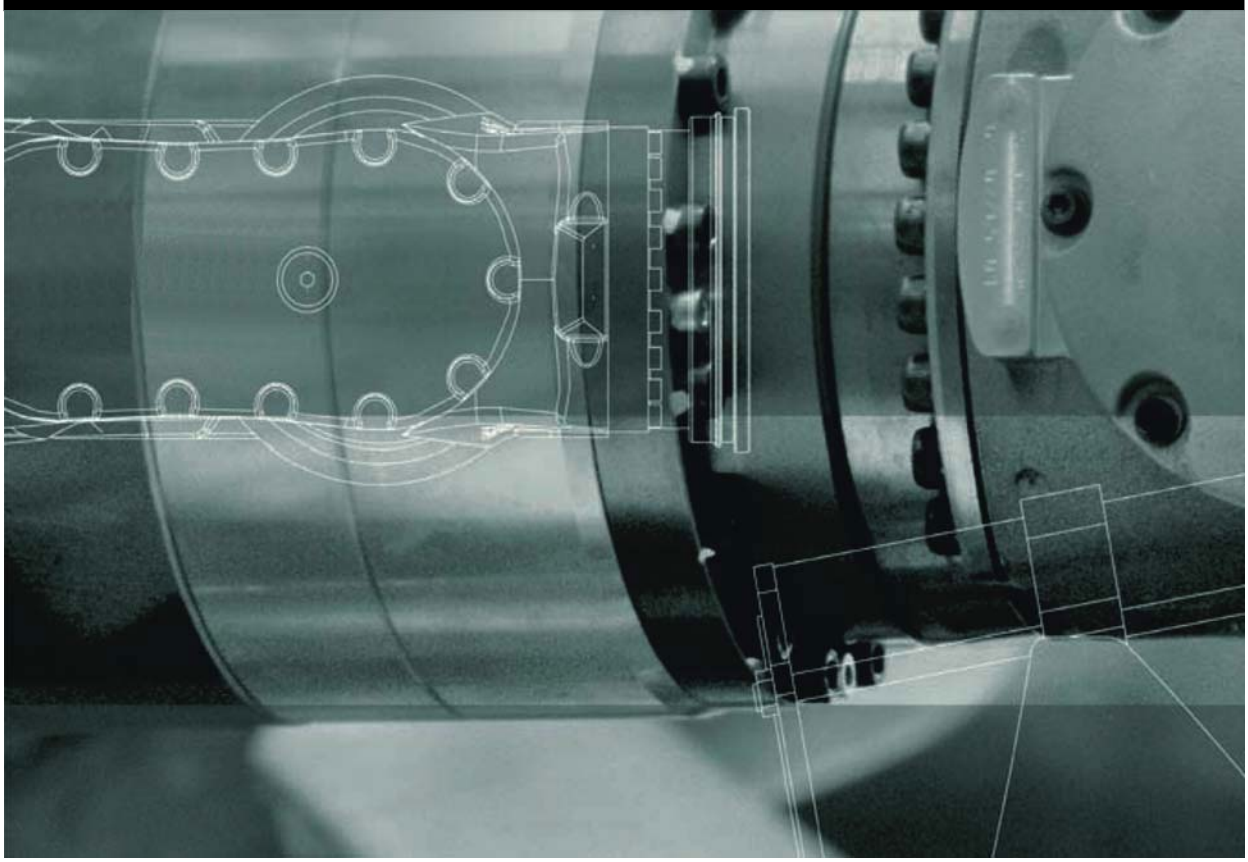
KUKA

KUKA System Software

KUKA Roboter GmbH

KUKA System Software 8.3

Bedien- und Programmieranleitung für Systemintegratoren



Stand: 04.12.2014

Version: KSS 8.3 SI V4

KUKA Roboter GmbH
Zugspitzstraße 140
D-86165 Augsburg
Deutschland

Es können weitere, in dieser Dokumentation nicht beschriebene Funktionen in der Steuerung lauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei Neulieferung oder 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 KSS 8.3 SI (PDF) de
Buchstruktur:	KSS 8.3 SI V4.3
Version:	KSS 8.3 SI V4

Inhaltsverzeichnis

1 Einleitung

1.1 Zielgruppe

Diese Dokumentation richtet sich an Benutzer mit folgenden Kenntnissen:

- Fortgeschrittene Systemkenntnisse der Robotersteuerung
- Fortgeschrittene KRL-Programmierkenntnisse

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 System Software
- Anleitungen zu Optionen und Zubehör
- Teilekatalog auf Datenträger

Jede Anleitung ist ein eigenes Dokument.

1.3 Darstellung von Hinweisen

Sicherheit

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

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

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

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

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

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

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

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

Hinweise

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

Hinweis zur Arbeitserleichterung oder Verweis auf weiterführende Informationen.

1.4 Marken

Windows ist eine Marke der Microsoft Corporation.

WordPad ist eine Marke der Microsoft Corporation.

2 Produktbeschreibung

2.1 Übersicht des Industrieroboters

Der Industrieroboter besteht aus folgenden Komponenten:

- Manipulator
- Robotersteuerung
- Programmierhandgerät
- Verbindungsleitungen
- Software
- Optionen, Zubehör

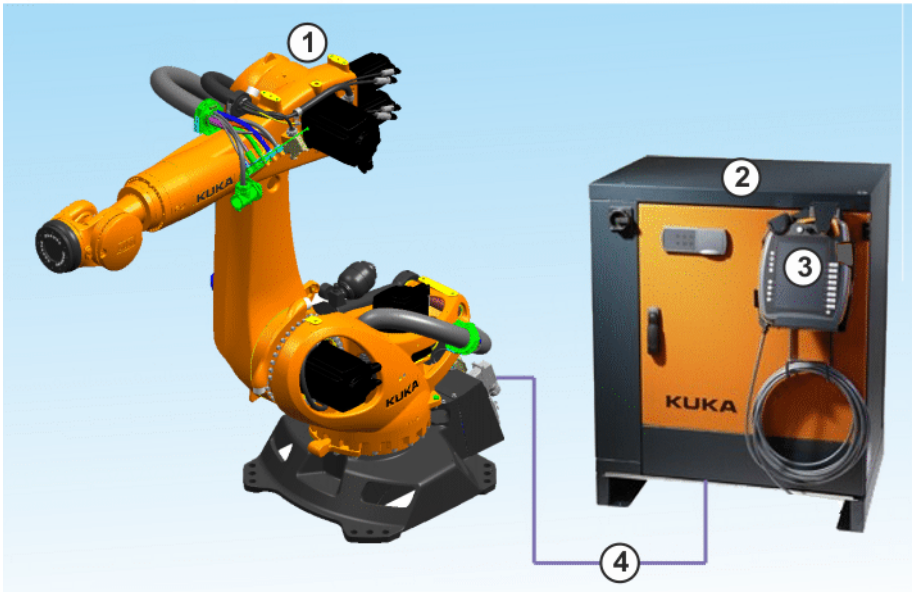


Abb. 2-1: Beispiel eines Industrieroboters

1	Manipulator	3	Programmierhandgerät
2	Robotersteuerung	4	Verbindungsleitungen

2.2 Übersicht KUKA System Software (KSS)

Beschreibung

Die KUKA System Software (KSS) übernimmt alle Grundfunktionen zum Betrieb des Industrieroboters.

- Bahnplanung
- I/O Management
- Daten- und Dateiverwaltung
- Etc.

Es können zusätzliche Technologiepakete, die applikationsspezifische Anweisungen und Konfigurationen enthalten, installiert werden.

smartHMI

Die Bedienoberfläche der KUKA System Software heißt KUKA smartHMI (smart Human-Machine Interface).

Merkmale:

- Benutzerverwaltung
- Programm-Editor
- KRL KUKA Robot Language

- Inline-Formulare zum Programmieren
- Meldungsanzeige
- Konfigurationsfenster
- Etc.

(>>> 4.2 "Bedienoberfläche KUKA smartHMI" Seite 51)

Abhängig von kundenspezifischen Einstellungen kann die Bedienoberfläche vom Standard abweichen.

2.3 Systemvoraussetzungen

Die KSS 8.3 kann auf folgender Robotersteuerung eingesetzt werden:

- KR C4
- mit Windows Embedded Standard 7 V4.x
- und mit 2 GB Arbeitsspeicher

2.4 Bestimmungsgemäße Verwendung der KUKA System Software

Verwendung

Die KUKA System Software ist ausschließlich zum Betreiben eines KUKA Industrieroboters oder einer Kundenkinematik bestimmt.

Jede Version der KUKA System Software darf ausschließlich unter den für sie spezifizierten Systemvoraussetzungen betrieben werden.

Fehlanwendung

Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Für Schäden, die aus einer Fehlanwendung resultieren, haftet die KUKA Roboter GmbH nicht. Das Risiko trägt allein der Betreiber.

Zu den Fehlanwendungen zählen z. B.:

- Betreiben einer Kinematik, die kein KUKA Industrieroboter oder keine Kundenkinematik ist
- Betreiben der KSS unter anderen als den spezifizierten Systemvoraussetzungen

2.5 KUKA-USB-Sticks

Für die Robotersteuerung KR C4 existieren folgende KUKA-USB-Sticks:

KUKA USB-Stick 2.0 NB 4GB

- Datenträger für Software und Archive
- Nicht bootfähig
- Art.-Nr. 00-197-266

KUKA.Recovery USB Stick 2.1 8GB

- Für die Erzeugung und Wiederherstellung von Systemabbildern
- Bootfähig
- Art.-Nr. 00-220-397

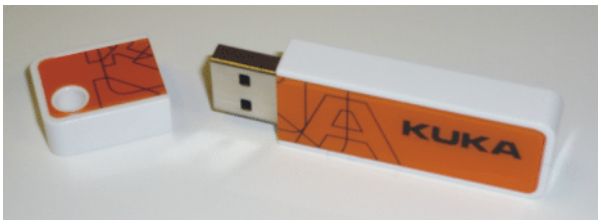


Abb. 2-2: KUKA USB-Stick 2.0 NB 4GB (Art.-Nr. 00-197-266)

Abb. 2-3: KUKA.Recovery USB Stick 2.1 8GB (Art.-Nr. 00-220-397)

3 Sicherheit

3.1 Allgemein

3.1.1 Haftungshinweis

Das im vorliegenden Dokument beschriebene Gerät ist entweder ein Industrieroboter oder eine Komponente davon.

Komponenten des Industrieroboters:

- Manipulator
- Robotersteuerung
- Programmierhandgerät
- Verbindungsleitungen
- Zusatzachsen (optional)
  z. B. Lineareinheit, Drehkipptisch, Positionierer
- Software
- Optionen, Zubehör

Der Industrieroboter ist nach dem Stand der Technik und den anerkannten sicherheitstechnischen Regeln gebaut. Dennoch können bei Fehlanwendung Gefahren für Leib und Leben und Beeinträchtigungen des Industrieroboters und anderer Sachwerte entstehen.

Der Industrieroboter darf nur in technisch einwandfreiem Zustand sowie bestimmungsgemäß, sicherheits- und gefahrenbewusst benutzt werden. Die Benutzung muss unter Beachtung des vorliegenden Dokuments und der dem Industrieroboter bei Lieferung beigefügten Einbauerklärung erfolgen. Störungen, die die Sicherheit beeinträchtigen können, müssen umgehend beseitigt werden.

Sicherheitsinformation

Angaben zur Sicherheit können nicht gegen die KUKA Roboter GmbH ausgelegt werden. Auch wenn alle Sicherheitshinweise befolgt werden, ist nicht gewährleistet, dass der Industrieroboter keine Verletzungen oder Schäden verursacht.

Ohne Genehmigung der KUKA Roboter GmbH dürfen keine Veränderungen am Industrieroboter durchgeführt werden. Es können zusätzliche Komponenten (Werkzeuge, Software etc.), die nicht zum Lieferumfang der KUKA Roboter GmbH gehören, in den Industrieroboter integriert werden. Wenn durch diese Komponenten Schäden am Industrieroboter oder anderen Sachwerten entstehen, haftet dafür der Betreiber.

Ergänzend zum Sicherheitskapitel sind in dieser Dokumentation weitere Sicherheitshinweise enthalten. Diese müssen ebenfalls beachtet werden.

3.1.2 Bestimmungsgemäße Verwendung des Industrieroboters

Der Industrieroboter ist ausschließlich für die in der Betriebsanleitung oder der Montageanleitung im Kapitel "Zweckbestimmung" genannte Verwendung bestimmt.

Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Für Schäden, die aus einer Fehlanwendung resultieren, haftet der Hersteller nicht. Das Risiko trägt allein der Betreiber.

Zur bestimmungsgemäßen Verwendung des Industrieroboters gehört auch die Beachtung der Betriebs- und Montageanleitungen der einzelnen Komponenten und besonders die Befolgung der Wartungsvorschriften.

Fehlanwendung

Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Dazu zählen z. B.:

- Transport von Menschen und Tieren
- Benutzung als Aufstiegshilfen
- Einsatz außerhalb der spezifizierten Betriebsgrenzen
- Einsatz in explosionsgefährdeter Umgebung
- Einsatz ohne zusätzliche Schutzeinrichtungen
- Einsatz im Freien
- Einsatz unter Tage

3.1.3 EG-Konformitätserklärung und Einbauerklärung

Der Industrieroboter ist eine unvollständige Maschine im Sinne der EG-Maschinenrichtlinie. Der Industrieroboter darf nur unter den folgenden Voraussetzungen in Betrieb genommen werden:

- Der Industrieroboter ist in eine Anlage integriert.

  Oder: Der Industrieroboter bildet mit anderen Maschinen eine Anlage.

  Oder: Am Industrieroboter wurden alle Sicherheitsfunktionen und Schutzeinrichtungen ergänzt, die für eine vollständige Maschine im Sinne der EG-Maschinenrichtlinie notwendig sind.
- Die Anlage entspricht der EG-Maschinenrichtlinie. Dies wurde durch ein Konformitäts-Bewertungsverfahren festgestellt.

Konformitätserklärung

Der Systemintegrator muss eine Konformitätserklärung gemäß der Maschinenrichtlinie für die gesamte Anlage erstellen. Die Konformitätserklärung ist Grundlage für die CE-Kennzeichnung der Anlage. Der Industrieroboter darf nur nach landesspezifischen Gesetzen, Vorschriften und Normen betrieben werden.

Die Robotersteuerung besitzt eine CE-Zertifizierung gemäß der EMV-Richtlinie und der Niederspannungsrichtlinie.

Einbauerklärung

Der Industrieroboter als unvollständige Maschine wird mit einer Einbauerklärung nach Anhang II B der Maschinenrichtlinie 2006/42/EG ausgeliefert. Bestandteile der Einbauerklärung sind eine Liste mit den eingehaltenen grundlegenden Anforderungen nach Anhang I und die Montageanleitung.

Mit der Einbauerklärung wird erklärt, dass die Inbetriebnahme der unvollständigen Maschine solange unzulässig bleibt, bis die unvollständige Maschine in eine Maschine eingebaut, oder mit anderen Teilen zu einer Maschine zusammengebaut wurde, diese den Bestimmungen der EG-Maschinenrichtlinie entspricht und die EG-Konformitätserklärung gemäß Anhang II A vorliegt.

3.1.4 Verwendete Begriffe

STOP 0, STOP 1 und STOP 2 sind die Stopp-Definitionen nach EN 60204-1:2006.

Begriff	Beschreibung
Achsbereich	Bereich jeder Achse in Grad oder Millimeter, in dem sie sich bewegen darf. Der Achsbereich muss für jede Achse definiert werden.
Anhalteweg	Anhalteweg = Reaktionsweg + Bremsweg Der Anhalteweg ist Teil des Gefahrenbereichs.
Arbeitsbereich	Bereich, in dem sich der Manipulator bewegen darf. Der Arbeitsbereich ergibt sich aus den einzelnen Achsbereichen.

Begriff	Beschreibung
Betreiber	Der Betreiber eines Industrieroboters kann der Unternehmer, Arbeitgeber oder die delegierte Person sein, die für die Benutzung des Industrieroboters verantwortlich ist.
Gefahrenbereich	Der Gefahrenbereich beinhaltet den Arbeitsbereich und die Anhaltewege des Manipulators und der Zusatzachsen (optional).
Gebrauchsdauer	Die Gebrauchsdauer eines sicherheitsrelevanten Bauteils beginnt ab dem Zeitpunkt der Lieferung des Teils an den Kunden. Die Gebrauchsdauer wird nicht beeinflusst davon, ob das Teil betrieben wird oder nicht, da sicherheitsrelevante Bauteile auch während der Lagerung altern.
KUKA smartPAD	Siehe "smartPAD"
Manipulator	Die Robotermechanik und die zugehörige Elektroinstallation
Schutzbereich	Der Schutzbereich befindet sich außerhalb des Gefahrenbereichs.
Sicherer Betriebshalt	Der sichere Betriebshalt ist eine Stillstandsüberwachung. Er stoppt die Roboterbewegung nicht, sondern überwacht, ob die Roboterachsen still stehen. Wenn diese während des sicheren Betriebshalts bewegt werden, löst dies einen Sicherheitshalt STOP 0 aus. Der sichere Betriebshalt kann auch extern ausgelöst werden. Wenn ein sicherer Betriebshalt ausgelöst wird, setzt die Robotersteuerung einen Ausgang zum Feldbus. Der Ausgang wird auch dann gesetzt, wenn zum Zeitpunkt des Auslösens nicht alle Achsen stillstanden und somit ein Sicherheitshalt STOP 0 ausgelöst wird.
Sicherheitshalt STOP 0	Ein Stopp, der von der Sicherheitssteuerung ausgelöst und durchgeführt wird. Die Sicherheitssteuerung schaltet sofort die Antriebe und die Spannungsversorgung der Bremsen ab. **Hinweis:** Dieser Stopp wird im Dokument als Sicherheitshalt 0 bezeichnet.
Sicherheitshalt STOP 1	Ein Stopp, der von der Sicherheitssteuerung ausgelöst und überwacht wird. Der Bremsvorgang wird vom nicht-sicherheitsgerichteten Teil der Robotersteuerung durchgeführt und von der Sicherheitssteuerung überwacht. Sobald der Manipulator stillsteht, schaltet die Sicherheitssteuerung die Antriebe und die Spannungsversorgung der Bremsen ab. Wenn ein Sicherheitshalt STOP 1 ausgelöst wird, setzt die Robotersteuerung einen Ausgang zum Feldbus. Der Sicherheitshalt STOP 1 kann auch extern ausgelöst werden. **Hinweis:** Dieser Stopp wird im Dokument als Sicherheitshalt 1 bezeichnet.
Sicherheitshalt STOP 2	Ein Stopp, der von der Sicherheitssteuerung ausgelöst und überwacht wird. Der Bremsvorgang wird vom nicht-sicherheitsgerichteten Teil der Robotersteuerung durchgeführt und von der Sicherheitssteuerung überwacht. Die Antriebe bleiben eingeschaltet und die Bremsen geöffnet. Sobald der Manipulator stillsteht, wird ein sicherer Betriebshalt ausgelöst. Wenn ein Sicherheitshalt STOP 2 ausgelöst wird, setzt die Robotersteuerung einen Ausgang zum Feldbus. Der Sicherheitshalt STOP 2 kann auch extern ausgelöst werden. **Hinweis:** Dieser Stopp wird im Dokument als Sicherheitshalt 2 bezeichnet.
Sicherheitsoptionen	Überbegriff für Optionen, die es ermöglichen, zu den Standard-Sicherheitsfunktionen zusätzliche sichere Überwachungen zu konfigurieren. Beispiel: SafeOperation

Begriff	Beschreibung
smartPAD	Programmierhandgerät für die KR C4 Das smartPAD hat alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung und Programmierung des Industrieroboters benötigt werden.
Stopp-Kategorie 0	Die Antriebe werden sofort abgeschaltet und die Bremsen fallen ein. Der Manipulator und die Zusatzachsen (optional) bremsen bahnnah. **Hinweis:** Diese Stopp-Kategorie wird im Dokument als STOP 0 bezeichnet.
Stopp-Kategorie 1	Der Manipulator und die Zusatzachsen (optional) bremsen bahntreu. ■ Betriebsart T1: Die Antriebe werden abgeschaltet, sobald der Roboter steht, spätestens jedoch nach 680 ms. ■ Betriebsarten T2, AUT, AUT EXT: Die Antriebe werden nach 1,5 s abgeschaltet. **Hinweis:** Diese Stopp-Kategorie wird im Dokument als STOP 1 bezeichnet.
Stopp-Kategorie 2	Die Antriebe werden nicht abgeschaltet und die Bremsen fallen nicht ein. Der Manipulator und die Zusatzachsen (optional) bremsen mit einer bahntreuen Bremsrampe. **Hinweis:** Diese Stopp-Kategorie wird im Dokument als STOP 2 bezeichnet.
Systemintegrator (Anlagenintegrator)	Der Systemintegrator ist dafür verantwortlich, den Industrieroboter sicherheitsgerecht in eine Anlage zu integrieren und in Betrieb zu nehmen
T1	Test-Betriebsart Manuell Reduzierte Geschwindigkeit (<= 250 mm/s)
T2	Test-Betriebsart Manuell Hohe Geschwindigkeit (> 250 mm/s zulässig)
Zusatzachse	Bewegungsachse, die nicht zum Manipulator gehört, aber mit der Robotersteuerung angesteuert wird. Z. B. KUKA Lineareinheit, Drehkipptisch, Posiflex

3.2 Personal

Folgende Personen oder Personengruppen werden für den Industrieroboter definiert:

- Betreiber
- Personal

Alle Personen, die am Industrieroboter arbeiten, müssen die Dokumentation mit dem Sicherheitskapitel des Industrieroboters gelesen und verstanden haben.

Betreiber

Der Betreiber muss die arbeitsschutzrechtlichen Vorschriften beachten. Dazu gehört z. B.:

- Der Betreiber muss seinen Überwachungspflichten nachkommen.
- Der Betreiber muss in festgelegten Abständen Unterweisungen durchführen.

Personal

Das Personal muss vor Arbeitsbeginn über Art und Umfang der Arbeiten sowie über mögliche Gefahren belehrt werden. Die Belehrungen sind regelmäßig durchzuführen. Die Belehrungen sind außerdem jedes Mal nach besonderen Vorfällen oder nach technischen Änderungen durchzuführen.

Zum Personal zählen:

- der Systemintegrator

- die Anwender, unterteilt in:
  - Inbetriebnahme-, Wartungs- und Servicepersonal
  - Bediener
  - Reinigungspersonal

Aufstellung, Austausch, Einstellung, Bedienung, Wartung und Instandsetzung dürfen nur nach Vorschrift der Betriebs- oder Montageanleitung der jeweiligen Komponente des Industrieroboters und von hierfür speziell ausgebildetem Personal durchgeführt werden.

Systemintegrator

Der Industrieroboter ist durch den Systemintegrator sicherheitsgerecht in eine Anlage zu integrieren.

Der Systemintegrator ist für folgende Aufgaben verantwortlich:

- Aufstellen des Industrieroboters
- Anschluss des Industrieroboters
- Durchführen der Risikobeurteilung
- Einsatz der notwendigen Sicherheitsfunktionen und Schutzeinrichtungen
- Ausstellen der Konformitätserklärung
- Anbringen des CE-Zeichens
- Erstellung der Betriebsanleitung für die Anlage

Anwender

Der Anwender muss folgende Voraussetzungen erfüllen:

- Der Anwender muss für die auszuführenden Arbeiten geschult sein.
- Tätigkeiten am Industrieroboter darf nur qualifiziertes Personal durchführen. Dies sind Personen, die aufgrund ihrer fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie aufgrund ihrer Kenntnis der einschlägigen Normen die auszuführenden Arbeiten beurteilen und mögliche Gefahren erkennen können.

Arbeiten an der Elektrik und Mechanik des Industrieroboters dürfen nur von Fachkräften vorgenommen werden.

3.3 Arbeits-, Schutz- und Gefahrenbereich

Arbeitsbereiche müssen auf das erforderliche Mindestmaß beschränkt werden. Ein Arbeitsbereich ist mit Schutzeinrichtungen abzusichern.

Die Schutzeinrichtungen (z. B. Schutztüre) müssen sich im Schutzbereich befinden. Bei einem Stopp bremsen Manipulator und Zusatzachsen (optional) und kommen im Gefahrenbereich zu stehen.

Der Gefahrenbereich beinhaltet den Arbeitsbereich und die Anhaltewege des Manipulators und der Zusatzachsen (optional). Sie sind durch trennende Schutzeinrichtungen zu sichern, um eine Gefährdung von Personen oder Sachen auszuschließen.

3.3.1 Ermittlung der Anhaltewege

Die Risikoberurteilung des Systemintegrators kann ergeben, dass für eine Applikation die Anhaltewege ermittelt werden müssen. Für die Ermittlung der Anhaltewege muss der Systemintegrator die sicherheitsrelevanten Stellen auf der programmierten Bahn identifizieren.

Bei der Ermittlung muss der Roboter mit dem Werkzeug und den Lasten verfahren werden, die auch in der Applikation verwendet werden. Der Roboter

muss Betriebstemperatur haben. Dies ist nach ca. 1 h im normalen Betrieb der Fall.

Beim Abfahren der Applikation muss der Roboter an der Stelle, ab der der Anhalteweg ermittelt werden soll, gestoppt werden. Dieser Vorgang ist mehrmals mit Sicherheitshalt 0 und Sicherheitshalt 1 zu wiederholen. Der ungünstigste Anhalteweg ist maßgebend.

Ein Sicherheitshalt 0 kann z. B. durch einen Sicheren Betriebshalt über die Sicherheitsschnittstelle ausgelöst werden. Wenn eine Sicherheitsoption installiert ist, kann er z. B. über eine Raumverletzung ausgelöst werden (z. B. Roboter überschreitet im Automatikbetrieb die Grenze eines aktivierten Arbeitsraums).

Ein Sicherheitshalt 1 kann z. B. durch Drücken des NOT-HALT-Geräts am smartPAD ausgelöst werden.

3.4 Auslöser für Stopp-Reaktionen

Stopp-Reaktionen des Industrieroboters werden aufgrund von Bedienhandlungen oder als Reaktion auf Überwachungen und Fehlermeldungen ausgeführt. Die folgende Tabelle zeigt die Stopp-Reaktionen in Abhängigkeit von der eingestellten Betriebsart.

Auslöser	T1, T2	AUT, AUT EXT
Start-Taste loslassen	STOP 2	-
STOP-Taste drücken	STOP 2	
Antriebe AUS	STOP 1	
Eingang "Fahrfreigabe" fällt weg	STOP 2	
Spannung über Hauptschalter abschalten oder Spannungsausfall	STOP 0	
Interner Fehler im nichtsicherheitsgerichteten Teil der Robotersteuerung	STOP 0 oder STOP 1 (abhängig von der Fehlerursache)	
Betriebsart wechseln während Betrieb	Sicherheitshalt 2	
Schutztür öffnen (Bedienerschutz)	-	Sicherheitshalt 1
Zustimmung lösen	Sicherheitshalt 2	-
Zustimmung durchdrücken oder Fehler	Sicherheitshalt 1	-
NOT-HALT betätigen	Sicherheitshalt 1	
Fehler in Sicherheitssteuerung oder Peripherie der Sicherheitssteuerung	Sicherheitshalt 0	

3.5 Sicherheitsfunktionen

3.5.1 Übersicht der Sicherheitsfunktionen

Folgende Sicherheitsfunktionen sind am Industrieroboter vorhanden:

- Betriebsartenwahl
- Bedienerschutz (= Anschluss für die Verriegelung von trennenden Schutzeinrichtungen)

- NOT-HALT-Einrichtung
- Zustimmeinrichtung
- Externer sicherer Betriebshalt
- Externer Sicherheitshalt 1 (nicht bei der Steuerungsvariante "KR C4 compact")
- Externer Sicherheitshalt 2
- Geschwindigkeitsüberwachung in T1

Die Sicherheitsfunktionen des Industrieroboters erfüllen folgende Anforderungen:

- **Kategorie 3** und **Performance Level d** nach EN ISO 13849-1:2008

Die Anforderungen werden jedoch nur unter folgender Voraussetzung erfüllt:

- Die NOT-HALT-Einrichtung wird mindestens alle 6 Monate betätigt.

An den Sicherheitsfunktionen sind folgende Komponenten beteiligt:

- Sicherheitssteuerung im Steuerungs-PC
- KUKA smartPAD
- Cabinet Control Unit (CCU)
- Resolver Digital Converter (RDC)
- KUKA Power Pack (KPP)
- KUKA Servo Pack (KSP)
- Safety Interface Board (SIB) (falls verwendet)

Zusätzlich gibt es Schnittstellen zu Komponenten außerhalb des Industrieroboters und zu anderen Robotersteuerungen.

⚠ GEFAHR Der Industrieroboter kann ohne funktionsfähige Sicherheitsfunktionen und Schutzeinrichtungen Personen- oder Sachschaden verursachen. Wenn Sicherheitsfunktionen oder Schutzeinrichtungen deaktiviert oder demontiert sind, darf der Industrieroboter nicht betrieben werden.

⚠ Während der Anlagenplanung müssen zusätzlich die Sicherheitsfunktionen der Gesamtanlage geplant und ausgelegt werden. Der Industrieroboter ist in dieses Sicherheitssystem der Gesamtanlage zu integrieren.

3.5.2 Sicherheitssteuerung

Die Sicherheitssteuerung ist eine Einheit innerhalb des Steuerungs-PCs. Sie verknüpft sicherheitsrelevante Signale sowie sicherheitsrelevante Überwachungen.

Aufgaben der Sicherheitssteuerung:

- Antriebe ausschalten, Bremsen einfallen lassen
- Überwachung der Bremsrampe
- Überwachung des Stillstands (nach dem Stopp)
- Geschwindigkeitsüberwachung in T1
- Auswertung sicherheitsrelevanter Signale
- Setzen von sicherheitsgerichteten Ausgängen

3.5.3 Betriebsartenwahl

Der Industrieroboter kann in folgenden Betriebsarten betrieben werden:

- Manuell Reduzierte Geschwindigkeit (T1)
- Manuell Hohe Geschwindigkeit (T2)
- Automatik (AUT)
- Automatik Extern (AUT EXT)

Die Betriebsart nicht wechseln, während ein Programm abgearbeitet wird. Wenn die Betriebsart gewechselt wird, während ein Programm abgearbeitet wird, stoppt der Industrieroboter mit einem Sicherheitshalt 2.

Betriebsart	Verwendung	Geschwindigkeiten
T1	Für Testbetrieb, Programmierung und Teachen	■ Programmverifikation: Programmierte Geschwindigkeit, maximal 250 mm/s ■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s
T2	Für Testbetrieb	■ Programmverifikation: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich
AUT	Für Industrieroboter ohne übergeordnete Steuerung	■ Programmbetrieb: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich
AUT EXT	Für Industrieroboter mit einer übergeordneten Steuerung, z. B. SPS	■ Programmbetrieb: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich

3.5.4 Signal "Bedienerschutz"

Das Signal "Bedienerschutz" dient zur Verriegelung trennender Schutzeinrichtungen, z. B. Schutztüren. Ohne dieses Signal ist kein Automatikbetrieb möglich. Bei einem Signalverlust während des Automatikbetriebs (z. B. Schutztüre wird geöffnet) stoppt der Manipulator mit einem Sicherheitshalt 1.

In den Betriebsarten Manuell Reduzierte Geschwindigkeit (T1) und Manuell Hohe Geschwindigkeit (T2) ist der Bedienerschutz nicht aktiv.

⚠ WARNUNG Nach einem Signalverlust darf es erst dann möglich sein, den Automatikbetrieb fortzusetzen, wenn die Schutzeinrichtung wieder geschlossen wurde und wenn diese Schließung quittiert wurde. Die Quittierung soll verhindern, dass der Automatikbetrieb versehentlich fortgesetzt wird, während sich Personen im Gefahrenbereich befinden, z. B. durch Zufallen der Schutztür.
Die Quittierung muss so gestaltet sein, dass vorher eine tatsächliche Prüfung des Gefahrenbereichs stattfinden kann. Andere Quittierungen (z. B. eine Quittierung, die automatisch auf das Schließen der Schutzeinrichtung folgt) sind unzulässig.
Der Systemintegrator ist dafür verantwortlich, dass diese Anforderungen erfüllt werden. Wenn sie nicht erfüllt werden, können Tod, schwere Verletzungen oder Sachschäden die Folge sein.

3.5.5 NOT-HALT-Einrichtung

Die NOT-HALT-Einrichtung des Industrieroboters ist das NOT-HALT-Gerät am smartPAD. Das Gerät muss bei einer gefahrbringenden Situation oder im Notfall gedrückt werden.

Reaktionen des Industrieroboters, wenn das NOT-HALT-Gerät gedrückt wird:

- Der Manipulator und die Zusatzachsen (optional) stoppen mit einem Sicherheitshalt 1.

Um den Betrieb fortsetzen zu können, muss das NOT-HALT-Gerät durch Drehen entriegelt werden.

⚠ WARNUNG Werkzeuge oder andere Einrichtungen, die mit dem Manipulator verbunden sind, müssen anlagenseitig in den NOT-HALT-Kreis eingebunden werden, wenn von ihnen Gefahren ausgehen können.
Wenn dies nicht beachtet wird, können Tod, schwere Verletzungen oder erheblicher Sachschaden die Folge sein.

Es muss immer mindestens eine externe NOT-HALT-Einrichtung installiert werden. Dies stellt sicher, dass auch bei abgestecktem smartPAD eine NOT-HALT-Einrichtung zur Verfügung steht.

(>>> 3.5.7 "Externe NOT-HALT-Einrichtung" Seite 30)

3.5.6 Abmelden von der übergeordneten Sicherheitssteuerung

Wenn die Robotersteuerung mit einer übergeordneten Sicherheitssteuerung verbunden ist, wird diese Verbindung in folgenden Fällen zwangsläufig unterbrochen:

- Abschalten der Spannung über den Hauptschalter der Robotersteuerung
  Oder Spannungsausfall
- Herunterfahren der Robotersteuerung über die smartHMI
- Aktivierung eines WorkVisual-Projekts von WorkVisual aus oder direkt auf der Robotersteuerung
- Änderungen unter **Inbetriebnahme** > **Netzwerkkonfiguration**
- Änderungen unter **Konfiguration** > **Sicherheitskonfiguration**
- **E/A Treiber** > **Rekonfigurieren**
- Wiederherstellen eines Archivs

Auswirkung der Unterbrechung:

- Wenn eine diskrete Sicherheitsschnittstelle verwendet wird, löst dies einen NOT-HALT für die Gesamtanlage aus.
- Wenn die Ethernet-Sicherheitsschnittstelle verwendet wird, erzeugt die KUKA-Sicherheitssteuerung ein Signal, das bewirkt, dass die übergeordnete Steuerung keinen NOT-HALT für die Gesamtanlage auslöst.

⚠ Wenn die Ethernet-Sicherheitsschnittstelle verwendet wird: Der Systemintegrator muss in seiner Risikobeurteilung berücksichtigen, ob die Tatsache, dass das Ausschalten der Robotersteuerung keinen NOT-HALT der Gesamtanlage auslöst, eine Gefahr darstellen kann und wie der Gefahr entgegenzuwirken ist.
Wenn diese Betrachtung unterlassen wird, können Tod, Verletzungen oder Sachschaden die Folge sein.

⚠ WARNUNG Wenn eine Robotersteuerung ausgeschaltet ist, ist die NOT-HALT-Einrichtung am smartPAD nicht funktionsfähig. Der Betreiber hat dafür Sorge zu tragen, dass das smartPAD entweder abgedeckt oder aus der Anlage entfernt wird. Dies dient dazu, Verwechslungen zwischen wirksamen und nicht wirksamen NOT-HALT-Einrichtungen zu vermeiden.
Wenn diese Maßnahme nicht beachtet wird, können Tod, Verletzungen oder Sachschaden die Folge sein.

3.5.7 Externe NOT-HALT-Einrichtung

Jede Bedienstation, die eine Roboterbewegung oder eine andere gefahrbringende Situation auslösen kann, muss mit einer NOT-HALT-Einrichtung ausgerüstet sein. Hierfür hat der Systemintegrator Sorge zu tragen.

Es muss immer mindestens eine externe NOT-HALT-Einrichtung installiert werden. Dies stellt sicher, dass auch bei abgestecktem smartPAD eine NOT-HALT-Einrichtung zur Verfügung steht.

Externe NOT-HALT-Einrichtungen werden über die Kundenschnittstelle angeschlossen. Externe NOT-HALT-Einrichtungen sind nicht im Lieferumfang des Industrieroboters enthalten.

3.5.8 Zustimmeinrichtung

Die Zustimmeinrichtung des Industrieroboters sind die Zustimmungsschalter am smartPAD.

Am smartPAD sind 3 Zustimmungsschalter angebracht. Die Zustimmungsschalter haben 3 Stellungen:

- Nicht gedrückt
- Mittelstellung
- Durchgedrückt (Panikstellung)

Der Manipulator kann in den Test-Betriebsarten nur bewegt werden, wenn ein Zustimmungsschalter in Mittelstellung gehalten wird.

- Das Loslassen des Zustimmungsschalters löst einen Sicherheitshalt 2 aus.
- Das Durchdrücken des Zustimmungsschalters löst einen Sicherheitshalt 1 aus.
- Es ist möglich, 2 Zustimmungsschalter bis zu 15 Sekunden gleichzeitig in Mittelstellung zu halten. Dies erlaubt das Umgreifen von einem Zustimmungsschalter auf einen anderen. Wenn die Zustimmungsschalter länger als 15 Sekunden gleichzeitig in Mittelstellung gehalten werden, löst dies einen Sicherheitshalt 1 aus.

Bei einer Fehlfunktion eines Zustimmungsschalter (Klemmen) kann der Industrieroboter mit folgenden Methoden gestoppt werden:

- Zustimmungsschalter durchdrücken
- NOT-HALT-Einrichtung betätigen
- Start-Taste loslassen

⚠ WARNUNG Die Zustimmungsschalter dürfen nicht mit Klebebändern oder anderen Hilfsmitteln fixiert oder in einer anderen Weise manipuliert werden.
Tod, Verletzungen oder Sachschaden können die Folge sein.

3.5.9 Externe Zustimmeinrichtung

Externe Zustimmeinrichtungen sind notwendig, wenn sich mehrere Personen im Gefahrenbereich des Industrieroboters aufhalten müssen.

Externe Zustimmeinrichtungen sind nicht im Lieferumfang des Industrieroboters enthalten.

Über welche Schnittstelle externe Zustimmeinrichtungen angeschlossen werden können, ist in der Betriebsanleitung und in der Montageanleitung für die Robotersteuerung in dem Kapitel "Planung" beschrieben.

3.5.10 Externer sicherer Betriebshalt

Der sichere Betriebshalt kann über einen Eingang an der Kundenschnittstelle ausgelöst werden. Der Zustand bleibt erhalten, so lange das externe Signal FALSE ist. Wenn das externe Signal TRUE ist, kann der Manipulator wieder verfahren werden. Es ist keine Quittierung notwendig.

3.5.11 Externer Sicherheitshalt 1 und externer Sicherheitshalt 2

Der Sicherheitshalt 1 und der Sicherheitshalt 2 können über einen Eingang an der Kundenschnittstelle ausgelöst werden. Der Zustand bleibt erhalten, so lange das externe Signal FALSE ist. Wenn das externe Signal TRUE ist, kann der Manipulator wieder verfahren werden. Es ist keine Quittierung notwendig.

Bei der Steuerungsvariante "KR C4 compact" steht kein externer Sicherheitshalt 1 zur Verfügung.

