

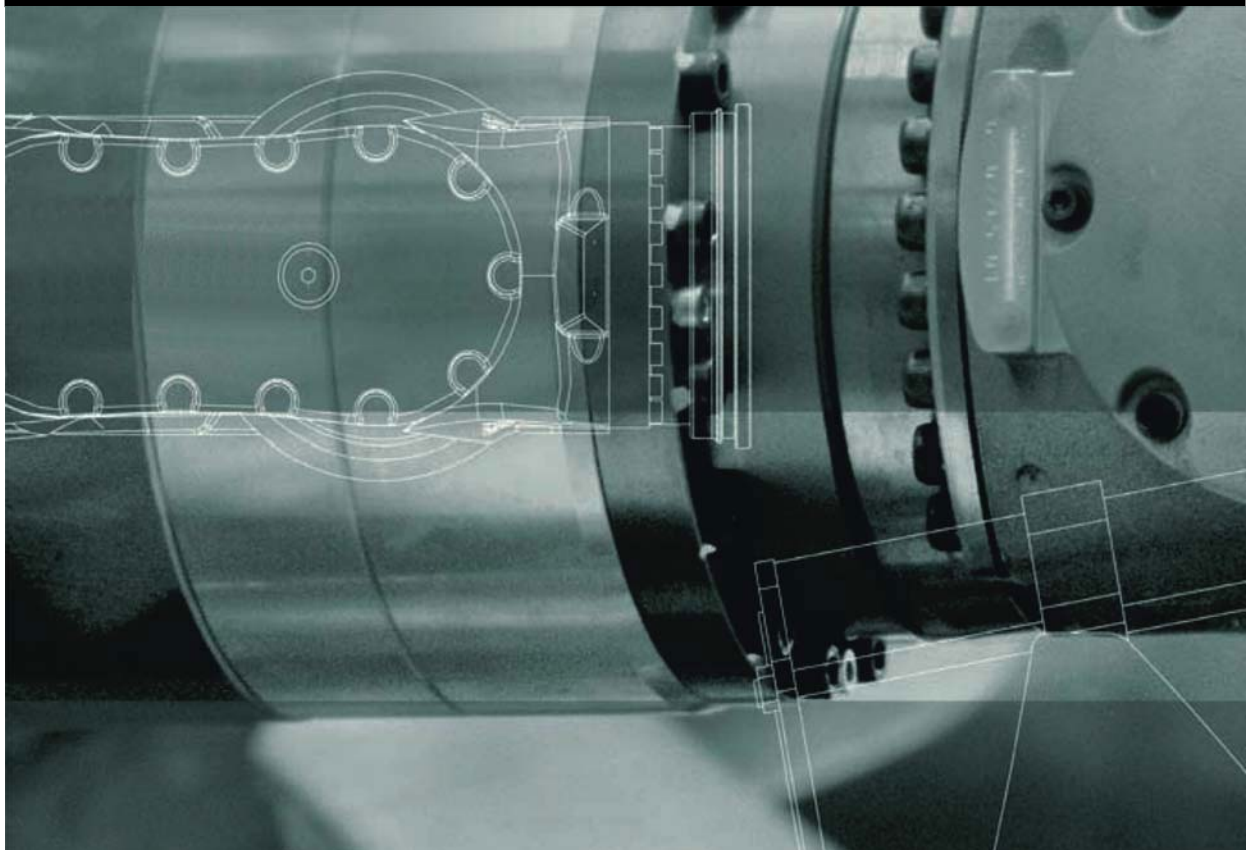


VW System Technology

KUKA Roboter GmbH

KUKA.Ethernet KRL 2.1

Für VW System Software 8.2



Stand: 16.03.2012

Version: VST Ethernet KRL 2.1 V3 de



© Copyright 2012

KUKA Roboter GmbH
Zugspitzstraße 140
D-86165 Augsburg
Deutschland

Diese Dokumentation darf – auch auszugsweise – nur mit ausdrücklicher Genehmigung der KUKA Roboter GmbH vervielfältigt oder Dritten zugänglich gemacht werden.

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 VST Ethernet KRL 2.1 de
Buchstruktur:	VST Ethernet KRL 2.1 V2.1
Version:	VST Ethernet KRL 2.1 V3 de

Inhaltsverzeichnis

1	Einleitung	5
1.1	Zielgruppe	5
1.2	Dokumentation des Industrieroboters	5
1.3	Darstellung von Hinweisen	5
1.4	Verwendete Begriffe	6
1.5	Warenzeichen	7
2	Produktbeschreibung	9
2.1	Übersicht Ethernet KRL	9
2.2	Konfiguration einer Ethernet-Verbindung	9
2.2.1	Verhalten bei Verbindungsverlust	9
2.2.2	Überwachen einer Verbindung	10
2.3	Datenaustausch	10
2.4	Datenspeicherung	11
2.5	Client-Server-Betrieb	12
2.6	Protokollarten	12
2.7	Ereignismeldungen	13
2.8	Fehlerbehandlung	13
3	Sicherheit	15
4	Installation	17
4.1	Systemvoraussetzungen	17
4.2	Ethernet KRL installieren oder updaten	17
4.3	Ethernet KRL deinstallieren	17
5	Konfiguration	19
5.1	Netzwerkverbindung über die KLI der Robotersteuerung	19
5.2	Netzwerkverbindung konfigurieren	19
6	Programmierung	21
6.1	Ethernet-Verbindung konfigurieren	21
6.1.1	XML-Struktur für Verbindungseigenschaften	21
6.1.2	XML-Struktur für den Datenempfang	23
6.1.3	XML-Struktur für den Datenversand	26
6.1.4	Konfiguration nach XPath-Schema	26
6.2	Ethernet KRL-Funktionen für den Datenaustausch	27
6.2.1	Programmiertipps	28
6.2.2	Initialisieren und Löschen einer Verbindung	28
6.2.3	Öffnen und Schließen einer Verbindung	29
6.2.4	Senden von Daten	30
6.2.5	Auslesen von Daten	31
6.2.6	Löschen empfangener Daten	33
6.2.7	Rückgabewert der Ethernet KRL-Funktionen	34
6.2.8	Konfigurieren von Ereignismeldungen	34
6.2.9	Empfang vollständiger XML-Datensätze	35
6.2.10	Fehlerbehandlung	36
7	Beispiele	37

7.1	Beispielapplikationen	37
7.1.1	Beispielapplikationen implementieren	37
7.1.2	Beispielprogramm in Modul VW_USER einbinden	38
7.1.3	Bedienoberfläche Server-Programm	39
7.1.4	Kommunikationsparameter im Server-Programm einstellen	41
7.2	Beispielkonfigurationen und -programme	42
7.2.1	Beispielkonfiguration BinaryFixed	42
7.2.2	Beispielkonfiguration BinaryStream	43
7.2.3	Beispielkonfiguration XmlTransmit	44
7.2.4	Beispielkonfiguration XmlServer	46
7.2.5	Beispielkonfiguration XmlCallback	47
8	Diagnose	51
8.1	Diagnosedaten zu Ethernet KRL anzeigen	51
8.2	Fehlerprotokoll (EKI-Logbuch)	51
8.3	Fehlermeldungen	51
9	Anhang	57
9.1	Erweiterte XML-Struktur für Verbindungseigenschaften	57
9.2	Speicher erhöhen	57
9.3	Anzeige von Meldungen auf der smartHMI deaktivieren	58
9.4	Warnmeldungen im EKI-Logbuch deaktivieren	58
9.5	Ethernet KRL-Funktionen Befehlsreferenz	59
9.5.1	Funktionen zur Initialisierung und Verbindung	59
9.5.2	Sendefunktion	60
9.5.3	Schreibfunktionen	60
9.5.4	Zugriffsfunktionen	62
9.5.5	Funktion zur Fehlerbehandlung	66
9.5.6	Sonstige Funktionen	66
10	KUKA Service	69
10.1	Support-Anfrage	69
10.2	KUKA Customer Support	69
	Index	77

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.



