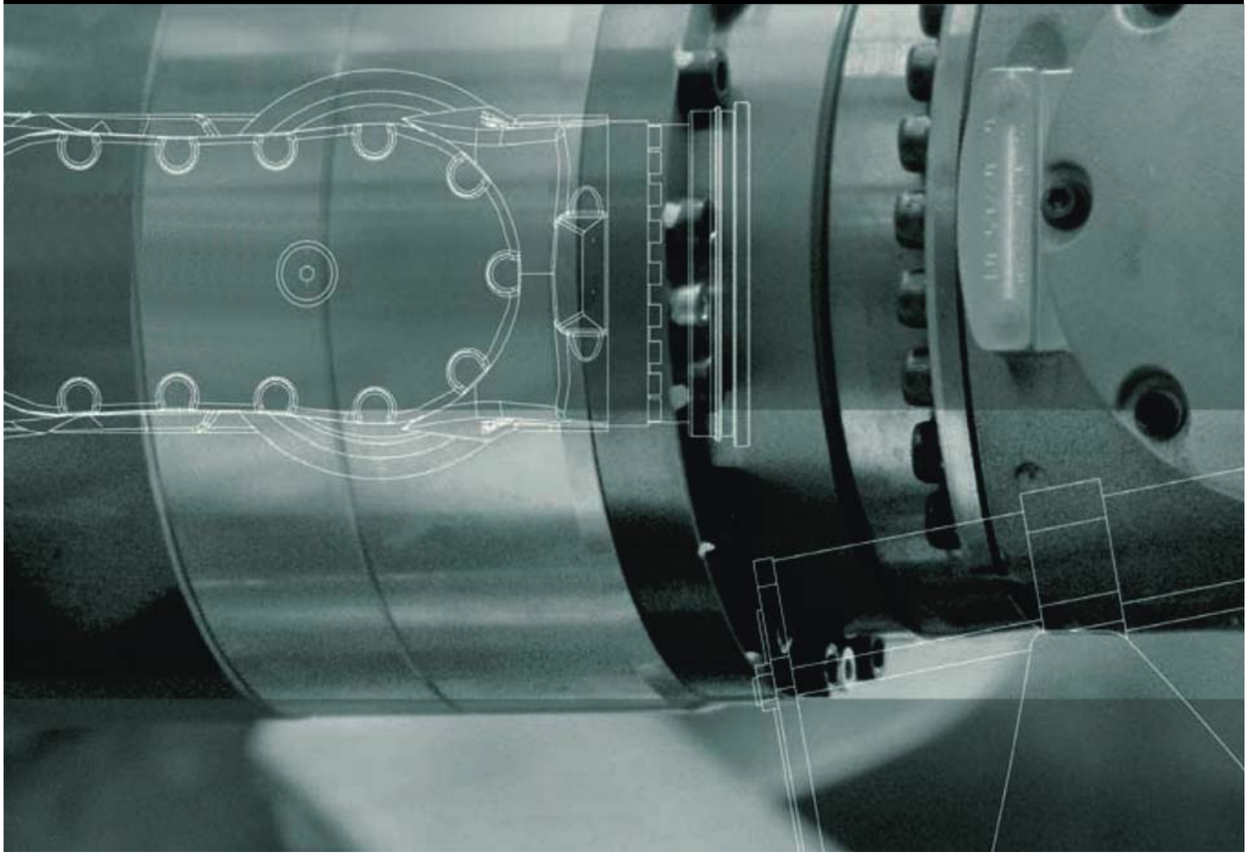
KUKA

KUKA System Technology

KUKA Roboter GmbH

KUKA.RobotSensorInterface 2.3

For KUKA System Software 5.4, 5.5, 5.6, 7.0



Issued: 13.05.2009

Version: KST RSI 2.3 V1 en

Publication:	Pub KST RSI 2.3 en
Book structure:	KST RSI 2.3 V1.1
Label:	KST RSI 2.3 V1

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 field bus and network connections
- Knowledge of XML
- 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 Robot system documentation

The robot system documentation consists of the following parts:

- Operating instructions for the robot
- Operating instructions for the robot controller
- Operating and programming instructions for the KUKA System Software
- Documentation relating to options and accessories

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

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

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

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

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

Notes

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

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

1.4 Trademarks

Suse Linux is a trademark of Linus Torvalds.

VxWorks is a trademark of Wind River Systems Inc.

Windows is a trademark of Microsoft Corporation.

1.5 Terms used

Term	Description
Container	Containers are used to group RSI objects together and structure them. All RSI objects in a single container can be deleted, activated or deactivated simultaneously.
KUKA.HMI	Human/Machine Interface KUKA.HMI is the KUKA user interface.
Object ID	The object ID is the identifier of an RSI object. The value for the object ID is automatically assigned by the robot system when the RSI object is created. The unique object ID can be used to address a specific RSI object by means of RSI commands.
Object parameters	The object parameters of an RSI object influence its functionality. The number of object parameters is specific for each RSI object.
Parameter ID	A parameter ID is the number of an object parameter. The object parameters of an RSI object are numbered consecutively, always commencing with 1.
Parser	A parser is a program that syntactically interprets textual components of a document and replaces them with commands or codes.
RSI context	The RSI context is the entire signal processing programmed with KUKA.RobotSensorInterface and consists of RSI objects and links between the RSI objects.
RSI monitor	The RSI monitor can record and display up to 24 signals from the RSI context.
RSI object	Each RSI object has a functionality and corresponding signal inputs and/or outputs.
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 tool direction (+X axis of the TOOL coordinate system) and the resulting normal vector. The tool-based moving frame coordinate system is defined as follows: X_{TTS}: path tangent Y_{TTS}: normal vector to the path tangent and tool direction Z_{TTS}: negative tool direction The path tangent and the tool direction must not be parallel, otherwise the TTS cannot be calculated.
VxWorks	Real-time operating system
XML	Extensible Markup Language Standard for creating machine- and human-readable documents in the form of a specified tree structure.

2 Product description

2.1 KUKA.RobotSensorInterface overview

Functions

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

- Configuration of signal processing in the real-time system of the robot controller.
- Influence on the robot motion or program execution by means of the signal processing.
- Display of the signals via a monitor.
- Configuration of real-time data exchange between the robot controller and an external system via Ethernet.
- Influence on the robot motion or path planning by means of the data exchange via Ethernet.

Characteristics

Signal processing:

- Cyclical signal processing and evaluation in the interpolation cycle (12 ms) parallel to program execution.
- Configuration of the signal processing in the KRL program via RSI commands.
- Library with function blocks for signal processing (e.g. filters, logic gates, transformations, controllers, etc.).
- Creation of signal processing with up to 256 active RSI objects.
- Combination of different sensor technologies.

Areas of application

- As basic technology for real-time sensor applications with cyclical signal processing and evaluation
- Implementation of external applications (e.g. transferring computing processes to an external system)
- Implementation of extensive diagnosis and analysis functions on an external system
- Integration of microprocessor-supported sensors with a network connection
- Monitoring of the position of the robot with an external system

Communication

- The robot controller communicates with the sensor system via a field bus. The sensor data and signals are read by the field bus. KUKA.RobotSensorInterface accesses the data and signals and processes them.

Further information about the field buses can be found in the corresponding KUKA documentation.

- Only if the data exchange is configured via Ethernet: the robot controller communicates with the external system via a real-time-capable network connection. The exchanged data are transmitted via the Ethernet TCP/IP or UDP/IP protocol as XML strings.

  Characteristics:

  - Cyclical data transmission from the robot controller to an external system in the interpolation cycle of 12 ms parallel to program execution (e.g. position data, axis angles, operating mode, etc.)
  - Cyclical data transmission from an external system to the robot controller in the interpolation cycle of 12 ms parallel to program execution (e.g. sensor data)

2.2 Functional principle of signal processing

Description

Signal processing is established using RSI objects. An RSI object has a functionality and corresponding signal inputs and/or outputs.

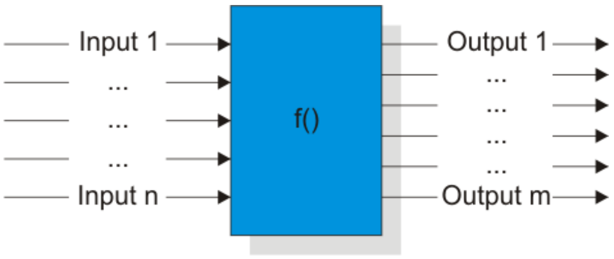


Fig. 2-1: Schematic structure of an RSI object

The following groups of RSI objects exist:

- Data access objects
- Signal processing objects
- Action objects

The RSI context is derived from the links of the functionalities of multiple RSI objects. The RSI context is a freely configurable signal flow and is generated from the KRL program.

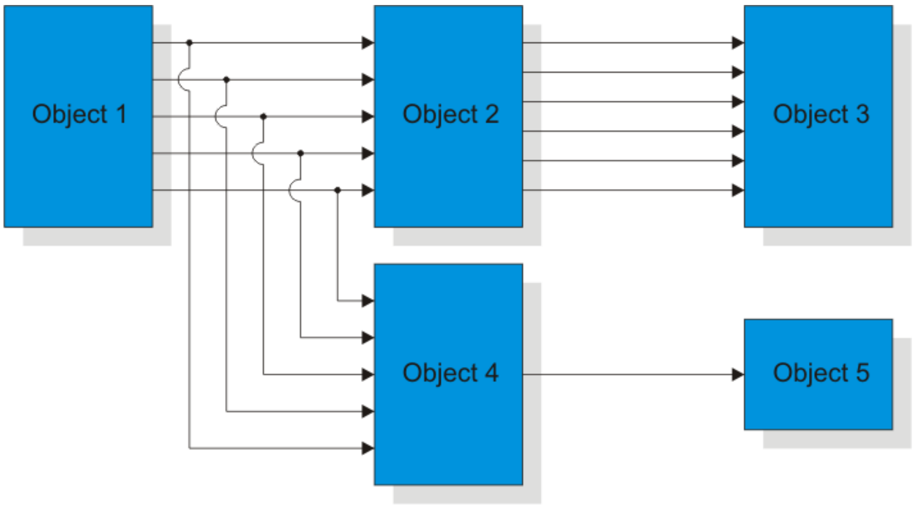
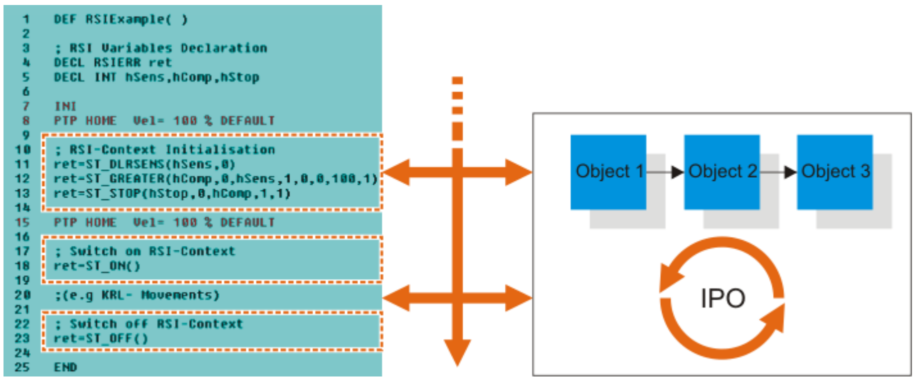


Fig. 2-2: Schematic structure of an RSI context

The KRL program also activates and deactivates calculation of the signal processing parallel to program execution.



```
DEF RSIExample( )

; RSI Variables Declaration
DECL RSIERR ret
DECL INT hSens,hComp,hStop

INI
PTP HOME  Vel= 100 % DEFAULT

; RSI-Context Initialisation
ret=ST_DLRSENS(hSens,0)
ret=ST_GREATER(hComp,0,hSens,1,0,0,100,1)
ret=ST_STOP(hStop,0,hComp,1,1)

PTP HOME  Vel= 100 % DEFAULT

; Switch on RSI-Context
ret=ST_ON()

;(e.g KRL- Movements)

; Switch off RSI-Context
ret=ST_OFF()

END
```

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

2.3 Functional principle of data exchange

2.3.1 Data exchange via a field bus

Description

The sensor data and signals are read by a field bus, processed in the RSI context and forwarded by this field bus to the robot controller.

The following RSI objects are used for this:

- ST_ANAIN and ST_DIGIN access the field bus and transfer the sensor data and signals to the signal processing.
- ST_ANAOUT and ST_DIGOUT access the processed signals and transfer them to the field bus.

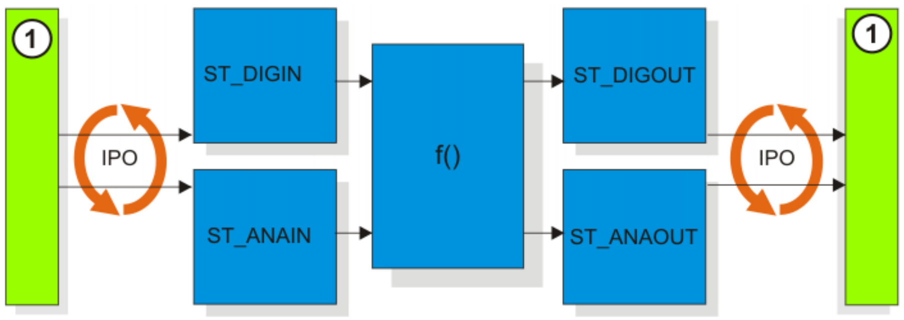


Fig. 2-4: Functional principle of data exchange via a field bus

1 Field bus

2.3.2 Data exchange via Ethernet

Description

Data exchange via Ethernet must be configured in the KRL program. The RSI object ST_ETHERNET is used for this.

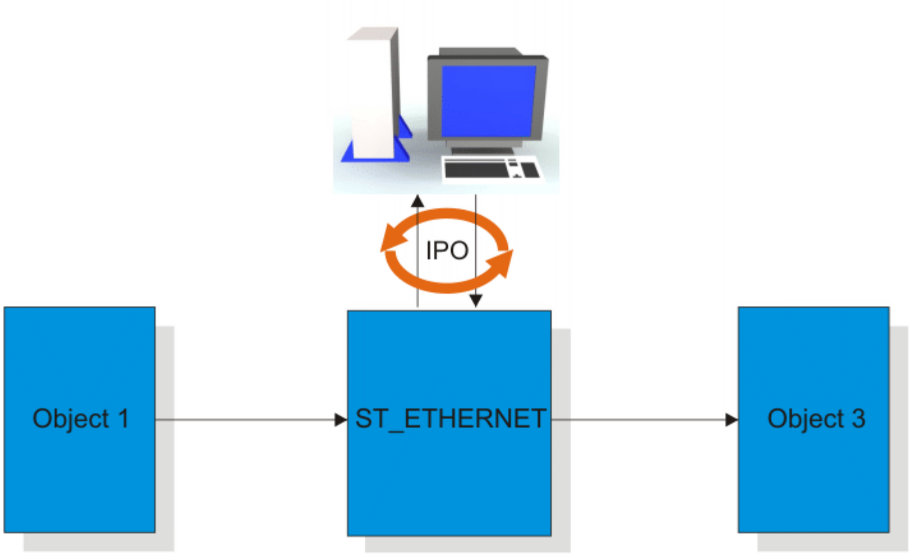


Fig. 2-5: Functional principle of data exchange via Ethernet

If signal processing is activated with ST_ETHERNET, the robot controller connects to the external system as a client. The robot controller initiates the cyclical data exchange with a KRC data packet and transfers further KRC data packets to the external system in the interpolation cycle of 12 ms. The external system must respond to the KRC data packets received with a data packet of its own.

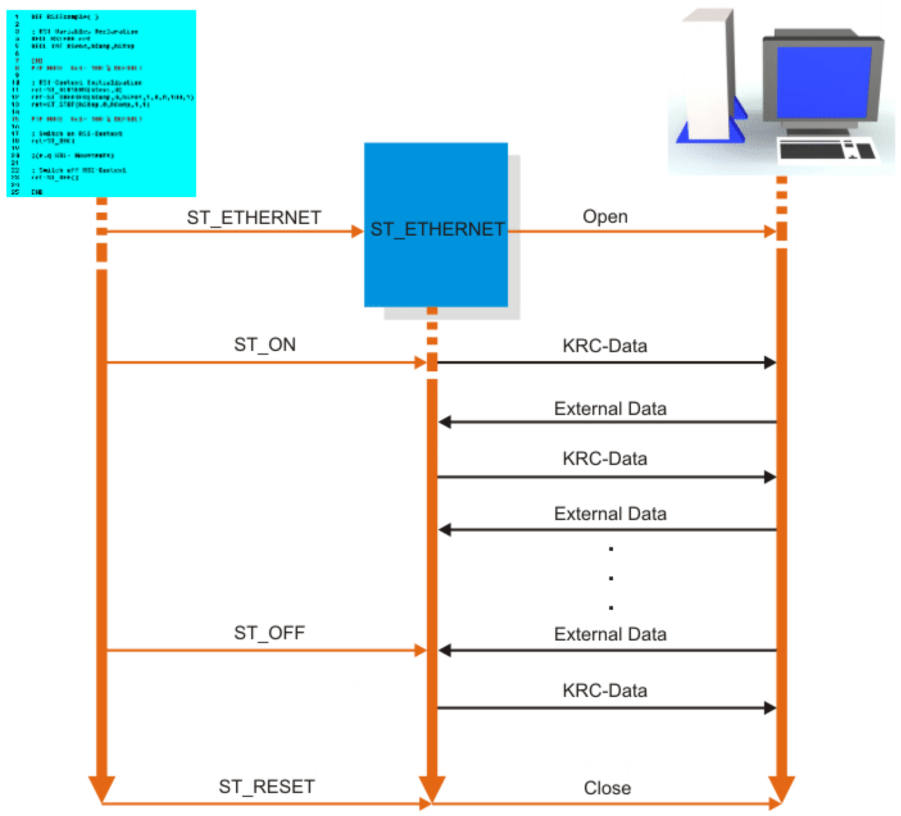


Fig. 2-6: Data exchange sequence via Ethernet

A data packet received by the external system must be answered within approx. 12 ms. If the data packet is not received by the robot controller within this period, the response is classified as too late. 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. If the communication object ST_ETHERNET is deleted, the connection between the robot controller and the external system is interrupted. Both sides exchange data in the form of XML strings.

3 Safety

Personnel

- All persons working with the robot system must have read and understood the robot system documentation, including the safety chapter.

Further information is contained in the operating and programming instructions, in the robot operating instructions and in the robot controller operating instructions.

- KUKA.RobotSensorInterface is a software package and contains no hardware components. The system integrator is responsible for correct selection of the necessary components.
- The system integrator can generate complete applications with this technology package. For this, the system integrator must configure the technology package as appropriate for the specific application.

Robot system

- The robot system with KUKA.RobotSensorInterface must be operated in accordance with the applicable national laws, regulations and standards.
- The user must ensure that the system can be operated in complete safety.