3.5.12 Geschwindigkeitsüberwachung in T1

In der Betriebsart T1 wird die Geschwindigkeit am TCP überwacht. Wenn die Geschwindigkeit 250 mm/s überschreitet, wird ein Sicherheitshalt 0 ausgelöst.

3.6 Zusätzliche Schutzausstattung

3.6.1 Tippbetrieb

Die Robotersteuerung kann in den Betriebsarten Manuell Reduzierte Geschwindigkeit (T1) und Manuell Hohe Geschwindigkeit (T2) ein Programm nur im Tippbetrieb abarbeiten. Das bedeutet: Ein Zustimmungsschalter und die Start-Taste müssen gedrückt gehalten werden, um ein Programm abzuarbeiten.

- Das Loslassen des Zustimmungsschalters löst einen Sicherheitshalt 2 aus.
- Das Durchdrücken des Zustimmungsschalters löst einen Sicherheitshalt 1 aus.
- Das Loslassen der Start-Taste löst einen STOP 2 aus.

3.6.2 Software-Endschalter

Die Achsbereiche aller Manipulator- und Positioniererachsen sind über einstellbare Software-Endschalter begrenzt. Diese Software-Endschalter dienen nur als Maschinenschutz und sind so einzustellen, dass der Manipulator/Positionierer nicht gegen die mechanischen Endanschläge fahren kann.

Die Software-Endschalter werden während der Inbetriebnahme eines Industrieroboters eingestellt.

Weitere Informationen sind in der Bedien- und Programmieranleitung zu finden.

3.6.3 Mechanische Endanschläge

Die Achsbereiche der Grund- und Handachsen des Manipulators sind je nach Robotervariante teilweise durch mechanische Endanschläge begrenzt.

An den Zusatzachsen können weitere mechanische Endanschläge montiert sein.

⚠ WARNUNG Wenn der Manipulator oder eine Zusatzachse gegen ein Hindernis oder einen mechanischen Endanschlag oder die Achsbereichsbegrenzung fährt, kann der Manipulator nicht mehr sicher betrieben werden. Der Manipulator muss außer Betrieb gesetzt werden und vor der Wiederinbetriebnahme ist Rücksprache mit der KUKA Roboter GmbH erforderlich (>>> 15 "KUKA Service" Seite 493).

3.6.4 Mechanische Achsbereichsbegrenzung (Option)

Einige Manipulatoren können in den Achsen A1 bis A3 mit mechanischen Achsbereichsbegrenzungen ausgerüstet werden. Die verstellbaren Achsbereichsbegrenzungen beschränken den Arbeitsbereich auf das erforderliche Minimum. Damit wird der Personen- und Anlagenschutz erhöht.

Bei Manipulatoren, die nicht für die Ausrüstung mit mechanischen Achsbereichsbegrenzungen vorgesehen sind, ist der Arbeitsraum so zu gestalten, dass auch ohne mechanische Arbeitsbereichsbegrenzungen keine Gefährdung von Personen oder Sachen eintreten kann.

Wenn dies nicht möglich ist, muss der Arbeitsbereich durch anlagenseitige Lichtschranken, Lichtvorhänge oder Hindernisse begrenzt werden. An Einlege- und Übergabebereichen dürfen keine Scher- und Quetschstellen entstehen.

Diese Option ist nicht für alle Robotermodelle verfügbar. Informationen zu bestimmten Robotermodellen können bei der KUKA Roboter GmbH erfragt werden.

3.6.5 Achsbereichsüberwachung (Option)

Einige Manipulatoren können in den Grundachsen A1 bis A3 mit 2-kanaligen Achsbereichsüberwachungen ausgerüstet werden. Die Positioniererachsen können mit weiteren Achsbereichsüberwachungen ausgerüstet sein. Mit einer Achsbereichsüberwachung kann für eine Achse der Schutzbereich eingestellt und überwacht werden. Damit wird der Personen- und Anlagenschutz erhöht.

Diese Option ist nicht für alle Robotermodelle verfügbar. Informationen zu bestimmten Robotermodellen können bei der KUKA Roboter GmbH erfragt werden.

3.6.6 Möglichkeiten zum Bewegen des Manipulators ohne Antriebsenergie

Der Betreiber der Anlage muss dafür Sorge tragen, dass die Ausbildung des Personals hinsichtlich des Verhaltens in Notfällen oder außergewöhnlichen Situationen auch umfasst, wie der Manipulator ohne Antriebsenergie bewegt werden kann.

Beschreibung

Um den Manipulator nach einem Unfall oder Störfall ohne Antriebsenergie zu bewegen, stehen folgende Möglichkeiten zur Verfügung:

- Freidreh-Vorrichtung (Option)

  Die Freidreh-Vorrichtung kann für die Grundachs-Antriebsmotoren und je nach Robotervariante auch für die Handachs-Antriebsmotoren verwendet werden.
- Bremsenöffnungs-Gerät (Option)

  Das Bremsenöffnungs-Gerät ist für Robotervarianten bestimmt, deren Motoren nicht frei zugänglich sind.
- Handachsen direkt mit der Hand bewegen

  Bei Varianten der niedrigen Traglastklasse steht für die Handachsen keine Freidreh-Vorrichtung zur Verfügung. Diese ist nicht notwendig, da die Handachsen direkt mit der Hand bewegt werden können.

Informationen dazu, welche Möglichkeiten für welche Robotermodelle verfügbar sind und wie sie anzuwenden sind, sind in der Montage- oder Betriebsanleitung für den Roboter zu finden oder können bei der KUKA Roboter GmbH erfragt werden.

HINWEIS Wenn der Manipulator ohne Antriebsenergie bewegt wird, kann dies die Motorbremsen der betroffenen Achsen beschädigen. Wenn die Bremse beschädigt wurde, muss der Motor getauscht werden. Der Manipulator darf deshalb nur in Notfällen ohne Antriebsenergie bewegt werden, z. B. zur Befreiung von Personen.

3.6.7 Kennzeichnungen am Industrieroboter

Alle Schilder, Hinweise, Symbole und Markierungen sind sicherheitsrelevante Teile des Industrieroboters. Sie dürfen nicht verändert oder entfernt werden.

Kennzeichnungen am Industrieroboter sind:

- Leistungsschilder
- Warnhinweise
- Sicherheitssymbole
- Bezeichnungsschilder
- Leitungsmarkierungen
- Typenschilder

Weitere Informationen sind in den Technischen Daten der Betriebsanleitungen oder Montageanleitungen der Komponenten des Industrieroboters zu finden.

3.6.8 Externe Schutzeinrichtungen

Der Zutritt von Personen in den Gefahrenbereich des Industrieroboters ist durch Schutzeinrichtungen zu verhindern. Der Systemintegrator hat hierfür Sorge zu tragen.

Trennende Schutzeinrichtungen müssen folgende Anforderungen erfüllen:

- Sie entsprechen den Anforderungen von EN 953.
- Sie verhindern den Zutritt von Personen in den Gefahrenbereich und können nicht auf einfache Weise überwunden werden.
- Sie sind ausreichend befestigt und halten den vorhersehbaren Betriebs- und Umgebungskräften stand.
- Sie stellen nicht selbst eine Gefährdung dar und können keine Gefährdungen verursachen.
- Der vorgeschriebene Mindestabstand zum Gefahrenbereich wird eingehalten.

Schutztüren (Wartungstüren) müssen folgende Anforderungen erfüllen:

- Die Anzahl ist auf das notwendige Minimum beschränkt.
- Die Verriegelungen (z. B. Schutztürschalter) sind über Schutztür-Schaltgeräte oder Sicherheits-SPS mit dem Bedienerschutz-Eingang der Robotersteuerung verbunden.
- Schaltgeräte, Schalter und Art der Schaltung entsprechen den Anforderungen von Performance Level d und Kategorie 3 nach EN ISO 13849-1.
- Je nach Gefährdungslage: Die Schutztür ist zusätzlich mit einer Zuhaltung gesichert, die das Öffnen der Schutztür erst erlaubt, wenn der Manipulator sicher stillsteht.
- Der Taster zum Quittieren der Schutztür ist außerhalb des durch Schutzeinrichtungen abgegrenzten Raums angebracht.

Weitere Informationen sind in den entsprechenden Normen und Vorschriften zu finden. Hierzu zählt auch EN 953.

Andere Schutzeinrichtungen

Andere Schutzeinrichtungen müssen nach den entsprechenden Normen und Vorschriften in die Anlage integriert werden.

3.7 Übersicht Betriebsarten und Schutzfunktionen

Die folgende Tabelle zeigt, bei welcher Betriebsart die Schutzfunktionen aktiv sind.

Schutzfunktionen	T1	T2	AUT	AUT EXT
Bedienerschutz	-	-	aktiv	aktiv
NOT-HALT-Einrichtung	aktiv	aktiv	aktiv	aktiv
Zustimmeinrichtung	aktiv	aktiv	-	-
Reduzierte Geschwindigkeit bei Programmverifikation	aktiv	-	-	-
Tippbetrieb	aktiv	aktiv	-	-
Software-Endschalter	aktiv	aktiv	aktiv	aktiv

3.8 Sicherheitsmaßnahmen

3.8.1 Allgemeine Sicherheitsmaßnahmen

Der Industrieroboter darf nur in technisch einwandfreiem Zustand sowie bestimmungsgemäß und sicherheitsbewußt benutzt werden. Bei Fehlhandlungen können Personen- und Sachschäden entstehen.

Auch bei ausgeschalteter und gesicherter Robotersteuerung ist mit möglichen Bewegungen des Industrieroboters zu rechnen. Durch falsche Montage (z. B. Überlast) oder mechanische Defekte (z. B. Bremsdefekt) können Manipulator oder Zusatzachsen absacken. Wenn am ausgeschalteten Industrieroboter gearbeitet wird, sind Manipulator und Zusatzachsen vorher so in Stellung zu bringen, dass sie sich mit und ohne Traglast nicht selbständig bewegen können. Wenn das nicht möglich ist, müssen Manipulator und Zusatzachsen entsprechend abgesichert werden.

⚠ GEFAHR Der Industrieroboter kann ohne funktionsfähige Sicherheitsfunktionen und Schutzeinrichtungen Personen- oder Sachschaden verursachen. Wenn Sicherheitsfunktionen oder Schutzeinrichtungen deaktiviert oder demontiert sind, darf der Industrieroboter nicht betrieben werden.

⚠ GEFAHR Der Aufenthalt unter der Robotermechanik kann zum Tod oder zu Verletzungen führen. Aus diesem Grund ist der Aufenthalt unter der Robotermechanik verboten!

⚠ VORSICHT Die Motoren erreichen während des Betriebs Temperaturen, die zu Hautverbrennungen führen können. Berührungen sind zu vermeiden. Es sind geeignete Schutzmaßnahmen zu ergreifen, z. B. Schutzhandschuhe tragen.

smartPAD

Der Betreiber hat sicherzustellen, dass der Industrieroboter nur von autorisierten Personen mit dem smartPAD bedient wird.

Wenn mehrere smartPADs an einer Anlage verwendet werden, muss darauf geachtet werden, dass jedes smartPAD dem zugehörigen Industrieroboter eindeutig zugeordnet ist. Es darf keine Verwechslung stattfinden.

⚠ WARNUNG Der Betreiber hat dafür Sorge zu tragen, dass abgekoppelte smartPADs sofort aus der Anlage entfernt werden und außer Sicht- und Reichweite des am Industrieroboter arbeitenden Personals verwahrt werden. Dies dient dazu, Verwechslungen zwischen wirksamen und nicht wirksamen NOT-HALT-Einrichtungen zu vermeiden.
Wenn dies nicht beachtet wird, können Tod, schwere Verletzungen oder erheblicher Sachschaden die Folge sein.

Änderungen

Nach Änderungen am Industrieroboter muss geprüft werden, ob das erforderliche Sicherheitsniveau gewährleistet ist. Für diese Prüfung sind die geltenden staatlichen oder regionalen Arbeitsschutzvorschriften zu beachten. Zusätzlich sind alle Sicherheitsfunktionen auf ihre sichere Funktion zu testen.

Neue oder geänderte Programme müssen immer zuerst in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden.

Nach Änderungen am Industrieroboter müssen bestehende Programme immer zuerst in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden. Dies gilt für sämtliche Komponenten des Industrieroboters und schließt damit auch Änderungen an Software und Konfigurationseinstellungen ein.

Störungen

Bei Störungen am Industrieroboter ist wie folgt vorzugehen:

- Robotersteuerung ausschalten und gegen unbefugtes Wiedereinschalten (z. B. mit einem Vorhängeschloss) sichern.
- Störung durch ein Schild mit entsprechendem Hinweis kennzeichnen.
- Aufzeichnungen über Störungen führen.
- Störung beheben und Funktionsprüfung durchführen.

3.8.2 Transport

Manipulator

Die vorgeschriebene Transportstellung für den Manipulator muss beachtet werden. Der Transport muss gemäß der Betriebsanleitung oder Montageanleitung für den Manipulator erfolgen.

Erschütterungen oder Stöße während des Transports vermeiden, damit keine Schäden an der Robotermechanik entstehen.

Robotersteuerung

Die vorgeschriebene Transportstellung für die Robotersteuerung muss beachtet werden. Der Transport muss gemäß der Betriebsanleitung oder Montageanleitung für die Robotersteuerung erfolgen.

Erschütterungen oder Stöße während des Transports vermeiden, damit keine Schäden in der Robotersteuerung entstehen.

Zusatzachse (optional)

Die vorgeschriebene Transportstellung für die Zusatzachse (z. B. KUKA Lineareinheit, Drehkipptisch, Positionierer) muss beachtet werden. Der Transport muss gemäß der Betriebsanleitung oder Montageanleitung für die Zusatzachse erfolgen.

3.8.3 Inbetriebnahme und Wiederinbetriebnahme

Vor der ersten Inbetriebnahme von Anlagen und Geräten muss eine Prüfung durchgeführt werden, die sicherstellt, dass Anlagen und Geräte vollständig und funktionsfähig sind, dass diese sicher betrieben werden können und dass Schäden erkannt werden.

Für diese Prüfung sind die geltenden staatlichen oder regionalen Arbeitsschutzvorschriften zu beachten. Zusätzlich sind alle Sicherheitsfunktionen auf ihre sichere Funktion zu testen.

Vor der Inbetriebnahme müssen in der KUKA System Software die Passwörter für die Benutzergruppen geändert werden. Die Passwörter dürfen nur autorisiertem Personal mitgeteilt werden.

Die Robotersteuerung ist für den jeweiligen Industrieroboter vorkonfiguriert. Der Manipulator und die Zusatzachsen (optional) können bei vertauschten Kabeln falsche Daten erhalten und dadurch Personen- oder Sachschaden verursachen. Wenn eine Anlage aus mehreren Manipulatoren besteht, die Verbindungsleitungen immer an Manipulator und zugehöriger Robotersteuerung anschließen.

Wenn zusätzliche Komponenten (z. B. Leitungen), die nicht zum Lieferumfang der KUKA Roboter GmbH gehören, in den Industrieroboter integriert werden, ist der Betreiber dafür verantwortlich, dass diese Komponenten keine Sicherheitsfunktionen beeinträchtigen oder außer Funktion setzen.

HINWEIS Wenn die Schrankinnentemperatur der Robotersteuerung stark von der Umgebungstemperatur abweicht, kann sich Kondenswasser bilden, das zu Schäden an der Elektrik führt. Robotersteuerung erst in Betrieb nehmen, wenn sich die Schrankinnentemperatur der Umgebungstemperatur angepasst hat.

Funktionsprüfung

Vor der Inbetriebnahme und Wiederinbetriebnahme sind folgende Prüfungen durchzuführen:

Prüfung allgemein:

Sicherzustellen ist:

- Der Industrieroboter ist gemäß den Angaben in der Dokumentation korrekt aufgestellt und befestigt.
- Es sind keine Fremdkörper oder defekte, lockere oder lose Teile am Industrieroboter.
- Alle erforderlichen Schutzeinrichtungen sind korrekt installiert und funktionsfähig.
- Die Anschlusswerte des Industrieroboters stimmen mit der örtlichen Netzspannung und Netzform überein.
- Der Schutzleiter und die Potenzialausgleichs-Leitung sind ausreichend ausgelegt und korrekt angeschlossen.
- Die Verbindungskabel sind korrekt angeschlossen und die Stecker verriegelt.

Prüfung der Sicherheitsfunktionen:

Bei folgenden Sicherheitsfunktionen muss durch einen Funktionstest sichergestellt werden, dass sie korrekt arbeiten:

- Lokale NOT-HALT-Einrichtung
- Externe NOT-HALT-Einrichtung (Ein- und Ausgang)
- Zustimmeinrichtung (in den Test-Betriebsarten)
- Bedienerschutz
- Alle weiteren verwendeten sicherheitsrelevanten Ein- und Ausgänge
- Weitere externe Sicherheitsfunktionen

3.8.3.1 Prüfung Maschinendaten und Sicherheitskonfiguration

⚠ WARNUNG Wenn die falschen Maschinendaten oder eine falsche Steuerungskonfiguration geladen sind, darf der Industrieroboter nicht verfahren werden! Tod, schwere Verletzungen oder erhebliche Sachschäden können sonst die Folge sein. Die richtigen Daten müssen geladen werden.

- Es ist sicherzustellen, dass das Typenschild an der Robotersteuerung die gleichen Maschinendaten besitzt, die in der Einbauerklärung eingetragen sind. Die Maschinendaten auf dem Typenschild des Manipulators und der Zusatzachsen (optional) müssen bei der Inbetriebnahme eingetragen werden.
- Im Rahmen der Inbetriebnahme müssen die Praxistests für die Maschinendaten durchgeführt werden.
- Nach Änderungen an den Maschinendaten muss die Sicherheitskonfiguration geprüft werden.
- Nach der Aktivierung eines WorkVisual-Projekts auf der Robotersteuerung muss die Sicherheitskonfiguration geprüft werden.

- Wenn bei der Prüfung der Sicherheitskonfiguration Maschinendaten übernommen wurden (gleichgültig, aus welchem Grund die Sicherheitskonfiguration geprüft wurde), müssen die Praxistests für die Maschinendaten durchgeführt werden.
- Ab System Software 8.3: Wenn sich die Prüfsumme der Sicherheitskonfiguration geändert hat, müssen die sicheren Achsüberwachungen geprüft werden.

> Informationen zum Prüfen der Sicherheitskonfiguration und der sicheren Achsüberwachungen sind in der Bedien- und Programmieranleitung für Systemintegratoren zu finden.

Wenn die Praxistests bei einer Erstinbetriebnahme nicht erfolgreich bestanden werden, muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

Wenn die Praxistests bei einer anderen Durchführung nicht erfolgreich bestanden werden, müssen die Maschinendaten und die sicherheitsrelevante Steuerungskonfiguration kontrolliert und korrigiert werden.

Praxistest allgemein

Wenn Praxistests für die Maschinendaten erforderlich sind, muss dieser Test immer durchgeführt werden.

Es gibt folgende Möglichkeiten, den allgemeinen Praxistest durchzuführen:

- TCP-Vermessung mit der XYZ 4-Punkt-Methode

  Der Praxistest ist bestanden, wenn der TCP erfolgreich vermessen werden konnte.

Oder:

1. Den TCP auf einen selbst gewählten Punkt ausrichten.

   Der Punkt dient als Referenzpunkt. Er muss so liegen, dass umorientiert werden kann.
2. Den TCP je 1-mal mindestens 45° in A-, B- und C-Richtung manuell verfahren.

   Die Bewegungen brauchen sich nicht addieren, d. h. wenn in eine Richtung verfahren wurde, kann man wieder zurückfahren, bevor man in die nächste Richtung verfährt.

   Der Praxistest ist bestanden, wenn der TCP insgesamt nicht weiter als 2 cm vom Referenzpunkt abweicht.

Praxistest für nicht math. gekoppelte Achsen

Wenn Praxistests für die Maschinendaten erforderlich sind, muss dieser Test durchgeführt werden, wenn Achsen vorhanden sind, die nicht mathematisch gekoppelt sind.

1. Die Ausgangsposition der mathematisch nicht gekoppelten Achse markieren.
2. Die Achse manuell eine selbst gewählte Weglänge verfahren. Die Weglänge auf der smartHMI über die Anzeige **Istposition** ermitteln.
   - Lineare Achsen eine bestimmte Strecke verfahren.
   - Rotatorische Achsen einen bestimmten Winkel verfahren.
3. Den zurückgelegten Weg messen und mit dem laut smartHMI gefahrenen Weg vergleichen.

   Der Praxistest ist bestanden, wenn die Werte maximal um 10 % voneinander abweichen.
4. Den Test für jede mathematisch nicht gekoppelte Achse wiederholen.

Praxistest für koppelbare Achsen

Wenn Praxistests für die Maschinendaten erforderlich sind, muss dieser Test durchgeführt werden, wenn physikalisch an-/abkoppelbare Achsen vorhanden sind, z. B. eine Servozange.

1. Die koppelbare Achse physikalisch abkoppeln.

2. Alle verbleibenden Achsen einzeln verfahren.
   Der Praxistest ist bestanden, wenn alle verbleibenden Achsen verfahren werden konnten.

3.8.3.2 Inbetriebnahme-Modus

Beschreibung

Der Industrieroboter kann über die Bedienoberfläche smartHMI in einen Inbetriebnahme-Modus gesetzt werden. In diesem Modus ist es möglich, den Manipulator in T1 zu verfahren, ohne dass die externen Schutzeinrichtungen in Betrieb sind.

Wann der Inbetriebnahme-Modus möglich ist, ist abhängig davon, welche Sicherheitsschnittstelle verwendet wird.

Diskrete Sicherheitsschnittstelle

- System Software 8.2 und kleiner:
  Der Inbetriebnahme-Modus ist immer dann möglich, wenn sämtliche Eingangssignale an der diskreten Sicherheitsschnittstelle den Zustand "logisch Null" haben. Wenn dies nicht der Fall ist, dann verhindert oder beendet die Robotersteuerung den Inbetriebnahme-Modus.
  Wenn zusätzlich eine diskrete Sicherheitsschnittstelle für Sicherheitsoptionen verwendet wird, müssen auch dort die Eingänge "logisch Null" sein.
- System Software 8.3 und höher:
  Der Inbetriebnahme-Modus ist immer möglich. Das bedeutet auch, dass er vom Zustand der Eingänge an der diskreten Sicherheitsschnittstelle unabhängig ist.
  Wenn zusätzlich eine diskrete Sicherheitsschnittstelle für Sicherheitsoptionen verwendet wird: Auch die Zustände dieser Eingänge spielen keine Rolle.

Ethernet-Sicherheitsschnittstelle

Die Robotersteuerung verhindert oder beendet den Inbetriebnahme-Modus, wenn eine Verbindung zu einem übergeordneten Sicherheitssystem besteht oder aufgebaut wird.

Auswirkung

Wenn der Inbetriebnahme-Modus aktiviert wird, gehen alle Ausgänge automatisch in den Zustand "logisch Null".

Wenn die Robotersteuerung ein Peripherieschütz (US2) besitzt und wenn in der Sicherheitskonfiguration festgelegt ist, dass dieses in Abhängigkeit von der Fahrfreigabe schaltet, dann gilt dies auch im Inbetriebnahme-Modus.
D. h., wenn die Fahrfreigabe vorhanden ist, ist – auch im Inbetriebnahme-Modus – die US2-Spannung eingeschaltet.

Gefahren

Mögliche Gefahren und Risiken bei Verwendung des Inbetriebnahme-Modus:

- Person läuft in den Gefahrenbereich des Manipulators.
- Im Gefahrenfall wird eine nicht aktive externe NOT-HALT-Einrichtung betätigt und der Manipulator wird nicht abgeschaltet.

Zusätzliche Maßnahmen zur Risikovermeidung bei Inbetriebnahme-Modus:

- Nicht funktionsfähige NOT-HALT-Einrichtungen abdecken oder mit entsprechendem Warnschild auf die nicht funktionierende NOT-HALT-Einrichtung hinweisen.
- Wenn kein Schutzzaun vorhanden ist, muss mit anderen Maßnahmen verhindert werden, dass Personen in den Gefahrenbereich des Manipulators gelangen, z. B. mit einem Sperrband.

Verwendung

Bestimmungsgemäße Verwendung des Inbetriebnahme-Modus:

- Zur Inbetriebnahme im T1-Betrieb, wenn die externen Schutzeinrichtungen noch nicht installiert oder in Betrieb genommen sind. Der Gefahrenbereich muss dabei mindestens mit einem Sperrband abgegrenzt werden.
- Zur Fehlereingrenzung (Peripheriefehler).
- Die Nutzung des Inbetriebnahme-Modus muss so gering wie möglich gehalten werden.

⚠ WARNUNG Bei Verwendung des Inbetriebnahme-Modus sind alle externen Schutzeinrichtungen außer Betrieb. Das Servicepersonal hat dafür zu sorgen, dass sich keine Personen im und in der Nähe des Gefahrenbereichs des Manipulators aufhalten, während die Schutzeinrichtungen außer Betrieb sind.
Wenn dies nicht beachtet wird, können Tod, Verletzungen oder Sachschäden die Folge sein.

Fehlanwendung

Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Für Schäden, die aus einer Fehlanwendung resultieren, haftet die KUKA Roboter GmbH nicht. Das Risiko trägt allein der Betreiber.

3.8.4 Manueller Betrieb

Der manuelle Betrieb ist der Betrieb für Einrichtarbeiten. Einrichtarbeiten sind alle Arbeiten, die am Industrieroboter durchgeführt werden müssen, um den Automatikbetrieb aufnehmen zu können. Zu den Einrichtarbeiten gehören:

- Tippbetrieb
- Teachen
- Programmieren
- Programmverifikation

Beim manuellen Betrieb ist Folgendes zu beachten:

- Neue oder geänderte Programme müssen immer zuerst in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden.
- Werkzeuge, Manipulator oder Zusatzachsen (optional) dürfen niemals den Absperrzaun berühren oder über den Absperrzaun hinausragen.
- Werkstücke, Werkzeuge und andere Gegenstände dürfen durch das Verfahren des Industrieroboters weder eingeklemmt werden, noch zu Kurzschlüssen führen oder herabfallen.
- Alle Einrichtarbeiten müssen so weit wie möglich von außerhalb des durch Schutzeinrichtungen abgegrenzten Raumes durchgeführt werden.

Wenn die Einrichtarbeiten von innerhalb des durch Schutzeinrichtungen abgegrenzten Raumes durchgeführt werden müssen, muss Folgendes beachtet werden.

In der Betriebsart **Manuell Reduzierte Geschwindigkeit (T1)**:

- Wenn vermeidbar, dürfen sich keine weiteren Personen im durch Schutzeinrichtungen abgegrenzten Raum aufhalten.

  Wenn es notwendig ist, dass sich mehrere Personen im durch Schutzeinrichtungen abgegrenzten Raum aufhalten, muss Folgendes beachtet werden:
  - Jede Person muss eine Zustimmeinrichtung zur Verfügung haben.
  - Alle Personen müssen ungehinderte Sicht auf den Industrieroboter haben.

  - Zwischen allen Personen muss immer Möglichkeit zum Blickkontakt bestehen.
- Der Bediener muss eine Position einnehmen, aus der er den Gefahrenbereich einsehen kann und einer Gefahr ausweichen kann.

In der Betriebsart **Manuell Hohe Geschwindigkeit (T2)**:

- Diese Betriebsart darf nur verwendet werden, wenn die Anwendung einen Test mit höherer als mit der Manuell Reduzierten Geschwindigkeit erfordert.
- Teachen und Programmieren sind in dieser Betriebsart nicht erlaubt.
- Der Bediener muss vor Beginn des Tests sicherstellen, dass die Zustimmeinrichtungen funktionsfähig sind.
- Der Bediener muss eine Position außerhalb des Gefahrenbereichs einnehmen.
- Es dürfen sich keine weiteren Personen im durch Schutzeinrichtungen abgegrenzten Raum aufhalten. Der Bediener muss hierfür Sorge tragen.

3.8.5 Simulation

Simulationsprogramme entsprechen nicht exakt der Realität. Roboterprogramme, die in Simulationsprogrammen erstellt wurden, sind an der Anlage in der Betriebsart **Manuell Reduzierte Geschwindigkeit (T1)** zu testen. Gegebenenfalls muss das Programm überarbeitet werden.

3.8.6 Automatikbetrieb

Der Automatikbetrieb ist nur zulässig, wenn folgende Sicherheitsmaßnahmen eingehalten werden:

- Alle Sicherheits- und Schutzeinrichtungen sind vorhanden und funktionsfähig.
- Es befinden sich keine Personen in der Anlage.
- Die festgelegten Arbeitsverfahren werden befolgt.

Wenn der Manipulator oder eine Zusatzachse (optional) ohne ersichtlichen Grund stehen bleibt, darf der Gefahrenbereich erst betreten werden, wenn ein NOT-HALT ausgelöst wurde.

3.8.7 Wartung und Instandsetzung

Nach Wartungs- und Instandsetzungsarbeiten muss geprüft werden, ob das erforderliche Sicherheitsniveau gewährleistet ist. Für diese Prüfung sind die geltenden staatlichen oder regionalen Arbeitsschutzvorschriften zu beachten. Zusätzlich sind alle Sicherheitsfunktionen auf ihre sichere Funktion zu testen.

Die Wartung und Instandsetzung soll sicherstellen, dass der funktionsfähige Zustand erhalten bleibt oder bei Ausfall wieder hergestellt wird. Die Instandsetzung umfasst die Störungssuche und die Reparatur.

Sicherheitsmaßnahmen bei Tätigkeiten am Industrieroboter sind:

- Tätigkeiten außerhalb des Gefahrenbereichs durchführen. Wenn Tätigkeiten innerhalb des Gefahrenbereichs durchzuführen sind, muss der Betreiber zusätzliche Schutzmaßnahmen festlegen, um einen sicheren Personenschutz zu gewährleisten.
- Industrieroboter ausschalten und gegen Wiedereinschalten (z. B. mit einem Vorhängeschloss) sichern. Wenn die Tätigkeiten bei eingeschalteter Robotersteuerung durchzuführen sind, muss der Betreiber zusätzliche

Schutzmaßnahmen festlegen, um einen sicheren Personenschutz zu gewährleisten.

- Wenn die Tätigkeiten bei eingeschalteter Robotersteuerung durchzuführen sind, dürfen diese nur in der Betriebsart T1 durchgeführt werden.
- Tätigkeiten mit einem Schild an der Anlage kennzeichnen. Dieses Schild muss auch bei zeitweiser Unterbrechung der Tätigkeiten vorhanden sein.
- Die NOT-HALT-Einrichtungen müssen aktiv bleiben. Wenn Sicherheitsfunktionen oder Schutzeinrichtungen aufgrund Wartungs- oder Instandsetzungsarbeiten deaktiviert werden, muss die Schutzwirkung anschließend sofort wiederhergestellt werden.

⚠ GEFAHR Vor Arbeiten an spannungsführenden Teilen des Robotersystems muss der Hauptschalter ausgeschaltet und gegen Wiedereinschalten gesichert werden. Anschließend muss die Spannungsfreiheit festgestellt werden.
Es genügt nicht, vor Arbeiten an spannungsführenden Teilen einen NOT-HALT oder einen Sicherheitshalt auszulösen oder die Antriebe auszuschalten, weil dabei das Robotersystem nicht vom Netz getrennt wird. Es stehen weiterhin Teile unter Spannung. Tod oder schwere Verletzungen können die Folge sein.

Fehlerhafte Komponenten müssen durch neue Komponenten, mit derselben Artikelnummer oder durch Komponenten, die von der KUKA Roboter GmbH als gleichwertig ausgewiesen sind, ersetzt werden.

Reinigungs- und Pflegearbeiten sind gemäß der Betriebsanleitung durchzuführen.

Robotersteuerung

Auch wenn die Robotersteuerung ausgeschaltet ist, können Teile unter Spannungen stehen, die mit Peripheriegeräten verbunden sind. Die externen Quellen müssen deshalb ausgeschaltet werden, wenn an der Robotersteuerung gearbeitet wird.

Bei Tätigkeiten an Komponenten in der Robotersteuerung müssen die EGB-Vorschriften eingehalten werden.

Nach Ausschalten der Robotersteuerung kann an verschiedenen Komponenten mehrere Minuten eine Spannung von über 50 V (bis zu 780 V) anliegen. Um lebensgefährliche Verletzungen zu verhindern, dürfen in diesem Zeitraum keine Tätigkeiten am Industrieroboter durchgeführt werden.

Das Eindringen von Wasser und Staub in die Robotersteuerung muss verhindert werden.

Gewichtsausgleich

Einige Robotervarianten sind mit einem hydropneumatischen, Feder- oder Gaszylinder-Gewichtsausgleich ausgestattet.

Die hydropneumatischen und Gaszylinder-Gewichtsausgleiche sind Druckgeräte. Sie gehören zu den überwachungspflichtigen Anlagen und unterliegen der Druckgeräterichtlinie.

Der Betreiber muss die landesspezifischen Gesetze, Vorschriften und Normen für Druckgeräte beachten.

Prüffristen in Deutschland nach Betriebssicherheitsverordnung §14 und §15. Prüfung vor Inbetriebnahme am Aufstellort durch den Betreiber.

Sicherheitsmaßnahmen bei Tätigkeiten an Gewichtsausgleichsystemen sind:

- Die von den Gewichtsausgleichsystemen unterstützten Baugruppen des Manipulators müssen gesichert werden.
- Tätigkeiten an den Gewichtsausgleichsystemen darf nur qualifiziertes Personal durchführen.

Gefahrstoffe

Sicherheitsmaßnahmen beim Umgang mit Gefahrstoffen sind:

- Längeren und wiederholten intensiven Hautkontakt vermeiden.
- Einatmen von Ölnebeln und -dämpfen vermeiden.
- Für Hautreinigung und Hautpflege sorgen.

Für den sicheren Einsatz unserer Produkte empfehlen wir unseren Kunden regelmäßig die aktuellen Sicherheitsdatenblätter von den Herstellern der Gefahrstoffe anzufordern.

3.8.8 Außerbetriebnahme, Lagerung und Entsorgung

Die Außerbetriebnahme, Lagerung und Entsorgung des Industrieroboter darf nur nach landesspezifischen Gesetzen, Vorschriften und Normen erfolgen.

3.8.9 Sicherheitsmaßnahmen für Single Point of Control

Übersicht

Wenn am Industrieroboter bestimmte Komponenten zum Einsatz kommen, müssen Sicherheitsmaßnahmen durchgeführt werden, um das Prinzip des "Single Point of Control" (SPOC) vollständig umzusetzen.

Die relevanten Komponenten sind:

- Submit-Interpreter
- SPS
- OPC-Server
- Remote Control Tools
- Tools zur Konfiguration von Bussystemen mit Online-Funktionalität
- KUKA.RobotSensorInterface

Die Ausführung weiterer Sicherheitsmaßnahmen kann notwendig sein. Dies muss je nach Anwendungsfall geklärt werden und obliegt dem Systemintegrator, Programmierer oder Betreiber der Anlage.

Da die sicheren Zustände von Aktoren in der Peripherie der Robotersteuerung nur dem Systemintegrator bekannt sind, obliegt es ihm diese Aktoren, z. B. bei NOT-HALT, in einen sicheren Zustand zu versetzen.

T1, T2

In den Betriebsarten T1 und T2 dürfen die oben genannten Komponenten nur auf den Industrieroboter zugreifen, wenn folgende Signale folgende Zustände haben:

Signal	Zustand erforderlich für SPOC
$USER_SAF	TRUE
$SPOC_MOTION_ENABLE	TRUE

Submit-Interpreter, SPS

Wenn mit dem Submit-Interpreter oder der SPS über das E/A-System Bewegungen (z. B. Antriebe oder Greifer) angesteuert werden und diese nicht anderweitig abgesichert sind, so wirkt diese Ansteuerung auch in den Betriebsarten T1 und T2 oder während eines anstehenden NOT-HALT.

Wenn mit dem Submit-Interpreter oder der SPS Variablen verändert werden, die sich auf die Roboterbewegung auswirken (z. B. Override), so wirkt dies auch in den Betriebsarten T1 und T2 oder während eines anstehenden NOT-HALT.

Sicherheitsmaßnahmen:

- In T1 und T2 darf die Systemvariable $OV_PRO vom Submit-Interpreter aus oder von der SPS nicht beschrieben werden.

- Sicherheitsrelevante Signale und Variablen (z. B. Betriebsart, NOT-HALT, Schutztürkontakt) nicht über Submit-Interpreter oder SPS ändern.

  Wenn dennoch Änderungen notwendig sind, müssen alle sicherheitsrelevanten Signale und Variablen so verknüpft werden, dass sie vom Submit-Interpreter oder der SPS nicht in einen sicherheitsgefährdenden Zustand gesetzt werden können. Dies liegt in der Verantwortung des Systemintegrators.

OPC-Server, Remote Control Tools

Mit diesen Komponenten ist es möglich, über schreibende Zugriffe Programme, Ausgänge oder sonstige Parameter der Robotersteuerung zu ändern, ohne dass dies von in der Anlage befindlichen Personen bemerkt wird.

Sicherheitsmaßnahme:

Wenn diese Komponenten verwendet werden, müssen Ausgänge, die eine Gefährdung verursachen können, in einer Risikobeurteilung ermittelt werden. Diese Ausgänge müssen so gestaltet werden, dass sie nicht ohne Zustimmung gesetzt werden können. Dies kann beispielsweise über eine externe Zustimmeinrichtung geschehen.

Tools zur Konfiguration von Bussystemen

Wenn diese Komponenten über eine Online-Funktionalität verfügen, ist es möglich, über schreibende Zugriffe Programme, Ausgänge oder sonstige Parameter der Robotersteuerung zu ändern, ohne dass dies von in der Anlage befindlichen Personen bemerkt wird.

- WorkVisual von KUKA
- Tools anderer Hersteller

Sicherheitsmaßnahme:

In den Test-Betriebsarten dürfen Programme, Ausgänge oder sonstige Parameter der Robotersteuerung mit diesen Komponenten nicht verändert werden.

3.9 Angewandte Normen und Vorschriften

Name	Definition	Ausgabe
2006/42/EG	**Maschinenrichtlinie:** Richtlinie 2006/42/EG des Europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen und zur Änderung der Richtlinie 95/16/EG (Neufassung)	2006
2004/108/EG	**EMV-Richtlinie:** Richtlinie 2004/108/EG des Europäischen Parlaments und des Rates vom 15. Dezember 2004 zur Angleichung der Rechtsvorschriften der Mitgliedstaaten über die elektromagnetische Verträglichkeit und zur Aufhebung der Richtlinie 89/336/EWG	2004
97/23/EG	**Druckgeräterichtlinie**: Richtlinie 97/23/EG des Europäischen Parlaments und des Rates vom 29. Mai 1997 zur Angleichung der Rechtsvorschriften der Mitgliedstaaten über Druckgeräte (Findet nur Anwendung für Roboter mit hydropneumatischem Gewichtsausgleich.)	1997

EN ISO 13850	**Sicherheit von Maschinen:** NOT-HALT-Gestaltungsleitsätze	2008
EN ISO 13849-1	**Sicherheit von Maschinen:** Sicherheitsbezogene Teile von Steuerungen; Teil 1: Allgemeine Gestaltungsleitsätze	2008
EN ISO 13849-2	**Sicherheit von Maschinen:** Sicherheitsbezogene Teile von Steuerungen; Teil 2: Validierung	2012
EN ISO 12100	**Sicherheit von Maschinen:** Allgemeine Gestaltungsleitsätze, Risikobeurteilung und Risikominderung	2010
EN ISO 10218-1	**Industrieroboter:** Sicherheit **Hinweis:** Inhalt entspricht **ANSI/RIA R.15.06-2012, Teil 1**	2011
EN 614-1	**Sicherheit von Maschinen:** Ergonomische Gestaltungsgrundsätze; Teil 1: Begriffe und allgemeine Leitsätze	2009
EN 61000-6-2	**Elektromagnetische Verträglichkeit (EMV):** Teil 6-2: Fachgrundnormen; Störfestigkeit für Industriebereich	2005
EN 61000-6-4 + A1	**Elektromagnetische Verträglichkeit (EMV):** Teil 6-4: Fachgrundnormen; Störaussendung für Industriebereich	2011
EN 60204-1 + A1	**Sicherheit von Maschinen:** Elektrische Ausrüstung von Maschinen; Teil 1: Allgemeine Anforderungen	2009

4 Bedienung

4.1 Programmierhandgerät KUKA smartPAD

4.1.1 Vorderseite

Funktion

Das smartPAD ist das Programmierhandgerät für den Industrieroboter. Das smartPAD hat alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung und Programmierung des Industrieroboters benötigt werden.