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



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



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



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.

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	Ethernet KRL 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	Verfahren, nach denen ein Datenspeicher abgearbeitet werden kann <ul style="list-style-type: none"> ■ 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.
LIFO	
KLI	KUKA Line Interface Linienbus zur Integration der Anlage in das Kunden-netz
KR C	KUKA Robot Controller KR C ist die KUKA Robotersteuerung.
KRL	KUKA Robot Language KRL ist die KUKA Roboter Programmiersprache.
smarHMI	smart Human-Machine Interface KUKA smarHMI ist die Bedienoberfläche der KUKA System Software.
Socket	Software-Schnittstelle, die IP-Adressen und Port-Nummern miteinander verbindet
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

Begriff	Beschreibung
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 Ethernet KRL

Funktionen	<p>Ethernet KRL ist ein nachladbares Technologiepaket mit folgenden Funktionen:</p> <ul style="list-style-type: none"> ■ Datenaustausch über die Ethernet KRL-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	<ul style="list-style-type: none"> ■ 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 Submit-Interpreter ■ Lesen und Schreiben von Daten aus Roboter-Interpreter
Kommunikation	<p>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).</p> <p>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.</p>

2.2 Konfiguration einer Ethernet-Verbindung

Beschreibung	<p>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.</p> <p>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.</p> <p>Das Löschen einer Verbindung kann an Roboter- und Submit-Interpreter-Aktionen oder Systemaktionen gekoppelt sein.</p>
---------------------	--

2.2.1 Verhalten bei Verbindungsverlust

Beschreibung	<p>Folgende Eigenschaften und Funktionen der EKI stellen sicher, dass empfangene Daten zuverlässig bearbeitet werden können:</p> <ul style="list-style-type: none"> ■ 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 Ethernet KRL kann die Robotersteuerung sowohl Daten von einem externen System empfangen als auch Daten an ein externes System senden.

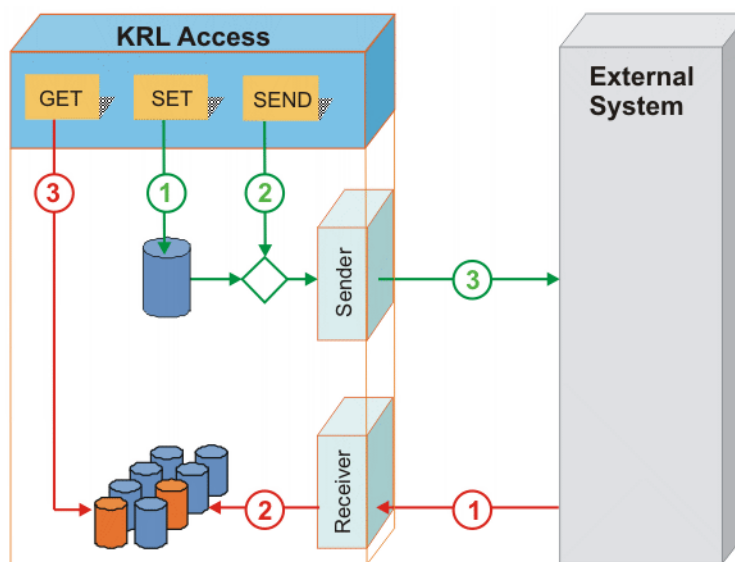


Abb. 2-1: Systemübersicht

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).

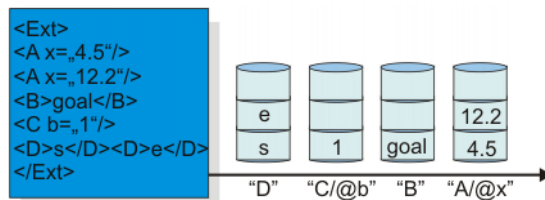


Abb. 2-2: XML-Daten-Speicher

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

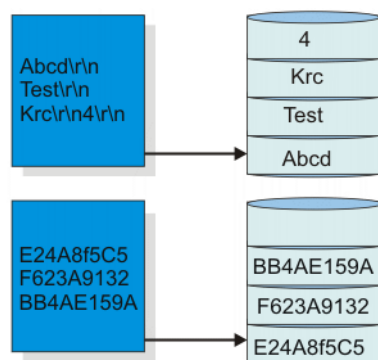


Abb. 2-3: Binäre Daten-Speicher

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. Die Speicher können weiter abgearbeitet werden. Die Verbindung kann über die Ethernet KRL-Funktion `EKI_OPEN()` wieder geöffnet werden.

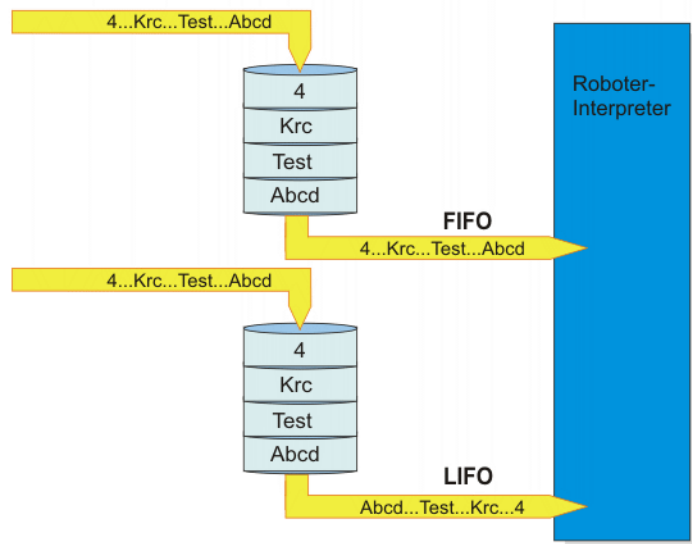


Abb. 2-4: Ausleseverfahren Übersicht

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.

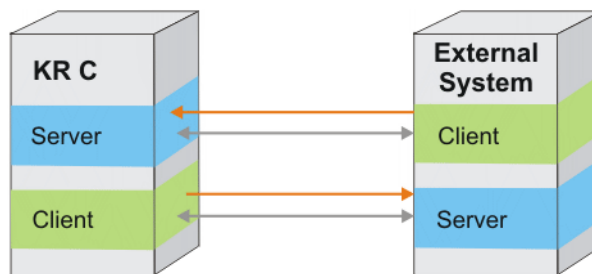


Abb. 2-5: Client-Server-Betrieb

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

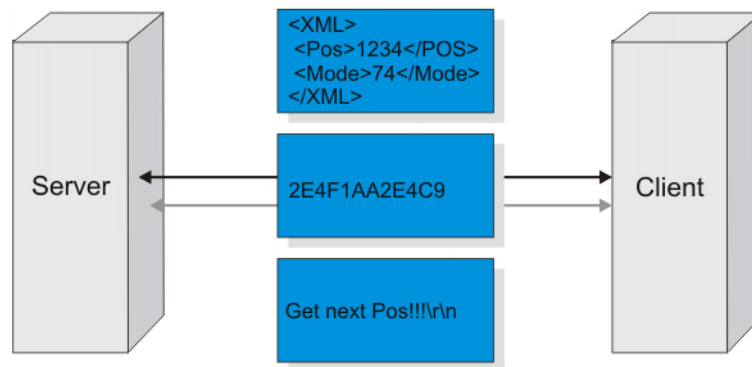


Abb. 2-6: Protokollarten

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

Folgende Kombinationen sind möglich:

Verbindung Vx	V1	V2	V3	V4	V5
Binär fest	x	-	x	-	-
Binär variabel	-	-	-	x	x
XML	x	x	-	-	x

Beispiele

F F E A 1 6 C C 0 1 2 3 B E 9 7 8 F F F

Abb. 2-7: Binär-Daten mit fester Länge (20 Byte)

H E L L O : E N D
G e t n e x t P o s ! ! ! \ R \ N

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 34)

2.8 Fehlerbehandlung

Beschreibung Für den Datenaustausch zwischen Robotersteuerung und externem System stellt Ethernet KRL Funktionen zur Verfügung.

