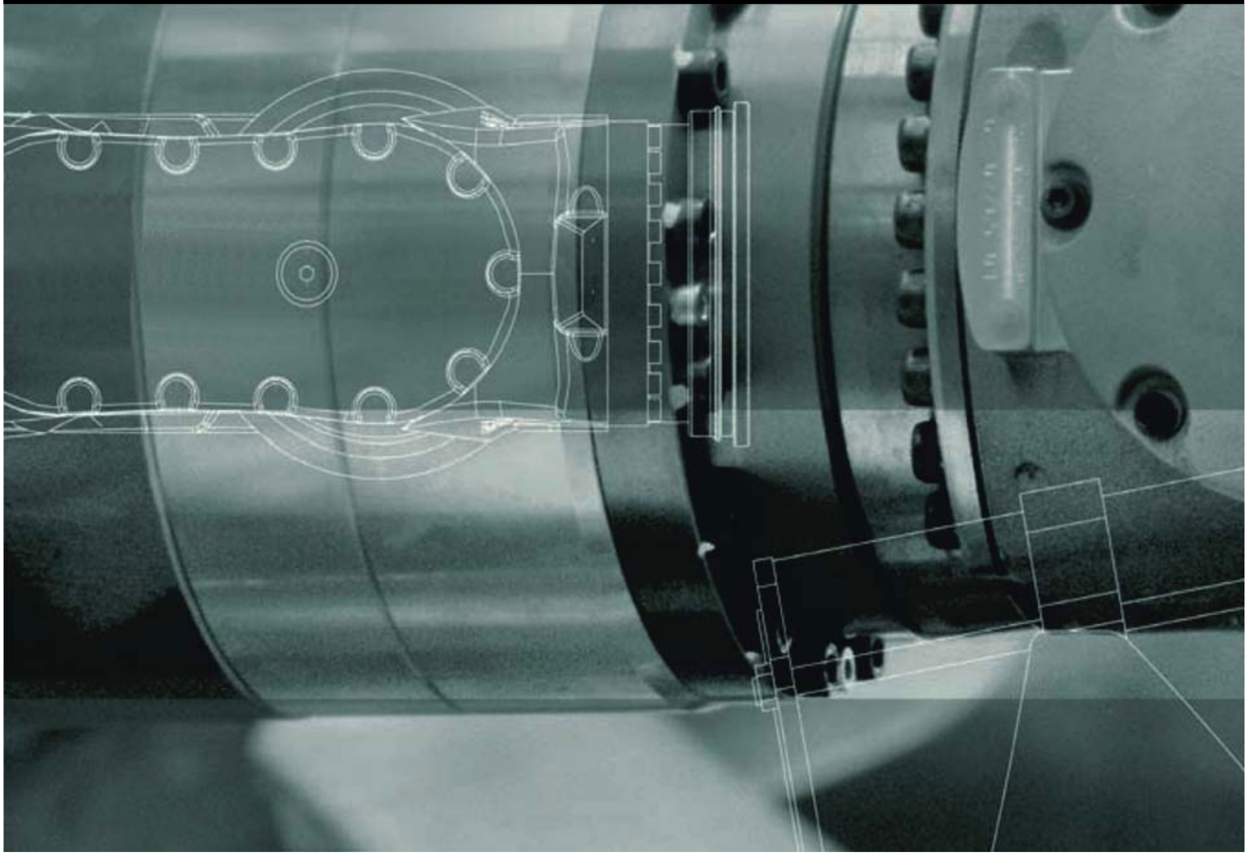
VW System Technology

KUKA Roboter GmbH

KUKA.Ethernet KRL 2.1

For VW System Software 8.2



Issued: 21.03.2012

Version: VST Ethernet KRL 2.1 V3 en

Publication:	Pub VST Ethernet KRL 2.1 en
Bookstructure:	VST Ethernet KRL 2.1 V2.1
Version:	VST Ethernet KRL 2.1 V3 en

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 XML
- Advanced knowledge of networks

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

1.2 Industrial robot documentation

The industrial robot documentation consists of the following parts:

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

Each of these sets of instructions is a separate document.

1.3 Representation of warnings and notes

Safety

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

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

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

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

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

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

Notes

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

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

1.4 Terms used

Term	Description
Data stream	Continuous sequences of data records of which the end cannot be foreseen in advance. The individual data records may be of any fixed type. The amount of data records per unit of time (data rate) may vary. Only sequential access to the data is possible.
EKI	Ethernet KRL interface
EOS	End of stream (end string) String that indicates the end of a data record
Ethernet	Ethernet is a data network technology for local area networks (LANs). It allows data to be exchanged between the connected devices in the form of data frames.
FIFO	Methods used to process a data memory ■ **First In First Out**: the elements saved first are taken first from the memory. ■ **Last In First Out**: the elements saved last are taken first from the memory.
LIFO	
KLI	KUKA Line Interface Line bus for the integration of the system in the customer network
KR C	KUKA Robot Controller KR C is the KUKA robot controller
KRL	KUKA Robot Language KRL is the KUKA robot programming language.
smartHMI	Smart human-machine interface KUKA smartHMI is the user interface of the KUKA system software.
Socket	Software interface that links IP addresses to port numbers.
TCP/IP	Transmission Control Protocol Protocol of the data exchange between devices of a network. TCP constitutes a virtual channel between two sockets in a network connection. Data can be transmitted on this channel in both directions.
UDP/IP	User Datagram Protocol Connectionless protocol of the data exchange between the devices of a network
IP	Internet Protocol The Internet protocol is used to define subnetworks by means of physical MAC addresses.
XML	Extensible Markup Language Standard for creating machine-readable and human-readable documents in the form of a specified tree structure.
XPath	XML Path Language Language used to write and read sections of an XML document

1.5 Trademarks

.NET Framework is a trademark of Microsoft Corporation.

Windows is a trademark of Microsoft Corporation.

2 Product description

2.1 Ethernet KRL overview

Functions

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

- Data exchange via the Ethernet KRL interface
- Receiving XML data from an external system
- Sending XML data to an external system
- Receiving binary data from an external system
- Sending binary data to an external system

Properties

- Robot controller and external system as a client or server
- Configuration of connections via XML-based configuration file
- Configuration of "event messages"
- Monitoring of connections by a ping on the external system
- Reading and writing data from Submit interpreter
- Reading and writing data from robot interpreter

Communication

Data are transmitted via the TCP/IP protocol. It is possible to use the UDP/IP protocol, but not recommended (connectionless network protocol, e.g. no data loss detection).

The communication time depends on the actions programmed in KRL and the data volume sent. Up to 2 ms package circulation time may be reached in KRL, depending on the programming method.

2.2 Configuration of an Ethernet connection

Description

The Ethernet connection is configured via an XML file. A configuration file must be defined for each connection in the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller. The configuration is read in when initializing a connection.

Ethernet connections can be created and operated by the robot interpreter or Submit interpreter. The channels can be used crosswise, e.g. a channel opened in the Submit interpreter can also be operated by the robot interpreter.

The deletion of a connection can be linked to robot interpreter and Submit interpreter actions or system actions.

2.2.1 Behavior in the event of a lost connection

Description

The following properties and functions of the EKI ensure the received data can be processed reliably:

- A connection is automatically closed when reaching the limit of a data memory.
- A connection is automatically closed if a data reception error occurs.
- The data memories continue to be read out with the connection closed.
- If a connection is lost, it can be restored without any influence on the saved data.
- A lost connection can be indicated, for example, by a flag.
- The error message for the error which caused a lost connection can be displayed on the smartHMI.

2.2.2 Monitoring a connection

Description

A connection can be monitored by a ping on the external system (<ALIVE…/> element in the connection configuration).

A flag or output can be set in the event of a successful connection, depending on the configuration. The output or flag is set as long as the ping is regularly sent and the connection to the external system is active. The output or flag is deleted if the connection to the external system is aborted.

2.3 Data exchange

Overview

The robot controller can receive data from an external system as well as send data to an external system via Ethernet KRL.

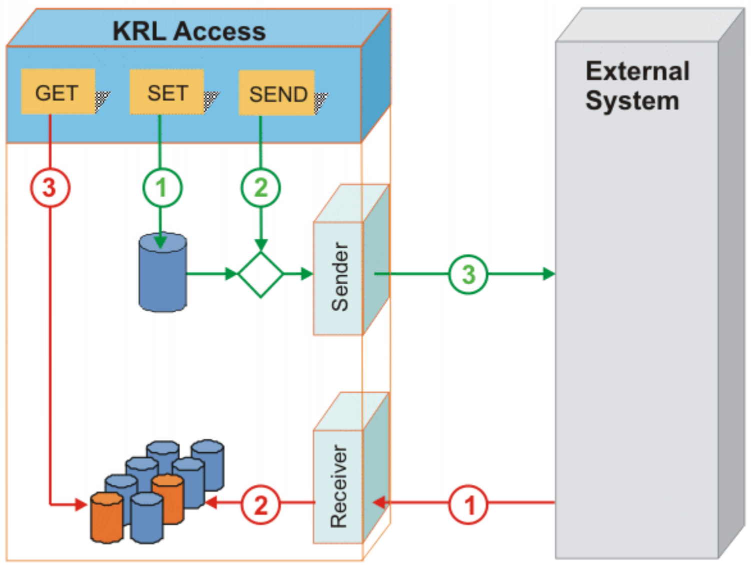


Fig. 2-1: System overview

Data reception

Basic sequence (marked in red) (>>> Fig. 2-1):

1. The external system sends data which are transmitted via a protocol and received by the EKI.
2. The data are stored in a structured manner in a data memory.
3. The data are accessed from a KRL program in a structured manner. KRL instructions are used to read the data and copy them into KRL variables.

Data transmission

Basic sequence (marked in green) (>>> Fig. 2-1):

1. KRL instructions are used to write the data in a data memory in a structured manner.
2. A KRL instruction is used to read the data out of the memory.
3. EKI sends the data to the external system via a protocol.

It is possible to send data directly without first storing the data in a memory.

2.4 Saving data

Description

All data received are automatically saved and, in this way, are available to KRL. XML and binary data are treated differently when saving them.

Each data memory is implemented as a memory stack. The individual memories are read out in FIFO or LIFO mode.

XML data

The received data are extracted and stored type-specifically in different memories (one memory per value).

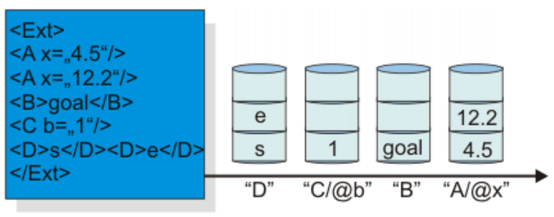


Fig. 2-2: XML data memory

Binary data

The received data are not extracted or interpreted. Only one memory exists for a connection in binary mode.