Sensor-assisted operation

- If used incorrectly, KUKA.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 user 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. workspace limitation).
- At the start of signal processing with KUKA.RobotSensorInterface, the system generates the following acknowledgement message in T1 or T2 mode:

  !!!Attention –RSI sensor mode active!!!
- New or modified programs must always be tested first in operating mode T1. If the reduced velocity in T1 mode is insufficient for the process, a program can be tested in T2 mode, as long as there is no-one in the danger zone of the robot.
- During programming, the presence of persons within the danger zone of the robot is to be avoided.
- The robot must only be moved at reduced velocity (max. 250 mm/s) in T1 mode during programming in the danger zone of the robot. This is to give the operator enough time to move out of the way of hazardous robot motions or to stop the robot.
- In T1 and T2 modes, an enabling switch and a Start key must be held down in order to move the robot. In the event of unexpected movements, the robot can be stopped immediately by releasing the enabling switch or pressing it down fully (panic position).

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 KUKA System Software (KSS) allows the configuration of a maximum of 8 Cartesian and 8 axis-specific workspaces. The user 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.

Path correction limitation

- By default, KUKA.RobotSensorInterface limits the maximum path correction to +/- 5 mm for translational direction corrections and +/- 5° for orientation or axis corrections.
- If the signal processing with KUKA.RobotSensorInterface results in a larger path correction than the limitation will permit, the correction is rejected and the following error message is generated:
  - For a Cartesian path correction (ST_PATHCORR): **SEN: ST_PATHCORR – correction out of range XXX**
  - For an axis angle correction (ST_AXISCORR): **SEN: ST_AXISCORR – correction out of range XXX**
- If the preset correction range is not sufficient for the process, it can be adapted. To do so, the RSI command ST_SETPARAM is used to assign correspondingly adapted values to the object parameters of the RSI objects ST_PATHCORR or ST_AXISCORR.

More detailed information about the RSI objects and commands can be found in the file ...\DOC\rsiCommands.chm on the CD-ROM.

4 Installation

4.1 System requirements

Hardware

- KR C2 edition2005 or KR C2 sr robot controller
- Only for real-time communication via Ethernet:
  - Processor-supported external system with real-time-capable operating system and real-time-capable network card with 10/100 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
  - KUKA network card

Field bus

The following field bus systems can be used for communication between the robot controller and the connected periphery:

- Interbus
- Profibus
- DeviceNet
- ProfiNet

Sensor system

- Sensor system components according to the specific application

Software

- KUKA System Software 5.4, 5.5, 5.6 or 7.0
- XML parser for the data exchange between an external system and the robot controller

  Recommended parser:
  - Microsoft .Net XML parser
  - Gnome parser, SuSE LINUX

Compatibility

- Real-time communication via Ethernet cannot be used in the RoboTeam.

KRL resources

The following KRL resources must be free:

KRL resource	Number
Outputs	$OUT[16]
Function generators	1
Interrupts	11

The KRL resources can be reconfigured following installation.
(>>> 5.4 "Reconfiguring KRL resources" page 19)

4.1.1 PCI slot assignment

Overview

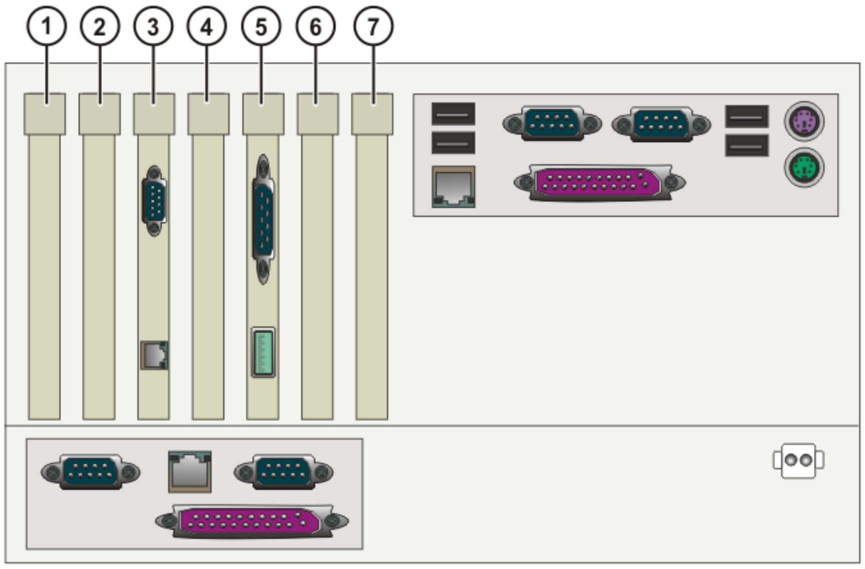


Fig. 4-1: PCI slots

The PC slots can be fitted with the following plug-in cards:

Slot	Plug-in card
1	■ Interbus card (FOC) (optional) ■ Interbus card (copper) (optional) ■ LPDN scanner card (optional) ■ Profibus master/slave card (optional) ■ CN_EthernetIP card (optional)
2	■ LPDN scanner card (optional)
3	KVGA card
4	DSE-IBS-C33 AUX card (optional)
5	MFC3 card
6	■ Network card (optional) ■ LPDN scanner card (optional) ■ Profibus master/slave card (optional) ■ LIBO-2PCI card (optional) ■ KUKA modem card (optional)
7	free

4.2 Installing or updating KUKA.RobotSensorInterface

Precondition

- CD-ROM with additional software
- Only required if there is no CD-ROM drive on the control cabinet:
  Bootable USB CD-ROM/DVD drive
- User group “Expert”

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

Procedure

1. Select the menu sequence **Setup** > **Install Additional Software**.
2. Press the **New SW** softkey. If the technology package **RSI** is not yet displayed, press the **Refresh** softkey.

3. Select **RSI** and press the **Install** softkey. Answer the request for confirmation with **Yes**.
4. Select the installation type in the **Realtime (RT) Ethernet support** window.

Installation type	Description
Install RT Ethernet	For the network connection with the real-time operating system VxWorks.
Keep actual configuration	No network connection is established with the real-time operating system VxWorks.

5. Press **Next** to proceed. If the installation type **Keep actual configuration** has been selected, the files are copied onto the hard drive. Continue with step 8.
6. If the installation type **Install RT Ethernet** has been selected: enter the IP address of the robot controller in the **KUKA - Kernel System Network Setup** window.

If an IP address is already displayed in the window, a network interface has already been installed. The displayed IP address can be changed.

The IP address range 192.01.x is disabled for configuration of the network connection with VxWorks. Entering an IP address in this range results in a system error. It is then no longer possible to boot the robot controller.

7. Confirm with **OK**. The files are copied onto the hard drive.
8. Reboot the robot controller. The installation is resumed and completed.

LOG file

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

4.2.1 Modifying the IP address for KSS 5.x

Precondition

- User group “Expert”
- Windows interface (CTRL+ESC)

Procedure

1. Open the file C:\Windows\vxWin.ini.
2. Modify the IP address under e={......}.
3. Save and close.
4. Reboot the robot controller.

4.2.2 Modifying the IP address for KSS 7.0

Precondition

- User group “Expert”
- Windows interface (CTRL+ESC)

Procedure

1. Open the file C:\KRC\ROBOTER\INIT\progress.ini.
2. Modify the IP address under IPADDR_ELPCI.
3. Save and close.
4. Reboot the robot controller.

4.3 Uninstalling KUKA.RobotSensorInterface

Precondition

- Expert user group

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

Procedure

1. Select the menu sequence **Setup** > **Install Additional Software**. All installed additional programs are displayed.
2. Select **RSI** and press the **Uninstall** softkey. Answer the request for confirmation with **Yes**.

   "*De-installation prepared*" is displayed in the **State** column.
3. Reboot the robot controller. Uninstallation is resumed and completed.

LOG file

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

5 Configuration

5.1 Units for signal processing

Description

For signal processing with KUKA.RobotSensorInterface, the signals must be assigned units. When RSI objects are linked, the assigned units are used to carry out a plausibility check of the signal flow. A check is made to see whether a signal with the permissible unit has been applied to the input of an RSI object.

The plausibility check only recognizes different units, not different unit prefixes (e.g. **k**m, **m**m, etc.).

The units for the signals include the base SI units:

Variable	Unit	RSI constant
Length	Meters [m]	RSIUNIT_m
Mass	Kilograms [kg]	RSIUNIT_kg
Time	Seconds [s]	RSIUNIT_s
Electric current	Amperes [A]	RSIUNIT_A
Temperature	Kelvin [K]	RSIUNIT_K
Luminous intensity	Candelas [cd]	RSIUNIT_Cd
Amount of substance	Moles [mol]	RSIUNIT_mol
No unit	-----	RSIUNIT_No

Additional units can be derived from the base SI units. The following derived units are already included in KUKA.RobotSensorInterface:

Variable	Unit	RSI constant
Force	Newtons [N]	RSIUNIT_N
Torque	Newton-meters [Nm]	RSIUNIT_Nm
Electric potential difference	Volts [V]	RSIUNIT_V
Pressure	Pascals [Pa]	RSIUNIT_Pa

The unit of a signal is displayed with a 32-bit INTEGER variable. There are 4 bits available for each base SI unit. The 4 bits can be used to raise each base SI unit to the power of -8 to +7.

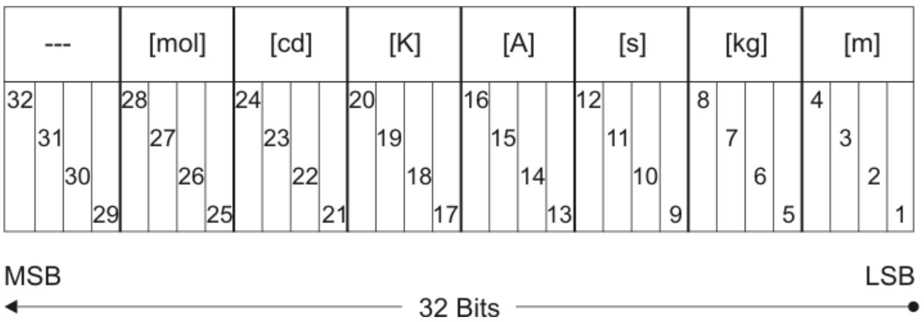


Fig. 5-1: Unit scheme of KUKA.RobotSensorInterface

LSB	Least significant bit
MSB	Most significant bit

5.2 Creating a new unit

Precondition

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

Procedure

1. Express the new unit is base SI units.

   Example: electric field strength [V/m] in base SI units corresponds to [kg]*[m]/[A]*[s]^3
2. Calculate the hexadecimal value of the new unit in accordance with the unit scheme.

   Example: electric field strength [kg]*[m]/[A]*[s]^3 corresponds to the hexadecimal value FD11$_{Hex}$
3. Open the file ...\R1\TP\RSI\RSILIB.DAT.
4. Create the new unit as a global constant in the “Composite Units” section and assign the calculated hexadecimal value.

```
;Composite Units:
GLOBAL CONST INT RSIUNIT_N='HE11' ;[N]  Newton
GLOBAL CONST INT RSIUNIT_Nm='HE12' ;[Nm] Newtonmeter
GLOBAL CONST INT RSIUNIT_V='HFD12' ;[V]  Volt
GLOBAL CONST INT RSIUNIT_Pa='HE1F' ;[Pa] Pascal
GLOBAL CONST INT RSIUNIT_E='HFD11' ;[V/m] Volt per meter
;End Composite Units
```

5. Save changes in the file by pressing the **Close** softkey.

   The new RSI unit [V/m] can be used in the signal processing.

Example

The unit Newton [N] is derived from the base SI units [kg]*[m]/[s]^2.

Base SI unit	Bit range in the unit scheme	Power	Value
[kg]	000000X0$_{Hex}$	1	00000010$_{Hex}$
[m]	0000000X$_{Hex}$	1	00000001$_{Hex}$
[s]$^{-2}$	00000X00$_{Hex}$	-2	00000E00$_{Hex}$
Total			00000E11$_{Hex}$

The base SI units [kg] and [m] appear in their basic form in the formula for the unit Newton [N] and are not raised to a power. The resulting hexadecimal value in the corresponding bit range of these units is thus 1$_{Hex}$ (=0001$_{Bin}$).

The base SI unit [s] appears in the formula for the unit Newton [N] as the denominator and is raised to the power -2$_{Dec}$. The hexadecimal value for the unit [s]$^{-2}$ is derived as follows:

1. The power, not preceded by a sign, specifies the binary starting value of the base SI unit:

   2$_{Dec}$ = 0010$_{Bin}$
2. For the negative sign of the power, the ones complement of the binary value must be formed:

   Ones complement of 0010$_{Bin}$ = 1101$_{Bin}$
3. A 1$_{Bin}$ is added to the ones complement and the twos complement is formed:

   1101$_{Bin}$ + 0001$_{Bin}$ = 1110$_{Bin}$

4. The calculated binary value 1110_{Bin} for the unit $[s]^{-2}$ corresponds to the hexadecimal value E_{Hex}. The calculated hexadecimal values of the individual units must now be grouped together using the unit scheme.

---	[mol]	[cd]	[K]	[A]	[s]	[kg]	[m]
0	0	0	0	0	E	1	1

For the unit Newton [N], this results in the hexadecimal value $00000E11_{Hex}$.

5.3 Configuring the RSI message display

Description

The display of the RSI-specific messages in the message window can be deactivated if required by means of the variable RSIERRMSG.

Variable	Description
RSIERRMSG	**TRUE**: The RSI-specific messages are displayed in the message window. (Default) **FALSE**: The RSI-specific messages are deactivated and are not displayed in the message window.

Precondition

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

Procedure

1. Open the file ...\R1\TP\RSI\RSILIB.DAT.
2. Set the global variable RSIERRMSG to TRUE or FALSE in the “RSI global Variables” section.

```
;RSI global Variables:
GLOBAL INT OV_RSI=30 ; Override for ST_SKIP/RET... movement after in-
terrupt
GLOBAL INT RSIBREAK=16 ;Index of break motion condition
GLOBAL BOOL RSIERRMSG=TRUE ;Flag for enabling BOF RSI error messages
GLOBAL INT RSITECHIDX=1 ; Tech Channel used for RSI
;End RSI global Variables
```

3. Save changes in the file by pressing the **Close** softkey.

5.4 Reconfiguring KRL resources

Description

The following KRL resources required by KUKA.RobotSensorInterface can be reconfigured:

- Digital output $OUT[16]
- Function generator 1

Interrupt 11 must not be modified.

Precondition

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

Procedure

1. Open the file ...\R1\TP\RSI\RSILIB.DAT.

2. To modify digital output $OUT[16], assign the new number of the digital output to the global variable RSIBREAK in the “RSI global Variables” section.

```
;RSI global Variables:
GLOBAL INT OV_RSI=30 ; Override for ST_SKIP/RET... movement after in-
terrupt
GLOBAL INT RSIBREAK=20 ;Index of break motion condition
GLOBAL BOOL RSIERRMSG=TRUE ;Flag for enabling BOF RSI error messages
GLOBAL INT RSITECHIDX=1 ; Tech Channel used for RSI
;End RSI global Variables
```

3. To modify the function generator, assign the new number of the function generator to the global variable RSITECHIDX in the “RSI global Variables” section.

```
;RSI global Variables:
GLOBAL INT OV_RSI=30 ; Override for ST_SKIP/RET... movement after in-
terrupt
GLOBAL INT RSIBREAK=16 ;Index of break motion condition
GLOBAL BOOL RSIERRMSG=TRUE ;Flag for enabling BOF RSI error messages
GLOBAL INT RSITECHIDX=3 ; Tech Channel used for RSI
;End RSI global Variables
```

4. Save changes in the file by pressing the **Close** softkey.

6 Programming

6.1 Programming overview – signal processing

Overview

Step	Description
1	Declaring variables. (>>> 6.1.1 "Declaring variables" page 22)
2	Creating RSI objects or containers. (>>> 6.1.2 "Creating RSI objects and containers" page 22)
3	Linking signals (optional). (>>> 6.1.3 "Linking signals" page 24)
4	Reading / setting object parameters (optional). (>>> 6.1.4 "Reading / setting object parameters" page 25)
5	Activating / deactivating RSI objects or containers (optional). (>>> 6.1.5 "Activating/deactivating RSI objects or containers" page 25)
6	Deleting RSI objects or containers (optional). (>>> 6.1.6 "Deleting RSI objects or containers" page 26)
7	Programming RSI motions. (>>> 6.1.8 "Programming an RSI motion" page 29)
8	Activating / deactivating signal processing. (>>> 6.1.9 "Activating / deactivating signal processing" page 29)

Description

The following elements are required for signal processing:

- RSI objects
- Links
- Containers
- RSI commands (optional)

More detailed information about the RSI objects and commands can be found in the file ...\DOC\rsiCommands.chm on the CD-ROM.

The RSI objects must be created in the KRL program. Each RSI object has a functionality and corresponding signal inputs and/or outputs. An RSI object can also have object parameters to adapt the functionality of the RSI object.

The signal inputs and outputs of the RSI objects must be linked to one another for a signal flow. Vital inputs must be linked when the RSI object is created (e.g. the first 2 inputs of an AND operation). Additional optional inputs can be linked subsequently.

Containers are used to group RSI objects together and structure them. All RSI objects in a single container can be deleted, activated or deactivated simultaneously. The RSI objects for the signal processing must be located in a container. Global container 0 is automatically created and activated when the system is booted. All RSI objects created in container 0 are calculated when signal processing is activated. Other containers with other RSI objects can be created inside container 0. A container can contain a maximum of 254 RSI objects.

If other containers are created in global container 0, these other containers and all RSI objects inside them are initially deactivated. To activate them, the RSI command ST_ENABLE is used.

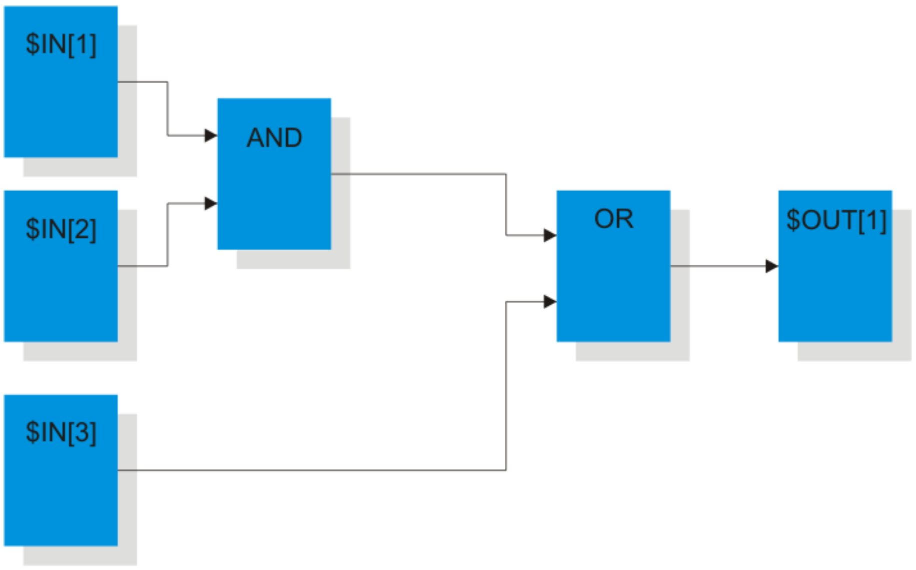


Fig. 6-1: Signal processing example

Element	Description
$IN[1] ... $IN[3]	RSI objects for reading the states of digital inputs $IN[1] ... $IN[3].
$OUT[1]	RSI object for setting digital output $OUT[1].
AND	ANDing
OR	ORing

6.1.1 Declaring variables

Description

The following variables must be declared:

Variable	Data type
Variable for the return values of the RSI commands	RSIERR
Variables for the object IDs of the RSI objects	INTEGER
Variables for the object parameters	Dependent on the RSI object
If RSI commands are used, variables for the RSI commands	Dependent on the RSI command

6.1.2 Creating RSI objects and containers

Precondition

- Expert user group

Procedure

1. Open a program.
2. Create RSI objects and containers for the signal processing.
3. Save changes in the program by pressing the **Close** softkey.