Jede dieser Funktionen besitzt einen Rückgabewert. Der Rückgabewert kann im KRL-Programm abgefragt und ausgewertet werden.

Der Rückgabewert kann je nach Funktion folgende Informationen enthalten:

- Fehlernummer

- Anzahl der Elemente, die sich noch im Speicher befinden
- Anzahl der Elemente, die bereits aus dem Speicher gelesen wurden
- Information, ob eine Verbindung besteht

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 Körperverletzungen oder erhebliche Sachschäden können sonst die Folge sein.

4 Installation

4.1 Systemvoraussetzungen

- | | |
|-----------------|--|
| Hardware | <ul style="list-style-type: none"> ■ Robotersteuerung VKR C4 ■ Externes System |
| Software | <ul style="list-style-type: none"> ■ VW System Software 8.2 |

4.2 Ethernet KRL installieren oder updaten



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

- Voraussetzung**
- Software auf KUKA.USBData-Stick
 - Es ist kein Programm angewählt.
 - Betriebsart T1 oder T2
 - Benutzergruppe Experte

HINWEIS

Es darf ausschließlich der KUKA.USBData-Stick verwendet werden. Wenn ein anderer USB-Stick verwendet wird, können Daten verloren gehen oder verändert werden.

- Vorgehensweise**
1. USB-Stick anstecken.
 2. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware installieren** wählen.
 3. Auf **Neue Software** drücken. Wenn eine Software, die sich auf dem USB-Stick befindet, nicht angezeigt wird, auf **Aktualisieren** drücken.
 4. Den Eintrag **EthernetKRL** markieren und auf **Installieren** drücken. Sicherheitsabfrage mit **Ja** beantworten. Die Dateien werden auf die Festplatte kopiert.
 5. Wenn eine weitere Software von diesem Stick installiert werden soll, Schritt 4 wiederholen.
 6. USB-Stick entfernen.
 7. Abhängig von der Zusatzsoftware kann ein Neustart notwendig sein. In diesem Fall wird eine Aufforderung zum Neustart angezeigt. Mit **OK** bestätigen und die Robotersteuerung neu starten. Die Installation wird fortgesetzt und abgeschlossen.

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

4.3 Ethernet KRL deinstallieren



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

- Voraussetzung**
- Benutzergruppe Experte

- Vorgehensweise**
1. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware installieren** 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 die KLI der Robotersteuerung

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

Das Ethernet-Kabel kann entweder am PROFINET-Switch Cu oder direkt am Steuerungs-PC angeschlossen werden.



Weitere Informationen zur Schnittstelle PROFINET-Switch Cu sind in der Betriebs- oder Montageanleitung der Robotersteuerung zu finden.

5.2 Netzwerkverbindung konfigurieren

Voraussetzung

- Benutzergruppe Experte
- Netzwerkverbindung über das KLI der Robotersteuerung

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **HMI minimieren** wählen.
2. Im Windows-Start-Menü **Alle Programme** > **EKI-Network** wählen.
Das Fenster **Network Setup** öffnet sich. Bereits eingerichtete Netzwerkverbindungen werden in der Baumstruktur unter **Other Installed Interfaces** angezeigt.
3. In der Baumstruktur unter **Ethernet KRL** den Eintrag **New** markieren und auf **Edit** drücken.
4. Die IP-Adresse eintragen und mit **Ok** bestätigen.



Der IP-Adressenbereich 192.168.0.x ist für die Konfiguration der Netzwerkverbindung gesperrt.

5. Robotersteuerung mit einem Kaltstart neu starten.

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.

Der Name der Datei ist gleichzeitig der Zugriffsschlüssel 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 (>>> 6.1.2 "XML-Struktur für den Datenempfang" Seite 23)
<SEND> ... </SEND>	Konfiguration der Sendestruktur (>>> 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) <ul style="list-style-type: none"> ■ Server: Externes System ist ein Server. ■ Client: Externes System ist ein Client. Default-Wert: Server
IP	IP-Adresse des externen Systems (optional, wenn TYPE = Client)
PORT	Port-Nummer des externen Systems (optional, wenn TYPE = Client) <ul style="list-style-type: none"> ■ 1 ... 65 534

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

Element	Attribut	Beschreibung
ENVIRONMENT	_____	<p>Löschen der Verbindung an Aktionen koppeln (optional)</p> <ul style="list-style-type: none"> ■ Program: Löschen nach Aktionen des Roboter-Interpreters <ul style="list-style-type: none"> ■ Programm zurücksetzen. ■ Programm abwählen. ■ System: Löschen nach Systemaktionen <ul style="list-style-type: none"> ■ E/As rekonfigurieren. ■ Robotersteuerung mit Kaltstart neu starten. ■ Submit: Löschen nach Aktionen des Submit-Interpreters <ul style="list-style-type: none"> ■ Submit-Interpreter abwählen. <p>Default-Wert: Program</p>
BUFFERING	Mode	<p>Verfahren, nach dem alle Datenspeicher abgearbeitet werden (optional)</p> <ul style="list-style-type: none"> ■ FIFO: First In First Out ■ LIFO: Last In First Out <p>Default-Wert: FIFO</p>
	Limit	<p>Maximale Anzahl an Datenelementen, die ein Datenspeicher aufnehmen kann (optional)</p> <ul style="list-style-type: none"> ■ 1 ... 512 <p>Default-Wert: 16</p>
BUFSIZE	Limit	<p>Maximale Anzahl an Bytes, die empfangen werden können, ohne dass sie interpretiert werden (optional)</p> <ul style="list-style-type: none"> ■ 1 ... 65 534 Bytes <p>Default-Wert: 16 384 Bytes</p>
TIMEOUT	Connect	<p>Zeit, nach der der Versuch eine Verbindung aufzubauen abgebrochen wird (optional)</p> <p>Einheit: ms</p> <ul style="list-style-type: none"> ■ 0 ... 65 534 ms <p>Default-Wert: 2 000 ms</p>

Element	Attribut	Beschreibung
ALIVE	Set_Out	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.
	Set_Flag	
	Ping	Intervall für das Senden eines Pings, um die Verbindung zum externen System zu überwachen (optional) ■ 1 ... 65 534 s
IP	_____	IP-Adresse der Schnittstelle (optional, wenn TYPE = Server) Wenn TYPE = Client, muss die IP-Adresse hier angegeben werden.
PORT	_____	Port-Nummer der Schnittstelle (optional, wenn TYPE = Server) ■ 49 152 ... 65 534 Wenn TYPE = Client, muss die Port-Nummer hier angegeben werden.
PROTOCOL	_____	Übertragungsprotokoll (optional) ■ TCP ■ UDP Default-Wert: TCP Es wird empfohlen, immer das TCP/IP-Protokoll zu verwenden.

