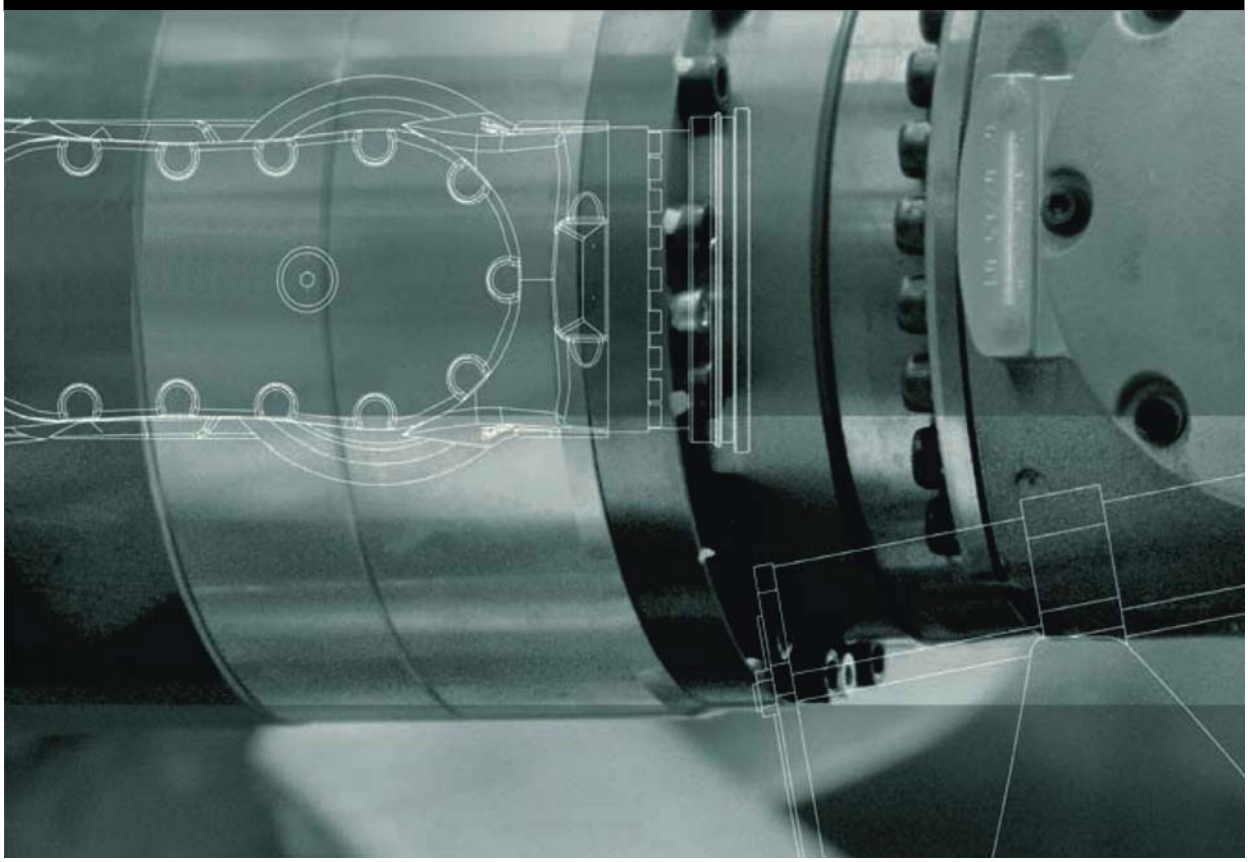
KUKA

KUKA System Technology

KUKA Roboter GmbH

KUKA.EthernetKRL 2.2

Für KUKA System Software 8.2 und 8.3



Stand: 19.11.2012

Version: KST EthernetKRL 2.2 V1 de (PDF)

KUKA Roboter GmbH
Zugspitzstraße 140
D-86165 Augsburg
Deutschland

Es können weitere, in dieser Dokumentation nicht beschriebene Funktionen in der Steuerung lauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei Neulieferung bzw. im Servicefall.

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden jedoch regelmäßig überprüft und notwendige Korrekturen sind in der nachfolgenden Auflage enthalten.

Technische Änderungen ohne Beeinflussung der Funktion vorbehalten.

Original-Dokumentation

KIM-PS5-DOC

Publikation:	Pub KST EthernetKRL 2.2 (PDF) de
Buchstruktur:	KST EthernetKRL 2.2 V1.1
Version:	KST EthernetKRL 2.2 V1 de (PDF)

Inhaltsverzeichnis

1 Einleitung

1.1 Zielgruppe

Diese Dokumentation richtet sich an Benutzer mit folgenden Kenntnissen:

- Fortgeschrittene KRL-Programmierkenntnisse
- Fortgeschrittene Systemkenntnisse der Robotersteuerung
- Fortgeschrittene XML-Kenntnisse
- Fortgeschrittene Netzwerk-Kenntnisse

Für den optimalen Einsatz unserer Produkte empfehlen wir unseren Kunden eine Schulung im KUKA College. Informationen zum Schulungsprogramm sind unter www.kuka.com oder direkt bei den Niederlassungen zu finden.

1.2 Dokumentation des Industrieroboters

Die Dokumentation zum Industrieroboter besteht aus folgenden Teilen:

- Dokumentation für die Robotermechanik
- Dokumentation für die Robotersteuerung
- Bedien- und Programmieranleitung für die KUKA System Software
- Anleitungen zu Optionen und Zubehör
- Teilekatalog auf Datenträger

Jede Anleitung ist ein eigenes Dokument.

1.3 Darstellung von Hinweisen

Sicherheit

Diese Hinweise dienen der Sicherheit und **müssen** beachtet werden.

GEFAHR Diese Hinweise bedeuten, dass Tod oder schwere Verletzungen sicher oder sehr wahrscheinlich eintreten **werden**, wenn keine Vorsichtsmaßnahmen getroffen werden.

WARNUNG Diese Hinweise bedeuten, dass Tod oder schwere Verletzungen eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

VORSICHT Diese Hinweise bedeuten, dass leichte Verletzungen eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

HINWEIS Diese Hinweise bedeuten, dass Sachschäden eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

Diese Hinweise enthalten Verweise auf sicherheitsrelevante Informationen oder allgemeine Sicherheitsmaßnahmen.
Diese Hinweise beziehen sich nicht auf einzelne Gefahren oder einzelne Vorsichtsmaßnahmen.

Dieser Hinweis macht auf Vorgehensweisen aufmerksam, die der Vorbeugung oder Behebung von Not- oder Störfällen dienen:

SICHERHEITS-ANWEISUNGEN	Mit diesem Hinweis gekennzeichnete Vorgehensweisen **müssen** genau eingehalten werden.

Hinweise

Diese Hinweise dienen der Arbeitserleichterung oder enthalten Verweise auf weiterführende Informationen.

	Hinweis zur Arbeitserleichterung oder Verweis auf weiterführende Informationen.

1.4 Verwendete Begriffe

Begriff	Beschreibung
Datenstrom	Kontinuierliche Abfolgen von Datensätzen, deren Ende nicht im Voraus abzusehen ist. Die einzelnen Datensätze sind von beliebigem, aber festem Typ. Die Menge der Datensätze pro Zeiteinheit (Datenrate) kann variieren. Es ist nur ein sequentieller Zugriff auf die Daten möglich.
EKI	EthernetKRL Interface
EOS	End Of Stream (Endzeichenfolge) Zeichenfolge, die das Ende eines Datensatzes kennzeichnet
Ethernet	Ethernet ist eine Datennetztechnologie für lokale Datennetze (LANs). Sie ermöglicht den Datenaustausch in Form von Datenrahmen zwischen den verbundenen Teilnehmern.
FIFO LIFO	Verfahren, nach denen ein Datenspeicher abgearbeitet werden kann ■ **First In First Out**: Elemente, die zuerst gespeichert wurden, werden zuerst wieder aus dem Speicher entnommen. ■ **Last In First Out**: Elemente, die zuletzt gespeichert wurden, werden zuerst wieder aus dem Speicher entnommen.
KLI	KUKA Line Interface Linienbus zur Integration der Anlage in das Kundennetz
KR C	KUKA Robot Controller KR C ist die KUKA Robotersteuerung.
KRL	KUKA Robot Language KRL ist die KUKA Roboter Programmiersprache.
smartHMI	smart Human-Machine Interface KUKA smartHMI ist die Bedienoberfläche der KUKA System Software.
Socket	Software-Schnittstelle, die IP-Adressen und Port-Nummern miteinander verbindet

Begriff	Beschreibung
TCP/IP	Transmission Control Protocol Protokoll über den Datenaustausch zwischen den Teilnehmern eines Netzwerks. TCP stellt einen virtuellen Kanal zwischen 2 Endpunkten einer Netzwerkverbindung her. Auf diesem Kanal können in beide Richtungen Daten übertragen werden.
UDP/IP	User Datagram Protocol Verbindungsloses Protokoll über den Datenaustausch zwischen den Teilnehmern eines Netzwerks
IP	Internet Protocol Das Internet-Protokoll hat die Aufgabe, Subnetze über physikalische MAC-Adressen zu definieren.
XML	Extensible Markup Language Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer vorgegebenen Baumstruktur
XPath	XML Path Language Sprache, um Teile eines XML-Dokumentes zu beschreiben und zu lesen

1.5 Warenzeichen

.NET Framework ist ein Warenzeichen der Microsoft Corporation.

Windows ist ein Warenzeichen der Microsoft Corporation.

2 Produktbeschreibung

2.1 Übersicht EthernetKRL

Funktionen

EthernetKRL ist ein nachladbares Technologiepaket mit folgenden Funktionen:

- Datenaustausch über die EthernetKRL-Schnittstelle
- Empfangen von XML-Daten eines externen Systems
- Senden von XML-Daten an ein externes System
- Empfangen von Binär-Daten eines externen Systems
- Senden von Binär-Daten an ein externes System

Eigenschaften

- Robotersteuerung und externes System als Client oder Server
- Konfiguration von Verbindungen über XML-basierte Konfigurationsdatei
- Konfiguration von "Ereignismeldungen"
- Überwachen von Verbindungen durch einen Ping auf das externe System
- Lesen und Schreiben von Daten aus dem Submit-Interpreter
- Lesen und Schreiben von Daten aus dem Roboter-Interpreter

Kommunikation

Daten werden über das TCP/IP-Protokoll übertragen. Die Verwendung des UDP/IP-Protokolls ist möglich, wird aber nicht empfohlen (verbindungsloses Netzwerkprotokoll, z. B. kein Erkennen von Datenverlust).

Die Kommunikationszeit ist abhängig von den in KRL programmierten Aktionen und vom gesendeten Datenvolumen. Je nach Programmierweise in KRL können bis zu 2 ms Paket-Umlaufzeit erreicht werden.

2.2 Konfiguration einer Ethernet-Verbindung

Beschreibung

Die Ethernet-Verbindung wird über eine XML-Datei konfiguriert. Für jede Verbindung muss im Verzeichnis C:\KRC\ROBOTER\Config\User\Common\EthernetKRL der Robotersteuerung eine Konfigurationsdatei definiert sein. Die Konfiguration wird beim Initialisieren einer Verbindung eingelesen.

Ethernet-Verbindungen können vom Roboter- oder Submit-Interpreter angelegt und bedient werden. Die Kanäle sind über Kreuz verwendbar, z. B. kann ein im Submit-Interpreter geöffneter Kanal auch vom Roboter-Interpreter bedient werden.

Das Löschen einer Verbindung kann an Roboter- und Submit-Interpreter-Aktionen oder Systemaktionen gekoppelt sein.

2.2.1 Verhalten bei Verbindungsverlust

Beschreibung

Folgende Eigenschaften und Funktionen der EKI stellen sicher, dass empfangene Daten zuverlässig bearbeitet werden können:

- Bei Erreichen des Limits eines Datenspeichers wird eine Verbindung automatisch geschlossen.
- Wenn ein Fehler beim Datenempfang auftritt, wird eine Verbindung automatisch geschlossen.
- Bei geschlossener Verbindung werden die Datenspeicher weiterhin ausgelesen.
- Bei Verbindungsverlust kann ohne Einfluss auf gespeicherte Daten die Verbindung wiederhergestellt werden.
- Ein Verbindungsverlust kann angezeigt werden, z. B. über ein Flag.

- Zum Fehler, der zu einem Verbindungsverlust geführt hat, kann die Fehlermeldung auf der smartHMI ausgegeben werden.

2.2.2 Überwachen einer Verbindung

Beschreibung

Eine Verbindung kann durch einen Ping auf das externe System überwacht werden. (Element <ALIVE…/> in der Verbindungskonfiguration)

Bei erfolgreicher Verbindung kann je nach Konfiguration ein Flag oder ein Ausgang gesetzt werden. Solange das Ping regelmäßig gesendet wird und die Verbindung zum externen System aktiv ist, ist der Ausgang oder das Flag gesetzt. Wenn die Verbindung zum externen System abbricht, wird der Ausgang oder das Flag gelöscht.

2.3 Datenaustausch

Übersicht

Über EthernetKRL kann die Robotersteuerung sowohl Daten von einem externen System empfangen als auch Daten an ein externes System senden.

