Operation

6.1 Overview of RSI Visual user interface

Depending on the selection made during installation, the user interface is available in the following languages:

- German
- English

Not all elements on the graphical user interface are visible by default, but they can be shown or hidden as required.

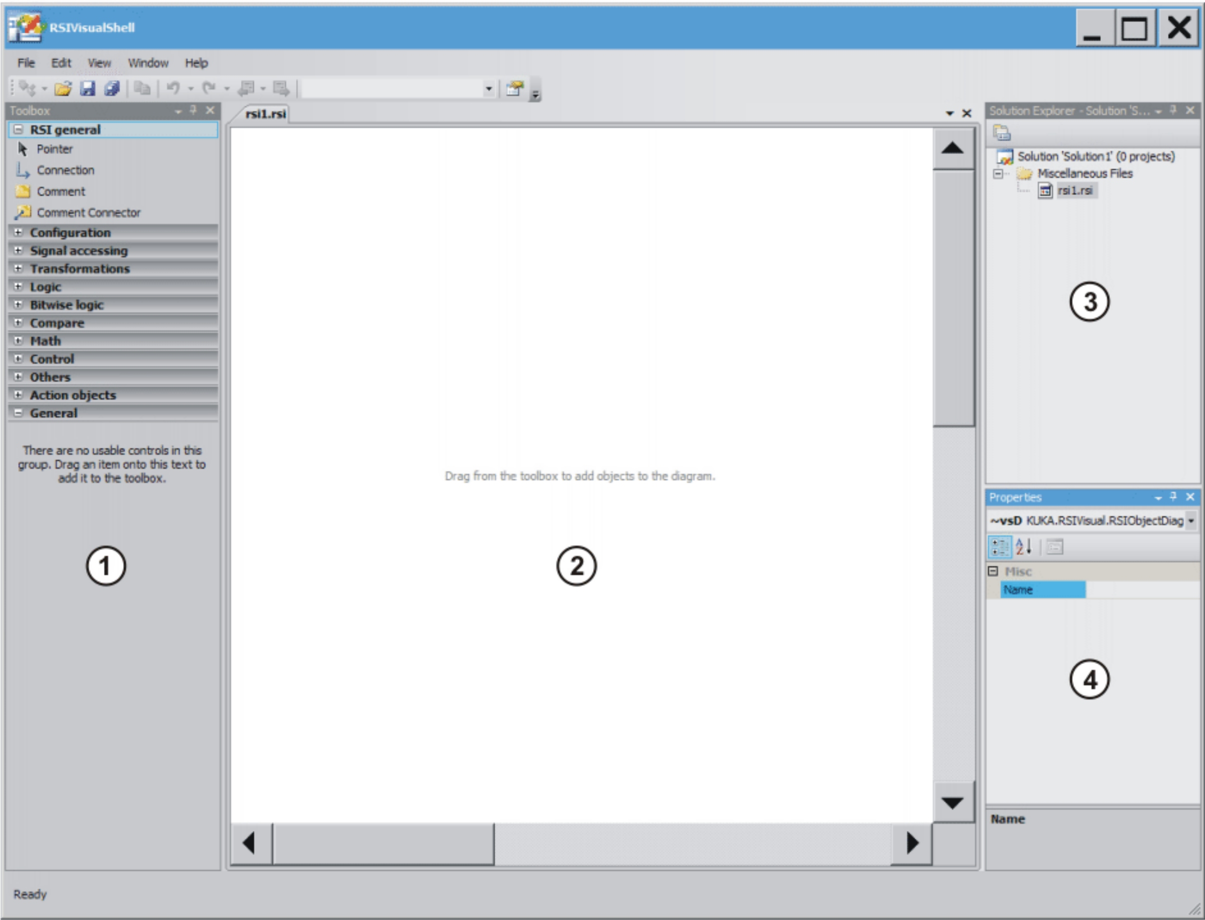


Fig. 6-1: Overview of the graphical user interface

Item	Description
1	**Toolbox** window Contains all the tools and RSI objects required for configuration of the RSI context. The RSI objects can be dragged into the signal flow editor. Tools under **RSI general**: ■ **Comment**: a comment object can be dragged into the editor. ■ **Comment Connector**: a comment object can be linked to the corresponding RSI object. A description of the RSI objects can be found in the appendix. (>>> 11.2 "RSI object library" Page 69)
2	Signal flow editor The signal flow configuration is created here.
3	**Solution Explorer** window All loaded files are displayed in this window in a tree structure.
4	**Properties** window If an RSI object, an RSI object parameter or a signal input/output is selected in the signal flow editor, its properties are displayed. Individual properties or parameters can be changed.

6.1.1 Opening the signal flow editor

Procedure

1. Select the menu sequence **File** > **New** > **File...**.
2. Load the **rsi** template with **Open**.

   A blank document is available for the signal flow configuration.

6.1.2 Linking signal inputs and outputs

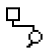
Description

The signal flow is configured using RSI objects that are dragged into the signal flow editor and linked together by means of the object-specific signal inputs and outputs. A signal output can be linked to more than one signal input.

Procedure

1. Point to the desired object output with the mouse pointer.
2. Once the link icon is displayed on the output, click on it and point to the desired object input with the mouse pointer.
3. Once the link icon is displayed on the input, click on it again.

Icons

Icon	Description
	Link icon on the signal output
	Link icon on the signal input

6.1.3 Inserting and linking a comment

Procedure

1. Drag a comment object into the editor.
2. Select the text box and enter the comment.
3. Select the **Comment Connector** tool in the **Toolbox**.
4. Point to the comment with the mouse pointer.
5. Once the link icon is displayed on the comment, click on it and point to the desired RSI object with the mouse pointer.

6. Once the link icon is displayed on the RSI object, click on it again.

Icons

Icon	Description
	Link icon on the comment
	Link icon on the RSI object

6.1.4 Setting an RSI object parameter

Procedure

1. Select an RSI object parameter in the signal flow editor.
   The properties of the parameter are displayed in the **Properties** window.
2. Enter or select the desired value in the **Value** box.

6.1.5 Enabling an RSI object parameter

Description

It is possible to read the value of an RSI object parameter in the KRL program and subsequently assign a new value to the object parameter.

(>>> 7.3.2 "Modifying the signal flow parameters in KRL" Page 42)

A precondition for this is that the parameter has been enabled in the signal flow configuration.

Procedure

1. Select an RSI object parameter in the signal flow editor.
   The properties of the parameter are displayed in the **Properties** window.
2. Set the **IsPublic** box to **True**.