Beispiel

```

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

```

6.1.2 XML-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).
ELEMENT	Type	Datentyp des Elements <ul style="list-style-type: none"> ■ STRING ■ REAL ■ INT ■ BOOL ■ FRAME Hinweis: Optional, wenn das Tag nur für Ereignismeldungen verwendet wird. Dann wird kein Speicherplatz für das Element reserviert. Beispiel für Ereignis-Flag: <ELEMENT Tag="Ext" Set_Flag="56"/>
ELEMENT	Set_Out	Setzen eines Ausgangs oder eines Flags, wenn das Element empfangen wurde (optional) Nummer des Ausgangs: <ul style="list-style-type: none"> ■ 1 ... 4 096 Nummer des Flags: <ul style="list-style-type: none"> ■ 1 ... 1 025
	Set_Flag	
ELEMENT	Mode	Verfahren, nach dem ein Datensatz im Datenspeicher abgearbeitet wird <ul style="list-style-type: none"> ■ 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 <ul style="list-style-type: none"> ■ BYTE: Binär-Datensatz mit fester Länge ■ STREAM: Binär-Datensatz variabel mit Endzeichenfolge

Element	Attribut	Beschreibung
ELEMENT	Set_Out	Setzen eines Ausgangs oder eines Flags, wenn das Element empfangen wurde (optional) Nummer des Ausgangs: ■ 1 ... 4 096 Nummer des Flags: ■ 1 ... 1 025
	Set_Flag	
ELEMENT	EOS	Endzeichenfolge einer elementaren Information (nur relevant, wenn TYPE = STREAM) ■ ASCII-Kodierung: 1 ... 32 Zeichen ■ Alternatives Ende wird durch das Zeichen "I" getrennt. Beispiele: ■ <ELEMENT ... EOS="123,134,21"/> ■ <ELEMENT ... EOS="123,134,21I13,10"/>
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/Framel" 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>`
- 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

Beschreibung

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 Ethernet KRL-Funktionen für den Datenaustausch

Übersicht

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

Die genaue Beschreibung der Funktionen ist im Anhang zu finden.

(>>> 9.5 "Ethernet KRL-Funktionen Befehlsreferenz" Seite 59)

Initialisierung und Verbindung
EKI_STATUS = EKI_Init(CHAR[])
EKI_STATUS = EKI_Open(CHAR[])
EKI_STATUS = EKI_Close(CHAR[])
EKI_STATUS = EKI_Clear(CHAR[])
Senden
EKI_STATUS = EKI_Send(CHAR[], CHAR[])
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[])
Datenzugriff
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[])
Fehlerbehandlung
EKI_CHECK(EKI_STATUS, EKrIMsgType, CHAR[])

Sonstige

EKI_STATUS = EKI_ClearBuffer(CHAR[], CHAR[])

EKI_STATUS = EKI_Lock(CHAR[])

EKI_STATUS = EKI_Unlock(CHAR[])

6.2.1 Programmiertipps



Es wird empfohlen, den Datenaustausch über die Ethernet KRL-Schnittstelle ausschließlich im Modul VW_USER zu programmieren und die Ethernet KRL-Funktionen nicht in Folgen zu verwenden.

- 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.
- Ethernet KRL kann 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.

6.2.2 Initialisieren und Löschen einer Verbindung

Beschreibung

Eine Verbindung muss mit der Funktion EKI_Init() initialisiert werden. Die in der Funktion angegebene Verbindungskonfiguration wird dabei eingelesen.

Das Löschen einer Verbindung kann über das Element <ENVIRONMENT...> in der Verbindungskonfiguration an Roboter- und Submit-Interpreter-Aktionen oder Systemaktionen gekoppelt sein.

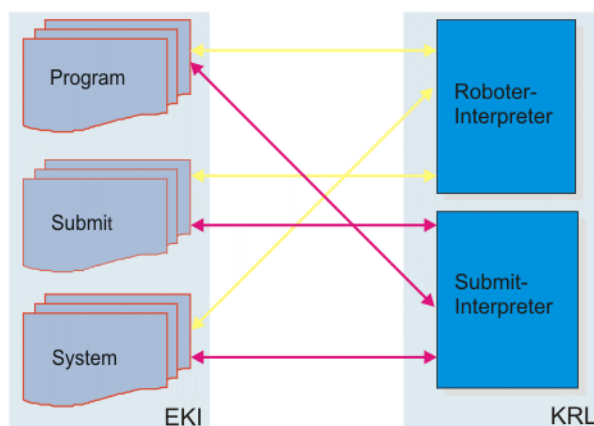


Abb. 6-1: Verbindungskonfiguration

Je nach Verbindungskonfiguration wird eine Verbindung nach folgenden Aktionen gelöscht:

- Konfiguration "**Program**"
 - Programm zurücksetzen.
 - Programm abwählen.
- Konfiguration "**Submit**"
 - Submit-Interpreter abwählen.
- Konfiguration "**System**"
 - Robotersteuerung mit Kaltstart neu starten.
 - 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 die Schnittstelle als Server konfiguriert ist, versetzt EKI_Open() den Server in einen Abhörungszustand. Der Server erwartet 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 die Schnittstelle als Client 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:

```
<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. Es sind nur Daten vom Typ REAL (4 Bytes) lesbar, kein Double.



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)
```

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.

Beispiel

Die Position des zu löschenden Speichers wird in XPATH angegeben.

```

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 Rückgabewert der Ethernet KRL-Funktionen

EKI_STATUS ist eine globale Variable, die den aktuellen Status der Schnittstelle in Bezug auf eine Ethernet KRL-Funktion enthält.

Syntax

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

Erläuterung der Syntax

Element	Beschreibung
Buff	Anzahl der Elemente, die sich im Speicher befinden
Read	Anzahl der Elemente, die aus dem Speicher gelesen wurden
Msg_No	Fehlernummer des Fehlers, der beim Datenempfang oder beim Aufruf der Funktion aufgetreten ist Mit EKI_CHECK() kann eine Meldung zum Fehler auf der smartHMI ausgegeben werden.
Connected	Information, ob eine Verbindung besteht <ul style="list-style-type: none"> ■ TRUE = Es besteht eine Verbindung. ■ FALSE = Es besteht keine Verbindung.

Welche Elemente der Struktur beschrieben werden ist abhängig von der Ethernet KRL-Funktion:

EKI_	Buff	Read	Msg_No	Connected
Get...()	x	x	x	x
Set...()	-	-	x	x
Send()	-	-	x	x
Init()	-	-	x	-
Sonstige	-	-	x	x

Beispiel

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

Der Rückgabewert der Funktion EKI_Open() wird ausgewertet. Wenn der Kanal nicht geöffnet und keine Datenverbindung hergestellt werden kann, wird ein Fehlerwert ausgelesen und eine Quittiermeldung angezeigt.

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

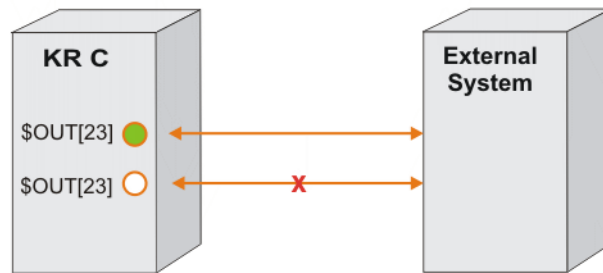


Abb. 6-2: Ereignis-Ausgang (Verbindung aktiv)

\$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

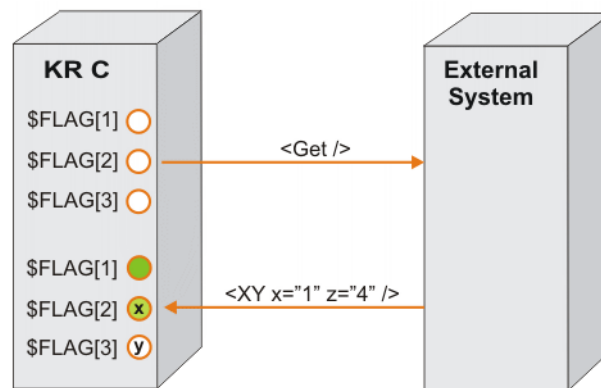


Abb. 6-3: Ereignis-Flag (Vollständige XML-Struktur)

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

(>>> 7.2.5 "Beispielkonfiguration XmlCallback" Seite 47)

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 Fehlerbehandlung**Beschreibung**

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

(>>> 9.3 "Anzeige von Meldungen auf der smartHMI deaktivieren" Seite 58)

Wenn die automatische Meldungs Ausgabe deaktiviert wurde, wird in jedem Fall empfohlen, eine EKI-Anweisung mit EKI_CHECK() zu prüfen. Mit der Funktion EKI_CHECK() kann eine Fehlernummer ausgelesen und die Meldung zu einem Fehler auf der smartHMI angezeigt werden. Wird in EKI_CHECK() ein Kanalname angegeben, wird beim Datenempfang abgefragt, ob Fehler vorliegen.

Bei jedem Aufruf von EKI_CHECK() wird das Programm EthernetKRL_USER.SRC aufgerufen. Die Datei befindet sich im Verzeichnis KRC:\R1\TP\EthernetKRL. In der Datei 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

Ethernet KRL beinhaltet Beispielapplikationen, mit denen eine Kommunikation zwischen einem Server-Programm und der Robotersteuerung hergestellt werden kann. Die Software befindet sich auf dem KUKA.USBData-Stick im Verzeichnis DOC\Example.

Die Software besteht aus folgenden Komponenten:

Komponente	Ordner
Server-Programm EthernetKRL_Server.exe	...\Application
Beispielprogramme in KRL <ul style="list-style-type: none"> ■ BinaryFixed.src ■ BinaryStream.src ■ XmlCallback.src ■ XmlServer.src ■ XmlTransmit.src 	...\Program
Beispielkonfigurationen in XML <ul style="list-style-type: none"> ■ BinaryFixed.xml ■ BinaryStream.xml ■ XmlCallBack.xml ■ XmlServer.xml ■ XmlTransmit.xml ■ XmlFullConfig.xml 	...\Config

7.1.1 Beispielapplikationen implementieren

Voraussetzung

Externes System:

- Betriebssystem Windows mit installiertem .NET-Framework

Robotersteuerung:

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

1. Server-Programm auf externes System kopieren.
2. Die SRC-Dateien in das Modul VW_USER einbinden.
(>>> 7.1.2 "Beispielprogramm in Modul VW_USER einbinden" Seite 38)
3. Die XML-Dateien in das Verzeichnis C:\KRC\ROBOTER\Config\User\Common\EthernetKRL der Robotersteuerung kopieren.
4. Server-Programm auf dem externen System starten.
(>>> 7.1.3 "Bedienoberfläche Server-Programm" Seite 39)
5. Menü-Button drücken. Das Fenster **Communication Properties** öffnet sich.
(>>> 7.1.4 "Kommunikationsparameter im Server-Programm einstellen" Seite 41)
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 Beispielprogramm in Modul VW_USER einbinden

- Voraussetzung**
- Benutzergruppe Experte
 - Betriebsart T1 oder T2

- Vorgehensweise**
1. Neues Programm (Folge) anlegen.
 2. In der Punkt-SPS der PTP-Bewegung vor Folgenstart den VW_USER aufrufen.
 3. Im Inline-Formular **VW User** für das gewünschte Beispielprogramm einen Parameter definieren, der an das Modul VW_USER übergeben wird.
 4. Im Unterprogramm USER_MAIN() der Datei VW_USR_R.SRC das Beispielprogramm über eine IF-Anweisung aufrufen, wenn der definierte Parameter übergeben wird.
 5. Am Ende der Datei VW_USR_R.SRC das Beispielprogramm als Unterprogramm einfügen.

Beispiel Im Programm FOLGE1.SRC wird der VW_USER aufgerufen und der Parameter X=12345 an das Modul VW_USER übergeben.