Syntax

Return_value=ST_*Object_name*(*Object_ID*, *Container_ID*, *<[ObjectX1_ID*, *ObjectX1_OUT_Index*, *...*, *ObjectXn_ID*, *ObjectXn_OUT_Index]><[Parameter 1*, *...*, *Parameter n]>*)

Explanation of the syntax

Element	Description
Return_value	Type: RSIERR The return value contains the error code that is transferred after an RSI command has been executed.
Object_name	Name of the RSI object, e.g. ST_ANAIN, ST_AND, ST_MAP2DIGOUT, etc.
Object_ID	Type: INT Variable for the object ID in order to access the RSI object. The value for the object ID is automatically assigned by the robot system when the RSI object is created.
Container_ID	Number of the container in which the RSI object is created.
ObjectX1_ID … *ObjectXn_ID*	Object ID of the RSI object that serves as the signal source. The object ID of the signal source must be specified for each signal input of an RSI object.
ObjectX1_OUT_Index … *ObjectXn_OUT_Index*	Number of the signal output of the RSI object that serves as the signal source. The signal output of the signal source must be specified for each signal input of an RSI object.
Parameter 1 … *Parameter n*	Object parameter for adapting the function of the RSI object.

Example

```
DEF Program( )
DECL RESIERR RET
DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID

INI

RET=ST_DIGIN(OBJECT1_ID,0,1,0,RSIUNIT_No)
RET=ST_DIGIN(OBJECT2_ID,0,2,0,RSIUNIT_No)
RET=ST_OR(OBJECT3_ID,0,OBJECT1_ID,1,OBJECT2_ID,1)
RET=ST_MAP2DIGOUT(OBJECT4_ID,0,OBJECT3_ID,1,3,0)

PTP HOME  Vel= 100 % DEFAULT

PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
LIN P3  Vel= 0.5 m/s CPDAT2 Tool[1] Base[1]
CIRC P4 P5  Vel= 0.5 m/s CPDAT3 Tool[1] Base[1]
LIN P6 CONT Vel= 0.5 m/s CPDAT4 Tool[1] Base[1]
PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]

PTP HOME  Vel= 100 % DEFAULT

END
```

Line	Description
6	RSI object ST_DIGIN reads a digital input. Parameters: ■ OBJECT1_ID: Object ID of ST_DIGIN ■ 0: RSI object in container 0 ■ 1: Reads value of bit 1 ■ 0: Reads value of a single bit ■ RSIUNIT_No: No unit
7	RSI object ST_DIGIN reads a digital input. Parameters: ■ OBJECT2_ID: Object ID of ST_DIGIN ■ 0: RSI object in container 0 ■ 2: Reads value of bit 2 ■ 0: Reads value of a single bit ■ RSIUNIT_No: No unit
8	RSI object ST_OR links the signal outputs of the ST_DIGIN RSI objects with a logic OR. Parameters: ■ OBJECT3_ID: Object ID of ST_OR ■ 0: RSI object in container 0 ■ OBJECT1_ID: Object ID of the signal source ■ 1: Number of the output of the signal source ■ OBJECT2_ID: Object ID of the signal source ■ 1: Number of the output of the signal source
9	RSI object ST_MAP2DIGOUT sends the result of the OR operation to a digital output. Parameters: ■ OBJECT4_ID: Object ID of ST_MAP2DIGOUT ■ 0: RSI object in container 0 ■ OBJECT3_ID: Object ID of the signal source ■ 1: Number of the output of the signal source ■ 3: Outputs value of bit 3 ■ 0: Outputs value of a single bit
11 … 19	Programmed motions

6.1.3 Linking signals

Description

The optional signal inputs of an RSI object can be linked subsequently.

The following RSI commands are available:

RSI command	Description
ST_NEWLINK	Links the signal output of an RSI object to an optional signal input of a different RSI object.

RSI command	Description
ST_DELLINK	Deletes the link between a signal output of an RSI object and the optional signal input of a different RSI object.
ST_CHANGELINK	Modifies the signal source of a link to an optional signal input. Precondition: ■ Signal processing is deactivated.

Precondition

- Expert user group

Procedure

1. Open a program.
2. Program RSI commands after the RSI objects.
3. Save changes in the program by pressing the **Close** softkey.

6.1.4 Reading / setting object parameters

Description

The object parameters of an RSI object can be used to influence its functionality. The object parameters are set when the RSI object is created. If an object parameter is not set when the RSI object is created, the default value automatically applies. The object parameters of an RSI object can be read using their parameter ID.

The following RSI commands are available:

RSI command	Description
ST_GETPARAM	Reads the REAL value of an object parameter.
ST_GETPARAMINT	Reads the INT value of an object parameter.
ST_SETPARAM	Sets the object parameter of an RSI object to a user-defined value.

Precondition

- Expert user group

Procedure

1. Open a program.
2. Program RSI commands after the RSI objects.
3. Save changes in the program by pressing the **Close** softkey.

6.1.5 Activating/deactivating RSI objects or containers

Description

Individual RSI objects or entire containers can be activated or deactivated in the signal flow.

The following RSI commands are available:

RSI command	Description
ST_ENABLE	Activates an RSI object or a container in order to integrate the RSI objects into the signal processing.
ST_DISABLE	Deactivates an RSI object or a container in order to exclude the RSI objects from the signal processing.

Precondition

- User group “Expert”
- For activation of an RSI object: the preceding RSI objects in the signal flow are active.

- For deactivation of an RSI object: the subsequent RSI objects in the signal flow are deactivated.

Procedure

1. Open a program.
2. Program RSI commands after the RSI objects.
3. Save changes in the program by pressing the **Close** softkey.

6.1.6 Deleting RSI objects or containers

Description

Individual RSI objects or entire containers can be deleted.

Precondition:

- An RSI object can be deleted if it is not followed in the signal flow by other RSI objects.

The following RSI commands are available:

RSI command	Description
ST_DELOBJ	Deletes an RSI object or a container.
ST_RESET	Deletes the entire RSI context in the program.

Precondition

- Expert user group

Procedure

1. Open a program.
2. Program RSI commands after the RSI objects.
3. Save changes in the program by pressing the **Close** softkey.

6.1.7 Overview of RSI motions

KUKA.RobotSensorInterface can be used to influence the programmed path of the robot by means of sensor data.

The following RSI commands are used for programming the RSI motions:

RSI motion	RSI command	Description
- - - -	ST_BREAKMOVE	The RSI object defines the sensor event that is used to terminate an RSI motion.
RSI motion to the next point but one	ST_SKIPPTP, ST_SKIPLIN, ST_SKIPCIRC	When the sensor event occurs, the robot stops the RSI motion and moves directly to the next point but one. (>>> Fig. 6-2)
RSI motion back to the start point	ST_RETPTP, ST_RETLIN, ST_RETCIRC	When the sensor event occurs, the robot stops the RSI motion and moves directly back to the start point of the RSI motion. (>>> Fig. 6-3)
Relative RSI motion	ST_PTPREL, ST_LINREL, ST_CIRCREL	The robot follows the coordinates of the RSI motion relative to the current position. When the sensor event occurs, the robot stops the relative RSI motion and moves directly to the next point but one. (>>> Fig. 6-4)

RSI motion	RSI command	Description
Sensor-guided motion	ST_SKIPSENS, ST_RETSENS, ST_MOVESENS	The robot moves solely in accordance with the sensor data and does not move to a defined end point. If KUKA.RobotSensorInterface calculates a path correction or axis angle correction, the robot moves. If there is no path correction or axis angle correction, the robot remains stationary. (>>> Fig. 6-5) When the sensor event occurs, the robot stops the sensor-guided motion: ■ ST_SKIPSENS: the robot moves directly to the next point but one. ■ ST_RETSENS: the robot moves directly back to the start point of the sensor-guided motion. ■ ST_MOVESENS: the response of the robot is defined with the parameter MODE: ■ **0**: the robot moves directly to the next point but one. ■ **1**: the robot moves via the interrupt to the next point but one. ■ **2**: the robot moves directly back to the start point of the sensor-guided motion.
Path correction, axis angle correction	ST_PATHCORR, ST_AXISCORR	If the robot is moving along the programmed path and KUKA.RobotSensorInterface calculates a path correction or axis angle correction, the robot corrects its path or axis angles. The path or axis angle correction can be absolute in relation to the programmed path, or relative to the correction of the previous interpolation cycle. (>>> Fig. 6-6)

Paths

RSI motion to the next point but one:

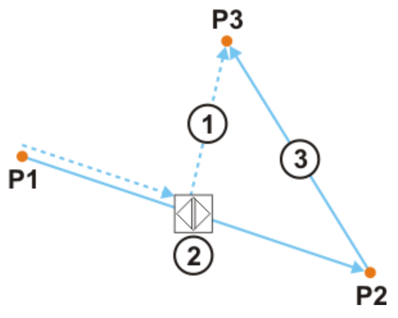


Fig. 6-2: Path of an RSI motion to the next point but one

1	RSI motion to the next point but one
2	Sensor event
3	Programmed path (terminated by sensor event)

RSI motion back to the start point:

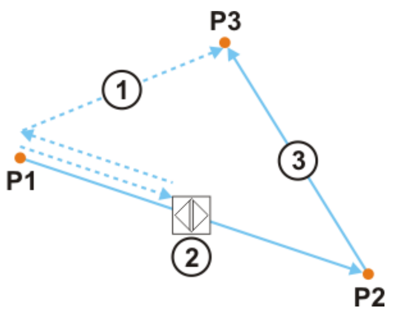


Fig. 6-3: Path of an RSI motion back to the start point

1	RSI motion back to the start point
2	Sensor event
3	Programmed path (terminated by sensor event)

Relative RSI motion relative to the current position:

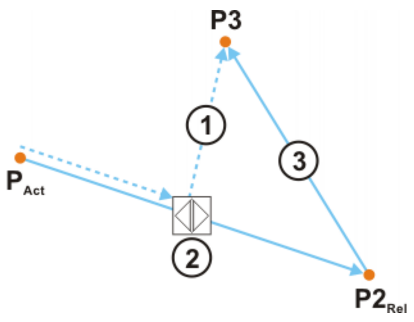


Fig. 6-4: Path of a relative RSI motion

1	RSI motion to the next point but one
2	Sensor event
3	Programmed path (terminated by sensor event)
P_{Act}	Current position: start point of the relative motion
$P2_{Rel}$	End point of the relative motion

Sensor-guided motion:

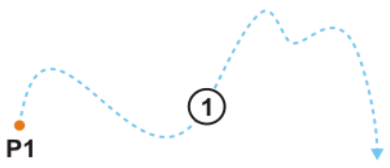


Fig. 6-5: Path of a sensor-guided motion

1	Sensor-guided robot motion
P1	Start point of the sensor-guided motion

Path correction:

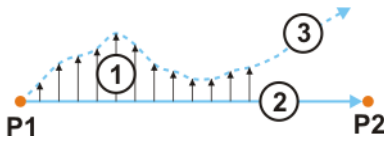


Fig. 6-6: Path of a path correction

1	Path corrections calculated with KUKA.RobotSensorInterface
2	Programmed path
3	Corrected path

6.1.8 Programming an RSI motion

Precondition

- User group "Expert"
- Program is selected.

Procedure

1. Move the TCP to the position that is to be taught as a point.
2. Teach point, e.g. P8.
3. Note the name of the taught point.
4. Replace motion instruction with RSI motion command, e.g. ST_SKIPLIN.
5. The coordinates of the taught point are saved in the data list, e.g. with the variable name XP8.

   Accept the variable name as the point name in the RSI motion command.
6. Save changes in the program by pressing the **Close** softkey.

Example

```
DEF Program( )
DECL RSIERR RET
DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID,OBJECT5_ID

INI

RET=ST_DIGOUT(OBJECT1_ID,0,1,0,RSIUNIT_No)
 ...
RET=ST_BREAKMOVE(OBJECT5_ID,0,OBJECT3_ID,1)

PTP HOME  Vel= 100 % DEFAULT

PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
 ...
PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]

ST_SKIPLIN(XP8)

PTP HOME  Vel= 100 % DEFAULT

END
```

6.1.9 Activating / deactivating signal processing

Description

Once the signal processing has been completely created, it must be activated and then deactivated again.

Signal processing with KUKA.RobotSensorInterface can be activated and deactivated more than once within a program.

The following RSI commands are available:

RSI command	Description
ST_ON	Activates the signal processing. The signals are processed and evaluated in the interpolation cycle. Path corrections with the RSI object ST_PATHCORR always refer to the BASE coordinate system.
ST_ON1	Activates the signal processing. The signals are processed and evaluated in the interpolation cycle. Path corrections with the RSI object ST_PATHCORR can refer to the following coordinate systems. ■ BASE coordinate system ■ TOOL coordinate system ■ Tool-based technological system (TTS coordinate system) ■ WORLD coordinate system The path corrections can be absolute with respect to the programmed path or relative to the corrected path.
ST_OFF	Deactivates the signal processing.

Precondition

- Expert user group

Procedure

1. Open a program.
2. Activate signal processing before the RSI motions by means of ST_ON or ST_ON1.
3. Deactivate signal processing after the RSI motions by means of ST_OFF.
4. Save changes in the program by pressing the **Close** softkey.

Example

```
DEF Program( )
DECL RSIERR RET
DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID,OBJECT5_ID

INI

RET=ST_DIGOUT(OBJECT1_ID,0,1,0,RSIUNIT_No)
 ...
RET=ST_BREAKMOVE(OBJECT5_ID,0,OBJECT3_ID,1)

PTP HOME  Vel= 100 % DEFAULT

PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
 ...
PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]

RET=ST_ON()

ST_SKIPLIN(XP8)

RET=ST_OFF()

PTP HOME  Vel= 100 % DEFAULT

END
```

6.2 Programming overview – data exchange via Ethernet

Overview

Step	Description
1	Create RSI object ST_ETHERNET. (>>> 6.2.2 "Creating ST_ETHERNET" page 32)
2	Set object parameters. (>>> 6.2.3 "Object parameters of ST_ETHERNET" page 33)
3	Define communication parameters. (>>> 6.2.4 "Communication parameters of ST_ETHERNET" page 35)
4	Configure XML structure for sending data. (>>> 6.2.5 "Object inputs of ST_ETHERNET" page 35)
5	Activate internal read function (optional). (>>> 6.2.6 "Activating the internal read function" page 37)
6	Configure XML structure for receiving data. (>>> 6.2.7 "Object outputs of ST_ETHERNET" page 38)
7	Activate internal write function (optional). (>>> 6.2.8 "Activating the internal write function" page 40)

Description

The real-time communication between the robot controller and the external system is implemented using the RSI object ST_ETHERNET. The RSI object ST_ETHERNET must be created and configured in the KRL program.

The RSI object ST_ETHERNET has the following functionalities:

- Message in the event of late data packets arriving acyclically
- Mode for data exchange: “Normal Mode” and “Fast Mode”
- Definition of the communication parameters in an XML file
- User-defined assignment of the object inputs and object outputs
- Selection of the transfer protocol: TCP or UDP
- Bidirectional and unidirectional communication

On creating the RSI object, the connection with the external system is established. The connection is only terminated when ST_ETHERNET is deleted.

If the RSI object ST_ETHERNET is deleted in the KRL program, it cannot be created again until 2 s after it has been deleted. If ST_ETHERNET is created within the 2 s, the network interface may become blocked.

Elements of the RSI object ST_ETHERNET:

- Object parameters: for adapting the function of ST_ETHERNET
- Object inputs: for loading data from the RSI context and forwarding them to the external system
- Object outputs: for forwarding data received from the external system to RSI objects
- XML file: for configuring the object inputs and outputs and the communication parameters

  With signal processing activated, ST_ETHERNET sends and receives a user-defined data set in the interpolation cycle. No fixed data frame is specified. The user must configure the data set in an XML file.

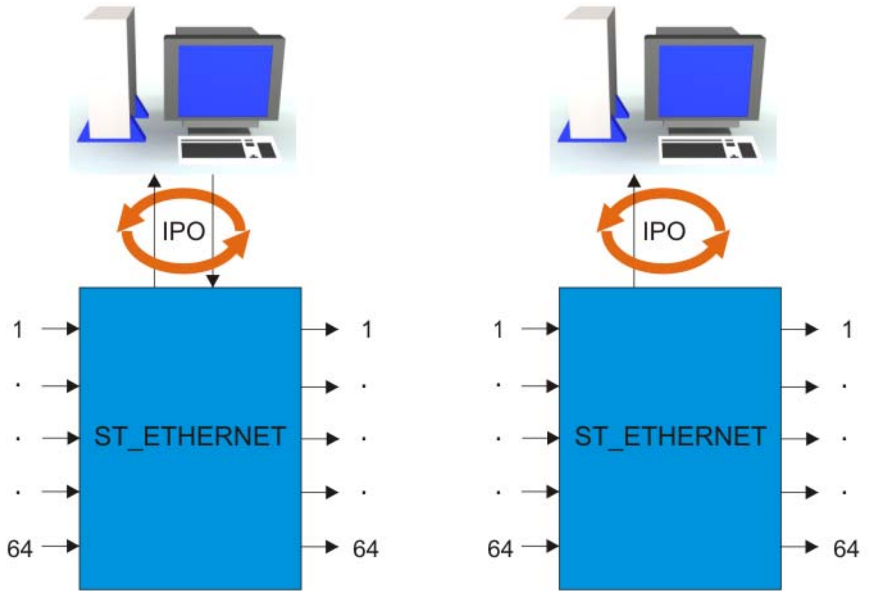


Fig. 6-7: RSI object ST_ETHERNET

6.2.1 Structure of the configuration file

Overview

To enable the robot controller to communicate with the external system, the user must define an XML file in the directory C:\KRC\ROBOTER\INIT. The configuration file is specified and loaded when the RSI object ST_ETHERNET is created.