> Setting parameters for specific objects during the runtime of the RSI is not permissible. In such a case, the ‘IsPublic’ box cannot be set to ‘True’.

6.1.6 Saving the signal flow configuration

Description

The following files are generated when the signal flow configuration is saved:

- **<File name>.rsi**: signal flow configuration from RSI Visual
- **<File name>.rsi.diagram**: signal flow layout from RSI Visual according to XML schema
- **<File name>.rsi.xml**: XML file for signal processing on the robot controller

> The RSI, DIAGRAM and XML files form a unit and must be transferred to the robot controller together.
> Target directory:
> C:\KRC\Roboter\Config\User\Common\SensorInterface

Procedure

1. Select the menu sequence **File** > **Save <File name>.rsi** or **Save <File name>.rsi as…**.
2. Give the configuration a name and save it in the desired directory with **Save**.

6.1.7 Loading the signal flow configuration

Procedure

1. Select the menu sequence **File** > **Open** > **File...**.
2. Load the desired RSI file with **Open**.

6.2 Overview of RSI monitor user interface

Call

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

Description

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

Fig. 6-2: 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.

6.2.1 Setting signal properties

Description

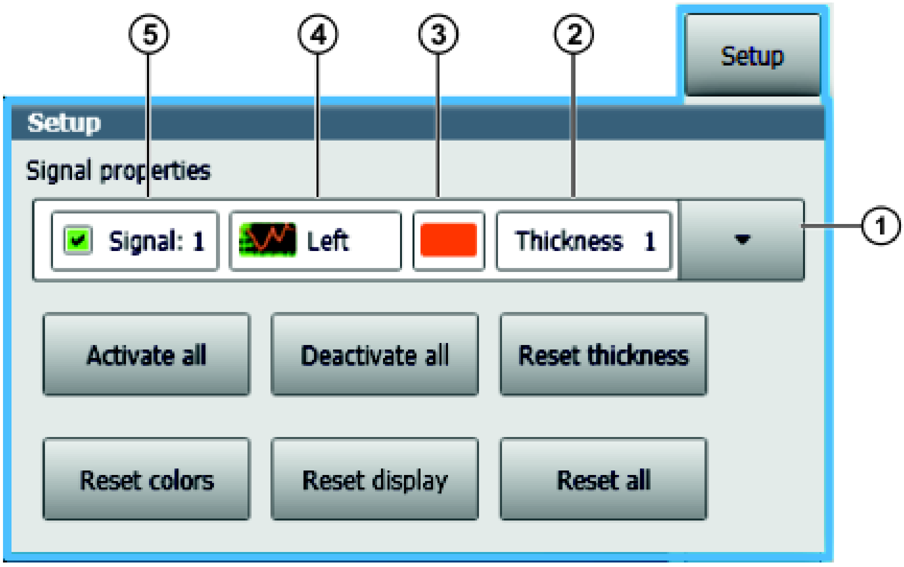


Fig. 6-3: RSI monitor – signal properties

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.

6.2.2 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.

6.2.3 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.

6.2.4 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.

7 Programming

7.1 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 queried 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()	(>>> 7.1.2 "RSI_CREATE()" Page 35)
RSI_DELETE()	(>>> 7.1.3 "RSI_DELETE()" Page 36)
RSI_ON()	(>>> 7.1.4 "RSI_ON()" Page 36)
RSI_OFF()	(>>> 7.1.5 "RSI_OFF()" Page 37)
RSI_MOVECORR()	(>>> 7.1.6 "RSI_MOVECORR()" Page 37)
RSI_GETPUBLICPAR()	(>>> 7.1.7 "RSI_GETPUBLICPAR()" Page 38)
RSI_SETPUBLICPAR()	(>>> 7.1.8 "RSI_SETPUBLICPAR()" Page 38)
RSI_RESET()	(>>> 7.1.9 "RSI_RESET()" Page 39)
RSI_CHECKID()	(>>> 7.1.10 "RSI_CHECKID()" Page 39)
RSI_ENABLE()	(>>> 7.1.11 "RSI_ENABLE()" Page 39)
RSI_DISABLE()	(>>> 7.1.12 "RSI_DISABLE()" Page 40)

7.1.1 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`

7.1.2 RSI_CREATE()

Description

RSI_CREATE() creates an RSI container and loads the signal flow configured with RSI Visual 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 flow link
File:IN	Type: CHAR array Name of the signal configuration: <*File name*>.rsi
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

7.1.3 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

7.1.4 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 IPO_FAST mode. 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 Reference coordinate system for the sensor correction (only relevant if sensor mode = #IPO) ■ #BASE ■ #TCP ■ #TTS ■ #WORLD Default: #BASE

7.1.5 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

7.1.6 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

7.1.7 RSI_GETPUBLICPAR()

Description

The parameter value of an RSI object can be read with RSI_GETPUBLICPAR(). A precondition is that the object parameter has been enabled in the RSI context.

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

7.1.8 RSI_SETPUBLICPAR()

Description

A new value can be assigned to the parameter of an RSI object with RSI_SETPUBLICPAR(). A precondition is that the object parameter has been enabled in the RSI context.

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

7.1.9 RSI_RESET()

Description RSI_RESET() deletes the signal processing and all RSI objects.

Syntax *RET*=RSI_RESET()

Explanation of the syntax

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

7.1.10 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

7.1.11 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

7.1.12 RSI_DISABLE()

Description

RSI_DISABLE() deactivates an RSI container.

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

Syntax

RET=RSI_DISABLE(*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

7.2 Response of the RSI commands

7.2.1 RSI_ENABLE() / RSI_DISABLE() response

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.

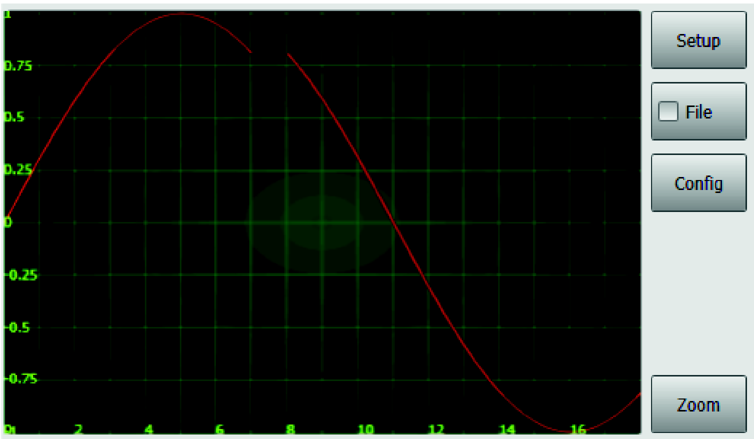


Fig. 7-1: Paused RSI context

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.rsi",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 can lose their values. This can lead to an unexpected response when restarting the context with RSI_ENABLE(contId).