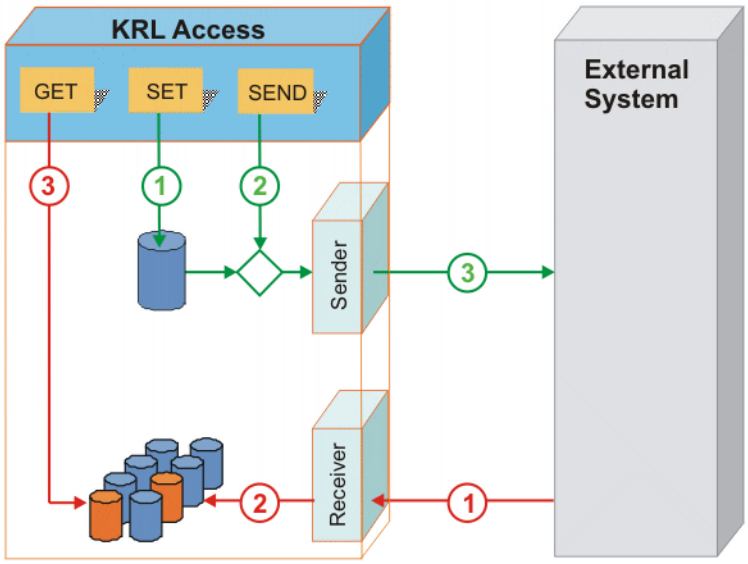


Abb. 2-1: Systemübersicht

Datenempfang

Prinzipieller Ablauf (Rot markiert) (>>> Abb. 2-1):

1. Das externe System sendet Daten, die über ein Protokoll übertragen und von der EKI empfangen werden.
2. Die Daten werden strukturiert in einem Datenspeicher abgelegt.
3. Aus einem KRL-Programm heraus wird strukturiert auf die Daten zugegriffen. Mithilfe von KRL-Anweisungen werden die Daten gelesen und in KRL-Variablen kopiert.

Datenversand

Prinzipieller Ablauf (Grün markiert) (>>> Abb. 2-1):

1. Mithilfe von KRL-Anweisungen werden die Daten strukturiert in einen Datenspeicher geschrieben.
2. Mit einer KRL-Anweisung werden die Daten aus dem Speicher gelesen.
3. EKI sendet die Daten über ein Protokoll an das externe System.

Es ist möglich Daten direkt zu versenden, ohne dass die Daten zuvor in einem Speicher abgelegt werden.

2.4 Datenspeicherung

Beschreibung

Alle empfangenen Daten werden automatisch gespeichert und stehen damit KRL zur Verfügung. Beim Speichern werden XML- und Binär-Daten unterschiedlich behandelt.

Jeder Datenspeicher ist als Stapelspeicher realisiert. Die einzelnen Speicher werden im FIFO- oder LIFO-Modus ausgelesen.

XML-Daten

Die empfangenen Daten werden extrahiert und typrichtig in verschiedene Speicher abgelegt (pro Wert ein Speicher).

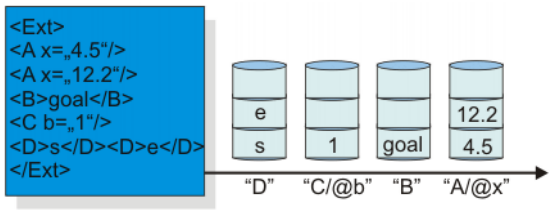


Abb. 2-2: XML-Daten-Speicher

Binär-Daten

Die empfangenen Daten werden nicht extrahiert oder interpretiert. Für eine Verbindung im Binär-Modus existiert nur ein Speicher.

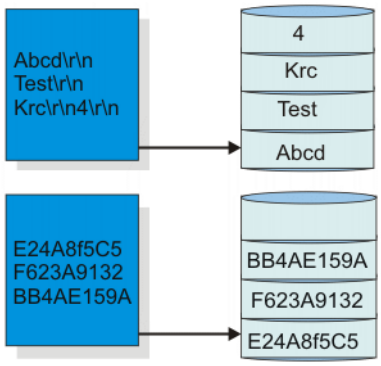


Abb. 2-3: Binäre Daten-Speicher

Ausleseverfahren

Datenelemente werden in der Reihenfolge aus dem Speicher entnommen, in der sie dort abgelegt wurden (FIFO). Das umgekehrte Verfahren, mit dem das zuletzt im Speicher abgelegte Datenelement zuerst entnommen wird, ist konfigurierbar (LIFO).

Jedem Speicher wird ein gemeinsames Limit maximal speicherbarer Daten zugeteilt. Wird das Limit überschritten, wird die Ethernet-Verbindung sofort geschlossen, um den Empfang weiterer Daten zu verhindern. Die aktuell empfangenen Daten werden noch gespeichert, d. h. der Speicher wird um 1 erhöht. Die Speicher können weiter abgearbeitet werden. Die Verbindung kann über die Funktion EKI_OPEN() wieder geöffnet werden.

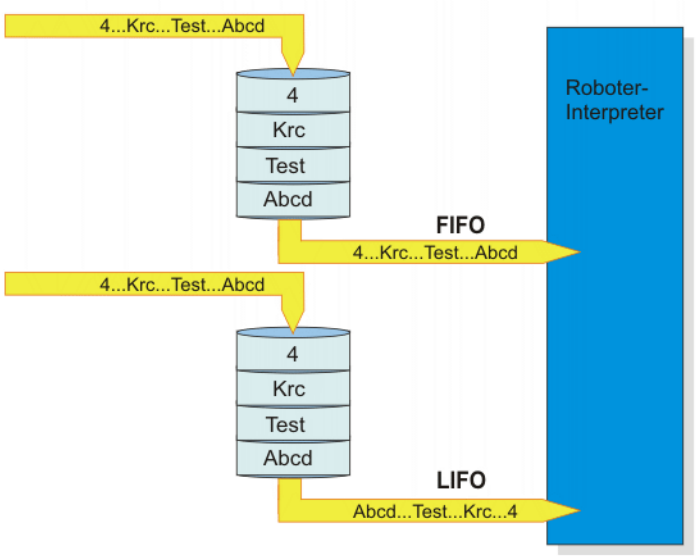


Abb. 2-4: Ausleseverfahren Übersicht

2.5 Client-Server-Betrieb

Beschreibung

Robotersteuerung und externes System verbinden sich als Client und Server. Dabei kann das externe System Client oder Server sein. Die Anzahl aktiver Verbindungen ist auf 16 beschränkt.

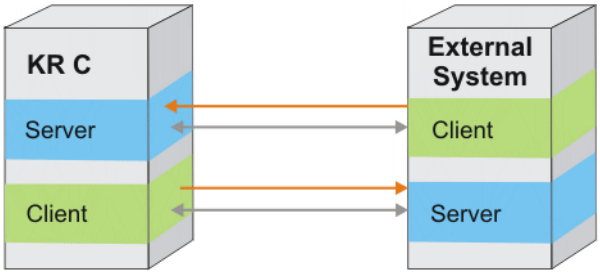


Abb. 2-5: Client-Server-Betrieb

Wenn die EKI als Server konfiguriert wird, kann sich nur ein einzelner Client mit dem Server verbinden. Werden mehrere Verbindungen benötigt, sind auch mehrere Server in der Schnittstelle anzulegen. Es ist möglich mehrere Clients und Server gleichzeitig innerhalb der EKI zu betreiben.

2.6 Protokollarten

Beschreibung

Die übertragenen Daten können in verschiedene Formate verpackt werden.

Folgende Formate werden unterstützt:

- XML-Struktur frei konfigurierbar
- Binär-Datensatz mit fester Länge
- Binär-Datensatz variabel mit Endzeichenfolge

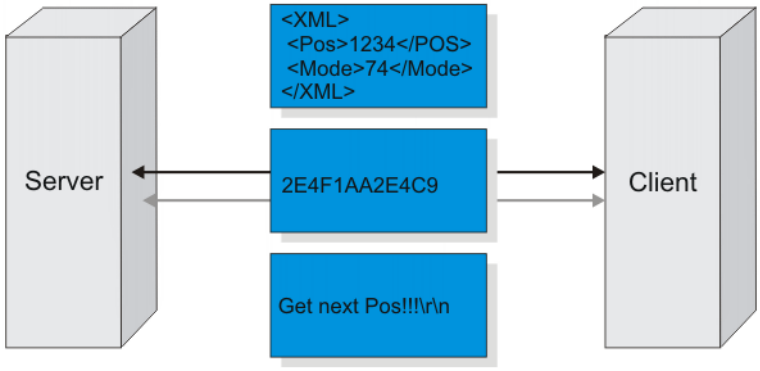


Abb. 2-6: Protokollarten

Die beiden Binär-Varianten können nicht gleichzeitig auf einer Verbindung betrieben werden.

Folgende Kombinationen sind möglich:

Verbindung V*x*	V1	V2	V3	V4	V5
Binär fest	✓	✗	✓	✗	✗
Binär variabel	✗	✗	✗	✓	✓
XML	✓	✓	✗	✗	✓

Beispiele

Abb. 2-7: Binär-Daten mit fester Länge (10 Byte)

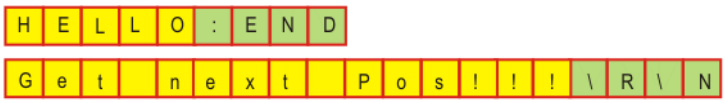


Abb. 2-8: Binär-Daten variabel mit Endzeichenfolge

2.7 Ereignismeldungen

Beschreibung

Über das Setzen eines Ausgangs oder Flags können folgende Ereignisse gemeldet werden:

- Verbindung ist aktiv.
- Ein einzelnes XML-Element ist an der Schnittstelle angekommen.
- Eine vollständige XML-Struktur oder ein vollständiger Binär-Datensatz ist an der Schnittstelle angekommen.

(>>> 6.2.8 "Konfigurieren von Ereignismeldungen" Seite 37)

2.8 Fehlerbehandlung

Beschreibung

Für den Datenaustausch zwischen Robotersteuerung und externem System stellt EthernetKRL Funktionen zur Verfügung.

Jede dieser Funktionen gibt Werte zurück. Diese Rückgabewerte können im KRL-Programm abgefragt und ausgewertet werden.

Abhängig von der Funktion werden folgende Werte zurückgegeben:

- Fehlernummer

- Anzahl der Elemente, die sich nach dem Zugriff noch im Speicher befinden
- Anzahl der Elemente, die aus dem Speicher gelesen wurden
- Information, ob eine Verbindung besteht
- Zeitstempel des Datenelements, das aus dem Speicher entnommen wurde

(>>> 6.2.7 "EKI_STATUS – Struktur für die funktionsspezifischen Rückgabewerte" Seite 36)

Zu jedem Fehler wird eine Meldung auf der smartHMI und im EKI-Logbuch ausgegeben. Die automatische Ausgabe von Meldungen kann deaktiviert werden.

3 Sicherheit

Diese Dokumentation enthält Sicherheitshinweise, die sich spezifisch auf die hier beschriebene Software beziehen.

Die grundlegenden Sicherheitsinformationen zum Industrieroboter sind im Kapitel "Sicherheit" der Bedien- und Programmieranleitung für Systemintegratoren oder der Bedien- und Programmieranleitung für Endanwender zu finden.

Das Kapitel "Sicherheit" in der Bedien- und Programmieranleitung muss beachtet werden. Tod von Personen, schwere Verletzungen oder erhebliche Sachschäden können sonst die Folge sein.

4 Installation

4.1 Systemvoraussetzungen

Hardware
- Robotersteuerung KR C4
- Externes System

Software
- KUKA System Software 8.2 oder 8.3

4.2 EthernetKRL installieren oder updaten

Es wird empfohlen, vor dem Update einer Software alle zugehörigen Daten zu archivieren.

Vorbereitung
- Die Software von der CD auf einen USB-Stick kopieren.

  Die Software muss so auf den Stick kopiert werden, dass die Datei Setup.exe in der obersten Ebene liegt. (Also nicht in einem Ordner.)

HINWEIS Empfehlung: Immer KUKA-Sticks verwenden. Wenn Sticks von anderen Herstellern verwendet werden, können Daten verloren gehen.

Voraussetzung
- Benutzergruppe Experte

Vorgehensweise

1. Den USB-Stick an der Robotersteuerung oder am smartPAD anstecken.
2. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware** wählen.
3. Auf **Neue Software** drücken: In der Spalte **Name** muss der Eintrag **EthernetKRL** angezeigt werden und in der Spalte **Pfad** das Laufwerk **E:** oder **K:**.

   Wenn nicht, auf **Aktualisieren** drücken.
4. Wenn die genannten Einträge jetzt angezeigt werden, weiter mit Schritt 5.

   Wenn nicht, muss erst das Laufwerk, von dem installiert werden soll, konfiguriert werden:
   - Auf die Schaltfläche **Konfiguration** drücken. Ein neues Fenster öffnet sich.
   - Im Bereich **Installationspfade für Optionen** eine Zeile markieren.

     Hinweis: Wenn die Zeile bereits einen Pfad enthält, wird dieser überschrieben.
   - Auf **Pfadauswahl** drücken. Die vorhandenen Laufwerke werden angezeigt.
   - **E:** markieren. (Wenn Stick an der Robotersteuerung gesteckt.)

     Oder **K:** markieren. (Wenn Stick am smartPAD gesteckt.)
   - **Speichern** drücken. Das Fenster schließt sich wieder.