The structure of the XML file is fixed:

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

Section	Description
<CONFIG ... </CON-FIG>	Definition of the communication parameters (>>> 6.2.4 "Communication parameters of ST_ETHERNET" page 35)
<SEND> ... </SEND>	Definition of the object inputs of ST_ETHERNET (>>> 6.2.5 "Object inputs of ST_ETHERNET" page 35)
<RECEIVE> ... </RE-CEIVE>	Definition of the object outputs of ST_ETHERNET (>>> 6.2.7 "Object outputs of ST_ETHERNET" page 38)

6.2.2 Creating ST_ETHERNET

Syntax

Return_value`=ST_ETHERNET(`*Object_ID*`,`*Container_ID*`,`*Configuration_file*`)`

Explanation of the syntax

Element	Description
Return_value	Type: RSIERR The return value contains the error code that is transferred after an RSI command has been executed.
Object_ID	Type: INT Variable for the object ID in order to access the RSI object. The value for the object ID is automatically assigned by the robot system when the RSI object is created.
Container_ID	Number of the container in which the RSI object is created.
Configuration _file	Type: CHAR Variable for the name of the configuration file. The name of the configuration file can also be entered directly, e.g. “RSIEthernet.xml”.

Example

```
1  DEF Program( )
2  DECL RSIERR RET
3  DECL INT hEthernet
4
5  INI
6
7  RET = ST_ETHERNET(hEthernet,0,"RSIEthernet.xml")
8
9  END
```

Line	Description
2	Variable for the return values
3	Variable for the object ID
7	Creation of the RSI object ST_ETHERNET

6.2.3 Object parameters of ST_ETHERNET

Description

The object parameters of the RSI object ST_ETHERNET are used to adapt the function in the program sequence. The object parameters are set using the RSI command ST_SETPARAM.

Object parameter	Description
eERXmaxLatePack-ages	Maximum number of data packets in a block that may arrive late at the robot controller. ■ **1 … 65,000** (default value: 10) If the value is exceeded, an error message is generated and data exchange in the RSI context is terminated.
eERXmaxLateInPer-cent	Maximum percentage of data packets in the sample which are allowed to arrive late. ■ **1 … 65,000** (default value: 10) If the value is exceeded, a message is gener-ated only if the output of RSI-specific messages has been enabled.

Object parameter	Description
eERXmaxFieldOfView	Sample size ■ **1 … 65,000** (default value: 1,000) **Example**: If the default value is used, 1,000 communication cycles are monitored and the data packets that arrive late are counted. If the parameter eERXmaxLateInPercent is set to 10, a message is generated after 101 late data packets.
eERXerrorFlag	Number of the flag that is set in the case of a transmission error ■ **1 … 999** (default: flag deactivated)
eERXFastCycle	**FALSE**: The RSI object operates in “Normal Mode” (default). The external system has 12 ms to respond to a data packet. If the robot controller receives no response within this period, the data packet is classified as late. **TRUE**: The RSI object operates in “Fast Mode”. Sent and received data are processed within the same cycle. The external system has 2 ms to respond to a data packet. If the robot controller receives no response within this period, the data packet is classified as late.
eERXprecision	Precision of the transferred values, with reference to the number of decimal places ■ **0 … 16** (default value: 4) **Example**: Value to be transferred = 1.23456 ■ 0: Value transferred = 1 ■ 4: Value transferred = 1.2345

Example

```
1  DEF Program( )
2  DECL RSIERR RET
3  DECL INT hEthernet
4
5  INI
6
7  RET=ST_ETHERNET(hEthernet,0,"RSIEthernet.xml")
8
9  RET=ST_SETPARAM(hEthernet,eERXmaxLatePackages,3)
10 RET=ST_SETPARAM(hEthernet,eERXmaxLateInPercent,8)
11 RET=ST_SETPARAM(hEthernet,eERXmaxFieldOfView,2345)
12 RET=ST_SETPARAM(hEthernet,eERXFastCycle,1)
13 RET=ST_SETPARAM(hEthernet,eERXerrorFlag,99)
14 RET=ST_SETPARAM(hEthernet,eERXprecision,4)
15
16 END
```

Line	Description
2	Variable for the return values
3	Variable for the object ID
7	Creation of the RSI object ST_ETHERNET
9 ... 14	Setting the object parameters of ST_ETHERNET

6.2.4 Communication parameters of ST_ETHERNET

Description

The following communication parameters can be defined in the section <CONFIG ... </CONFIG> of the configuration file:

Parameter	Description
IP_NUMBER	IP address of the external system
PORT	Port number of the external system
PROTOCOL	Type of transfer protocol ■ **TCP**: When ST_ETHERNET is created, a connection to the external system is established. ■ **UDP**: When ST_ETHERNET is created, no connection to the external system is established.
SENTYPE	Identifier of the external system; freely selectable The robot controller checks this identifier for every data packet it receives.
PROTCOLLENGTH	Transmission of the byte length of the protocol before an XML string is sent. This can simplify the programming of stream sockets. ■ **ON**: The protocol length is sent ■ **OFF**: The protocol length is not sent
ONLYSEND	Direction of data exchange ■ **TRUE**: The robot controller sends data and may not receive any data. The object outputs of ST_ETHERNET are reset. ■ **FALSE**: The robot controller sends and receives data (default).

Example

```
1  <CONFIG>
2    <IP_NUMBER>192.0.1.2</IP_NUMBER>
3    <PORT>6008</PORT>
4    <PROTOCOL>TCP</PROTOCOL>
5    <SENTYPE>ImFree</SENTYPE>
6    <PROTCOLLENGTH>Off</PROTCOLLENGTH>
7    <ONLYSEND>FALSE</ONLYSEND>
8  </CONFIG>
```

Line	Description
2	IP address of the external system: 192.0.1.2
3	Port number of the external system: 6008
4	Protocol: TCP
5	Identifier of the external system: ImFree
6	The protocol length is not sent
7	Data exchange in 2 directions: send and receive

6.2.5 Object inputs of ST_ETHERNET

Description

To configure the XML structure for sending data, up to 64 object inputs of ST_ETHERNET can be freely defined. For this, the inputs are linked to RSI objects from the RSI context. The XML format to be sent is generated automatically by the robot controller in accordance with the configuration. The data at the object inputs are sent in an XML string to the external system.

The following parameters of the incoming RSI signal must be defined in the section <SEND> ... </SEND> of the configuration file:

Parameter	Description
TAG	Name of the tag that is to be generated The following notations are possible: ■ TAG="Out": The following tag is generated in the XML string: <Out></Out> ■ TAG="Out.o1": The following tag with attribute is generated in the XML string: <Out o1="" />
TYPE	Data type of the incoming RSI signal Permissible data types are: ■ BOOL ■ LONG ■ DOUBLE
INDX	Number of the object input Example: ■ INDX="54": The value of the RSI signal is read from object input 54. **Note**: The numbering of the object inputs must be consecutive.
UNIT	Unit of the RSI signal A decimal or hexadecimal value must be entered.

Example

```
<SEND>
 <ELEMENTS>
  <ELEMENT TAG="Out.o1" TYPE="BOOL" INDX="1" UNIT="5467" />
  <ELEMENT TAG="Out.o2" TYPE="BOOL" INDX="2" UNIT="5467" />
  <ELEMENT TAG="Out.o3" TYPE="BOOL" INDX="3" UNIT="5467" />
  <ELEMENT TAG="Out.o4" TYPE="BOOL" INDX="4" UNIT="5467" />
  <ELEMENT TAG="Out.o5" TYPE="BOOL" INDX="5" UNIT="5467" />
  <ELEMENT TAG="FTC.Fx" TYPE="DOUBLE" INDX="6" UNIT="5467" />
  <ELEMENT TAG="FTC.Fy" TYPE="DOUBLE" INDX="7" UNIT="5467" />
  <ELEMENT TAG="FTC.Fz" TYPE="DOUBLE" INDX="8" UNIT="5467" />
  <ELEMENT TAG="FTC.Mx" TYPE="DOUBLE" INDX="9" UNIT="5467" />
  <ELEMENT TAG="FTC.My" TYPE="DOUBLE" INDX="10" UNIT="5467" />
  <ELEMENT TAG="FTC.Mz" TYPE="DOUBLE" INDX="11" UNIT="5467" />
  <ELEMENT TAG="Override" TYPE="LONG" INDX="12" UNIT="5467" />
 </ELEMENTS>
</SEND>
```

The following XML structure is generated by the robot controller and sent to the external system:

```
<Rob TYPE="KUKA">
 <Out o1="0" o2="1" o3="1" o4="" o5="0" />
 <FTC Fx="1.234" Fy="54.75" Fz="345.76" Mx="2346.6" My="" Mz="3546" />
 <Override>90</Override>
 <IPOC>123645634563</IPOC>
</Rob>
```

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

6.2.6 Activating the internal read function

Description

Large data sets can be structured by activating the internal read function of ST_ETHERNET. This simplifies linking with the RSI objects from the RSI context and saves space in the object inputs of ST_ETHERNET.

The read function is activated using keywords in the “TAG” attribute in the section <SEND> ... </SEND> of the configuration file.

The keywords must not be used for freely parameterizing the object inputs from the RSI context.

The following keywords are available:

Keyword	Description
DEF_RIst	Send the Cartesian actual position
DEF_RSol	Send the Cartesian command position
DEF_AIPos	Send the axis-specific actual position of robot axes A1 to A6
DEF_ASPos	Send the axis-specific command position of robot axes A1 to A6
DEF_EIPos	Send the axis-specific actual position of external axes E1 to E6
DEF_ESPos	Send the axis-specific command position of external axes E1 to E6
DEF_MACur	Send the motor currents of robot axes A1 to A6
DEF_MECur	Send the motor currents of external axes E1 to E6
DEF_Delay	Send the number of late data packets
DEF_Tech.C1 ... DEF_Tech.C6	Send the technology parameters in the advance run with the function generators 1 to 6
DEF_Tech.T1 ... DEF_Tech.T6	Send the technology parameters in the main run with the function generators 1 to 6

Notation in the configuration file:

```
<ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_RSol" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_AIPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_ASPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_EIPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_ESPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_MACur" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_MECur" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_Delay" TYPE="LONG" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
...
<ELEMENT TAG="DEF_Tech.C6" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_Tech.T1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
...
<ELEMENT TAG="DEF_Tech.T6" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
```

If the read function is activated, the robot controller generates the following XML structure in the send protocol:

```
<RIst X="0.0" Y="0.0" Z="0.0" A="0.0" B="0.0" C="0.0" />
<RSol 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" />
<ASPos A1="0.0" A2="0.0" A3="0.0" A4="0.0" A5="0.0" A6="0.0" />
<EIPos E1="0.0" E2="0.0" E3="0.0" E4="0.0" E5="0.0" E6="0.0" />
<ESPos E1="0.0" E2="0.0" E3="0.0" E4="0.0" E5="0.0" E6="0.0" />
<MACur A1="1.0" A2="1.0" A3="1.0" A4="1.0" A5="1.0" A6="1.0" />
<MECur E1="1.0" E2="1.0" E3="1.0" E4="1.0" E5="1.0" E6="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" />
...
<Tech T61="0.0" T62="0.0" T63="0.0" T64="0.0" T65="0.0" T66="0.0" _
      T67="0.0" T68="0.0" T69="0.0" T610="0.0" />
<Tech C11="0.0" C12="0.0" C13="0.0" C14="0.0" C15="0.0" C16="0.0" _
      C17="0.0" C18="0.0" C19="0.0" C110="0.0" />
...
<Tech C51="0.0" C52="0.0" C53="0.0" C54="0.0" C55="0.0" C56="0.0" _
      C57="0.0" C58="0.0" C59="0.0" C510="0.0" />
```

Example

```
<SEND>
 <ELEMENTS>
  <ELEMENT TAG="Out.o1" TYPE="BOOL" INDX="1" UNIT="5467" />
  <ELEMENT TAG="Out.o2" TYPE="BOOL" INDX="2" UNIT="5467" />
  <ELEMENT TAG="Out.o3" TYPE="BOOL" INDX="3" UNIT="5467" />
  <ELEMENT TAG="Out.o4" TYPE="BOOL" INDX="4" UNIT="5467" />
  <ELEMENT TAG="Out.o5" TYPE="BOOL" INDX="5" UNIT="5467" />
  <ELEMENT TAG="FTC.Fx" TYPE="DOUBLE" INDX="6" UNIT="5467" />
  <ELEMENT TAG="FTC.Fy" TYPE="DOUBLE" INDX="7" UNIT="5467" />
  <ELEMENT TAG="FTC.Fz" TYPE="DOUBLE" INDX="8" UNIT="5467" />
  <ELEMENT TAG="FTC.Mx" TYPE="DOUBLE" INDX="9" UNIT="5467" />
  <ELEMENT TAG="FTC.My" TYPE="DOUBLE" INDX="10" UNIT="5467" />
  <ELEMENT TAG="FTC.Mz" TYPE="DOUBLE" INDX="11" UNIT="5467" />
  <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="Override" TYPE="LONG" INDX="12" UNIT="5467" />
 </ELEMENTS>
</SEND>
```

The following XML structure is generated and sent by the robot controller:

```
<Rob TYPE="KUKA">
 <Out o1="0" o2="1" o3="1" o4="" o5="0" />
 <RIst X=“12.6“ Y=“234.456“ Z=“645.79“ A=“2.4“ B=“456.814“ C=“65.33“ />
 <FTC Fx="1.234" Fy="54.75" Fz="345.76" Mx="2346.6" My="" Mz="3546" />
 <Override>90</Override>
 <IPOC>123645634563</IPOC>
</Rob>
```

6.2.7 Object outputs of ST_ETHERNET

Description

To configure the XML structure for receiving data, up to 64 object outputs of ST_ETHERNET can be freely defined. For this, the outputs are linked to RSI objects from the RSI context. The robot controller expects an XML format which conforms to the configuration. The data at the object outputs are sent in an XML string to the robot controller.

The following parameters of the outgoing RSI signal must be defined in the section <RECEIVE> ... </RECEIVE> of the configuration file:

Parameter	Description
TAG	Name of the tag that is to be generated The following notations are possible: ■ TAG="Out": The following tag is generated in the XML string: <Out></Out> ■ TAG="Out.o1": The following tag with attribute is generated in the XML string: <Out o1="" />
TYPE	Data type of the outgoing RSI signal Permissible data types are: ■ BOOL ■ STRING ■ LONG ■ DOUBLE
INDX	Number of the object output Example: ■ INDX="3": The value of the RSI signal is sent to object output 3. **Note**: The numbering of the object outputs must be consecutive.
UNIT	Unit of the RSI signal A decimal or hexadecimal value must be entered.
HOLDON	Behavior of the object output with regard to invalid data packets that arrive late ■ HOLDON="0": The output is reset. ■ HOLDON="1": The most recent valid value to arrive remains at the output.

Example

```
<RECEIVE>
 <ELEMENTS>
  <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" UNIT="1" HOLDON="1" />
  <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" UNIT="1" HOLDON="1" />
  <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" UNIT="1" HOLDON="1" />
  <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" UNIT="0" HOLDON="1" />
  <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" UNIT="0" HOLDON="1" />
  <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" UNIT="0" HOLDON="1" />
  <ELEMENT TAG="AK.A1" TYPE="DOUBLE" INDX="7" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="AK.A2" TYPE="DOUBLE" INDX="8" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="AK.A3" TYPE="DOUBLE" INDX="9" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="AK.A4" TYPE="DOUBLE" INDX="10" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="AK.A5" TYPE="DOUBLE" INDX="11" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="AK.A6" TYPE="DOUBLE" INDX="12" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E1" TYPE="DOUBLE" INDX="13" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E2" TYPE="DOUBLE" INDX="14" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E3" TYPE="DOUBLE" INDX="15" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E4" TYPE="DOUBLE" INDX="16" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E5" TYPE="DOUBLE" INDX="17" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="EK.E6" TYPE="DOUBLE" INDX="18" UNIT="0" HOLDON="0" />
  <ELEMENT TAG="DiO" TYPE="LONG" INDX="19" UNIT="0" HOLDON="1" />
 </ELEMENTS>
</RECEIVE>
```

The following XML structure is generated and is expected by the robot controller:

```
<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 set with the keyword IPOC at the object output is checked. The data packet is only valid if the time stamp corresponds to the time stamp sent previously.

6.2.8 Activating the internal write function

Description

The internal write function of ST_ETHERNET is activated using keywords in the “TAG” attribute in the section <RECEIVE> ... </RECEIVE> of the configuration file.

The keywords must not be used for freely parameterizing the object outputs from the RSI context.

The following keywords are available:

Keyword	Description
DEF_EStr	Generation of a message in the message window If <EStr> ...</EStr>: Message for information If <EStr>Error: ...</EStr>: Acknowledgeable error message; the robot is stopped.
DEF_Tech.C1 ... DEF_Tech.C6	Write the technology parameters in the advance run with the function generators 1 to 6
DEF_Tech.T1 ... DEF_Tech.T6	Write the technology parameters in the main run with the function generators 1 to 6

Notation in the configuration file:

```
<ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
...
<ELEMENT TAG="DEF_Tech.C6" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
<ELEMENT TAG="DEF_Tech.T1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
...
<ELEMENT TAG="DEF_Tech.T6" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
```

If the write function is activated, the robot controller expects the following XML structure in the receive protocol:

```
<EStr>Message!</EStr>
<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" />
...
<Tech T61="0.0" T62="0.0" T63="0.0" T64="0.0" T65="0.0" T66="0.0" _
      T67="0.0" T68="0.0" T69="0.0" T610="0.0" />
<Tech C11="0.0" C12="0.0" C13="0.0" C14="0.0" C15="0.0" C16="0.0" _
      C17="0.0" C18="0.0" C19="0.0" C110="0.0" />
...
<Tech C51="0.0" C52="0.0" C53="0.0" C54="0.0" C55="0.0" C56="0.0" _
      C57="0.0" C58="0.0" C59="0.0" C510="0.0" />
```

Example

```
<RECEIVE>
 <ELEMENTS>
  <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" UNIT="1" HOLDON="1" />
  <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" UNIT="1" HOLDON="1" />
  <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" UNIT="1" HOLDON="1" />
   <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" UNIT="0" />
 <ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
 </ELEMENTS>
</RECEIVE>
```

The following XML structure is generated and is expected by the robot controller:

```
<Sen Type="ImFree">
 <EStr/>
 <RKorr X="4" Y="7" Z="32" />
 <Tech C11="0.0" C12="0.0" C13="0.0" C14="0.0" C15="0.0" C16="0.0" _
       C17="0.0" C18="0.0" C19="0.0" C110="0.0" />
 <IPOC>123645634563</IPOC>
</Sen>
```

Since the <EStr/> tag is empty, no message is generated. The data in the <RKorr .../> tag are available at the output of the RSI object ST_ETHERNET. The technology parameters are written directly to the controller.

7 Example

7.1 Example program for RSI motions