7.2.2 RSI_ON() / RSI_OFF() response

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.

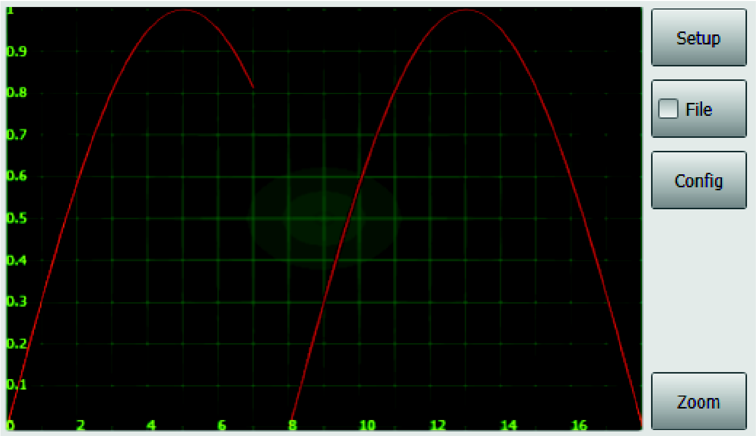


Fig. 7-2: Restarting the context

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.rsi",cont)
 ret=RSI_ON()
 wait sec 7
 ret=RSI_OFF()
 wait sec 1
 ret=RSI_ON()
 Wait sec 10
 ret=RSI_OFF()
END
```

7.3 Programming signal processing

Overview

Step	Description
1	Configure signal flow with RSI Visual.
2	Transfer signal flow configuration (3 files) to the robot controller. Target directory: C:\KRC\Roboter\Config\User\Common\SensorInterface
3	Integrate signal flow into the KRL program. (>>> 7.3.1 "Integrating the signal flow into the KRL program" Page 42)

7.3.1 Integrating the signal flow into the 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("test.rsi")
 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 signal flow configuration 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.

7.3.2 Modifying the signal flow parameters in KRL

Description

Signal flow parameters can be modified subsequently by means of the following functions in the KRL program.

- RSI_GETPUBLICPAR(): reads the configured value of an RSI object parameter.
- RSI_SETPUBLICPAR(): assigns a new value to the RSI object parameter.

Precondition

- RSI object parameter is enabled.

Example

(>>> 8.1.5 "Example of a sensor-guided circular motion" Page 56)

7.4 Configuring an XML file for the Ethernet connection

Overview

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.

> RSI Visual includes the template **RSIEthernet** (menu sequence **File** > **New** > **File…**). The template can be used to configure the Ethernet connection.

The name of the configuration file is specified in the ETHERNET object of the signal flow configuration and read during initialization of the signal processing in the KRL program.

```
<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 (>>> 7.4.1 "XML structure for connection properties" Page 43)
<SEND> … </SEND>	Configuration of the transmission structure (>>> 7.4.2 "XML structure for data transmission" Page 44)
<RECEIVE> … </RECEIVE>	Configuration of the reception structure (>>> 7.4.3 "XML structure for data reception" Page 45)

7.4.1 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**
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>
```

7.4.2 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). (>>> 7.4.4 "Configuration according to XML schema" Page 47)
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. (>>> 7.4.5 "Keywords – reading data" Page 48)
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 48)

7.4.3 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). (>>> 7.4.4 "Configuration according to XML schema" Page 47)
TYPE	Data type of the element ■ BOOL ■ DOUBLE ■ LONG

Attribute	Description
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" HOLDON="1" />
  <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" HOLDON="1" />
  <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" HOLDON="1" />
  <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" HOLDON="1" />
  <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" HOLDON="1" />
  <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" HOLDON="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" HOLDON="0" />
  <ELEMENT TAG="AK.A5" TYPE="DOUBLE" INDX="11" HOLDON="0" />
  <ELEMENT TAG="AK.A6" TYPE="DOUBLE" INDX="12" HOLDON="0" />
  <ELEMENT TAG="EK.E1" TYPE="DOUBLE" INDX="13" HOLDON="0" />
  <ELEMENT TAG="EK.E2" TYPE="DOUBLE" INDX="14" HOLDON="0" />
  <ELEMENT TAG="EK.E3" TYPE="DOUBLE" INDX="15" HOLDON="0" />
  <ELEMENT TAG="EK.E4" TYPE="DOUBLE" INDX="16" HOLDON="0" />
  <ELEMENT TAG="EK.E5" TYPE="DOUBLE" INDX="17" HOLDON="0" />
  <ELEMENT TAG="EK.E6" TYPE="DOUBLE" INDX="18" HOLDON="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

7.4.4 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="..." />
...
```

7.4.5 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="INTERNAL" />
 </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.

7.4.6 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.

8 Examples

8.1 Configuration and program examples

> The RSI, DIAGRAM and XML files form a unit and must be transferred to the robot controller together.
> Target directory:
> C:\KRC\Roboter\Config\User\Common\SensorInterface

Overview

RobotSensorInterface contains a sample application which can be used to establish and test Ethernet communication between a server program and the robot controller. The sample application and other sample configurations and programs can be found in the DOC\Examples directory of the software.

The sample application for the Ethernet communication consists of the following components:

Components	Folder
Server program TestServer.exe	...Ethernet\Server
Sample program in KRL: ■ RSI_Ethernet.src	...\Ethernet
Sample configuration for the signal flow: ■ RSI_Ethernet.rsi ■ RSI_Ethernet.rsi.xml ■ RSI_Ethernet.rsi.diagram XML file for the Ethernet connection: ■ RSI_EthernetConfig.xml	...\Ethernet\Config

Other sample configurations and programs:

Components	Folder
Sample program in KRL: ■ RSI_CircleCorr.src Sample configuration for the signal flow: ■ RSI_CircleCorr.rsi ■ RSI_CircleCorr.rsi.xml ■ RSI_CircleCorr.rsi.diagram	...\CircleCorr ...\CircleCorr \Config
Sample program in KRL: ■ RSI_DistanceCtrl.src Sample configuration for the signal flow: ■ RSI_DistanceCtrl.rsi ■ RSI_DistanceCtrl.rsi.xml ■ RSI_DistanceCtrl.rsi.diagram	...\DistanceCtrl ...\DistanceCtrl\Config
Sample program in KRL: ■ RSI_SigTransformation.src Sample configuration for the signal flow: ■ RSI_SigTransformation.rsi ■ RSI_SigTransformation.rsi.xml ■ RSI_SigTransformation.rsi.diagram	...\Transformations ...\Transformations\Config