Das smartPAD verfügt über einen Touch-Screen: Die smartHMI kann mit dem Finger oder einem Zeigestift bedient werden. Eine externe Maus oder externe Tastatur ist nicht notwendig.

Übersicht

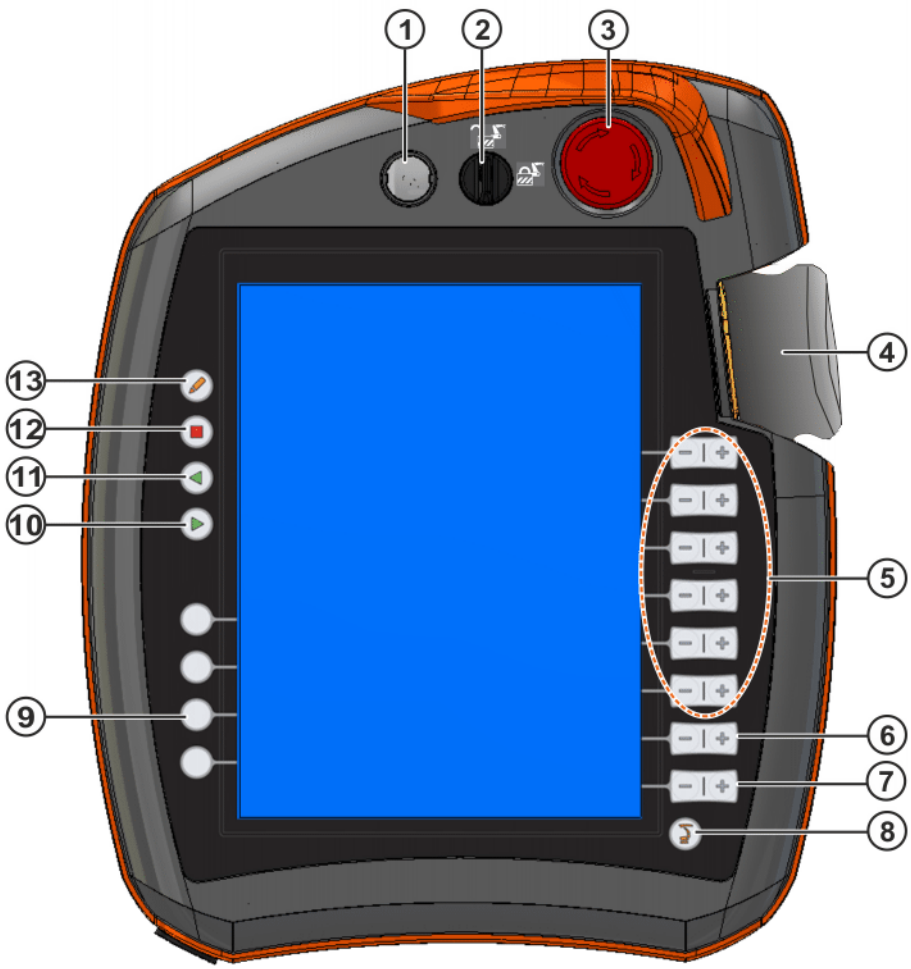


Abb. 4-1: KUKA smartPAD Vorderseite

Pos.	Beschreibung
1	Knopf zum Abstecken des smartPADs (>>> 4.1.3 "smartPAD abstecken und anstecken" Seite 50)
2	Schlüsselschalter zum Aufrufen des Verbindungs-Managers. Der Schalter kann nur umgelegt werden, wenn der Schlüssel steckt. Über den Verbindungs-Manager kann die Betriebsart gewechselt werden. (>>> 4.12 "Betriebsart wechseln" Seite 66)

Pos.	Beschreibung
3	NOT-HALT-Gerät. Zum Stoppen des Roboters in Gefahrensituationen. Das NOT-HALT-Gerät verriegelt sich, wenn es gedrückt wird.
4	Space Mouse: Zum manuellen Verfahren des Roboters (>>> 4.14 "Roboter manuell verfahren" Seite 68)
5	Verfahrtasten: Zum manuellen Verfahren des Roboters (>>> 4.14 "Roboter manuell verfahren" Seite 68)
6	Taste zum Einstellen des Programm-Overrides
7	Taste zum Einstellen des Hand-Overrides
8	Hauptmenü-Taste: Sie blendet auf der smartHMI die Menüpunkte ein (>>> 4.4 "Hauptmenü aufrufen" Seite 56)
9	Statustasten. Die Statustasten dienen hauptsächlich zur Einstellung von Parametern aus Technologiepaketen. Ihre genaue Funktion ist abhängig davon, welche Technologie-Pakete installiert sind.
10	Start-Taste: Mit der Start-Taste startet man ein Programm
11	Start-Rückwärts-Taste: Mit der Start-Rückwärts-Taste startet man ein Programm rückwärts. Das Programm wird schrittweise abgearbeitet.
12	STOP-Taste: Mit der STOP-Taste hält man ein laufendes Programm an
13	Tastatur-Taste: Blendet die Tastatur ein. In der Regel muss die Tastatur nicht eigens eingeblendet werden, da die smartHMI erkennt, wenn Eingaben über die Tastatur erforderlich sind und diese automatisch einblendet. (>>> 4.2.1 "Tastatur" Seite 52)

4.1.2 Rückseite

Übersicht

Abb. 4-2: KUKA smartPAD Rückseite

1	Zustimmungsschalter	4	USB-Anschluss
2	Start-Taste (grün)	5	Zustimmungsschalter
3	Zustimmungsschalter	6	Typenschild

Beschreibung

Element	Beschreibung
Typenschild	Typenschild
Start-Taste	Mit der Start-Taste startet man ein Programm.
Zustimmungs-schalter	Der Zustimmungsschalter hat 3 Stellungen: ■ Nicht gedrückt ■ Mittelstellung ■ Durchgedrückt Der Zustimmungsschalter muss in den Betriebsarten T1 und T2 in der Mittelstellung gehalten werden, damit der Manipulator verfahren kann. In den Betriebsarten Automatik und Automatik Extern hat der Zustimmungsschalter keine Funktion.
USB-An-schluss	Der USB-Anschluss wird z. B. verwendet für Archivierung/Wiederherstellung. Nur für FAT32 formatierte USB-Sticks.

4.1.3 smartPAD abstecken und anstecken

Beschreibung

Das smartPAD kann bei laufender Robotersteuerung abgesteckt werden.

⚠ WARNUNG Wenn das smartPAD abgesteckt ist, kann die Anlage nicht mehr über das NOT-HALT-Gerät des smartPAD abgeschaltet werden. Deshalb muss ein externer NOT-HALT an der Robotersteuerung angeschlossen werden.
Der Betreiber muss dafür sorgen, dass das abgesteckte smartPAD sofort aus der Anlage entfernt wird. Das smartPAD muss außer Sicht- und Reichweite des am Industrieroboter arbeitenden Personals verwahrt werden. Dadurch werden Verwechslungen zwischen wirksamen und nicht wirksamen NOT-HALT-Einrichtungen vermieden.
Wenn diese Maßnahmen nicht beachtet werden, können Tod, Verletzungen oder Sachschaden die Folge sein.

Vorgehensweise

Abstecken:

1. Auf dem smartPAD den Knopf zum Abstecken drücken.
   Auf der smartHMI werden eine Meldung und ein Zähler angezeigt. Der Zähler läuft 30 s. Während dieser Zeit kann das smartPAD von der Robotersteuerung abgesteckt werden.

i Wenn das smartPAD abgesteckt wird, ohne dass der Zähler läuft, löst dies einen NOT-HALT aus. Der NOT-HALT kann nur aufgehoben werden, indem das smartPAD wieder angesteckt wird.

2. Das smartPAD von der Robotersteuerung abstecken.
   Wenn der Zähler ausläuft, ohne dass das smartPAD abgesteckt wurde, hat dies keine Auswirkungen. Der Knopf zum Abstecken kann beliebig oft nochmal gedrückt werden, um den Zähler wieder anzuzeigen.

Anstecken:

- smartPAD an die Robotersteuerung anstecken.

Es kann jederzeit ein smartPAD angesteckt werden. Voraussetzung: Gleiche smartPAD-Variante wie das abgesteckte Gerät. 30 s nach dem Anstecken sind der NOT-HALT und die Zustimmungsschalter wieder funktionsfähig. Die smartHMI wird automatisch wieder angezeigt. (Kann länger dauern als 30 s.)

Das angesteckte smartPAD übernimmt die aktuelle Betriebsart der Robotersteuerung.

i Die aktuelle Betriebsart ist nicht in jedem Fall die gleiche wie vor dem Abstecken des smartPAD: Wenn die Robotersteuerung zu einem RoboTeam gehört, kann es sein, dass die Betriebsart nach dem Abstecken geändert wurde, z. B. durch den Master.

⚠ WARNUNG Der Benutzer, der ein smartPAD an die Robotersteuerung ansteckt, muss danach mindestens 30 s am smartPAD verbleiben, also bis der NOT-HALT und die Zustimmungsschalter wieder funktionsfähig sind. So wird z. B. vermieden, dass ein anderer Benutzer in einer Notsituation auf einen momentan nicht wirksamen NOT-HALT zugreift.
Wenn dies nicht beachtet wird, können Tod, Verletzungen oder Sachschäden die Folge sein.

4.2 Bedienoberfläche KUKA smartHMI

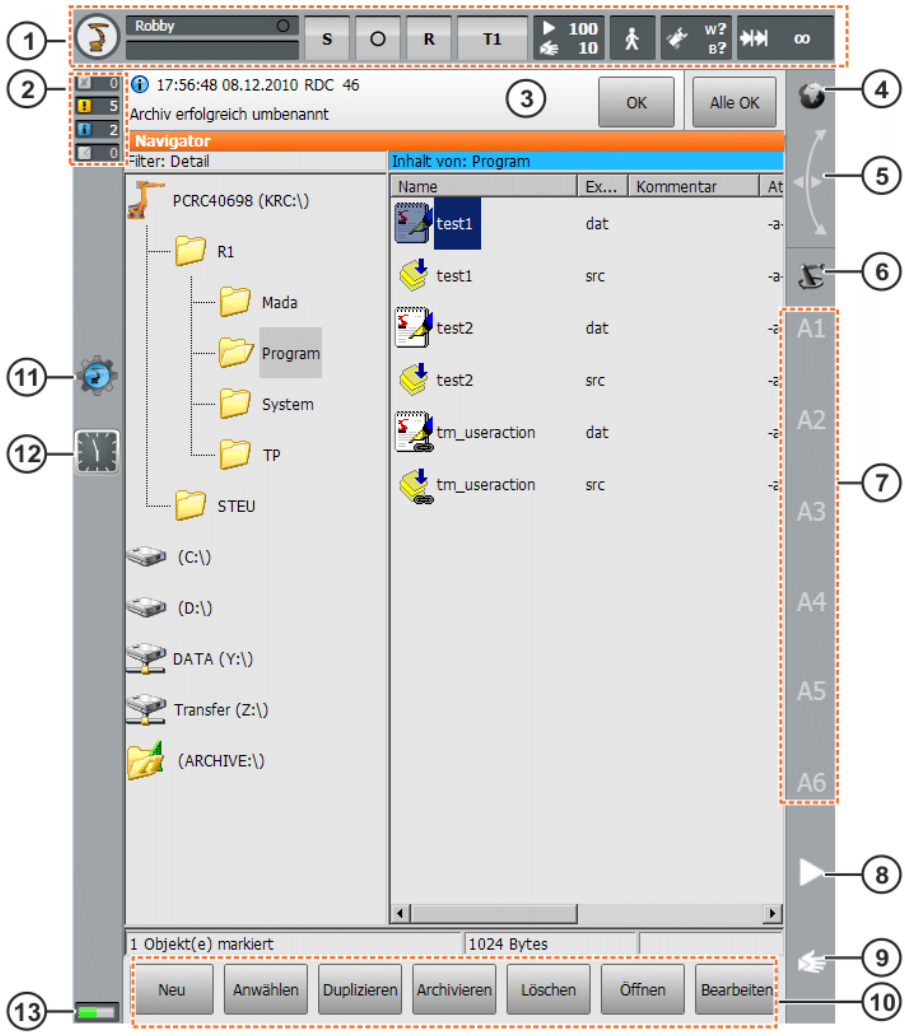


Abb. 4-3: Bedienoberfläche KUKA smartHMI

Pos.	Beschreibung
1	Statusleiste (>>> 4.2.2 "Statusleiste" Seite 53)
2	Meldungszähler Der Meldungszähler zeigt an, wieviele Meldungen von jedem Meldungstyp anstehen. Das Berühren des Meldungszählers vergrößert die Anzeige.
3	Meldungsfenster Defaultmäßig wird nur die letzte Meldung angezeigt. Das Berühren des Meldungsfensters expandiert dieses und zeigt alle anstehenden Meldungen an. Mit **OK** kann eine quittierbare Meldung quittiert werden. Mit **Alle OK** können alle quittierbaren Meldungen auf einmal quittiert werden.
4	Statusanzeige **Space Mouse** Diese Anzeige zeigt das aktuelle Koordinatensystem für das manuelle Verfahren mit der Space Mouse an. Das Berühren der Anzeige blendet alle Koordinatensysteme ein und ein anderes kann ausgewählt werden.

Pos.	Beschreibung
5	Anzeige **Ausrichtung Space Mouse** Das Berühren dieser Anzeige öffnet ein Fenster, in dem die aktuelle Ausrichtung der Space Mouse angezeigt wird und geändert werden kann. (>>> 4.14.8 "Ausrichtung der Space Mouse festlegen" Seite 77)
6	Statusanzeige **Verfahrtasten** Diese Anzeige zeigt das aktuelle Koordinatensystem für das manuelle Verfahren mit den Verfahrtasten an. Das Berühren der Anzeige blendet alle Koordinatensysteme ein und ein anderes kann gewählt werden.
7	Beschriftung Verfahrtasten Wenn das achsspezifische Verfahren ausgewählt ist, werden hier die Achsnummern angezeigt (A1, A2 etc.). Wenn das kartesische Verfahren ausgewählt ist, werden hier die Richtungen des Koordinatensystems angezeigt (X, Y, Z, A, B, C). Das Berühren der Beschriftung zeigt an, welche Kinematikgruppe ausgewählt ist.
8	Programm-Override (>>> 8.5 "Programm-Override (POV) einstellen" Seite 277)
9	Hand-Override (>>> 4.14.3 "Hand-Override (HOV) einstellen" Seite 73)
10	Schaltflächen-Leiste. Die Schaltflächen ändern sich dynamisch und beziehen sich immer auf das Fenster, das auf der smartHMI gerade aktiv ist. Ganz rechts befindet sich die Schaltfläche **Bearbeiten**. Mit ihr können zahlreiche Befehle aufgerufen werden, die sich auf den Navigator beziehen.
11	WorkVisual-Symbol Durch Berühren des Symbols gelangt man zum Fenster **Projektverwaltung**. (>>> 7.11.3 "Fenster Projektverwaltung" Seite 260)
12	Uhr Die Uhr zeigt die Systemzeit an. Das Berühren der Uhr blendet die Systemzeit in digitaler Darstellung sowie das aktuelle Datum ein.
13	Anzeige Lebenszeichen Wenn die Anzeige folgendermaßen blinkt, zeigt dies an, dass die smartHMI aktiv ist: Das linke und das rechte Lämpchen leuchten abwechselnd grün. Der Wechsel ist langsam (ca. 3 s) und gleichmäßig.

4.2.1 Tastatur

Das smartPAD verfügt über einen Touch-Screen: Die smartHMI kann mit dem Finger oder einem Zeigestift bedient werden.

Für die Eingabe von Buchstaben und Zahlen steht auf der smartHMI eine Tastatur zur Verfügung. Die smartHMI erkennt, wenn eine Eingabe von Buchstaben oder Zahlen notwendig ist und blendet die Tastatur automatisch ein.

Die Tastatur zeigt immer nur die erforderlichen Zeichen an. Wenn beispielsweise ein Feld editiert wird, in das nur Zahlen eingegeben werden dürfen, werden nur Zahlen und keine Buchstaben angezeigt.

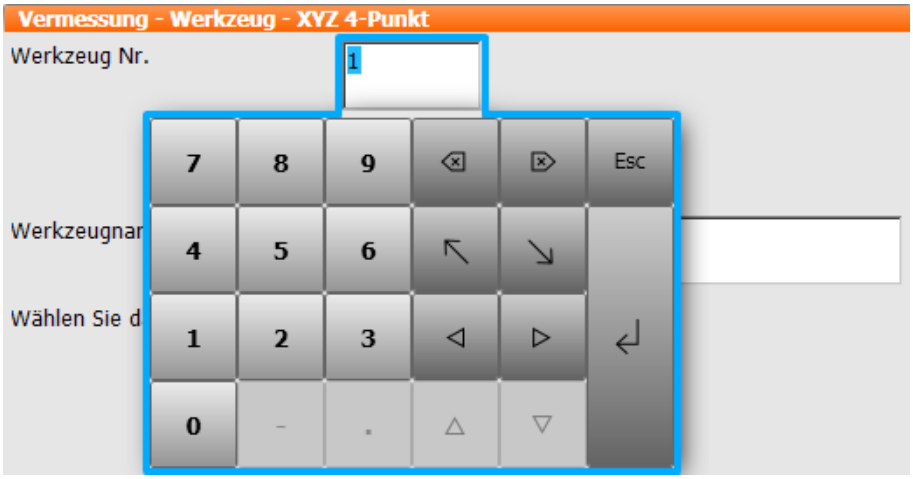


Abb. 4-4: Beispiel Tastatur

4.2.2 Statusleiste

Die Statusleiste zeigt den Zustand bestimmter zentraler Einstellungen des Industrieroboters an. Bei den meisten Anzeigen öffnet sich durch Berühren ein Fenster, in dem die Einstellungen geändert werden können.

Übersicht

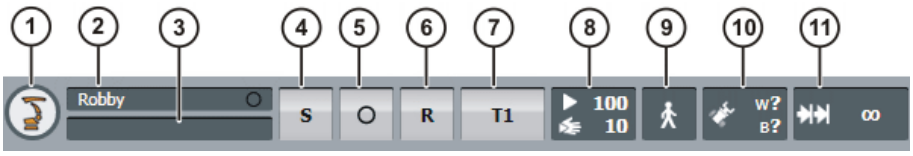


Abb. 4-5: KUKA smartHMI Statusleiste

Pos.	Beschreibung
1	Hauptmenü-Taste. Sie blendet auf der smartHMI die Menüpunkte ein. (>>> 4.4 "Hauptmenü aufrufen" Seite 56)
2	Robotername. Der Robotername kann geändert werden. (>>> 4.17.15 "Roboterdaten anzeigen/bearbeiten" Seite 94)
3	Wenn ein Programm angewählt ist, wird der Name hier angezeigt.
4	Statusanzeige **Submit-Interpreter** (>>> 12 "Submit-Interpreter" Seite 469)
5	Statusanzeige **Antriebe**. Das Berühren der Anzeige öffnet ein Fenster, in dem die Antriebe ein- oder ausgeschaltet werden können. (>>> 4.2.3 "Statusanzeige Antriebe und Fenster Fahrbedingungen" Seite 54)
6	Statusanzeige **Roboter-Interpreter**. Hier können Programme zurückgesetzt oder abgewählt werden. (>>> 8.6 "Statusanzeige Roboter-Interpreter" Seite 277) (>>> 7.5.1 "Programm anwählen und abwählen" Seite 245) (>>> 8.10 "Programm zurücksetzen" Seite 279)

Pos.	Beschreibung
7	Aktuelle Betriebsart (>>> 4.12 "Betriebsart wechseln" Seite 66)
8	Statusanzeige **POV/HOV**. Zeigt den aktuellen Programm-Override und den aktuellen Hand-Override an. (>>> 8.5 "Programm-Override (POV) einstellen" Seite 277) (>>> 4.14.3 "Hand-Override (HOV) einstellen" Seite 73)
9	Statusanzeige **Programmablaufart**. Zeigt die aktuelle Programmablaufart an. (>>> 8.2 "Programmablaufarten" Seite 273)
10	Statusanzeige **Werkzeug/Basis**. Zeigt das aktuelle Werkzeug und die aktuelle Basis an. (>>> 4.14.4 "Werkzeug und Basis auswählen" Seite 74)
11	Statusanzeige **Inkrementelles Handverfahren** (>>> 4.14.10 "Inkrementelles Handverfahren" Seite 78)

4.2.3 Statusanzeige Antriebe und Fenster Fahrbedingungen

Statusanzeige Antriebe

Die Statusanzeige **Antriebe** kann folgende Zustände anzeigen:

Zustände			

Bedeutung der Symbole und Farben:

Symbol: **I**	Die Antriebe sind EIN. ($PERI_RDY == TRUE) ■ Der Zwischenkreis ist vollständig geladen.
Symbol: **O**	Die Antriebe sind AUS. ($PERI_RDY == FALSE) ■ Der Zwischenkreis ist nicht oder nicht vollständig geladen.
Farbe: Grün	$COULD_START_MOTION == TRUE ■ Der Zustimmungsschalter ist gedrückt (Mittelstellung), oder er ist nicht erforderlich. ■ Und: Es stehen keine Meldungen an, die das Verfahren des Roboters verhindern.
Farbe: Grau	$COULD_START_MOTION == FALSE ■ Der Zustimmungsschalter ist nicht gedrückt oder durchgedrückt. ■ Und/oder: Es stehen Meldungen an, die das Verfahren des Roboters verhindern.

■ Antriebe EIN bedeutet nicht automatisch, dass die KSPs in Regelung gehen und die Motoren mit Strom versorgen.

■ Antriebe AUS bedeutet nicht automatisch, dass die KSPs die Stromversorgung der Motoren abbrechen.

Ob die KSPs die Motoren mit Strom versorgen, ist abhängig davon, ob von der Sicherheitssteuerung die Antriebsfreigabe vorliegt.

Fenster Fahrbedingungen

Das Berühren der Statusanzeige **Antriebe** öffnet das Fenster **Fahrbedingungen**. Hier können die Antriebe ein- oder ausgeschaltet werden.

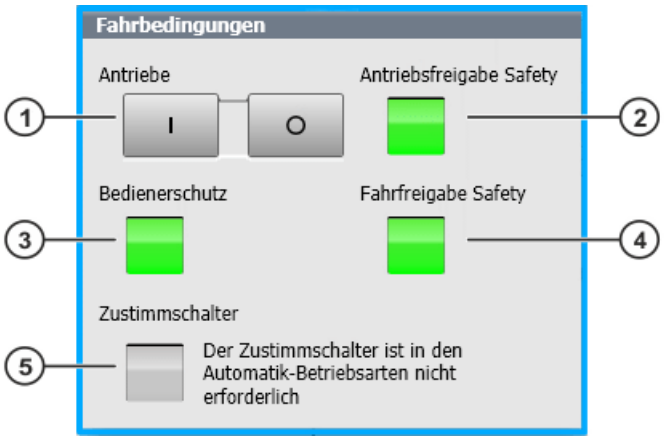


Abb. 4-6: Fenster Fahrbedingungen

Pos.	Beschreibung
1	**I**: Berühren, um die Antriebe einzuschalten. **O**: Berühren, um die Antriebe auszuschalten.
2	**Grün**: Die Antriebsfreigabe von der Sicherheitssteuerung ist vorhanden. **Grau**: Die Sicherheitssteuerung hat einen Sicherheitshalt 0 ausgelöst oder einen Sicherheitshalt 1 beendet. Keine Antriebsfreigabe vorhanden, d. h. die KSPs sind nicht in Regelung und versorgen die Motoren nicht mit Strom.
3	Signal **Bedienerschutz** **Grün**: $USER_SAF == TRUE **Grau**: $USER_SAF == FALSE (>>> "$USER_SAF == TRUE" Seite 55)
4	**Grün**: Die Fahrfreigabe von der Sicherheitssteuerung ist vorhanden. **Grau**: Die Sicherheitssteuerung hat einen Sicherheitshalt 1 oder einen Sicherheitshalt 2 ausgelöst. Keine Fahrfreigabe vorhanden. **Hinweis:** Der Zustand von **Fahrfreigabe Safety** korreliert nicht mit dem Zustand von $MOVE_ENABLE!
5	**Grün**: Der Zustimmungsschalter ist gedrückt (Mittelstellung). **Grau**: Der Zustimmungsschalter ist nicht gedrückt oder durchgedrückt, oder er ist nicht erforderlich.

$USER_SAF == TRUE

Die Bedingungen, unter denen $USER_SAF TRUE ist, sind abhängig von der Steuerungsvariante und der Betriebsart:

Steuerung	Betriebsart	Bedingung
KR C4	T1, T2	■ Die Zustimmung ist gedrückt.
	AUT, AUT EXT	■ Die trennende Schutzeinrichtung ist geschlossen.

Steuerung	Betriebsart	Bedingung
VKR C4	T1	■ Die Zustimmung ist gedrückt. ■ E2 ist geschlossen.
	T2	■ Die Zustimmung ist gedrückt. ■ E2 und E7 sind geschlossen
	AUT EXT	■ Die trennende Schutzeinrichtung ist geschlossen. ■ E2 und E7 sind offen.

4.2.4 KUKA smartHMI minimieren (Windows-Ebene anzeigen)

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **HMI minimieren** wählen.
   Die smartHMI wird minimiert und die Windows-Ebene wird sichtbar.
2. Um die smartHMI wieder zu maximieren, in der Taskleiste folgendes Icon berühren:

4.3 Robotersteuerung einschalten und KSS starten

Vorgehensweise

- Hauptschalter an der Robotersteuerung auf ON stellen.
  Das Betriebssystem und die KSS starten automatisch.

Wenn die KSS nicht automatisch startet, beispielsweise weil der Autostart unterbunden wurde, im Pfad C:\KRC das Programm StartKRC.exe starten.

Wenn die Robotersteuerung am Netzwerk angemeldet wird, kann der Start länger dauern.

4.4 Hauptmenü aufrufen

Vorgehensweise

- Hauptmenü-Taste auf dem smartPAD drücken. Das Fenster **Hauptmenü** öffnet sich.
  Es wird immer die Ansicht gezeigt, die das Fenster beim letzten Schließen hatte.

Beschreibung

Eigenschaften Fenster **Hauptmenü**:

- In der linken Spalte wird das Hauptmenü angezeigt.
- Das Berühren eines Menüpunkts mit Pfeil blendet das zugehörige Untermenü ein (z. B. **Konfiguration**).
  Je nachdem, wie viele ineinander verschachtelte Untermenüs geöffnet sind, kann es sein, dass die Spalte **Hauptmenü** nicht mehr sichtbar ist, sondern nur noch die Untermenüs.
- Die Pfeiltaste rechts oben blendet das zuletzt geöffnete Untermenü wieder aus.
- Die Home-Taste links oben blendet alle geöffneten Untermenüs aus.
- Im unteren Bereich werden die zuletzt gewählten Menüpunkte angezeigt (maximal 6).
  Dies ermöglicht es, diese Menüpunkte direkt wieder auszuwählen, ohne vorher eventuell geöffnete Untermenüs schließen zu müssen.

- Das weiße Kreuz links schließt das Fenster.

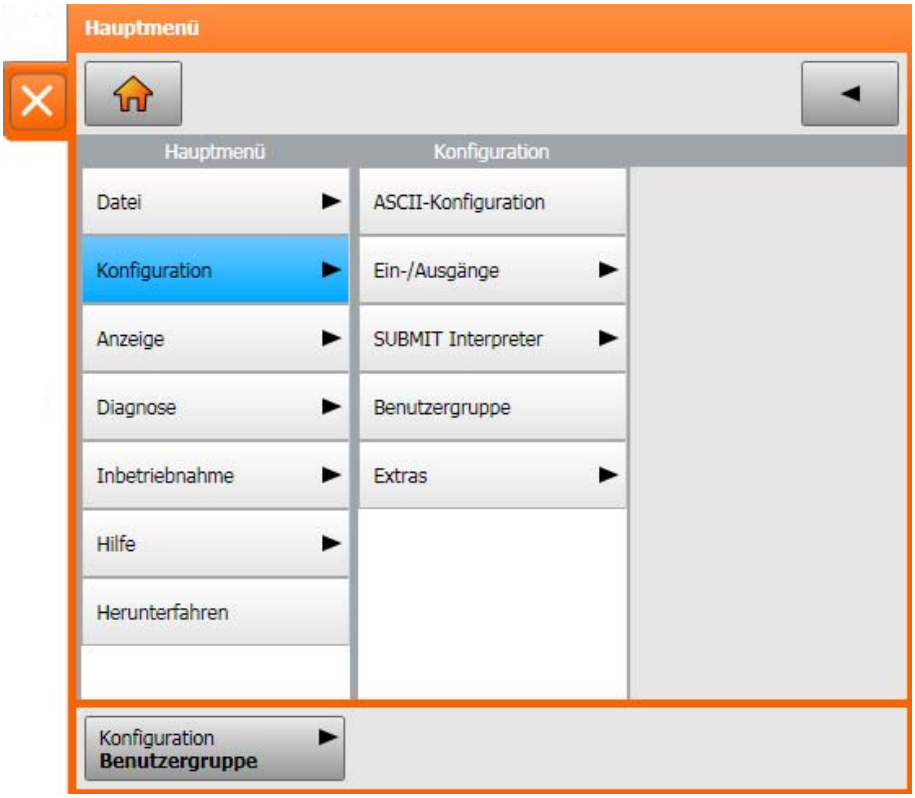


Abb. 4-7: Beispiel: Untermenü Konfiguration ist geöffnet

4.5 Starttyp für KSS festlegen

Beschreibung

Man kann wählen, ob die Robotersteuerung standardmäßig mit Kaltstart oder mit Hibernate starten soll.

> In folgenden Fällen führt die Robotersteuerung immer einen initialen Kaltstart aus, unabhängig davon, welcher Starttyp gewählt wurde:
> - Nach Installation oder Update der KSS
> - Wenn die Robotersteuerung beim Herunterfahren einen Fehler festgestellt hat

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Herunterfahren** wählen. Ein Fenster öffnet sich.
2. Starttyp wählen: **Kaltstart** oder **Hibernate**.
   (>>> "Starttypen" Seite 60)
3. Das Fenster schließen. Der gewählte Starttyp wird übernommen.

Speziell für den nächsten Start können Einstellungen gewählt werden, die vom Standard-Starttyp abweichen.

(>>> 4.6 "KSS beenden oder neu starten" Seite 57)

4.6 KSS beenden oder neu starten

Voraussetzung

- Für bestimmte Optionen: Benutzergruppe Experte

HINWEIS Wenn beim Beenden die Option **Steuerungs-PC neu starten** gewählt wird, darf der Hauptschalter an der Robotersteuerung nicht betätigt werden, solange der Neustart noch nicht abgeschlossen ist. Systemdateien können sonst zerstört werden.
Wenn beim Beenden diese Option nicht gewählt wurde, kann der Hauptschalter betätigt werden, wenn die Steuerung heruntergefahren ist.

Vorgehensweise

1. Im Hauptmenü **Herunterfahren** wählen.
2. Die gewünschten Optionen wählen.
3. Auf **Steuerungs-PC herunterfahren** oder **Steuerungs-PC neu starten** drücken.
4. Sicherheitsabfrage mit **Ja** bestätigen. Die System Software wird beendet und je nach gewählter Option wieder gestartet.

Nach dem Neustart wird folgende Meldung angezeigt:

- *Steuerungskaltstart*
- Oder, wenn **Dateien neu einlesen** gewählt wurde: *Initialer Steuerungskaltstart*

Beschreibung

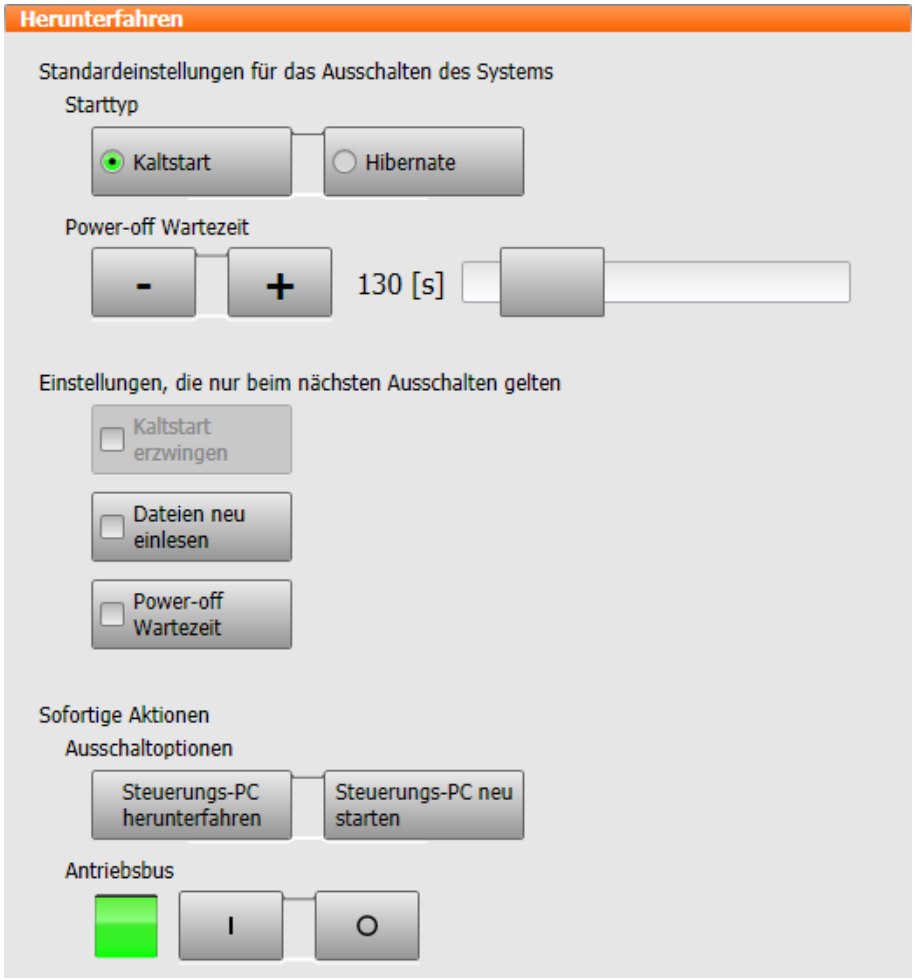


Abb. 4-8: Fenster Herunterfahren

Option	Beschreibung
Standardeinstellungen für das Ausschalten des Systems Diese Einstellungen können nur in der Benutzergruppe Experte geändert werden.	
Kaltstart	**Kaltstart** ist der Standard-Starttyp. (>>> "Starttypen" Seite 60)
Hibernate	**Hibernate** ist der Standard-Starttyp.

Option	Beschreibung
Power-off Wartezeit	Wenn die Robotersteuerung mittels Hauptschalter ausgeschaltet wird, fährt sie erst nach der hier festgelegten Wartezeit herunter. Während der Wartezeit wird die Robotersteuerung über ihren Akku versorgt. Die Wartezeit kann nur in der Benutzergruppe Experte geändert werden. **Hinweise:** ■ Die Power-off-Wartezeit gilt nur, wenn die Spannung über den Hauptschalter abgeschaltet wird. Für echte Spannungsausfälle gilt die Power-fail-Wartezeit. ■ Ausnahme "KR C4 compact": Bei dieser Steuerungsvariante ist die Power-off-Wartezeit ohne Funktion! Hier gilt auch beim Abschalten über den Hauptschalter die Power-fail-Wartezeit. (>>> 4.6.1 "Herunterfahren nach Spannungsausfall" Seite 60)
Einstellungen, die nur beim nächsten Ausschalten gelten	
Kaltstart erzwingen	Aktiv: Der nächste Start ist ein Kaltstart. Steht nur zur Verfügung, wenn **Hibernate** ausgewählt ist.
Dateien neu einlesen	Aktiv: Der nächste Start ist ein initialer Kaltstart. Diese Option muss in folgenden Fällen gewählt werden: ■ Wenn XML-Dateien direkt geändert worden sind, d. h. wenn der Benutzer die Datei geöffnet und geändert hat. (Etwaige andere Änderungen an XML-Dateien, z. B. wenn die Robotersteuerung diese im Hintergrund ändert, sind unrelevant.) ■ Wenn nach dem Herunterfahren Hardware-Komponenten getauscht werden sollen. Kann nur in der Benutzergruppe Experte gewählt werden. Steht nur zur Verfügung, wenn **Kaltstart** oder **Kaltstart erzwingen** ausgewählt ist. Je nach Hardware dauert der initiale Kaltstart ca. 30 bis 150 sec länger als ein normaler Kaltstart.
Power-off Wartezeit	Aktiv: Die Wartezeit wird beim nächsten Herunterfahren eingehalten. Inaktiv: Die Wartezeit wird beim nächsten Herunterfahren ignoriert.
Sofortige Aktionen Stehen nur in den Betriebsarten T1 und T2 zur Verfügung.	
Steuerungs-PC herunterfahren	Die Robotersteuerung fährt herunter.
Steuerungs-PC neu starten	Die Robotersteuerung fährt herunter und startet dann mit einem Kaltstart neu.
Antriebsbus **AUS** / **EIN**	Der Antriebsbus kann aus- oder eingeschaltet werden. Anzeige **Status des Antriebsbusses**: ■ grün: Antriebsbus ist an. ■ rot: Antriebsbus ist aus. ■ grau: Status des Antriebsbusses ist unbekannt.

Starttypen

Starttyp	Beschreibung
Kaltstart	Nach einem Kaltstart zeigt die Robotersteuerung den Navigator an. Es ist kein Programm angewählt. Die Robotersteuerung wird neu initialisiert, z. B. werden alle Anwenderausgänge auf FALSE gesetzt. **Hinweis:** Wenn XML-Dateien direkt geändert worden sind, d. h. wenn der Benutzer die Datei geöffnet und geändert hat, werden diese Änderungen bei einem Kaltstart mit **Dateien neu einlesen** berücksichtigt. Dieser Kaltstart heißt "initialer Kaltstart". Bei einem Kaltstart ohne **Dateien neu einlesen** bleiben diese Änderungen unberücksichtigt.
Hibernate	Nach einem Start mit Hibernate kann das vorher angewählte Roboterprogramm fortgesetzt werden. Der Zustand des Grundsystems, wie Programme, Satzzeiger, Variableninhalte und Ausgänge, wird komplett wieder hergestellt. Zusätzlich sind alle Programme, die parallel zur Robotersteuerung geöffnet waren, wieder geöffnet und in dem Zustand, den sie vor dem Herunterfahren hatten. Bei Windows wird ebenfalls der letzte Zustand wieder hergestellt.

4.6.1 Herunterfahren nach Spannungsausfall

Bei einem Spannungsausfall bleibt der Roboter stehen. Die Robotersteuerung fährt jedoch nicht gleich herunter, sondern erst nach Ablauf der Power-fail-Wartezeit. Kurze Spannungsaufälle werden also mit Hilfe dieser Wartezeit überbrückt. Danach müssen lediglich die Fehlermeldungen quittiert werden und das Programm kann fortgesetzt werden.