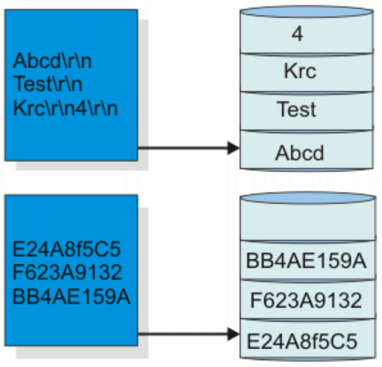


Fig. 2-3: Binary data memory

Read-out methods

Data elements are taken out of the memory in the order in which they were stored there (FIFO). The reverse method, in which the data element stored last in the memory is taken out first, can be configured (LIFO).

Each memory is assigned a common maximum limit for the data which can be saved. If the limit is exceeded, the Ethernet connection is immediately closed to prevent the reception of further data. The data currently received are still saved. The memories can still be further processed. The connection can be re-opened via the EKI_OPEN() Ethernet KRL function.

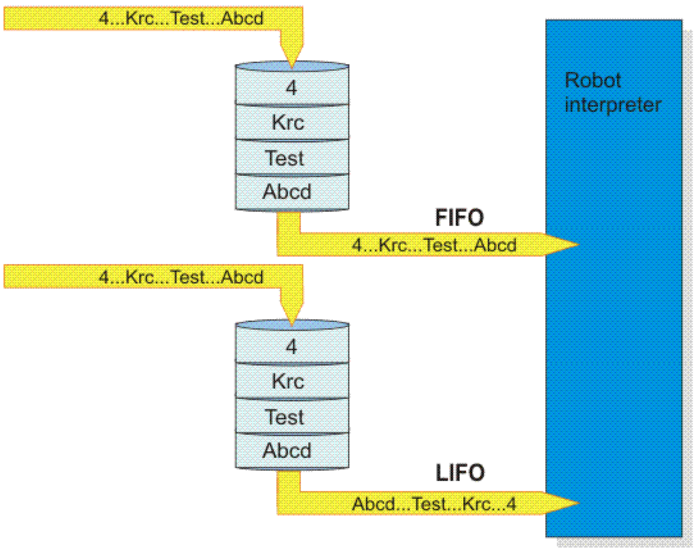


Fig. 2-4: Read-out method overview

2.5 Client-server mode

Description

The robot controller and external system are connected as a client and server. The external system may be the client or server. The number of active connections is limited to 16.

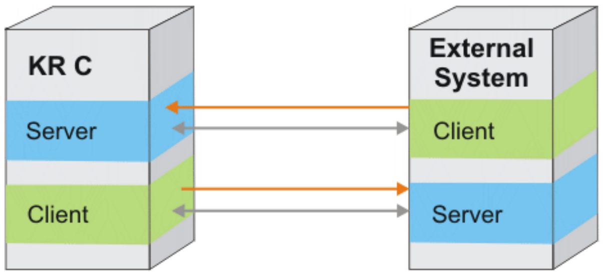


Fig. 2-5: Client-server mode

If the EKI is configured as a server, only an individual client can connect to the server. If several connections are required, several servers should also be created at the interface. It is possible to operate several clients and servers simultaneously within the EKI.

2.6 Protocol types

Description

The transmitted data can be packed in different formats.

The following formats are supported:

- Freely configurable XML structure
- Binary data record of fixed length
- Variable binary data record with end string

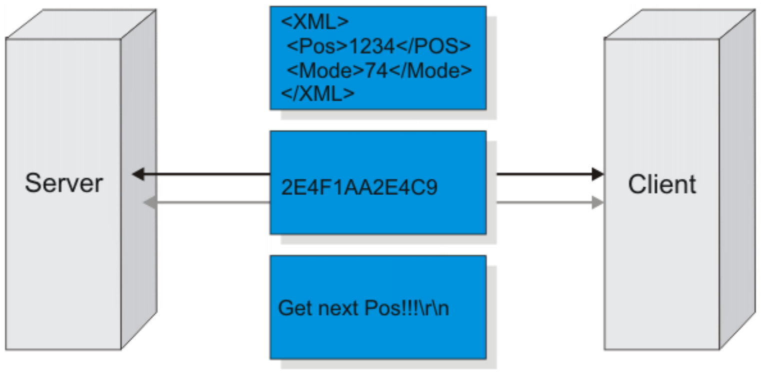


Fig. 2-6: Protocol types

The two binary variants cannot be operated simultaneously at the same connection.

The following combinations are possible:

Connection C*x*	C1	C2	C3	C4	C5
Binary, fixed	**x**	-	**x**	-	-
Binary, variable	-	-	-	**x**	**x**
XML	**x**	**x**	-	-	**x**

Examples

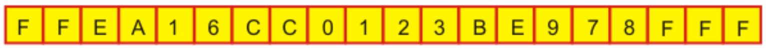


Fig. 2-7: Binary data of fixed length (20 bytes)

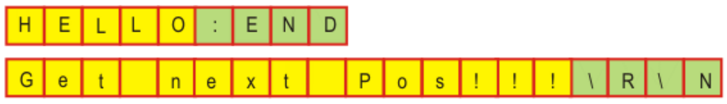


Fig. 2-8: Variable binary data with end string

2.7 Event messages

Description

The following events can be signaled by setting an output or flag:

- Connection is active.
- An individual XML element has arrived at the interface.
- A complete XML structure or complete binary data record has arrived at the interface.

(>>> 6.2.8 "Configuration of event messages" Page 34)

2.8 Error treatment

Description

Ethernet KRL provides functions for data exchange between the robot controller and an external system.

Each of these functions has a return value. The return value can be queried and evaluated in the KRL program.

The return value may contain the following information, depending on the function:

- Error number

- Number of elements still in the memory
- Number of elements which have already been read out of the memory
- Information on whether a connection exists

A message is generated for each error on the smartHMI and in the EKI logbook. The automatic generation of messages can be deactivated.

3 Safety

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

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

⚠ The “Safety” chapter in the operating and programming instructions must be observed. Death to persons, severe physical injuries or considerable damage to property may otherwise result.

4 Installation

4.1 System requirements

Hardware

- VKR C4 robot controller
- External system

Software

- VW System Software 8.2

4.2 Installing or updating Ethernet KRL

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

Precondition

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

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

Procedure

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

LOG file

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

4.3 Uninstalling Ethernet KRL

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

Precondition

- “Expert” user group

Procedure

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

LOG file

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

5 Configuration

5.1 Network connection via the KLI of the robot controller

Description

A network connection must be established via the KLI of the robot controller in order to exchange data via Ethernet.

The Ethernet cable can be connected either to the PROFINET Cu switch or directly to the control PC.

Further information on the PROFINET Cu switch interface can be found in the operating or assembly instructions for the robot controller.

5.2 Configuring a network connection

Precondition

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

Procedure

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

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

The IP address range 192.168.0.x is blocked for the configuration of the network connection.

5. Reboot the robot controller with a cold restart.

6 Programming

6.1 Configuring an Ethernet connection

Overview

An Ethernet connection is configured via an XML file. A configuration file must be defined for each connection in the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller.

The name of the file is also the access key in KRL.

Example: …\EXT.XML —> EKI_INIT("EXT")

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL></EXTERNAL>
    <INTERNAL></INTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <ELEMENTS></ELEMENTS>
  </RECEIVE>
  <SEND>
    <ELEMENTS></ELEMENTS>
  </SEND>
</ETHERNETKRL>
```

Section	Description
<CONFIGURATION> … </CONFIGURATION>	Configuration of the connection parameters between an external system and an interface (>>> 6.1.1 "XML structure for connection properties" Page 21)
<RECEIVE> … </RECEIVE>	Configuration of the reception structure (>>> 6.1.2 "XML structure for data reception" Page 23)
<SEND> … </SEND>	Configuration of the transmission structure (>>> 6.1.3 "XML structure for data transmission" Page 25)

6.1.1 XML structure for connection properties

Description

The settings for the external system are defined in the section <EXTERNAL> … </EXTERNAL>:

Element	Description
TYPE	Defines whether the external system is to communicate as a server or client with the interface (optional) ■ **Server**: external system is a server. ■ **Client**: external system is a client. Default value: **server**
IP	IP address of the external system (optional if TYPE = client)
PORT	Port number of the external system (optional if TYPE = client) ■ **1 … 65,534**

The settings for the interface are defined in the section <INTERNAL> … </INTERNAL>:

Element	Attribute	Description
ENVIRONMENT	———	Link the deletion of the connection to actions (optional) ■ **Program**: deletion after actions of the robot interpreter ■ Reset program. ■ Deselect program. ■ **System**: deletion after system actions ■ Reconfigure I/Os. ■ Reboot robot controller with cold restart. ■ **Submit**: deletion after actions of the Submit interpreter ■ Cancel submit interpreter. Default value: **Program**
BUFFERING	Mode	Method used to process all data memories (optional) ■ **FIFO**: First In First Out ■ **LIFO**: Last In First Out Default value: **FIFO**
	Limit	Maximum number of data elements which can be stored in a data memory (optional) ■ **1 … 512** Default value: **16**
BUFFSIZE	Limit	Maximum number of bytes which can be received without being interpreted (optional) ■ **1 … 65,534 bytes** Default value: **16,384 bytes**
TIMEOUT	Connect	Time until the attempt to establish a connection is aborted (optional) Unit: ms ■ **0 … 65,534 ms** Default value: **2,000 ms**
ALIVE	Set_Out Set_Flag	Sets an output or a flag for a successful connection (optional) Number of the output: ■ **1 … 4 096** Number of the flag: ■ **1 … 1 025** The output or flag is set as long as a connection to the external system is active. The output or flag is deleted if the connection to the external system is aborted.
	Ping	Interval for sending a ping in order to monitor the connection to the external system (optional) ■ **1 … 65,534 s**

Element	Attribute	Description
IP	———	IP address of the interface (optional if TYPE = server) The IP address must be entered here if TYPE = client.
PORT	———	Port number of the interface (optional if TYPE = server) ■ **49 152 … 65 534** The port number must be entered here if TYPE = client.
PROTOCOL	———	Transmission protocol (optional) ■ **TCP** ■ **UPD** Default value: **TCP** It is recommended to always use the TCP/IP protocol.

Example