   Das Laufwerk muss nur einmal konfiguriert werden und bleibt für weitere Installationen gespeichert.
5. Den Eintrag **EthernetKRL** markieren und auf **Installieren** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
6. Die Aufforderung zum Neustart mit **OK** bestätigen.
7. Den Stick abziehen.
8. Die Robotersteuerung neu starten.

LOG-Datei Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

4.3 EthernetKRL deinstallieren

Es wird empfohlen, vor der Deinstallation einer Software alle zugehörigen Daten zu archivieren.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware** wählen. Alle installierten Zusatzprogramme werden angezeigt.
2. Den Eintrag **EthernetKRL** markieren und auf **Deinstallieren** drücken. Sicherheitsabfrage mit **Ja** beantworten. Die Deinstallation wird vorbereitet.
3. Die Robotersteuerung neu starten. Die Deinstallation wird fortgesetzt und abgeschlossen.

LOG-Datei

Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

5 Konfiguration

5.1 Netzwerkverbindung über das KLI der Robotersteuerung

Beschreibung

Für den Datenaustausch über Ethernet muss eine Netzwerkverbindung über das KLI der Robotersteuerung hergestellt werden.

> Detaillierte Informationen zur Netzwerk-Konfiguration über das KUKA Line Interface (KLI) der Robotersteuerung sind in der Bedien- und Programmieranleitung für Systemintegratoren zu finden.

Je nach Spezifikation stehen an der Kundenschnittstelle der Robotersteuerung folgende Ethernet-Schnittstellen als Option zur Verfügung:

- Schnittstelle X66 (1 Steckplatz)
- Schnittstelle X67.1-3 (3 Steckplätze)

> Weitere Informationen zu den Ethernet-Schnittstellen sind in der Betriebs- oder Montageanleitung der Robotersteuerung zu finden.

6 Programmierung

6.1 Ethernet-Verbindung konfigurieren

Übersicht

Eine Ethernet-Verbindung wird über eine XML-Datei konfiguriert. Für jede Verbindung muss im Verzeichnis C:\KRC\ROBOTER\Config\User\Common\EthernetKRL der Robotersteuerung eine Konfigurationsdatei definiert sein.

XML-Dateien sind "case sensitive". Die Groß-/Kleinschreibung muss beachtet werden!

Der Name der XML-Datei ist gleichzeitig der Zugriffschlüssel in KRL.

Beispiel: …\EXT.XML —> EKI_INIT("EXT")

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

Abschnitt	Beschreibung
<CONFIGURATION> … </CONFIGURATION>	Konfiguration der Verbindungsparameter zwischen externem System und Schnittstelle (>>> 6.1.1 "XML-Struktur für Verbindungseigenschaften" Seite 21)
<RECEIVE> … </RECEIVE>	Konfiguration der Empfangsstruktur, die von der Robotersteuerung empfangen wird (>>> 6.1.2 "XML-Struktur für den Datenempfang" Seite 24)
<SEND> … </SEND>	Konfiguration der Sendestruktur, die von der Robotersteuerung gesendet wird (>>> 6.1.3 "XML-Struktur für den Datenversand" Seite 26)

6.1.1 XML-Struktur für Verbindungseigenschaften

Beschreibung

Im Abschnitt <EXTERNAL> … </EXTERNAL> werden die Einstellungen für das externe System definiert:

Element	Beschreibung
TYPE	Legt fest, ob das externe System als Server oder als Client mit der Schnittstelle kommuniziert (optional) ■ **Server**: Externes System ist ein Server. ■ **Client**: Externes System ist ein Client. Default-Wert: **Server**
IP	IP-Adresse des externen Systems, wenn dieses als Server definiert ist (TYPE = Server) Wenn TYPE = Client, wird die IP-Adresse ignoriert.
PORT	Port-Nummer des externen Systems, wenn dieses als Server definiert ist (TYPE = Server) ■ **1 … 65 534** Wenn TYPE = Client, wird die Port-Nummer ignoriert.

Im Abschnitt <INTERNAL> … </INTERNAL> werden die Einstellungen für die Schnittstelle definiert:

Element	Attribut	Beschreibung

ENVIRONMENT	———	Löschen der Verbindung an Aktionen koppeln (optional)
		■ **Program**: Löschen nach Aktionen des Roboter-Interpreters
		  ■ Programm zurücksetzen.
		  ■ Programm abwählen.
		  ■ E/As rekonfigurieren.
		■ **Submit**: Löschen nach Aktionen des Submit-Interpreters
		  ■ Submit-Interpreter abwählen.
		  ■ E/As rekonfigurieren.
		■ **System**: Löschen nach Systemaktionen
		  ■ E/As rekonfigurieren.
		Default-Wert: **Program**

BUFFERING	Mode	Verfahren, nach dem alle Datenspeicher abgearbeitet werden (optional) ■ **FIFO**: First In First Out ■ **LIFO**: Last In First Out Default-Wert: **FIFO**
	Limit	Maximale Anzahl an Datenelementen, die ein Datenspeicher aufnehmen kann (optional) ■ **1 … 512** Default-Wert: **16**
BUFFSIZE	Limit	Maximale Anzahl an Bytes, die empfangen werden können, ohne dass sie interpretiert werden (optional) ■ **1 … 65 534 Bytes** Default-Wert: **16 384 Bytes**

Element	Attribut	Beschreibung
TIMEOUT	Connect	Zeit, nach der der Versuch eine Verbindung aufzubauen abgebrochen wird (optional) Einheit: ms ■ **0 … 65 534 ms** Default-Wert: **2 000 ms**
ALIVE	Set_Out Set_Flag	Setzen eines Ausgangs oder eines Flags bei erfolgreicher Verbindung (optional) Nummer des Ausgangs: ■ **1 … 4 096** Nummer des Flags: ■ **1 … 1 025** Solange eine Verbindung zum externen System aktiv ist, ist der Ausgang oder das Flag gesetzt. Wenn die Verbindung zum externen System abbricht, wird der Ausgang oder das Flag gelöscht.
	Ping	Intervall für das Senden eines Pings, um die Verbindung zum externen System zu überwachen (optional) ■ **1 … 65 534 s**
IP	———	IP-Adresse der EKI, wenn diese als Server definiert ist (EXTERNAL/TYPE = Client) Wenn EXTERNAL/TYPE = Server, wird die IP-Adresse ignoriert.
PORT	———	Port-Nummer der EKI, wenn diese als Server definiert ist (EXTERNAL/TYPE = Client) ■ **54 600 … 54 615** Wenn EXTERNAL/TYPE = Server, wird die Port-Nummer ignoriert.
PROTOCOL	———	Übertragungsprotokoll (optional) ■ **TCP** ■ **UPD** Default-Wert: **TCP** Es wird empfohlen, immer das TCP/IP-Protokoll zu verwenden.

Element	Attribut	Beschreibung
Messages	Display	Meldungsausgabe auf smartHMI deaktivieren (optional). ■ **error**: Meldungsausgabe ist aktiv. ■ **disabled**: Meldungsausgabe ist deaktiviert. Default-Wert: **error**
	Logging	Schreiben von Meldungen in EKI-Logbuch deaktivieren (optional). ■ **warning**: Warnmeldungen und Fehlermeldungen werden geloggt. ■ **error**: Nur Fehlermeldungen werden geloggt. ■ **disabled**: Logging ist deaktiviert. Default-Wert: **error**
	(>>> 9.3 "Meldungsausgabe und Loggen von Meldungen deaktivieren" Seite 60)	

Beispiel

```
<CONFIGURATION>
   <EXTERNAL>
     <IP>172.1.10.5</IP>
     <PORT>60000</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>54600</PORT>
     <PROTOCOL>TCP</PROTOCOL>
     <Messages Display="disabled" Logging="error"/>
   </INTERNAL>
</CONFIGURATION>
```

6.1.2 XML-Struktur für den Datenempfang

Beschreibung

Die Konfiguration ist abhängig davon, ob XML-Daten oder Binär-Daten empfangen werden.

- Für den Empfang von XML-Daten muss eine XML-Struktur definiert werden: <XML> … </XML>
- Für den Empfang von Binär-Daten müssen Rohdaten definiert werden: <RAW> … </RAW>

Attribute in den Elementen der XML-Struktur <XML> … </XML>:

Element	Attribut	Beschreibung
ELEMENT	Tag	Name des Elements Hier wird die XML-Struktur für den Datenempfang definiert (XPath). (>>> 6.1.4 "Konfiguration nach XPath-Schema" Seite 27)
ELEMENT	Type	Datentyp des Elements ■ **STRING** ■ **REAL** ■ **INT** ■ **BOOL** ■ **FRAME** **Hinweis**: Optional, wenn das Tag nur für Ereignismeldungen verwendet wird. In diesem Fall wird kein Speicherplatz für das Element reserviert. **Beispiel für Ereignis-Flag**: <ELEMENT Tag="Ext" Set_Flag="56"/>
ELEMENT	Set_Out Set_Flag	Setzen eines Ausgangs oder eines Flags, wenn das Element empfangen wurde (optional) Nummer des Ausgangs: ■ **1 … 4 096** Nummer des Flags: ■ **1 … 1 025**
ELEMENT	Mode	Verfahren, nach dem ein Datensatz im Datenspeicher abgearbeitet wird ■ **FIFO**: First In First Out ■ **LIFO**: Last In First Out Nur relevant, wenn einzelne Datensätze anders behandelt werden sollen wie unter BUFFERING für die Schnittstelle konfiguriert.

Attribute für das Element in den Rohdaten <RAW> … </RAW>:

Element	Attribut	Beschreibung
ELEMENT	Tag	Name des Elements
ELEMENT	Type	Datentyp des Elements ■ **BYTE**: Binär-Datensatz mit fester Länge ■ **STREAM**: Binär-Datensatz variabel mit Endzeichenfolge
ELEMENT	Set_Out Set_Flag	Setzen eines Ausgangs oder eines Flags, wenn das Element empfangen wurde (optional) Nummer des Ausgangs: ■ **1 … 4 096** Nummer des Flags: ■ **1 … 1 025**

Element	Attribut	Beschreibung
ELEMENT	EOS	Endzeichenfolge einer elementaren Information (nur relevant, wenn TYPE = STREAM) ■ ASCII-Kodierung: **1 … 32 Zeichen** ■ Alternatives Ende wird durch das Zeichen "|" getrennt. Beispiele: ■ <ELEMENT … EOS="123,134,21"/> ■ <ELEMENT … EOS="123,134,21|13,10"/>
ELEMENT	Size	Feste Größe der Information, wenn TYPE = BYTE ■ **1 … 3 600 Bytes** Maximale Größe der Information, wenn TYPE = STREAM ■ **1 … 3 600 Bytes**

Beispiele

```
<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-Struktur für den Datenversand

Beschreibung

Die Konfiguration ist abhängig davon, ob XML-Daten oder Binär-Daten gesendet werden.

- Für das Senden von XML-Daten muss eine XML-Struktur definiert werden: <XML> … </XML>

Die XML-Struktur wird beim Senden in der Reihenfolge angelegt, in der sie konfiguriert ist.

- Das Senden von Binär-Daten wird direkt in der KRL-Programmierung realisiert. Es muss keine Konfiguration angegeben werden.

Attribut in den Elementen der XML-Struktur <XML> … </XML>:

Attribut	Beschreibung
Tag	Name des Elements Hier wird die XML-Struktur für den Datenversand definiert (XPath).

Beispiel

```
<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 Konfiguration nach XPath-Schema

Bescheibung

Wenn XML verwendet wird um Daten auszutauschen, ist es notwendig, dass die ausgetauschten XML-Dokumente nach demselben Schema aufgebaut sind. Zum Beschreiben und Lesen der XML-Dokumente verwendet Ethernet-KRL das XPath-Schema.

Nach XPath sind folgende Fälle zu unterscheiden:

- Beschreiben und Lesen von Elementen
- Beschreiben und Lesen von Attributen

Element-Schreibweise

- Gespeichertes XML-Dokument für den Datenversand:

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

- Konfigurierte XML-Struktur für den Datenversand:

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

Attribut-Schreibweise

- Gespeichertes XML-Dokument für den Datenversand:

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

- Konfigurierte XML-Struktur für den Datenversand:

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

6.2 Funktionen für den Datenaustausch

Übersicht

Für den Datenaustausch zwischen Robotersteuerung und externem System stellt EthernetKRL Funktionen zur Verfügung.

Die genaue Beschreibung der Funktionen ist im Anhang zu finden.
(>>> 9.4 "Befehlsreferenz" Seite 60)

Verbindung initialisieren, öffnen, schließen und löschen
EKI_STATUS = EKI_Init(CHAR[])
EKI_STATUS = EKI_Open(CHAR[])
EKI_STATUS = EKI_Close(CHAR[])
EKI_STATUS = EKI_Clear(CHAR[])

Daten senden
EKI_STATUS = EKI_Send(CHAR[], CHAR[])

Daten schreiben
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[])

Daten auslesen
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[])

Funktion auf Fehler prüfen
EKI_CHECK(EKI_STATUS, EKrlMsgType, CHAR[])

Speicher löschen, sperren, entsperren und prüfen
EKI_STATUS = EKI_ClearBuffer(CHAR[], CHAR[])
EKI_STATUS = EKI_Lock(CHAR[])
EKI_STATUS = EKI_Unlock(CHAR[])
EKI_STATUS = EKI_CheckBuffer(CHAR[], CHAR[])

6.2.1 Programmiertipps

- Wenn eine Verbindung im Submit-Interpreter angelegt wird, sind folgende Punkte zu beachten:
  - In der Verbindungskonfiguration muss über das Element <ENVIRONMENT> angegeben werden, dass es sich um einen Submit-Kanal handelt.
  - Ein im Submit-Interpreter geöffneter Kanal kann auch vom Roboter-Interpreter angesprochen werden.
  - Wird der Submit-Interpreter abgewählt, wird die Verbindung per Konfiguration automatisch gelöscht.