Während der Wartezeit wird die Robotersteuerung über ihren Akku versorgt.

Robotersteuerung	Power-fail-Wartezeit
Variante "KR C4 compact"	1 sec
Alle anderen Varianten der KR C4	3 sec

Wenn der Spannungsausfall länger dauert als die Power-fail-Wartezeit und die Robotersteuerung herunterfährt, dann gilt für den Neustart der Standard-Starttyp, der im Fenster **Herunterfahren** festgelegt ist.

(>>> 4.6 "KSS beenden oder neu starten" Seite 57)

- Die Power-fail-Wartezeit gilt nicht, wenn die Spannung über den Hauptschalter abgeschaltet wird. Hierfür gilt die Power-off-Wartezeit.
  Ausnahme "KR C4 compact": Bei dieser Steuerungsvariante gilt auch beim Abschalten über den Hauptschalter die Power-fail-Wartezeit.
- Die Power-fail-Wartezeit ist besonders von Bedeutung für Anlagen, die keine zuverlässige Netzversorgung haben. Wartezeiten bis 240 sec sind möglich. Wenn die bestehenden Zeiten geändert werden sollen, bitte Kontakt zur KUKA Roboter GmbH aufnehmen.

4.7 Antriebe ein-/ausschalten

Vorgehensweise

1. In der Statusleiste die Anzeige **Antriebe** berühren. Das Fenster **Fahrbedingungen** öffnet sich.

(>>> 4.2.3 "Statusanzeige Antriebe und Fenster Fahrbedingungen" Seite 54)

2. Die Antriebe ein- oder ausschalten.

4.8 Robotersteuerung ausschalten

> Der Hauptschalter darf nicht betätigt werden, wenn die Robotersteuerung vorher mit der Option **Steuerungs-PC neu starten** beendet wurde und der Neustart noch nicht abgeschlossen ist. Systemdateien können sonst zerstört werden.

Vorgehensweise

- Den Hauptschalter an der Robotersteuerung auf OFF stellen.

Beschreibung

Der Roboter bleibt stehen und die Robotersteuerung fährt herunter. Die Robotersteuerung sichert die Daten automatisch.

Wenn eine Power-off-Wartezeit konfiguriert ist, fährt die Robotersteuerung erst nach Ablauf dieser Zeit herunter. Kurze Spannungsabschaltungen werden also mit Hilfe dieser Wartezeit überbrückt. Danach müssen lediglich die Fehlermeldungen quittiert werden und das Programm kann fortgesetzt werden.

Während der Wartezeit wird die Robotersteuerung über ihren Akku versorgt.

4.9 Sprache der Bedienoberfläche einstellen

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Sprache** wählen.
2. Gewünschte Sprache markieren. Mit **OK** bestätigen.

Beschreibung

Folgende Sprachen stehen zur Auswahl:

Chinesisch (Kurzzeichen)	Polnisch
Dänisch	Portugiesisch
Deutsch	Rumänisch
Englisch	Russisch
Finnisch	Schwedisch
Französisch	Slowakisch
Griechisch	Slowenisch
Italienisch	Spanisch
Japanisch	Tschechisch
Koreanisch	Türkisch
Niederländisch	Ungarisch

4.10 Online-Dokumentation und Online-Hilfe

4.10.1 Online-Dokumentation aufrufen

Beschreibung

Die Dokumentation zur KUKA System Software kann auf der Robotersteuerung angezeigt werden. Einige Technologiepakete verfügen ebenfalls über Dokumentationen, die auf der Robotersteuerung angezeigt werden können.

Vorgehensweise

1. Im Hauptmenü **Hilfe** > **Dokumentation** wählen. Dann entweder **Systemsoftware** wählen oder den Menüpunkt für das Technologiepaket.
   Das Fenster **KUKA Embedded Information Service** öffnet sich. Das Inhaltsverzeichnis der Dokumentation wird angezeigt.

2. Ein Kapitel berühren. Die enthaltenen Themen werden angezeigt.
3. Ein Thema berühren. Die Beschreibung wird angezeigt.

Beispiel

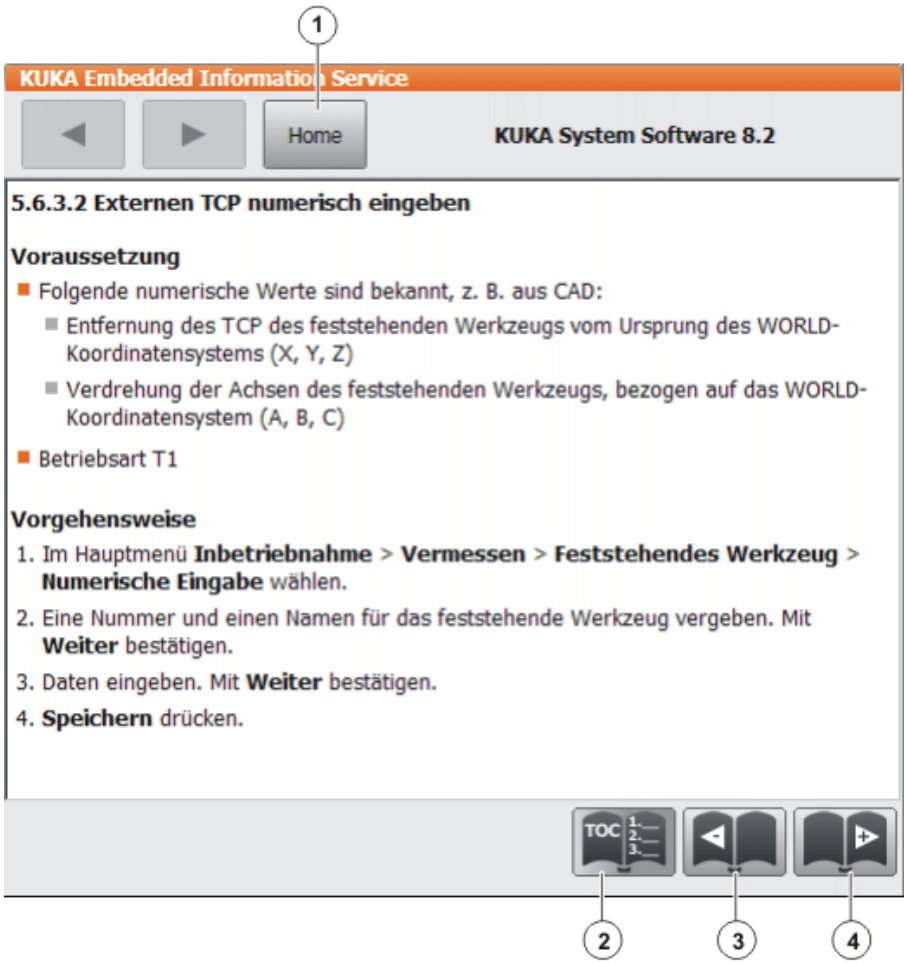


Abb. 4-9: Online-Dokumentation - Beispiel aus der KUKA System Software

Pos.	Beschreibung
1, 2	Zeigt das Inhaltsverzeichnis an.
3	Zeigt das im Inhaltsverzeichnis vorausgehende Thema an.
4	Zeigt das folgende Thema an.

4.10.2 Online-Hilfe aufrufen

Beschreibung

Die Online-Hilfe bezieht sich auf die Meldungen. Es gibt folgende Möglichkeiten, die Online-Hilfe aufzurufen:

- Zu einer Meldung, die aktuell im Meldungsfenster angezeigt wird, die Hilfe aufrufen.
- Eine Übersicht der möglichen Meldungen anzeigen und dort zu einer Meldung die Hilfe aufrufen.

Vorgehensweise

Die Online-Hilfe zu einer Meldung im Meldungsfenster aufrufen

Die meisten Meldungen enthalten einen Fragezeichen-Button. Für diese Meldungen steht eine Online-Hilfe zur Verfügung.

1. Den Fragezeichen-Button berühren. Das Fenster **KUKA Embedded Information Service** – **Meldungsseite** öffnet sich.

Das Fenster enthält verschiedene Informationen zur Meldung. (>>> Abb. 4-10)

2. Oft enthält das Fenster auch Infos zu den Ursachen der Meldung und den dazugehörigen Lösungen. Hierzu kann man Details anzeigen:
   a. Den Lupen-Button neben der Ursache berühren. Die Detailseite öffnet sich. (>>> Abb. 4-11)
   b. Die Beschreibungen zu Ursache und Lösung aufklappen.
   c. Wenn die Meldung mehrere mögliche Ursachen hat: Über die Lupen-Buttons mit Pfeil kann man zur vorherigen oder nächsten Detailseite springen.

Vorgehensweise **Eine Übersicht der Meldungen anzeigen und zu einer Meldung die Online-Hilfe aufrufen**

1. Im Hauptmenü **Hilfe** > **Meldungen** wählen. Dann entweder **Systemsoftware** wählen oder den Menüpunkt für das Technologiepaket.

   Das Fenster **KUKA Embedded Information Service** – **Indexseite** öffnet sich. Die Meldungen sind nach Modulen sortiert. (Unter "Modul" ist hier ein Teilbereich der Software zu verstehen.)
2. Einen Eintrag berühren. Die Meldungen dieses Moduls werden angezeigt.
3. Eine Meldung berühren. Die Meldungsseite wird angezeigt.

   Das Fenster enthält verschiedene Informationen zur Meldung. (>>> Abb. 4-10)
4. Oft enthält das Fenster auch Infos zu den Ursachen der Meldung und den dazugehörigen Lösungen. Hierzu kann man Details anzeigen:
   a. Den Lupen-Button neben der Ursache berühren. Die Detailseite öffnet sich. (>>> Abb. 4-11)
   b. Die Beschreibungen zu Ursache und Lösung aufklappen.
   c. Wenn die Meldung mehrere mögliche Ursachen hat: Über die Lupen-Buttons mit Pfeil kann man zur vorherigen oder nächsten Detailseite springen.

Meldungsseite

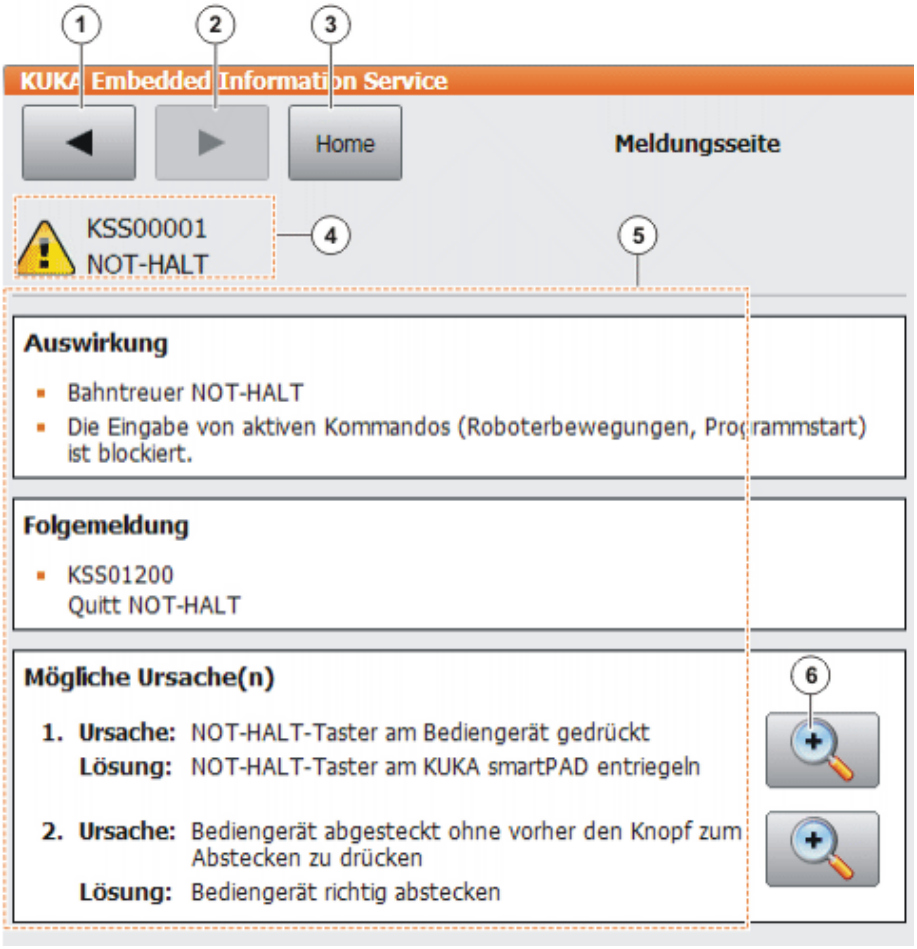


Abb. 4-10: Meldungsseite – Beispiel aus der KUKA System Software

Pos.	Beschreibung
1	Zeigt die vorherige Seite an.
2	Dieser Button ist nur aktiv, wenn mit dem anderen Pfeil-Button zur vorherigen Seite gesprungen wurde. Über diesen Button kann man dann wieder zur ursprünglichen Seite zurückkehren.
3	Zeigt die Liste mit den Software-Modulen an.
4	Meldungsnummer und -text
5	Informationen zur Meldung Es können auch weniger Informationen vorhanden sein als im Beispiel.
6	Zeigt Details zu dieser Ursache/Lösung an. (>>> Abb. 4-11)

Detailseite

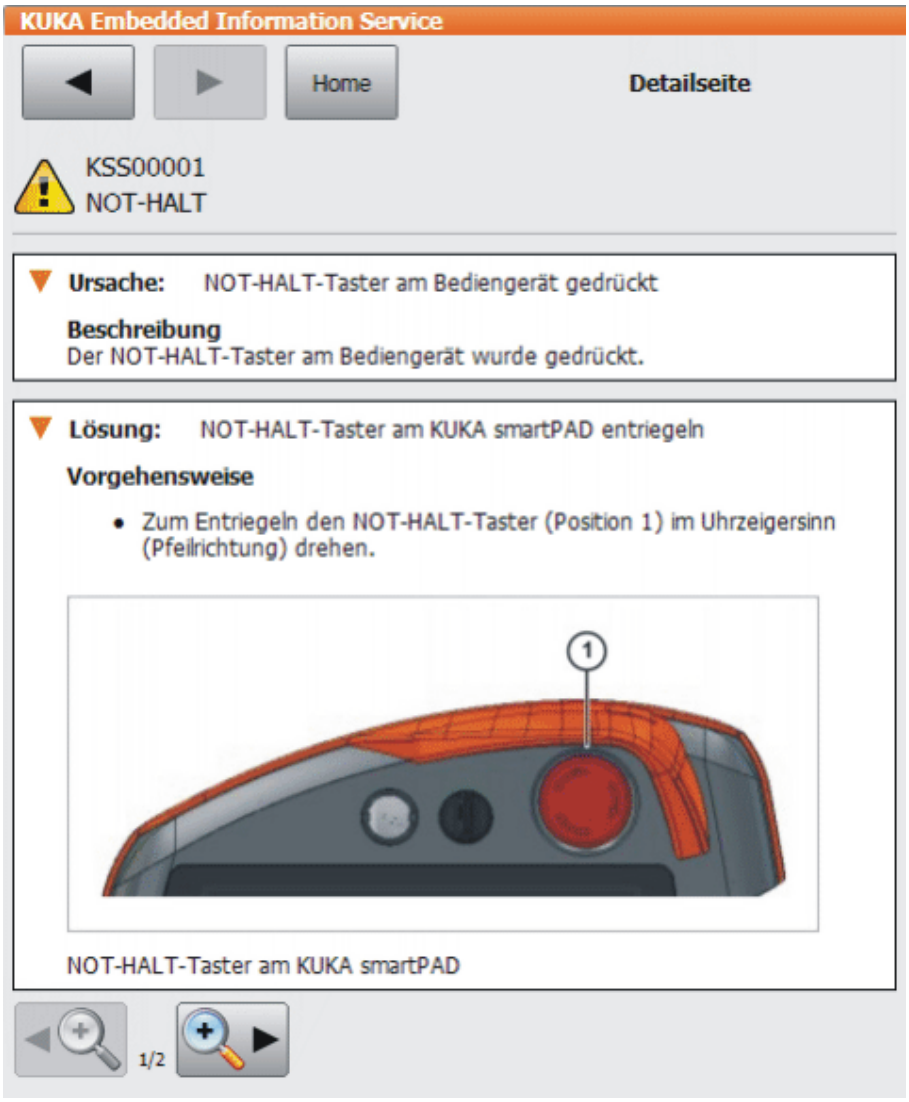


Abb. 4-11: Detailseite – Beispiel aus der KUKA System Software

4.11 Benutzergruppe wechseln

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Benutzergruppe** wählen. Die aktuelle Benutzergruppe wird angezeigt.
2. Um in die Default-Benutzergruppe zu wechseln: **Standard** drücken. (**Standard** steht nicht zur Verfügung, wenn man sich bereits in der Default-Benutzergruppe befindet.)

   Um in eine andere Benutzergruppe zu wechseln: **Anmelden…** drücken. Die gewünschte Benutzergruppe markieren.
3. Falls gefordert: Passwort eingeben und mit **Anmelden** bestätigen.

Beschreibung

In der KSS stehen je nach Benutzergruppe unterschiedliche Funktionen zur Verfügung. Es gibt folgende Benutzergruppen:

- **Bediener**

  Benutzergruppe für den Bediener. Dies ist die Default-Benutzergruppe.
- **Anwender**

  Benutzergruppe für den Bediener. (Die Benutzergruppen Bediener und Anwender sind defaultmäßig für die gleiche Zielgruppe angelegt.)

- **Experte**

  Benutzergruppe für den Programmierer. Diese Benutzergruppe ist durch ein Passwort geschützt.
- **Sicherheitsinstandhalter**

  Diese Benutzergruppe kann die Sicherheitskonfiguration des Roboters aktivieren und konfigurieren. Diese Benutzergruppe ist durch ein Passwort geschützt.
- **Sicherheitsinbetriebnehmer**

  Diese Benutzergruppe ist nur relevant, wenn KUKA.SafeOperation oder KUKA.SafeRangeMonitoring verwendet wird. Die Benutzergruppe ist durch ein Passwort geschützt.
- **Administrator**

  Funktionen wie bei der Benutzergruppe Experte. Zusätzlich ist die Integration von Plug-Ins in die Robotersteuerung möglich.

  Diese Benutzergruppe ist durch ein Passwort geschützt.

Das Default-Passwort lautet "kuka". Das Passwort kann geändert werden.

(>>> 6.9 "Passwort ändern" Seite 176)

Beim Neustart ist die Default-Benutzergruppe ausgewählt.

Wenn in die Betriebsart AUT oder AUT EXT gewechselt wird, wechselt die Robotersteuerung aus Sicherheitsgründen in die Default-Benutzergruppe. Wenn eine andere Benutzergruppe gewünscht ist, muss danach in diese gewechselt werden.

Wenn während einer bestimmten Zeitdauer an der Bedienoberfläche keine Handlung erfolgt, wechselt die Robotersteuerung aus Sicherheitsgründen in die Default-Benutzergruppe. Die Default-Einstellung ist 300 s.

4.12 Betriebsart wechseln

Die Betriebsart nicht wechseln, während ein Programm abgearbeitet wird. Wenn die Betriebsart gewechselt wird, während ein Programm abgearbeitet wird, stoppt der Industrieroboter mit einem Sicherheitshalt 2.

Voraussetzung

- Die Robotersteuerung arbeitet kein Programm ab.
- Schlüssel für den Schalter zum Aufrufen des Verbindungs-Managers

Vorgehensweise

1. Am smartPAD den Schalter für den Verbindungs-Manager umlegen. Der Verbindungs-Manager wird angezeigt.
2. Die Betriebsart wählen (>>> 3.5.3 "Betriebsartenwahl" Seite 27).
3. Den Schalter für den Verbindungs-Manager wieder in die ursprüngliche Position bringen.

   Die gewählte Betriebsart wird in der Statusleiste des smartPAD angezeigt.

Betriebsart	Verwendung	Geschwindigkeiten
T1	Für Testbetrieb, Programmierung und Teachen	■ Programmverifikation: Programmierte Geschwindigkeit, maximal 250 mm/s ■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s
T2	Für Testbetrieb	■ Programmverifikation: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich
AUT	Für Industrieroboter ohne übergeordnete Steuerung	■ Programmbetrieb: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich
AUT EXT	Für Industrieroboter mit einer übergeordneten Steuerung, z. B. SPS	■ Programmbetrieb: Programmierte Geschwindigkeit ■ Handbetrieb: Nicht möglich

4.13 Koordinatensysteme

Übersicht

In der Robotersteuerung sind folgende kartesische Koordinatensysteme definiert:

- WORLD
- ROBROOT
- BASE
- TOOL

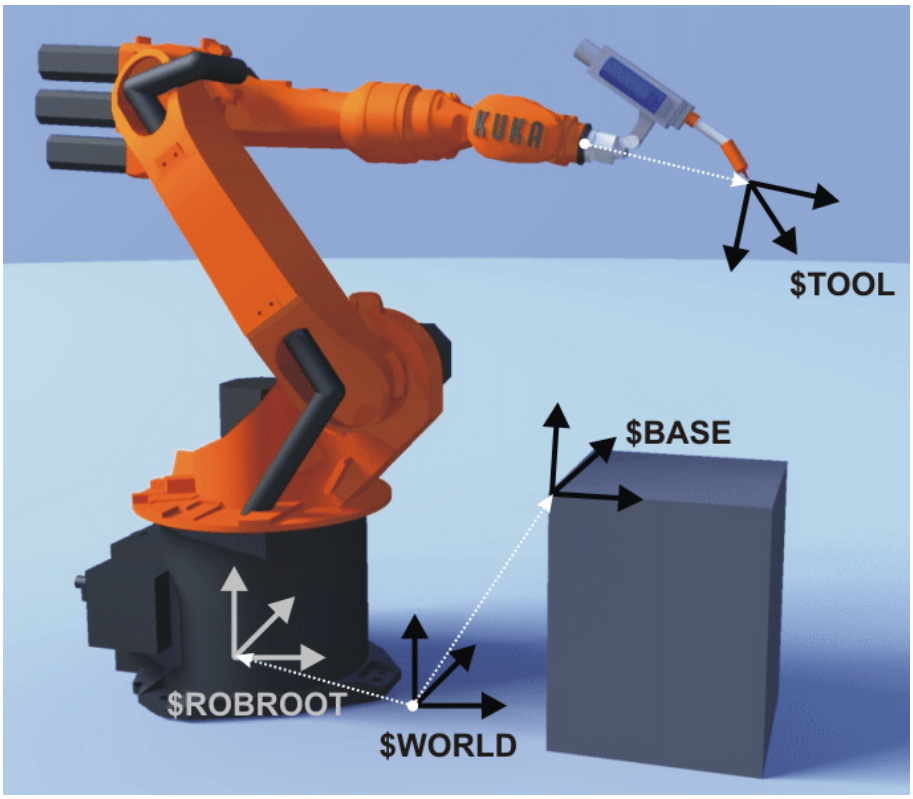


Abb. 4-12: Übersicht Koordinatensysteme

Beschreibung

WORLD

Das WORLD-Koordinatensystem ist ein fest definiertes kartesisches Koordinatensystem. Es ist das Ursprungskoordinatensystem für die Koordinatensysteme ROBROOT und BASE.

Defaultmäßig liegt das WORLD-Koordinatensystem im Roboterfuß.

ROBROOT

Das ROBROOT-Koordinatensystem ist ein kartesisches Koordinatensystem, das immer im Roboterfuß liegt. Es beschreibt die Position des Roboters in Bezug auf das WORLD-Koordinatensystem.

Defaultmäßig ist das ROBROOT-Koordinatensystem mit dem WORLD-Koordinatensystem deckungsgleich. Mit $ROBROOT kann eine Verschiebung des Roboters zum WORLD-Koordinatensystem definiert werden.

BASE

Das BASE-Koordinatensystem ist ein kartesisches Koordinatensystem, das die Position des Werkstücks beschreibt. Es bezieht sich auf das WORLD-Koordinatensystem.

Defaultmäßig ist das BASE-Koordinatensystem mit dem WORLD-Koordinatensystem deckungsgleich. Es wird vom Benutzer in das Werkstück verschoben.

(>>> 5.10.3 "Basis vermessen" Seite 136)

TOOL

Das TOOL-Koordinatensystem ist ein kartesisches Koordinatensystem, das im Arbeitspunkt des Werkzeugs liegt.

Defaultmäßig liegt der Ursprung des TOOL-Koordinatensystems im Flanschmittelpunkt. (Es wird dann FLANGE-Koordinatensystem genannt.) Das TOOL-Koordinatensystem wird vom Benutzer in den Arbeitspunkt des Werkzeugs verschoben.

(>>> 5.10.2 "Werkzeug vermessen" Seite 128)

Drehwinkel der Roboter-Koordinatensysteme

Winkel	Drehung um Achse
Winkel A	Drehung um die Z-Achse
Winkel B	Drehung um die Y-Achse
Winkel C	Drehung um die X-Achse

4.14 Roboter manuell verfahren

Beschreibung

Es gibt 2 Arten, den Roboter manuell zu verfahren:

- Kartesisch verfahren

  Der TCP wird in positiver oder negativer Richtung entlang der Achsen eines Koordinatensystems verfahren.
- Achsspezifisch verfahren

  Jede Achse kann einzeln in positiver und negativer Richtung verfahren werden.

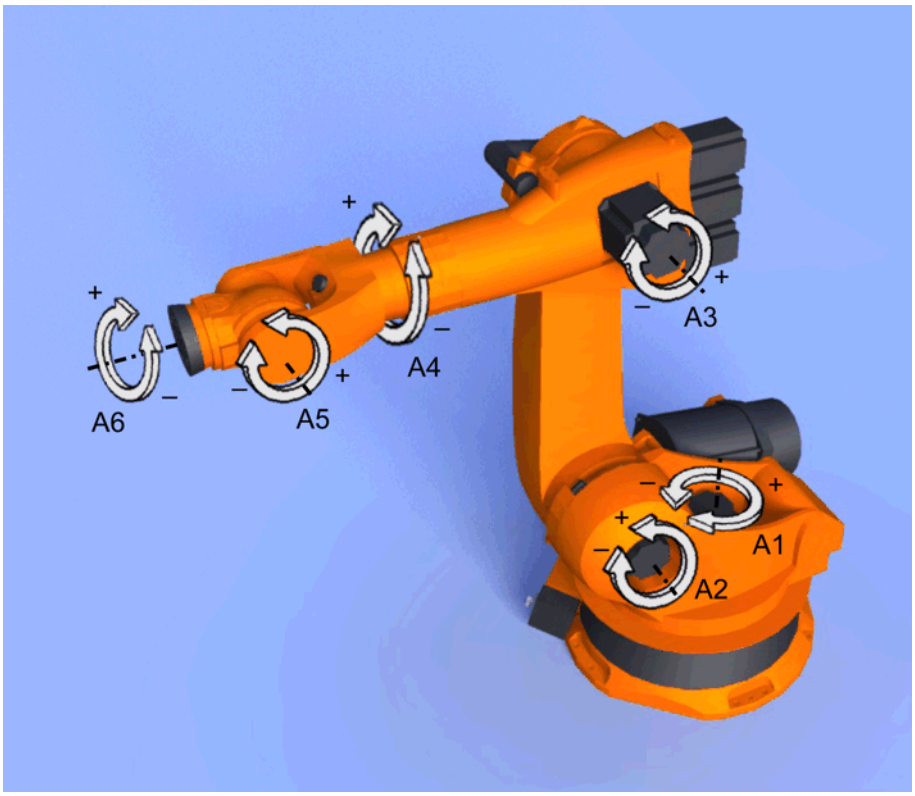


Abb. 4-13: Achsspezifisch verfahren

Es gibt 2 Bedienelemente, mit denen der Roboter verfahren werden kann:

- Verfahrtasten
- Space Mouse

Übersicht

	Kartesisch verfahren	Achsspezifisch verfahren
Verfahrtasten	(>>> 4.14.6 "Mit Verfahrtasten kartesisch verfahren" Seite 74)	(>>> 4.14.5 "Mit Verfahrtasten achsspezifisch verfahren" Seite 74)
Space Mouse	(>>> 4.14.9 "Mit Space Mouse kartesisch verfahren" Seite 78)	Das achsspezifische Verfahren mit der Space Mouse ist möglich, wird jedoch nicht beschrieben.

4.14.1 Fenster Handverfahroptionen

Beschreibung

Alle Parameter für das manuelle Verfahren des Roboters können im Fenster **Handverfahroptionen** eingestellt werden.

Vorgehensweise

Fenster **Handverfahroptionen** öffnen:

1. Auf der smartHMI eine Statusanzeige öffnen, z. B. die Statusanzeige **POV**.
   (Nicht möglich bei den Statusanzeigen **Submit-Interpreter**, **Antriebe** und **Roboter-Interpreter**.)
   Ein Fenster öffnet sich.
2. Auf **Optionen** drücken. Das Fenster **Handverfahroptionen** öffnet sich.

Für die meisten Parameter braucht nicht eigens das Fenster **Handverfahroptionen** geöffnet werden. Sie können direkt über die Statusanzeigen der smartHMI eingestellt werden.

4.14.1.1 Registerkarte Allgemein

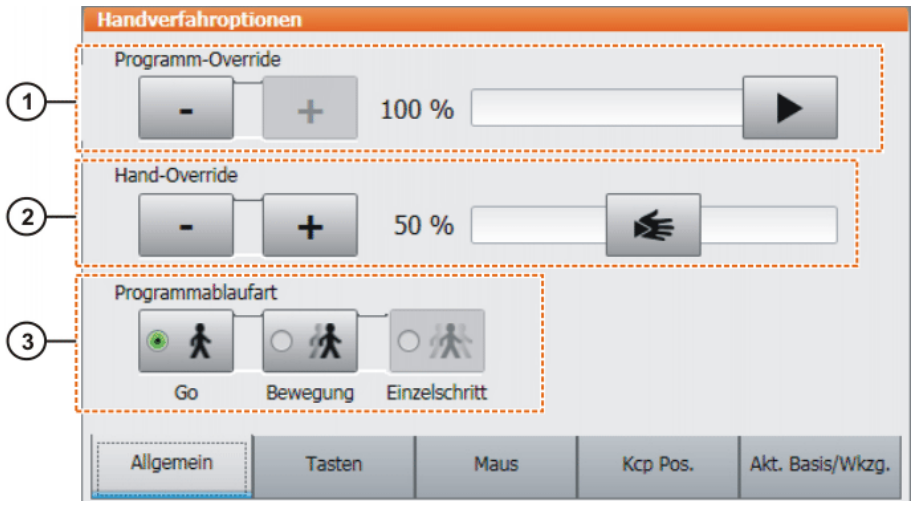


Abb. 4-14: Registerkarte Allgemein

Beschreibung

Pos.	Beschreibung
1	Programm-Override einstellen (>>> 8.5 "Programm-Override (POV) einstellen" Seite 277)
2	Hand-Override einstellen (>>> 4.14.3 "Hand-Override (HOV) einstellen" Seite 73)
3	Programmablaufart wählen (>>> 8.2 "Programmablaufarten" Seite 273)

4.14.1.2 Registerkarte Tasten

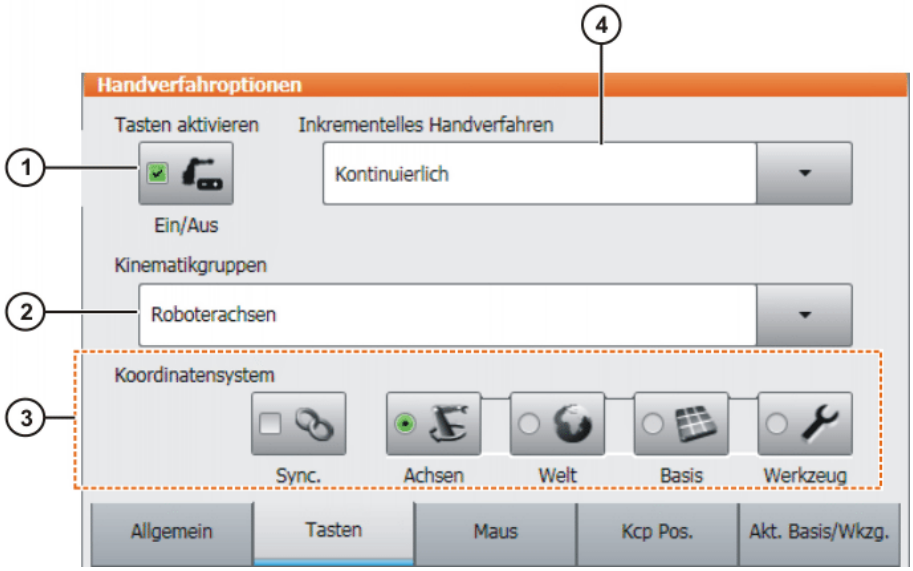


Abb. 4-15: Registerkarte Tasten

Beschreibung

Pos.	Beschreibung
1	Verfahrart "Verfahrtasten" aktivieren (>>> 4.14.2 "Verfahrart aktivieren" Seite 73)
2	Kinematikgruppe auswählen. Die Kinematikgruppe definiert, auf welche Achsen sich die Verfahrtasten beziehen. Default: **Roboterachsen** (= A1 … A6) Abhängig von der Anlagenkonfiguration können weitere Kinematikgruppen zur Verfügung stehen. (>>> 4.15 "Zusatzachsen manuell verfahren" Seite 79)
3	Das Koordinatensystem für das Verfahren mit den Verfahrtasten auswählen: ■ **Achsen**, **Welt**, **Basis** oder **Werkzeug** Checkbox **Sync.**: ■ Ohne Häkchen (Default): In den Registerkarten **Tasten** und **Maus** können verschiedene Koordinatensysteme ausgewählt sein. ■ Mit Häkchen: In den Registerkarten **Tasten** und **Maus** kann nur ein und dasselbe Koordinatensystem ausgewählt sein. Wenn das Koordinatensystem in der einen Registerkarte geändert wird, passt sich die Einstellung in der anderen automatisch daran an.
4	Inkrementelles Handverfahren (>>> 4.14.10 "Inkrementelles Handverfahren" Seite 78)

4.14.1.3 Registerkarte Maus

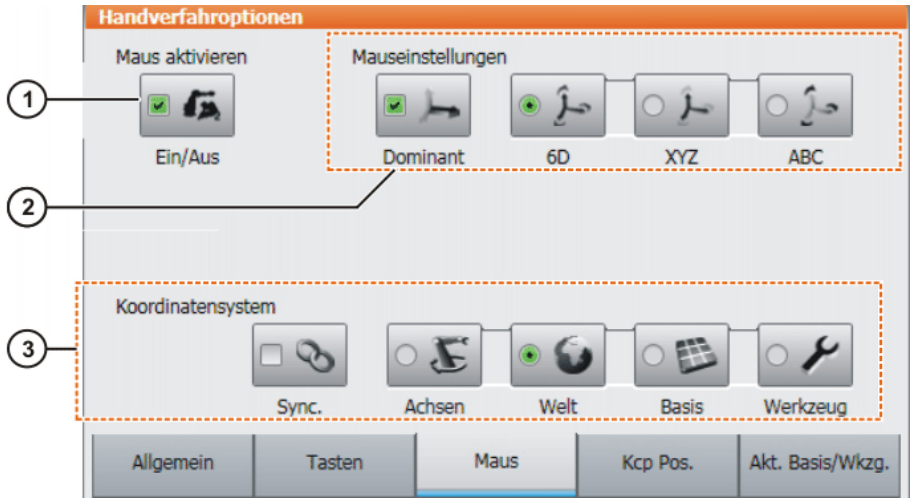


Abb. 4-16: Registerkarte Maus

Beschreibung

Pos.	Beschreibung
1	Verfahrart "Space Mouse" aktivieren (>>> 4.14.2 "Verfahrart aktivieren" Seite 73)

Pos.	Beschreibung
2	Space Mouse konfigurieren (>>> 4.14.7 "Space Mouse konfigurieren" Seite 75)
3	Das Koordinatensystem für das Verfahren mit der Space Mouse auswählen: ■ **Achsen**, **Welt**, **Basis** oder **Werkzeug** Checkbox **Sync.**: ■ Ohne Häkchen (Default): In den Registerkarten **Tasten** und **Maus** können verschiedene Koordinatensysteme ausgewählt sein. ■ Mit Häkchen: In den Registerkarten **Tasten** und **Maus** kann nur ein und dasselbe Koordinatensystem ausgewählt sein. Wenn das Koordinatensystem in der einen Registerkarte geändert wird, passt sich die Einstellung in der anderen automatisch daran an.

4.14.1.4 Registerkarte Kcp Pos.

Abb. 4-17: Registerkarte Kcp Pos.

Beschreibung

Pos.	Beschreibung
1	(>>> 4.14.8 "Ausrichtung der Space Mouse festlegen" Seite 77)

4.14.1.5 Registerkarte Akt. Basis/Wkzg.

Abb. 4-18: Registerkarte Akt. Basis/Wkzg.

Beschreibung

Pos.	Beschreibung
1	Hier wird das aktuelle Werkzeug angezeigt. Ein anderes Werkzeug kann ausgewählt werden. (>>> 4.14.4 "Werkzeug und Basis auswählen" Seite 74) Die Anzeige **Unbekannt [?]** bedeutet, dass noch kein Werkzeug vermessen wurde.
2	Hier wird die aktuelle Basis angezeigt. Eine andere Basis kann ausgewählt werden. (>>> 4.14.4 "Werkzeug und Basis auswählen" Seite 74) Die Anzeige **Unbekannt [?]** bedeutet, dass noch keine Basis vermessen wurde.
3	Interpolationsmodus auswählen: ■ **Flansch**: Das Werkzeug ist am Anbauflansch montiert. ■ **Ext. Wkzg.**: Das Werkzeug ist ein feststehendes Werkzeug.

4.14.2 Verfahrart aktivieren

Vorgehensweise

1. Fenster **Handverfahroptionen** öffnen.
   (>>> 4.14.1 "Fenster Handverfahroptionen" Seite 69)
2. Um die Verfahrart "Verfahrtasten" zu aktivieren:
   In der Registerkarte **Tasten** die Checkbox **Tasten aktivieren** aktivieren.
   Um die Verfahrart "Space Mouse" zu aktivieren:
   In der Registerkarte **Maus** die Checkbox **Maus aktivieren** aktivieren.

Beschreibung

Die beiden Verfahrarten "Verfahrtasten" und "Space Mouse" können gleichzeitig aktiviert sein. Wenn man den Roboter mit den Tasten verfährt, ist dann die Space Mouse gesperrt, bis der Roboter wieder still steht. Wenn man die Space Mouse betätigt, sind die Tasten gesperrt.

4.14.3 Hand-Override (HOV) einstellen

Beschreibung

Der Hand-Override bestimmt die Geschwindigkeit des Roboters beim manuellen Verfahren. Welche Geschwindigkeit der Roboter bei 100 % Hand-Over-

ride tatsächlich erreicht, ist abhängig von verschiedenen Faktoren, u. a. vom Robotertyp. Die Geschwindigkeit kann jedoch 250 mm/s nicht übersteigen.

Vorgehensweise

1. Die Statusanzeige **POV/HOV** berühren. Das Fenster **Overrides** öffnet sich.
2. Den gewünschten Hand-Override einstellen. Er kann entweder über die Plus-Minus-Tasten oder über den Regler eingestellt werden.
   - Plus-Minus-Tasten: Einstellung möglich in den Schritten 100%, 75%, 50%, 30%, 10%, 3%, 1%
   - Regler: Der Override kann in 1%-Schritten geändert werden.
3. Die Statusanzeige **POV/HOV** erneut berühren. (Oder den Bereich außerhalb des Fensters berühren.)