```
<CONFIGURATION>
   <EXTERNAL>
     <IP>172.1.10.5</IP>
     <PORT>49152</PORT>
     <TYPE>Server</TYPE>
   </EXTERNAL>
   <INTERNAL>
     <ENVIRONMENT>Program</ENVIRONMENT>
     <BUFFERING Mode="FIFO" Limit="10"/>
     <BUFFSIZE Limit="16384"/>
     <TIMEOUT Connect="60000"/>
     <ALIVE Set_Out="666" Ping="200"/>
     <IP>192.1.10.20</IP>
     <PORT>49152</PORT>
     <PROTOCOL>TCP</PROTOCOL>
   </INTERNAL>
</CONFIGURATION>
```

6.1.2 XML structure for data reception

Description

The configuration depends on whether XML data or binary data are received.

- An XML structure has to be defined for the reception of XML data: <XML> … </XML>
- Raw data have to be defined for the reception of binary data: <RAW> … </RAW>

Attributes in the elements of the XML structure <XML> … </XML>:

Element	Attribute	Description
ELEMENT	Tag	Name of the element The XML structure for data reception is defined here (XPath).
ELEMENT	Type	Data type of the element ▪ **STRING** ▪ **REAL** ▪ **INT** ▪ **BOOL** ▪ **FRAME** **Note**: Optional if the tag is used only for event messages. In this case no memory capacity is reserved for the element. **Event flag example**: <ELEMENT Tag="Ext" Set_Flag="56"/>
ELEMENT	Set_Out Set_Flag	Sets an output or flag after receiving the element (optional) Number of the output: ▪ **1 … 4,096** Number of the flag: ▪ **1 … 1,025**
ELEMENT	Mode	Method used to process a data record in the data memory ▪ **FIFO**: First In First Out ▪ **LIFO**: Last In First Out Only relevant if individual data records are to be treated differently than configured under BUFFERING for the interface.

Attributes for the element in the raw data <RAW> … </RAW>:

Element	Attribute	Description
ELEMENT	Tag	Name of the element
ELEMENT	Type	Data type of the element ▪ **BYTE**: Binary data record of fixed length ▪ **STREAM**: Variable binary data record with end string
ELEMENT	Set_Out Set_Flag	Sets an output or flag after receiving the element (optional) Number of the output: ▪ **1 … 4 096** Number of the flag: ▪ **1 … 1 025**

Element	Attribute	Description
ELEMENT	EOS	End string of an elementary piece of information (only relevant if TYPE = STREAM) ■ ASCII encoding: **1 … 32 characters** ■ Alternative end is separated by means of the "|" character. Examples: ■ <ELEMENT … EOS="123,134,21"/> ■ <ELEMENT … EOS="123,134,21|13,10"/>
ELEMENT	Size	Fixed size of information if TYPE = BYTE ■ **1 … 3,600 bytes** Maximum size of information if TYPE = STREAM ■ **1 … 3,600 bytes**

Examples

```
<RECEIVE>
  <XML>
    <ELEMENT Tag="Ext/Str" Type="STRING"/>
    <ELEMENT Tag="Ext/Pos/XPos" Type="REAL" Mode="LIFO"/>
    <ELEMENT Tag="Ext/Pos/YPos" Type="REAL"/>
    <ELEMENT Tag="Ext/Pos/ZPos" Type="REAL"/>
    <ELEMENT Tag="Ext/Temp/Cpu" Type="REAL" Set_Out="1"/>
    <ELEMENT Tag="Ext/Temp/Fan" Type="REAL" Set_Flag="14"/>
    <ELEMENT Tag="Ext/Integer/AState" Type="INT"/>
    <ELEMENT Tag="Ext/Integer/BState" Type="INT"/>
    <ELEMENT Tag="Ext/Boolean/CState" Type="BOOL"/>
    <ELEMENT Tag="Ext/Frames/Frame1" Type="FRAME"/>
    <ELEMENT Tag="Ext/Attributes/@A1" Type="STRING"/>
    <ELEMENT Tag="Ext/Attributes/@A2" Type="INT"/>
    <ELEMENT Tag="Ext" Set_Flag="56"/>
  </XML>
</RECEIVE>
```

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="RawData" Type="BYTE" Size="1408"
                  Set_Flag="14"/>
  </RAW>
</RECEIVE>
```

```
<RECEIVE>
  <RAW>
    <ELEMENT Tag="MyStream" Type="STREAM" EOS="123,134,21"
                  Size="836" Set_Flag="14"/>
  </RAW>
</RECEIVE>
```

6.1.3 XML structure for data transmission

Description

The configuration depends on whether XML data or binary data are sent.

- An XML structure has to be defined for the transmission of XML data: <XML> … </XML>
- The transmission of binary data is implemented directly in the KRL programming. No configuration has to be specified.

Attribute in the elements of the XML structure <XML> … </XML>:

Attribute	Description
Tag	Name of the element The XML structure for data transmission is defined here (XPath).

Example

```
<SEND>
  <XML>
    <ELEMENT Tag="Robot/Data/ActPos/@X"/>
    <ELEMENT Tag="Robot/Data/ActPos/@Y"/>
    <ELEMENT Tag="Robot/Data/ActPos/@Z"/>
    <ELEMENT Tag="Robot/Data/ActPos/@A"/>
    <ELEMENT Tag="Robot/Data/ActPos/@B"/>
    <ELEMENT Tag="Robot/Data/ActPos/@C"/>
    <ELEMENT Tag="Robot/Status"/>
    <ELEMENT Tag="Robot/Mode"/>
    <ELEMENT Tag="Robot/Complex/Tickcount"/>
    <ELEMENT Tag="Robot/RobotType/Robot/Type"/>
  </XML>
</SEND>
```

6.1.4 Configuration according to the XPath schema

Description

If XML is used to exchange data, it is necessary for the exchanged XML documents to be structured in the same way. Ethernet KRL uses the XPath schema to write and read the XML documents.

The following cases are to be distinguished for XPath:

- Writing and reading elements
- Writing and reading attributes

Element notation

- Saved XML document for data transmission:

```
<Robot>
  <Mode>...</Mode>
  <RobotLamp>
    <GrenLamp>
      <LightOn>...</LightOn>
    </GrenLamp>
  </RobotLamp>
</Robot>
```

- Configured XML structure for data transmission:

```
<SEND>
  <XML>
    <ELEMENT Tag="Robot/Mode" />
    <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
  </XML>
<SEND />
```

Attribute notation

- Saved XML document for data transmission:

```
<Robot>
  <Data>
    <ActPos X="...">
    </ActPos>
    <LastPos A="..." B="..." C="..." X="..." Y="..." Z="...">
    </LastPos>
  </Data>
</Robot>
```

- Configured XML structure for data transmission:

```
<SEND>
  <XML>
    <ELEMENT Tag="Robot/Data/LastPos/@X" />
    <ELEMENT Tag="Robot/Data/LastPos/@Y" />
    ...
    <ELEMENT Tag="Robot/Data/ActPos/@X" />
  </XML>
<SEND />
```

6.2 Ethernet KRL functions for data exchange

Overview

Ethernet KRL provides functions for data exchange between the robot controller and an external system.

Exact descriptions of the functions can be found in the appendix.
(>>> 9.5 "Ethernet KRL functions command reference" Page 59)

Initialization and connection
EKI_STATUS = EKI_Init(CHAR[])
EKI_STATUS = EKI_Open(CHAR[])
EKI_STATUS = EKI_Close(CHAR[])
EKI_STATUS = EKI_Clear(CHAR[])

Sending
EKI_STATUS = EKI_Send(CHAR[], CHAR[])

Writing
EKI_STATUS = EKI_SetReal(CHAR[], CHAR[], REAL)
EKI_STATUS = EKI_SetInt(CHAR[], CHAR[], INTEGER)
EKI_STATUS = EKI_SetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_SetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_SetString(CHAR[], CHAR[], CHAR[])

Data access
EKI_STATUS = EKI_GetBool(CHAR[], CHAR[], BOOL)
EKI_STATUS = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])
EKI_STATUS = EKI_GetInt(CHAR[], CHAR[], Int)
EKI_STATUS = EKI_GetIntArray(CHAR[], CHAR[], Int[])
EKI_STATUS = EKI_GetReal(CHAR[], CHAR[], Real)
EKI_STATUS = EKI_GetRealArray(CHAR[], CHAR[], Real[])
EKI_STATUS = EKI_GetString(CHAR[], CHAR[], CHAR[])
EKI_STATUS = EKI_GetFrame(CHAR[], CHAR[], FRAME)
EKI_STATUS = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])

Error treatment
EKI_CHECK(EKI_STATUS, EKrlMsgType, CHAR[])

Other
EKI_STATUS = EKI_ClearBuffer(CHAR[], CHAR[])
EKI_STATUS = EKI_Lock(CHAR[])
EKI_STATUS = EKI_Unlock(CHAR[])

6.2.1 Programming tips

It is advisable to program the data exchange via the Ethernet KRL interface exclusively in the VW_USER module and not to use the Ethernet KRL functions in programs (Folgen).

- The following points should be observed if a connection is created in the Submit interpreter:
  - The <ENVIRONMENT…> element must be used in the connection configuration to specify that the channel concerned is a Submit channel.
  - An open channel in the Submit interpreter can also be addressed by the robot interpreter.
  - If the Submit interpreter is deselected, the connection is automatically deleted by means of the configuration.
- EKI instructions are executed in advance. If an EKI instruction is to be executed in the main run, instructions must be used which trigger an advance run stop, e.g. WAIT SEC.
- Since each access to the interface consumes time, it is recommended to call up large amounts of data with the field access functions EKI_Get…Array().
- Ethernet KRL can access a maximum of 512 array elements. It is possible to create a larger array in KRL, e.g. myFrame[1000], but only a maximum of 512 elements can be read.

6.2.2 Initializing and deleting a connection

Description

A connection must be initialized with the EKI_Init() function. The connection configuration specified in the function is read in.

The deletion of a connection can be linked to robot interpreter and Submit interpreter actions or system actions via the <ENVIRONMENT…> element in the connection configuration.

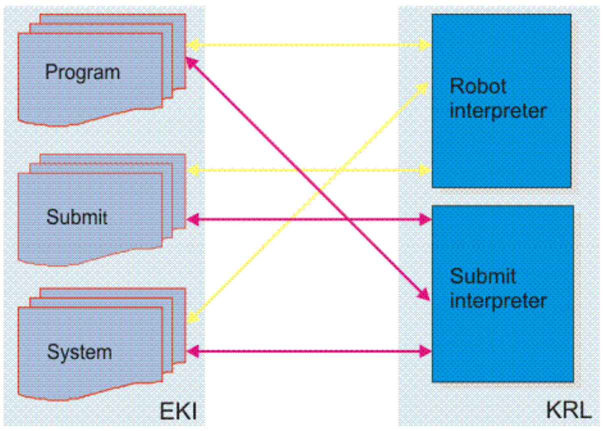


Fig. 6-1: Connection configuration .

Depending on the connection configuration, a connection is deleted after the following actions:

- **"Program"** configuration
  - Reset program.
  - Deselect program.
- **"Submit"** configuration
  - Deselect Submit interpreter.

- **"System"** configuration
  - Reboot robot controller with cold restart.
  - Reconfigure I/Os.

The driver is reloaded when reconfiguring the I/Os, i.e. all initializations are deleted.

6.2.3 Opening and closing a connection

Description

The connection to the external system is established by means of a KRL program. Most KRL programs are structured as follows:

```
1   DEF Connection()
    ...
2   RET=EKI_Init("Connection")
3   RET=EKI_Open("Connection")
    ...
4   Write data, send data or get received data
    ...
5   RET=EKI_Close("Connection")
6   RET=EKI_Clear("Connection")
    ...
7   END
```

Line	Description
2	EKI_Init() initializes the channel used by the interface to connect to the external system.
3	EKI_Open() opens the channel.
4	KRL instructions used to write data in the memory, send data or access received data
5	EKI_Close() closes the channel.
6	EKI_Clear () deletes the channel.