```
1   DEF Program( )
2   DECL RSIERR RET
3   DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID,OBJECT5_ID
4
5   INI
6
7   RET=ST_DIGIN(OBJECT1_ID,0,1,0,RSIUNIT_No)
8   RET=ST_DIGIN(OBJECT2_ID,0,2,0,RSIUNIT_No)
9   RET=ST_OR(OBJECT3_ID,0,OBJECT1_ID,1,OBJECT2_ID,1)
10  RET=ST_MAP2DIGOUT(OBJECT4_ID,0,OBJECT3_ID,1,3,0)
11  RET=ST_BREAKMOVE(OBJECT5_ID,0,OBJECT3_ID,1)
12
13  PTP HOME  Vel= 100 % DEFAULT
14
15  PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
16  LIN P3  Vel= 0.5 m/s CPDAT2 Tool[1] Base[1]
17  CIRC P4 P5  Vel= 0.5 m/s CPDAT3 Tool[1] Base[1]
18  LIN P6 CONT Vel= 0.5 m/s CPDAT4 Tool[1] Base[1]
19  PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]
20
21  RET=ST_ON()
22
23  ST_SKIPLIN(XP8)
24  ST_RETLIN(XP8)
25  ST_LINREL {X 100}
23
24  RET=ST_OFF()
25
26  PTP HOME  Vel= 100 % DEFAULT
27
28  END
```

Line	Description
3	Variables for the object IDs
11	RSI object ST_BREAKMOVE for defining the sensor event that is used to terminate the RSI motion. Sensor event: the RSI object ST_OR generates a HIGH level signal.
23	RSI motion to the next point but one
24	RSI motion back to the start point
25	Relative RSI motion in X direction

7.2 Example program for sensor-guided motion

```
1   DEF Program( )
2   DECL RSIERR RET
3   DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID, _
             OBJECT5_ID,OBJECT6_ID,OBJECT7_ID
4
5   INI
6
7   RET=ST_DIGIN(OBJECT1_ID,0,1,0,RSIUNIT_No)
8   RET=ST_DIGIN(OBJECT2_ID,0,2,0,RSIUNIT_No)
9   RET=ST_OR(OBJECT3_ID,0,OBJECT1_ID,1,OBJECT2_ID,1)
10  RET=ST_MAP2DIGOUT(OBJECT4_ID,0,OBJECT3_ID,1,3,0)
11  RET=ST_BREAKMOVE(OBJECT5_ID,0,OBJECT3_ID,1)
12
13  RET=ST_DIGIN(OBJECT6_ID,0,2,1,RSIUNIT_m)
14  RET=ST_PATHCORR(OBJECT7_ID,0)
15  RET=ST_NEWLINK(OBJECT6_ID,1,OBJECT7_ID,1)
16
17  PTP HOME  Vel= 100 % DEFAULT
18
19  PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
20  LIN P3  Vel= 0.5 m/s CPDAT2 Tool[1] Base[1]
21  CIRC P4 P5  Vel= 0.5 m/s CPDAT3 Tool[1] Base[1]
22  LIN P6 CONT Vel= 0.5 m/s CPDAT4 Tool[1] Base[1]
23  PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]
24
25  RET=ST_ON1(#TOOL,1)
26
27  ST_MOVESENS(1)
28
29  RET=ST_OFF()
30
31  PTP HOME  Vel= 100 % DEFAULT
32
33  END
```

Line	Description
3	Variables for the object IDs
11	RSI object ST_BREAKMOVE for defining the sensor event that is used to terminate the sensor-guided motion. Sensor event: the RSI object ST_OR generates a HIGH level signal.
13	RSI object ST_DIGIN for recording the sensor data via digital inputs $IN[9] ... $IN[16] with a width of one byte
14	RSI object ST_PATHCORR for calculating the path correction from the recorded sensor data
15	RSI command ST_NEWLINK for creating a link between the RSI objects ST_DIGIN 6 and ST_PATHCORR
25	RSI command ST_ON1 for activating the signal processing and making the path correction relative to the TOOL coordinate system
27	Sensor-guided motion If the sensor-guided motion is terminated by the sensor event, the robot moves via the interrupt to the next point but one.

7.3 Example program for path correction

```
1   DEF Program( )
2   DECL RSIERR RET
3   DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID, _
             OBJECT5_ID,OBJECT6_ID
4
5   INI
6
7   RET=ST_DIGIN(OBJECT1_ID,0,1,0,RSIUNIT_No)
8   RET=ST_DIGIN(OBJECT2_ID,0,2,0,RSIUNIT_No)
9   RET=ST_OR(OBJECT3_ID,0,OBJECT1_ID,1,OBJECT2_ID,1)
10  RET=ST_MAP2DIGOUT(OBJECT4_ID,0,OBJECT3_ID,1,3,0)
11
12  RET=ST_DIGIN(OBJECT5_ID,0,2,1,RSIUNIT_m)
13  RET=ST_PATHCORR(OBJECT6_ID,0)
14  RET=ST_NEWLINK(OBJECT5_ID,1,OBJECT6_ID,1)
15
16  PTP HOME  Vel= 100 % DEFAULT
17
18  PTP P1 CONT Vel= 100 % PDAT1 Tool[1] Base[1]
19  LIN P3  Vel= 0.5 m/s CPDAT2 Tool[1] Base[1]
20  CIRC P4 P5  Vel= 0.5 m/s CPDAT3 Tool[1] Base[1]
21  LIN P6 CONT Vel= 0.5 m/s CPDAT4 Tool[1] Base[1]
22  PTP P7 CONT Vel= 100 % PDAT2 Tool[1] Base[1]
23
24  RET=ST_ON1(#TTS,1)
25
26  LIN XP8
27
28  RET=ST_OFF()
29
30  PTP HOME  Vel= 100 % DEFAULT
31
32  END
```

Line	Description
3	Variables for the object IDs
12	RSI object ST_DIGIN for recording the sensor data via digital inputs $IN[9] ... $IN[16] with a width of one byte
13	RSI object ST_PATHCORR for calculating the path correction from the recorded sensor data
14	RSI command ST_NEWLINK for creating a link between the RSI objects ST_DIGIN 6 and ST_PATHCORR
24	RSI command ST_ON1 for activating the signal processing and making the path correction relative to the tool-based coordinate system (TTS)
26	LIN motion with path correction

7.4 Example program for adapting the maximum path correction

```
1   DEF Program( )
2   DECL RSIERR RET
3   DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID
4
5   INI
6   PTP HOME
7
8   RET=ST_DIGIN(OBJECT1_ID,0,1,1,RSIUNIT_m)
9   RET=ST_PATHCORR(OBJECT2_ID,0)
10  RET=ST_NEWLINK(OBJECT1_ID,1,OBJECT2_ID,1)
11
12  RET=ST_SETPARAM(OBJECT2_ID,1,-25) ;Min corr. in x: -25 mmm
13  RET=ST_SETPARAM(OBJECT2_ID,7,25)  ;Max corr. in x: 25 mmm
14
15  LIN P1
16  RET=ST_ON()
17  ST_MOVESENS(1)
18
19  END
```

Line	Description
3	Variables for the object IDs
9	RSI object ST_PATHCORR for calculating the path correction from the recorded sensor data
12	RSI command ST_SETPARAM to assign the value -25 to object parameter 1 of ST_PATHCORR; object parameter 1 defines the lower limit for a correction in the X direction.
13	RSI command ST_SETPARAM to assign the value +25 to object parameter 7 of ST_PATHCORR; object parameter 7 defines the upper limit for a correction in the X direction.

7.5 Example program for transformation

Description

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

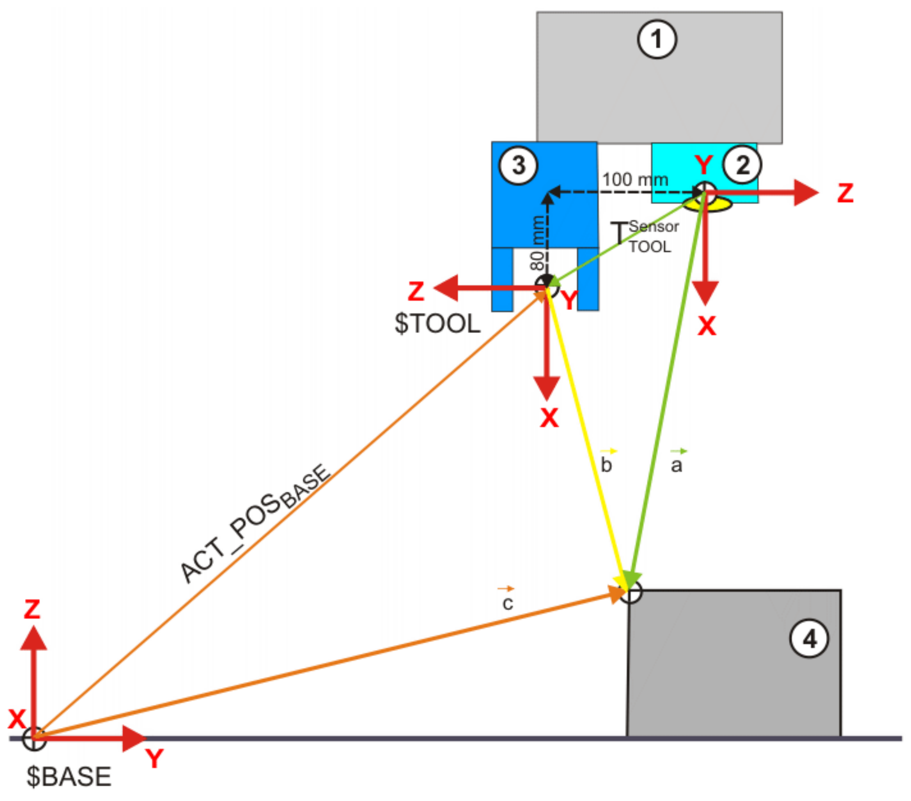


Fig. 7-1: Example of a transformation

1	Mounting flange	3	Tool
2	Sensor	4	Workpiece

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. These sensor data must be transformed from the sensor coordinate system to the current BASE of the robot controller.

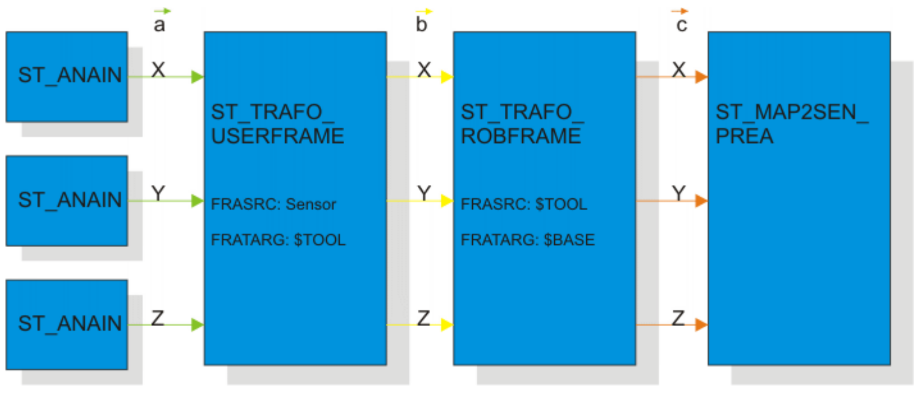


Fig. 7-2: Schematic structure of the RSI context for example transformation

RSI object	Description
ST_ANAIN	The RSI object ST_ANAIN reads the point coordinates acquired by the sensor.
ST_TRAFO_USERFRAME	The RSI object ST_TRAFO_USERFRAME transforms the acquired coordinates from the sensor coordinate system to the TOOL coordinate system of the robot controller. The position of the TOOL coordinate system relative to the origin of the sensor coordinate system must be known and specified when the RSI object is created.
ST_TRAFO_ROBFRAME	The RSI object ST_TRAFO_ROBFRAME transforms the TOOL data transferred by the RSI object ST_TRAFO_USERFRAME to the BASE coordinate system of the robot controller.
ST_MAP2SEN_PREA	The RSI object ST_MAP2SEN_PREA writes the transformation data to the system variable $SEN_PREA[]. They can be viewed via the variable display.

Program

```
1   DEF Program( )
2   DECL RSIERR err
3   DECL INT hIN1,hIN2,hIN3
4   DECL INT hTrafo1,hTrafo2
5   DECL INT hMap
6   DECL FRAME T
7
8   INI
9
10  err=ST_ANAIN(hIN1,0,1,RSIUNIT_No)
11  err=ST_ANAIN(hIN2,0,2,RSIUNIT_No)
12  err=ST_ANAIN(hIN3,0,3,RSIUNIT_No)
13
14  T={X 80,Y 0,Z -100,A 0,B 0,C 180}
15
16  err=ST_TRAFO_USERFRAME(hTrafo1,0,hIn1,1,hIn2,1,hIn3,1,0,T)
17
18  err=ST_TRAFO_ROBFRAME(hTrafo2,0,hTrafo1,1,hTrafo1,2, _
                          hTrafo1,3,0,#TCP,#BASE)
19
20  err=ST_MAP2SEN_PREA(hMap,0,hIn1,1,1)
21  err=ST_MAP2SEN_PREA(hMap,0,hIn2,1,2)
22  err=ST_MAP2SEN_PREA(hMap,0,hIn3,1,3)
23
24  err=ST_MAP2SEN_PREA(hMap,0,hTrafo1,1,4)
25  err=ST_MAP2SEN_PREA(hMap,0,hTrafo1,2,5)
26  err=ST_MAP2SEN_PREA(hMap,0,hTrafo1,3,6)
27
28  err=ST_MAP2SEN_PREA(hMap,0,hTrafo2,1,7)
29  err=ST_MAP2SEN_PREA(hMap,0,hTrafo2,2,8)
30  err=ST_MAP2SEN_PREA(hMap,0,hTrafo2,3,9)
31
32  err=ST_ON()
33  wait sec 1
34  err=ST_OFF()
35
36  END
```

Line	Description	Parameter
2	Variable for the return values of the RSI commands	-----
3	Variables for the object IDs of ST_ANAIN	-----

Line	Description	Parameter
4	Variables for the object IDs of ST_TRAFO_USERFRAME and ST_TRAFO_ROBFRAME	-----
5	Variable for the object ID of ST_MAP2SEN_PREA	-----
6	Variable for the transformation data of the TOOL coordinate system relative to the sensor coordinate system	-----
10 … 12	Creation of the RSI objects ST_ANAIN	hIN1 … hIN3: object IDs
		0: RSI objects in container 0
		1 … 3: numbers of the analog inputs to be read.
		RSIUNIT_No: No unit
14	Position of the TOOL coordinate system relative to the origin of the sensor coordinate system	-----
16	Creation of the RSI object ST_TRAFO_USERFRAME	hTrafo1: object ID
		0: RSI object in container 0
		hIN1 … hIN3: IDs of the signal source for the signal input
		1: output index of the signal source
		0: transformation type (translation and rotation)
		T: transformation data of the TOOL coordinate system
18	Creation of the RSI object ST_TRAFO_ROBFRAME	hTrafo2: object ID
		0: RSI object in container 0
		hTrafo1: ID of the signal source for the signal input
		1 … 3: output indices of the signal source
		0: transformation type (translation and rotation)
		#TCP: original coordinate system ($TOOL)
		#BASE: target coordinate system ($BASE)
20 … 22	Creation of the RSI objects ST_MAP2SEN_PREA for displaying the position data in the sensor coordinate system	hMap: object ID
		0: RSI object in container 0
		hIN1 … hIN3: IDs of the signal source for the signal input
		1: output index of the signal source
		1 … 3: output indices of the system variable $SEN_PREA[]
24 … 26	Creation of the RSI objects ST_MAP2SEN_PREA for displaying the position data in the TOOL coordinate system	hMap: object ID
		0: RSI object in container 0
		hTrafo1: ID of the signal source for the signal input
		1 … 3: output indices of the signal source
		4 … 6: output indices of the system variable $SEN_PREA[]

Line	Description	Parameter
28 … 30	Creation of the RSI objects ST_MAP2SEN_PREA for displaying the position data in the BASE coordinate system	hMap: object ID
		0: RSI object in container 0
		hTrafo2: ID of the signal source for the signal input
		1 … 3: output indices of the signal source
		7 … 9: output indices of the system variable $SEN_PREA[]
32	Activation of signal processing with the ST_ON() command	-----
34	Deactivation of signal processing with the ST_OFF() command	-----

7.6 Example of a sensor application

Description The programming of a sensor application is described here using the example of force control with a force sensor. A precondition for this example is that the field bus interfacing of the force sensor and tool to the robot system has been carried out correctly.

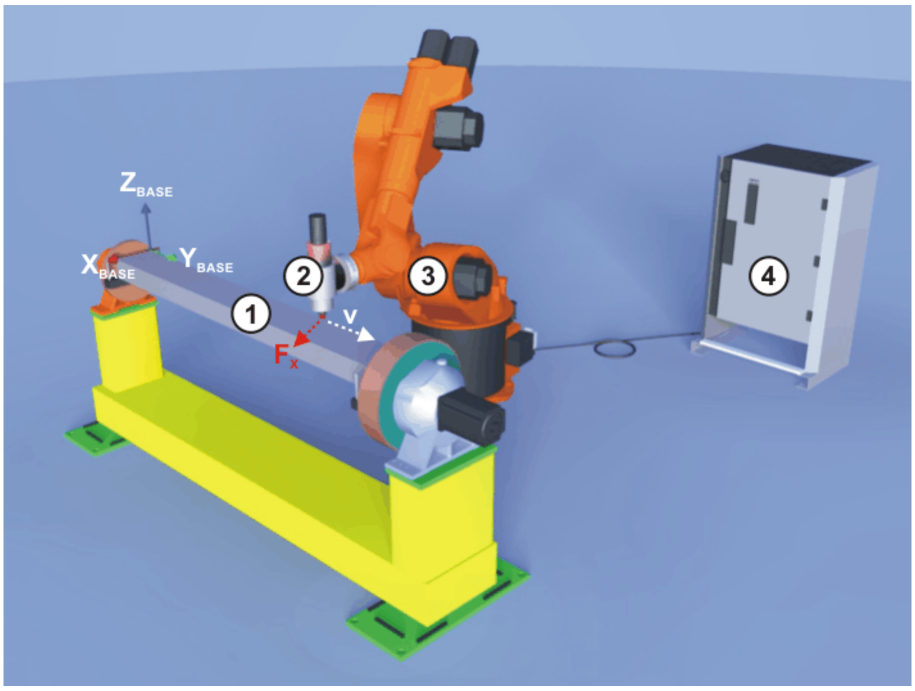


Fig. 7-3: Example of a sensor application

1	Workpiece that is to be deburred along the edge under force control
2	Tool with force sensor
3	Robot
4	Robot controller
F_X	Measured force in the X direction of the BASE coordinate system, perpendicular to the programmed path
v	Direction of motion

A force sensor, which supplies the sensor data for the force control, is mounted between the mounting flange of the robot and the tool. The robot moves to the workpiece and then moves in the Y direction of the BASE coordinate system. At the same time, the robot moves in the X direction of the BASE coordinate system because of the force control. Once contact is made with the workpiece, the robot continues to move in the X direction of the BASE coordinate system under force control and maintains the force setpoint value throughout the motion. The maximum deviation from the programmed path is limited. If the robot exceeds the maximum permissible deviation, an output is set.

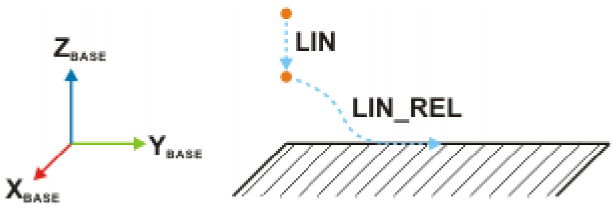


Fig. 7-4: Path of the example sensor application

The following signals are displayed on the RSI monitor:

- Measured value of the force sensor
- Path correction of the robot
- Output for monitoring the maximum permissible deviation from the programmed path

For a real sensor application with force control, the KUKA.ForceTorqueControl technology package is recommended.

Block diagram

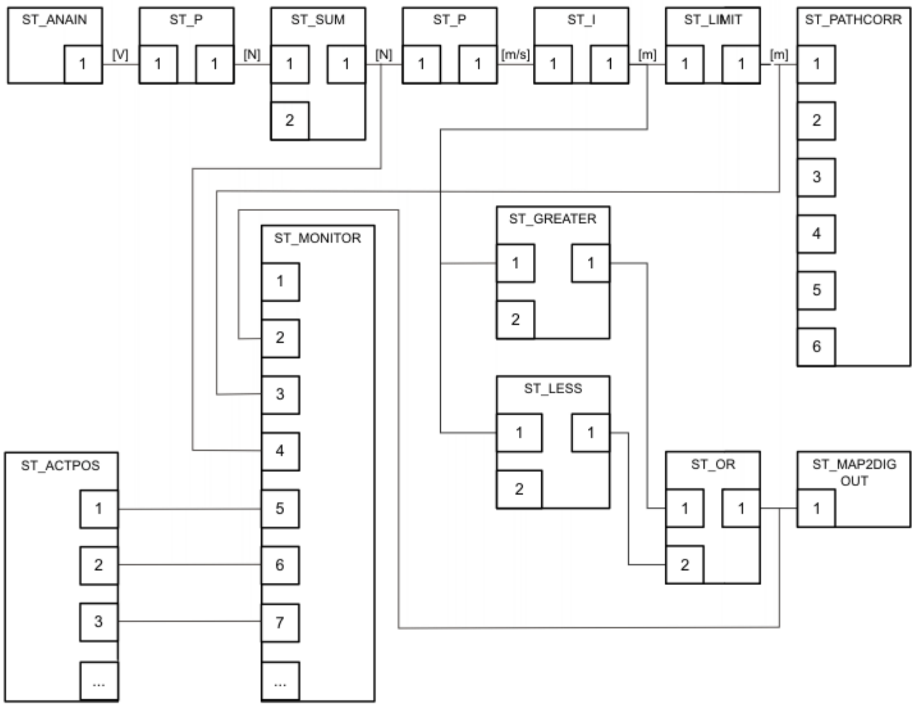


Fig. 7-5: Block diagram of the example sensor application

RSI object	Description
ST_ANAIN	The RSI object ST_ANAIN records the ±10 V analog voltage supplied by the force sensor and assigns the signal the unit [V].
ST_P	The RSI object ST_P converts the recorded analog voltage to a force value in Newtons and changes the unit of the signal to [N].
ST_SUM	The RSI object ST_SUM calculates the control difference from the deviation between the actual force value and the force setpoint.

RSI object	Description
ST_P	The RSI object ST_P calculates the correction velocity on the basis of the control difference. The RSI object ST_P also converts the unit of the signal to [m/s].
ST_I	The RSI object ST_I calculates the correction distance by integrating the correction velocity. The RSI object ST_I also adds the calculated correction of the current cycle to the correction of the previous cycle.
ST_LIMIT	The RSI object ST_LIMIT limits the correction distance with a maximum and minimum value.
ST_PATHCORR	The RSI object ST_PATHCORR superposes the correction distance onto the programmed path. The direction in which the path is corrected depends on which signal input of the RSI object ST_PATHCORR is linked.
ST_GREATER ST_LESS	The RSI objects ST_GREATER and ST_LESS compare the correction distance with a constant. The output is set if the correction distance is greater or less than the constant.
ST_OR	The RSI object ST_OR establishes an OR operation between the signal outputs of the RSI objects ST_GREATER and ST_LESS.
ST_MAP2DIGOUT	The RSI object ST_MAP2DIGOUT sets a digital output.
ST_MONITOR	The RSI object ST_MONITOR records the signals that are to be displayed with the RSI monitor.
ST_ACTPOS	The RSI object ST_ACTPOS reads the position of the current tool (TCP) in the BASE coordinate system.

Program

The example sensor application can be programmed in several different ways. Here, the programming has been implemented as follows:

- The sensor application does not contain subprograms.
- All RSI objects are situated in global container 0.
- The variable for the return values is declared in the KRL program; i.e. the variable cannot be read in the variable display.