   Das Fenster schließt sich und der gewählte Override wird übernommen.

> Im Fenster **Overrides** kann über **Optionen** das Fenster **Handverfahroptionen** geöffnet werden.

Alternative Vorgehensweise

Alternativ kann der Override mit der Plus-Minus-Taste rechts am smartPAD eingestellt werden.

Einstellung möglich in den Schritten 100%, 75%, 50%, 30%, 10%, 3%, 1%.

4.14.4 Werkzeug und Basis auswählen

Beschreibung

In der Robotersteuerung können maximal 16 TOOL- und 32 BASE-Koordinatensysteme gespeichert sein. Für das kartesische Verfahren müssen ein Werkzeug (TOOL-Koordinatensystem) und eine Basis (BASE-Koordinatensystem) ausgewählt werden.

Vorgehensweise

1. Die Statusanzeige **Werkzeug/Basis** berühren. Das Fenster **Akt. Basis/Wkzg.** öffnet sich.
2. Das gewünschte Werkzeug und die gewünschte Basis auswählen.
3. Das Fenster schließt sich und die Auswahl wird übernommen.

4.14.5 Mit Verfahrtasten achsspezifisch verfahren

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1

Vorgehensweise

1. Als Koordinatensystem für die Verfahrtasten **Achsen** auswählen.
2. Hand-Override einstellen.
3. Zustimmungsschalter drücken und halten.

   Neben den Verfahrtasten werden die Achsen A1 bis A6 angezeigt.
4. Auf die Plus- oder Minus-Verfahrtaste drücken, um eine Achse in positiver oder negativer Richtung zu bewegen.

> Die Position des Roboters beim Verfahren kann eingeblendet werden: Im Hauptmenü **Anzeige** > **Istposition** wählen.

4.14.6 Mit Verfahrtasten kartesisch verfahren

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1
- Werkzeug und Basis sind ausgewählt.

  (>>> 4.14.4 "Werkzeug und Basis auswählen" Seite 74)

Vorgehensweise

1. Als Koordinatensystem für die Verfahrtasten **Welt**, **Basis** oder **Werkzeug** auswählen.
2. Hand-Override einstellen.
3. Zustimmungsschalter drücken und halten.

   Neben den Verfahrtasten werden folgende Bezeichnungen angezeigt:

   - **X**, **Y**, **Z**: für die linearen Bewegungen entlang der Achsen des gewählten Koordinatensystems
   - **A**, **B**, **C**: für die rotatorischen Bewegungen um die Achsen des gewählten Koordinatensystems
4. Auf die Plus- oder Minus-Verfahrtaste drücken, um den Roboter in positiver oder negativer Richtung zu bewegen.

Die Position des Roboters beim Verfahren kann eingeblendet werden: Im Hauptmenü **Anzeige** > **Istposition** wählen.

4.14.7 Space Mouse konfigurieren

Vorgehensweise

1. Das Fenster **Handverfahroptionen** öffnen und die Registerkarte **Maus** wählen.

   (>>> 4.14.1 "Fenster Handverfahroptionen" Seite 69)
2. Gruppe **Mauseinstellungen**:
   - Checkbox **Dominant**:

     Den Dominantmodus wie gewünscht ein- oder ausschalten.
   - Optionsfeld **6D**/**XYZ**/**ABC**:

     Auswählen, ob der TCP translatorisch, rotatorisch oder auf beide Arten bewegt werden kann.
3. Das Fenster **Handverfahroptionen** schließen.

Beschreibung

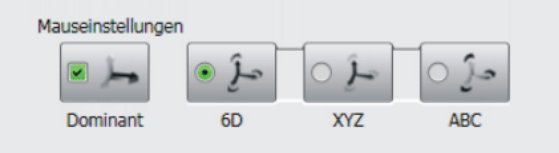


Abb. 4-19: Mauseinstellungen

Checkbox **Dominant**:

Abhängig vom Dominantmodus können mit der Space Mouse nur eine Achse oder mehrere Achsen gleichzeitig bewegt werden.

Checkbox	Beschreibung
Aktiv	Der Dominantmodus ist eingeschaltet. Nur die Achse, die über die Space Mouse die größte Auslenkung erfährt, wird verfahren.
Inaktiv	Der Dominantmodus ist ausgeschaltet. Je nach Achsauswahl können 3 oder 6 Achsen gleichzeitig bewegt werden.

Option	Beschreibung
6D	Der Roboter kann durch Ziehen, Drücken, Drehen und Kippen der Space Mouse bewegt werden. Beim kartesischen Verfahren sind folgende Bewegungen möglich: ■ Translatorische Bewegungen in X-, Y- und Z-Richtung ■ Rotatorische Bewegungen um die X-, Y- und Z-Achse
XYZ	Der Roboter kann nur durch Ziehen oder Drücken der Space Mouse bewegt werden. Beim kartesischen Verfahren sind folgende Bewegungen möglich: ■ Translatorische Bewegungen in X-, Y- und Z-Richtung
ABC	Der Roboter kann nur durch Drehen oder Kippen der Space Mouse bewegt werden. Beim kartesischen Verfahren sind folgende Bewegungen möglich: ■ Rotatorische Bewegungen um die X-, Y- und Z-Achse

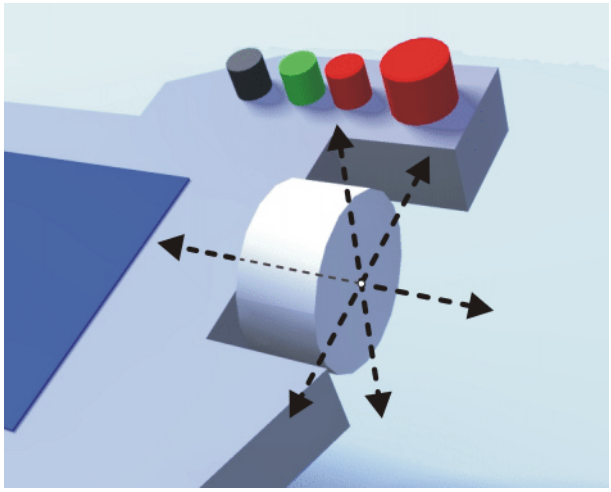


Abb. 4-20: Space Mouse ziehen und drücken

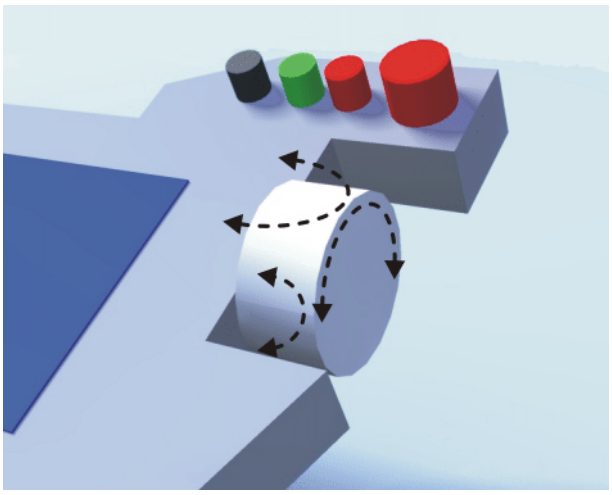


Abb. 4-21: Space Mouse drehen und kippen

4.14.8 Ausrichtung der Space Mouse festlegen

Beschreibung

Die Funktion der Space Mouse kann an den Standort des Benutzers angepasst werden, damit die Verfahrrichtung des TCP der Auslenkung der Space Mouse entspricht.

Der Standort des Benutzers wird in Grad angegeben. Die Bezugspunkt für die Gradangabe ist der Anschlusskasten am Grundgestell. Die Position des Roboterarms oder der Achsen ist unrelevant.

Default-Einstellung: 0°. Dies entspricht einem Benutzer, der gegenüber vom Anschlusskasten steht.

Beim Umschalten in die Betriebsart Automatik Extern wird die Ausrichtung der Space Mouse automatisch auf 0° zurückgesetzt.

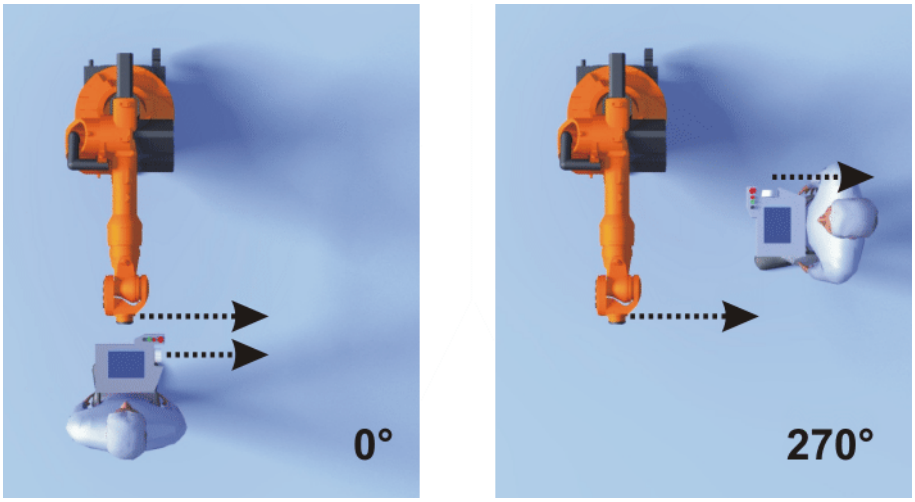


Abb. 4-22: Space Mouse: 0° und 270°

Voraussetzung

- Betriebsart T1

Vorgehensweise

1. Das Fenster **Handverfahroptionen** öffnen und die Registerkarte **Kcp Pos.** wählen.

Abb. 4-23: Ausrichtung der Space Mouse festlegen

2. Das smartPAD auf die Position ziehen, die dem Standort des Benutzers entspricht. (Schritteinteilung = 45°)
3. Das Fenster **Handverfahroptionen** schließen.

4.14.9 Mit Space Mouse kartesisch verfahren

Voraussetzung

- Die Verfahrart "Space Mouse" ist aktiv.
- Betriebsart T1
- Werkzeug und Basis sind ausgewählt.
  (>>> 4.14.4 "Werkzeug und Basis auswählen" Seite 74)
- Die Space Mouse ist konfiguriert.
  (>>> 4.14.7 "Space Mouse konfigurieren" Seite 75)
- Die Ausrichtung der Space Mouse ist festgelegt.
  (>>> 4.14.8 "Ausrichtung der Space Mouse festlegen" Seite 77)

Vorgehensweise

1. Als Koordinatensystem für die Space Mouse **Welt**, **Basis** oder **Werkzeug** auswählen.
2. Hand-Override einstellen.
3. Zustimmungsschalter drücken und halten.
4. Roboter mit der Space Mouse in die gewünschte Richtung bewegen.

> Die Position des Roboters beim Verfahren kann eingeblendet werden: Im Hauptmenü **Anzeige** > **Istposition** wählen.

4.14.10 Inkrementelles Handverfahren

Beschreibung

Das inkrementelle Handverfahren ermöglicht es, den Roboter um eine definierte Distanz zu bewegen, z. B. um 10 mm oder 3°. Danach stoppt der Roboter selbständig.

Das inkrementelle Handverfahren kann beim Verfahren mit den Verfahrtasten zugeschaltet werden. Beim Verfahren mit der Space Mouse ist das inkrementelle Handverfahren nicht möglich.

Anwendungsbereiche:

- Positionieren von Punkten in gleichen Abständen
- Herausfahren aus einer Position um eine definierte Distanz, z. B. im Fehlerfall
- Justage mit der Messuhr

Folgende Optionen stehen zur Verfügung:

Einstellung	Beschreibung
Kontinuierlich	Das inkrementelle Handverfahren ist ausgeschaltet.
100mm / 10°	1 Inkrement = 100 mm oder 10°
10mm / 3°	1 Inkrement = 10 mm oder 3°
1mm / 1°	1 Inkrement = 1 mm oder 1°
0,1mm / 0,005°	1 Inkrement = 0,1 mm oder 0,005°

Inkremente in mm:

- Gültig beim kartesischen Verfahren in X-, Y- oder Z-Richtung.

Inkremente in Grad:

- Gültig beim kartesischen Verfahren in A-, B- oder C-Richtung.
- Gültig beim achsspezifischen Verfahren.

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1

Vorgehensweise

1. In der Statusleiste die Inkrementgröße auswählen.
2. Den Roboter mit den Verfahrtasten verfahren. Er kann kartesisch oder achsspezifisch verfahren werden.

   Wenn das eingestellte Inkrement erreicht ist, stoppt der Roboter.

Wenn die Roboterbewegung unterbrochen wird, z. B. durch Loslassen des Zustimmungsschalters, wird bei der nächsten Bewegung das unterbrochene Inkrement nicht fortgesetzt, sondern ein neues Inkrement begonnen.

4.15 Zusatzachsen manuell verfahren

Zusatzachsen können nicht mit der Space Mouse verfahren werden. Wenn die Verfahrart "Space Mouse" ausgewählt ist, kann nur der Roboter mit der Space Mouse verfahren werden. Die Zusatzachsen müssen dagegen mit den Verfahrtasten verfahren werden.

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1

Vorgehensweise

1. Im Fenster **Handverfahroptionen** in der Registerkarte **Tasten** die gewünschte Kinematikgruppe auswählen, z. B. **Zusatzachsen**.

   Die Art und Anzahl der zur Verfügung stehenden Kinematikgruppen ist abhängig von der Anlagenkonfiguration.
2. Hand-Override einstellen.
3. Zustimmungsschalter drücken und halten.

   Neben den Verfahrtasten werden die Achsen der gewählten Kinematikgruppe angezeigt.
4. Auf die Plus- oder Minus-Verfahrtaste drücken, um eine Achse in positiver oder negativer Richtung zu bewegen.

Beschreibung

Abhängig von der Anlagenkonfiguration können folgende Kinematikgruppen zur Verfügung stehen:

Kinematikgruppe	Beschreibung
Roboterachsen	Mit den Verfahrtasten können die Roboterachsen verfahren werden. Die Zusatzachsen können nicht verfahren werden.
Zusatzachsen	Mit den Verfahrtasten können alle konfigurierten Zusatzachsen verfahren werden, z. B. die Zusatzachsen E1 … E5.
NAME / **Externe Kinematikgruppe** *n*	Mit den Verfahrtasten können die Achsen einer externen Kinematikgruppe verfahren werden. Der Name wird übernommen aus der Systemvariablen $ET*n*_*NAME* (*n* = Nummer der externen Kinematik). Wenn $ET*n*_*NAME* leer ist, wird als Default-Name **Externe Kinematikgruppe** *n* angezeigt.
[Benutzerdefinierte Kinematikgruppe]	Mit den Verfahrtasten können die Achsen einer benutzerdefinierten Kinematikgruppe verfahren werden. Der Name entspricht dem Namen der benutzerdefinierten Kinematikgruppe.

4.16 Arbeitsraumüberwachung überbrücken

Beschreibung

Für einen Roboter können Arbeitsräume konfiguriert sein. Arbeitsräume dienen dem Anlagenschutz.

Es gibt 2 Arten von Arbeitsräumen:

- Der Arbeitsraum ist ein nicht erlaubter Bereich.
  Der Roboter darf sich nur außerhalb des Arbeitsraums bewegen.
- Nur der Arbeitsraum ist ein erlaubter Bereich.
  Der Roboter darf sich nicht außerhalb des Arbeitsraums bewegen.

Welche Reaktionen auftreten, wenn der Roboter einen Arbeitsraum verletzt, ist abhängig von der Konfiguration. (>>> 6.11 "Arbeitsräume konfigurieren" Seite 177)

Die Reaktion kann beispielsweise sein, dass der Roboter stoppt und eine Meldung ausgegeben wird. In diesem Fall muss die Arbeitsraumüberwachung überbrückt werden. Dann kann der Roboter wieder aus dem nicht erlaubten Raum herausgefahren werden.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Arbeitsraumüberwachung** > **Überbrücken** wählen.
2. Den Roboter manuell aus dem nicht erlaubten Raum herausfahren.
   Wenn der Roboter den nicht erlaubten Raum verlassen hat, ist die Arbeitsraumüberwachung automatisch wieder aktiv.

4.17 Anzeigefunktionen

4.17.1 Energieverbrauch messen und anzeigen

Beschreibung

Der Gesamt-Energieverbrauch von Roboter und Robotersteuerung kann auf der smartHMI angezeigt werden. Voraussetzung ist, dass die Verbrauchsmessung für den verwendeten Robotertyp möglich ist.

Die Verbräuche optionaler Komponenten der Robotersteuerung (z. B. US1, US2 usw.) und von anderen Steuerungen werden nicht betrachtet. Angezeigt wird immer der Verbrauch der letzten 60 min seit dem letzten Kaltstart. Darüber hinaus hat der Benutzer die Möglichkeit, selber Messungen zu starten und zu stoppen.

Die Verbrauchswerte können getraced werden. Hierfür steht die vordefinierte Konfiguration Tracedef_KRC_EnergyCalc zur Verfügung.

Außerdem können die Daten mit PROFIenergy an eine übergeordnete Steuerung übertragen werden. PROFIenergy ist Bestandteil von KR C4 PROFINET.

Es gibt 2 Möglichkeiten, um Messungen zu starten und zu stoppen:

- Im Fenster **Energieverbrauch** (>>> Abb. 4-24)
- Über KRL

Voraussetzung

- Die Verbrauchsmessung ist für den verwendeten Robotertyp möglich.
  Wenn nicht, sind die Felder im Fenster **Energieverbrauch** gegraut.

Vorgehensweise

Eine Messung im Fenster **Energieverbrauch** starten und stoppen:

1. Im Hauptmenü **Anzeige** > **Energieverbrauch** wählen. Das Fenster **Energieverbrauch** öffnet sich.
2. Bei Bedarf das Häkchen bei **Aktualisieren** setzen.
3. Auf **Messung starten** drücken. Rechts neben der obersten Zeile zeigt jetzt ein roter Punkt an, dass eine Messung läuft.
4. Um die Messung zu stoppen, auf **Messung stoppen** drücken. Das Ergebnis wird angezeigt.

Eine Messung über KRL starten und stoppen:

1. Die Messung über $ENERGY_MEASURING.ACTIVE = TRUE starten (über KRL-Programm oder Variablenkorrektur möglich). Die Messung startet.
2. Im Hauptmenü **Anzeige** > **Energieverbrauch** wählen. Das Fenster **Energieverbrauch** öffnet sich. Rechts neben der obersten Zeile wird die laufende Messung durch einen roten Punkt angezeigt.
3. Bei Bedarf das Häkchen bei **Aktualisieren** setzen.
4. Die Messung über $ENERGY_MEASURING.ACTIVE = FALSE stoppen.

Das Fenster **Energieverbrauch** kann auch unabhängig von der Messung geöffnet werden. Die oberste Zeile zeigt immer das Ergebnis der laufenden oder der letzten Messung an.

Eigenschaften Messung

- Eine gestartete Messung läuft, bis sie gestoppt wird. Dies ist unabhängig davon, ob das Fenster **Energieverbrauch** geöffnet oder geschlossen ist.
- Eine Messung, die über KRL gestartet wurden, kann sowohl über KRL als auch über **Messung stoppen** gestoppt werden.
- Eine Messung, die über **Messung starten** gestartet wurde, kann nur über **Messung stoppen** gestoppt werden, so lange das Fenster **Energieverbrauch** geöffnet bleibt. Wenn versucht wird, die Messung über KRL zu stoppen, zeigt die Robotersteuerung folgende Meldung an: *Die Energiemessung kann aktuell nicht gestoppt werden*.

  Wenn das Fenster **Energieverbrauch** wieder geschlossen wird, kann die Messung auch über KRL gestoppt werden. Dies verhindert, dass eine im Fenster **Energieverbrauch** gestartete Messung dauerhaft Messungen über KRL blockiert.
- Eine Messung zu starten, während bereits eine Messung läuft, ist nicht möglich. Die Robotersteuerung zeigt dann folgende Meldung an: *Eine*

Energiemessung ist schon aktiv. Die laufende Messung muss erst gestoppt werden.

Fenster Energieverbrauch

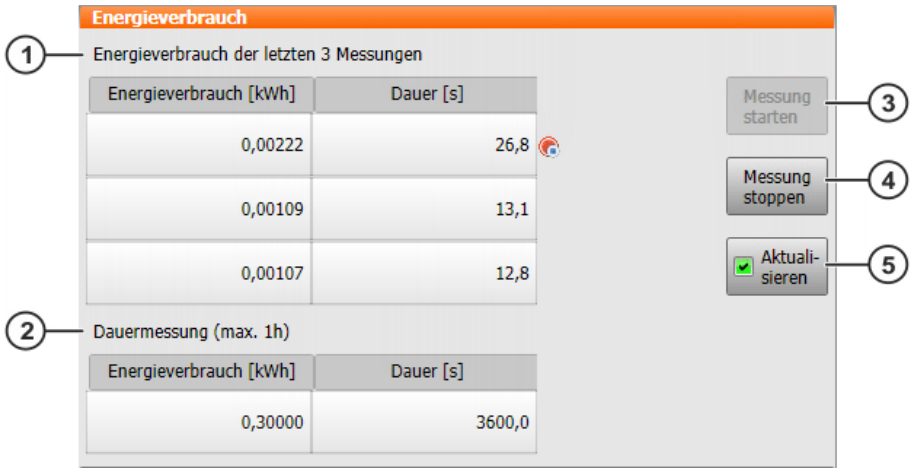


Abb. 4-24: Fenster Energieverbrauch

Pos.	Beschreibung
1	Ergebnisse der Messungen, die der Benutzer gestartet hat Es werden die letzten 3 Ergebnisse angezeigt. Das jüngste Ergebnis wird in der obersten Zeile angezeigt. Wenn gerade eine Messung läuft, wird dies durch einen roten Punkt rechts neben der Zeile angezeigt.
2	Energieverbrauch in den letzten 60 min seit dem letzten Kaltstart
3	Startet eine Messung. **Messung starten** steht nicht zur Verfügung, wenn gerade eine Messung läuft.
4	Stoppt eine laufende Messung. Wie die Messung gestartet wurde, ob über **Messung starten** oder über KRL, spielt keine Rolle.
5	■ Mit Häkchen: Während eine Messung läuft, wird die Ergebnisanzeige laufend aktualisiert. ■ Ohne Häkchen: Während eine Messung läuft, bleibt in der Anzeige der zuletzt aktualisierte Wert stehen. Erst nach dem Stoppen der Messung wird das Ergebnis angezeigt.

4.17.2 Istposition anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Istposition** wählen. Die kartesische Istposition wird angezeigt.
2. Um die achsspezifische Istposition anzuzeigen, auf **Achsspezifisch** drücken.
3. Um wieder die kartesische Istposition anzuzeigen, auf **Kartesisch** drücken.

Beschreibung

Istposition kartesisch:

Die aktuelle Position (X, Y, Z) und Orientierung (A, B, C) des TCP werden angezeigt. Außerdem werden Status und Turn angezeigt.

Istposition achsspezifisch:

Die aktuelle Position der Achsen A1 bis A6 wird angezeigt. Wenn Zusatzachsen vorhanden sind, wird auch die Position der Zusatzachsen angezeigt.

Die Istposition kann auch angezeigt werden, während der Roboter verfährt.

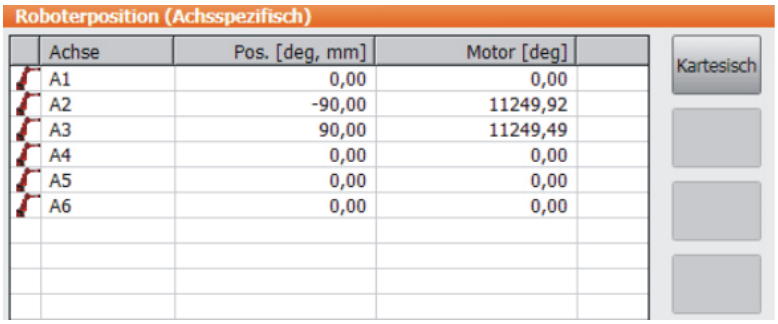


Abb. 4-25: Istposition achsspezifisch

4.17.3 Digitale Ein-/Ausgänge anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Ein-/Ausgänge** > **Digitale E/A** wählen.
2. Um einen bestimmten Ein-/Ausgang anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Ein-/Ausgang mit dieser Nummer.

Beschreibung

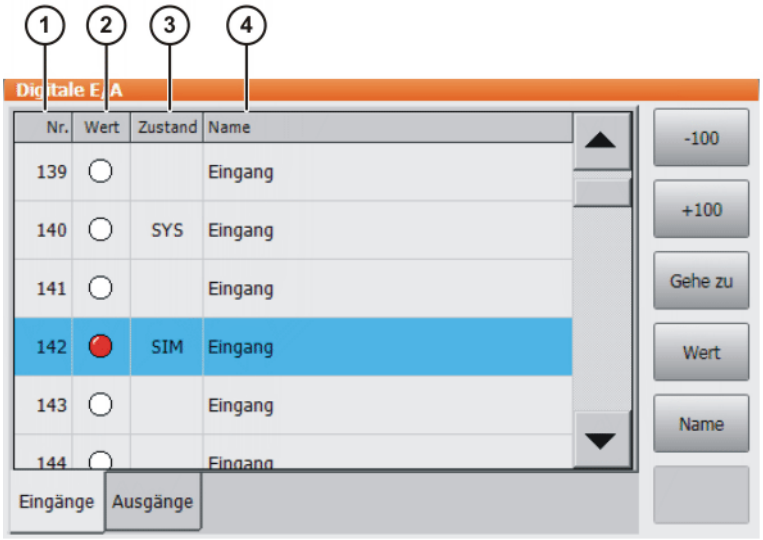


Abb. 4-26: Digitale Eingänge

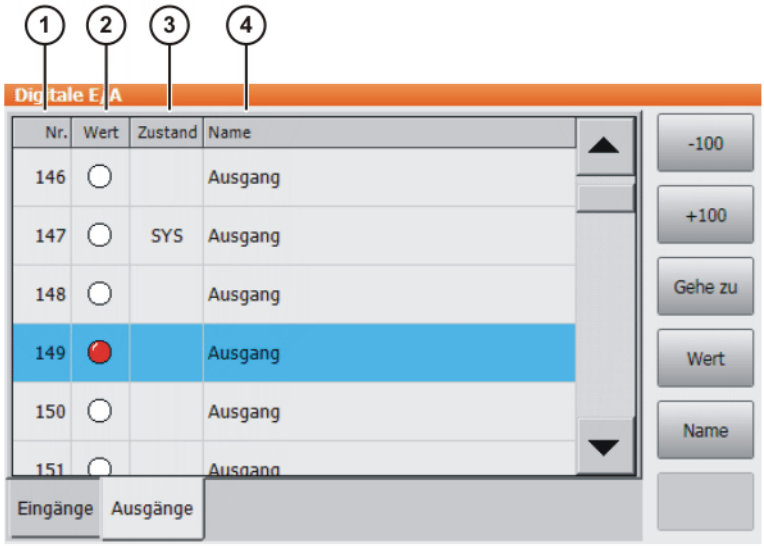


Abb. 4-27: Digitale Ausgänge

Pos.	Beschreibung
1	Nummer des Ein-/Ausgangs
2	Wert des Ein-/Ausgangs. Wenn ein Ein- oder Ausgang TRUE ist, ist er rot markiert.
3	Eintrag SIM: Der Ein-/Ausgang ist simuliert. Eintrag SYS: Der Wert des Ein-/Ausgangs ist einer Systemvariablen gespeichert. Dieser Ein-/Ausgang ist schreibgeschützt.
4	Name des Ein-/Ausgangs

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
-100	Schaltet in der Anzeige 100 Ein- oder Ausgänge zurück.
+100	Schaltet in der Anzeige 100 Ein- oder Ausgänge weiter.
Gehe zu	Die Nummer des gesuchten Ein- oder Ausgangs kann eingegeben werden.
Wert	Schaltet den markierten Ein- oder Ausgang zwischen TRUE und FALSE um. Voraussetzung: Der Zustimmungsschalter ist gedrückt. In der Betriebsart AUT EXT steht diese Schaltfläche nicht zur Verfügung und für Eingänge nur dann, wenn die Simulation eingeschaltet ist.
Name	Der Name des markierten Ein- oder Ausgangs kann geändert werden.

4.17.4 Analoge Ein-/Ausgänge anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Ein-/Ausgänge** > **Analoge E/A** wählen.
2. Um einen bestimmten Ein-/Ausgang anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Ein-/Ausgang mit dieser Nummer.

Folgende Schaltflächen stehen zur Verfügung:

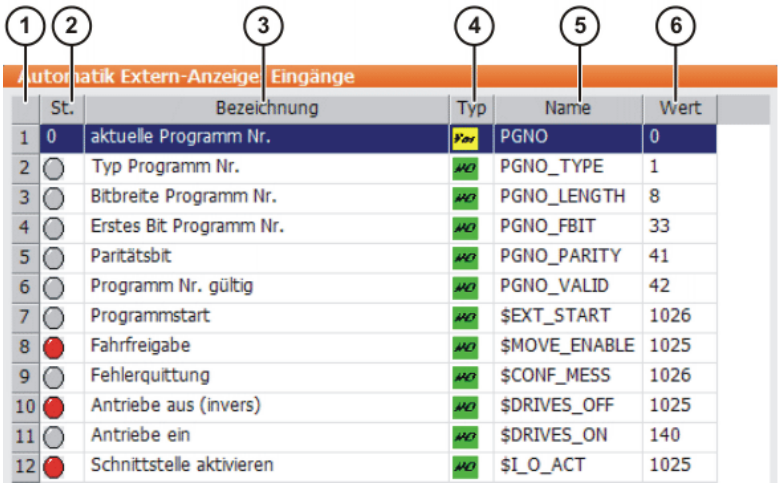
Schaltfläche	Beschreibung
Gehe zu	Die Nummer des gesuchten Ein- oder Ausgangs kann eingegeben werden.
Spannung	Für den markierten Ausgang kann eine Spannung eingegeben werden. ■ **-10 … 10 V** Diese Schaltfläche steht nur für Ausgänge zur Verfügung.
Name	Der Name des markierten Ein-/Ausgangs kann geändert werden.

4.17.5 Ein-/Ausgänge für Automatik Extern anzeigen

Vorgehensweise

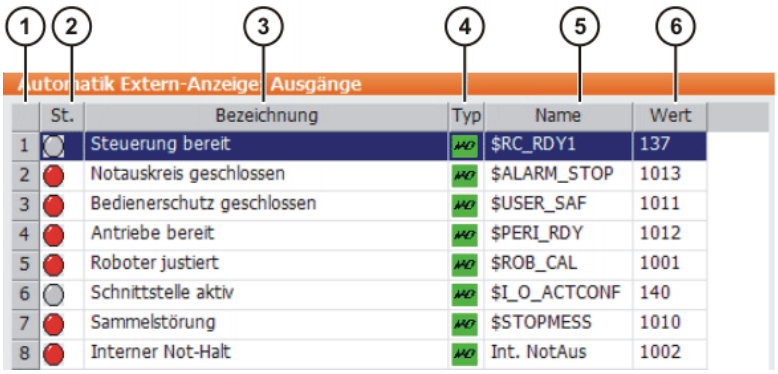
- Im Hauptmenü **Anzeige** > **Ein-/Ausgänge** > **Automatik Extern** wählen.

Beschreibung



	St.	Bezeichnung	Typ	Name	Wert
1	0	aktuelle Programm Nr.		PGNO	0
2		Typ Programm Nr.		PGNO_TYPE	1
3		Bitbreite Programm Nr.		PGNO_LENGTH	8
4		Erstes Bit Programm Nr.		PGNO_FBIT	33
5		Paritätsbit		PGNO_PARITY	41
6		Programm Nr. gültig		PGNO_VALID	42
7		Programmstart		$EXT_START	1026
8		Fahrfreigabe		$MOVE_ENABLE	1025
9		Fehlerquittung		$CONF_MESS	1026
10		Antriebe aus (invers)		$DRIVES_OFF	1025
11		Antriebe ein		$DRIVES_ON	140
12		Schnittstelle aktivieren		$I_O_ACT	1025

Abb. 4-28: Eingänge Automatik Extern (Detailanzeige)



	St.	Bezeichnung	Typ	Name	Wert
1		Steuerung bereit		$RC_RDY1	137
2		Notauskreis geschlossen		$ALARM_STOP	1013
3		Bedienerschutz geschlossen		$USER_SAF	1011
4		Antriebe bereit		$PERI_RDY	1012
5		Roboter justiert		$ROB_CAL	1001
6		Schnittstelle aktiv		$I_O_ACTCONF	140
7		Sammelstörung		$STOPMESS	1010
8		Interner Not-Halt		Int. NotAus	1002

Abb. 4-29: Ausgänge Automatik Extern (Detailanzeige)

Pos.	Beschreibung
1	Nummer
2	Status ■ **Grau**: Inaktiv (FALSE) ■ **Rot**: Aktiv (TRUE)
3	Langtext-Name des Ein-/Ausgangs

Pos.	Beschreibung
4	Typ ■ **Grün**: Ein-/Ausgang ■ **Gelb**: Variable oder Systemvariable ($...)
5	Name des Signals oder der Variable
6	Ein-/Ausgangsnummer oder Kanalnummer

Die Spalten 4, 5 und 6 werden nur angezeigt, wenn **Details** gedrückt wurde.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Konfig.	Schaltet zur Konfiguration für Automatik Extern um. (>>> 6.17.2 "Automatik Extern Ein-/Ausgänge konfigurieren" Seite 198)
Eingänge/Ausgänge	Schaltet zwischen den Fenstern für Eingänge und Ausgänge um.
Details/Normal	Schaltet zwischen den Ansichten **Details** und **Normal** um.

4.17.6 Wert einer Variablen anzeigen und ändern

Diese Funktionalität wird auch "Variablenkorrektur" genannt.

Voraussetzung

Um eine Variable zu ändern:

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Einzeln** wählen.
   Das Fenster **Variablenanzeige – Einzeln** öffnet sich.
2. Im Feld **Name** den Namen der Variablen eingeben und mit der Eingabe-Taste bestätigen.
3. Wenn ein Programm angewählt ist, ist im Feld **Modul** automatisch das Programm eingetragen.
   Wenn eine Variable aus einem anderen Programm angezeigt werden soll, das Programm folgendermaßen eingeben:
   /R1/*Programmname*
   - Zwischen /R1/ und dem Programmnamen keine Ordner angeben. Beim Programmnamen keine Dateiendung angeben.
   - Bei Systemvariablen braucht im Feld **Modul** kein Programm angegeben werden.
4. Im Feld **Aktueller Wert** wird der aktuelle Wert der Variablen angezeigt. Wenn nichts angezeigt wird, dann ist der Variablen noch kein Wert zugewiesen worden.
   Um die Variable zu ändern:
5. Im Feld **Neuer Wert** den gewünschten Wert eingeben.
6. Die Schaltfläche **Wert setzen** drücken. Im Feld **Aktueller Wert** wird der neue Wert angezeigt.

Beschreibung

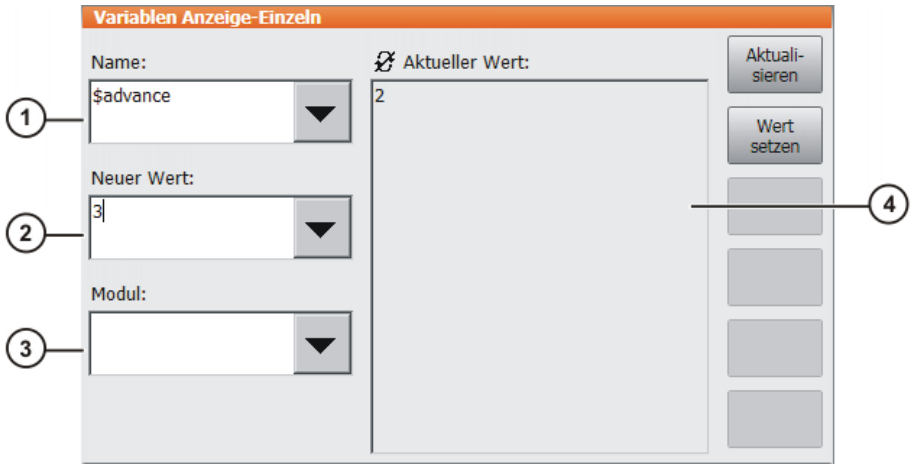


Abb. 4-30: Fenster Variablen Anzeige-Einzeln

Pos.	Beschreibung
1	Name der Variablen, die geändert werden soll
2	Neuer Wert, der der Variablen zugewiesen werden soll
3	Programm, in dem nach der Variablen gesucht wird Bei Systemvariablen ist das Feld **Modul** unrelevant.
4	Dieses Feld hat zwei Zustände: ■ : Der angezeigte Wert wird nicht automatisch aktualisiert. ■ : Der angezeigte Wert wird automatisch aktualisiert. Zwischen den Zuständen wechseln: ■ **Aktualisieren** drücken.

4.17.7 Zustand einer Variablen anzeigen

Beschreibung

Variablen können folgende Zustände haben:

- UNKNOWN: Die Variable ist unbekannt.
- DECLARED: Die Variable ist deklariert.
- INITIALIZED: Die Variable ist initialisiert.

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Einzeln** wählen.
   Das Fenster **Variablen Anzeige-Einzeln** öffnet sich.
2. Im Feld **Name** eingeben: =VARSTATE("*name*")
   name = Name der Variablen, deren Zustand man anzeigen möchte
3. Wenn ein Programm angewählt ist, ist im Feld **Modul** automatisch das Programm eingetragen.
   Wenn eine Variable aus einem anderen Programm angezeigt werden soll, das Programm folgendermaßen eingeben:
   /R1/*Programmname*
   - Zwischen /R1/ und dem Programmnamen keine Ordner angeben. Beim Programmnamen keine Dateiendung angeben.
   - Bei Systemvariablen braucht im Feld **Modul** kein Programm angegeben werden.
4. **Aktual.** drücken.
   Im Feld **Aktueller Wert** wird der aktuelle Zustand der Variablen angezeigt.

4.17.8 Variablenübersicht anzeigen und Variable ändern

In der Variablenübersicht werden Variablen in Gruppen angezeigt. Die Variablen können geändert werden.

Die Anzahl der Gruppen und welche Variablen sie enthalten, definiert man in der Konfiguration. Defaultmäßig ist die Variablenübersicht leer.

(>>> 6.8 "Variablenübersicht konfigurieren" Seite 175)

In der Benutzergruppe Anwender können die Variablen nur angezeigt und geändert werden, wenn diese Funktionen in der Konfiguration freigeschaltet sind.

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Übersicht** > **Anzeigen** wählen.
   Das Fenster **Variablenübersicht – Anzeige** öffnet sich.
2. Die gewünschte Gruppe auswählen.
3. Die Zelle markieren, die geändert werden soll. Änderung über Schaltfläche vornehmen.
4. **OK** drücken, um die Änderung zu speichern und das Fenster zu schließen.