EKI-Anweisungen werden im Vorlauf ausgeführt!

- Wenn eine EKI-Anweisung im Hauptlauf ausgeführt werden soll, müssen Anweisungen verwendet werden, die einen Vorlaufstopp auslösen, z. B. WAIT SEC.
- Da jeder Zugriff auf die Schnittstelle Zeit kostet, wird empfohlen große Datenmengen mit den Feld-Zugriffsfunktionen EKI_Get…Array() abzurufen.
- EthernetKRL kann über EKI_Get…Array() auf maximal 512 Feld-Elemente zugreifen. Es ist möglich in KRL ein größeres Feld anzulegen, z. B. myFrame[1000], es können aber immer nur maximal 512 Elemente gelesen werden.
- Es gibt verschiedene Möglichkeiten auf Daten zu warten:
  - Über das Setzen eines Flags oder eines Ausgangs kann angezeigt werden, dass ein bestimmtes Datenelement oder ein vollständiger Datensatz empfangen wurde (Element Set_Flag oder Set_Out in der XML-Struktur für den Datenempfang).

    Beispiel: Interrupt des KRL-Programms über WAIT FOR $FLAG[*x*] (>>> 7.2.5 "Beispielkonfiguration XmlCallback" Seite 49)
  - Über die Funktion EKI_CheckBuffer() kann zyklisch abgefragt werden, ob neue Datenelemente im Speicher vorhanden sind.

6.2.2 Initialisieren und Löschen einer Verbindung

Beschreibung

Eine Verbindung muss mit der Funktion EKI_Init() angelegt und initialisiert werden. Die in der Funktion angegebene Verbindungskonfiguration wird dabei eingelesen. Eine Verbindung kann sowohl im Roboter- als auch im Submit-Interpreter angelegt werden.

Eine Verbindung kann im Roboter- oder Submit-Interpreter über die Funktion EKI_Clear() wieder gelöscht werden. Zusätzlich kann das Löschen einer Verbindung an Roboter- und Submit-Interpreter-Aktionen oder Systemaktionen gekoppelt sein. (Konfigurierbar über das Element <ENVIRONMENT> in der Verbindungskonfiguration)

Konfiguration "Program"

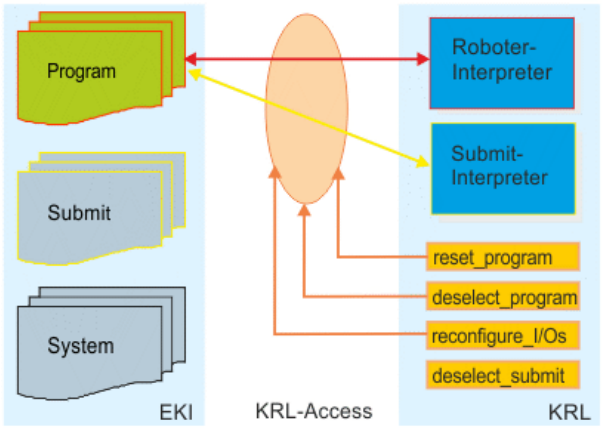


Abb. 6-1: Verbindungskonfiguration "Program"

Bei dieser Konfiguration wird eine Verbindung nach folgenden Aktionen gelöscht.

- Programm zurücksetzen.
- Programm abwählen.
- E/As rekonfigurieren.

Beim Rekonfigurieren der E/As wird der Treiber neu geladen, d. h. alle Initialisierungen werden gelöscht.

Konfiguration "Submit"

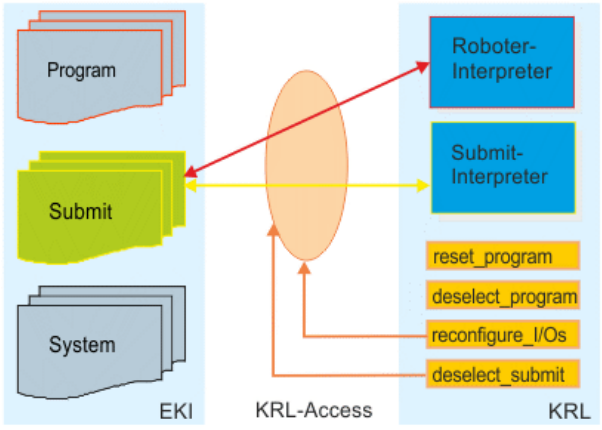


Abb. 6-2: Verbindungskonfiguration "Submit"

Bei dieser Konfiguration wird eine Verbindung nach folgenden Aktionen gelöscht.

- Submit-Interpreter abwählen.
- E/As rekonfigurieren.

Beim Rekonfigurieren der E/As wird der Treiber neu geladen, d. h. alle Initialisierungen werden gelöscht.

Konfiguration "System"

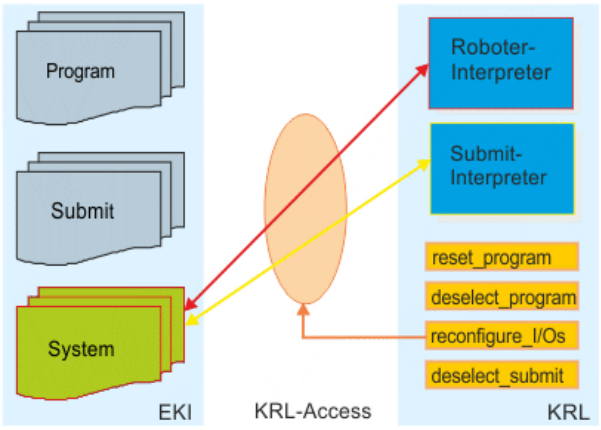


Abb. 6-3: Verbindungskonfiguration "System"

Bei dieser Konfiguration wird eine Verbindung nach folgenden Aktionen gelöscht.

- E/As rekonfigurieren.

> Beim Rekonfigurieren der E/As wird der Treiber neu geladen, d. h. alle Initialisierungen werden gelöscht.

6.2.3 Öffnen und Schließen einer Verbindung

Beschreibung

Die Verbindung zum externen System wird über ein KRL-Programm hergestellt. Die meisten KRL-Programme sind wie folgt aufgebaut:

```
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
```

Zeile	Beschreibung
2	EKI_Init() initialisiert den Kanal, über den sich die Schnittstelle mit dem externen System verbindet.
3	EKI_Open() öffnet den Kanal.
4	KRL-Anweisungen, um Daten in den Speicher zu schreiben, Daten zu senden oder auf empfangene Daten zuzugreifen
5	EKI_Close() schließt den Kanal.
6	EKI_Clear () löscht den Kanal.

Bei der Programmierung ist zu beachten, ob die Schnittstelle als Server oder als Client konfiguriert ist.

Server-Betrieb

Wenn das externe System als Client konfiguriert ist, versetzt EKI_Open() die Schnittstelle (= Server) in einen Abhörzustand. Der Server wartet auf die Verbindungsanfrage eines Clients, ohne dass der Programmablauf unterbrochen wird. Wenn in der Konfigurationsdatei das Element <TIMEOUT Connect="…"/> nicht beschrieben wird, wartet der Server solange bis ein Client eine Verbindung anfordert.

Eine Verbindungsanfrage durch einen Client wird durch Zugriff auf die Schnittstelle oder durch eine Ereignismeldung signalisiert, z. B. über das Element <ALIVE SET_OUT="…"/>.

Wenn der Programmablauf unterbrochen werden soll solange der Server die Verbindungsanfrage erwartet, muss ein Ereignis-Flag oder -Ausgang programmiert werden, z. B. WAIT FOR $OUT[…].

Es wird empfohlen, EKI_Close() im Server-Betrieb nicht zu verwenden. Im Server-Betrieb wird der Kanal vom externen Client aus geschlossen.

Client-Betrieb

Wenn das externe System als Server konfiguriert ist, unterbricht EKI_Open() den Programmablauf bis die Verbindung zum externen System aktiv ist. EKI_Close() schließt die Verbindung zum externen Server.

6.2.4 Senden von Daten

Beschreibung

Je nach Konfiguration und Programmierung können folgende Daten mit EKI_Send() gesendet werden:

- Vollständige XML-Struktur
- Partielle XML-Struktur
- XML-Daten direkt als Zeichenkette
- Binär-Datensatz mit Endzeichenfolge (EOS) direkt als Zeichenkette
- Binär-Datensatz fester Länge direkt als Zeichenkette

  Binär-Datensätze fester Länge müssen im KRL-Programm mit CAST_TO() eingelesen werden. Es sind nur Daten vom Typ REAL (4 Bytes) lesbar, kein Double.

Detaillierte Informationen zum Befehl CAST_TO() sind in der Dokumentation CREAD/CWRITE zu finden.

Beispiel XML-Daten

Senden der vollständigen XML-Struktur

- Gespeicherte XML-Struktur für den Datenversand:

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

- Programmierung:

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

- Gesendete XML-Struktur:

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

Senden eines Teils der XML-Struktur

- Gespeicherte XML-Struktur für den Datenversand (wird beim direkten Senden nicht verwendet):

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

- Programmierung:

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

- Gesendete XML-Struktur:

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

Direktes Senden der XML-Daten als Zeichenkette

- Gespeicherte XML-Struktur für den Datenversand:

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

- Programmierung:

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

- Gesendete Zeichenkette:

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

Beispiel Binär-Daten

Direktes Senden eines Binär-Datensatzes fester Länge (10 Byte)

- Konfigurierte Rohdaten:

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

- Programmierung:

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

- Gesendete Daten:

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

Direktes Senden eines Binär-Datensatzes mit Endzeichenfolge

- Konfigurierte Rohdaten:

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

- Programmierung:

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

- Gesendete Daten:

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

6.2.5 Auslesen von Daten

Zum Auslesen von Daten müssen die zugehörigen KRL-Variablen initialisiert sein, z. B. durch die Zuweisung von Werten.

Beschreibung

Beim Speichern und Auslesen der Daten werden XML- und Binär-Daten unterschiedlich behandelt:

- XML-Daten werden von der EKI extrahiert und typrichtig in verschiedene Speicher abgelegt. Es ist möglich auf jeden gespeicherten Wert einzeln zuzugreifen.

  Um XML-Daten auszulesen können alle Zugriffsfunktionen EKI_Get…() verwendet werden.
- Binär-Datensätze werden von der EKI nicht interpretiert und als Ganzes in einem Speicher abgelegt.

  Um einen Binär-Datensatz aus einem Speicher zu lesen muss die Zugriffsfunktion EKI_GetString() verwendet werden. Binär-Datensätze werden als Zeichenfolgen aus dem Speicher gelesen.

  Binär-Datensätze fester Länge müssen im KRL-Programm mit CAST_FROM() wieder in einzelne Variablen aufgeteilt werden.

Detaillierte Informationen zum Befehl CAST_FROM() sind in der Dokumentation CREAD/CWRITE zu finden.