```
1 DEF FOLGE1()  
2   PTP VB=100% VE=0% ACC=100% RobWzg=0 Base=0 SPSTrig=0[1/100s] P  
3   1: VW USER X=12345 [mm] P2= 1 P3=1 P4=1 P5=1 P6=1 P7=EIN  
4   Warte auf Folgenstart  
...
```

Im Unterprogramm USER_MAIN() der Datei VW_USR_R.SRC wird überprüft, ob der Parameter X=12345 (= PAR1) übergeben wurde. Wenn dies der Fall ist, wird das Unterprogramm BinaryFixed() aufgerufen.

```

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

```

Zeile	Beschreibung
24 ... 26	Bedingung für den Aufruf des Unterprogramms BinaryFixed(): PAR1=12345
38 ... 61	Unterprogramm BinaryFixed()

7.1.3 Bedienoberfläche Server-Programm

Beschreibung

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

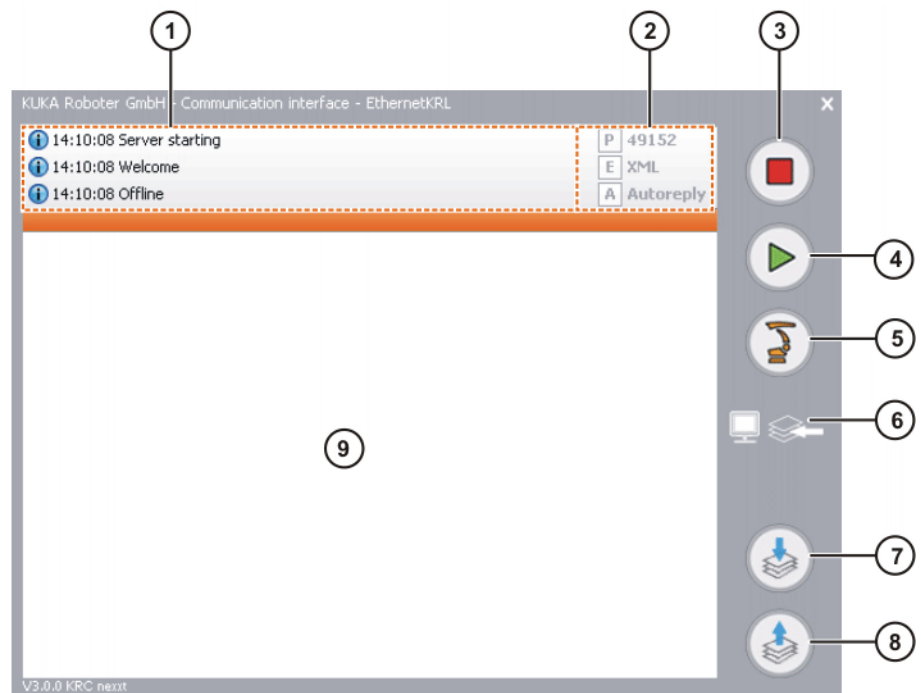


Abb. 7-1: Server-Programm Bedienoberfläche

Pos.	Beschreibung
1	Meldungsfenster
2	<p>Anzeige der eingestellten Kommunikationsparameter (>>> 7.1.4 "Kommunikationsparameter im Server-Programm einstellen" Seite 41)</p> <ul style="list-style-type: none"> ■ P: Port-Nummer ■ E: Beispieldaten <ul style="list-style-type: none"> ■ Xml: XML-Daten ■ BinaryFixed: Binär-Daten mit fester Länge ■ BinaryStream: Binär-Datenstrom variabel mit Endzeichenfolge ■ A: Kommunikationsmodus <ul style="list-style-type: none"> ■ Autoreply: Der Server beantwortet jedes empfangene Datenpaket automatisch. ■ Manual: Nur manueller Datenempfang oder Datenversand
3	<p>Stopp-Button</p> <p>Die Kommunikation mit der Robotersteuerung wird beendet und der Server wird zurückgesetzt.</p>
4	<p>Start-Button</p> <p>Der Datenaustausch zwischen Server-Programm und Robotersteuerung wird ausgewertet. Die erste eingehende Verbindungsanfrage wird gebunden und als Kommunikationsadapter benutzt.</p>
5	<p>Menü-Button zum Einstellen der Kommunikationsparameter (>>> 7.1.4 "Kommunikationsparameter im Server-Programm einstellen" Seite 41)</p>

Pos.	Beschreibung
6	Anzeigeoptionen <ul style="list-style-type: none"> ■ 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.

7.1.4 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. <ul style="list-style-type: none"> ■ Xml: XML-Daten ■ BinaryFixed: Binär-Daten mit fester Länge ■ BinaryStream: Binär-Datenstrom variabel mit Endzeichenfolge Default-Wert: xml
Autoresponder	Kommunikationsmodus auswählen. <ul style="list-style-type: none"> ■ Autoreply: Der Server beantwortet jedes empfangene Datenpaket automatisch. ■ Manual: Nur manueller Datenempfang oder Datenversand Default-Wert: Autoreply

Element	Beschreibung
Portnummer	<p>Port-Nummer der Socket-Verbindung eingeben.</p> <p>An diesem Port erwartet das externe System die Verbindungsanfrage der Robotersteuerung. Es muss eine freie Nummer gewählt werden, die nicht als Standarddienst belegt ist.</p> <p>Default-Wert: 49152</p>
Network interface card index	<p>Nummer des Netzwerkadapters eingeben.</p> <p>Nur relevant, wenn das externe System mehrere Netzwerkkarten benutzt, z. B. WLAN und LAN.</p> <p>Default-Wert: 0</p>

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>49152</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 sein. Wenn die Schnittstelle dieses Element empfangen hat, wird \$FLAG[1] gesetzt.

XML-Datei

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

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

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

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 connected
9 wait for $FLAG[1]
10 ; wait until server is disconnected
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>49152</PORT>
    </EXTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <XML>
      <ELEMENT Tag="Sensor/Message" Type="STRING" />
      <ELEMENT Tag="Sensor/Positions/Current/@X" Type="REAL" />
      <ELEMENT Tag="Sensor/Positions/Before/X" Type="REAL" />
      <ELEMENT Tag="Sensor/Nmb" Type="INT" />
      <ELEMENT Tag="Sensor/Status/IsActive" Type="BOOL" />
      <ELEMENT Tag="Sensor/Read/xyzabc" Type="FRAME" />
      <ELEMENT Tag="Sensor/Show/@error" Type="BOOL" />
      <ELEMENT Tag="Sensor/Show/@temp" Type="INT" />
      <ELEMENT Tag="Sensor/Show" Type="STRING" />
      <ELEMENT Tag="Sensor/Free" Type="INT" />
      <ELEMENT Tag="Sensor/Show/@flag" Type="BOOL" Set_Flag="999" />
      <ELEMENT Tag="Sensor/Show/@out" Type="INT" Set_Out="999" />
      <ELEMENT Tag="Sensor" Set_Flag="998" />
    </XML>
  </RECEIVE>
  <SEND>
    <XML>
      <ELEMENT Tag="Robot/Data/LastPos/@X" />
      <ELEMENT Tag="Robot/Data/LastPos/@Y" />
      <ELEMENT Tag="Robot/Data/LastPos/@Z" />
      <ELEMENT Tag="Robot/Data/LastPos/@A" />
      <ELEMENT Tag="Robot/Data/LastPos/@B" />
      <ELEMENT Tag="Robot/Data/LastPos/@C" />
      <ELEMENT Tag="Robot/Status" />
      <ELEMENT Tag="Robot/Mode" />
      <ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn" />
      <ELEMENT Tag="Robot/Data/ActPos/@X" />
    </XML>
  </SEND />
</ETHERNETKRL>
```