Beschreibung

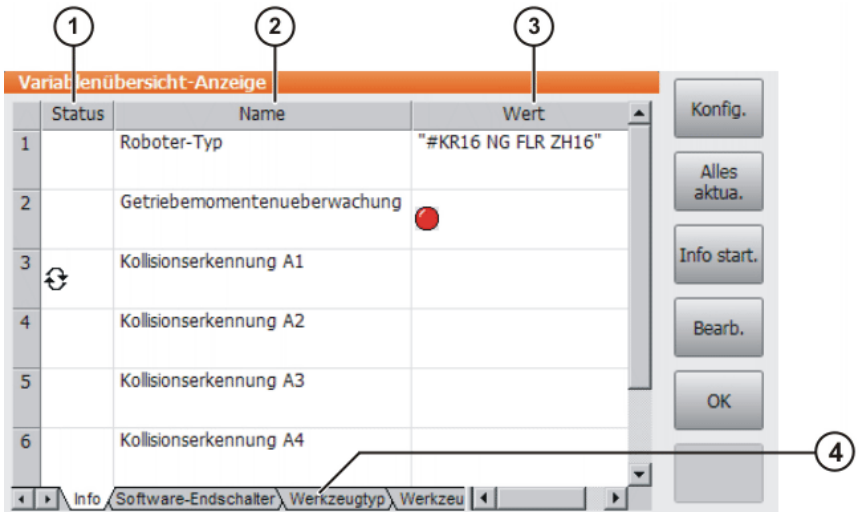


Abb. 4-31: Fenster Variablenübersicht Anzeige

Pos.	Beschreibung
1	Pfeilsymbol: Wenn sich der Wert der Variablen ändert, wird die Anzeige automatisch aktualisiert. Kein Pfeilsymbol: Die Anzeige wird nicht automatisch aktualisiert.
2	Beschreibender Name
3	Wert der Variablen. Bei Ein-/Ausgängen ist der Status angegeben: ■ **Grau**: Inaktiv (FALSE) ■ **Rot**: Aktiv (TRUE)
4	Pro Gruppe ist eine Registerkarte vorhanden.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Konfig.	Schaltet zur Konfiguration der Variablenübersicht um. (>>> 6.8 "Variablenübersicht konfigurieren" Seite 175) Diese Schaltfläche steht in der Benutzergruppe Anwender nicht zur Verfügung.
Alles aktua.	Aktualisiert die Anzeige.
Info abbr.	Schaltet die automatische Aktualisierung aus.
Info start.	Schaltet die automatische Aktualisierung ein. Pro Gruppe können bis zu 12 Variablen automatisch aktualisiert werden.
Bearb.	Schaltet die aktuelle Zelle in den Editiermodus, so dass Name oder Wert geändert werden können. In der Spalte **Wert** ändert diese Schaltfläche bei Ein-/Ausgängen den Status (TRUE/FALSE). Diese Schaltfläche steht in der Benutzergruppe Anwender nur zur Verfügung, wenn sie in der Konfiguration freigeschaltet wurde. **Hinweis:** Die Werte schreibgeschützter Variablen können nicht geändert werden.

4.17.9 Zyklische Flags anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Zyklische Flags** wählen. Das Fenster **Zyklische Flags** öffnet sich.
2. Um ein bestimmtes Flag anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Flag mit dieser Nummer.

Beschreibung

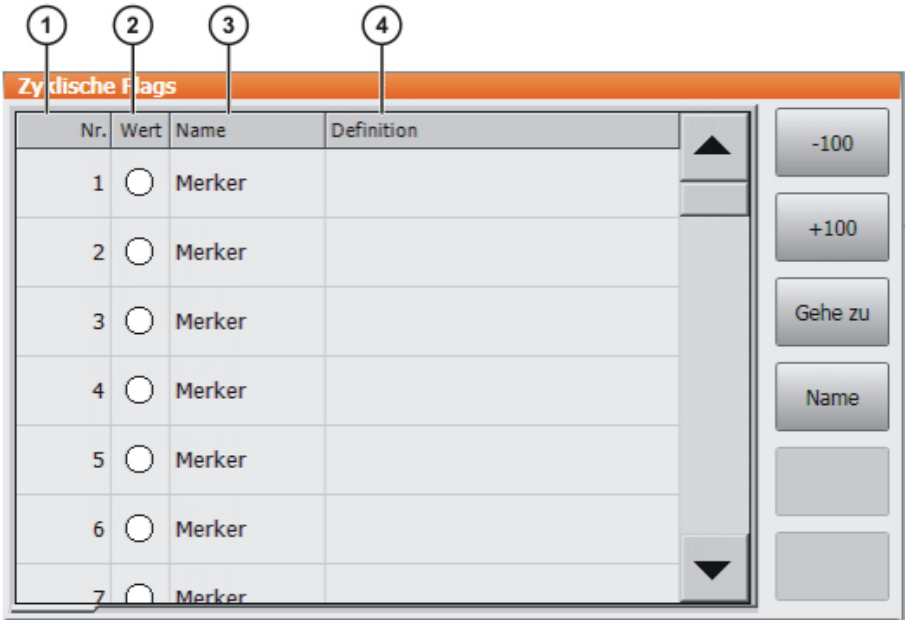


Abb. 4-32: Zyklische Flags

Pos.	Beschreibung
1	Nummer des Flags
2	Wert des Flags. Wenn ein Flag gesetzt ist, ist es rot markiert.
3	Name des Flags
4	Hier wird angezeigt, mit welchen Bedingungen das Setzen eines zyklischen Flags verknüpft ist.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
-100	Schaltet in der Anzeige 100 Flags zurück.
+100	Schaltet in der Anzeige 100 Flags weiter.
Gehe zu	Die Nummer des gesuchten Flags kann eingegeben werden.
Name	Der Name des markierten Flags kann geändert werden.

4.17.10 Flags anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Flags** wählen. Das Fenster **Flags** öffnet sich.
2. Um ein bestimmtes Flag anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Flag mit dieser Nummer.

Beschreibung

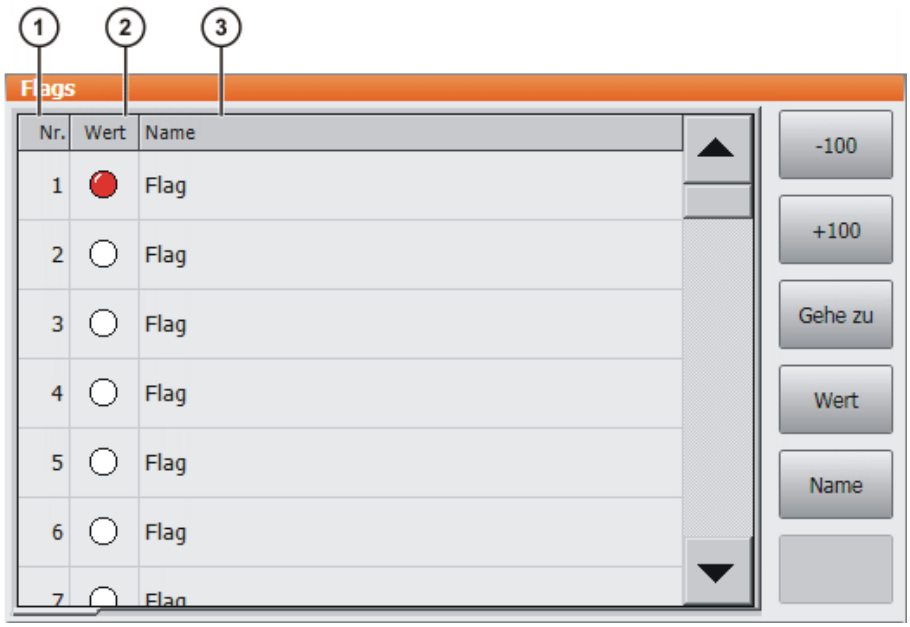


Abb. 4-33: Flags

Pos.	Beschreibung
1	Nummer des Flags
2	Wert des Flags. Wenn ein Flag gesetzt ist, ist es rot markiert.
3	Name des Flags

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
-100	Schaltet in der Anzeige 100 Flags zurück.
+100	Schaltet in der Anzeige 100 Flags weiter.
Gehe zu	Die Nummer des gesuchten Flags kann eingegeben werden.
Wert	Schaltet das markierte Flag zwischen TRUE und FALSE um. Voraussetzung: Der Zustimmungsschalter ist gedrückt. In der Betriebsart AUT EXT steht diese Schaltfläche nicht zur Verfügung.
Name	Der Name des markierten Flags kann geändert werden.

4.17.11 Zähler anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Zähler** wählen. Das Fenster **Zähler** öffnet sich.
2. Um einen bestimmten Zähler anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Zähler mit dieser Nummer.

Beschreibung

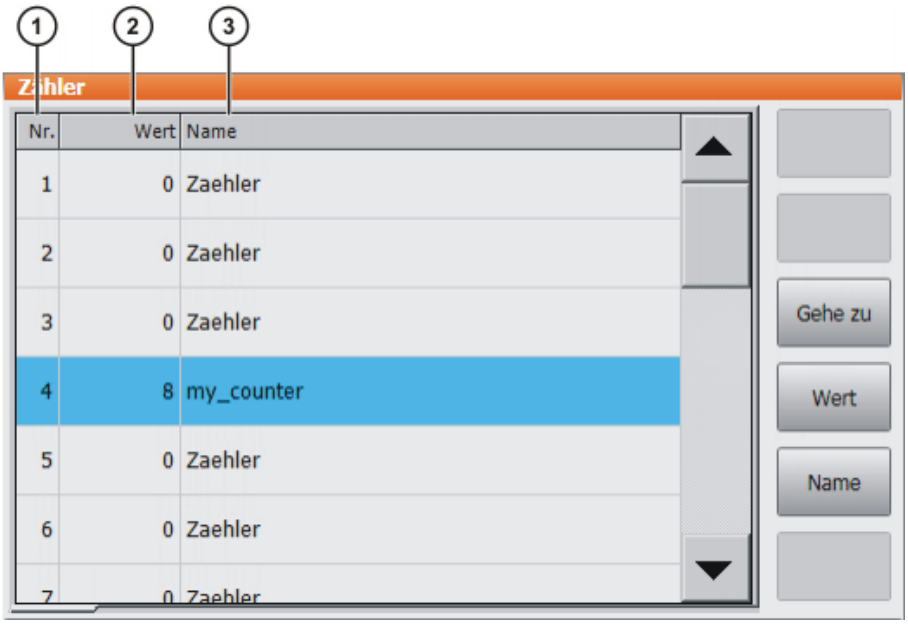


Abb. 4-34: Zähler

Pos.	Beschreibung
1	Nummer des Zählers
4	Wert des Zählers
5	Name des Zählers

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Gehe zu	Die Nummer des gesuchten Zählers kann eingegeben werden.
Wert	Für den markierten Zähler kann ein Wert eingegeben werden.
Name	Der Name des markierten Zählers kann geändert werden.

4.17.12 Timer anzeigen

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Timer** wählen. Das Fenster **Timer** öffnet sich.
2. Um einen bestimmten Timer anzuzeigen:
   - Auf die Schaltfläche **Gehe zu** drücken. Das Feld **Gehe zu:** wird angezeigt.
   - Die Nummer eingeben und mit der Eingabe-Taste bestätigen.

   Die Anzeige springt zu dem Timer mit dieser Nummer.

Beschreibung

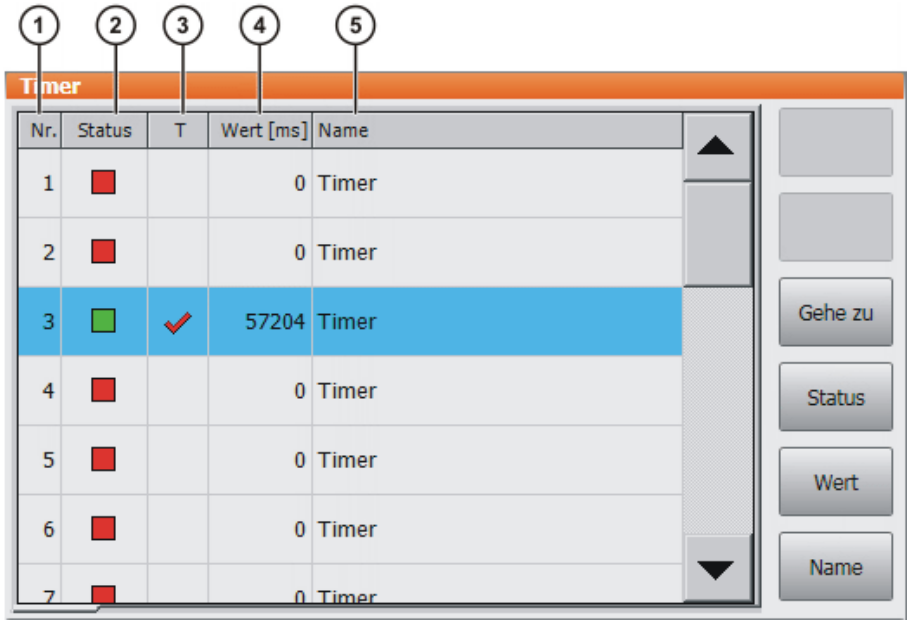


Abb. 4-35: Timer

Pos.	Beschreibung
1	Nummer des Timers
2	Status des Timers ■ Wenn der Timer aktiviert ist, ist er grün markiert. ■ Wenn der Timer deaktiviert ist, ist er rot markiert.
3	Zustand des Timers ■ Wenn der Wert des Timers > 0 ist, wird das Timer-Flag gesetzt (roter Haken). ■ Wenn der Wert des Timers ≤ 0 ist, wird kein Timer-Flag gesetzt.
4	Wert des Timers (Einheit: ms)
5	Name des Timers

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Gehe zu	Die Nummer des gesuchten Timers kann eingegeben werden.
Status	Schaltet den markierten Timer zwischen TRUE und FALSE um. Voraussetzung: Der Zustimmungsschalter ist gedrückt.
Wert	Für den markierten Timer kann ein Wert eingegeben werden.
Name	Der Name des markierten Timers kann geändert werden.

4.17.13 Vermessungsdaten anzeigen

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Messpunkte** wählen und den gewünschten Menüpunkt auswählen:
   - **Werkzeugtyp**
   - **Basistyp**
   - **Externe Achse**
2. Nummer des Werkzeugs, der Basis oder der externen Kinematik eingeben.

   Die Vermessungsmethode und die Vermessungsdaten werden angezeigt.

4.17.14 Infos zu Roboter und Robotersteuerung anzeigen

Vorgehensweise

- Im Hauptmenü **Hilfe** > **Info** wählen.

Beschreibung

Die Informationen werden beispielsweise für Anfragen beim KUKA Customer Support benötigt.

Die Registerkarten enthalten folgende Informationen:

Registerkarte	Beschreibung
Info	■ Typ der Robotersteuerung ■ Version der Robotersteuerung ■ Version der Bedienoberfläche ■ Version des Grundsystems
Roboter	■ Robotername ■ Typ und Konfiguration des Roboters ■ Betriebsdauer Der Betriebsstundenzähler läuft, wenn die Antriebe eingeschaltet sind. Alternativ kann die Betriebsdauer über die Variable $ROBRUNTIME angezeigt werden. ■ Anzahl der Achsen ■ Liste der Zusatzachsen ■ Version der Maschinendaten
System	■ Name des Steuerungs-PCs ■ Version des Betriebssystems ■ Speicherkapazitäten
Optionen	Zusätzlich installierte Optionen und Technologiepakete

Registerkarte	Beschreibung
Kommentare	Zusätzliche Kommentare
Module	Name und Version wichtiger Systemdateien Die Schaltfläche **Exportieren** exportiert den Inhalt der Registerkarte **Module** in die Datei C:\KRC\ROBOTER\LOG\FILEVERSIONS.TXT.

4.17.15 Roboterdaten anzeigen/bearbeiten

Voraussetzung

- Betriebsart T1 oder T2
- Kein Programm ist angewählt.

Vorgehensweise

- Im Hauptmenü **Inbetriebnahme** > **Roboterdaten** wählen.

Beschreibung

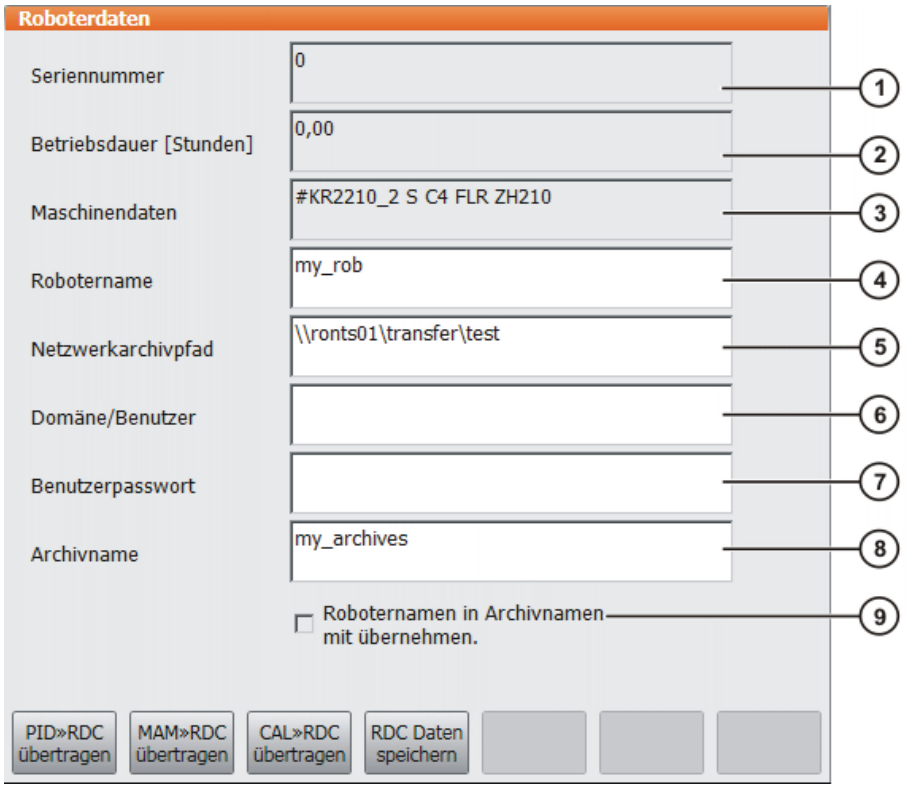


Abb. 4-36: Fenster Roboterdaten

Pos.	Beschreibung
1	Seriennummer
2	Betriebsdauer. Der Betriebsstundenzähler läuft, wenn die Antriebe eingeschaltet sind. Alternativ kann die Betriebsdauer über die Variable $ROBRUNTIME angezeigt werden.
3	Name der Maschinendaten
4	Robotername. Der Robotername kann geändert werden.
5	Daten der Robotersteuerung können archiviert werden. Hier wird das Ziel-Verzeichnis festgelegt. Es kann ein Netzwerk-Verzeichnis oder ein lokales Verzeichnis sein. Wenn hier ein Verzeichnis festgelegt ist, dann steht dieses auch für den Import/Export von Langtexten zur Verfügung.

Pos.	Beschreibung
6 7	Wenn für die Archivierung auf das Netzwerk ein Benutzernamen und ein Passwort erforderlich sind, können diese hier eingetragen werden. Sie brauchen dann bei der Archivierung nicht jedesmal angegeben werden.
8	Dieses Feld wird nur angezeigt, wenn die Checkbox **Robotername in Archivnamen mit übernehmen.** nicht aktiv ist. Hier kann ein Name für die Archivdatei festgelegt werden.
9	■ **Checkbox aktiv**: Als Name für die Archivdatei wird der Robotername verwendet. Falls kein Robotername festgelegt ist, wird als Name *archive* verwendet. ■ **Checkbox inaktiv**: Für die Archivdatei kann ein eigener Name festgelegt werden.

In der Benutzergruppe Experte stehen folgende Schaltflächen zur Verfügung:

Schaltfläche	Beschreibung
PID>>RDC übertragen	Nur relevant bei positioniergenauen Robotern: Die XML-Datei mit den Daten für den positioniergenauen Roboter kann manuell auf den RDC übertragen werden. Mit der Schaltfläche wird die Verzeichnisstruktur angezeigt. Über diese wählt man das Verzeichnis aus, in dem sich die Datei mit der aktuellen Seriennummer befindet. Die Datei kann markiert und auf den RDC übertragen werden.
MAM>>RDC übertragen	Nur relevant bei Robotern, deren Justagemarken fix gesetzt sind: Die MAM-Datei mit den roboterspezifischen Justage-Offset-Daten kann manuell auf den RDC übertragen werden. Mit der Schaltfläche wird die Verzeichnisstruktur angezeigt. Über diese wählt man das Verzeichnis aus, in dem sich die Datei mit der aktuellen Seriennummer befindet. Die Datei kann markiert und auf den RDC übertragen werden.
CAL>>RDC übertragen	Die CAL-Datei mit den Daten aus der EMD-Justage kann manuell auf den RDC übertragen werden. Mit der Schaltfläche wird die Verzeichnisstruktur angezeigt. Über diese wählt man das Verzeichnis aus, in dem sich die Datei mit der aktuellen Seriennummer befindet. Die Datei kann markiert und auf den RDC übertragen werden.
RDC Daten speichern	Mit der Schaltfläche können die Daten, die sich auf dem RDC befinden, als Backup im Verzeichnis C:\KRC\Roboter\RDC temporär gesichert werden. **Hinweis**: Bei einem Neustart der Robotersteuerung oder bei einer Archivierung von Daten wird das Verzeichnis gelöscht. Wenn die RDC-Daten dauerhaft erhalten bleiben sollen, müssen sie an anderer Stelle gesichert werden.

4.18 Akku-Zustand anzeigen

Beschreibung

Bei einer Spannungsabschaltung (d. h. Abschaltung über den Hauptschalter) oder einem Spannungsausfall wird die Robotersteuerung über einen Akku gepuffert und fährt geregelt herunter (ohne Datenverlust). Der Ladezustand dieses Akkus kann dem Benutzer angezeigt werden. Der Benutzer kann ihn außerdem an die SPS übermitteln.

Der Ladezustand des Akkus wird über die Systemvariable $ACCU_STATE angezeigt.

Der Zustand kann nur angezeigt, aber nicht geändert werden.

Bei jedem Hochlauf der Robotersteuerung wird der Verlauf des Ladestroms beobachtet. Zusätzlich wird in zyklischen Abständen ein zusätzlicher Akku-Test durchgeführt. Aus den Informationen bezüglich des Ladestroms und des Akku-Tests ergibt sich der Zustand von $ACCU_STATE.

Zustände

Die folgenden Tabellen geben die möglichen Zuständen von $ACCU_STATE an.

Die Info an die SPS muss der Benutzer selbst konfigurieren.

Informationen zum Tauschen des Akkus sind in der Betriebsanleitung für die Robotersteuerung zu finden.

#CHARGE_OK
Bedeutung: Der Ladestrom ist nach dem Hochlauf wie erforderlich abgesunken und/oder der Akku wurde beim Akku-Test positiv getestet.
Erforderliche Aktion des Benutzers: Akku nicht tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung ist ok.
Meldung: Keine Meldung

#CHARGE_OK_LOW
Bedeutung: Der Ladestrom ist nach dem Hochlauf wie erforderlich abgesunken und/oder der Akku wurde beim Akku-Test positiv getestet. Der Akku ist jedoch nach der maximalen Ladezeit nicht geladen.
Erforderliche Aktion des Benutzers: Akku tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung ist ok.
Meldung: *Akku Warnung - volle Ladung nicht möglich*

#CHARGE_UNKNOWN
Bedeutung: Der Akku wird geladen. Oder der Akku wurde nach dem Hochlauf noch nicht vom Akku-Test geprüft. Oder der Ladestrom ist noch nicht ausreichend abgesunken.
Erforderliche Aktion des Benutzers: Akku nicht tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung kann zu Fehlern beim Hibernate führen.
Meldung: Keine Meldung

#CHARGE_TEST_NOK
Bedeutung: Der Akku wurde beim Akku-Test negativ getestet
Erforderliche Aktion des Benutzers: Akku tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung kann zu Fehlern beim Hibernate führen.
Meldung: *Akku defekt - Belastungstest fehlgeschlagen*

#CHARGE_NOK
Bedeutung: Es ist kein Akku-Test möglich. Der Akku ist nach der maximalen Ladezeit nicht geladen.
Erforderliche Aktion des Benutzers: Akku tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung kann zu Fehlern beim Warmstart führen.
Meldung: *Akku defekt - Pufferung kann nicht zuverlässig sichergestellt werden*

#CHARGE_OFF
Bedeutung: Es ist kein Akku vorhanden oder der Akku ist defekt.
Erforderliche Aktion des Benutzers: Akku tauschen.
Info an SPS: Das Abschalten der Versorgungsspannung kann zu Fehlern beim Warmstart führen.
Meldung: *Akku defekt - Pufferung nicht möglich*

5 Inbetriebnahme und Wiederinbetriebnahme

5.1 Inbetriebnahme-Assistent

Beschreibung

Die Inbetriebnahme kann mit Hilfe des Inbetriebnahme-Assistenten durchgeführt werden. Dieser leitet den Benutzer durch die grundlegenden Schritte der Inbetriebnahme.

Voraussetzung

- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

- Im Hauptmenü **Inbetriebnahme** > **Inbetriebnahme-Assistent** auswählen.

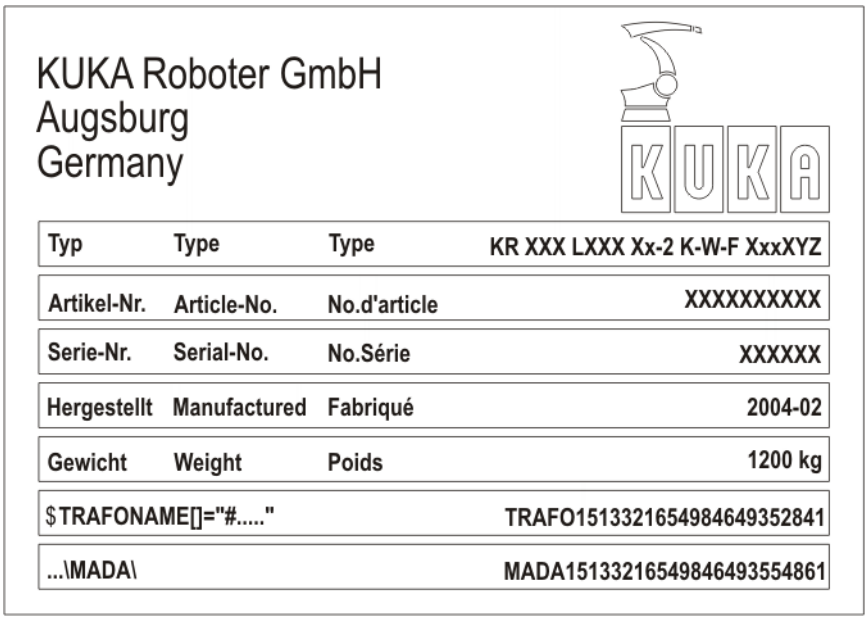
5.2 Maschinendaten prüfen

Beschreibung

Die richtigen Maschinendaten müssen geladen sein. Dies muss überprüft werden, indem man die geladenen Maschinendaten mit den Maschinendaten auf dem Typenschild vergleicht.

Wenn Maschinendaten neu geladen werden, muss der Stand der Maschinendaten exakt zum Stand der KSS passen. Dies ist gewährleistet, wenn die Maschinendaten verwendet werden, die zusammen mit dem verwendeten KSS-Release ausgeliefert wurden.

⚠ WARNUNG Wenn die falschen Maschinendaten geladen sind, darf der Industrieroboter nicht verfahren werden! Tod, schwere Verletzungen oder erhebliche Sachschäden können sonst die Folge sein. Die richtigen Maschinendaten müssen geladen werden.


KUKA Roboter GmbH
Augsburg
Germany

Typ	Type	Type	KR XXX LXXX Xx-2 K-W-F XxxXYZ
Artikel-Nr.	Article-No.	No.d'article	XXXXXXXXX
Serie-Nr.	Serial-No.	No.Série	XXXXXX
Hergestellt	Manufactured	Fabriqué	2004-02
Gewicht	Weight	Poids	1200 kg
$TRAFONAME[]="#....."			TRAFO15133216549846493528 41
...\MADA\			MADA15133216549846493554861

Abb. 5-1: Typenschild

Der Pfad, auf dem sich die Maschinendaten auf der CD befinden, ist auf dem Typenschild in der Zeile **...\MADA** angegeben.

Voraussetzung

- Betriebsart T1 oder T2
- Kein Programm ist angewählt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Roboterdaten** wählen.

Das Fenster **Roboterdaten** öffnet sich.

2. Folgende Angaben abgleichen:
   - Im Fenster **Roboterdaten**: Angabe im Feld **Maschinendaten**
   - Auf dem Typenschild an der Basis des Roboters: Angabe in der Zeile **$TRAFONAME()="# ..... "**

5.3 Hardware-Optionen festlegen

Voraussetzung

- Benutzergruppe Sicherheitsinstandhalter
- Betriebsart T1 oder T2

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Sicherheitskonfiguration** wählen.
2. Auf **Hardware-Optionen** drücken.
3. Hardware-Optionen ändern und **Speichern** drücken.

⚠ WARNUNG Nach Änderungen an der Sicherheitskonfiguration müssen die sicheren Achsüberwachungen geprüft werden. (>>> 6.4 "Sichere Achsüberwachungen prüfen" Seite 172)

Beschreibung

Parameter	Beschreibung
Kundenschnittstelle	Hier muss ausgewählt werden, welche Schnittstelle verwendet wird: ■ **automatisch** ■ **SIB mit Betriebsartausgang**
Schaltung des Peripherieschützes (US2)	■ **deaktiviert**: Das Peripherieschütz wird nicht verwendet. (Default) ■ **per externer SPS**: Das Peripherieschütz wird von einer externen SPS geschaltet, über den Eingang US2. ■ **per KRC**: Das Peripherieschütz wird in Abhängigkeit von der Fahrfreigabe geschaltet. Wenn die Fahrfreigabe vorhanden ist, wird das Schütz eingeschaltet. **Hinweise:** ■ Bei Robotersteuerungen mit Peripherieschütz und der Option "UL" muss dieser Parameter auf **per KRC** gesetzt werden. ■ Bei Robotersteuerungen, die kein Peripherieschütz besitzen, ist dieser Parameter ohne Auswirkung. Die Systemvariable $US2_VOLTAGE_ON zeigt den Status der Peripherie-Spannung US2 an: ■ TRUE: Spannung ist eingeschaltet. ■ FALSE: Spannung ist ausgeschaltet.
Bedienerschutz Quittierung	Wenn das Signal "Bedienerschutz" im Automatikbetrieb verlorenging und wieder gesetzt wird, muss es quittiert werden, bevor der Betrieb fortgesetzt werden kann. ■ **per Quittierungstaste**: Die Quittierung erfolgt z. B. über einen Quittierungstaster (außerhalb der Zelle angebracht). Die Quittierung wird an die Sicherheitssteuerung kommuniziert. Die Sicherheitssteuerung gibt den Automatikbetrieb erst nach der Quittierung wieder frei. ■ **externe Baugruppe**: Die Quittierung erfolgt über die Anlagen-SPS.

5.4 Sicherheits-ID des PROFINET-Geräts ändern

Beschreibung

Wenn mehrere KUKA-Robotersteuerung an einer PROFIsafe Master-SPS betrieben werden, muss jedes PROFINET-Gerät eine eindeutige Sicherheits-ID besitzen. Die Default-ID ist immer 7.

Voraussetzung

- Benutzergruppe Sicherheitsinstandhalter
- Betriebsart T1 oder T2

> Wenn auf der Robotersteuerung eine der Optionen KUKA.SafeOperation oder KUKA.SafeRangeMonitoring installiert ist, können andere Benutzergruppen gelten. Informationen sind den Dokumentationen der genannten Optionen zu entnehmen.

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Sicherheitskonfiguration** wählen.
2. Auf **Kommunikationsparameter** drücken.
3. In der Spalte **Neue Safety ID** auf die zu ändernde ID drücken und die ID ändern.
4. Auf **Safety-IDs übernehmen** drücken.
5. Es wird eine Sicherheitsabfrage angezeigt, ob die Änderung übernommen werden soll. Abfrage mit **Ja** bestätigen.
6. Es wird eine Meldung angezeigt, dass die Änderung gespeichert wurden. Meldung mit **OK** bestätigen.

> Mit dieser Vorgehensweise können nur Änderungen an der Sicherheits-ID gespeichert werden. Wenn es in der restlichen Sicherheitskonfiguration noch ungespeicherte Änderungen gab, wurden diese nicht gespeichert.
>
> Wenn man die Sicherheitskonfiguration schließt, kommt in diesem Fall eine Abfrage, ob man die Änderungen verwerfen will oder den Vorgang abbrechen will. Um nun alle Änderungen zu speichern, folgendermaßen vorgehen:
>
> 1. Vorgang abbrechen.
> 2. In der Sicherheitskonfiguration auf **Speichern** drücken. (Wenn die Schaltfläche **Speichern** nicht zur Verfügung steht, zuerst mit **Zurück** eine Ebene zurück.)
> 3. Es wird eine Sicherheitsabfrage angezeigt, ob alle Änderung übernommen werden soll. Abfrage mit **Ja** bestätigen.
> 4. Es wird eine Meldung angezeigt, dass die Änderung gespeichert wurden. Meldung mit **OK** bestätigen.
>
>   Alle Änderungen in der Sicherheitskonfiguration wurden gespeichert.

> **WARNUNG** Nach Änderungen an der Sicherheitskonfiguration müssen die sicheren Achsüberwachungen geprüft werden.
> (>>> 6.4 "Sichere Achsüberwachungen prüfen" Seite 172)

5.5 Roboter ohne übergeordnete Sicherheitssteuerung verfahren

Beschreibung

Um den Roboter ohne übergeordnete Sicherheitssteuerung zu verfahren, muss der Inbetriebnahme-Modus aktiviert werden. Der Roboter kann dann in T1 verfahren werden.

Wenn die Option RoboTeam verwendet wird, dann ist es nur über das lokale smartPAD möglich, den Inbetriebnahme-Modus zu aktivieren und den Roboter zu verfahren.

GEFAHR Im Inbetriebnahme-Modus sind die externen Schutzeinrichtungen außer Betrieb. Die Sicherheitshinweise zum Inbetriebnahme-Modus beachten.
(>>> 3.8.3.2 "Inbetriebnahme-Modus" Seite 39)

In folgenden Fällen beendet die Robotersteuerung den Inbetriebnahme-Modus automatisch:

- Wenn 30 min nach der Aktivierung noch keine Bedienhandlung vorgenommen wurde.
- Wenn das smartPAD passiv geschaltet wird oder von der Robotersteuerung getrennt wird.
- Wenn die Ethernet-Sicherheitsschnittstelle verwendet wird: Wenn eine Verbindung zu einer übergeordneten Sicherheitssteuerung aufgebaut wird.
- Wenn eine diskrete Sicherheitsschnittstelle verwendet wird:

  System Software 8.2 und kleiner: Die Robotersteuerung beendet den Inbetriebnahme-Modus automatisch, wenn nicht mehr sämtliche Eingangssignale an der diskreten Schnittstelle (und, falls verwendet, an der diskreten Sicherheitsschnittstelle für Sicherheitsoptionen) den Zustand "logisch Null" haben.

  Ab der System Software 8.3 dagegen ist der Inbetriebnahme-Modus unabhängig von den Eingängen an den diskreten Sicherheitsschnittstellen.

Auswirkung

Wenn der Inbetriebnahme-Modus aktiviert wird, gehen alle Ausgänge automatisch in den Zustand "logisch Null".

Wenn die Robotersteuerung ein Peripherieschütz (US2) besitzt und wenn in der Sicherheitskonfiguration festgelegt ist, dass dieses in Abhängigkeit von der Fahrfreigabe schaltet, dann gilt dies auch im Inbetriebnahme-Modus. D. h., wenn die Fahrfreigabe vorhanden ist, ist – auch im Inbetriebnahme-Modus – die US2-Spannung eingeschaltet.

Im Inbetriebnahme-Modus wird auf folgendes simuliertes Eingangsabbild umgeschaltet:

- Der externe NOT-HALT liegt nicht an.
- Die Schutztür ist geöffnet.
- Der Sicherheitshalt 1 wird nicht angefordert.
- Der Sicherheitshalt 2 wird nicht angefordert.
- Der sichere Betriebshalt wird nicht angefordert.
- Nur für VKR C4: E2 ist geschlossen.

Wenn SafeOperation oder SafeRangeMonitoring verwendet wird, beeinflusst der Inbetriebnahme-Modus weitere Signale.

Informationen zu den Auswirkungen des Inbetriebnahme-Modus, wenn SafeOperation oder SafeRangeMonitoring verwendet wird, sind in den Dokumentationen **SafeOperation** und **SafeRangeMonitoring** zu finden.

Voraussetzung

- Betriebsart T1
- Bei VKR C4: Es sind keine E2/E7-Signale über USB-Stick oder Retrofit-Schnittstelle aktiviert.
- Bei RoboTeam: Das lokale smartPAD wird verwendet.
- Wenn die Ethernet-Sicherheitsschnittstelle verwendet wird: Keine Verbindung zu einer übergeordneten Sicherheitssteuerung

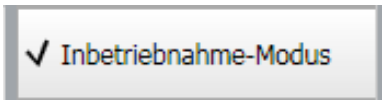
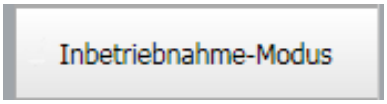
- Wenn eine diskrete Sicherheitsschnittstelle verwendet wird:

  Nur für System Software 8.2: Sämtliche Eingangssignale haben den Zustand "logisch Null". Wenn zusätzlich die diskrete Sicherheitsschnittstelle für Sicherheitsoptionen verwendet wird, müssen auch dort die Eingänge "logisch Null" sein.

  (Ab der System Software 8.3 ist der Inbetriebnahme-Modus unabhängig vom Zustand dieser Eingänge.)

Vorgehensweise

- Im Hauptmenü **Inbetriebnahme** > **Service** > **Inbetriebnahme-Modus** wählen.

Menü	Beschreibung
	Der Inbetriebnahme-Modus ist aktiv. Berühren des Menüpunkt deaktiviert den Modus.
	Der Inbetriebnahme-Modus ist nicht aktiv. Berühren des Menüpunkt aktiviert den Modus.

5.6 Aktivierung des positioniergenauen Robotermodells prüfen

Beschreibung

Wenn ein positioniergenauer Roboter verwendet wird, muss überprüft werden, ob das positioniergenaue Robotermodell aktiviert ist.

Bei positioniergenauen Robotern werden Positionsabweichungen aufgrund von Bauteiltoleranzen und elastischen Effekten der einzelnen Roboter kompensiert. Der positioniergenaue Roboter positioniert den programmierten TCP im gesamten kartesischen Arbeitsraum innerhalb der Toleranzgrenzen. Die Modellparameter des positioniergenauen Roboters werden an einem Messplatz ermittelt und dauerhaft am Roboter gespeichert (RDC).

Das positioniergenaue Robotermodell ist nur für den Auslieferungszustand des Roboters gültig.
Nach Um- oder Nachrüsten des Roboters, z. B. durch Armverlängerung oder neue Hand, muss der Roboter neu vermessen werden.

Funktionen

Ein positioniergenauer Roboter verfügt über folgende Funktionen:

- Erhöhte Positioniergenauigkeit, ca. um den Faktor 10
- Erhöhte Bahngenauigkeit

Voraussetzung für die erhöhte Positionier- und Bahngenauigkeit ist die korrekte Eingabe der Lastdaten in die Robotersteuerung.

- Vereinfachte Übernahme von Programmen bei Austausch des Roboters (Kein Nachteachen)
- Vereinfachte Übernahme von Programmen nach Offline-Programmierung mit WorkVisual (Kein Nachteachen)

Vorgehensweise

1. Im Hauptmenü **Hilfe** > **Info** wählen.
2. In der Registerkarte **Roboter** prüfen, ob das positioniergenaue Robotermodell aktiviert ist. (= Angabe **Positioniergenauer Roboter**).

5.7 Palettiermodus aktivieren

Beschreibung

Nur relevant für Palettierroboter mit 6 Achsen!