Beispiel XML-Daten

Gespeicherte XML-Struktur für den Datenempfang:

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

Programmierung:

```
; 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)
```

Empfangene Daten:

```
valueChar[] "Example message"
valueBOOL[] TRUE
```

Beispiel Binär-Daten

Auslesen eines Binär-Datensatzes fester Länge (10 Byte)

- Konfigurierte Rohdaten:

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

- Programmierung:

```
; 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)
```

Auslesen eines Binär-Datensatzes mit Endzeichenfolge

- Konfigurierte Rohdaten:

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

- Programmierung:

```
; 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 Löschen empfangener Daten

Beschreibung

Beim Löschen empfangener Daten sind folgende Fälle zu unterscheiden:

- Löschen über EKI_Clear(): Die Ethernet-Verbindung wird beendet und alle Speicher, die die Verbindung verwendet, werden gelöscht.
- Löschen über EKI_ClearBuffer(): Löscht empfangene noch nicht abgerufene Daten aus einem Speicher oder aus allen Speichern.

> XML-Daten werden von der EKI extrahiert und typrichtig in verschiedene Speicher abgelegt. Beim Löschen einzelner Speicher muss sichergestellt sein, dass keine zusammengehörigen Daten verlorengehen.

Beispiele

- Die Position des zu löschenden Speichers wird in XPATH angegeben. Alle Elemente nach <Root><Activ><Flag>... werden gelöscht.

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

- Alle Speicher des Elements <Root>…</Root> werden gelöscht.

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

6.2.7 EKI_STATUS – Struktur für die funktionsspezifischen Rückgabewerte

Beschreibung

Jede EthernetKRL-Funktion gibt funktionsspezifische Werte zurück. EKI_STATUS ist die globale Strukturvariable, in die diese Werte geschrieben werden.

Syntax

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

Erläuterung der Syntax

Element	Beschreibung
Buff	Anzahl der Elemente, die sich nach dem Zugriff noch im Speicher befinden
Read	Anzahl der Elemente, die aus dem Speicher gelesen wurden
Msg_No	Fehlernummer des Fehlers, der beim Aufruf einer Funktion oder beim Datenempfang aufgetreten ist Wenn die automatische Meldungsausgabe deaktiviert wurde, kann mit EKI_CHECK() die Fehlernummer ausgelesen und die Fehlermeldung auf der smartHMI ausgegeben werden.
Connected	Gibt an, ob eine Verbindung besteht ■ TRUE = Verbindung vorhanden ■ FALSE = Verbindung unterbrochen
Counter	Zeitstempel für empfangene Datenpakete Im Speicher eintreffende Datenpakete werden fortlaufend nummeriert, und zwar in der Reihenfolge, in der sie im Speicher abgelegt werden. Wenn einzelne Daten gelesen werden, wird das Strukturelement Counter mit dem Zeitstempel des Datenpakets belegt, aus dem das Datenelement stammt. (>>> 6.2.10 "Verarbeiten unvollständiger Datensätze" Seite 38)

Rückgabewerte

Abhängig von der Funktion werden folgende Elemente der Struktur EKI_STATUS beschrieben:

Funktion	Buff	Read	Msg_No	Connected	Counter
EKI_Init()	✗	✗	✓	✗	✗
EKI_Open()	✗	✗	✓	✓	✗
EKI_Close()	✗	✗	✓	✓	✗
EKI_Clear()	✗	✗	✓	✓	✗
EKI_Send()	✗	✗	✓	✓	✗
EKI_Set…()	✗	✗	✓	✓	✗
EKI_Get…()	✓	✓	✓	✓	✓
EKI_ClearBuffer()	✗	✗	✓	✓	✗

Funktion	Buff	Read	Msg_No	Connected	Counter
EKI_Lock()	✗	✗	✓	✓	✗
EKI_Unock()	✗	✗	✓	✓	✗
EKI_CheckBuffer()	✓	✗	✓	✓	✓

6.2.8 Konfigurieren von Ereignismeldungen

Beschreibung

Über das Setzen eines Ausgangs oder Flags können folgende Ereignisse gemeldet werden:

- Verbindung ist aktiv.
- Ein einzelnes XML-Element ist an der Schnittstelle angekommen.
- Eine vollständige XML-Struktur oder ein vollständiger Binär-Datensatz ist an der Schnittstelle angekommen.

Ereignis-Ausgang

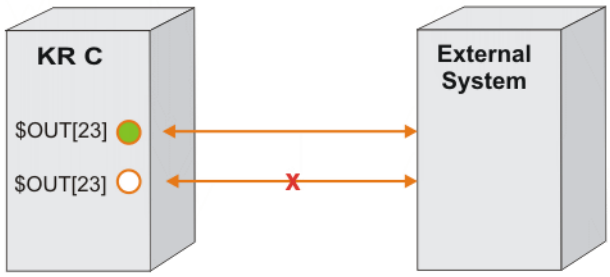


Abb. 6-4: Ereignis-Ausgang (Verbindung aktiv)

$OUT[23] ist gesetzt, solange die Verbindung zum externen System aktiv ist. Wenn die Verbindung nicht mehr aktiv ist, wird $OUT[23] zurückgesetzt.

> Die Verbindung kann nur mit der Funktion EKI_OPEN() wiederhergestellt werden.

Ereignis-Flag

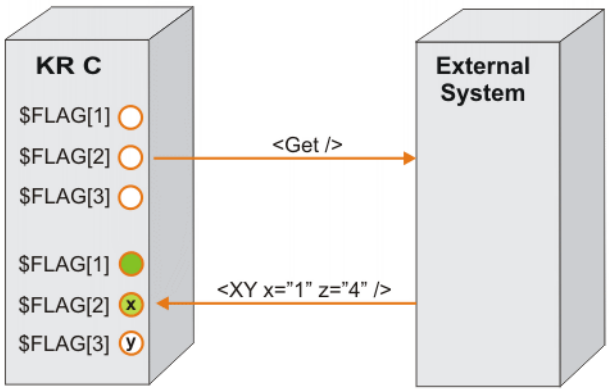


Abb. 6-5: Ereignis-Flag (Vollständige XML-Struktur)

Die XML-Struktur <XY /> enthält die Datenelemente "XY/@x" und "XY/@z". $FLAG[1] wird gesetzt, da die vollständige XML-Struktur an der Schnittstelle angekommen ist. $FLAG[2] wird gesetzt, da das Element "x" in "XY" enthalten ist. $FLAG[3] wird nicht gesetzt, da das Element "y" nicht übermittelt wurde.

Beispiel

6.2.9 Empfang vollständiger XML-Datensätze

Beschreibung

Die Zugriffsfunktionen EKI_Get...() sind solange gesperrt bis alle Daten eines XML-Datensatzes im Speicher liegen.

Wenn LIFO konfiguriert ist und 2 oder mehr XML-Datensätze direkt hintereinander an der Schnittstelle ankommen, ist nicht mehr sichergestellt, dass ein Datensatz zusammenhängend aus dem Speicher geholt wird. Es kann z. B. vorkommen, dass die Daten des zweiten Datensatzes bereits im Speicher abgelegt werden, obwohl der erste Datensatz noch nicht vollständig abgearbeitet ist. Da im LIFO-Betrieb immer zuerst auf die zuletzt gespeicherten Daten zugegriffen wird, ist der in KRL verfügbare Datensatz inkonsistent.

Um die Fragmentierung von Datensätzen im LIFO-Betrieb zu verhindern, muss die Verarbeitung neu empfangener Daten gesperrt werden bis alle zusammengehörigen Daten aus dem Speicher geholt wurden.

Beispiel

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

6.2.10 Verarbeiten unvollständiger Datensätze

Es kann vorkommen, dass ein externes System unvollständige Datensätze sendet. Einzelne XML-Elemente sind entweder leer oder fehlen ganz, so dass in einer Speicherschicht Daten aus verschiedenen Datenpaketen liegen.

Wenn die Datensätze zusammenhängend in KRL vorhanden sein müssen, kann das Strukturelement Counter der Variablen EKI_STATUS benutzt werden. Bei der Verwendung von EKI_Get…Array-Funktionen werden zeitlich nicht zusammenhängende Daten daran erkannt, dass Counter = 0 zurückgegeben wird.

6.2.11 EKI_CHECK() – Funktionen auf Fehler prüfen

Beschreibung

EthernetKRL gibt bei jedem Fehler eine Meldung auf der smartHMI aus. Die automatische Ausgabe von Meldungen kann deaktiviert werden.

(>>> 9.3 "Meldungsausgabe und Loggen von Meldungen deaktivieren" Seite 60)

Wenn die automatische Meldungsausgabe deaktiviert worden ist, wird empfohlen mit EKI_CHECK() zu prüfen, ob beim Ausführen einer EthernetKRL-Funktion ein Fehler aufgetreten ist:

- Die Fehlernummer wird ausgelesen und die zugehörige Meldung auf der smartHMI ausgegeben.
- Wird in EKI_CHECK() ein Kanalname angegeben, wird beim Datenempfang abgefragt, ob Fehler vorliegen.

(>>> 9.4.5 "Funktion auf Fehler prüfen" Seite 67)

Bei jedem Aufruf von EKI_CHECK() wird das Programm KRC:\R1\TP\EthernetKRL\EthernetKRL_USER.SRC aufgerufen. In diesem Programm können benutzerspezifische Fehlerreaktionen programmiert werden.

Beispiel

Eine Verbindung wird immer geschlossen, wenn ein Fehler im Empfang auftritt. Als Fehlerstrategie kann für den Fall, dass die Ethernet-Verbindung abbricht, ein Interrupt programmiert werden.

- In der Konfigurationsdatei XmlTransmit.XML ist definiert, dass bei erfolgreicher Verbindung FLAG[1] gesetzt wird. Bei Verbindungsverlust wird FLAG[1] zurückgesetzt.

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

- Im KRL-Programm wird der Interrupt deklariert und eingeschaltet. Wird FLAG[1] zurückgesetzt, wird das Interrupt-Programm ausgeführt.

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

- Im Interrupt-Programm wird mit EKI_CHECK() abgefragt, was für ein Fehler aufgetreten ist, und dann die Verbindung wieder geöffnet.

```
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 Beispiele

7.1 Beispielapplikationen

Übersicht

EthernetKRL beinhaltet Beispielapplikationen, mit denen eine Kommunikation zwischen einem Server-Programm und der Robotersteuerung hergestellt werden kann. Die Software befindet sich auf der ausgelieferten CD im Verzeichnis DOC\Example.

Die Software besteht aus folgenden Komponenten:

Komponente	Ordner
Server-Programm EthernetKRL_Server.exe	...\Application
Beispielprogramme in KRL ■ BinaryFixed.src ■ BinaryStream.src ■ XmlCallback.src ■ XmlServer.src ■ XmlTransmit.src	...\Program
Beispielkonfigurationen in XML ■ BinaryFixed.xml ■ BinaryStream.xml ■ XmlCallBack.xml ■ XmlServer.xml ■ XmlTransmit.xml ■ XmlFullConfig.xml	...\Config

7.1.1 Beispielapplikationen implementieren

Voraussetzung

Externes System:

- Windows-Betriebssystem mit installiertem .NET-Framework 3.5 oder höher

Robotersteuerung:

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

1. Server-Programm auf externes System kopieren.
2. Alle SRC-Dateien in das Verzeichnis C:\KRC\ROBOTER\Program der Robotersteuerung kopieren.
3. Alle XML-Dateien in das Verzeichnis C:\KRC\ROBOTER\Config\User\Common\EthernetKRL der Robotersteuerung kopieren.
4. Server-Programm auf dem externen System starten.
5. Menü-Button drücken. Das Fenster **Communication Properties** öffnet sich.
6. Nur wenn am externen System mehrere Netzwerk-Schnittstellen zur Verfügung stehen: Nummer des Netzwerkadapters (= Netzwerkkarten-Index) eingeben, der zur Kommunikation mit der Robotersteuerung genutzt wird.
7. Fenster **Communication Properties** schließen und Start-Button drücken. Die zur Kommunikation verfügbare IP-Adresse wird im Meldungsfenster angezeigt.
8. Angezeigte IP-Adresse des externen Systems in der gewünschten XML-Datei einstellen.

7.1.2 Bedienoberfläche Server-Programm

Beschreibung

Das Server-Programm ermöglicht es, die Kommunikation zwischen einem externen System und der Robotersteuerung zu testen, indem eine stabile Verbindung zur Robotersteuerung hergestellt wird.

Das Server-Programm enthält folgende Funktionalitäten:

- Senden und Empfangen von Daten (automatisch oder manuell)
- Anzeige der empfangenen Daten
- Anzeige der gesendeten Daten

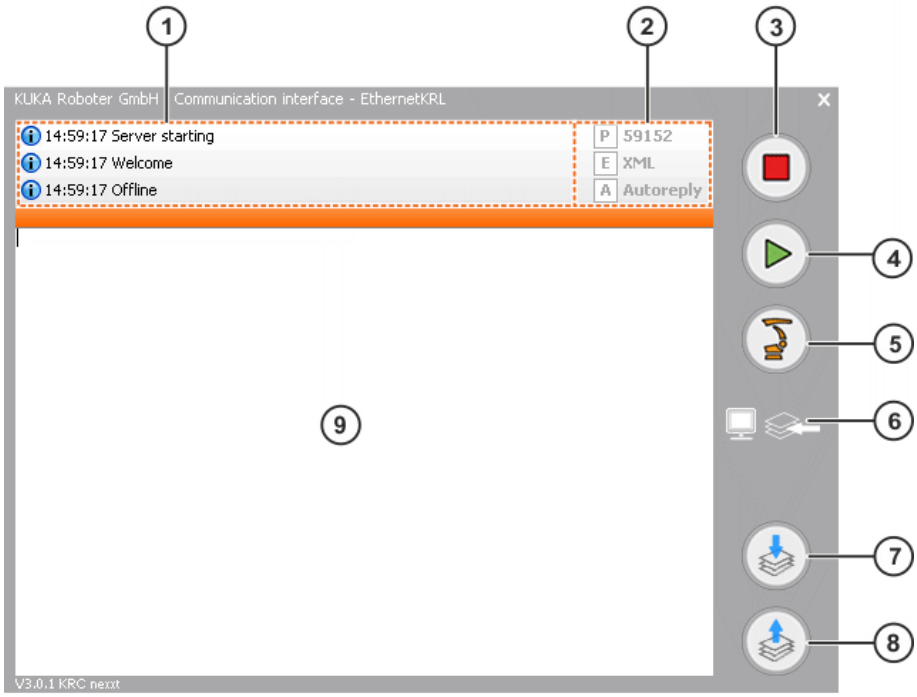


Abb. 7-1: Server-Programm Bedienoberfläche

Pos.	Beschreibung
1	Meldungsfenster

2	Anzeige der eingestellten Kommunikationsparameter
	(>>> 7.1.3 "Kommunikationsparameter im Server-Programm einstellen" Seite 43)
	■ **P**: Port-Nummer
	■ **E**: Beispieldaten
	  ■ **Xml**: XML-Daten
	  ■ **BinaryFixed**: Binär-Daten mit fester Länge
	  ■ **BinaryStream**: Binär-Datenstrom variabel mit Endzeichenfolge
	■ **A**: Kommunikationsmodus
	  ■ **Autoreply**: Der Server beantwortet jedes empfangene Datenpaket automatisch.
	  ■ **Manual**: Nur manueller Datenempfang oder Datenversand

3	Stopp-Button Die Kommunikation mit der Robotersteuerung wird beendet und der Server wird zurückgesetzt.

Pos.	Beschreibung
4	Start-Button Der Datenaustausch zwischen Server-Programm und Robotersteuerung wird ausgewertet. Die erste eingehende Verbindungsanfrage wird gebunden und als Kommunikationsadapter benutzt.
5	Menü-Button zum Einstellen der Kommunikationsparameter (>>> 7.1.3 "Kommunikationsparameter im Server-Programm einstellen" Seite 43)
6	Anzeigeoptionen ■ Pfeil zeigt nach links: Die empfangenen Daten werden angezeigt. (Default) ■ Pfeil zeigt nach rechts: Die gesendeten Daten werden angezeigt.
7	Button für den manuellen Datenempfang
8	Button für den manuellen Datenversand
9	Anzeigefenster Je nach eingestellter Anzeigeoption werden die gesendeten oder die empfangenen Daten angezeigt.