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

```

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 zu Ethernet KRL anzeigen

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

Beschreibung Diagnosedaten zu Ethernet KRL:

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)

8.2 Fehlerprotokoll (EKI-Logbuch)

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

8.3 Fehlermeldungen

Jede Ethernet KRL-Funktion besitzt einen Rückgabewert EKI_STATUS, der eine Fehlernummer enthält. 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	<i>Unbekannter Fehler</i>	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	<i>Der Systemspeicher ist verbraucht</i>	Der für Ethernet KRL reservierte Speicher ist vollständig belegt. Es können keine weiteren Elemente gespeichert werden.	Programmierung 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 57)
3	<i>Zugriff auf Datei fehlgeschlagen</i>	Eine Datei konnte nicht gefunden werden oder ist nicht lesbar.	Überprüfen, ob die Datei vorhanden ist oder ob die Datei sich öffnen lässt.
4	<i>Angeforderte Funktion nicht implementiert</i>	Software-Fehler: Die verwendete Ethernet KRL-Funktion ist nicht implementiert.	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
5	<i>Fehler bei Erstellen des XML Parsers</i>	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	<i>Interpretieren der Konfiguration fehlgeschlagen</i>	Fehler beim Lesen der Verbindungskonfiguration	Konfiguration der Ethernet-Verbindung überprüfen.
7	<i>Schreiben der Daten zum Senden fehlgeschlagen</i>	Fehler beim Beschreiben der XML-Struktur für den Datenversand	Konfiguration der Sendestruktur überprüfen.
8	<i>Neues Element konnte nicht angelegt werden</i>	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	<i>Verbindung nicht vorhanden</i>	Wegen fehlender Initialisierung ist kein Zugriff auf die Verbindung möglich.	Ethernet-Verbindung mit EKI_Init() initialisieren.
10	<i>Ethernet ist getrennt</i>	Es ist keine Ethernet-Verbindung vorhanden.	Ethernet-Verbindung mit EKI_Open() öffnen.
11	<i>Ethernetverbindung zu externem System bereits vorhanden</i>	Ethernet-Verbindung ist bereits vorhanden.	Funktion EKI_Open() nicht aufrufen, wenn Ethernet-Verbindung bereits vorhanden ist.

Nr.	Meldungstext	Ursache	Abhilfe
12	<i>Erstellen des Servers fehlgeschlagen</i>	Eine Ethernet-Verbindung, die als Server konfiguriert ist, konnte nicht erstellt werden.	Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
13	<i>Ethernetparameter konnten nicht initialisiert werden</i>	Fehler beim Initialisieren der Ethernet-Verbindung	Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
14	<i>Ethernetverbindung zu externem System konnte nicht hergestellt werden</i>	Keine Ethernet-Verbindung: <ul style="list-style-type: none"> ■ Hardware-Fehler, z. B. Netzkabel, Switch, externes System ■ Software-Fehler (externes System) ■ Fehler bei der Verbindungskonfiguration 	Ethernet-Verbindung herstellen: <ul style="list-style-type: none"> ■ Hardware überprüfen. ■ Software des externen Systems überprüfen. ■ Konfiguration der Verbindungsparameter überprüfen. (Elemente IP, PORT)
15	<i>Zugriff auf leeren Empfangsspeicher</i>	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 "Rückgabewert der Ethernet KRL-Funktionen" Seite 34)
16	<i>Element konnte nicht gefunden werden</i>	Ein in der Zugriffsfunktion EKI_Get...() angegebenes Datenelement kann nicht gefunden werden.	<ul style="list-style-type: none"> ■ Name des Datenelements und seine Schreibweise im KRL-Programm überprüfen. ■ Konfiguration der Empfangsstruktur überprüfen.
17	<i>Zusammenstellen der Daten zum Senden fehlgeschlagen</i>	<ul style="list-style-type: none"> ■ 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 	<ul style="list-style-type: none"> ■ Senden von XML-Daten: Konfiguration der Sendestruktur überprüfen. ■ Senden von Binär-Daten: Funktion EKI_Send() im KRL-Programm überprüfen.
18	<i>Senden von Daten fehlgeschlagen</i>	Keine Ethernet-Verbindung: <ul style="list-style-type: none"> ■ Hardware-Fehler, z. B. Netzkabel, Switch, externes System ■ Software-Fehler (externes System) 	Ethernet-Verbindung herstellen: <ul style="list-style-type: none"> ■ Hardwareüberprüfen. ■ Software des externen Systemsüberprüfen.
19	<i>Keine Daten zum Senden vorhanden</i>	In einer Funktion EKI_Send() sind die zu sendenden Daten nicht angegeben.	Funktion EKI_Send() im KRL-Programm überprüfen.

Nr.	Meldungstext	Ursache	Abhilfe
20	<i>Datentypen passen nicht zusammen</i>	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	<i>Mit maximaler Datenhaltung Systemspeicher nicht ausreichend</i>	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 57)
22	<i>Fehler bei Lesen der Konfiguration. XML Fehler.</i>	Beim Einlesen der Konfiguration wurde ein Fehler in der XML-Struktur festgestellt.	XML-Struktur in der Konfigurationsdatei überprüfen.
24	<i>Bindung an interne Parameter (Port,IP) fehlgeschlagen</i>	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	<i>Interner Softwarefehler</i>	Interner Software-Fehler	KUKA Roboter GmbH kontaktieren und Logbuch mit den Details zum Fehler zur Verfügung stellen. (>>> 10 "KUKA Service" Seite 69)
26	<i>Das FRAME-Feld ist nicht initialisiert</i>	Ein Feld vom Typ FRAME wurde nicht initialisiert.	Feld vom Typ FRAME initialisieren (Wert zuweisen).
27	<i>Das KRL CHAR[] Feld ist zu klein.</i>	Ein Feld vom Typ CHAR ist zu klein.	Anzahl der Feld-Elemente erhöhen.
512	<i>Ethernetverbindung gestört</i>	Keine Ethernet-Verbindung: <ul style="list-style-type: none">■ Hardware-Fehler, z. B. Netzkabel, Switch, externes System■ Software-Fehler (externes System)	Ethernet-Verbindung wiederherstellen: <ul style="list-style-type: none">■ Hardware überprüfen.■ Software des externen Systems überprüfen.
768	<i>Ping meldet kein Kontakt</i>	Das externe System antwortet nicht mehr auf das gesendete Ping. Die Verbindung ist abgebrochen.	Externes System überprüfen.

Nr.	Meldungstext	Ursache	Abhilfe