Bei Palettierrobotern mit 6 Achsen ist der Palettiermodus defaultmäßig inaktiv und muss aktiviert werden. Wenn der Palettiermodus aktiv ist, ist A4 auf 0 ° arretiert und der Anbauflansch ist parallel zum Boden.

Voraussetzung

- Der Roboter ist justiert.
- Der Roboter ist ohne Last. D. h., es ist kein Werkzeug oder Werkstück und keine Zusatzlast montiert.

Vorgehensweise

- Den Palettiermodus im Programm folgendermaßen aktivieren:

```
$PAL_MODE = TRUE
```

Alternative Vorgehensweise

1. $PAL_MODE über die Variablenkorrektur auf TRUE setzen.
2. Folgende Meldung wird angezeigt: *Palettiermodus: Achse A4 [Richtung] in Position fahren*.

   Die A4 in der in der Meldung angegebenen Richtung (Plus oder Minus) verfahren.
3. Sobald die A4 ihre Position (0 °) erreicht hat, wird folgende Meldung angezeigt: *Palettiermodus: Achse A5 [Richtung] in Position fahren*.

   Die A5 in der in der Meldung angegebenen Richtung (Plus oder Minus) verfahren.

   Sobald die A5 ihre Position (90 °) erreicht hat, blendet sich die Meldung aus.

Neben den Verfahrtasten sind nun die Bezeichnungen **A4** und **A5** ausgeblendet. Diese Achsen können nicht mehr verfahren werden.

Einschränkungen

- $PAL_MODE wird bei jedem Kaltstart der Robotersteuerung automatisch auf FALSE gesetzt.

Empfehlung: $PAL_MODE = TRUE in den Initialisierungsteil aller Programme für den Palettierroboter aufnehmen.

- Bei Robotern mit aktivem Palettiermodus ist keine Traglastermittlung mit KUKA.LoadDataDetermination möglich.

⚠ WARNUNG Bei Robotern mit aktivem Palettiermodus darf keine Traglastermittlung mit KUKA.LoadDataDetermination durchgeführt werden. Verletzungen oder Sachschäden können die Folge sein.

- Wenn der Palettiermodus aktiv ist, kann der Roboter nicht justiert werden. Wenn dennoch eine Justage notwendig ist, folgendermaßen vorgehen:
  a. Alle Lasten vom Roboter entfernen.
  b. $PAL_MODE über die Variablenkorrektur auf FALSE setzen.
  c. Roboter justieren.
  d. $PAL_MODE auf TRUE setzen.

     (Nicht notwendig, wenn $PAL_MODE = TRUE im Initialisierungsteil aller Programme für den Palettierroboter steht.)
  e. Roboter in die Palettierstellung verfahren.
  f. Alle Lasten wieder am Roboter anbringen.

5.8 Justage

Übersicht

Jeder Roboter muss justiert werden. Nur wenn der Roboter justiert ist, kann er kartesisch bewegt werden und programmierte Positionen anfahren. Bei der Justage werden die mechanische Position und die elektronische Position des Roboters in Übereinstimmung gebracht. Dazu wird der Roboter in eine definierte mechanische Position gebracht, die Justagestellung. Dann wird für jede Achse der Geberwert gespeichert.

Die Justagestellung ist bei allen Robotern ähnlich, jedoch nicht gleich. Die genauen Positionen können sich auch zwischen den einzelnen Robotern eines Robotertyps unterscheiden.

Abb. 5-2: Justagestellung - Ungefähre Position

Ein Roboter muss in folgenden Fällen justiert werden:

Fall	Bemerkung
Bei der Inbetriebnahme	- - -
Nach Instandhaltungsmaßnahmen, bei denen der Roboter die Justage verliert, z. B. Austausch von Motor oder RDC	(>>> 5.8.8 "Referenzjustage" Seite 118)
Wenn der Roboter ohne die Robotersteuerung bewegt wurde (z. B. mit der Freidreh-Vorrichtung)	- - -
Nach Austausch eines Getriebes	Vor der neuen Justage müssen die alten Justagedaten gelöscht werden! Justagedaten werden gelöscht, indem man die Achsen manuell dejustiert. (>>> 5.8.10 "Achsen manuell dejustieren" Seite 125)
Nach dem Auffahren auf einen Endanschlag mit mehr als 250 mm/s	
Nach einer Kollision	

5.8.1 Justagemethoden

Übersicht

Welche Justagemethoden für einen Roboter verwendet werden können, ist abhängig davon, mit welchem Typ Messpatrone er ausgestattet ist. Die Typen unterscheiden sich hinsichtlich der Größe ihrer Schutzkappen.

Typ Messpatrone	Justagemethoden
Messpatrone für **SEMD** (Standard Electronic Mastering Device) Schutzkappe mit M20-Feingewinde	Justage mit dem Messtaster, Typ **SEMD** (>>> 5.8.5 "Justieren mit dem SEMD" Seite 110)
	Justage mit der Messuhr (>>> 5.8.6 "Justieren mit der Messuhr" Seite 116)
	Referenzjustage Nur für die Justage nach bestimmten Instandhaltungs-Maßnahmen (>>> 5.8.8 "Referenzjustage" Seite 118)
Messpatrone für **MEMD** (Mikro Electronic Mastering Device) Schutzkappe mit M8-Feingewinde	Justage mit dem Messtaster, Typ **MEMD** Teilweise an der A6: Justage auf Strichmarkierung (>>> 5.8.9 "Justieren mit MEMD und Strichmarkierung" Seite 119)

SEMD/MEMD

SEMD und/oder MEMD sind im Justage-Set von KUKA enthalten. Es gibt mehrere Varianten des Justage-Sets.

Abb. 5-3: Justage-Set mit SEMD und MEMD

1	Justage-Box	4	SEMD
2	Schraubendreher für MEMD	5	Leitungen
3	MEMD		

Die dünnere Leitung ist die Messleitung. Sie verbindet das SEMD oder MEMD mit der Justage-Box.

Die dickere Leitung ist die EtherCAT-Leitung. Sie wird an der Justage-Box und am Roboter am X32 angeschlossen.

HINWEIS

- Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.
- Für Messtaster, an denen die Messleitung nicht fest angebracht ist, gilt: Das Gerät immer ohne Messleitung an der Messpatrone anschrauben. Danach erst die Leitung am Gerät anbringen. Anderenfalls kann die Leitung beschädigt werden. Ebenso beim Entfernen des Geräts immer zuerst die Messleitung vom Gerät entfernen. Danach erst das Gerät von der Messpatrone entfernen.
- Nach der Justage die EtherCAT-Leitung vom Anschluss X32 entfernen. Anderenfalls können Störsignale auftreten oder Schäden verursacht werden.

5.8.2 Achsen mittels Justagemarken in Vorjustagestellung verfahren

Beschreibung

Vor jeder Justage müssen die Achsen in Vorjustagestellung gebracht werden. Dafür wird jede Achse wird so verfahren, dass die Justagemarken übereinander liegen.

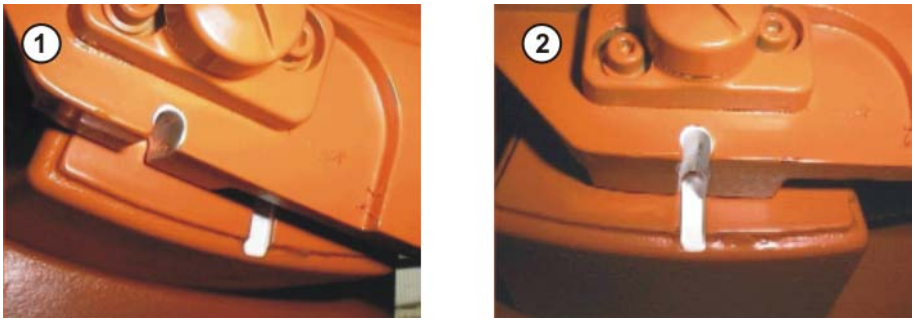


Abb. 5-4: Achse in Vorjustagestellung verfahren

In manchen Fällen ist es nicht möglich, die Achsen mit Hilfe der Justagemarken auszurichten, z. B. weil die Marken aufgrund von Verunreinigungen nicht mehr erkennbar sind. Statt mit den Justagemarken können die Achsen auch mit Hilfe des Messtasters ausgerichtet werden.
(>>> 5.8.3 "Achsen mittels Messtaster in Vorjustagestellung verfahren" Seite 108)

Die folgende Abbildung zeigt, wo am Roboter sich die Justagemarken befinden. Je nach Robotertyp weichen die Positionen geringfügig von der Abbildung ab.

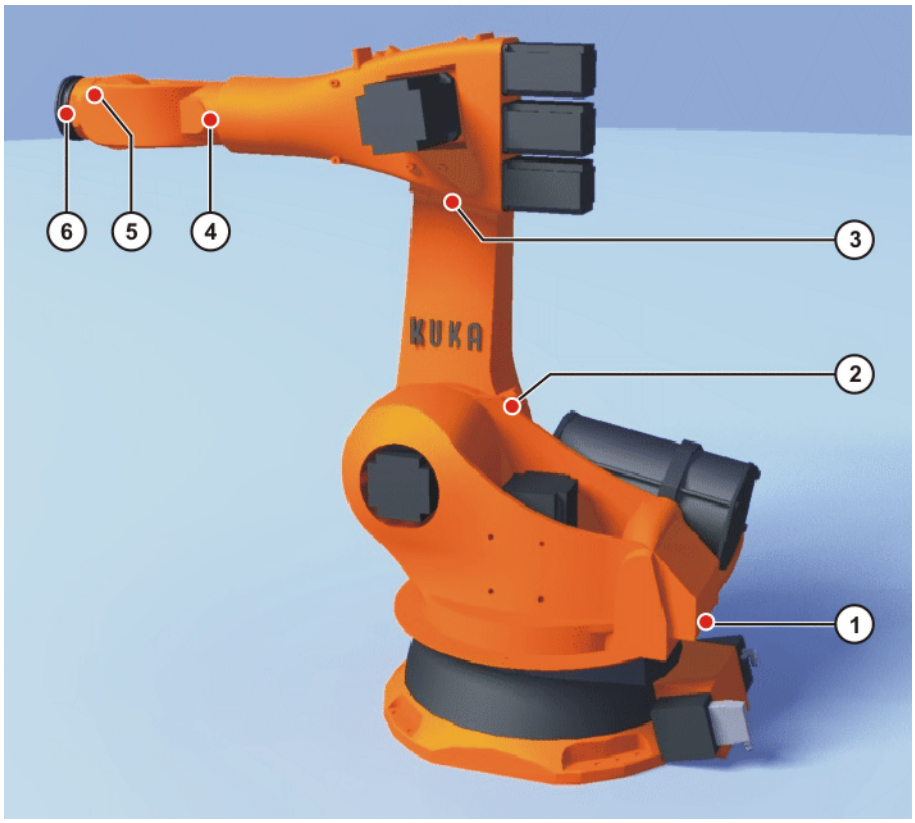


Abb. 5-5: Justagemarken am Roboter

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1

HINWEIS Bevor A4 und A6 in die Vorjustagestellung verfahren werden, sicherstellen, dass sich die Energiezuführung – sofern vorhanden – in ihrer korrekten Position befindet und nicht um 360° verdreht ist.

Vorgehensweise

1. Als Koordinatensystem für die Verfahrtasten **Achsen** auswählen.
2. Zustimmungsschalter drücken und halten.
   Neben den Verfahrtasten werden die Achsen A1 bis A6 angezeigt.
3. Auf die Plus- oder Minus-Verfahrtaste drücken, um eine Achse in positiver oder negativer Richtung zu bewegen.
4. Die Achsen aufsteigend von A1 an so verfahren, dass die Justagemarken übereinander liegen. (Außer die A6 bei Robotern, bei denen diese Achse über die Strichmarkierung justiert wird.)

5.8.3 Achsen mittels Messtaster in Vorjustagestellung verfahren

Beschreibung

Vor jeder Justage müssen die Achsen in Vorjustagestellung gebracht werden. In der Regel geschieht dies mit Hilfe der Justagemarken.

Manchmal ist dies jedoch nicht möglich, z. B. weil die Marken aufgrund von Verunreinigungen nicht mehr erkennbar sind. Statt mit den Justagemarken können die Achsen auch mit Hilfe des Messtasters ausgerichtet werden. Eine LED auf der smartHMI zeigt an, wann die Vorjustagestellung erreicht ist.

Voraussetzung

- Die Verfahrart "Verfahrtasten" ist aktiv.
- Betriebsart T1
- Es ist kein Programm angewählt.

- Der Benutzer kennt ungefähr die Vorjustagestellung der Achsen.

HINWEIS Bevor A4 und A6 in die Vorjustagestellung verfahren werden, sicherstellen, dass sich die Energiezuführung – sofern vorhanden – in ihrer korrekten Position befindet und nicht um 360° verdreht ist.

Vorgehensweise

1. Den Roboter manuell in eine Position verfahren, in der die Achsen sich ein kurzes Stück neben ihrer Vorjustagestellung befinden. Sie sollen nachher in Minus-Richtung in die Vorjustagestellung gefahren werden können.
2. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur-** wählen.

   Abhängig davon, für welchen Vorgang man die Achsen ausrichten will, wählt man nun **Erstjustage** oder **Offset lernen** oder **Mit Offset**.
3. Fortfahren gemäß der Anleitung zum jeweiligen Justagevorgang, bis der Messtaster an der A1 angebracht ist und über die Justage-Box mit dem X32 verbunden ist.

Danach NICHT weiter der Beschreibung des Justageablaufs folgen! D. h., NICHT **Justiere** oder **Lernen** oder **Prüfen** drücken!

4. Auf der smartHMI wird die LED **EMD im Justagebereich** angezeigt. Sie muss jetzt rot sein. Diese LED aufmerksam beobachten.

   (>>> 5.8.4 "Justage-LEDs" Seite 109)
5. Den Roboter manuell in Minus-Richtung verfahren. Sobald die LED von Rot auf Grün wechselt, den Roboter stoppen.

   Die A1 ist jetzt in Vorjustagestellung.

Die Achsen, die neben den LEDs angezeigt werden, blenden sich nicht wie gewohnt nach und nach aus. Dies geschieht erst bei der tatsächlichen Justage.

Die Achse nicht jetzt justieren. Erst wenn alle Achsen in Vorjustagestellung sind, darf die eigentliche Justage durchgeführt werden. Wenn dies nicht beachtet wird, kann keine korrekte Justage erreicht werden.

6. Den Messtaster so, wie im Justageablauf beschrieben, von der Messpatrone entfernen und die Schutzkappe wieder anbringen.
7. Die verbleibenden Achsen in aufsteigender Reihenfolge und auf die gleiche Weise in Vorjustagestellung bringen. (Außer die A6 bei Robotern, bei denen diese Achse über die Strichmarkierung justiert wird.)
8. Das Fenster mit den Justage-LEDs schließen.
9. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.4 Justage-LEDs

Bei den meisten Justagevorgängen zeigt die smartHMI eine Liste mit Achsen an. Rechts neben der Liste befinden sich 2 LEDs.

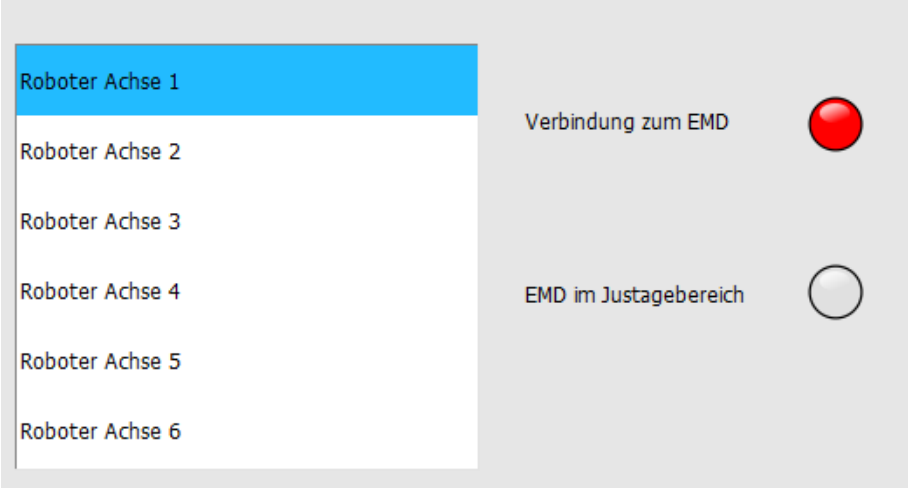


Abb. 5-6: Justage-LEDs

LED	Beschreibung
Verbindung zum EMD	■ **Rot**: Der Messtaster ist nicht mit dem Anschluss X32 verbunden. ■ **Grün**: Der Messtaster ist mit dem Anschluss X32 verbunden. Wenn diese LED rot ist, ist die LED **EMD im Justagebereich** grau.
EMD im Justagebereich	■ **Grau**: Der Messtaster ist nicht mit dem Anschluss X32 verbunden. ■ **Rot**: Der Messtaster befindet sich an einer Position, an der keine Justage möglich ist. ■ **Grün**: Der Messtaster befindet sich entweder unmittelbar neben der Justagekerbe oder in der Kerbe.

Die LED **EMD im Justagebereich** kann verwendet werden, um die Achsen mit Hilfe des Messtasters in Vorjustagestellung zu bringen. In dem Moment, in dem beim Handverfahren in Minus-Richtung die LED von Rot auf Grün wechselt, ist die Vorjustagestellung erreicht.

(>>> 5.8.3 "Achsen mittels Messtaster in Vorjustagestellung verfahren" Seite 108)

5.8.5 Justieren mit dem SEMD

Übersicht

Beim Justieren mit dem SEMD wird die Justagestellung von der Robotersteuerung automatisch angefahren. Es wird zuerst ohne, dann mit Last justiert. Es ist möglich, mehrere Justagen für verschiedene Lasten zu speichern.

Schritt	Beschreibung
1	**Erstjustage** (>>> 5.8.5.1 "Erstjustage durchführen (mit SEMD)" Seite 111) Die Erstjustage wird ohne Last durchgeführt.

Schritt	Beschreibung
2	**Offset lernen** (>>> 5.8.5.2 "Offset lernen (mit SEMD)" Seite 114) "Offset lernen" wird mit Last durchgeführt. Die Differenz zur Erstjustage wird gespeichert.
3	Bei Bedarf: **Lastjustage mit Offset prüfen** (>>> 5.8.5.3 "Lastjustage mit Offset prüfen (mit SEMD)" Seite 115) "Lastjustage mit Offset prüfen" wird mit einer Last durchgeführt, für die bereits ein Offset gelernt wurde. Anwendungsbereich: ■ Prüfen der Erstjustage ■ Wiederherstellen der Erstjustage, wenn diese verlorengegangen ist (z. B. nach Motortausch oder Kollision). Da ein gelernter Offset auch bei einem Justageverlust erhalten bleibt, kann die Robotersteuerung die Erstjustage errechnen.

5.8.5.1 Erstjustage durchführen (mit SEMD)

Voraussetzung

- Der Roboter ist ohne Last. D. h., es ist kein Werkzeug oder Werkstück und keine Zusatzlast montiert.
- Alle Achsen sind in Vorjustagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

HINWEIS Das SEMD immer ohne Messleitung an der Messpatrone anschrauben. Danach erst die Leitung am SEMD anbringen. Anderenfalls kann die Leitung beschädigt werden.
Ebenso beim Entfernen des SEMD immer zuerst die Messleitung vom SEMD entfernen. Danach erst das SEMD von der Messpatrone entfernen.
Nach der Justage die EtherCAT-Leitung vom Anschluss X32 entfernen. Anderenfalls können Störsignale auftreten oder Schäden verursacht werden.

Das tatsächlich verwendete SEMD muss nicht genau dem in den Bildern dargestellten Modell entsprechen. Die Anwendung ist die gleiche.

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Erstjustage** wählen.
   Ein Fenster öffnet sich. Alle zu justierenden Achsen werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
2. Am Anschluss X32 den Deckel abnehmen.

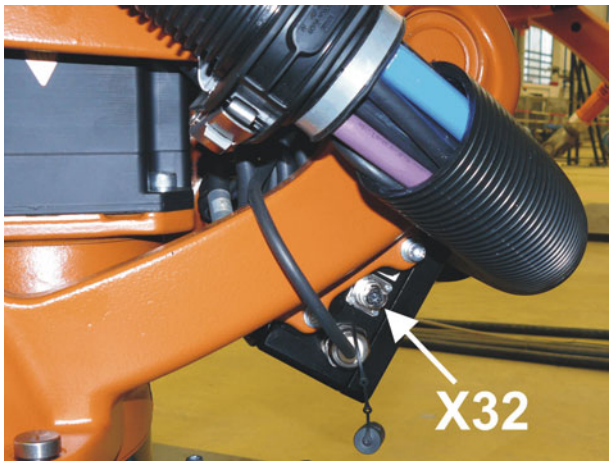


Abb. 5-7: Deckel vom X32 abnehmen

3. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.

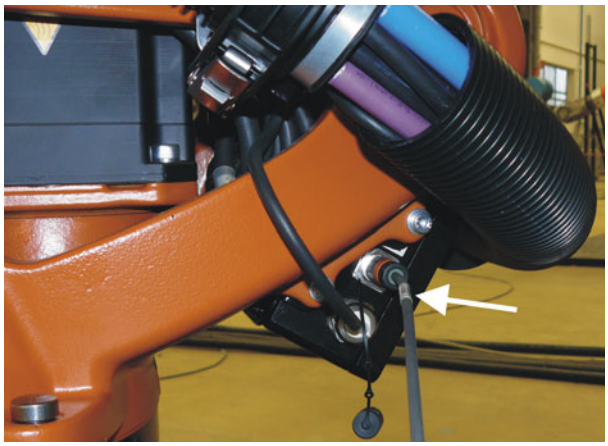


Abb. 5-8: EtherCAT-Leitung am X32 anschließen

4. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen. (Das umgedrehte SEMD kann als Schraubendreher verwendet werden.)

Abb. 5-9: Schutzkappe der Messpatrone entfernen

5. Das SEMD auf die Messpatrone schrauben.

Abb. 5-10: SEMD auf Messpatrone schrauben

6. Die Messleitung am SEMD anbringen. An der Kabelbuchse kann man erkennen, wie herum sie auf die Steckerstifte am SEMD gehört.

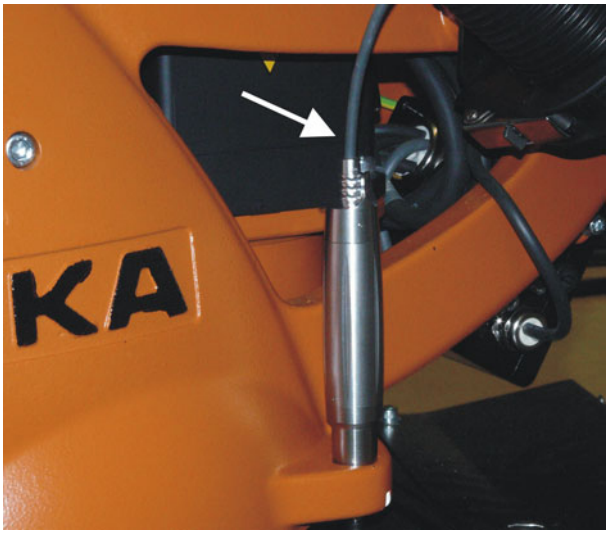


Abb. 5-11: Messleitung am SEMD anbringen

7. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
8. **Justiere** drücken.
9. Zustimmungsschalter und Start-Taste drücken.

   Wenn das SEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Die Werte werden gespeichert. Im Fenster wird die Achse ausgeblendet.
10. Messleitung vom SEMD entfernen. Dann SEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
11. Schritte 4. bis 10. für alle zu justierenden Achsen wiederholen.
12. Das Fenster schließen.
13. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.5.2 Offset lernen (mit SEMD)

Beschreibung

Offset lernen wird mit Last durchgeführt. Die Differenz zur Erstjustage wird gespeichert.

Wenn der Roboter mit verschiedenen Lasten arbeitet, muss **Offset lernen** für jede Last durchgeführt werden. Bei Greifern, die schwere Teile aufnehmen, muss **Offset lernen** jeweils für den Greifer ohne Teil und für den Greifer mit Teil durchgeführt werden.

Voraussetzung

- Gleiche Umgebungsbedingungen (Temperatur etc.) wie bei der Erstjustage
- Die Last ist am Roboter montiert.
- Alle Achsen sind in Vorjustagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

HINWEIS Das SEMD immer ohne Messleitung an der Messpatrone anschrauben. Danach erst die Leitung am SEMD anbringen. Anderenfalls kann die Leitung beschädigt werden.
Ebenso beim Entfernen des SEMD immer zuerst die Messleitung vom SEMD entfernen. Danach erst das SEMD von der Messpatrone entfernen.
Nach der Justage die EtherCAT-Leitung vom Anschluss X32 entfernen. Anderenfalls können Störsignale auftreten oder Schäden verursacht werden.

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Offset lernen** wählen.
2. Werkzeugnummer eingeben. Mit **Werkzeug OK** bestätigen.
   Ein Fenster öffnet sich. Alle Achsen, für die das Werkzeug noch nicht gelernt wurde, werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
3. Am Anschluss X32 den Deckel abnehmen. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.
4. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen. (Das umgedrehte SEMD kann als Schraubendreher verwendet werden.)
5. Das SEMD auf die Messpatrone schrauben.
6. Die Messleitung am SEMD anbringen. An der Kabelbuchse kann man erkennen, wie herum sie auf die Steckerstifte am SEMD gehört.
7. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
8. **Lernen** drücken.
9. Zustimmungsschalter und Start-Taste drücken.
   Wenn das SEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Ein Fenster öffnet sich. Die Abweichung bei dieser Achse gegenüber der Erstjustage wird in Inkrementen und Grad angezeigt.
10. Mit **OK** bestätigen. Im Fenster wird die Achse ausgeblendet.
11. Messleitung vom SEMD entfernen. Dann SEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
12. Schritte 4. bis 11. für alle zu justierenden Achsen wiederholen.
13. Das Fenster schließen.
14. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.5.3 Lastjustage mit Offset prüfen (mit SEMD)

Beschreibung

Anwendungsbereich:

- Prüfen der Erstjustage
- Wiederherstellen der Erstjustage, wenn diese verlorengegangen ist (z. B. nach Motortausch oder Kollision). Da ein gelernter Offset auch bei einem Justageverlust erhalten bleibt, kann die Robotersteuerung die Erstjustage errechnen.

Eine Achse kann nur geprüft werden, wenn alle Achsen mit niedrigerer Nummer justiert sind.

Voraussetzung

- Gleiche Umgebungsbedingungen (Temperatur etc.) wie bei der Erstjustage
- Am Roboter ist eine Last montiert, für die **Offset lernen** durchgeführt wurde.
- Alle Achsen sind in Vorjustagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

HINWEIS Das SEMD immer ohne Messleitung an der Messpatrone anschrauben. Danach erst die Leitung am SEMD anbringen. Anderenfalls kann die Leitung beschädigt werden.
Ebenso beim Entfernen des SEMD immer zuerst die Messleitung vom SEMD entfernen. Danach erst das SEMD von der Messpatrone entfernen.
Nach der Justage die EtherCAT-Leitung vom Anschluss X32 entfernen. Anderenfalls können Störsignale auftreten oder Schäden verursacht werden.

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Lastjustage** > **Mit Offset** wählen.
2. Werkzeugnummer eingeben. Mit **Werkzeug OK** bestätigen.
   Ein Fenster öffnet sich. Alle Achsen, für die ein Offset mit diesem Werkzeug gelernt wurde, werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
3. Am Anschluss X32 den Deckel abnehmen. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.
4. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen. (Das umgedrehte SEMD kann als Schraubendreher verwendet werden.)
5. Das SEMD auf die Messpatrone schrauben.
6. Die Messleitung am SEMD anbringen. An der Kabelbuchse kann man erkennen, wie herum sie auf die Steckerstifte am SEMD gehört.
7. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
8. **Prüfen** drücken.
9. Zustimmungsschalter halten und Start-Taste drücken.
   Wenn das SEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Die Differenz zu "Offset lernen" wird angezeigt.

10. Bei Bedarf die Werte mit **Sichern** speichern. Die alten Justagewerte werden dadurch gelöscht.

    Um eine verlorene Erstjustage wiederherzustellen, Werte immer speichern.

> Die Achsen A4, A5 und A6 sind mechanisch gekoppelt. Dies bedeutet:
> Wenn die Werte von A4 gelöscht werden, werden dadurch auch die Werte von A5 und A6 gelöscht.
> Wenn die Werte von A5 gelöscht werden, werden dadurch auch die Werte von A6 gelöscht.

11. Messleitung vom SEMD entfernen. Dann SEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
12. Schritte 4. bis 11. für alle zu justierenden Achsen wiederholen.
13. Das Fenster schließen.
14. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

> **HINWEIS** Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.6 Justieren mit der Messuhr

Beschreibung

Beim Justieren mit der Messuhr wird die Justagestellung vom Benutzer manuell angefahren. Es wird immer mit Last justiert. Es ist nicht möglich, mehrere Justagen für verschiedene Lasten zu speichern.

Abb. 5-12: Messuhr

Voraussetzung

- Die Last ist am Roboter montiert.
- Alle Achsen sind in Vorjustagestellung.
- Die Verfahrart "Verfahrtasten" ist aktiv und als Koordinatensystem ist **Achsen** ausgewählt.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Uhr** wählen.

   Ein Fenster öffnet sich. Alle Achsen, die nicht justiert sind, werden angezeigt. Die Achse, die als erste justiert werden muss, ist markiert.

2. An der Achse die Schutzkappe der Messpatrone entfernen und die Messuhr auf der Messpatrone anbringen.

   Mit dem Innensechskant-Schlüssel die Schrauben am Hals der Messuhr lockern. Das Zifferblatt so drehen, dass es gut eingesehen werden kann. Den Bolzen der Messuhr bis zum Anschlag in die Messuhr hineindrücken.

   Mit dem Innensechskant-Schlüssel die Schrauben am Hals der Messuhr wieder festziehen.
3. Hand-Override auf 1% reduzieren.
4. Achse von "+" nach "-" verfahren. An der tiefsten Stelle der Messkerbe, erkennbar an der Umkehr des Zeigers, die Messuhr auf Null stellen.

   Wenn die tiefste Stelle versehentlich überschritten wurde, Achse solange hin- und herverfahren, bis die tiefste Stelle erreicht ist. Es spielt keine Rolle, ob von "+" nach "-" oder "-" von "+" nach verfahren wird.
5. Achse wieder in Vorjustagestellung bringen.
6. Achse von "+" nach "-" verfahren, bis sich der Zeiger etwa 5 bis 10 Skalenteile vor Null befindet.
7. Auf das inkrementelle Handverfahren umschalten.
8. Achse von "+" nach "-" verfahren, bis die Null erreicht ist.

Wenn die Null überschritten wurde: Schritte 5. bis 8. wiederholen.

9. **Justiere** drücken. Die justierte Achse wird aus dem Fenster ausgeblendet.
10. Messuhr von der Messpatrone entfernen und Schutzkappe wieder anbringen.
11. Vom inkrementellen Handverfahren wieder in den normalen Verfahrmodus zurückschalten.
12. Schritte 2. bis 11. für alle zu justierenden Achsen wiederholen.
13. Das Fenster schließen.

5.8.7 Zusatzachsen justieren

Beschreibung

- Zusatzachsen von KUKA können sowohl mit dem Messtaster als auch mit der Messuhr justiert werden.
- Zusatzachsen, die nicht von KUKA stammen, können mit der Messuhr justiert werden. Wenn eine Justage mit dem Messtaster gewünscht ist, muss die Zusatzachse mit Messpatronen ausgerüstet werden.

Vorgehensweise

- Der Ablauf der Justage von Zusatzachsen ist der gleiche wie bei der Justage der Roboterachsen. Neben den Roboterachsen erscheinen in der Achsauswahl nun auch die projektierten Zusatzachsen.

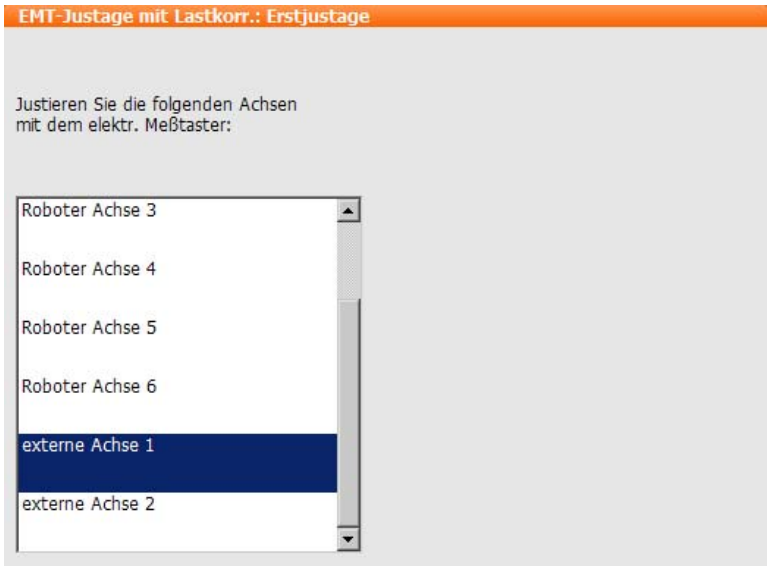


Abb. 5-13: Auswahlliste der zu justierenden Achsen

Justage bei Industrierobotern mit mehr als 2 Zusatzachsen: Bei mehr als 8 Achsen im System ist darauf zu achten, dass die Messleitung des Messtasters ggf. am zweiten RDC angeschlossen wird.

5.8.8 Referenzjustage

Die hier beschriebene Vorgehensweise darf nicht bei der Inbetriebnahme des Roboters verwendet werden.

Beschreibung

Die Referenzjustage ist geeignet, wenn bei einem korrekt justierten Roboter Instandhaltungsmaßnahmen anstehen und damit zu rechnen ist, dass der Roboter dabei die Justage verliert. Beispiele:

- RDC-Tausch
- Motortausch

Der Roboter wird vor den Instandhaltungsmaßnahmen in die Position $MAMES verfahren. Danach werden dem Roboter durch die Referenzjustage die Achswerte dieser Systemvariablen wieder zugewiesen. Der Zustand des Roboters ist dann wieder so, wie er vor dem Justageverlust war. Gelernte Offsets bleiben erhalten. Ein EMD oder eine Messuhr werden nicht benötigt.

Bei der Referenzjustage ist es unrelevant, ob am Roboter eine Last montiert ist oder nicht. Die Referenzjustage kann auch für Zusatzachsen verwendet werden.

Vorbereitung

- Vor den Instandhaltungsmaßnahmen den Roboter in die Position $MAMES verfahren. Dafür einen Punkt PTP $MAMES programmieren und anfahren. Dies kann nur von der Benutzergruppe Experte durchgeführt werden!

⚠ WARNUNG Der Roboter darf nicht statt auf $MAMES auf die Default-HOME-Position verfahren werden. $MAMES ist teilweise, aber nicht immer identisch mit der Default-HOME-Position. Nur auf der Position $MAMES wird der Roboter mit der Referenzjustage korrekt justiert. Wenn der Roboter auf einer anderen Position als $MAMES mit der Referenzjustage justiert wird, können Verletzungen und Sachschäden die Folge sein.

Voraussetzung

- Es ist kein Programm angewählt.
- Betriebsart T1
- Die Position des Roboters wurde während der Instandhaltungsmaßnahmen nicht verändert.
- Wenn der RDC getauscht wurde: Die Roboterdaten wurden von der Festplatte auf den RDC übertragen. (Dies kann nur von der Benutzergruppe Experte durchgeführt werden!)

  (>>> 4.17.15 "Roboterdaten anzeigen/bearbeiten" Seite 94)

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Referenz** wählen.

   Das Optionsfenster **Referenz-Justage** öffnet sich. Alle Achsen, die nicht justiert sind, werden angezeigt. Die Achse, die als erste justiert werden muss, ist markiert.
2. **Justiere** drücken. Die markierte Achse wird justiert und aus dem Optionsfenster ausgeblendet.
3. Schritt 2 für alle zu justierenden Achsen wiederholen.

5.8.9 Justieren mit MEMD und Strichmarkierung

Übersicht

Beim Justieren mit dem MEMD wird die Justagestellung von der Robotersteuerung automatisch angefahren. Es wird zuerst ohne, dann mit Last justiert. Es ist möglich, mehrere Justagen für verschiedene Lasten zu speichern.

- Bei Robotern, die an der A6 keine herkömmlichen Justagemarken haben, sondern Strichmarkierungen, wird die A6 ohne MEMD justiert.

  (>>> 5.8.9.1 "A6 in Justagestellung bringen (mit Strichmarkierung)" Seite 120)
- Bei Roboter, die an der A6 Justagemarken haben, wird die A6 wie die anderen Achsen justiert.

Schritt	Beschreibung
1	**Erstjustage** (>>> 5.8.9.2 "Erstjustage durchführen (mit MEMD)" Seite 120) Die Erstjustage wird ohne Last durchgeführt.
2	**Offset lernen** (>>> 5.8.9.3 "Offset lernen (mit MEMD)" Seite 123) "Offset lernen" wird mit Last durchgeführt. Die Differenz zur Erstjustage wird gespeichert.
3	Bei Bedarf: **Lastjustage mit Offset prüfen** (>>> 5.8.9.4 "Lastjustage mit Offset prüfen (mit MEMD)" Seite 124) "Lastjustage mit Offset prüfen" wird mit einer Last durchgeführt, für die bereits ein Offset gelernt wurde. Anwendungsbereich: ■ Prüfen der Erstjustage ■ Wiederherstellen der Erstjustage, wenn diese verlorengegangen ist (z. B. nach Motortausch oder Kollision). Da ein gelernter Offset auch bei einem Justageverlust erhalten bleibt, kann die Robotersteuerung die Erstjustage errechnen.

5.8.9.1 A6 in Justagestellung bringen (mit Strichmarkierung)

Beschreibung

Bei Robotern, die an der A6 keine herkömmlichen Justagemarken haben, sondern Strichmarkierungen, wird die A6 ohne MEMD justiert.

Vor der Justage muss die A6 in ihre Justagestellung gebracht werden. (Gemeint ist vor dem Gesamt-Justagevorgang, nicht direkt vor der Justage der A6 selber). Zu diesem Zweck befinden sich an der A6 feine Strichmarkierungen im Metall.

- Um die A6 in Justagestellung zu bringen, die Striche exakt aufeinander ausrichten.

Beim Anfahren der Justagestellung ist es wichtig, in gerader Linie von vorn auf den feststehenden Strich zu blicken. Wenn man von der Seite auf den Strich blickt, kann der bewegliche Strich nicht exakt genug ausgerichtet werden. Die Folge ist eine unkorrekte Justage.