```
1  DEF RSI_Example( )
2
3  ; Declaration of KRL Variables
4  DECL RSIERR RET;Return Value of RSI Commands
5  DECL INT HAI;ObjectID for ST_ANAIN
6  DECL INT HP1;ObjectID for ST_P
7  DECL INT HSUM;ObjectID for ST_SUM
8  DECL INT HP2;ObjectID for ST_P
9  DECL INT HI;ObjectID for ST_I
10 DECL INT HLIM;ObjectID for ST_LIMIT
11 DECL INT HPC;ObjectID for ST_PATHCORR
12 DECL INT HMON;ObjectID for ST_MONITOR
13 DECL INT HGR;ObjectID for ST_GREATER
14 DECL INT HLE;ObjectID for ST_LESS
15 DECL INT HOR;ObjectID for ST_OR
16 DECL INT HM2DO;ObjectID for ST_MAP2DIGOUT
17 DECL INT HAP ;ObjectID for ST_ACTPOS
18 DECL CHAR IP[16];IP-Address for RSIMonitor
19
20 INI
21
22 PTP {A1 0, A2 -90, A3 90, A4 0, A5 45, A6 0}
23
24 ; Create RSI Context
25 RET=ST_ANAIN(HAI,0,1,RSIUNIT_V)
26 RET=ST_P(HP1,0,HAI,1,1,RSIUNIT_N)
27 RET=ST_SUM(HSUM,0,HP1,1,0,0,-50)
28 RET=ST_P(HP2,0,HSUM,1,GAIN,'HF01') ;Unit m/s
29 RET=ST_I(HI,0,HP2,1,0.012,1)
30 RET=ST_LIMIT(HLIM,0,HI,1,-20,20)
31 RET=ST_PATHCORR(HPC,0)
32 RET=ST_NEWLINK(HLIM,1,HPC,1)
33 RET=ST_GREATER(HGR,0,HI,1,0,0,20,0)
34 RET=ST_LESS(HLE,0,HI,1,0,0,LOWERLIMIT,0)
35 RET=ST_OR(HOR,0,HGR,1,HLE,1)
36 RET=ST_MAP2DIGOUT(HM2DO,0,HOR,1,1,0)
37 RET=ST_ACTPOS(HAP,0)
38 IP[]="192.0.1.2"
39 RET=ST_MONITOR(HMON,0,IP[],6000,1)
40 RET=ST_SETPARAM(HMON,1,1)
41 RET=ST_NEWLINK(HOR,1,HMON,2)  ;Chart 1 red: Signal-Correction ex-
ceeded limitation
42 RET=ST_NEWLINK(HLIM,1,HMON,3) ;Chart 1 green: Correction
43 RET=ST_NEWLINK(HSUM,1,HMON,4) ;Chart 1 blue: Offset
44 RET=ST_NEWLINK(HAP,1,HMON,5)  ;Chart 2 red: Position in X
45 RET=ST_NEWLINK(HAP,2,HMON,6)  ;Chart 2 green: Position in Y
46 RET=ST_NEWLINK(HAP,3,HMON,7)  ;Chart 2 blue: Position in Z
47
48 PTP {A1 0, A2 -90, A3 90, A4 0, A5 90, A6 0}
49
50 ;Move Base in actual position
51 $BASE.X=$POS_ACT.X
52 $BASE.Y=$POS_ACT.Y
53 $BASE.Z=$POS_ACT.z
54
55 ; Start RSI execution
56 RET=ST_ON()
57 LIN_REL {Y 100}
58 RET=ST_OFF()
59
60 PTP {A1 0, A2 -90, A3 90, A4 0, A5 45, A6 0}
61
62 END
```

Line	Description	Parameter
4	Variable for the return values of the RSI commands	-----
5 ... 17	Variables for the IDs of the RSI objects	-----

Line	Description	Parameter
18	Variable for the IP address of the Windows operating system to enable creation of the RSI monitor	-----
25	Creation of the RSI object ST_ANAIN	HAI: object ID
		0: RSI object in container 0
		1: number of the analog input to be read.
		RSIUNIT_V: unit of the output signal
26	Creation of the RSI object ST_P	HP1: object ID
		0: RSI object in container 0
		HAI: ID of the signal source for the signal input
		1: output index of the signal source
		1: scaling factor for the conversion of voltage to force
		RSIUNIT_N: unit of the output signal
27	Creation of the RSI object ST_SUM	HSUM: object ID
		0: RSI object in container 0
		HP1: ID of the signal source for the signal input
		1: output index of the signal source
		0: second signal input of the RSI object is not assigned
		0: second signal input of the RSI object is not assigned
		-50: constant that is added to the input signal
28	Creation of the RSI object ST_P	HP2: object ID
		0: RSI object in container 0
		HSUM: ID of the signal source for the signal input
		1: output index of the signal source
		0.001: scaling factor for the conversion of force to velocity
		'HF01': coding of the unit m/s according to the unit scheme
29	Creation of the RSI object ST_I	HI: object ID
		0: RSI object in container 0
		HP2: ID of the signal source for the signal input
		1: output index of the signal source
		0.012: integration time in [ms], in which the correction velocity is calculated and added
		1: integration type, to ensure that the integration is only carried out during a CP motion

Line	Description	Parameter
30	Creation of the RSI object ST_LIMIT	HLIM: object ID
		0: RSI object in container 0
		HI: ID of the signal source for the signal input
		1: output index of the signal source
		-20: lower limit of the signal
		20: upper limit of the signal
31	Creation of the RSI object ST_PATHCORR	HPC: object ID
		0: RSI object in container 0
32	Linking of signal input 1 of the RSI object ST_PATHCORR	HLIM: ID of the signal source
		1: output index of the signal source
		HPC: ID of the target object
		3: index of the signal input of the target object
33	Creation of the RSI object ST_GREATER	HGR: object ID
		0: RSI object in container 0
		HI: ID of the signal source for the signal input
		1: output index of the signal source
		0: second signal input of the RSI object is not assigned
		0: second signal input of the RSI object is not assigned
		20: upper limit of the signal
		0: hysteresis deactivated
34	Creation of the RSI object ST_LESS	HLE: object ID
		0: RSI object in container 0
		HI: ID of the signal source for the signal input
		1: output index of the signal source
		0: second signal input of the RSI object is not assigned
		0: second signal input of the RSI object is not assigned
		-20: lower limit of the signal
		0: hysteresis deactivated
35	Creation of the RSI object ST_OR	HOR: object ID
		0: RSI object in container 0
		HGR: ID of the signal source for the signal input 1
		1: output index of the signal source
		HLE: ID of the signal source for the signal input 2
		1: output index of the signal source
36	Creation of the RSI object ST_MAP2DIGOUT	HM2DO: object ID
		0: RSI object in container 0
		HOR: ID of the signal source for the signal input
		1: output index of the signal source
		1: number of the digital output that is set
		0: the RSI object ST_MAP2DIGOUT sends a bit to its signal output

Line	Description	Parameter
37	Creation of the RSI object ST_ACTPOS	HAP: object ID
		0: RSI object in container 0
38	IP address of the Windows operating system	-----
39	Creation of the RSI object ST_MONITOR	HMON: object ID
		0: RSI object in container 0
		IP[]: IP address of the Windows operating system
		6000: port to which the RSI object ST_MONITOR sends the data packets
		1: the signals displayed in the RSI monitor are refreshed every 12 ms
40	Activation of data transfer of the RSI object ST_MONITOR to the RSI monitor	HMON: ID of the RSI object whose object parameters are to be set
		1: number of the object parameter to be set
		1: value to be assigned to the object parameter
41	Linking of signal input 2 of the RSI object ST_MONITOR	HOR: ID of the signal source for the signal input 2
		1: output index of the signal source
		HMON: ID of the target object
		2: index of the signal input of the target object
42	Linking of signal input 3 of the RSI object ST_MONITOR	HLIM: ID of the signal source for signal input 3
		1: output index of the signal source
		HMON: ID of the target object
		3: index of the signal input of the target object
43	Linking of signal input 4 of the RSI object ST_MONITOR	HSUM: ID of the signal source for signal input 4
		1: output index of the signal source
		HMON: ID of the target object
		4: index of the signal input of the target object
44	Linking of signal input 5 of the RSI object ST_MONITOR	HAP: ID of the signal source for the signal input 5
		1: output index of the signal source
		HMON: ID of the target object
		5: index of the signal input of the target object
45	Linking of signal input 6 of the RSI object ST_MONITOR	HAP: ID of the signal source for the signal input 6
		2: output index of the signal source
		HMON: ID of the target object
		6: index of the signal input of the target object

Line	Description	Parameter
46	Linking of signal input 7 of the RSI object ST_MONITOR	HAP: ID of the signal source for the signal input 7
		3: output index of the signal source
		HMON: ID of the target object
		7: index of the signal input of the target object
48	Motion to workpiece	-----
51 … 53	Offset of the BASE to the current position of the tool (TCP)	-----
56	Activation of signal processing with the ST_ON() command	-----
57	Relative LIN motion	-----
58	Deactivation of signal processing with the ST_OFF() command	-----
60	Motion away from the workpiece	-----

7.7 Example application for real-time communication

Overview

KUKA.RobotSensorInterface includes a sample application that can be used to establish real-time communication between a server program and the robot controller. The software is located on the CD-ROM, in the DOC\Example\RealtimeEthernet directory.

The sample application consists of the following components:

Component	Directory
Server program ServerApplication.exe	...\Server_app
KRL program RSIEthernet.src	...\SRC_KRL\PROGRAM
Configuration file RSIEthernet.xml	...\SRC_KRL\INIT
Sample source code C#_ServerApplication.cs	...\SRC_Server

7.7.1 Implementing the example application

Precondition

- CD with technology package
- User group “Expert”

Procedure

1. Copy the contents of the ...\Server_app directory to a Windows system with installed .NET Framework.
2. Copy the KRL program RSIEthernet.src with the data list RSIEthernet.dat into the directory C:\KRC\ROBOTER\KRC\R1\Program of the robot controller.
3. Copy the configuration file RSIEthernet.xml to the directory C:\KRC\ROBOTER\INIT of the robot controller.
4. Start the server program ServerApplication.exe on the external system.
5. In the server program, select the network adapter (NetcardIndex) to be used for communication.
6. Set the IP address of the external system in the configuration file RSIEthernet.xml.

If no external system is available, communication can be carried out via the "Shared Memory" of the robot controller. In the server program, the network adapter (NetcardIndx) must be set in such a way that the network address "192.0.1.2" is displayed.

7.7.2 Server program ServerApplication.exe

The server program ServerApplication.exe makes it possible to test the connection between an external system and the robot controller by establishing permanent communication to the robot controller.

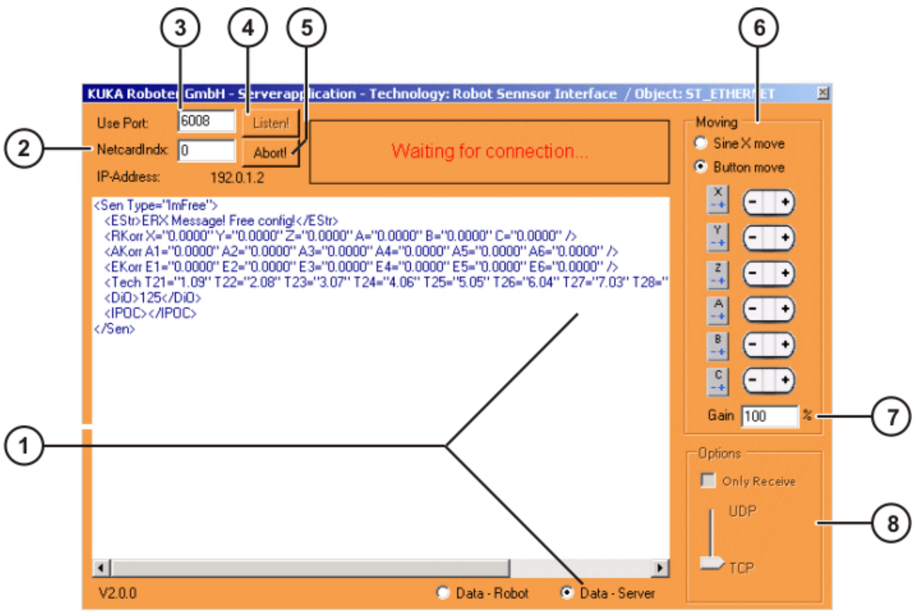
For this purpose, the received data are evaluated and the current interpolation cycle counter (the time stamp of the packet) is copied to the form that is to be sent. Depending on the setting in the server program, the form can be sent with correction data from the **Moving** group or with zero values.

Functions

- Stable communication: transmission and receipt of data in the interpolation cycle (12 ms)
- Motion correction in X: TOOL, BASE or WORLD

The reference system for motion correction is set in the KRL program RSI-Ethernet.src.

- Free Cartesian motion correction using operator control elements
- Display of the data received
- Display of the data sent


3
4
5
6
2
1
7
8
KUKA Roboter GmbH - Serverapplication - Technology: Robot Sennsor Interface / Object: ST_ETHERNET
Listen!
Abort!
Waiting for connection...
Moving
Sine X move
Button move
X
Y
Z
A
B
C
Options
UDP
TCP
V2.0.0

Use Port: 6008
NetcardIndx: 0
IP-Address: 192.0.1.2