8.1.1 Implementing the sample application

Precondition

External system:

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

Robot controller:

- “Expert” user group
- Operating mode T1 or T2

Procedure

1. Copy the server program onto an external system.
2. Copy KRL programs into the directory C:\KRC\ROBOTER\KRC\R1\Program of the robot controller.
3. Copy the sample configurations and the XML file for the Ethernet connection to the directory C:\KRC\ROBOTER\Config\User\Common\SensorInterface of the robot controller.
4. Start the server program on the external system.
5. Press the menu button. The **Server Properties** window is opened.
6. The available IP addresses of the server PC are displayed under Available Network Interfaces.
7. Set the IP address for the connection between robot and PC in the XML file for the Ethernet connection.

8.1.2 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.

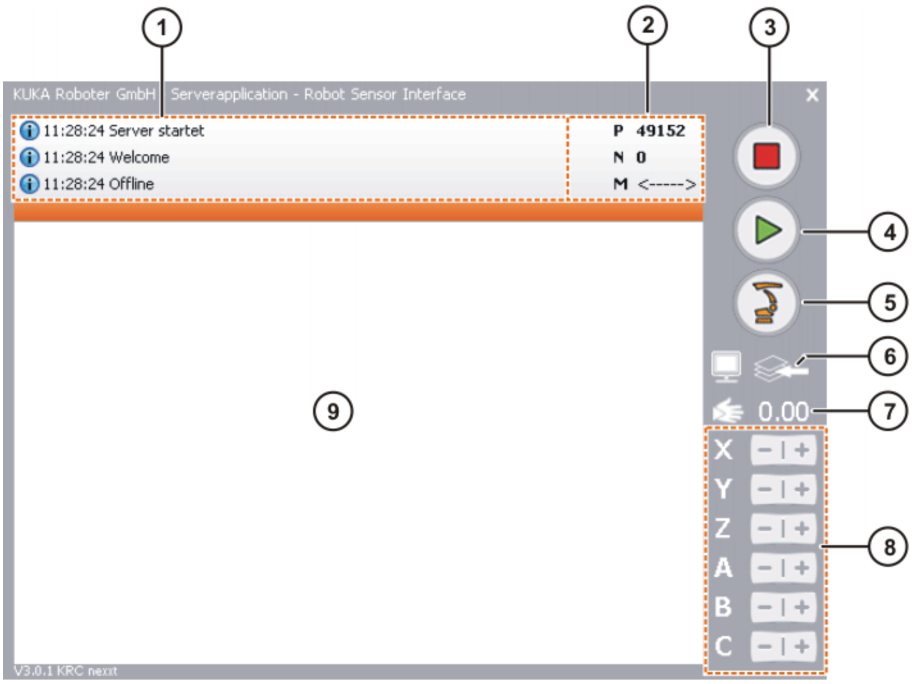


Fig. 8-1: Server program

Item	Description
1	Message window

2	Display of the communication parameters set
	■ **P**: port number
	■ **N**: network card index
	■ **M**: communication mode
	  ■ **<---->**: 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
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**

Item	Description
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.

8.1.3 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

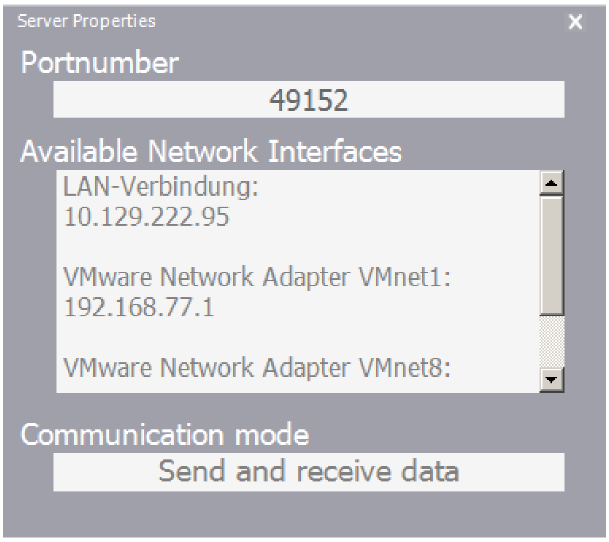


Fig. 8-2: Server Properties window

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: **49152**
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**

8.1.4 Example of a 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 correction 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.rsi",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 signal flow configuration 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