It should be taken into account during programming whether the interface is configured as a server or client.

Server mode

EKI_Open() sets the server to a listening state if the interface is configured as a server. The server awaits the connection request of a client without interruption of the program run. If the <TIMEOUT Connect="…"/> element is not assigned data in the configuration file, the server waits until a client requests a connection.

A connection request by a client is indicated by access to the interface or by an event message, e.g. via the <ALIVE SET_OUT="…"/> element.

An event flag or output has to be programmed, e.g. WAIT FOR $OUT[…], if the program run is to be interrupted as long as the server waits for the connection request.

It is recommended not to use EKI_Close() in server mode. In server mode, the channel is closed from the external client.

Client mode

EKI_Open() interrupts the program run until the connection to the external system is active if the interface is configured as a client. EKI_Close() closes the connection to the external server.

6.2.4 Sending data

Description

Depending on the configuration and programming, the following data can be sent with EKI_Send():

- Complete XML structure
- Partial XML structure
- XML data directly as string
- Binary data record with end string (EOS) directly as string
- Binary data record of fixed length directly as string

  Binary data records of fixed length must be read into the KRL program with CAST_TO(). Only data of REAL type (4 bytes) are legible, not Double.

Detailed information on the CAST_TO() command can be found in the CREAD/CWRITE documentation.

XML data example

Sending the complete XML structure

- Saved XML structure for data transmission:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1","Robot")
```

- Sent XML structure:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

Sending part of the XML structure

- Saved XML structure for data transmission:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1","Robot/ActPos")
```

- Sent XML structure:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
</Robot>
```

Direct transmission of XML data as a string

- Saved XML structure for data transmission:

```
<Robot>
  <ActPos X="1000.12"></ActPos>
  <Status>12345678</Status>
</Robot>
```

- Programming:

```
DECL EKI_STATUS RET
RET=EKI_Send("Channel_1","<POS><XPOS>1</XPOS></POS>")
```

- Sent string:

```
<POS><XPOS>1</XPOS></POS>
```

Binary data example

Direct sending of a binary data record of fixed length (10 bytes)

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
</RAW>
```

- Programming:

```
DECL EKI_STATUS RET
CHAR Bytes[10]
OFFSET=0
CAST_TO(Bytes[],OFFSET,91984754,913434.2,TRUE,"X")
RET=EKI_Send("Channel_1",Bytes[])
```

- Sent data:

```
"r?{ ? _I X"
```

Direct sending of a binary data record with end string

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="STREAM" EOS="65,66" />
</RAW>
```

- Programming:

```
DECL EKI_STATUS RET
CHAR Bytes[64]
Bytes[]="Stream ends with:"
RET=EKI_Send("Channel_1",Bytes[])
```

- Sent data:

```
"Stream ends with:AB"
```

6.2.5 Reading out data

In order to read out data, the corresponding KRL variables have to be initialized, e.g. by the assignment of values.

Description

XML and binary data are treated differently when saved and read out:

- XML data are extracted by the EKI and stored type-specifically in different memories. It is possible to access each saved value individually.

  All EKI_Get…() access functions can be used to read out XML data.
- Binary data records are not interpreted by the EKI and stored together in a memory.

  The EKI_GetString() access function must be used to read a binary data record out of a memory. Binary data records are read out of the memory as strings.

  Binary data records of fixed length must be divided into individual variables again in the KRL program with CAST_FROM(). Only data of REAL type (4 bytes) are legible, not Double.

Detailed information on the CAST_FROM() command can be found in the CREAD/CWRITE documentation.

XML data example

Saved XML structure for data reception:

```
<Sensor>
  <Message>Example message</Message>
  <Status>
    <IsActive>1</IsActive>
  </Status>
</Sensor>
```

Programming:

```
; Declaration
INT i
DECL EKI_STATUS RET
CHAR valueChar[256]
BOOL valueBOOL
; Initialization
FOR i=(1) TO (256)
 valueChar[i]=0
ENDFOR
valueBOOL=FALSE

RET=EKI_GetString("Channel_1","Sensor/Message",valueChar[])
RET=EKI_GetBool("Channel_1","Sensor/Status/IsActive",valueBOOL)
```

Binary data example

Reading out a binary data record of fixed length (10 bytes)

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="BYTE" Size="10" />
</RAW>
```

- Programming:

```
; Declaration
INT i
INT OFFSET
DECL EKI_STATUS RET
CHAR Bytes[10]
INT valueInt
REAL valueReal
BOOL valueBool
CHAR valueChar[1]
; Initialization
FOR i=(1) TO (10)
 Bytes[i]=0
ENDFOR
OFFSET=0
valueInt=0
valueBool=FALSE
valueReal=0
valueChar[1]=0
RET=EKI_GetString("Channel_1","Buffer",Bytes[])
OFFSET=0
CAST_FROM(Bytes[],OFFSET,valueReal,valueInt,valueChar[],valueBool)
```

Reading out a binary data record with end string

- Configured raw data:

```
<RAW>
  <ELEMENT Tag="Buffer" Type="STREAM" EOS="13,10" />
</RAW>
```

- Programming:

```
; Declaration
INT i
DECL EKI_STATUS RET
CHAR Bytes[64]
; Initialization
FOR i=(1) TO (64)
 Bytes[i]=0
ENDFOR
RET=EKI_GetString("Channel_1","Buffer",Bytes[])
```

6.2.6 Deleting received data

Description

A distinction is to be made between the following cases when deleting received data:

- Deletion with EKI_Clear(): the Ethernet connection is terminated and all memories used by the connection are deleted.
- Deletion with EKI_ClearBuffer(): data received but not yet called up are deleted from one or all of the memories.

XML data are extracted by the EKI and stored type-specifically in different memories. When deleting individual memories, it must be ensured that no data that belong together are lost.

Example

The position of the memory to be deleted is specified in XPATH.

```
EKI_STATUS RET
RET = EKI_ClearBuffer("Channel_1","Root/Activ/Flag")
```

All memories of the element <Root>…</Root> are deleted.

```
EKI_STATUS RET
RET = EKI_ClearBuffer("Channel_1","Root")
```

6.2.7 Return value of the Ethernet KRL functions

EKI_STATUS is a global variable which contains the current status of the interface with regard to an Ethernet KRL function.

Syntax

`GLOBAL STRUC EKI_STATUS INT` *Buff*`,`*Read*`,`*Msg_No*`, BOOL` *Connected*

Explanation of the syntax

Element	Description
Buff	Number of elements in the memory
Read	Number of elements read out of the memory
Msg_No	Error number of the error that occurred during data reception or function call. With EKI_CHECK() a message relating to the error can be displayed on the smartHMI.
Connected	Information on whether a connection exists - TRUE = a connection exists. - FALSE = no connection exists.

Which elements of the structure are assigned data depends on the Ethernet KRL function:

EKI_	Buff	Read	Msg_No	Connected
Get…()	x	x	x	x
Set…()	-	-	x	x

EKI_	Buff	Read	Msg_No	Connected
Send()	-	-	x	x
Init()	-	-	x	-
Other	-	-	x	x

Example

```
EKI_STATUS RET
...
RET=EKI_Open("Channel_1")
EKI_CHECK(RET,#QUIT)
```

The return value of the EKI_Open() function is evaluated. If the channel cannot be opened and no data connection can be established, an error value is read out and an acknowledgement message displayed.

6.2.8 Configuration of event messages

Description

The following events can be signaled by setting an output or flag:

- Connection is active.
- An individual XML element has arrived at the interface.
- A complete XML structure or complete binary data record has arrived at the interface.

Event output

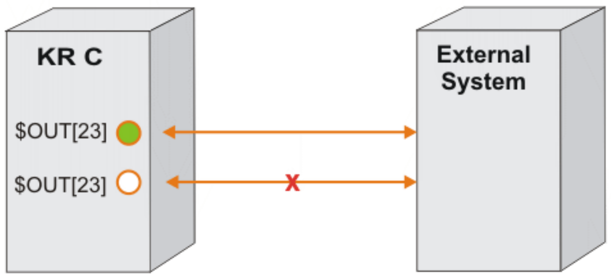


Fig. 6-2: Event output (active connection)

$OUT[23] is set as long as the connection to the external system is active. $OUT[23] is reset when the connection is no longer active.

The connection can be restored only with the EKI_OPEN() function.

Event flag

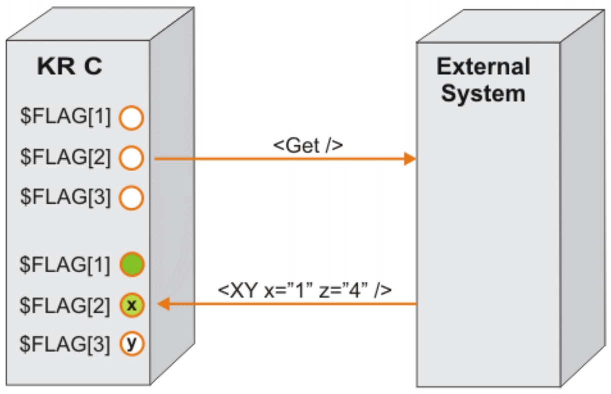


Fig. 6-3: Event flag (complete XML structure)

The XML structure <XY /> contains the “XY/x” and “XY/z” data elements. $FLAG[1] is set since the complete XML structure has arrived at the interface.

$FLAG[2] is set since the “x” element is contained in “XY”. $FLAG[3] is not set since the “y” element has not been transferred.

Example (>>> 7.2.5 "XmlCallback configuration example" Page 46)

6.2.9 Reception of complete XML data records

Description The EKI_Get...() access functions are disabled until all data of an XML data record are in the memory.

If LIFO is configured and two or more XML data records arrive at the interface directly in succession, it is no longer ensured that a data record can be fetched in a non-fragmented condition from the memory. It may, for example, be the case that the data of the second data record are already stored in the memory although the first data record has not yet been completely processed. The data record available in the KRL is inconsistent since, in LIFO mode, the data saved last are always accessed first.

To prevent the fragmentation of data records in LIFO mode, the processing of newly received data must be disabled until all data belonging together have been fetched from the memory.

Example

```
...
RET=EKI_Lock("MyChannel")
RET=EKI_Get...()
RET=EKI_Get...()
...
RET=EKI_Get...()
RET=EKI_Unlock("MyChannel")
...
```

6.2.10 Error treatment

Description For each error or warning EthernetKRL displays a message on the smartHMI. The automatic generation of messages can be deactivated.

(>>> 9.3 "Deactivating the display of messages on the smartHMI" Page 58)

If automatic message generation has been deactivated, it is in any case advisable to check an EKI instruction using EKI_CHECK(). The EKI_CHECK() function can be used to read out an error number and display the message for an error on the smartHMI. If a channel name is specified in EKI_CHECK(), it is checked during data reception whether errors have occurred.