7.1.3 Kommunikationsparameter im Server-Programm einstellen

Vorgehensweise

1. Im Server-Programm auf den Menü-Button klicken.
   Das Fenster **Communication Properties** öffnet sich.
2. Kommunikationsparameter einstellen.
3. Fenster schließen.

Beschreibung

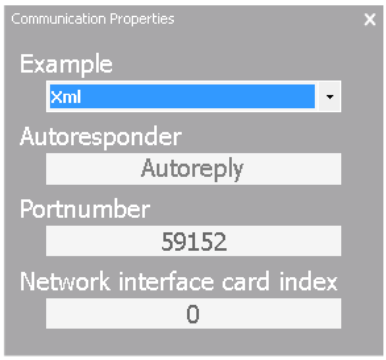


Abb. 7-2: Fenster Communication Properties

Element	Beschreibung
Example	Beispieldaten auswählen. ■ **Xml**: XML-Daten ■ **BinaryFixed**: Binär-Daten mit fester Länge ■ **BinaryStream**: Binär-Datenstrom variabel mit Endzeichenfolge Default-Wert: **xml**
Autoresponder	Kommunikationsmodus auswählen. ■ **Autoreply**: Der Server beantwortet jedes empfangene Datenpaket automatisch. ■ **Manual**: Nur manueller Datenempfang oder Datenversand Default-Wert: **Autoreply**
Portnumber	Port-Nummer der Socket-Verbindung eingeben. An diesem Port erwartet das externe System die Verbindungsanfrage der Robotersteuerung. Es muss eine freie Nummer gewählt werden, die nicht als Standarddienst belegt ist. Default-Wert: **59152**
Network interface card index	Nummer des Netzwerkadapters eingeben. Nur relevant, wenn das externe System mehrere Netzwerkkarten benutzt, z. B. WLAN und LAN. Default-Wert: **0**

7.2 Beispielkonfigurationen und -programme

7.2.1 Beispielkonfiguration BinaryFixed

Zur Kommunikation mit der Robotersteuerung müssen im Server-Programm die passenden Beispieldaten eingestellt sein, hier **BinaryFixed**.

Die EKI ist als Client konfiguriert. Über die Verbindung können nur Binär-Datensätze mit einer festen Länge von 10 Bytes und dem Element-Namen "Buffer" empfangen werden. Das Server-Programm sendet einen Datensatz. Wenn die Schnittstelle externe Daten empfangen hat, wird $FLAG[1] gesetzt.

XML-Datei

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="BYTE" Set_Flag="1" Size="10" />
    </RAW>
  </RECEIVE>
  <SEND />
</ETHERNETKRL>
```

Binär-Datensätze fester Länge müssen im KRL-Programm mit CAST_TO() und CAST_FROM() ein- und ausgelesen werden. Es sind nur Daten vom Typ REAL (4 Bytes) lesbar, kein Double.

Detaillierte Informationen zu den Befehlen CAST_TO() und CAST_FROM() sind in der Dokumentation CREAD/CWRITE zu finden.

Programm

```
 1  DEF BinaryFixed( )
 2  Declaration
 3  INI
 4  Initialize sample data
 5
 6  RET=EKI_Init("BinaryFixed")
 7  RET=EKI_Open("BinaryFixed")
 8
 9  OFFSET=0
10  CAST_TO(Bytes[],OFFSET,34.425,674345,"R",TRUE)
11
12  RET = EKI_Send("BinaryFixed",Bytes[])
13
14  WAIT FOR $FLAG[1]
15  RET=EKI_GetString("BinaryFixed","Buffer",Bytes[])
16  $FLAG[1]=FALSE
17
18  OFFSET=0
19  CAST_FROM(Bytes[],OFFSET,valueReal,valueInt,
            valueChar[],valueBool)
20
21
22  RET=EKI_Close("BinaryFixed")
23  RET=EKI_Clear("BinaryFixed")
24  END
```

Zeile	Beschreibung
4	Initialisieren der KRL-Variablen durch Zuweisung von Werten
6	EKI_Init() initialisiert den Kanal, über den sich die Schnittstelle mit dem externen System verbindet.
7	EKI_Open() öffnet den Kanal und verbindet sich mit dem Server.
9, 10	CAST_TO schreibt die Werte in das CHAR-Feld Bytes[].
12	EKI_Send() sendet das CHAR-Feld Bytes[] an das externe System.
14 … 16	$FLAG[1] signalisiert den Empfang des konfigurierten Datenelements. EKI_GetString greift auf den Speicher zu und kopiert die Daten in das CHAR-Feld Bytes[]. $FLAG[1] wird wieder zurückgesetzt.
18, 19	CAST_FROM liest die im CHAR-Feld Bytes[] enthaltenen Werte aus und kopiert sie typgerecht in die angegebenen Variablen.
22	EKI_Close() schließt den Kanal.
23	EKI_Clear () löscht den Kanal.

7.2.2 Beispielkonfiguration BinaryStream

Zur Kommunikation mit der Robotersteuerung müssen im Server-Programm die passenden Beispieldaten eingestellt sein, hier **BinaryStream**.

Die EKI ist als Client konfiguriert. Über die Verbindung können nur Binär-Datensätze mit einer Länge von maximal 64 Bytes und dem Element-Namen "Buffer" empfangen werden. Das Ende des Binär-Datensatzes muss mit der

Endzeichenfolge CR, LF gekennzeichnet seien. Wenn die Schnittstelle dieses Element empfangen hat, wird $FLAG[1] gesetzt.

XML-Datei

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <RAW>
      <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="1"
                    Size="64" EOS="13,10" />
    </RAW>
  </RECEIVE>
  <SEND />
</ETHERNETKRL>
```

Programm

```
 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
```

Zeile	Beschreibung
4	Initialisieren der KRL-Variablen durch Zuweisung von Werten
6	EKI_Init() initialisiert den Kanal, über den sich die Schnittstelle mit dem externen System verbindet.
7	EKI_Open() öffnet den Kanal und verbindet sich mit dem Server.
9	Das CHAR-Feld Bytes[] wird mit Daten beschrieben.
11	EKI_Send() sendet das CHAR-Feld Bytes[] an das externe System.
13 … 15	$FLAG[1] signalisiert den Empfang des konfigurierten Datenelements. EKI_GetString liest die Zeichenfolge im CHAR-Feld Bytes[] aus dem Speicher. $FLAG[1] wird wieder zurückgesetzt.
17	EKI_Close() schließt den Kanal.
18	EKI_Clear() löscht den Kanal.

7.2.3 Beispielkonfiguration XmlTransmit

Zur Kommunikation mit der Robotersteuerung müssen im Server-Programm die passenden Beispieldaten eingestellt sein, hier **Xml**.

Die EKI ist als Client konfiguriert. Es werden Roboterdaten gesendet und nach einer Wartezeit von 1 sec die empfangenen Sensordaten aus dem Speicher gelesen.

XML-Datei

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</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/Data/ActPos/@X" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
      <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
      </XML>
  <SEND />
</ETHERNETKRL>
```

Programm

```
 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
```

Zeile	Beschreibung
5	Initialisieren der KRL-Variablen durch Zuweisung von Werten
7	EKI_Init() initialisiert den Kanal, über den sich die Schnittstelle mit dem externen System verbindet.
8	EKI_Open() öffnet den Kanal und verbindet sich mit dem externen System.
10	Schreibt Daten in das gespeicherte XML-Dokument für den Datenversand.

Zeile	Beschreibung
11	Sendet das beschriebene XML-Dokument an das externe System.
12	Liest die empfangenen Sensordaten aus dem Speicher.
14	EKI_Close() schließt den Kanal.
15	EKI_Clear () löscht den Kanal.

7.2.4 Beispielkonfiguration XmlServer

Wenn die Schnittstelle als Server konfiguriert ist, kann das Server-Programm auf dem externen System nicht verwendet werden. Ein einfacher Client kann mit Windows Hyperterminal realisiert werden.

Die EKI ist als Server konfiguriert. Solange eine Verbindung zum externen System besteht, ist $FLAG[1] gesetzt.

XML-Datei

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <TYPE>Client</TYPE>
    </EXTERNAL>
    <INTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>54600</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>
```

Programm

```
 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
```

Zeile	Beschreibung
5	EKI_Init() initialisiert den Kanal, über den sich das externe System mit der Schnittstelle verbindet.
6	EKI_Open() öffnet den Kanal.
9	Wenn sich der externe Client erfolgreich mit dem Server verbunden hat, wird $FLAG[1] gesetzt.

Zeile	Beschreibung
11	Da die Schnittstelle als Server konfiguriert ist, erwartet die Robotersteuerung, dass der Kanal vom externen Client geschlossen wird. Wenn dies der Fall ist, wird $FLAG[1] gelöscht.
13	EKI_Clear () löscht den Kanal.

7.2.5 Beispielkonfiguration XmlCallback

Zur Kommunikation mit der Robotersteuerung müssen im Server-Programm die passenden Beispieldaten eingestellt sein, hier **Xml**.

Die EKI ist als Client konfiguriert. Es werden Roboterdaten gesendet, Sensordaten empfangen und dann auf $FLAG[1] gewartet. $FLAG[1] signalisiert, dass die Sensordaten ausgelesen wurden.

In der XML-Datei ist konfiguriert, dass $FLAG[998] gesetzt wird, wenn die Schnittstelle alle Sensordaten empfangen hat. Dieses Flag löst einen Interrupt im Programm aus. Durch die Konfiguration des Tags "Sensor" als Ereignis-Tag wird sichergestellt, dass die Sensordaten erst abgeholt werden, wenn alle Daten in den Speichern liegen.

Wenn die Sensordaten ausgelesen sind, wird $FLAG[998] wieder zurückgesetzt und $FLAG[1] gesetzt.

XML-Datei

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL>
      <IP>x.x.x.x</IP>
      <PORT>59152</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" Set_Out="998" />
      <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/Data/ActPos/@X" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
      <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
    </XML>
  <SEND />
</ETHERNETKRL>
```

Programm

```
 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
```

Zeile	Beschreibung
5	Deklaration und Einschalten des Interrupts
7	EKI_Init() initialisiert den Kanal, über den sich die Schnittstelle mit dem externen System verbindet.
8	EKI_Open() öffnet den Kanal.
10	Schreibt Daten in das gespeicherte XML-Dokument für den Datenversand.
11	Sendet die Daten.
14	Wartet auf $FLAG[1]. Das Ereignis-Flag meldet, dass alle Daten gelesen wurden.
16	EKI_Close() schließt den Kanal.
17	EKI_Clear () löscht den Kanal.
20 … 24	Initialisieren der KRL-Variablen durch Zuweisung von Werten und Auslesen der Daten Wenn alle Daten gelesen sind, wird $FLAG[1] gesetzt.

Datenversand

Das XML-Dokument wird vom KRL-Programm mit Roboterdaten beschrieben und über die EKI an das externe System gesendet.

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

Datenempfang

Das XML-Dokument wird vom Server-Programm mit Sensordaten beschrieben und von der EKI empfangen.

```
<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 Diagnose

8.1 Diagnosedaten anzeigen

Vorgehensweise

1. Im Hauptmenü **Diagnose** > **Diagnosemonitor** wählen.
2. Im Feld **Modul** das Modul **EKI (EthernetKRL)** auswählen.

Beschreibung

Name	Beschreibung
Gesamtspeicher	Insgesamt verfügbarer Speicher (Bytes)
Verbrauchter Speicher	Benutzer Speicher (Bytes)
Verbindungen Roboterprogramm	Anzahl der vom Roboter-Interpreter initialisierten Verbindungen
Verbindungen Submitprogramm	Anzahl der vom Submit-Interpreter initialisierten Verbindungen
Verbindungen System	Anzahl der vom System initialisierten Verbindungen
Ethernet Verbindungen	Anzahl offener Verbindungen
Verarbeitungszeit	Maximale Zeit, die benötigt wird, um empfangene Daten zu bearbeiten (Aktualisierung alle 5 sec)
Warnmeldungen	Anzahl der Warnmeldungen
Fehlermeldungen	Anzahl der Fehlermeldungen

Die Warn- und Fehlermeldungen werden auch dann gezählt, wenn die automatische Meldungsausgabe und das Loggen von Meldungen deaktiviert wurde.