Signal flow configuration

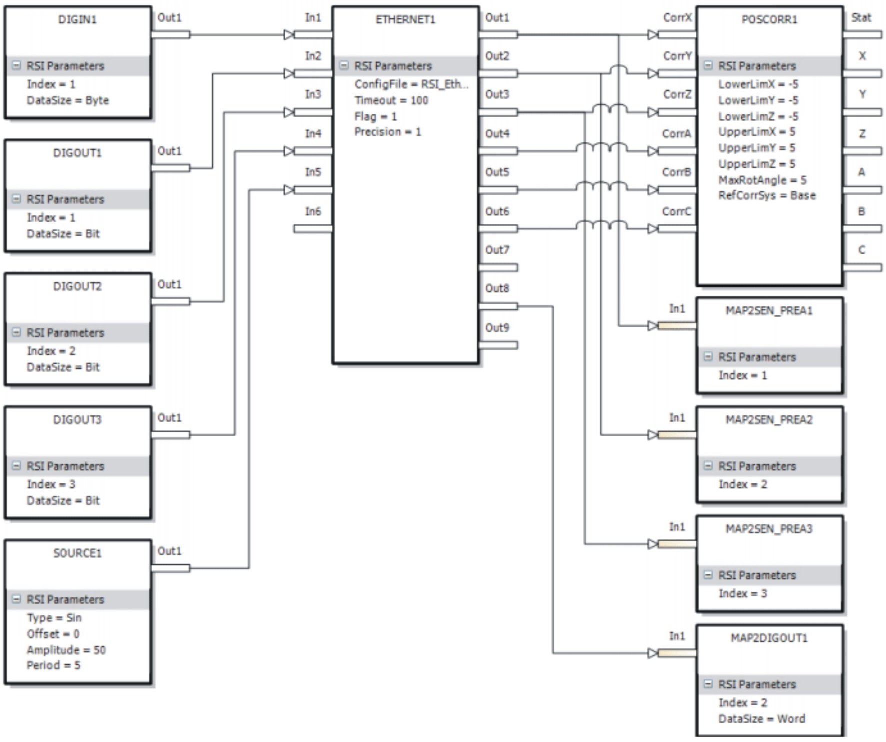


Fig. 8-3: Signal flow – Cartesian correction via Ethernet

RSI object	Description
DIGIN1	Loads sensor data via 8 digital inputs and transfers them to the Ethernet interface (input 1).
DIGOUT1 … DIGOUT3	Loads robot data via 3 digital outputs and transfers them to the Ethernet interface (inputs 2 to 4).
SOURCE1	Supplies a periodical sinusoidal signal with an amplitude of 50 every 5 s.
ETHERNET1	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.
POSCORR1	Loads the sensor data that are available at outputs 1 to 6 of the Ethernet interface and determines the Cartesian correction data.
MAP2SEN_PREA1 … MAP2SEN_PREA3	Writes the Cartesian correction data to system variable $SEN_PREA.
MAP2DIGOUT1	Accesses the processed signals and sets 16 digital outputs.

8.1.5 Example of a 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 in the Z direction and again, with a delay, as a sine 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 correction is the BASE coordinate system. After a defined time, the sensor-guided motion is aborted by means of a timer.

Program

```
DEF RSI_CircleCorr( )
; =============================================
```

```
;
; RSI EXAMPLE: Lissajous circle
; Create a cirle movement with two sine corrections
;
; =============================================

; Declaration of KRL variables
DECL INT ret; Return value for RSI commands
DECL INT CONTID; ContainerID

INI

; Move to start position
PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}

; Base in actual position
$BASE.X=$POS_ACT.X
$BASE.Y=$POS_ACT.Y
$BASE.Z=$POS_ACT.z

; Create RSI Context
ret=RSI_CREATE("RSI_CircleCorr.rsi",CONTID)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; Start RSI execution
ret=RSI_ON(#ABSOLUTE)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; Sensor guided movement
RSI_MOVECORR()

; Turn off RSI
ret=RSI_OFF()
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; Modify RSI parameter
ret=RSI_GETPUBLICPAR(CONTID,"SOURCE1","Amplitude", fVar)
...
ret=RSI_SETPUBLICPAR(CONTID,"SOURCE1","Amplitude", fVar/2)
...
; Start RSI execution
ret=RSI_ON(#ABSOLUTE)
...

; Sensor guided movement
RSI_MOVECORR()

; Turn off RSI
ret=RSI_OFF()
...

PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}

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 signal flow configuration 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.

Signal flow configuration

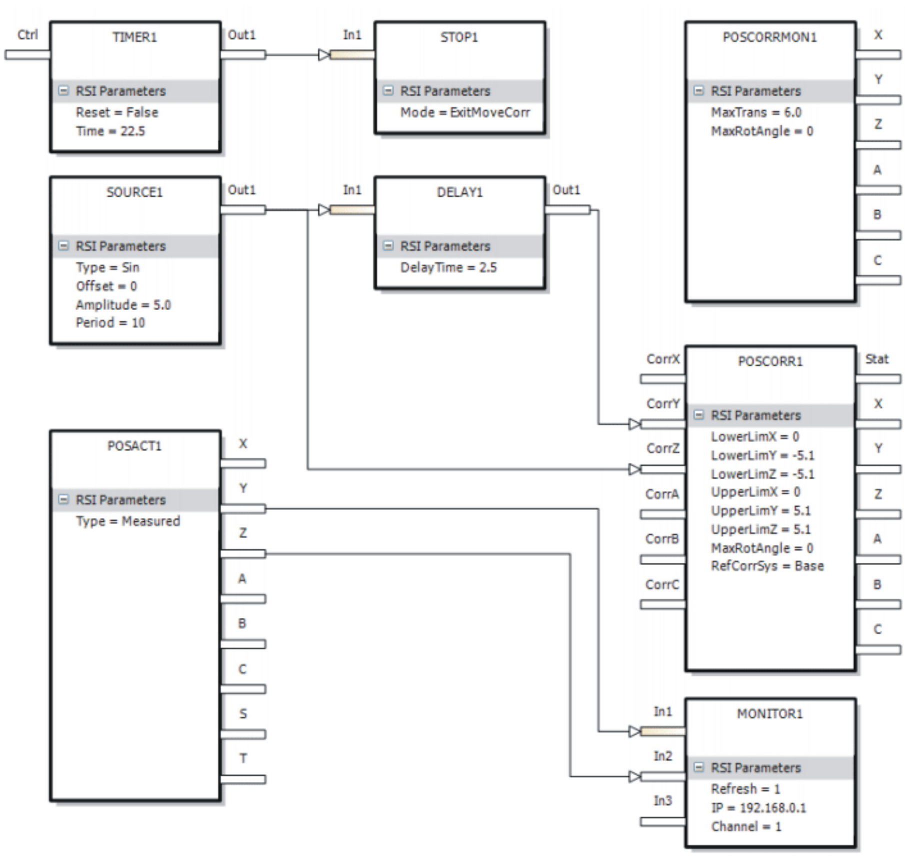


Fig. 8-4: Signal flow – sensor-guided circular motion

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.

RSI object	Description
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 correction: BASE
MONITOR1	The following signals are linked to the MONITOR object and can be displayed on the robot controller using RSI monitor: ■ Cartesian actual position of the robot in the Y and Z directions [mm]

8.1.6 Example of a 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.

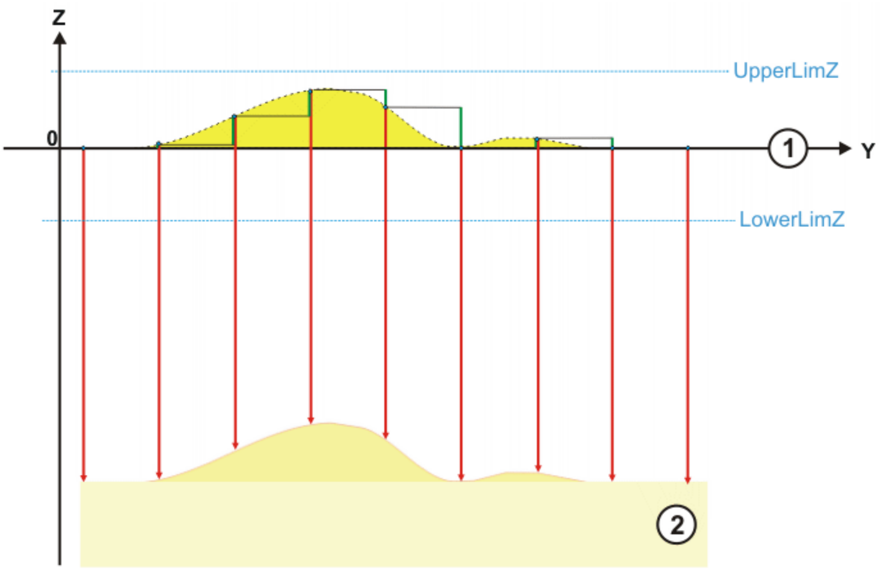


Fig. 8-5: Path correction for distance control

1	Workpiece	2	Programmed path

Program