The program EthernetKRL_USER.SRC is called up each time EKI_CHECK() is called up. The file can be found in the directory KRC:\R1\TP\EthernetKRL. User-specific error responses can be programmed in the file.

Example A connection is closed whenever a reception error occurs. An interrupt can be programmed as a fault service function if the Ethernet connection is terminated.

- It is defined in the XmlTransmit.XML configuration file that FLAG[1] is set in the event of a successful connection. FLAG[1] is reset if the connection is lost.

```
<ALIVE Set_Flag="1"/>
```

- The interrupt is declared and switched on in the KRL program. The interrupt program is run if FLAG[1] is reset.

```
;FOLD Define callback
   INTERRUPT DECL 89 WHEN $FLAG[1]==FALSE DO CON_ERR()
   INTERRUPT ON 89
;ENDFOLD
```

- EKI_CHECK() is used in the interrupt program to query what sort of error occurred and then re-open the connection.

```
DEF CON_ERR()
 DECL EKI_STATUS RET
 RET={Buff 0,Read 0, Msg_no 0, Connected false}
 EKI_CHECK(RET,#Quit,"XmlTransmit")
 EKI_OPEN("XmlTransmit")
END
```

7 Examples

7.1 Application examples

Overview

Ethernet KRL comprises application examples which can be used to establish communication between a server program and the robot controller. The software can be found in the directory DOC\Example on the KUKA.USB data stick.

The software consists of the following components:

Component	Folder
EthernetKRL_Server.exe server program	...\Application
Program examples in KRL ■ BinaryFixed.src ■ BinaryStream.src ■ XmlCallback.src ■ XmlServer.src ■ XmlTransmit.src	...\Program
Configuration examples in XML ■ BinaryFixed.xml ■ BinaryStream.xml ■ XmlCallBack.xml ■ XmlServer.xml ■ XmlTransmit.xml ■ XmlFullConfig.xml	...\Config

7.1.1 Implementing application examples

Precondition

External system:

- Windows operating system with .NET Framework installed

Robot controller:

- Expert user group
- T1 or T2 operating mode

Procedure

1. Copy the server program onto an external system.
2. Integrate the SRC files into the VW_USER module.

   (>>> 7.1.2 "Integrating an example program into the VW_USER module" Page 38)
3. Copy the XML files into the directory C:\KRC\ROBOTER\Config\User\Common\EthernetKRL of the robot controller.
4. Start the server program on the external system.

   (>>> 7.1.3 "Server program user interface" Page 39)
5. Press the menu button. The **Communication Properties** window appears.

   (>>> 7.1.4 "Setting communication parameters in the server program" Page 41)
6. Only if several network interfaces are available at the external system: Enter the number of the network adapter (= network card index) used for communication with the robot controller.

7. Close the **Communication Properties** window and press the Start button. The IP address available for communication is displayed in the message window.
8. Set the displayed IP address of the external system in the desired XML file.

7.1.2 Integrating an example program into the VW_USER module

Precondition

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

Procedure

1. Create a new program (Folge).
2. Call VW_USER in the Point PLC of the PTP motion before the start of the Folge.
3. In the inline form **VW User** define for the desired example program a parameter that is to be transferred to the VW_USER module.
4. In the subprogram USER_MAIN() of the file VW_USR_R.SRC, call the example program by means of an IF statement when the defined parameter is transferred.
5. Insert the example program as a subprogram at the end of the file VW_USR_R.SRC.

Example

In the program FOLGE1.SRC, VW_USER is called and the parameter X=12345 is transferred to the VW_USER module.

```
DEF FOLGE1()
  PTP VB=100% VE=0% ACC=100% RobWzg=0 Base=0 SPSTrig=0[1/100s] P
     1: VW USER X=12345 [mm] P2= 1 P3=1 P4=1 P5=1 P6=1 P7=EIN
  Warte auf Folgenstart
...
```

In the subprogram USER_MAIN() of the file VW_USR_R.SRC, it is checked whether the parameter X=12345 (= PAR1) has been transferred. If this is the case, the subprogram BinaryFixed() is called.

```
 1  DEF VW_USR_R(USER_CMD :IN,CMD_SEL :IN, PARA1 :IN,PARA2 :IN,PARA3
                 :IN,PARA4 :IN,PARA5 :IN,PARA6 :IN,PARA7 :IN )
    ...
19  DEF  USER_MAIN (CMD_SEL :IN,PAR1 :IN,PAR2 :IN, PAR3
                    :IN,PAR4 :IN,PAR5 :IN,PAR6 :IN,PAR7 :IN)
20  ;Aufruf im Hauptlauf
21  INT CMD_SEL,PAR1,PAR2,PAR3,PAR4,PAR5,PAR6
22  BOOL PAR7
23
24  IF PAR1 == 12345 THEN
25    BinaryFixed()
26  ENDIF
27
28  END
    ...
32  DEF  USER_MAKRO (CMD_SEL :IN,PAR1 :IN,PAR2 :IN, PAR3
                     :IN,PAR4 :IN,PAR5 :IN,PAR6 :IN,PAR7 :IN)
33  ;Aufruf im Vorlauf
34  INT CMD_SEL,PAR1,PAR2,PAR3,PAR4,PAR5,PAR6
35  BOOL PAR7
36  END
37
38  DEF BinaryFixed( )
39  Declaration
40  INI
41  Initialize sample data
42
43  RET=EKI_Init("BinaryFixed")
44  RET=EKI_Open("BinaryFixed")
45  EKI_CHECK(RET,#QUIT)
46
47  OFFSET=0
48  CAST_TO(Bytes[],OFFSET,34.425,674345,"R",TRUE)
49
50  RET = EKI_Send("BinaryFixed",Bytes[])
51
52  WAIT FOR $FLAG[1]
53  RET=EKI_GetString("BinaryFixed","Buffer",Bytes[])
54  $FLAG[1]=FALSE
55
56  OFFSET=0
57  CAST_FROM(Bytes[],OFFSET,valueReal,valueInt,
              valueChar[],valueBool)
58
59  RET=EKI_Close("BinaryFixed")
60  RET=EKI_Clear("BinaryFixed")
61  END
```

Line	Description
24 … 26	Condition for calling the subprogram BinaryFixed(): PAR1=12345
38 … 61	Subprogram BinaryFixed()

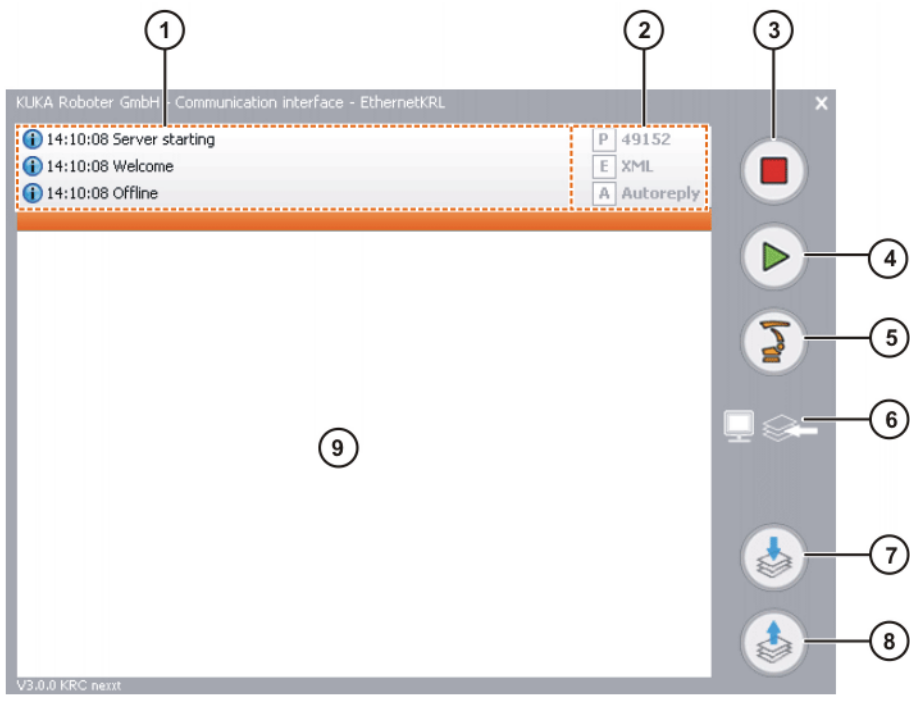
7.1.3 Server program user interface

Description

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

The server program has the following functions:

- Sending and receiving data (automatically or manually)
- Displaying the data received
- Displaying the data sent



14:10:08 Server starting
14:10:08 Welcome
14:10:08 Offline

Fig. 7-1: Server program user interface

Item	Description
1	Message window
2	Display of the communication parameters set (>>> 7.1.4 "Setting communication parameters in the server program" Page 41) ■ **P**: port number ■ **E**: example data ■ **Xml**: XML data ■ **BinaryFixed**: binary data of fixed length ■ **BinaryStream**: variable binary data stream with end string ■ **A**: communication mode ■ **Autoreply**: The server automatically responds to each data package received. ■ **Manual**: only manual data reception or data transmission
3	Stop button Communication with the robot controller is terminated and the server is reset.
4	Start button Data exchange between the server program and robot controller is evaluated. The first incoming connection request is linked and used as a communication adapter.
5	Menu button for setting the communication parameters (>>> 7.1.4 "Setting communication parameters in the server program" Page 41)
6	Display options ■ Arrow pointing to the left: the received RDC data are displayed. (default) ■ Arrow pointing to the right: the sent RDC data are displayed.
7	Button for manual data reception

Item	Description
8	Button for manual data transmission
9	Display window The sent or received data are displayed, depending on the display option set.

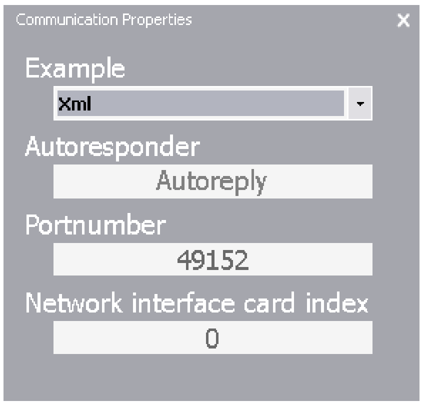
7.1.4 Setting communication parameters in the server program

Procedure

1. Click on the menu button in the server program.

   The **Communication Properties** window appears.
2. Set the communication parameters.
3. Close the window.

Description



Communication Properties

Example: Xml

Autoresponder: Autoreply

Portnumber: 49152

Network interface card index: 0

Fig. 7-2: Communication Properties window

Element	Description
Example	Select example data. ■ **Xml**: XML data ■ **BinaryFixed**: binary data of fixed length ■ **BinaryStream**: variable binary data stream with end string Default value: **xml**
Autoresponder	Select communication mode. ■ **Autoreply**: The server automatically responds to each data package received. ■ **Manual**: only manual data reception or data transmission Default value: **Autoreply**
Portnumber	Enter the port number of the socket connection. The external system awaits the connection request from the robot controller at this port. A free number that is not assigned a standard service must be selected. Default value: **49,152**
Network interface card index:	Enter the number of the network adapter. Only relevant if the external system uses several network cards, e.g. WLAN and LAN. Default value: **0**

7.2 Configuration and program examples

7.2.1 BinaryFixed configuration example

For communication with the robot controller, the appropriate example data must have been set in the server program; in this case **Binary-Fixed**.