8.2 Fehlerprotokoll (EKI-Logbuch)

Alle Fehlermeldungen der Schnittstelle werden in einer LOG-Datei unter C:\KRC\ROBOTER\LOG\EthernetKRL protokolliert.

8.3 Fehlermeldungen

Wenn beim Aufruf einer Funktion oder beim Datenempfang ein Fehler aufgetreten ist, gibt EthernetKRL die Fehlernummer zurück. Den Fehlernummern ist ein Meldungstext zugeordnet, der auf der smartHMI angezeigt wird. Ist die automatische Ausgabe von Meldungen deaktiviert, kann über EKI_CHECK() die Meldung weiterhin auf der smartHMI angezeigt werden.

Nr.	Meldungstext	Ursache	Abhilfe
1	*Unbekannter Fehler*	Dem Fehler wurde keine Meldung zugewiesen.	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
2	*Der Systemspeicher ist verbraucht*	Der für EthernetKRL reservierte Speicher ist vollständig belegt. Es können keine weiteren Elemente gespeichert werden.	Programmierweise in KRL und Konfiguration der Ethernet-Verbindung überprüfen. Wenn keine andere Programmierung oder Konfiguration möglich ist, kann nach Rücksprache mit der KUKA Roboter GmbH der Speicher erhöht werden. (>>> 9.2 "Speicher erhöhen" Seite 59)
3	*Zugriff auf Datei fehlgeschlagen*	Eine Datei konnte nicht gefunden werden oder ist nicht lesbar.	Überprüfen, ob die Datei vorhanden ist oder ob die Datei sich öffnen lässt.
4	*Angeforderte Funktion nicht implementiert*	Software-Fehler: Die verwendete EthernetKRL-Funktion ist nicht implementiert.	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
5	*Fehler bei Erstellen des XML Parsers*	Die Verbindung wurde nicht initialisiert, da der systeminterne Parser nicht aktiviert werden konnte.	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
6	*Interpretieren der Konfiguration fehlgeschlagen*	Fehler beim Lesen der Verbindungskonfiguration	Konfiguration der Ethernet-Verbindung überprüfen.
7	*Schreiben der Daten zum Senden fehlgeschlagen*	Fehler beim Beschreiben der XML-Struktur für den Datenversand	Konfiguration der Sendestruktur überprüfen.
8	*Neues Element konnte nicht angelegt werden*	Fehler beim Anlegen der Datenspeichers	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
9	*Verbindung nicht vorhanden*	Wegen fehlender Initialisierung ist kein Zugriff auf die Verbindung möglich.	Ethernet-Verbindung mit EKI_Init() initialisieren.
10	*Ethernet ist getrennt*	Es ist keine Ethernet-Verbindung vorhanden.	Ethernet-Verbindung mit EKI_Open() öffnen.
11	*Ethernetverbindung zu externem System bereits vorhanden*	Ethernet-Verbindung ist bereits vorhanden.	Funktion EKI_Open() nicht aufrufen, wenn Ethernet-Verbindung bereits vorhanden ist.

Nr.	Meldungstext	Ursache	Abhilfe
12	*Erstellen des Servers fehlgeschlagen*	Eine Ethernet-Verbindung, die als Server konfiguriert ist, konnte nicht erstellt werden.	Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
13	*Ethernetparameter konnten nicht initialisiert werden*	Fehler beim Initialisieren der Ethernet-Verbindung	Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
14	*Ethernetverbindung zu externem System konnte nicht hergestellt werden*	Keine Ethernet-Verbindung: ■ Hardware-Fehler, z. B. Netzwerkkabel, Switch, externes System ■ Software-Fehler (externes System) ■ Fehler bei der Verbindungskonfiguration	Ethernet-Verbindung herstellen: ■ Hardware überprüfen. ■ Software des externen Systems überprüfen. ■ Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
15	*Zugriff auf leeren Empfangsspeicher*	Keine Datenelemente im Speicher bei Zugriff mit EKI_Get…()	Im KRL-Programm den Rückgabewert der Funktion EKI_Get…() auswerten, um nicht auf leere Speicher zuzugreifen. (Element "Buff") (>>> 6.2.7 "EKI_STATUS – Struktur für die funktionsspezifischen Rückgabewerte" Seite 36)
16	*Element konnte nicht gefunden werden*	Ein in der Zugriffsfunktion EKI_Get…() angegebenes Datenelement kann nicht gefunden werden.	■ Name des Datenelements und seine Schreibweise im KRL-Programm überprüfen. ■ Konfiguration der Empfangsstruktur überprüfen.
17	*Zusammenstellen der Daten zum Senden fehlgeschlagen*	■ Senden von XML-Daten: Fehler beim Beschreiben des XML-Dokumentes für den Datenversand ■ Senden von Binär-Daten: Fehler beim Überprüfen der zu sendenden Binär-Daten	■ Senden von XML-Daten: Konfiguration der Sendestruktur überprüfen. ■ Senden von Binär-Daten: Funktion EKI_Send() im KRL-Programm überprüfen.
18	*Senden von Daten fehlgeschlagen*	Keine Ethernet-Verbindung: ■ Hardware-Fehler, z. B. Netzwerkkabel, Switch, externes System ■ Software-Fehler (externes System)	Ethernet-Verbindung herstellen: ■ Hardware überprüfen. ■ Software des externen Systems überprüfen.
19	*Keine Daten zum Senden vorhanden*	In einer Funktion EKI_Send() sind die zu sendenden Daten nicht angegeben.	Funktion EKI_Send() im KRL-Programm überprüfen.

Nr.	Meldungstext	Ursache	Abhilfe
20	*Datentypen passen nicht zusammen*	Es wurde versucht ein Element zu lesen, das zu einem anderen Datentyp gehört.	Datentyp des Elements in der Konfiguration der Empfangsstruktur überprüfen. ODER Im KRL-Programm den Datentyp verwenden, der in der Konfiguration der Empfangsstruktur definiert ist.
21	*Mit maximaler Datenhaltung Systemspeicher nicht ausreichend*	Beim Einlesen der Konfiguration wurde festgestellt, dass der Systemspeicher nicht ausreicht.	Konfiguration der Ethernet-Verbindung überprüfen und so anpassen, dass weniger Speicher verbraucht wird. Wenn keine andere Konfiguration möglich ist, kann nach Rücksprache mit der KUKA Roboter GmbH der Speicher erhöht werden. (>>> 9.2 "Speicher erhöhen" Seite 59)
22	*Fehler bei Lesen der Konfiguration. XML Fehler.*	Beim Einlesen der Konfiguration wurde ein Fehler in der XML-Struktur festgestellt.	XML-Struktur in der Konfigurationsdatei überprüfen.
24	*Bindung an interne Parameter (Port,IP) fehlgeschlagen*	Die Ethernet-Verbindung, d. h. die Schnittstelle, ist als Server konfiguriert. Die in der Konfiguration angegebene IP-Adresse und Port-Nummer des externen Systems stehen nicht zur Verfügung.	In der Konfiguration der Verbindungsparameter die richtige IP-Adresse und Port-Nummer verwenden. (Elemente IP, PORT)
25	*Interner Softwarefehler*	Interner Software-Fehler	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
26	*Das FRAME-Feld ist nicht initialisiert*	Ein Feld vom Typ FRAME wurde nicht initialisiert.	Feld vom Typ FRAME initialisieren (Wert zuweisen).
27	*Das KRL CHAR[] Feld ist zu klein.*	Ein Feld vom Typ CHAR ist zu klein.	Anzahl der Feld-Elemente erhöhen.
512	*Ethernetverbindung gestört*	Keine Ethernet-Verbindung: ■ Hardware-Fehler, z. B. Netzwerkkabel, Switch, externes System ■ Software-Fehler (externes System)	Ethernet-Verbindung wiederherstellen: ■ Hardware überprüfen. ■ Software des externen Systems überprüfen.
768	*Ping meldet kein Kontakt*	Das externe System antwortet nicht mehr auf das gesendete Ping. Die Verbindung ist abgebrochen.	Externes System überprüfen.

Nr.	Meldungstext	Ursache	Abhilfe
1024	*Fehler bei Lesen empfangener XML-Daten*	Ein vom externen System empfangenes XML-Dokument entspricht nicht dem XPath-Schema.	Das vom externen System gesendete XML-Dokument überprüfen.
1280	*Grenze speicherbarer Elemente erreicht*	Der Datenspeicher ist mit der maximalen Anzahl an Datenelementen belegt. Die Ethernet-Verbindung wird geschlossen.	Im KRL-Programm den Rückgabewert der Funktion EKI_Get…() auswerten, um das Verarbeiten empfangener Daten zu sperren. (Element "Buff") (>>> 6.2.7 "EKI_STATUS – Struktur für die funktionsspezifischen Rückgabewerte" Seite 36) ODER Datenspeicher erhöhen. (Element BUFFERING in der Verbindungskonfiguration)
1536	*Empfangene Zeichenkette zu lang*	Programmierfehler auf externem System: Eine vom externen System empfangene Zeichenfolge überschreitet die maximal zulässige Länge. (Maximal 3 600 Zeichen)	Die vom externen System gesendeten Daten überprüfen.
1792	*Limit Empfangsspeicher erreicht*	Der Datenspeicher ist mit der maximalen Anzahl an Bytes belegt. Die Ethernet-Verbindung wird geschlossen.	Datenspeicher erhöhen. (Element BUFFSIZE in der Verbindungskonfiguration)
2048	*Server Zeitgrenze erreicht*	Server wartet auf einen Anruf.	Externes System überprüfen.

9 Anhang

9.1 Erweiterte XML-Struktur für Verbindungseigenschaften

> Die erweiterte XML-Struktur darf nur nach Rücksprache mit der KUKA Roboter GmbH verwendet werden. (>>> 10 "KUKA Service" Seite 69)

Beschreibung Im Abschnitt <INTERNAL> … </INTERNAL> der Konfigurationsdatei können weitere Schnittstellen-Eigenschaften konfiguriert werden:

Element	Attribut	Beschreibung
TIMEOUT	Receive	Zeit, nach der der Versuch Daten zu empfangen abgebrochen wird (optional) ■ **0 … 65 534 ms** Default-Wert: **0 ms**
	Send	Zeit, nach der der Versuch Daten zu senden abgebrochen wird (optional) ■ **0 … 65 534 ms** Default-Wert: **0 ms**
BUFFSIZE	Receive	Größe des verwendeten Sockets beim Datenempfang (optional) ■ **1 … 65 534 Bytes** Default-Wert: Vom System vorgegeben
	Send	Größe des verwendeten Sockets beim Datenversand (optional) ■ **1 … 65 534 Bytes** Default-Wert: Vom System vorgegeben

9.2 Speicher erhöhen

> Der Speicher darf nur nach Rücksprache mit der KUKA Roboter GmbH erhöht werden. (>>> 10 "KUKA Service" Seite 69)

Beschreibung Wenn der zur Verfügung stehende Speicher nicht ausreicht, wird empfohlen die Programmierweise in KRL und die Konfiguration zu überprüfen.

- Überprüfen, ob eine Verbindung so konfiguriert ist , dass der Speicher vollständig mit empfangenen Daten belegt wird.
- Überprüfen, ob mehrere Verbindungen mit hohem Datenaufkommen definiert und aktiviert sind.

Voraussetzung

- Windows-Ebene

Vorgehensweise

1. Datei C:\KRC\ROBOTER\Config\User\Common\EthernetKRL.XML öffnen.
2. Im Abschnitt <EthernetKRL> im Element <MemSize> die gewünschte Speicherkapazität in Byte eintragen.

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

3. Änderung speichern und Datei schließen.

9.3 Meldungsausgabe und Loggen von Meldungen deaktivieren

Beschreibung

In folgenden Fällen wird empfohlen, die automatische Meldungsausgabe auf der smartHMI zu deaktivieren:

- Es treten Laufzeitfehler auf.
- Die EKI wird im Submit-Interpreter verwendet.

Wenn die automatische Meldungsanausgabe deaktiviert ist, werden default-mäßig weiterhin alle Fehlermeldungen protokolliert. Wenn diese Fehler oder Warnungen bewusst ignoriert werden sollen, z. B. weil das Loggen der Meldungen zu einer hohen Systembelastung und -verlangsamung führt, kann dieser Mechanismus ebenfalls deaktiviert werden.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Verzeichnis C:\KRC\ROBOTER\Config\User\Common\EthernetKRL der Robotersteuerung die Konfigurationsdatei der Ethernet-Verbindung öffnen.
2. Im Abschnitt <INTERNAL> … </INTERNAL> der XML-Datei folgende Zeile eintragen:

```
<Messages Display="disabled" Logging="disabled"/>
```

3. Änderung speichern und Datei schließen.

Wenn die automatische Meldungsausgabe deaktiviert ist, kann die Funktion EKI_CHECK() verwendet werden, um einzelne EKI-Anweisungen auf Fehler zu überprüfen.