```
<Sen Type="ImFree">
  <EStr>ERX Message! Free config!</EStr>
  <RKorr X="0.0000" Y="0.0000" Z="0.0000" A="0.0000" B="0.0000" C="0.0000" />
  <AKorr A1="0.0000" A2="0.0000" A3="0.0000" A4="0.0000" A5="0.0000" A6="0.0000" />
  <EKorr E1="0.0000" E2="0.0000" E3="0.0000" E4="0.0000" E5="0.0000" E6="0.0000" />
  <Tech T21="1.09" T22="2.08" T23="3.07" T24="4.06" T25="5.05" T26="6.04" T27="7.03" T28="
  <DiO>125</DiO>
  <IPOC></IPOC>
</Sen>
```

[✓ Button move] Sine X move
Gain 100 %
[ ] Only Receive
TCP
Data - Robot [✓ Data - Server]

Fig. 7-6: Server program

Item	Description
1	Display box ■ If the option **Data - Robot** is selected, the received data are displayed. ■ If the option **Data - Server** is selected, the sent data are displayed. These data are refreshed at the interpolation cycle rate (12 ms).
2	**NetcardIndx** Input box for the number of the network adapter Default value: 0
3	**Use Port** Input box for 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 as a standard service must be selected. Default value: 6008
4	**Listen!** Button establishes the socket connection: data exchange between the server program and the robot controller is evaluated. The first incoming connection request is connected and used as a communications adapter.
5	**Abort!** Button terminates the communication and resets the server.
6	**Moving** Option for the motion correction ■ **Button move**: free motion in space by an incremental 0.01 mm per interpolation cycle (12 ms) when the “**-/+**” button is pressed with, for example, X' = -/+0.01 mm ■ **SineX move**: correction in the X direction of the set reference system in the sine range with X' = sin(2*PI*0.00133*IPO cycle) **Note**: These correction values apply with **Gain** = 100%.

Item	Description
7	**Gain** Input box for modifying the correction value for motion correction as a percentage Example: **Gain** = 70% ■ **Button move**: increment of 70% * 0.01 mm = 0.007 mm per interpolation cycle ■ **SineX move**: correction in the X direction of sin(2*PI*0.00133*IPO cycle) * 70%
8	Options for data exchange ■ **Only Receive**: If the check box is active, data can now only be received and no longer sent. ■ **TCP**: When the RSI object is created, a connection is established with the external system. ■ **UDP**: When the RSI object is created, no connection is established with the external system. **Note**: The protocol selected here must correspond to the protocol set in the configuration file RSIEthernet.xml.

7.7.3 Configuration file RSIEthernet.xml

The configuration file RSIEthernet.xml contains all the parameters required for configuring the communication between the server program and the robot controller. It also contains the definitions of the object inputs and object outputs of ST_ETHERNET for configuring the XML structures for sending and receiving data.

(>>> 6.2.1 "Structure of the configuration file" page 32)

7.7.4 KRL program RSIEthernet.src

Description The KRL program RSIEthernet.src receives the corrections of the external system and sends them to the robot. Motion of the robot is controlled purely by means of the corrections, i.e. without a programmed path.

Warning!
By activating the motion correction, the external system controls the direction of motion and the orientation of the robot.
The instructions in the safety chapter must be followed and workspaces must be created! Serious injuries and substantial material damage may otherwise result.

The cyclical correction is activated and the reference system for the motion correction is set in the line 'err = ST_ON1(#BASE,1)'.

- #WORLD
- #BASE
- #TOOL

In the case of #BASE and #TOOL, the base or tool most recently used by the robot controller is accepted.

7.7.4.1 Structure of the XML string when sending data (KrcData.xml)

Description The XML string sent to the external system has the following structure:

```
<Rob TYPE="KUKA">
 <RIst X="1620.0008" Y="0.0000" Z="1910.0000" A="0.0000" _
       B="90.0000" C="0.0000"/>
 <RSol X="1620.0000" Y="0.0000" Z="1910.0000" A="0.0000" _
       B="90.0000" C="0.0000"/>
 <AIPos A1="0.0000" A2="-90.0000" A3="90.0000" A4="0.0000" _
        A5="0.0000" A6="0.0000"/>
 <AIPos A1="0.0000" A2="-90.0000" A3="90.0000" A4="0.0000" _
        A5="0.0000" A6="0.0000"/>
 <ASPos A1="0.0000" A2="-90.0000" A3="90.0000" A4="0.0000" _
        A5="0.0000" A6="0.0000"/>
 <EIPos E1="0.0000" E2="0.0000" E3="0.0000" E4="0.0000" _
        E5="0.0000" E6="0.0000"/>
 <ESPos E1="0.0000" E2="0.0000" E3="0.0000" E4="0.0000" _
        E5="0.0000" E6="0.0000"/>
 <MACur A1="0.0000" A2="0.0000" A3="0.0000" A4="0.0000" _
        A5="0.0000" A6="0.0000"/>
 <MECur E1="0.0000" E2="0.0000" E3="0.0000" E4="0.0000" _
        E5="0.0000" E6="0.0000"/>
 <Delay D="0" />
 <Tech C11="0.000000" C12="0.000000" C13="0.000000" C14="0.000000" _
       C15="0.000000" C16="0.000000" C17="0.000000" C18="0.000000" _
       C19="1.000000" C110="0.000000" />
 <DiL>0</DiL>
 <Digout o1="0" o2="0" o3="0" />
 <ST_SOURCE>17.232147</ST_SOURCE>
 <IPOC>4208163634</IPOC>
</Rob>
```

TAG in the XML string	Description	Function
Rob TYPE	Protocol identification, version	-----
RIst	Cartesian actual position	Internal read function of the RSI object ST_ETHERNET
RSol	Cartesian setpoint position	Internal read function of the RSI object ST_ETHERNET
AIPos	Axis-specific actual position of robot axes A1 to A6	Internal read function of the RSI object ST_ETHERNET
ASPos	Axis-specific setpoint position of robot axes A1 to A6	Internal read function of the RSI object ST_ETHERNET
EIPos	Axis-specific actual position of external axes E1 to E6	Internal read function of the RSI object ST_ETHERNET
ESPos	Axis-specific setpoint position of external axes E1 to E6	Internal read function of the RSI object ST_ETHERNET
MACur	Motor currents of robot axes A1 to A6	Internal read function of the RSI object ST_ETHERNET
MECur	Motor currents of external axes E1 to E6	Internal read function of the RSI object ST_ETHERNET
Delay	Number of late packets	Internal read function of the RSI object ST_ETHERNET
Tech	Technology parameter in the advance run of function generator 1	Internal read function of the RSI object ST_ETHERNET
DiL	Input of data type LONG	KRL link to RSI object ST_DIGIN
Digout	3 inputs of data type BOOL	KRL link to RSI object ST_DIGOUT

TAG in the XML string	Description	Function
ST_SOURCE	Input of data type DOUBLE	KRL link to RSI object ST_SOURCE
IPOC	Current time stamp of the data packet	Internal function of the RSI object ST_ETHERNET

The XML format to be sent is generated automatically by the robot controller in accordance with the definition of the object inputs in the configuration file RSIEthernet.xml:

```
<SEND>
 <ELEMENTS>
  <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_RSol" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_AIPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_ASPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_EIPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_ESPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_MACur" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_MECur" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DEF_Delay" TYPE="LONG" INDX="INTERNAL" UNIT="0" />
 <ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
  <ELEMENT TAG="DiL" TYPE="LONG" INDX="1" UNIT="0" />
  <ELEMENT TAG="Digout.o1" TYPE="BOOL" INDX="2" UNIT="0" />
  <ELEMENT TAG="Digout.o2" TYPE="BOOL" INDX="3" UNIT="0" />
  <ELEMENT TAG="Digout.o3" TYPE="BOOL" INDX="4" UNIT="0" />
  <ELEMENT TAG="ST_Source" TYPE="DOUBLE" INDX="5" UNIT="3601" />
 <\ELEMENTS>
<\SEND>
```

7.7.4.2 Structure of the XML string when importing data (ExternalData.xml)

Description

The XML string imported from the external system has the following structure:

```
<Sen Type="ImFree">
 <EStr>ERX Message! Free config!</EStr>
 <RKorr X="0.0000" Y="0.0000" Z="0.0000" A="0.0000" B="0.0000" _
        C="0.0000" />
 <AKorr A1="0.0000" A2="0.0000" A3="0.0000" A4="0.0000" _
        A5="0.0000" A6="0.0000" />
 <EKorr E1="0.0000" E2="0.0000" E3="0.0000" E4="0.0000" _
        E5="0.0000" E6="0.0000" />
 <Tech T21="1.09" T22="2.08" T23="3.07" T24="4.06" T25="5.05" _
       T26="6.04" T27="7.03" T28="8.02" T29="9.01" T210="10.00" />
 <DiO>125</DiO>
 <IPOC></IPOC>
</Sen>
```

Tag in the XML string	Description	Function
Sen Type	Protocol identification	-----
EStr	Generation of a message in the message window If <EStr> ...</EStr>: Message for information If <EStr>Error: ...</EStr>: Acknowledgeable error message; the robot is stopped.	Internal write function of the RSI object ST_ETHERNET. No output exists.

Tag in the XML string	Description	Function
RKorr	Data array of data type DOUBLE	Object output 1 ... 6 of ST_ETHERNET linked to the RSI objects ST_PATHCORR and ST_MAP2SEN_PREA. The values received are saved in $SEN_PREA [1 ... 6].
AKorr	Data of data type DOUBLE	Without link. Object outputs 7 ... 18 are not further processed in the RSI context.
EKorr	Data of data type DOUBLE	Without link. Object outputs 7 ... 18 are not further processed in the RSI context.
Tech	Technology parameters in the main run on function generator 2	Internal write function of the RSI object ST_ETHERNET.
DiO	Data of data type LONG	Object output 19 of ST_ETHERNET. This is linked in the KRL program to the RSI object ST_MAP2SEN_PINT. The data are saved in $SEN_PINT [1].
IPOC	Current time stamp of the data packet	Internal function of the RSI object ST_ETHERNET

The robot controller expects an XML format corresponding to the definition of the object outputs of ST_ETHERNET in the configuration file RSIEthernet.xml:

```
<RECEIVE>
<ELEMENTS>
 <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" UNIT="0" />
 <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" UNIT="1" HOLDON="1" />
 <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" UNIT="1" HOLDON="1" />
 <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" UNIT="1" HOLDON="1" />
 <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" UNIT="0" HOLDON="1" />
 <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" UNIT="0" HOLDON="1" />
 <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" UNIT="0" HOLDON="1" />
 <ELEMENT TAG="AKorr.A1" TYPE="DOUBLE" INDX="7" UNIT="0" HOLDON="0" />
 <ELEMENT TAG="AKorr.A2" TYPE="DOUBLE" INDX="8" UNIT="0" HOLDON="0" />
 <ELEMENT TAG="AKorr.A3" TYPE="DOUBLE" INDX="9" UNIT="0" HOLDON="0" />
 <ELEMENT TAG="AKorr.A4" TYPE="DOUBLE" INDX="10" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="AKorr.A5" TYPE="DOUBLE" INDX="11" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="AKorr.A6" TYPE="DOUBLE" INDX="12" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E1" TYPE="DOUBLE" INDX="13" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E2" TYPE="DOUBLE" INDX="14" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E3" TYPE="DOUBLE" INDX="15" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E4" TYPE="DOUBLE" INDX="16" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E5" TYPE="DOUBLE" INDX="17" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="EKorr.E6" TYPE="DOUBLE" INDX="18" UNIT="0" HOLDON="0"
/>
 <ELEMENT TAG="DEF_Tech.T2" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
 <ELEMENT TAG="DiO" TYPE="LONG" INDX="19" UNIT="0" HOLDON="1" />
<\ELEMENTS>
<\RECEIVE>
```

7.7.5 Sample source code C#_ServerApplication.cs

The sample source code contains the programming of a network connection to the robot controller in the programming language C#. Simple integration is possible, for example, with a console project. For this, the member function “private static void anyfunction()” must be called.

The program generates a second process which communicates with the robot controller, independently of the server program. The basic functionality, the mirroring of the interpolation cycle, is already implemented.

Port 6008 and network card (NetcardIndx) 0 are set by default. The data to be sent are loaded via the XML model class. The file ExternalData.xml from the directory ...\Server_app must be integrated into the project for this purpose.

8 Diagnosis

8.1 Overview of diagnosis

Description

There are 2 options available for diagnosis of an error within the RSI context:

- Displaying signals with the RSI monitor.
  (>>> 8.2 "Overview of signal display" page 65)
- Displaying information about the RSI commands on Telnet and/or in a LOG file.
  (>>> 8.3 "Overview of RSI information output on Telnet and/or in a LOG file" page 71)

Telnet is used for diagnosis of errors occurring during configuration or communication with the real-time operating system VxWorks.

- Display and check IP address.
  (>>> 8.4 "Displaying the IP address" page 73)
- Check communication of the robot controller with the external system.
  (>>> 8.5 "Checking communication via VxWorks" page 73)

8.2 Overview of signal display

Overview

Step	Display on robot controller	Display on external PC
1	Configure the RSI monitor. (>>> 8.2.4 "Configuring RSI monitor" page 67)	Install KUKA.Router on the robot controller. (>>> 8.2.2 "Installing KUKA.Router" page 66)
2	Display signals on the robot controller. (>>> 8.2.5 "Displaying signals on the robot controller" page 69)	Configure KUKA.Router. (>>> 8.2.3 "Configuring KUKA.Router" page 66)
3	-----	Copy the program ...\KRC\TP\RSI\UTIL\RSI-Monitor.exe from the robot controller to the external PC.
4	-----	Configure the RSI monitor. (>>> 8.2.4 "Configuring RSI monitor" page 67)
5	-----	Display signals on external PC. (>>> 8.2.6 "Displaying signals on external PC" page 69)

8.2.1 RSI monitor

Description

The RSI monitor can record and display up to 24 signals from the RSI context. The signals can be recorded directly to the robot controller by the RSI monitor or via a network to an external PC.

If the signals are recorded by the RSI monitor on an external PC via a network, the KUKA.Router program is required.

If the signals are recorded directly on the robot controller, they cannot be viewed until the program has been terminated. If an external PC is used, the signals can be viewed while the program is being executed.

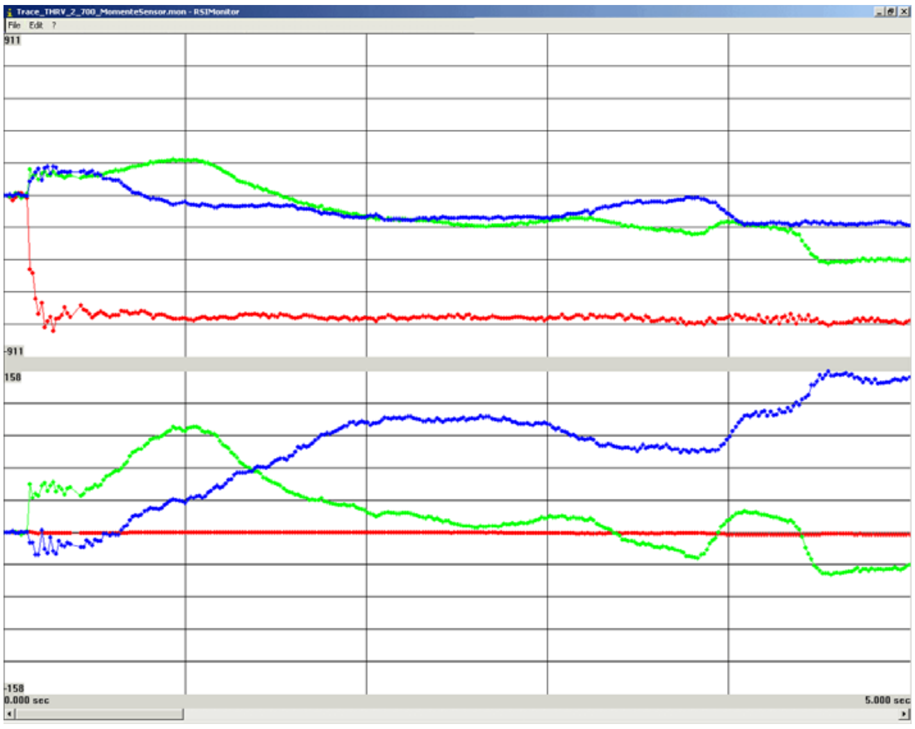


Fig. 8-1: Example of a recording with the RSI monitor

The data for the recording are acquired using the RSI object ST_MONITOR and sent to the RSI monitor. The signals to be displayed can be applied to signal inputs 2 to 25. The recorded signals can be distributed over up to 5 diagrams.

The parameter ID 1 of the RSI object ST_MONITOR activates the data transfer. Signal input 1 of the RSI object ST_MONITOR can be used additionally to control the activated data transfer by means of a Boolean signal.

8.2.2 Installing KUKA.Router

Precondition

- Network connection in Windows between robot controller and external PC

Further information about KUKA.Router can be found in the KUKA_Router directory on the CD-ROM.

Procedure

1. Start the program **Setup.exe** in the KUKA_Router directory on the CD-ROM.
2. Accept the license agreement and press **Next**.

   The Router will then be installed.
3. Reboot the robot controller.

8.2.3 Configuring KUKA.Router

Precondition

- User group “Expert”
- Windows interface (CTRL+ESC)

Procedure

1. On the Windows interface, open the program KUKA.Router by selecting the menu sequence **Start** > **Programs** > **Startup** > **Router**.

2. Add a new route using the menu sequence **File** > **New**.

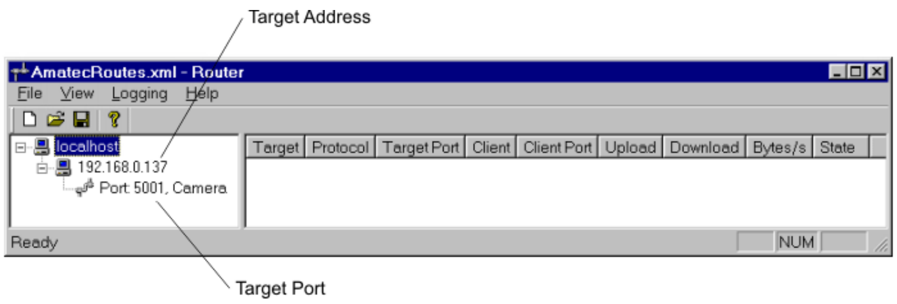

Target Address
AmatecRoutes.xml - Router
File View Logging Help
localhost
192.168.0.137
Port 5001, Camera
Target Protocol Target Port Client Client Port Upload Download Bytes/s State
Ready
NUM
Target Port

Fig. 8-2: KUKA.Router program

3. Using the arrow keys, select the IP address and press the Enter key.
   The **Configure target** window is opened.
4. In the **Target host** box, enter the target address and press **OK**.
5. Using the arrow keys, select the port and press the Enter key.
   The **Configure route** window is opened.
6. Enter the source port in the **Source Port** box.
7. Enter the target port in the **Target Port** box, and press **OK**.
8. Minimize the **KUKA.Router** window.

Do not close the program, as otherwise no data will be sent.

8.2.4 Configuring RSI monitor

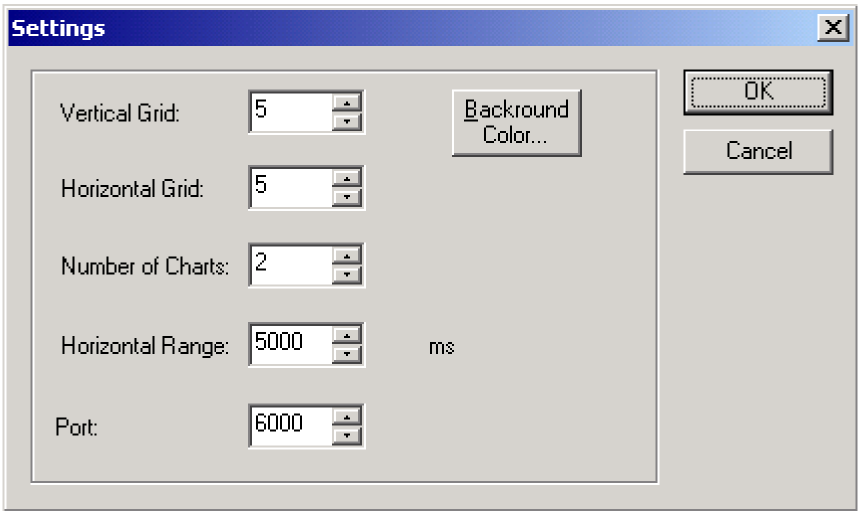
Precondition

- If the RSI monitor is configured on the robot controller: user group “Expert”.

Procedure

1. Select the menu sequence **Monitor** > **RSIMonitor**.
   The RSI monitor opens.
2. In the RSI monitor, select the menu sequence **Edit** > **Settings...**.
3. In the **Settings** window, configure the RSI monitor.
4. Accept the settings by pressing **OK**.
5. To modify the properties of the lines for the signals, select the menu sequence **Edit** > **Line Settings...** in the RSI monitor.
6. Set the line properties in the **Line Properties** window.
7. Accept the settings by pressing **OK**.

Description


Settings

Vertical Grid: 5

Horizontal Grid: 5

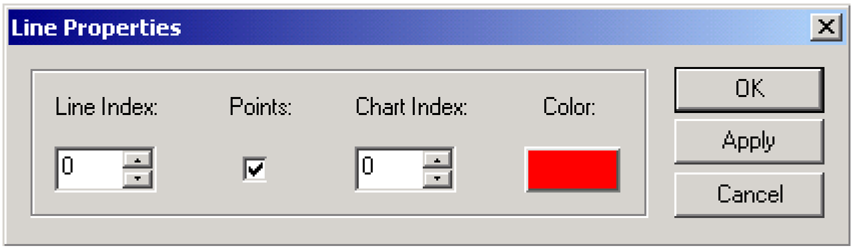
Number of Charts: 2

Horizontal Range: 5000 ms

Port: 6000

Fig. 8-3: “Settings” window

Parameter	Description
Vertical Grid	Number of vertical lines per displayed chart
Horizontal Grid	Number of horizontal lines per displayed chart
Number of Charts	Number of charts displayed
Horizontal Range	Value for the time axis of the charts in [ms]
Port	Number of the port at which the RSI monitor is expecting data packets The port of the RSI monitor must correspond to the target port in KUKA.Router.


Line Properties

Line Index: 0

Points: ✓

Chart Index: 0

Color: 🟥

Fig. 8-4: “Line Properties” window

Parameter	Description
Line Index	Number of the signal diagram ■ **0 … 24** Signal diagram at signal input 2 … 26
Points	Check box not activated: the selected signal diagram is displayed as a line in the RSI monitor. Check box activated: a point is set in the selected signal diagram in the RSI monitor after every measurement cycle.

Parameter	Description
Chart Index	Number of the chart in which the signal diagram is displayed. Automatically changes if the signal diagram number is changed. By default, 3 signal diagrams are displayed per chart. ■ **0 … 8** Signal diagram 0 … 3 … signal diagram 22 … 24
Color	Color of the selected signal diagram. Automatically changes if the signal diagram number is changed.

8.2.5 Displaying signals on the robot controller

Precondition

- User group “Expert”
- Signal processing is programmed.

Procedure

1. Open a program.
2. Create the RSI object ST_MONITOR and apply the required signals to the signal inputs of the RSI object ST_MONITOR.
   (>>> 8.2.7 "Example program for displaying signals" page 70)
3. Save changes in the program by pressing the **Close** softkey.
4. Select the menu sequence **Monitor** > **RSIMonitor**.
   The RSI monitor opens.
5. Select and execute the program.
   The recording of the RSI monitor starts and finishes automatically with the selected program.

If the program is reset, the recording of the RSI monitor is deleted.

6. To save the recording as a MON file, select the menu sequence **File** > **Save as...** in the RSI monitor and specify a target folder and file name.
7. To save the recording as a DAT or TXT file, select the menu sequence **File** > **Export** in the RSI monitor and specify a target folder and file name.
8. To open a recording, select the menu sequence **File** > **Open** and the desired MON file.

Only MON files can be opened in the RSI monitor.

8.2.6 Displaying signals on external PC

Precondition

- User group “Expert”
- Signal processing is programmed.
- Network connection between robot controller and external PC is established.

Procedure

1. Open a program.
2. Create the RSI object ST_MONITOR and apply the required signals to the signal inputs of the RSI object ST_MONITOR.
   (>>> 8.2.7 "Example program for displaying signals" page 70)

3. Save changes in the program by pressing the **Close** softkey.
4. Start the RSIMonitor.exe program on the external PC.
5. Select and execute the program.

   The recording of the RSI monitor starts and finishes automatically with the selected program.

If the program is reset, the recording of the RSI monitor is deleted.

6. To save the recording as a MON file, select the menu sequence **File** > **Save as...** in the RSI monitor and specify a target folder and file name.
7. To save the recording as a DAT or TXT file, select the menu sequence **File** > **Export** in the RSI monitor and specify a target folder and file name.
8. To open a recording, select the menu sequence **File** > **Open** and the desired MON file.

Only MON files can be opened in the RSI monitor.

8.2.7 Example program for displaying signals

Example