The EKI is configured as a client. Only binary data records with a fixed length of 10 bytes and the element name "Buffer" can be received via the connection. The server program sends a data record. $FLAG[1] is set if the interface has received external data.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>49152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="BYTE" Set_Flag="1" Size="10" />
    </RAW>
  </RECEIVE>
  <SEND />
</ETHERNETKRL>
```

Binary data records of fixed length must be read into and out of the KRL program with CAST_TO() and CAST_FROM(). Only data of REAL type (4 bytes) are legible, not Double.

Detailed information on the CAST_TO() and CAST_FROM() commands can be found in the CREAD/CWRITE documentation.

Program

```
DEF BinaryFixed( )
Declaration
INI
Initialize sample data

RET=EKI_Init("BinaryFixed")
RET=EKI_Open("BinaryFixed")

OFFSET=0
CAST_TO(Bytes[],OFFSET,34.425,674345,"R",TRUE)

RET = EKI_Send("BinaryFixed",Bytes[])

WAIT FOR $FLAG[1]
RET=EKI_GetString("BinaryFixed","Buffer",Bytes[])
$FLAG[1]=FALSE

OFFSET=0
CAST_FROM(Bytes[],OFFSET,valueReal,valueInt,
          valueChar[],valueBool)


RET=EKI_Close("BinaryFixed")
RET=EKI_Clear("BinaryFixed")
END
```

Line	Description
4	Initialization of KRL variables by the assignment of values
6	EKI_Init() initializes the channel used by the interface to connect to the external system.
7	EKI_Open() opens the channel and connects to the server.
9, 10	CAST_TO writes the values in the Bytes[] CHAR array.
12	EKI_Send() sends the Bytes[] CHAR array to the external system.
14 … 16	$FLAG[1] indicates the reception of the configured data element. EKI_GetString accesses the memory and copies the data into the Bytes[] CHAR array. $FLAG[1] is reset again.
18, 19	CAST_FROM reads the values out of the Bytes[] CHAR array and copies them type-specifically into the specified variables.
22	EKI_Close() closes the channel.
23	EKI_Clear() clears the channel.

7.2.2 BinaryStream configuration example

For communication with the robot controller, the appropriate example data must have been set in the server program; in this case **BinaryStream**.

The EKI is configured as a client. Only binary data records with a maximum length of 64 bytes and the element name "Buffer" can be received via this connection. The end of the binary data record must be indicated with the end string CR, LF. $FLAG[1] is set when the interface has received this element.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>49152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="1"
                     Size="64" EOS="13,10" />
    </RAW>
  </RECEIVE>
  <SEND />
</ETHERNETKRL>
```

Program

```
 1  DEF BinaryStream( )
 2  Declaration
 3  INI
 4  Initialize sample data
 5
 6  RET=EKI_Init("BinaryStream")
 7  RET=EKI_Open("BinaryStream")
 8
 9  Bytes[]="Stream ends with CR,LF"
10
11  RET = EKI_Send("BinaryStream",Bytes[])
12
13  WAIT FOR $FLAG[1]
14  RET=EKI_GetString("BinaryStream","Buffer",Bytes[])
15  $FLAG[1]=FALSE
16
17  RET=EKI_Close("BinaryStream")
18  RET=EKI_Clear("BinaryStream")
19
20  END
```

Line	Description
4	Initialization of KRL variables by the assignment of values
6	EKI_Init() initializes the channel used by the interface to connect to the external system.
7	EKI_Open() opens the channel and connects to the server.
9	The Bytes[] CHAR array is assigned data.
11	EKI_Send() sends the Bytes[] CHAR array to the external system.
13 … 15	$FLAG[1] indicates the reception of the configured data element. EKI_GetString reads the string in the Bytes[] CHAR array out of the memory. $FLAG[1] is reset again.
17	EKI_Close() closes the channel.
18	EKI_Clear() clears the channel.

7.2.3 XmlTransmit configuration example

For communication with the robot controller, the appropriate example data have to be set in the server program; in this case **Xml**.

The EKI is configured as a client. Robot data are sent and the received sensor data read out of the memory after a waiting time of 1 second.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>49152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/Message" Type="STRING" />
      <ELEMENT Tag="Sensor/Positions/Current/@X" Type="REAL" />
      <ELEMENT Tag="Sensor/Positions/Before/X" Type="REAL" />
      <ELEMENT Tag="Sensor/Nmb" Type="INT" />
      <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
      <ELEMENT Tag="Sensor/Read/xyzabc" Type="FRAME" />
      <ELEMENT Tag="Sensor/Show/@error" Type="BOOL" />
      <ELEMENT Tag="Sensor/Show/@temp" Type="INT" />
      <ELEMENT Tag="Sensor/Show" Type="STRING" />
      <ELEMENT Tag="Sensor/Free" Type="INT" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Robot/Data/LastPos/@X" />
      <ELEMENT Tag="Robot/Data/LastPos/@Y" />
      <ELEMENT Tag="Robot/Data/LastPos/@Z" />
      <ELEMENT Tag="Robot/Data/LastPos/@A" />
      <ELEMENT Tag="Robot/Data/LastPos/@B" />
      <ELEMENT Tag="Robot/Data/LastPos/@C" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
      <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
      <ELEMENT Tag="Robot/Data/ActPos/@X" />
    </XML>
  <SEND />
</ETHERNETKRL>
```

Program

```
1   DEF XmlTransmit( )
2   Declaration
3   Communicated data
4   INI
5   Initialize sample data
6
7   RET=EKI_Init("XmlTransmit")
8   RET=EKI_Open("XmlTransmit")
9
10  Write data to connection
11  Send data to external program
12  Get received sensor data
13
14  RET=EKI_Close("XmlTransmit")
15  RET=EKI_Clear("XmlTransmit")
16
17  END
```

Line	Description
5	Initialization of KRL variables by the assignment of values
7	EKI_Init() initializes the channel used by the interface to con-nect to the external system.
8	EKI_Open() opens the channel and connects to the external system.
10	Writes data in the saved XML document for data transmission.
11	Sends the written XML document to the external system.
12	Reads the received sensor data out of the memory.
14	EKI_Close() closes the channel.
15	EKI_Clear() clears the channel.

7.2.4 XmlServer configuration example

If the interface has been configured as a server, the server program cannot be used on the external system. A simple client can be implemented with Windows HyperTerminal.

The EKI is configured as a server. $FLAG[1] is set as long as a connection to the external system exists.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <TYPE>Client</TYPE>
    </EXTERNAL>
    <INTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>x</PORT>
      <ALIVE Set_Flag="1" />
    </INTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/A" Type="BOOL" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Robot/B" />
    </XML>
  </SEND>
</ETHERNETKRL>
```

Program

```
 1  DEF XmlServer( )
 2  Declaration
 3  INI
 4
 5  RET=EKI_Init("XmlServer")
 6  RET=EKI_Open("XmlServer")
 7
 8  ; wait until server is conntected
 9  wait for $FLAG[1]
10  ; wait until server is deconnected
11  wait for $FLAG[1]==FALSE
12
13  RET=EKI_Clear("XmlServer")
14  END
```

Line	Description
5	EKI_Init() initializes the channel used by the external system to connect to the interface.
6	EKI_Open() opens the channel.
9	$FLAG[1] is set when the external client has connected successfully to the server.
11	Since the interface is configured as a server, the robot controller expects the channel to be closed by the external client. In this case, $FLAG[1] is deleted.
13	EKI_Clear() clears the channel.

7.2.5 XmlCallback configuration example

For communication with the robot controller, the appropriate example data have to be set in the server program; in this case **Xml**.

The EKI is configured as a client. Robot data are sent, sensor data received and then $FLAG[1] awaited. $FLAG[1] indicates that the sensor data have been read out.

It is configured in the XML file that $FLAG[998] is set when the interface has received all sensor data. This flag triggers an interrupt in the program. The configuration of the "Sensor" tag as event tag ensures that the sensor data are fetched only when all data are in the memories.

$FLAG[998] is reset and $FLAG[1] set when the sensor data have been read out.

XML file

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>49152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/Message" Type="STRING" />
      <ELEMENT Tag="Sensor/Positions/Current/@X" Type="REAL" />
      <ELEMENT Tag="Sensor/Positions/Before/X" Type="REAL" />
      <ELEMENT Tag="Sensor/Nmb" Type="INT" />
      <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
      <ELEMENT Tag="Sensor/Read/xyzabc" Type="FRAME" />
      <ELEMENT Tag="Sensor/Show/@error" Type="BOOL" />
      <ELEMENT Tag="Sensor/Show/@temp" Type="INT" />
      <ELEMENT Tag="Sensor/Show" Type="STRING" />
      <ELEMENT Tag="Sensor/Free" Type="INT" />
      <ELEMENT Tag="Sensor/Show/@flag" Type="BOOL" Set_Flag="999" />
      <ELEMENT Tag="Sensor/Show/@out" Type="INT" Set_Out="999" />
      <ELEMENT Tag="Sensor" Set_Flag="998" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Robot/Data/LastPos/@X" />
      <ELEMENT Tag="Robot/Data/LastPos/@Y" />
      <ELEMENT Tag="Robot/Data/LastPos/@Z" />
      <ELEMENT Tag="Robot/Data/LastPos/@A" />
      <ELEMENT Tag="Robot/Data/LastPos/@B" />
      <ELEMENT Tag="Robot/Data/LastPos/@C" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
      <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
      <ELEMENT Tag="Robot/Data/ActPos/@X" />
    </XML>
  <SEND />
</ETHERNETKRL>
```

Program

```
 1  DEF XmlCallBack( )
 2  Declaration
 3  Communicated data
 4  INI
 5  Define callback
 6
 7  RET=EKI_Init("XmlCallBack")
 8  RET=EKI_Open("XmlCallBack")
 9
10  Write data to connection
11  RET = EKI_Send("XmlCallBack","Robot")
12
13  ;wait until data read
14  WAIT FOR $FLAG[1]
15
16  RET=EKI_Close("XmlCallBack")
17  RET=EKI_Clear("XmlCallBack")
18  END
19
20  DEF GET_DATA()
21  Declaration
22  Initialize sample data
23  Get received sensor data
24  Signal read
```

Line	Description
5	Declaring and switching on the interrupt
7	EKI_Init() initializes the channel used by the interface to connect to the external system.
8	EKI_Open() opens the channel.
10	Writes data in the saved XML document for data transmission.
11	Sends the data.
14	Waits for $FLAG[1]. The event flag signals that all data have been read.
16	EKI_Close() closes the channel.
17	EKI_Clear() clears the channel.
20 … 24	initialization of KRL variables by assigning values and reading out data $FLAG[1] is set when all data have been read.

Data transmission

The XML document is assigned robot data by the KRL program and sent to the external system via the EKI.