1024	<i>Fehler bei Lesen empfangener XML-Daten</i>	Ein vom externen System empfangenes XML-Dokument entspricht nicht dem XPath-Schema.	Das vom externen System gesendete XML-Dokument überprüfen.
1280	<i>Grenze speicherbarer Elemente erreicht</i>	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 "Rückgabewert der Ethernet KRL-Funktionen" Seite 34) ODER Datenspeicher erhöhen. (Element BUFFERING in der Verbindungskonfiguration)
1536	<i>Empfangene Zeichenkette zu lang</i>	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	<i>Limit Empfangsspeicher erreicht</i>	Der Datenspeicher ist mit der maximalen Anzahl an Bytes belegt. Die Ethernet-Verbindung wird geschlossen.	Datenspeicher erhöhen. (Element BUFFSIZE in der Verbindungskonfiguration)
2048	<i>Server Zeitgrenze erreicht</i>	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 Anzeige von Meldungen auf der smartHMI deaktivieren

Beschreibung Das Prüfen auf Fehler und die automatische Ausgabe von Meldungen auf der smartHMI kann deaktiviert werden. Dies wird empfohlen, wenn Laufzeitprobleme auftreten oder wenn die EKI im Submit-Interpreter verwendet wird.

Wenn die automatischen Meldungen deaktiviert sind, kann die Funktion EKI_CHECK() verwendet werden, um einzelne EKI-Anweisungen auf Fehler zu überprüfen.

Voraussetzung ■ Benutzergruppe Experte

Vorgehensweise

1. Datei KRC:\R1\TP\EthernetKRL\EthernetKRL.DAT öffnen.
2. Variable SHOWMSG auf FALSE setzen.
3. Änderung speichern und Datei schließen.



Die Deaktivierung betrifft nur die auf der smartHMI angezeigten Meldungen. Das EKI-Logbuch wird weiterhin mit Warnhinweisen und Fehlermeldungen beschrieben.

9.4 Warnmeldungen im EKI-Logbuch deaktivieren

Beschreibung Wenn die automatische Meldungsanzeige auf der smartHMI deaktiviert ist, werden weiterhin alle Warnungen und Fehler in das EKI-Logbuch geschrieben. Wenn diese Fehler oder Warnungen bewusst ignoriert werden sollen, ist dieser Mechanismus zu deaktivieren. (Ein Zugriff auf Dateien erzeugt sonst eine unnötige Systembelastung.)

Voraussetzung ■ Windows-Ebene

Vorgehensweise

1. Datei C:\KRC\ROBOTERConfig\User\Common\Logging_EthernetKRL.XML öffnen.
2. Das Attribut **LogLevel** im XML-Element **Class** ändern.
3. Änderung speichern und Datei schließen.

Folgende Werte stehen zur Verfügung:

LogLevel	Beschreibung
warning	Warnmeldungen und Fehlermeldungen werden in das Logbuch geschrieben. (Default)
error	Nur Fehlermeldungen werden in das Logbuch geschrieben.
disabled	Es werden keine Warn- oder Fehlermeldungen mehr in das Logbuch geschrieben.

9.5 Ethernet KRL-Funktionen Befehlsreferenz

9.5.1 Funktionen zur Initialisierung und Verbindung

RET = EKI_Init(CHAR[])	
Funktion	<p>Initialisiert einen Kanal für die Ethernet-Kommunikation</p> <p>Folgende Aktionen werden ausgeführt:</p> <ul style="list-style-type: none"> ■ Einlesen der Konfiguration ■ Erstellen der Datenspeicher ■ Vorbereiten der Ethernet-Verbindung
Parameter	<p>Typ: CHAR</p> <p>Name des Kanals</p>
RET	<p>Typ: EKI_STATUS</p> <p>Rückgabewert, der die Meldungsnummer des Fehlers enthält</p> <p>(>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)</p>
Beispiel	RET = EKI_Init("Channel_1")

RET = EKI_Open(CHAR[])	
Funktion	<p>Öffnet einen initialisierten Kanal</p> <p>Wenn die Ethernet KRL-Schnittstelle als Client konfiguriert ist, verbindet sich die Schnittstelle mit dem Server.</p> <p>Wenn die Ethernet KRL-Schnittstelle als Server konfiguriert ist, wartet die Schnittstelle auf die Verbindung.</p>
Parameter	<p>Typ: CHAR</p> <p>Name des Kanals</p>
RET	<p>Typ: EKI_STATUS</p> <p>Rückgabewert, der die Meldungsnummer des Fehlers enthält</p> <p>(>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)</p>
Beispiel	RET = EKI_Open("Channel_1")

RET = EKI_Close(CHAR[])	
Funktion	Schließt einen geöffneten Kanal
Parameter	<p>Typ: CHAR</p> <p>Name des Kanals</p>
RET	<p>Typ: EKI_STATUS</p> <p>Rückgabewert, der die Meldungsnummer des Fehlers enthält</p> <p>(>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)</p>
Beispiel	RET = EKI_Close("Channel_1")

RET = EKI_Clear(CHAR[])	
Funktion	Löscht einen Kanal und beendet die Verbindung.
Parameter	<p>Typ: CHAR</p> <p>Name des Kanals</p>

RET = EKI_Clear(Char[])	
RET	Typ: EKI_STATUS Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
Beispiel	RET = EKI_Clear("Channel_1")

9.5.2 Sendefunktion

RET = EKI_Send(Char[], Char[])	
Funktion	Sendet eine XML-Struktur oder Rohdaten (>>> 6.2.4 "Senden von Daten" Seite 30)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
Beispiel 1	RET = EKI_Send("Channel_1", "Root/Test")
Beispiel 2	RET = EKI_Send("Channel_1", MyBytes[])

9.5.3 Schreibfunktionen

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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
Beispiel	RET = EKI_SetString("Channel_1", "Root/Message", "Hello")

9.5.4 Zugriffsfunktionen

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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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

RET = EKI_GetBoolArray(Char[], Char[], Bool[])	
Parameter 3	Typ: BOOL Feld, das aus dem Speicher gelesen wird Maximale Anzahl lesbarer Feld-Elemente: ■ 512
RET	Typ: EKI_STATUS Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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	Typ: EKI_STATUS Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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
Parameter 3	Typ: CHAR Zeichenfolge, die aus dem Speicher gelesen wird Maximale Zeichen-Anzahl: ■ 3 600

RET = EKI_GetString(CHAR[], CHAR[], CHAR[])	
RET	Typ: EKI_STATUS Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
Beispiel	RET = EKI_GetFrameArray("Channel_1", "Root/TCP", MyFrame[])

9.5.5 Funktion zur Fehlerbehandlung

EKI_CHECK(EKI_STATUS, EKIMsgType, CHAR[])	
Funktion	Gibt eine Meldung zur Fehlernummer in Parameter 1 aus oder prüft auf Fehler, wenn ein Kanalname in Parameter 3 angegeben wurde. Die Meldung zum Fehler wird im Meldungsfenster angezeigt.
Parameter 1	Rückgabewert einer Ethernet KRL-Funktion (>>> 6.2.7 "Rückgabewert der Ethernet KRL-Funktionen" Seite 34)