```
DEF RSI_DistanceCtrl( )
; =============================================
;
; RSI EXAMPLE: Distance Crtl
; Move on a LIN path with superimposed corrections
; Deviation from programmed path is controlled with
; a analog input $ANIN[1]
;
; =============================================

; Declaration of KRL variables
DECL INT ret; Return value for RSI commands
DECL INT CONTID; ContainerID

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.rsi")
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 signal flow configuration 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.

Signal flow configuration

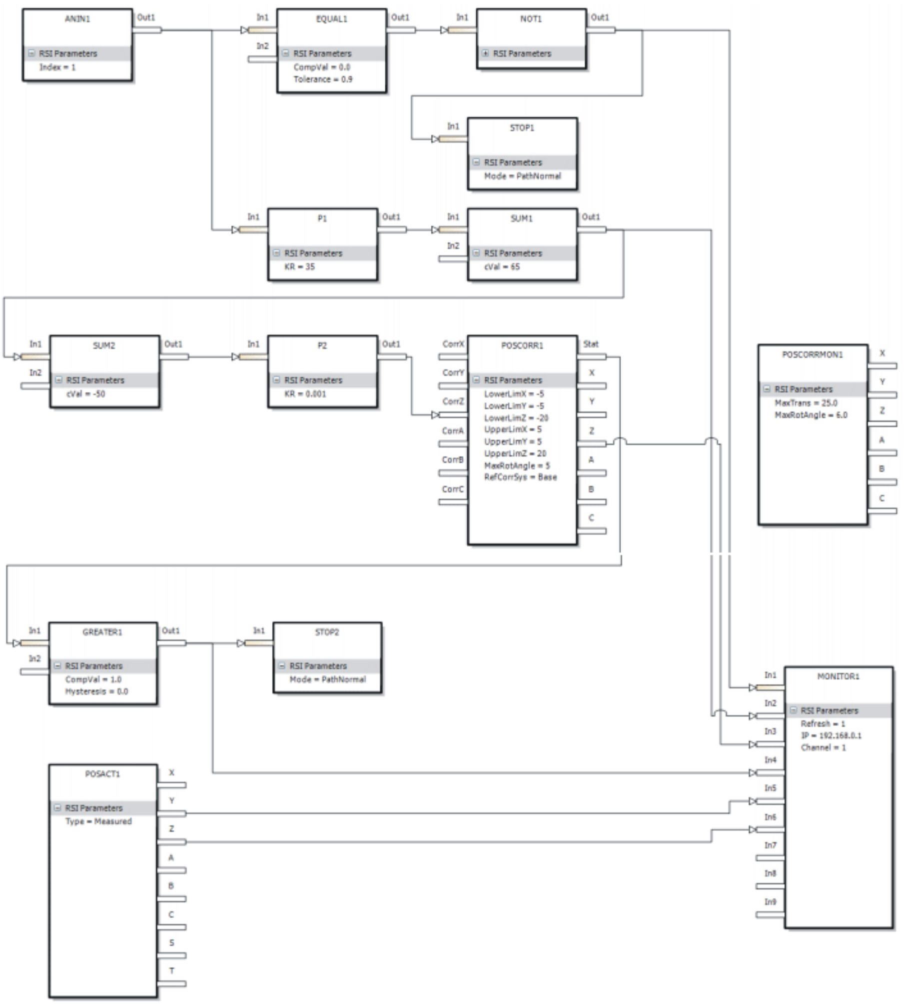


Fig. 8-6: Signal flow – path correction for distance control

RSI object	Description
ANIN1	Loads the sensor signal via an analog input.
EQUAL1 NOT1 STOP1	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.
P1 SUM1 SUM2 P2	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.
POSCORR1	Loads the calculated correction value in the Z direction that is present as a signal at the output of object P2. ■ Reference coordinate system for correction: 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.

RSI object	Description
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 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]

8.1.7 Example of a transformation to 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.

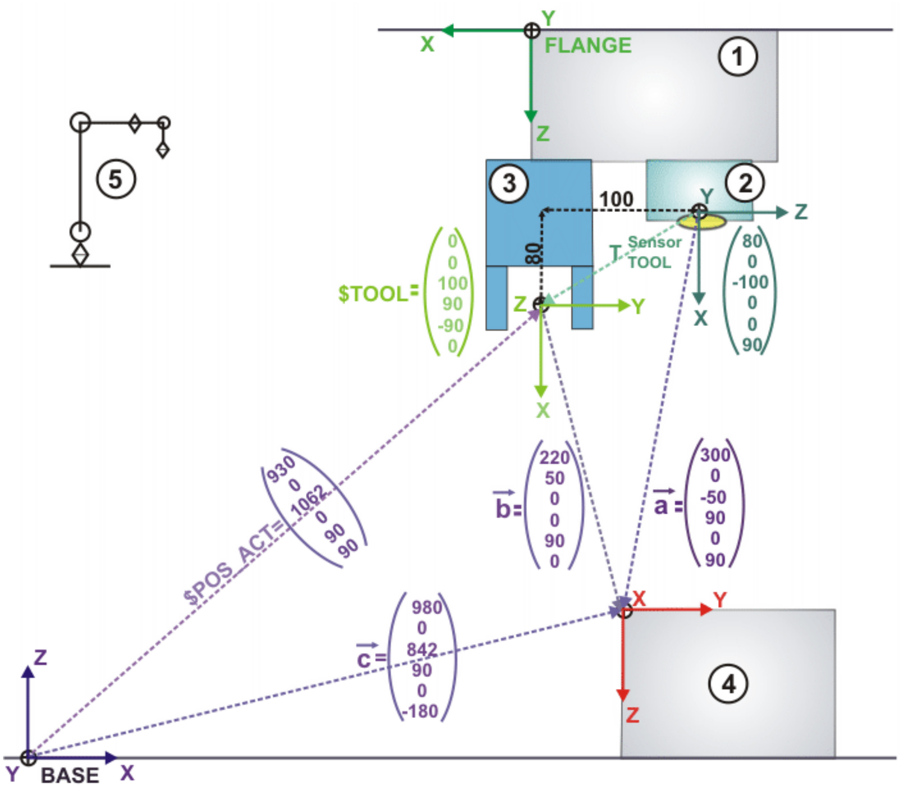


Fig. 8-7: Example of a transformation

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.rsi") <> 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 signal flow configuration 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.