```
<Robot>
  <Data>
    <ActPos X="1000.12">
    </ActPos>
    <LastPos A="..." B="..." C="..." X="..." Y="..." Z="...">
    </LastPos>
  </Data>
  <Mode>ConnectSensor</Mode>
  <RobotLamp>
    <GrenLamp>
      <LightOn>1</LightOn>
    </GrenLamp>
  </RobotLamp>
  <Status>12345678</Status>
</Robot>
```

Data reception

The XML document is assigned sensor data by the server program and received by the EKI.

```
<Sensor>
  <Message>Example message</Message>
  <Positions>
    <Current X="4645.2" />
    <Before>
      <X>0.9842</X>
    </Before>
  </Positions>
  <Nmb>8</Nmb>
  <Status>
    <IsActive>1</IsActive>
  </Status>
  <Read>
    <xyzabc X="210.3" Y="825.3" Z="234.3" A="84.2" B="12.3"
            C="43.5" />
  </Read>
  <Show error="0" temp="9929">Taginfo in attributes</Show>
  <Free>2912</Free>
</Sensor>
```

8 Diagnosis

8.1 Displaying diagnostic data for Ethernet KRL

Procedure

1. Select **Diagnosis** > **Diagnostic monitor** in the main menu.
2. Select the **EKI (EthernetKRL)** module in the **Module** field.

Description

Diagnostic data for Ethernet KRL:

Name	Description
Total memory	Total available memory (bytes)
Allocated memory	Used memory (bytes)
Robot program connections	Number of connections initialized by the robot interpreter
Submit program connections	Number of connections initialized by the Submit interpreter
System connections	Number of connections initialized by the system
Ethernet connections	Number of open connections
Execution time	Maximum time required to process received data (refreshed every 5 seconds)

8.2 Error protocol (EKI logbook)

All error messages of the interface are logged in a LOG file under C:\KRC\ROBOTER\LOG\EthernetKRL.

8.3 Error messages

Each Ethernet KRL function has an EKI_STATUS return value which contains an error number. The error numbers are assigned a message text which is displayed on the smartHMI. If the automatic generation of messages is deactivated, the message can still be displayed on the smartHMI by means of EKI_CHECK().

No.	Message text	Cause	Remedy
1	*Unknown error*	No message has been assigned to the error.	Contact KUKA Roboter GmbH and submit the logbook with details on the error. (>>> 10 "KUKA Service" Page 69)
2	*Out of system memory*	The memory reserved for Ethernet KRL is completely occupied. No more elements can be saved.	Check the programming method in KRL and the configuration of the Ethernet connection. If no other type of programming or configuration is possible, the memory can be increased in consultation with KUKA Roboter GmbH. (>>> 9.2 "Increasing the memory" Page 57)

No.	Message text	Cause	Remedy
3	*File access failed*	A file could not be found or is not legible.	Check whether the file is present or whether the file can be opened.
4	*Requested function not implemented*	Software error: The Ethernet KRL function used has not been implemented.	Contact KUKA Roboter GmbH and submit the logbook with details on the error. (>>> 10 "KUKA Service" Page 69)
5	*Creation of XML parser failed*	The connection has not been initialized, since the internal system parser could not be activated.	Contact KUKA Roboter GmbH and submit the logbook with details on the error. (>>> 10 "KUKA Service" Page 69)
6	*Interpretation of configuration failed*	Error when reading the connection configuration	Check the configuration of the Ethernet connection.
7	*Writing of data to send failed*	Error when writing the XML structure for the data transmission	check configuration of transmission structure.
8	*Add new element failed*	Error when creating the data memory	Contact KUKA Roboter GmbH and submit the logbook with details on the error. (>>> 10 "KUKA Service" Page 69)
9	*Connection not available*	No access to the connection is possible due to the missing initialization.	Initialize the Ethernet connection with EKI_Init().
10	*Ethernet is disconnected*	No Ethernet connection is present.	Open the Ethernet connection with EKI_Open().
11	*Ethernet connection to external system established*	Ethernet connection is already present.	Do not call the EKI_Open() function if the Ethernet connection already exists.
12	*Create server failed*	An Ethernet connection configured as a server could not be created.	Check configuration of connection parameters (IP elements, PORT).
13	*Initialization of Ethernet parameters failed*	Error when initializing the Ethernet connection	Check configuration of connection parameters (IP elements, PORT).
14	*Ethernet connection to external system failed*	No Ethernet connection: ■ Hardware error, e.g. network cable, switch, external system ■ Software error (external system) ■ Error during the connection configuration	Establish an Ethernet connection: ■ Check hardware. ■ Check software of external system. ■ Check configuration of connection parameters (IP elements, PORT).

No.	Message text	Cause	Remedy
15	*Access to empty element memory*	No data elements in the memory when accessed with EKI_Get…()	Evaluate the return value of the EKI_Get…() function in the KRL program to avoid accessing empty memories ("Buff" element). (>>> 6.2.7 "Return value of the Ethernet KRL functions" Page 33)
16	*Element not found*	A data element specified in the EKI_Get…() access function cannot be found.	■ Check the name of the data element and its notation in the KRL program. ■ Check the configuration of the reception structure.
17	*Assembly of data to send failed*	■ Sending XML data: error when writing the XML document for data transmission ■ Sending binary data: error when checking the binary data to be sent	■ Sending XML data: check configuration of transmission structure. ■ Sending binary data: Check the EKI_Send() function in the KRL program.
18	*Send data failed*	No Ethernet connection: ■ Hardware error, e.g. network cable, switch, external system ■ Software error (external system)	Establish an Ethernet connection: ■ Check hardware. ■ Check software of external system.
19	*No data to send*	The data to be sent are not specified in an EKI_Send() function.	Check the EKI_Send() function in the KRL program.
20	*Mismatch in type of data*	An attempt was made to read an element which belongs to a different data type.	Check the data type of the element in the configuration of the reception structure. OR Use the data type defined in the configuration of the reception structure in the KRL program.
21	*System memory insufficient with maximum data storage*	It was established that the system memory is insufficient when reading in the configuration.	Check the configuration of the Ethernet connection and adjust it so that less memory is used. If no other configuration is possible, the memory can be increased in consultation with KUKA Roboter GmbH. (>>> 9.2 "Increasing the memory" Page 57)

No.	Message text	Cause	Remedy
22	*Error while reading the configuration. XML not valid.*	An error in the XML structure was detected when reading in the configuration.	Check the XML structure in the configuration file.
24	*Link to internal parameters (Port, IP) failed*	The Ethernet connection, i.e. the interface, is configured as a server. The IP address and port number of the external system specified in the configuration are not available.	Use the correct IP address and port number in the configuration of the connection parameters (IP elements, PORT).
25	*Internal software error*	Internal software error	Contact KUKA Roboter GmbH and submit the logbook with details on the error. (>>> 10 "KUKA Service" Page 69)
26	*FRAME array not initialized*	An FRAME type array has not been initialized.	Initialize the FRAME type array (assign value).
27	*CHAR[] Array too small.*	A CHAR type array is too small.	Increase the number of array elements.
512	*Ethernet connection disrupted*	No Ethernet connection: ■ Hardware error, e.g. network cable, switch, external system ■ Software error (external system)	Restore the Ethernet connection: ■ Check hardware. ■ Check software of external system.
768	*Ping reports no contact*	The external system no longer responds to the ping sent. The connection is aborted.	Check external system.
1024	*Error while reading received XML data*	An XML document received from the external system does not correspond to the XPath schema.	Check the XML document sent by the external system.
1280	*Limit of element storage reached*	The data memory is assigned the maximum number of data elements. The Ethernet connection is closed.	Evaluate the return value of the EKI_Get…() function in the KRL program to disable the processing of received data ("Buff" element). (>>> 6.2.7 "Return value of the Ethernet KRL functions" Page 33) OR Increase the data memory (BUFFERING element in the connection configuration).

No.	Message text	Cause	Remedy
1536	*Received string too long*	Programming error on external system: a string received from the external system exceeds the maxi-mum permissible length (max. 3,600 characters).	Check the data sent by the external system.
1792	*Limit of element memory reached*	The data memory is assigned the maximum number of bytes. The Ethernet connection is closed.	Increase the data memory (BUFFSIZE element in the connection configuration).
2048	*Server time limit reached*	Server is waiting for a call.	Check external system.

9 Appendix

9.1 Extended XML structure for connection properties

The extended XML structure may only be used in consultation with KUKA Roboter GmbH. (>>> 10 "KUKA Service" Page 69)

Description

Further interface properties can be configured in the section <INTERNAL> … </INTERNAL> of the configuration file:

Element	Attribute	Description
TIMEOUT	Receive	Time until the attempt to receive data is aborted (optional) ■ **0 … 65,534 ms** Default value: **0 ms**
	Send	Time until the attempt to send data is aborted (optional) ■ **0 … 65,534 ms** Default value: **0 ms**
BUFFSIZE	Receive	Size of the socket used to receive data (optional) ■ **1 … 65,534 bytes** Default value: Predefined by the system
	Send	Size of the socket used to send data (optional) ■ **1 … 65,534 bytes** Default value: Predefined by the system

9.2 Increasing the memory

The memory may be increased only in consultation with KUKA Robot-er GmbH. (>>> 10 "KUKA Service" Page 69)

Description

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

- Check whether a connection is configured in such a manner that the memory is completely occupied with received data.
- Check whether several connections with high levels of data have been defined and activated.

Precondition

- Windows interface

Procedure

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