Parameter 2	Typ: ENUM Meldungstyp, der zum Fehler angezeigt wird <ul style="list-style-type: none"> ■ #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")

9.5.6 Sonstige Funktionen

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 33)
RET	Typ: EKI_STATUS Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)
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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)

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 Rückgabewert, der die Meldungsnummer des Fehlers enthält (>>> 9.5.5 "Funktion zur Fehlerbehandlung" Seite 66)

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	<p>Zur Abwicklung einer Anfrage werden folgende Informationen benötigt:</p> <ul style="list-style-type: none"> ■ Typ und Seriennummer des Roboters ■ Typ und Seriennummer der Steuerung ■ Typ und Seriennummer der Lineareinheit (optional) ■ Version der VW System Software ■ Optionale Software oder Modifikationen ■ Archiv der Software <p>Für VW System Software V8: Statt eines herkömmlichen Archivs das spezielle Datenpaket für die Fehleranalyse erzeugen (über KrcDiag).</p> <ul style="list-style-type: none"> ■ 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	<p>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</p>
Australien	<p>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</p>

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. Avenida Franz Liszt, 80 Parque Novo Mundo Jd. Guançã CEP 02151 900 São Paulo SP Brasilien Tel. +55 11 69844900 Fax +55 11 62017883 info@kuka-roboter.com.br
Chile	Robotec S.A. (Agency) Santiago de Chile Chile Tel. +56 2 331-5951 Fax +56 2 331-5952 robotec@robotec.cl www.robotec.cl
China	KUKA 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 Automatismes + 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. Daiba Garden City Building 1F 2-3-5 Daiba, Minato-ku Tokyo 135-0091 Japan Tel. +81 3 6380-7311 Fax +81 3 6380-7312 info@kuka.co.jp
Korea	KUKA Robotics Korea Co. Ltd. RIT Center 306, Gyeonggi Technopark 1271-11 Sa 3-dong, Sangnok-gu Ansan City, Gyeonggi Do 426-901 Korea Tel. +82 31 501-1451 Fax +82 31 501-1461 info@kukakorea.com

Malaysia	<p>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</p>
Mexiko	<p>KUKA de Mexico S. de R.L. de C.V. Rio San Joaquin #339, Local 5 Colonia Pensil Sur C.P. 11490 Mexico D.F. Mexiko Tel. +52 55 5203-8407 Fax +52 55 5203-8148 info@kuka.com.mx</p>
Norwegen	<p>KUKA Sveiseanlegg + Roboter Sentrumsvegen 5 2867 Hov Norwegen Tel. +47 61 18 91 30 Fax +47 61 18 62 00 info@kuka.no</p>
Österreich	<p>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</p>
Polen	<p>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</p>

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	<p>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</p>
Taiwan	<p>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</p>
Thailand	<p>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</p>
Tschechien	<p>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</p>
Ungarn	<p>KUKA Robotics Hungaria Kft. Fő út 140 2335 Taksony Ungarn Tel. +36 24 501609 Fax +36 24 477031 info@kuka-robotics.hu</p>

USA

KUKA Robotics Corp.
22500 Key Drive
Clinton Township
48036
Michigan
USA
Tel. +1 866 8735852
Fax +1 586 5692087
info@kukarobotics.com
www.kukarobotics.com

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

A

Anhang 57

B

Befehlsreferenz 59

Begriffe, verwendete 6

Beispielapplikationen 37

Beispielapplikationen, implementieren 37

Beispiele 37

Beispielkonfigurationen 42

Beispielprogramme 42

C

CAST_FROM() 32, 42

CAST_TO() 30, 42

Client-Betrieb 12, 30

D

Datenaustausch 10

Datenspeicherung 11

Datenstrom 6

Defragmentierung 35

Deinstallieren, Ethernet KRL 17

Diagnose 51

Diagnosemonitor (Menüpunkt) 51

Dokumentation, Industrieroboter 5

E

Eigenschaften 9

Einleitung 5

EKI 6

EKI_CHECK() 36, 66

EKI_Clear() 33, 59

EKI_ClearBuffer() 33, 66

EKI_Close() 30, 59

EKI_GetBool() 62

EKI_GetBoolArray() 62

EKI_GetFrame() 65

EKI_GetFrameArray() 65

EKI_GetInt() 63

EKI_GetIntArray() 63

EKI_GetReal() 64

EKI_GetRealArray() 64

EKI_GetString() 32, 64

EKI_Init() 28, 59

EKI_Lock() 66

EKI_Open() 29, 30, 59

EKI_Send() 30, 60

EKI_SetBool() 61

EKI_SetFrame() 61

EKI_SetInt() 61

EKI_SetReal() 60

EKI_SetString() 62

EKI_STATUS, Rückgabewert 34

EKI_Unlock() 67

EKI-Logbuch, Warnmeldungen deaktivieren 58

Endzeichenfolge 6

EOS 6

Ereignismeldungen 13, 34

Ethernet 6

Ethernet KRL, Übersicht 9

Ethernet-Verbindung, Konfiguration 9, 21

Ethernet, Schnittstellen 19

EthernetKRL_Server.exe 37

F

Fehlerbehandlung 13

Fehlermeldungen 51

Fehlerprotokoll 51

FIFO 6, 11

Fragmentierung 35

Funktionen 9

H

Hardware 17

Hinweise 5

I

Installation 17

Installieren, Ethernet KRL 17

IP 7

K

Kenntnisse, benötigte 5

KLI 6, 19

Kommunikation 9

Konfiguration 19

Konfiguration, Ethernet-Verbindung 9, 21

KR C 6

KRL 6

KRL-Funktionen, Übersicht 27

KRL-Programm, Beispiele 37

KUKA Customer Support 69

L

LIFO 6, 11, 35

Logbuch 51

M

Meldungsanzeige, smartHMI, deaktivieren 58

N

Netzwerkverbindung 19

Netzwerkverbindung, konfigurieren 19

P

Ping 10

Produktbeschreibung 9

Programmiertipps 28

Programmierung 21

Protokollarten 12

R

Rückgabewert, EKI_STATUS 34

S

Schulungen 5
Server-Betrieb 12, 29
Server-Programm 37
Server-Programm, Bedienoberfläche 39
Server-Programm, Kommunikationsparameter
einstellen 41
Service, KUKA Roboter 69
Sicherheit 15
Sicherheitshinweise 5
smarHMI 6
Socket 6
Software 17
Speicher, erhöhen 57
Support-Anfrage 69
Systemvoraussetzungen 17

T

TCP/IP 6

U

UDP/IP 6
Updaten, Ethernet KRL 17

Ü

Übersicht, Ethernet KRL 9
Übersicht, KRL-Funktionen 27
Überwachen, Verbindung 10

V

Verbindung, überwachen 10
Verbindungsverlust 9
Verwendete Begriffe 6
VW_USER, Beispielpogramm einbinden 38

W

Warenzeichen 7
Warnmeldungen, EKI-Logbuch, deaktivieren 58

X

XML 7
XML-Datei, Beispiele 37
XPath 7, 24, 26

Z

Zielgruppe 5