```
1  DEF Program( )
2  DECL RSIERR RET
3  DECL INT OBJECT1_ID,OBJECT2_ID,OBJECT3_ID,OBJECT4_ID, _
            OBJECT5_ID,PORT,CYCLE
4  DECL CHAR IP[16]
5
6  INI
7
8  PTP HOME  Vel= 100 % DEFAULT
9
10 RET=ST_DIGOUT(OBJECT1_ID,0,1,0,RSIUNIT_No)
11 RET=ST_DIGOUT(OBJECT2_ID,0,2,0,RSIUNIT_No)
12 RET=ST_OR(OBJECT3_ID,0,OBJECT1_ID,1,OBJECT2_ID,1)
13 RET=ST_MAP2DIGOUT(OBJECT4_ID,0,OBJECT3_ID,1,3,0)
14
15 IP[]="192.0.1.2"
16 PORT=6000
17 CYCLE=1
18
19 RET=ST_MONITOR(OBJECT5_ID,0,IP[],PORT,CYCLE)
20 RET=ST_NEWLINK(OBJECT1_ID,1,OBJECT5_ID,2)
21 RET=ST_SETPARAM(OBJECT5_ID,1,1)
22
23 RET=ST_ON()
24
25 Program2()
26
27 RET=ST_SETPARAM(OBJECT5_ID,1,0)
28
29 RET=ST_OFF()
30
31 PTP HOME  Vel= 100 % DEFAULT
32
33 END
```

Line	Description
3	Variables for the RSI object ST_MONITOR: ■ OBJECT5_ID ■ PORT ■ CYCLE
4	Variable for the IP address of the Windows operating system

Line	Description
15	IP address of the Windows operating system
16	Port to which the RSI object ST_MONITOR sends the data packets
17	Factor for the cycle time in which the RSI object ST_MONITOR sends the data packets (CYCLE=1 corresponds to 12 ms, CYCLE=2 corresponds to 24 ms, etc.)
19	Creation of the RSI object ST_MONITOR
20	Link to signal input 2 of the RSI object ST_MONITOR
21	The ST_SETPARAM command is used to set the object parameter of the RSI object ST_MONITOR and to activate transmission of the data packets.
27	The ST_SETPARAM command is used to set the object parameter of the RSI object ST_MONITOR and to deactivate transmission of the data packets.

8.3 Overview of RSI information output on Telnet and/or in a LOG file

Description

Information about the RSI commands can be output on Telnet and/or in RSI-specific LOG files. The RSI-specific error messages are always output on Telnet and in the LOG file rsiErrors.log.

The following RSI-specific LOG files are located in the folder ...\KRC\ROBOTER\LOG:

LOG file	Description
rsiErrors.log	Contains the RSI-specific error messages, irrespective of the filter settings.
rsiAll.log	Contains information about each RSI command in accordance with the filter settings.

The entries in the LOG files cannot be deleted. When the LOG file reaches a file size of 200 kB, a backup file is created in the folder ...\KRC\ROBOTER\LOG.
Example: backup file of rsiAll.log is renamed rsiAll.bkp.

Overview

Step	Information on Telnet	Information in LOG file	Information on Telnet and in LOG file
1	Set filter for information output. (>>> 8.3.1 "Configuring filters for information output" page 72)	Set filter for information output. (>>> 8.3.1 "Configuring filters for information output" page 72)	Set filter for information output. (>>> 8.3.1 "Configuring filters for information output" page 72)
2	Open Telnet. (>>> 8.3.2 "Opening Telnet" page 73)	Select and execute the program.	Open Telnet. (>>> 8.3.2 "Opening Telnet" page 73)
3	Select and execute the program.	Reset program in order to be able to write information to the LOG file.	Select and execute the program.

Step	Information on Telnet	Information in LOG file	Information on Telnet and in LOG file
4	Reset filter for information output to the default value. (>>> 8.3.1 "Configuring filters for information output" page 72)	Reset filter for information output to the default value. (>>> 8.3.1 "Configuring filters for information output" page 72)	Reset program in order to be able to write information to the LOG file.
5	-----	-----	Reset filter for information output to the default value. (>>> 8.3.1 "Configuring filters for information output" page 72)

It is advisable to reset the filter to the default value when the diagnostic function is no longer required. The performance of the system may otherwise be reduced in program mode.

8.3.1 Configuring filters for information output

Precondition

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

Procedure

1. Open the file ...\KRC\ROBOTER\INIT\amSysObj.ini.
2. In the [ALog] section, assign the desired value to the Filter parameter.

```
[ALog]
MQueueSize=1000; Message queue size
Filter=0x0      ; 0x73F for all types & shell, file1, file2
```

Value for Filter	Description
0x0	Only the RSI-specific error messages are output on Telnet and in the LOG file rsiErrors.log. (Default value)
0x33F	The information about every RSI command is output on Telnet.
0x63F	The information about every RSI command is output in the LOG file rsiAll.log.
0x73F	The information about every RSI command is output on Telnet and in the LOG file rsiAll.log.

3. Close and save file.
4. Select the menu sequence **Configure** > **I/O Driver** > **Reconfigure I/O Driver**.

8.3.2 Opening Telnet

Precondition

- Windows interface (CTRL+ESC)

Procedure

1. Select the menu sequence **Start** > **Run...**.
2. In the **Open** box, enter.
   - Windows 95: **Telnet 192.0.1.1**
   - Windows XP Security Patch 2 or higher: **Telnetk 192.0.1.1**
3. Click on **OK**.

   The Telnet window is opened.

In all Telnet entries: observe upper/lower case!

8.4 Displaying the IP address

Precondition

- KUKA System Software 5.x

Procedure

1. Open Telnet.
2. Enter the command **version**.

   The IP address is displayed in the **Boot line** under **e=...**.

```
-> version
VxWorks (for VxWin RTAcc) version 5.4.2.
Kernel: WIND version 2.5.
Made on Mar  7 2005, 11:13:54.
Boot line:
esmc(0,1)pc:vxworks h=192.0.1.2 b=192.0.1.1 e=160.160.62.118 u=target
pw=vxworks
value = 92 = 0x5c = '\'
```

8.5 Checking communication via VxWorks

Procedure

1. Open Telnet.
2. Enter the IP address of the external system with the command **ping "xxx.xxx.xxx.xxx"**.
   - Connection present:

```
-> ping "192.0.1.2"
PING 192.0.1.2: 56 data bytes
64 bytes from pc (192.0.1.2): icmp_seq=0. time=0. ms
64 bytes from pc (192.0.1.2): icmp_seq=1. time=0. ms
64 bytes from pc (192.0.1.2): icmp_seq=2. time=0. ms
64 bytes from pc (192.0.1.2): icmp_seq=3. time=0. ms
...
```

   To terminate communication, close the Telnet window.
   - No connection:

```
-> ping "123.123.45.2"
PING 123.123.45.2: 56 data bytes
no answer from 123.123.45.2
```

9 Messages

9.1 Error messages

Message	Cause	Remedy
RSI: Error in Function XXX	Error in programming of RSI command XXX	Correct the RSI command in the program.
SEN: ST_PATHCORR – correction out of range XXX	The Cartesian path correction resulting from the signal processing is too great.	Expand the permissible correction range. To do so, adapt the object parameters of ST_PATHCORR with ST_SETPARAM.
SEN: ST_AXISCORR – correction out of range XXX	The axis angle correction resulting from the signal processing is too great.	Expand the permissible correction range. To do so, adapt the object parameters of ST_AXISCORR with ST_SETPARAM.

10 KUKA Service

10.1 Requesting support

Introduction

The KUKA Roboter GmbH documentation offers information on operation and provides assistance with troubleshooting. For further assistance, please contact your local KUKA subsidiary.

Faults leading to production downtime should be reported to the local KUKA subsidiary within one hour of their occurrence.

Information

The following information is required for processing a support request:

- Model and serial number of the robot
- Model and serial number of the controller
- Model and serial number of the linear unit (if applicable)
- Version of the KUKA System Software
- Optional software or modifications
- Archive of the software
- Application used
- Any external axes used
- Description of the problem, duration and frequency of the fault

10.2 KUKA Customer Support

Availability

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

Argentina

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

Australia

Marand Precision Engineering Pty. Ltd. (Agency)
153 Keys Road
Moorabbin
Victoria 31 89
Australia
Tel. +61 3 8552-0600
Fax +61 3 8552-0605
robotics@marand.com.au

Belgium	KUKA Automatisering + Robots N.V. Centrum Zuid 1031 3530 Houthalen Belgium Tel. +32 11 516160 Fax +32 11 526794 info@kuka.be www.kuka.be
Brazil	KUKA Roboter do Brasil Ltda. Avenida Franz Liszt, 80 Parque Novo Mundo Jd. Guançã CEP 02151 900 São Paulo SP Brazil Tel. +55 11 69844900 Fax +55 11 62017883 info@kuka-roboter.com.br
Chile	Robotec S.A. (Agency) Santiago de Chile Chile Tel. +56 2 331-5951 Fax +56 2 331-5952 robotec@robotec.cl www.robotec.cl
China	KUKA Flexible Manufacturing Equipment (Shanghai) Co., Ltd. Shanghai Qingpu Industrial Zone No. 502 Tianying Rd. 201712 Shanghai P.R. China Tel. +86 21 5922-8652 Fax +86 21 5922-8538 Franz.Poeckl@kuka-sha.com.cn www.kuka.cn
Germany	KUKA Roboter GmbH Zugspitzstr. 140 86165 Augsburg Germany Tel. +49 821 797-4000 Fax +49 821 797-1616 info@kuka-roboter.de www.kuka-roboter.de

France	KUKA 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, Private Limited 621 Galleria Towers DLF Phase IV 122 002 Gurgaon Haryana India Tel. +91 124 4148574 info@kuka.in www.kuka.in
Italy	KUKA Roboter Italia S.p.A. Via Pavia 9/a - int.6 10098 Rivoli (TO) Italy Tel. +39 011 959-5013 Fax +39 011 959-5141 kuka@kuka.it www.kuka.it
Japan	KUKA Robotics Japan K.K. Daiba Garden City Building 1F 2-3-5 Daiba, Minato-ku Tokio 135-0091 Japan Tel. +81 3 6380-7311 Fax +81 3 6380-7312 info@kuka.co.jp
Korea	KUKA Robot Automation Korea Co. Ltd. 4 Ba 806 Sihwa Ind. Complex Sung-Gok Dong, Ansan City Kyunggi Do 425-110 Korea Tel. +82 31 496-9937 or -9938 Fax +82 31 496-9939 info@kukakorea.com

Malaysia	KUKA Robot Automation Sdn Bhd South East Asia Regional Office No. 24, Jalan TPP 1/10 Taman Industri Puchong 47100 Puchong Selangor Malaysia Tel. +60 3 8061-0613 or -0614 Fax +60 3 8061-7386 info@kuka.com.my
Mexico	KUKA de Mexico S. de R.L. de C.V. Rio San Joaquin #339, Local 5 Colonia Pensil Sur C.P. 11490 Mexico D.F. Mexico Tel. +52 55 5203-8407 Fax +52 55 5203-8148 info@kuka.com.mx
Norway	KUKA Sveiseanlegg + Roboter Bryggeveien 9 2821 Gjövik Norway Tel. +47 61 133422 Fax +47 61 186200 geir.ulsrud@kuka.no
Austria	KUKA Roboter Austria GmbH Vertriebsbüro Österreich Regensburger Strasse 9/1 4020 Linz Austria Tel. +43 732 784752 Fax +43 732 793880 office@kuka-roboter.at www.kuka-roboter.at
Portugal	KUKA Sistemas de Automatización S.A. Rua do Alto da Guerra n° 50 Armazém 04 2910 011 Setúbal Portugal Tel. +351 265 729780 Fax +351 265 729782 kuka@mail.telepac.pt

Russia	OOO KUKA Robotics Rus Webnaja ul. 8A 107143 Moskau Russia Tel. +7 495 781-31-20 Fax +7 495 781-31-19 kuka-robotics.ru
Sweden	KUKA Svetsanläggningar + Robotar AB A. Odhners gata 15 421 30 Västra Frölunda Sweden Tel. +46 31 7266-200 Fax +46 31 7266-201 info@kuka.se
Switzerland	KUKA Roboter Schweiz AG Riedstr. 7 8953 Dietikon Switzerland Tel. +41 44 74490-90 Fax +41 44 74490-91 info@kuka-roboter.ch www.kuka-roboter.ch
Spain	KUKA Sistemas de Automatización S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 814-2353 Fax +34 93 814-2950 Comercial@kuka-e.com www.kuka-e.com
South Africa	Jendamark Automation LTD (Agency) 76a York Road North End 6000 Port Elizabeth South Africa Tel. +27 41 391 4700 Fax +27 41 373 3869 www.jendamark.co.za

Taiwan	KUKA Robot Automation Taiwan Co. Ltd. 136, Section 2, Huanjung E. Road Jungli City, Taoyuan Taiwan 320 Tel. +886 3 4371902 Fax +886 3 2830023 info@kuka.com.tw www.kuka.com.tw
Thailand	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 atika@ji-net.com www.kuka-roboter.de
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 Corp. 22500 Key Drive Clinton Township 48036 Michigan USA Tel. +1 866 8735852 Fax +1 586 5692087 info@kukarobotics.com www.kukarobotics.com
UK	KUKA Automation + Robotics Hereward Rise Halesowen B62 8AN UK Tel. +44 121 585-0800 Fax +44 121 585-0900 sales@kuka.co.uk

Index