```
<EthernetKRL>
   <Interface>
      <MemSize>1048576</MemSize>
...
</EthernetKRL>
```

3. Save the change and close the file.

9.3 Deactivating the display of messages on the smartHMI

Description

Checking for errors and automatically displaying messages on the smartHMI can be deactivated. This is recommended if runtime problems arise or the EKI is used in the submit interpreter.

If automatic messages are deactivated, the EKI_CHECK() function can be used to check individual EKI instructions for errors.

Precondition

- "Expert" user group

Procedure

1. Open the file KRC:\R1\TP\EthernetKRL\EthernetKRL.DAT.
2. Set the variable SHOWMSG to FALSE.
3. Save the change and close the file.

Deactivation affects only those messages displayed on the smartHMI. Warnings and error messages are still registered in the EKI logbook.

9.4 Deactivating warning messages in the EKI logbook

Description

If the automatic message display on the smartHMI is deactivated, all warnings and errors are still written into the EKI logbook. If these errors and warnings are to be deliberately ignored, this mechanism must be deactivated. (Access to files otherwise causes an unnecessary load on the system.)

Precondition

- Windows interface

Procedure

1. Open the file C:\KRC\ROBOTERConfig\User\Common\Logging_EthernetKRL.XML.
2. Change the **LogLevel** attribute in the XML element **Class**.
3. Save the change and close the file.

The following values are available:

LogLevel	Description
warning	Warning messages and error messages are written into the logbook. (Default)
error	Only error messages are written into the logbook.
disabled	No more warning or error messages are written into the logbook.

9.5 Ethernet KRL functions command reference

9.5.1 Initialization and connection functions

RET = EKI_Init(CHAR[])	
Function	Initializes a channel for Ethernet communication The following actions are performed: ■ The configuration is read in. ■ The data memories are created. ■ The Ethernet connection is prepared.
Parameter	Type: CHAR Name of channel
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_Init("Channel_1")

RET = EKI_Open(CHAR[])	
Function	Opens an initialized channel If the Ethernet KRL interface is configured as a client, the interface connects to the server. If the Ethernet KRL interface is configured as a server, the interface waits for the connection.
Parameter	Type: CHAR Name of channel
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_Open("Channel_1")

RET = EKI_Close(CHAR[])	
Function	Closes an open channel
Parameter	Type: CHAR Name of channel
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_Close("Channel_1")

RET = EKI_Clear(CHAR[])	
Function	Deletes a channel and terminates the connection.
Parameter	Type: CHAR Name of channel

RET = EKI_Clear(CHAR[])	
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_Clear("Channel_1")

9.5.2 Transmission function

RET = EKI_Send(CHAR[], CHAR[])	
Function	Sends an XML structure or raw data (>>> 6.2.4 "Sending data" Page 30)
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure or name of the element in the raw data If the position or element is not found, the function sends the information contained here
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example 1	RET = EKI_Send("Channel_1", "Root/Test")
Example 2	RET = EKI_Send("Channel_1", MyBytes[])

9.5.3 Write functions

RET = EKI_SetReal(CHAR[], CHAR[], REAL)	
Function	Writes a floating point value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Value written in the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_SetReal("Channel_1", "Root/Number", 1.234)

RET = EKI_SetInt(CHAR[], CHAR[], INTEGER)	
Function	Writes an integer value in a memory
Parameter 1	Type: CHAR Name of the open channel

RET = EKI_SetInt(CHAR[], CHAR[], INTEGER)	
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Value written in the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_SetInt("Channel_1", "Root/List", 67234)

RET = EKI_SetBool(CHAR[], CHAR[], BOOL)	
Function	Writes a Boolean value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL Value written in the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_SetBool("Channel_1", "Root/Activ", true)

RET = EKI_SetFrame(CHAR[], CHAR[], FRAME)	
Function	Writes a FRAME type value in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Value written in the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET= EKI_SetFrame("Channel_1", "Root/BASE", {X 0.0, Y 0.0, Z 0.0, A 0.0, B 0.0, C 0.0})

RET = EKI_SetString(CHAR[], CHAR[], CHAR[])	
Function	Writes a string in a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure

RET = EKI_SetString(CHAR[], CHAR[], CHAR[])	
Parameter 3	Type: CHAR String written in the memory Maximum number of characters: ■ **3 600**
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_SetString("Channel_1", "Root/Message", "Hello")

9.5.4 Access functions

RET = EKI_GetBool(CHAR[], CHAR[], BOOL)	
Function	Reads a Boolean value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL Value read out of the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetBool("Channel_1", "Root/Activ", MyBool)

RET = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])	
Function	Reads a Boolean value out of the memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: BOOL Array read out of the memory Maximum number of readable array elements: ■ **512**

RET = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])	
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetBoolArray("Channel_1", "Root/Activ", MyBool[])

RET = EKI_GetInt(CHAR[], CHAR[], Int)	
Function	Reads an integer value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Value read out of the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetInt("Channel_1", "Root/Numbers/One", MyInteger)

RET = EKI_GetIntArray(CHAR[], CHAR[], Int[])	
Function	Reads an integer value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: INT Array read out of the memory Maximum number of readable array elements: ■ **512**
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetIntArray("Channel_1", "Root/Numbers/One", MyInteger[])

RET = EKI_GetReal(CHAR[], CHAR[], Real)	
Function	Reads a floating point value out of a memory
Parameter 1	Type: CHAR Name of the open channel

RET = EKI_GetReal(CHAR[], CHAR[], Real)	
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Value read out of the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetReal("Channel_1", "Root/Position", MyReal)

RET = EKI_GetRealArray(CHAR[], CHAR[], Real[])	
Function	Reads a floating point value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: REAL Array read out of the memory Maximum number of readable array elements: ■ **512**
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetRealArray("Channel_1", "Root/Position", MyReal[])

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
Function	Reads a string out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure or name of the element in the raw data
Parameter 3	Type: CHAR String read out of the memory Maximum number of characters: ■ **3 600**

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
XML example	RET = EKI_GetString("Channel_1", "Root/Message", MyChars[])
Binary example	RET = EKI_GetString("Channel_1", "Streams", MyS-tream[])

RET = EKI_GetFrame(CHAR[], CHAR[], FRAME)	
Function	Reads a FRAME type value out of a memory
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Value read out of the memory
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetFrame("Channel_1", "Root/TCP", MyFrame)

RET = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])	
Function	Reads a FRAME type value out of a memory and copies the value into the array transferred by the KRL program Values are read until the array is full or no element is present anymore.
Parameter 1	Type: CHAR Name of the open channel
Parameter 2	Type: CHAR Name of the position in the XML structure
Parameter 3	Type: FRAME Array read out of the memory Maximum number of readable array elements: ■ **512**
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example	RET = EKI_GetFrameArray("Channel_1", "Root/TCP", MyFrame[])

9.5.5 Error treatment function

EKI_CHECK(EKI_STATUS, EKrlMsgType, CHAR[])	
Function	Displays a message relating to the error number in parameter 1 or checks for errors if a channel name is specified in parameter 3. The error message is displayed in the message window.
Parameter 1	Return value of an Ethernet KRL function (>>> 6.2.7 "Return value of the Ethernet KRL functions" Page 33)
Parameter 2	Type: ENUM Message type displayed for the error ■ **#NOTIFY**: Notification message ■ **#STATE**: Status message ■ **#QUIT**: Acknowledgement message ■ **#WAITING**: Wait message
Parameter 3 (optional)	Type: CHAR Name of the open channel
Example 1	EKI_CHECK(RET,#QUIT)
Example 2	EKI_CHECK(RET,#NOTIFY,"MyChannelName")

9.5.6 Other functions

RET = EKI_ClearBuffer(CHAR[], CHAR[])	
Function	Deletes data which have been received but not yet called up from a memory
Parameter 1	Type: CHAR Name of channel
Parameter 2	Type: CHAR Position of the memory or all memories (>>> 6.2.6 "Deleting received data" Page 33)
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)
Example 1	RET = EKI_ClearBuffer("Channel_1", "Root/Activ/Flag")
Example 2	RET = EKI_ClearBuffer("Channel_1", "Root")

RET = EKI_Lock(CHAR[])	
Function	Disables the processing of received data, i.e. the data can no longer be stored in the memory.
Parameter	Type: CHAR Name of channel
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)

RET = EKI_Unlock(CHAR[])	
Function	Enables the processing of received data, i.e. the data are stored in the memory again.
Parameter	Type: CHAR Name of channel
RET	Type: EKI_STATUS Return value which contains the message number of the error (>>> 9.5.5 "Error treatment function" Page 66)

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.

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 VW System Software
- Optional software or modifications
- Archive of the software

  For VW System Software V8: instead of a conventional archive, generate the special data package for fault analysis (via **KrcDiag**).
- 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

Headland Machinery Pty. Ltd.
Victoria (Head Office & Showroom)
95 Highbury Road
Burwood
Victoria 31 25
Australia
Tel. +61 3 9244-3500
Fax +61 3 9244-3501
vic@headland.com.au
www.headland.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 Automation Equipment (Shanghai) Co., Ltd. Songjiang Industrial Zone No. 388 Minshen Road 201612 Shanghai China Tel. +86 21 6787-1808 Fax +86 21 6787-1805 info@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 India Pvt. Ltd. Office Number-7, German Centre, Level 12, Building No. - 9B DLF Cyber City Phase III 122 002 Gurgaon Haryana India Tel. +91 124 4635774 Fax +91 124 4635773 info@kuka.in www.kuka.in
Italy	KUKA Roboter Italia S.p.A. Via Pavia 9/a - int.6 10098 Rivoli (TO) Italy Tel. +39 011 959-5013 Fax +39 011 959-5141 kuka@kuka.it www.kuka.it
Japan	KUKA Robotics Japan K.K. Daiba Garden City Building 1F 2-3-5 Daiba, Minato-ku Tokyo 135-0091 Japan Tel. +81 3 6380-7311 Fax +81 3 6380-7312 info@kuka.co.jp
Korea	KUKA Robotics Korea Co. Ltd. RIT Center 306, Gyeonggi Technopark 1271-11 Sa 3-dong, Sangnok-gu Ansan City, Gyeonggi Do 426-901 Korea Tel. +82 31 501-1451 Fax +82 31 501-1461 info@kukakorea.com

Malaysia	KUKA Robot Automation 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 Sentrumsvegen 5 2867 Hov Norway Tel. +47 61 18 91 30 Fax +47 61 18 62 00 info@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
Poland	KUKA Roboter Austria GmbH Spółka z ograniczoną odpowiedzialnością Oddział w Polsce Ul. Porcelanowa 10 40-246 Katowice Poland Tel. +48 327 30 32 13 or -14 Fax +48 327 30 32 26 ServicePL@kuka-roboter.de

Portugal	KUKA Sistemas de Automatización S.A. Rua do Alto da Guerra n° 50 Armazém 04 2910 011 Setúbal Portugal Tel. +351 265 729780 Fax +351 265 729782 kuka@mail.telepac.pt
Russia	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 Industriestr. 9 5432 Neuenhof Switzerland Tel. +41 44 74490-90 Fax +41 44 74490-91 info@kuka-roboter.ch www.kuka-roboter.ch
Spain	KUKA Robots IBÉRICA, S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spain Tel. +34 93 8142-353 Fax +34 93 8142-950 Comercial@kuka-e.com www.kuka-e.com

South Africa	Jendamark Automation LTD (Agency) 76a York Road North End 6000 Port Elizabeth South Africa Tel. +27 41 391 4700 Fax +27 41 373 3869 www.jendamark.co.za
Taiwan	KUKA Robot Automation Taiwan Co., Ltd. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 info@kuka.com.tw www.kuka.com.tw
Thailand	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 atika@ji-net.com www.kuka-roboter.de
Czech Republic	KUKA Roboter Austria GmbH Organisation Tschechien und Slowakei Sezemická 2757/2 193 00 Praha Horní Počernice Czech Republic Tel. +420 22 62 12 27 2 Fax +420 22 62 12 27 0 support@kuka.cz
Hungary	KUKA Robotics Hungaria Kft. Fö út 140 2335 Taksony Hungary Tel. +36 24 501609 Fax +36 24 477031 info@kuka-robotics.hu

USA	KUKA Robotics 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