Signal flow configuration

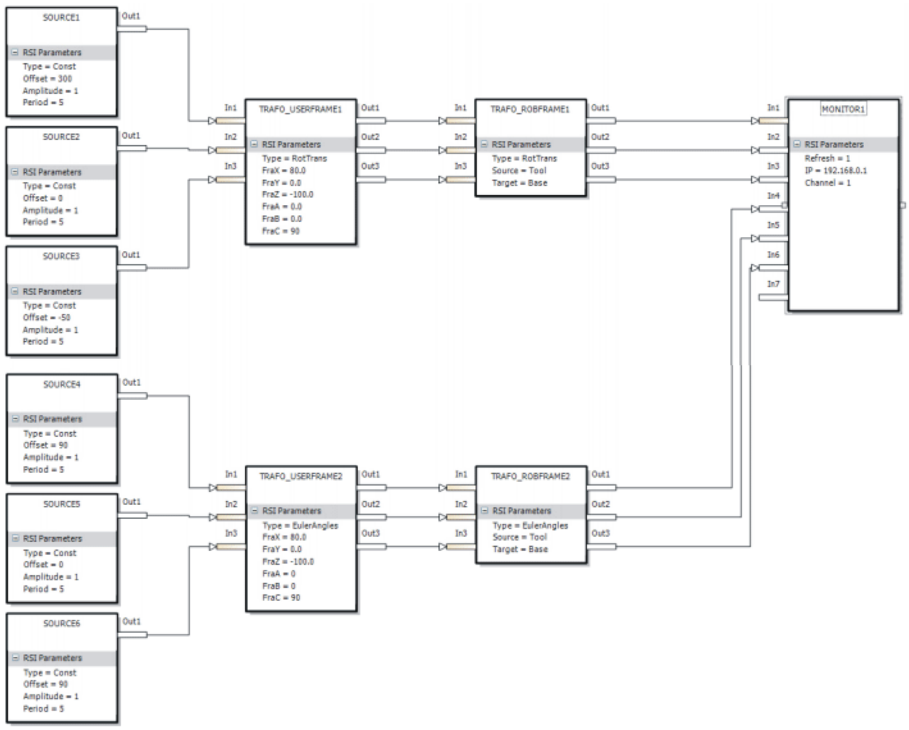


Fig. 8-8: Signal flow – transformation

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 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 Displaying RSI diagnostic data

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 IPO mode ■ **Running (IPO_FAST)**: signal processing in IPO_FAST mode ■ **Stopped**: no signal processing
Cycle time	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)
Roundtrip	Successful communication cycles since the start of signal processing
Total lost	Number of packet losses since the start of signal processing
Quality of communication	Quality of the signal processing ■ **0 … 100%** 100% = all packets have arrived successfully. 0% = no packet has arrived successfully.
Max of following lost packet	Largest contiguous loss of packets since the start of signal processing

9.2 Error protocol (logbook)

The error messages of the interface are logged by default in a LOG file under C:\KRC\ROBOTER\LOG\SensorInterface.

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

9.2.1 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 Messages during operation

No.	Message	Cause	Remedy
29000	*{Type} Permissible overall correction exceeded: RSI is stopped*	The permissible overall correction has been exceeded.	Acknowledge message
29001	*{Type} Correction outside the permissible range: {Value}*	The commanded correction exceeds the defined permissible range for the correction object.	■ Increase the permissible correction ■ Check the signal processing
29002	*Signal flow ({Mode}): Object {ObjName} returns error {ErrorCode}*	The named RSI object cannot be calculated in RSI Visual.	Check whether the RSI object has been correctly configured in RSI Visual and whether the necessary configuration data are present and correct.
29004	*Internal RSI error*	An unexpected return value has been supplied by a function.	If the problem persists, contact KUKA Service.
29005	*RSI cannot set any outputs due to operator protection*	RSI cannot set any outputs due to operator safety.	The operator safety signal must not be set.
29006	*RSI: Signal calculation timeout {CalcTime} usec*	The configured RSI context is too extensive to be calculated in the time available.	Reduce the RSI context

11 Appendix

11.1 Increasing the memory

The memory may be increased only in consultation with KUKA Roboter GmbH. (>>> 12 "KUKA Service" Page 75)

Description

If the available memory is insufficient, it is recommended to check the programming method in KRL as well as the signal flow configuration.

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 correction monitoring

Name	Description
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.
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.

11.2.2 RSI objects for signal transfer

Name	Description
DIGIN	Returns the value of a range of digital inputs $IN.
DIGOUT	Returns the value of a range of digital outputs $OUT
ANIN	Returns the value of an analog input $ANIN.
ANOUT	Returns the value of an analog output $ANOUT.
SEN_PINT	Returns the value of the system variable $SEN_PINT.
SEN_PREA	Returns the value of the system variable $SEN_PREA.
POSACT	Returns the current Cartesian robot position.
AXISACT	Returns the current axis angles of robot axes A1 to A6.
AXISACTEXT	Returns the current positions of external axes E1 to E6.
SOURCE	Signal generator Generates a defined signal curve, e.g. for a constant signal, a sine or cosine signal, etc.
GEARTORQUE	Returns the gear torques of robot axes A1 to A6.
GEARTORQUEEXT	Returns the gear torques of external axes E1 to E6.

Name	Description
MOTORCURRENT	Returns the motor currents of robot axes A1 to A6.
MOTORCURREN-TEXT	Returns the motor currents of external axes E1 to E6.
STATUS	Returns robot controller status information, e.g. current status of submit or robot interpreter, current operating mode, etc.
OV_PRO	Returns the current program override $OV_PRO.

11.2.3 RSI objects for coordinate transformation

Name	Description
TRAFO_ USERFRAME	Transforms a vector consisting of inputs 1 to 3 to a new reference coordinate system with a defined translational and rotational offset.
TRAFO_ ROBFRAME	Transforms a vector consisting of inputs 1 to 3 from one robot reference coordinate system to another.