9.4 Befehlsreferenz

9.4.1 Verbindung initialisieren, öffnen, schließen und löschen

RET = EKI_Init(CHAR[])	
Funktion	Initialisiert einen Kanal für die Ethernet-Kommunikation Folgende Aktionen werden ausgeführt: ■ Einlesen der Konfiguration ■ Erstellen der Datenspeicher ■ Vorbereiten der Ethernet-Verbindung
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_Init("Channel_1")

RET = EKI_Open(CHAR[])	
Funktion	Öffnet einen initialisierten Kanal Wenn die EthernetKRL-Schnittstelle als Client konfiguriert ist, verbindet sich die Schnittstelle mit dem Server. Wenn die EthernetKRL-Schnittstelle als Server konfiguriert ist, wartet die Schnittstelle auf die Verbindung.
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_Open("Channel_1")

RET = EKI_Close(CHAR[])	
Funktion	Schließt einen geöffneten Kanal
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_Close("Channel_1")

RET = EKI_Clear(CHAR[])	
Funktion	Löscht einen Kanal und beendet die Verbindung.
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_Clear("Channel_1")

9.4.2 Daten senden

RET = EKI_Send(CHAR[], CHAR[])	
Funktion	Sendet eine XML-Struktur oder Rohdaten (>>> 6.2.4 "Senden von Daten" Seite 32)
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur oder Name des Elements in den Rohdaten Wenn die Position oder das Element nicht gefunden wird, sendet die Funktion die hier enthaltene Information
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)

RET = EKI_Send(CHAR[], CHAR[])	
Beispiel 1	RET = EKI_Send("Channel_1", "Root/Test")
Beispiel 2	RET = EKI_Send("Channel_1", MyBytes[])

9.4.3 Daten schreiben

RET = EKI_SetReal(CHAR[], CHAR[], REAL)	
Funktion	Schreibt einen Gleitkommawert in einen Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: REAL Wert, der in den Speicher geschrieben wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_SetReal("Channel_1", "Root/Number", 1.234)

RET = EKI_SetInt(CHAR[], CHAR[], INTEGER)	
Funktion	Schreibt einen ganzzahligen Wert in einen Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: INT Wert, der in den Speicher geschrieben wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_SetInt("Channel_1", "Root/List", 67234)

RET = EKI_SetBool(CHAR[], CHAR[], BOOL)	
Funktion	Schreibt einen booleschen Wert in einen Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: BOOL Wert, der in den Speicher geschrieben wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_SetBool("Channel_1", "Root/Activ", true)

RET = EKI_SetFrame(CHAR[], CHAR[], FRAME)	
Funktion	Schreibt einen Wert vom Typ FRAME in einen Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: FRAME Wert, der in den Speicher geschrieben wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	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[])	
Funktion	Schreibt eine Zeichenfolge in einen Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: CHAR Zeichenfolge, die in den Speicher geschrieben wird Maximale Zeichen-Anzahl: ■ **3 600**
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_SetString("Channel_1", "Root/Message", "Hello")

9.4.4 Daten auslesen

RET = EKI_GetBool(CHAR[], CHAR[], BOOL)	
Funktion	Liest einen booleschen Wert aus einem Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: BOOL Wert, der aus dem Speicher gelesen wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetBool("Channel_1", "Root/Activ", MyBool)

RET = EKI_GetBoolArray(CHAR[], CHAR[], BOOL[])	
Funktion	Liest einen booleschen Wert aus einem Speicher und kopiert den Wert in das vom KRL-Programm übergebene Feld Es werden so lange Werte gelesen bis das Feld voll ist oder kein Element mehr vorhanden ist.
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: BOOL Feld, das aus dem Speicher gelesen wird Maximale Anzahl lesbarer Feld-Elemente: ■ **512**
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetBoolArray("Channel_1", "Root/Activ", MyBool[])

RET = EKI_GetInt(CHAR[], CHAR[], Int)	
Funktion	Liest einen ganzzahligen Wert aus einem Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: INT Wert, der aus dem Speicher gelesen wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetInt("Channel_1", "Root/Numbers/One", MyInteger)

RET = EKI_GetIntArray(CHAR[], CHAR[], Int[])	
Funktion	Liest einen ganzzahligen Wert aus einem Speicher und kopiert den Wert in das vom KRL-Programm übergebene Feld Es werden so lange Werte gelesen bis das Feld voll ist oder kein Element mehr vorhanden ist.
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: INT Feld, das aus dem Speicher gelesen wird Maximale Anzahl lesbarer Feld-Elemente: ■ **512**

RET = EKI_GetIntArray(CHAR[], CHAR[], Int[])	
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetIntArray("Channel_1", "Root/Numbers/One", MyInteger[])

RET = EKI_GetReal(CHAR[], CHAR[], Real)	
Funktion	Liest einen Gleitkommawert aus einem Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: REAL Wert, der aus dem Speicher gelesen wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetReal("Channel_1", "Root/Position", MyReal)

RET = EKI_GetRealArray(CHAR[], CHAR[], Real[])	
Funktion	Liest einen Gleitkommawert aus einem Speicher und kopiert den Wert in das vom KRL-Programm übergebene Feld Es werden so lange Werte gelesen bis das Feld voll ist oder kein Element mehr vorhanden ist.
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: REAL Feld, das aus dem Speicher gelesen wird Maximale Anzahl lesbarer Feld-Elemente: ■ **512**
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetRealArray("Channel_1", "Root/Position", MyReal[])

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
Funktion	Liest eine Zeichenfolge aus einem Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur oder Name des Elements in den Rohdaten

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
Parameter 3	Typ: CHAR Zeichenfolge, die aus dem Speicher gelesen wird Maximale Zeichen-Anzahl: ■ **3 600**
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
XML-Beispiel	RET = EKI_GetString("Channel_1", "Root/Message", MyChars[])
Binär-Beispiel	RET = EKI_GetString("Channel_1", "Streams", MyStream[])

RET = EKI_GetFrame(CHAR[], CHAR[], FRAME)	
Funktion	Liest einen Wert vom Typ FRAME aus einem Speicher
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: FRAME Wert, der aus dem Speicher gelesen wird
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetFrame("Channel_1", "Root/TCP", MyFrame)

RET = EKI_GetFrameArray(CHAR[], CHAR[], FRAME[])	
Funktion	Liest einen Wert vom Typ FRAME aus einem Speicher und kopiert den Wert in das vom KRL-Programm übergebene Feld Es werden so lange Werte gelesen bis das Feld voll ist oder kein Element mehr vorhanden ist.
Parameter 1	Typ: CHAR Name des geöffneten Kanals
Parameter 2	Typ: CHAR Name der Position in der XML-Struktur
Parameter 3	Typ: FRAME Feld, das aus dem Speicher gelesen wird Maximale Anzahl lesbarer Feld-Elemente: ■ **512**
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_GetFrameArray("Channel_1", "Root/TCP", MyFrame[])

9.4.5 Funktion auf Fehler prüfen

EKI_CHECK(EKI_STATUS, EKrlMsgType, CHAR[])	
Funktion	Prüft, ob beim Ausführen einer EthernetKRL-Funktion ein Fehler aufgetreten ist: ■ Die Fehlernummer wird ausgelesen und die zugehörige Meldung auf der smartHMI ausgegeben. (Parameter 1) ■ Optional: Wird der Kanalname angegeben, wird beim Datenempfang abgefragt, ob Fehler vorliegen (Parameter 3)
Parameter 1	EKI_STATUS Rückgabewerte der geprüften Funktion (>>> "Rückgabewerte" Seite 36)
Parameter 2	Typ: ENUM Meldungstyp, der auf der smartHMI ausgegeben wird: ■ #**NOTIFY**: Hinweismeldung ■ #**STATE**: Zustandsmeldung ■ #**QUIT**: Quittiermeldung ■ #**WAITING**: Wartemeldung
Parameter 3 (optional)	Typ: CHAR Name des geöffneten Kanals
Beispiel 1	EKI_CHECK(RET,#QUIT)
Beispiel 2	EKI_CHECK(RET,#NOTIFY,"MyChannelName")

9.4.6 Speicher löschen, sperren, entsperren und prüfen

RET = EKI_ClearBuffer(CHAR[], CHAR[])	
Funktion	Löscht empfangene noch nicht abgerufene Daten aus einem Speicher
Parameter 1	Typ: CHAR Name des Kanals
Parameter 2	Typ: CHAR Position des Speichers oder alle Speicher (>>> 6.2.6 "Löschen empfangener Daten" Seite 35)
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel 1	RET = EKI_ClearBuffer("Channel_1", "Root/Activ/Flag")
Beispiel 2	RET = EKI_ClearBuffer("Channel_1", "Root")

RET = EKI_Lock(CHAR[])	
Funktion	Sperrt das Verarbeiten empfangener Daten, d. h. die Daten können nicht mehr im Speicher abgelegt werden.

RET = EKI_Lock(CHAR[])	
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)

RET = EKI_Unlock(CHAR[])	
Funktion	Entsperrt das Verarbeiten empfangener Daten, d. h. die Daten werden wieder im Speicher abgelegt.
Parameter	Typ: CHAR Name des Kanals
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)

RET = EKI_CheckBuffer(CHAR[], CHAR[])	
Funktion	Prüft wieviele Daten sich noch im Speicher befinden. Der Speicher wird nicht verändert. Zusätzlich wird der Zeitstempel des Datenelements zurückgegeben, das als nächstes zur Entnahme aus dem Speicher bereitsteht.
Parameter 1	Typ: CHAR Name des Kanals
Parameter 2	Typ: CHAR Position des Speichers
RET	Typ: EKI_STATUS Rückgabewerte der Funktion (>>> "Rückgabewerte" Seite 36)
Beispiel	RET = EKI_CheckBuffer("Channel_1", "Root/Activ/Flag")

10 KUKA Service

10.1 Support-Anfrage

Einleitung

Die Dokumentation der KUKA Roboter GmbH bietet Informationen zu Betrieb und Bedienung und unterstützt Sie bei der Behebung von Störungen. Für weitere Anfragen steht Ihnen die lokale Niederlassung zur Verfügung.

Informationen

Zur Abwicklung einer Anfrage werden folgende Informationen benötigt:

- Typ und Seriennummer des Roboters
- Typ und Seriennummer der Steuerung
- Typ und Seriennummer der Lineareinheit (optional)
- Typ und Seriennummer der Energiezuführung (optional)
- Version der KUKA System Software
- Optionale Software oder Modifikationen
- Archiv der Software

  Für KUKA System Software V8: Statt eines herkömmlichen Archivs das spezielle Datenpaket für die Fehleranalyse erzeugen (über **KrcDiag**).
- Vorhandene Applikation
- Vorhandene Zusatzachsen (optional)
- Problembeschreibung, Dauer und Häufigkeit der Störung

10.2 KUKA Customer Support

Verfügbarkeit

Der KUKA Customer Support ist in vielen Ländern verfügbar. Bei Fragen stehen wir gerne zur Verfügung!

Argentinien

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

Australien

Headland Machinery Pty. Ltd.
Victoria (Head Office & Showroom)
95 Highbury Road
Burwood
Victoria 31 25
Australien
Tel. +61 3 9244-3500
Fax +61 3 9244-3501
vic@headland.com.au
www.headland.com.au

Belgien

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

Brasilien

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

Chile

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

China

KUKA Robotics China Co.,Ltd.
Songjiang Industrial Zone
No. 388 Minshen Road
201612 Shanghai
China
Tel. +86 21 6787-1888
Fax +86 21 6787-1803
www.kuka-robotics.cn

Deutschland

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

Frankreich

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

Indien

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

Italien

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

Japan

KUKA Robotics Japan K.K.
YBP Technical Center
134 Godo-cho, Hodogaya-ku
Yokohama, Kanagawa
240 0005
Japan
Tel. +81 45 744 7691
Fax +81 45 744 7696
info@kuka.co.jp

Kanada

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

Korea

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

Malaysia

KUKA Robot Automation 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

Mexiko

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

Norwegen

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

Österreich

KUKA Roboter Austria GmbH
Regensburger Strasse 9/1
4020 Linz
Österreich
Tel. +43 732 784752
Fax +43 732 793880
office@kuka-roboter.at
www.kuka-roboter.at

Polen

KUKA Roboter Austria GmbH
Spółka z ograniczoną odpowiedzialnością
Oddział w Polsce
Ul. Porcelanowa 10
40-246 Katowice
Polen
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

Russland

OOO KUKA Robotics Rus
Webnaja ul. 8A
107143 Moskau
Russland
Tel. +7 495 781-31-20
Fax +7 495 781-31-19
kuka-robotics.ru

Schweden

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

Schweiz

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

Spanien

KUKA Robots IBÉRICA, S.A.
Pol. Industrial
Torrent de la Pastera
Carrer del Bages s/n
08800 Vilanova i la Geltrú (Barcelona)
Spanien
Tel. +34 93 8142-353
Fax +34 93 8142-950
Comercial@kuka-e.com
www.kuka-e.com

Südafrika

Jendamark Automation LTD (Agentur)
76a York Road
North End
6000 Port Elizabeth
Südafrika
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

Tschechien

KUKA Roboter Austria GmbH
Organisation Tschechien und Slowakei
Sezemická 2757/2
193 00 Praha
Horní Počernice
Tschechische Republik
Tel. +420 22 62 12 27 2
Fax +420 22 62 12 27 0
support@kuka.cz

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

Index