11.2.4 RSI objects for logic operations

Name	Description
AND	AND operation Up to 10 input signals can be connected.
OR	OR operation Up to 10 input signals can be connected.
XOR	EITHER/OR operation Up to 10 input signals can be connected.
NOT	Logical negation

11.2.5 RSI objects for binary logic operations

Name	Description
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.
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.
BCOMPL	Binary complement

11.2.6 RSI objects for mathematical comparisons

Name	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.

11.2.7 RSI objects for mathematical operations

Name	Description
SUM	Addition of signals Up to 10 input signals can be connected. A constant value can be added with the RSI object parameter **cVal**.
MULTI	Multiplication of signals
ABS	Absolute value function
POW	Power function
SIN	Sine function
COS	Cosine function
TAN	Tangent function
ASIN	Arc sine function
ACOS	Arc cosine function
ATAN	Arc tangent function
EXP	Exponential function
LOG	Logarithm function
CEIL	Smallest integer greater than or equal to input signal
FLOOR	Greatest integer greater than or equal to input signal
ROUND	Rounding 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.

11.2.8 RSI objects for signal control

Name	Description
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>)

Name	Description
I	Integration object (trapezoid algorithm) y(k) = B0 * (x(k) + x(k-1)) + y(k-1) B0 = <sensor cycle> / (2 * TI)
D	Differentiation object y(k) = B0 * (x(k) - x(k-1)) B0 = KD / <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)
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)
IIRFILTER	IIR FILTER

11.2.9 Other RSI objects

Name	Description
TIMER	On expiry of the set time, a positive edge is set on signal output “Out1”.
LIMIT	Limits the signal to values within a lower and upper limit (LowerLimit, UpperLimit).
MINMAX	Returns the current smallest and largest signal across all input signals. Up to 10 input signals can be connected.
DELAY	Delays the input signal by a defined time.

Name	Description
SIGNALSWITCH	Switches between 2 signal paths via control signal.
ETHERNET	UDP Ethernet communication in XML data format Up to 64 inputs and 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.

11.2.10 RSI objects for actions

Name	Description
MAP2OV_PRO	Changes the program override ($OV_PRO).
STOP	Stops a motion at a positive signal edge. A purely sensor-guided motion with RSI_MOVECORR can be terminated with **ExitMoveCorr** mode.
MAP2SEN_PINT	Changes the value of the system variable $SEN_PINT.
MAP2SEN_PREA	Changes the value of the system variable $SEN_PREA.
MAP2DIGOUT	Describes a digital output $OUT or a range of digital outputs.
MAP2ANOUT	Describes an analog output $ANOUT.
SETDIGOUT	Sets a digital output $OUT at a positive edge. The set output is maintained even at a negative edge.
RESETDIGOUT	Resets a digital output $OUT at a positive edge. The reset output is maintained even at a negative edge.
POSCORR	Cartesian correction with limitation
AXISCORR	Axis-specific correction with limitation, robot axes A1 to A6
AXISCORREXT	Axis-specific correction with limitation, external axes E1 to E6
MONITOR	RSI monitor Visualization of up to 24 RSI signals

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:

- Model and serial number of the manipulator
- Model and serial number of the controller
- Model and serial number of the linear unit (if present)
- Model and serial number of the energy supply system (if present)
- Version of the system software
- Optional software 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
- Description of the problem, duration and frequency of the fault

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 Roboter GmbH Zugspitzstr. 140 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 info@kuka-roboter.de www.kuka-roboter.de

France	KUKA 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 Robotics India Pvt. Ltd. Office Number-7, German Centre, Level 12, Building No. - 9B DLF Cyber City Phase III 122 002 Gurgaon Haryana India Tel. +91 124 4635774 Fax +91 124 4635773 info@kuka.in www.kuka.in
Italy	KUKA Roboter Italia S.p.A. Via Pavia 9/a - int.6 10098 Rivoli (TO) Italy Tel. +39 011 959-5013 Fax +39 011 959-5141 kuka@kuka.it www.kuka.it
Japan	KUKA Robotics Japan K.K. YBP Technical Center 134 Godo-cho, Hodogaya-ku Yokohama, Kanagawa 240 0005 Japan Tel. +81 45 744 7691 Fax +81 45 744 7696 info@kuka.co.jp
Canada	KUKA Robotics Canada Ltd. 6710 Maritz Drive - Unit 4 Mississauga L5W 0A1 Ontario Canada Tel. +1 905 670-8600 Fax +1 905 670-8604 info@kukarobotics.com www.kuka-robotics.com/canada

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

Poland

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

Portugal

KUKA Sistemas de Automatización S.A.
Rua do Alto da Guerra n° 50
Armazém 04
2910 011 Setúbal
Portugal
Tel. +351 265 729780
Fax +351 265 729782
kuka@mail.telepac.pt

Russia

KUKA Robotics RUS
Werbnaja ul. 8A
107143 Moskau
Russia
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
info@kuka-robotics.ru
www.kuka-robotics.ru

Sweden

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

Switzerland

KUKA Roboter Schweiz AG
Industriestr. 9
5432 Neuenhof
Switzerland
Tel. +41 44 74490-90
Fax +41 44 74490-91
info@kuka-roboter.ch
www.kuka-roboter.ch

Spain	KUKA Robots IBÉRICA, S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 8142-353 Fax +34 93 8142-950 Comercial@kuka-e.com www.kuka-e.com
South Africa	Jendamark Automation LTD (Agency) 76a York Road North End 6000 Port Elizabeth South Africa Tel. +27 41 391 4700 Fax +27 41 373 3869 www.jendamark.co.za
Taiwan	KUKA Robot Automation Taiwan Co., Ltd. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 info@kuka.com.tw www.kuka.com.tw
Thailand	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 atika@ji-net.com www.kuka-roboter.de
Czech Republic	KUKA Roboter Austria GmbH Organisation Tschechien und Slowakei Sezemická 2757/2 193 00 Praha Horní Počernice Czech Republic Tel. +420 22 62 12 27 2 Fax +420 22 62 12 27 0 support@kuka.cz

Hungary	KUKA Robotics Hungaria Kft. Fö út 140 2335 Taksony Hungary Tel. +36 24 501609 Fax +36 24 477031 info@kuka-robotics.hu
USA	KUKA Robotics Corporation 51870 Shelby Parkway Shelby Township 48315-1787 Michigan USA Tel. +1 866 873-5852 Fax +1 866 329-5852 info@kukarobotics.com www.kukarobotics.com
UK	KUKA Automation + Robotics Hereward Rise Halesowen B62 8AN UK Tel. +44 121 585-0800 Fax +44 121 585-0900 sales@kuka.co.uk

Index