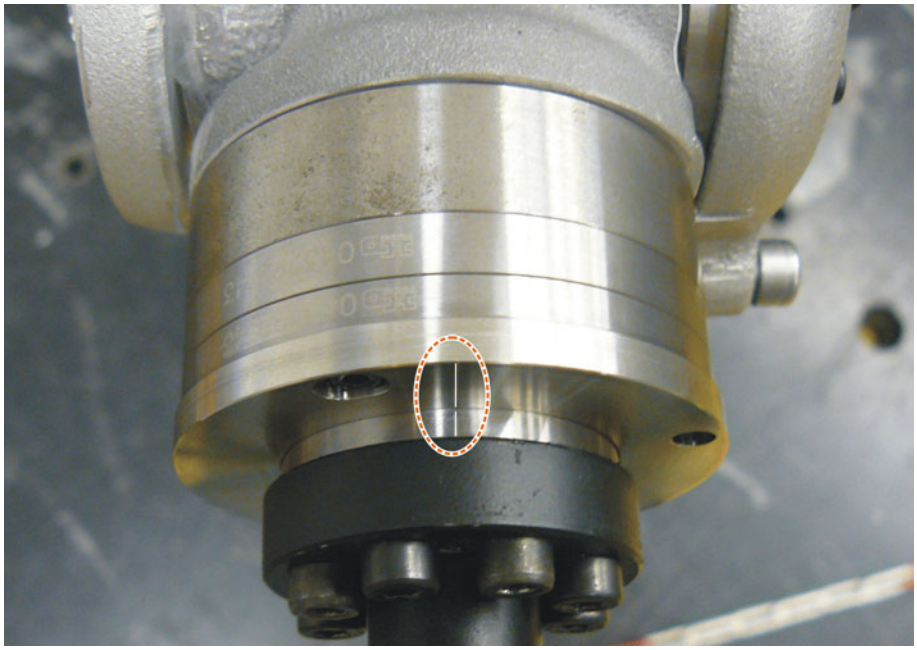


Abb. 5-14: Justagestellung A6 – Ansicht von oben vorn

Justagevorrichtung

Für die Justage der A6 am KR AGILUS existiert eine Justagevorrichtung. Diese kann optional verwendet werden. Mit der Vorrichtung lässt sich bei der Justage eine höhere Genauigkeit und eine höhere Wiederholgenauigkeit erreichen.

Weitere Informationen zur Justagevorrichtung sind in der Dokumentation **Justagevorrichtung A6** zu finden.

5.8.9.2 Erstjustage durchführen (mit MEMD)

Voraussetzung

- Der Roboter ist ohne Last. D. h., es ist kein Werkzeug oder Werkstück und keine Zusatzlast montiert.
- Die Achsen sind in Vorjustagestellung.

  Ausnahme A6, falls diese eine Strichmarkierung hat: A6 ist in Justagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Erstjustage** wählen.

   Ein Fenster öffnet sich. Alle zu justierenden Achsen werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
2. Am Anschluss X32 den Deckel abnehmen.

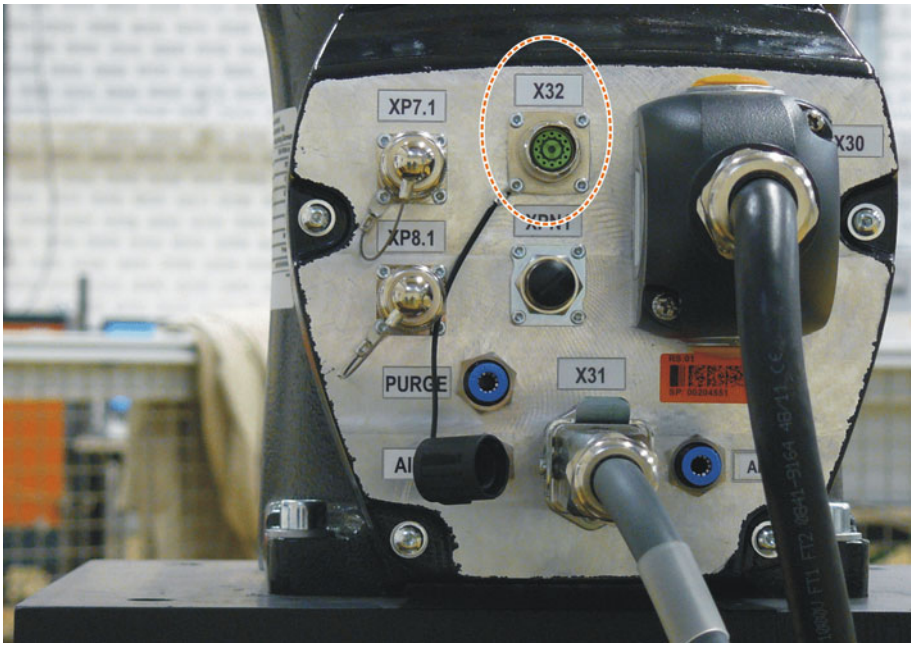


Abb. 5-15: X32 ohne Deckel

3. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.

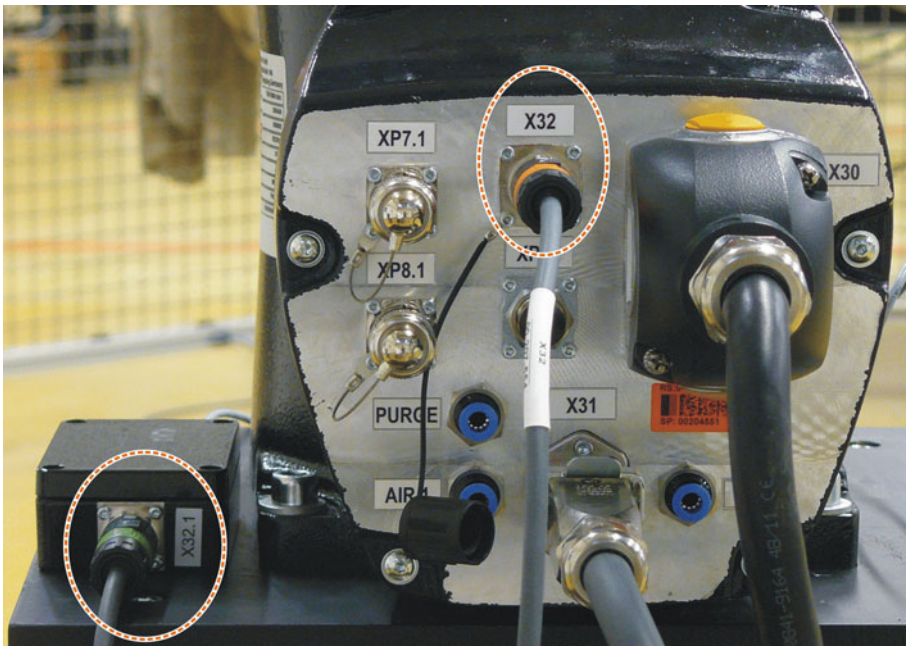


Abb. 5-16: Leitung am X32 anschließen

4. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen.

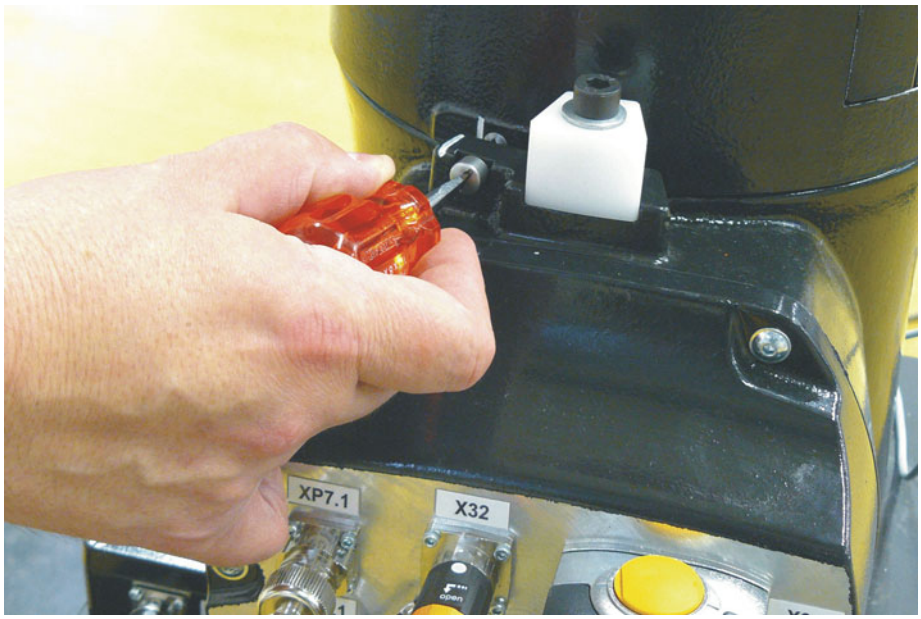


Abb. 5-17: Schutzkappe der Messpatrone entfernen

5. Das MEMD auf die Messpatrone schrauben.

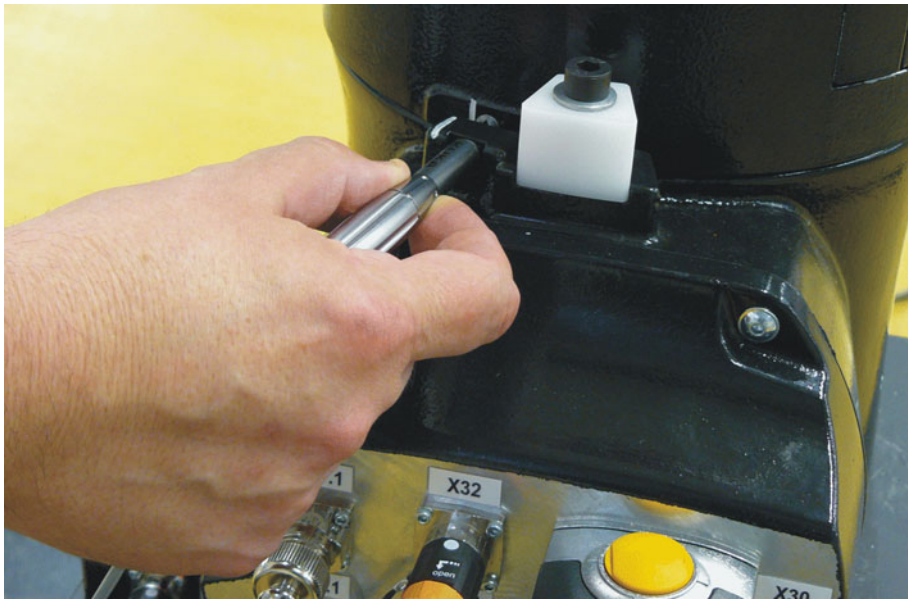


Abb. 5-18: MEMD auf Messpatrone schrauben

6. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
7. **Justiere** drücken.
8. Zustimmungsschalter und Start-Taste drücken.

   Wenn das MEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Die Werte werden gespeichert. Im Fenster wird die Achse ausgeblendet.
9. MEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
10. Schritte 4. bis 9. für alle zu justierenden Achsen wiederholen.

    Ausnahme: Nicht für die A6, wenn diese eine Strichmarkierung hat.
11. Das Fenster schließen.
12. Nur durchführen, wenn die A6 eine Strichmarkierung hat:
    a. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Referenz** wählen.

Das Optionsfenster **Referenz-Justage** öffnet sich. Die A6 wird angezeigt und ist markiert.

b. **Justiere** drücken. Die A6 wird justiert und aus dem Optionsfenster ausgeblendet.

c. Das Fenster schließen.

13. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.9.3 Offset lernen (mit MEMD)

Beschreibung

Offset lernen wird mit Last durchgeführt. Die Differenz zur Erstjustage wird gespeichert.

Wenn der Roboter mit verschiedenen Lasten arbeitet, muss **Offset lernen** für jede Last durchgeführt werden. Bei Greifern, die schwere Teile aufnehmen, muss **Offset lernen** jeweils für den Greifer ohne Teil und für den Greifer mit Teil durchgeführt werden.

Voraussetzung

- Gleiche Umgebungsbedingungen (Temperatur etc.) wie bei der Erstjustage
- Die Last ist am Roboter montiert.
- Die Achsen sind in Vorjustagestellung.

  Ausnahme A6, falls diese eine Strichmarkierung hat: A6 ist in Justagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Offset lernen** wählen.
2. Werkzeugnummer eingeben. Mit **Werkz. OK** bestätigen.

   Ein Fenster öffnet sich. Alle Achsen, für die das Werkzeug noch nicht gelernt wurde, werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
3. Am Anschluss X32 den Deckel abnehmen.
4. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.
5. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen.
6. Das MEMD auf die Messpatrone schrauben.
7. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
8. **Lernen** drücken.
9. Zustimmungsschalter und Start-Taste drücken.

   Wenn das MEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Ein Fenster öffnet sich. Die Abweichung bei dieser Achse gegenüber der Erstjustage wird in Inkrementen und Grad angezeigt.
10. Mit **OK** bestätigen. Im Fenster wird die Achse ausgeblendet.
11. MEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
12. Schritte 5. bis 11. für alle zu justierenden Achsen wiederholen.

Ausnahme: Nicht für die A6, wenn diese eine Strichmarkierung hat.

13. Das Fenster schließen.
14. Nur durchführen, wenn die A6 eine Strichmarkierung hat:
    a. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Referenz** wählen.
       Das Optionsfenster **Referenz-Justage** öffnet sich. Die A6 wird angezeigt und ist markiert.
    b. **Justiere** drücken. Die A6 wird justiert und aus dem Optionsfenster ausgeblendet.
    c. Das Fenster schließen.
15. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.9.4 Lastjustage mit Offset prüfen (mit MEMD)

Beschreibung

Anwendungsbereich:

- Prüfen der Erstjustage
- Wiederherstellen der Erstjustage, wenn diese verlorengegangen ist (z. B. nach Motortausch oder Kollision). Da ein gelernter Offset auch bei einem Justageverlust erhalten bleibt, kann die Robotersteuerung die Erstjustage errechnen.

Eine Achse kann nur geprüft werden, wenn alle Achsen mit niedrigerer Nummer justiert sind.

Bei Robotern, bei denen die A6 eine Strichmarkierung hat, wird für diese Achse der ermittelte Wert nicht angezeigt. D. h., für die A6 kann die Erstjustage nicht geprüft werden. Es ist aber möglich, eine verlorene Erstjustage wiederherzustellen.

Voraussetzung

- Gleiche Umgebungsbedingungen (Temperatur etc.) wie bei der Erstjustage
- Am Roboter ist eine Last montiert, für die **Offset lernen** durchgeführt wurde.
- Die Achsen sind in Vorjustagestellung.
  Ausnahme A6, falls diese eine Strichmarkierung hat: A6 ist in Justagestellung.
- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **EMD** > **Mit Lastkorrektur** > **Lastjustage** > **Mit Offset** wählen.
2. Werkzeugnummer eingeben. Mit **Werkz. OK** bestätigen.
   Ein Fenster öffnet sich. Alle Achsen, für die ein Offset mit diesem Werkzeug gelernt wurde, werden angezeigt. Die Achse mit der niedrigsten Nummer ist markiert.
3. Am Anschluss X32 den Deckel abnehmen.
4. Die EtherCAT-Leitung am X32 und an der Justage-Box anschließen.
5. An der Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen.
6. Das MEMD auf die Messpatrone schrauben.

7. Die Messleitung an der Justage-Box anschließen, falls nicht bereits verbunden.
8. **Prüfen** drücken.
9. Zustimmungsschalter halten und Start-Taste drücken.

   Wenn das MEMD die Messkerbe durchlaufen hat, wird die Justagestellung berechnet. Der Roboter stoppt automatisch. Die Differenz zu "Offset lernen" wird angezeigt.
10. Bei Bedarf die Werte mit **Sichern** speichern. Die alten Justagewerte werden dadurch gelöscht.

    Um eine verlorene Erstjustage wiederherzustellen, Werte immer speichern.

Die Achsen A4, A5 und A6 sind mechanisch gekoppelt. Dies bedeutet:
Wenn die Werte von A4 gelöscht werden, werden dadurch auch die Werte von A5 und A6 gelöscht.
Wenn die Werte von A5 gelöscht werden, werden dadurch auch die Werte von A6 gelöscht.

11. MEMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
12. Schritte 5. bis 11. für alle zu justierenden Achsen wiederholen.

    Ausnahme: Nicht für die A6, wenn diese eine Strichmarkierung hat.
13. Das Fenster schließen.
14. Nur durchführen, wenn die A6 eine Strichmarkierung hat:
    a. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Referenz** wählen.

       Das Optionsfenster **Referenz-Justage** öffnet sich. Die A6 wird angezeigt und ist markiert.
    b. **Justiere** drücken, um eine verlorene Erstjustage wiederherzustellen. Die A6 wird aus dem Optionsfenster ausgeblendet.
    c. Das Fenster schließen.
15. Die EtherCAT-Leitung vom Anschluss X32 und von der Justage-Box entfernen.

HINWEIS Die Messleitung an der Justage-Box angesteckt lassen und möglichst selten abstecken. Die Steckbarkeit des Sensor-Steckverbinders M8 ist begrenzt. Bei häufigem Ab-/Anstecken können am Steckverbinder Schäden auftreten.

5.8.10 Achsen manuell dejustieren

Beschreibung

Die Justagewerte der einzelnen Achsen können gelöscht werden. Beim Dejustieren bewegen sich die Achsen nicht.

Die Achsen A4, A5 und A6 sind mechanisch gekoppelt. Dies bedeutet:
Wenn die Werte von A4 gelöscht werden, werden dadurch auch die Werte von A5 und A6 gelöscht.
Wenn die Werte von A5 gelöscht werden, werden dadurch auch die Werte von A6 gelöscht.

HINWEIS Bei einem dejustierten Roboter sind die Software-Endschalter deaktiviert. Der Roboter kann gegen die Puffer an den Endanschlägen fahren, wodurch er beschädigt werden kann und die Puffer ausgetauscht werden müssen. Einen dejustierten Roboter möglichst nicht verfahren oder Hand-Override soweit wie möglich reduzieren.

Voraussetzung

- Es ist kein Programm angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Justieren** > **Dejustieren** wählen. Ein Fenster öffnet sich.
2. Die zu dejustierende Achse markieren.
3. Auf **Dejustiere** drücken. Die Justagedaten der Achse werden gelöscht.
4. Schritte 2 und 3 für alle zu dejustierenden Achsen wiederholen.
5. Das Fenster schließen.

5.9 Software-Endschalter ändern

Es gibt 2 Möglichkeiten, die Software-Endschalter zu ändern:

- Die gewünschten Werte manuell eingeben.
- Oder die Endschalter automatisch an ein oder mehrere Programme anpassen.

  Hierbei ermittelt die Robotersteuerung die minimalen und maximalen Achspositionen, die in den Programmen vorkommen. Diese Werte können dann als Software-Endschalter gesetzt werden.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1, T2 oder AUT

Vorgehensweise

Software-Endschalter manuell ändern:

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **Software-Endschalter** wählen. Das Fenster **Software-Endschalter** öffnet sich.
2. In den Spalten **Negativ** und **Positiv** die Endschalter nach Bedarf ändern.
3. Die Änderungen mit **Speichern** speichern.

Software-Endschalter an Programm anpassen:

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **Software-Endschalter** wählen. Das Fenster **Software-Endschalter** öffnet sich.
2. Auf **Automat. ermitteln** drücken. Folgende Meldung wird angezeigt: *Automatische Ermittlung läuft.*
3. Das Programm starten, an das die Endschalter angepasst werden sollen. Das Programm vollständig ablaufen lassen und dann abwählen.

   Im Fenster **Software-Endschalter** wird die erreichte maximale und minimale Position jeder Achse angezeigt.
4. Schritt 3 für alle Programme wiederholen, an die die Endschalter angepasst werden sollen.

   Im Fenster **Software-Endschalter** wird die erreichte maximale und minimale Position jeder Achse angezeigt, und zwar bezogen auf die durchlaufenen Programme insgesamt.
5. Wenn alle gewünschten Programme durchlaufen wurden, im Fenster **Software-Endschalter** auf **Ende** drücken.
6. Auf **Speichern** drücken, um die ermittelten Werte als Software-Endschalter zu übernehmen.
7. Bei Bedarf die automatisch ermittelten Werte noch manuell ändern.

Empfehlung: Die ermittelten Minimalwerte um 5° verringern. Die ermittelten Maximalwerte um 5° erhöhen.
Dieser Puffer verhindert, dass die Achsen während des Programmlaufs die Endschalter erreichen und dadurch ein Stopp ausgelöst wird.

8. Die Änderungen mit **Speichern** speichern.

Beschreibung Fenster **Software-Endschalter**:

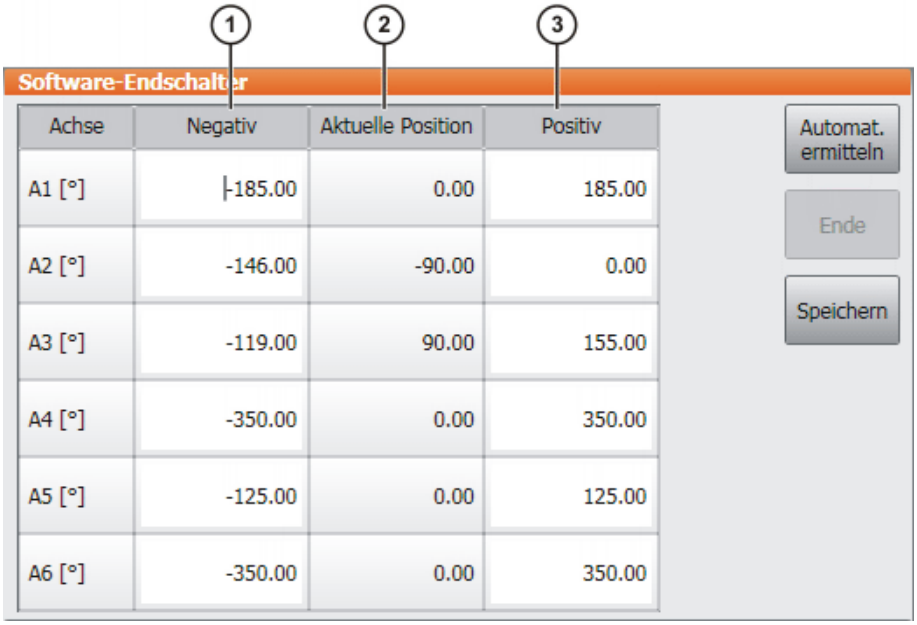


Abb. 5-19: Vor der automatischen Ermittlung

Pos.	Beschreibung
1	Aktueller negativer Endschalter
2	Aktuelle Position der Achse
3	Aktueller positiver Endschalter

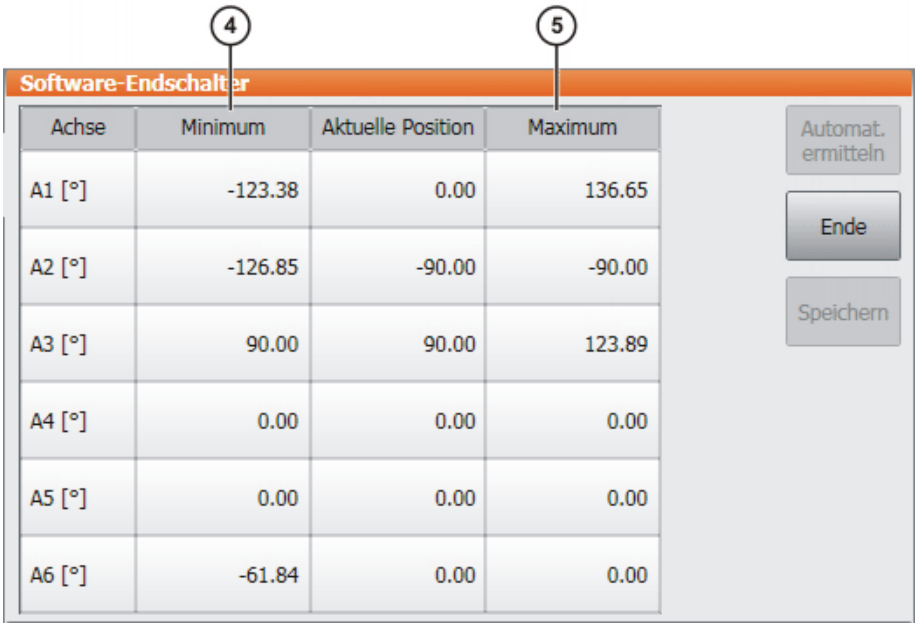


Abb. 5-20: Während der automatischen Ermittlung

Pos.	Beschreibung
4	Minimale Position, die die Achse seit Start der Ermittlung eingenommen hat
5	Maximale Position, die die Achse seit Start der Ermittlung eingenommen hat

Schaltflächen

Folgende Schaltflächen stehen zur Verfügung (nur in Benutzergruppe Experte):

Schaltfläche	Beschreibung
Automat. ermitteln	Startet die automatische Ermittlung: Die Robotersteuerung schreibt die Minimal- und Maximalpositionen, die die Achsen ab jetzt einnehmen, ins Fenster **Software-Endschalter** in die Spalten **Minimum** und **Maximum**.
Ende	Beendet die automatische Ermittlung. Überträgt die ermittelten Minimal-/Maximalpositionen in die Spalten **Negativ** und **Positiv**, aber speichert sie noch nicht.
Speichern	Speichert die Werte in den Spalten **Negativ** und **Positiv** als Software-Endschalter.

5.10 Vermessen

5.10.1 Werkzeug-Stoßrichtung festlegen

Beschreibung

Als Stoßrichtung des Werkzeugs ist im System defaultmäßig die X-Richtung festgelegt. Die Stoßrichtung kann über die Systemvariable $TOOL_DIRECTION geändert werden.

- Die Änderung bezieht sich nur auf Spline-Bewegungen. Für LIN- und CIRC-Bewegungen ist die Stoßrichtung unveränderlich die X-Richtung.
- Die Änderung gilt für alle Werkzeuge. Es ist nicht möglich, für verschiedene Werkzeuge verschiedene Stoßrichtungen festzulegen.

⚠ WARNUNG Die Stoßrichtung muss vor dem Vermessen und vor der Programmerstellung festgelegt werden. Danach darf sie nicht mehr geändert werden. Wenn dies nicht berücksichtigt wird, können unerwartete Änderungen im Fahrverhalten des Roboters auftreten. Tod, Verletzungen oder Sachschaden können die Folge sein.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

- Im Verzeichnis KRC\Steu\MaDa in der Datei $CUSTOM.DAT die Systemvariable $TOOL_DIRECTION auf den gewünschten Wert setzen.
  Mögliche Werte: #X (Default); #Y; #Z

Es ist nicht möglich, $TOOL_DIRECTION über die Variablenkorrektur oder über Schreiben vom Programm aus zu ändern.

5.10.2 Werkzeug vermessen

Beschreibung

Bei der Werkzeugvermessung weist der Benutzer einem Werkzeug, das am Anbauflansch angebracht ist, ein kartesisches Koordinatensystem (TOOL-Koordinatensystem) zu.

Das TOOL-Koordinatensystem hat seinen Ursprung in einem vom Benutzer festgelegten Punkt. Dieser heißt TCP (Tool Center Point). In der Regel wird der TCP in den Arbeitspunkt des Werkzeugs gelegt.

Bei einem feststehenden Werkzeug darf die hier beschriebene Vermessung nicht verwendet werden. Für feststehende Werkzeuge muss eine eigene Art der Vermessung verwendet werden.
(>>> 5.10.4 "Feststehendes Werkzeug vermessen" Seite 139)

Vorteile der Werkzeugvermessung:

- Das Werkzeug kann geradlinig in Stoßrichtung verfahren werden.
- Das Werkzeug kann um den TCP gedreht werden, ohne dass sich die Position des TCPs ändert.
- Im Programmbetrieb: Die programmierte Verfahrgeschwindigkeit wird entlang der Bahn am TCP gehalten.

Maximal 16 TOOL-Koordinatensysteme können gespeichert werden. Variable: TOOL_DATA[1…16]).

Gespeichert werden folgende Daten:

- X, Y, Z:

  Ursprung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem
- A, B, C:

  Orientierung des TOOL-Koordinatensystems, bezogen auf das FLANGE-Koordinatensystem

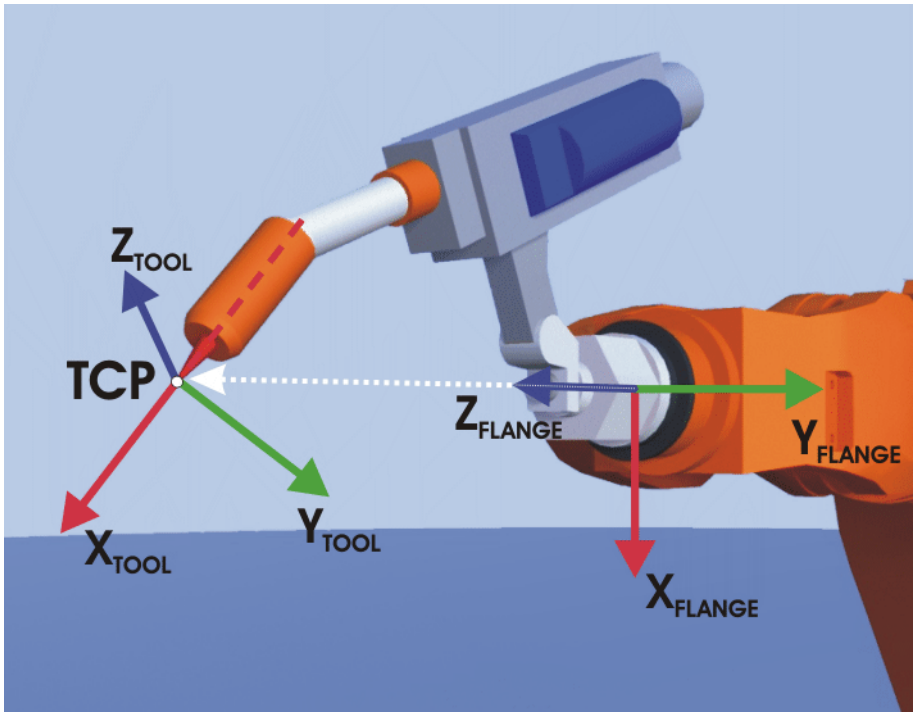


Abb. 5-21: Prinzip der TCP-Vermessung

Übersicht

Die Werkzeugvermessung besteht aus 2 Schritten:

Schritt	Beschreibung
1	**Ursprung des TOOL-Koordinatensystems festlegen** Folgende Methoden stehen zur Auswahl: ■ XYZ-4-Punkt (>>> 5.10.2.1 "TCP vermessen: XYZ 4-Punkt-Methode" Seite 130) ■ XYZ-Referenz (>>> 5.10.2.2 "TCP vermessen: XYZ Referenz-Methode" Seite 132)
2	**Orientierung des TOOL-Koordinatensystems festlegen** Folgende Methoden stehen zur Auswahl: ■ ABC-2-Punkt (>>> 5.10.2.4 "Orientierung festlegen: ABC 2-Punkt-Methode" Seite 134) ■ ABC-World (>>> 5.10.2.3 "Orientierung festlegen: ABC World-Methode" Seite 133)

Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.2.5 "Numerische Eingabe" Seite 135)

5.10.2.1 TCP vermessen: XYZ 4-Punkt-Methode

Die XYZ 4-Punkt-Methode kann für Palettierroboter nicht verwendet werden.

Beschreibung

Mit dem TCP des zu vermessenden Werkzeugs fährt man einen Referenzpunkt aus 4 verschiedenen Richtungen an. Der Referenzpunkt kann beliebig gewählt werden. Aus den unterschiedlichen Flanschpositionen berechnet die Robotersteuerung den TCP.

Die 4 Flanschpositionen, mit denen der Referenzpunkt angefahren wird, müssen ausreichend weit auseinander liegen.

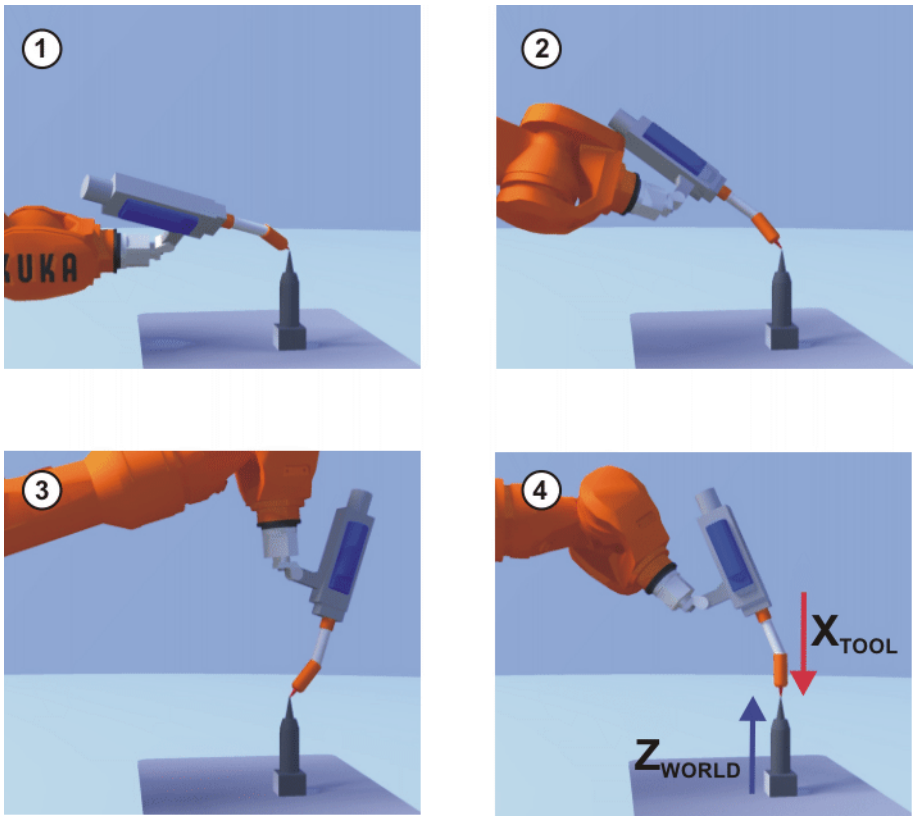


Abb. 5-22: XYZ-4-Punkt-Methode

Voraussetzung

- Das zu vermessende Werkzeug ist am Anbauflansch montiert.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **XYZ 4-Punkt** wählen.
2. Eine Nummer und einen Namen für das zu vermessende Werkzeug vergeben. Mit **Weiter** bestätigen.
3. Mit dem TCP einen Referenzpunkt anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
4. Mit dem TCP den Referenzpunkt aus einer anderen Richtung anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Schritt 4 zweimal wiederholen.
6. Die Traglastdaten eingeben. (Dieser Schritt kann übersprungen werden, wenn die Traglastdaten stattdessen gesondert eingegeben werden.)
   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
7. Mit **Weiter** bestätigen.
8. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
9. Entweder: **Speichern** drücken und dann das Fenster über das **Schließen**-Symbol schließen.

   Oder: **ABC 2-Punkt** oder **ABC World** drücken. Die bisherigen Daten werden automatisch gespeichert und es öffnet sich ein Fenster, in dem man die Orientierung des TOOL-Koordinatensystems festlegen kann.
   (>>> 5.10.2.4 "Orientierung festlegen: ABC 2-Punkt-Methode" Seite 134)
   (>>> 5.10.2.3 "Orientierung festlegen: ABC World-Methode" Seite 133)

5.10.2.2 TCP vermessen: XYZ Referenz-Methode

Beschreibung

Bei der XYZ Referenz-Methode wird ein neues Werkzeug mit einem bereits vermessenen Werkzeug vermessen. Die Robotersteuerung vergleicht die Flanschpositionen und errechnet den TCP des neuen Werkzeugs.

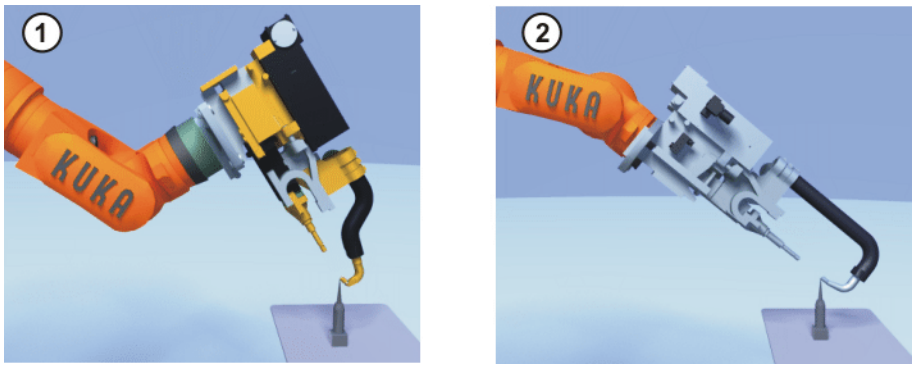


Abb. 5-23: XYZ-Referenz-Methode

Voraussetzung

- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Betriebsart T1

Vorbereitung

Die TCP-Daten des vermessenen Werkzeugs ermitteln:

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **XYZ Referenz** wählen.
2. Die Nummer des vermessenen Werkzeugs eingeben.
3. Die Werkzeugdaten werden angezeigt. Den X-, Y- und Z-Wert notieren.
4. Das Fenster schließen.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **XYZ Referenz** wählen.
2. Eine Nummer und einen Namen für das neue Werkzeug vergeben. Mit **Weiter** bestätigen.
3. TCP-Daten des bereits vermessenen Werkzeugs eingeben. Mit **Weiter** bestätigen.
4. Mit dem TCP einen Referenzpunkt anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Das Werkzeug freifahren und abmontieren. Das neue Werkzeug montieren.
6. Mit dem TCP des neuen Werkzeugs den Referenzpunkt anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
7. Die Traglastdaten eingeben. (Dieser Schritt kann übersprungen werden, wenn die Traglastdaten stattdessen gesondert eingegeben werden.)
   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
8. Mit **Weiter** bestätigen.
9. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
10. Entweder: **Speichern** drücken und dann das Fenster über das **Schließen**-Symbol schließen.

    Oder: **ABC 2-Punkt** oder **ABC World** drücken. Die bisherigen Daten werden automatisch gespeichert und es öffnet sich ein Fenster, in dem man die Orientierung des TOOL-Koordinatensystems festlegen kann.
    (>>> 5.10.2.4 "Orientierung festlegen: ABC 2-Punkt-Methode" Seite 134)

(>>> 5.10.2.3 "Orientierung festlegen: ABC World-Methode" Seite 133)

5.10.2.3 Orientierung festlegen: ABC World-Methode

Beschreibung

Der Benutzer richtet die Achsen des TOOL-Koordinatensystems parallel zu den Achsen des WORLD-Koordinatensystems aus. Dadurch wird der Robotersteuerung die Orientierung des TOOL-Koordinatensystems bekanntgegeben.

Die Methode hat 2 Varianten:

- **5D**: Der Benutzer gibt der Robotersteuerung die Stoßrichtung des Werkzeugs bekannt. Die Stoßrichtung ist defaultmäßig die X-Achse. Die Orientierung der anderen Achsen wird vom System festgelegt und kann vom Benutzer nicht beeinflusst werden.

  Das System legt die Orientierung der anderen Achsen immer gleich fest. Falls das Werkzeug später ein weiteres Mal vermessen werden muss, z. B. nach einem Crash, reicht es deshalb, die Stoßrichtung erneut festzulegen. Auf die Verdrehung um die Stoßrichtung muss nicht geachtet werden.
- **6D**: Der Benutzer gibt der Robotersteuerung die Richtung aller 3 Achsen bekannt.

Voraussetzung

- Das zu vermessende Werkzeug ist am Anbauflansch montiert.
- Der TCP des Werkzeugs ist bereits vermessen.
- Betriebsart T1

> Die folgende Vorgehensweise ist gültig, wenn die Werkzeug-Stoßrichtung die Default-Stoßrichtung (= X-Richtung) ist. Wenn die Stoßrichtung auf Y oder Z geändert wurde, muss auch die Vorgehensweise entsprechend geändert werden. (>>> 5.10.1 "Werkzeug-Stoßrichtung festlegen" Seite 128)

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **ABC World** wählen.
2. Die Nummer des Werkzeugs eingeben. Mit **Weiter** bestätigen.
3. Im Feld **5D/6D** eine Variante auswählen. Mit **Weiter** bestätigen.
4. Wenn **5D** gewählt wurde:

   $+X_{TOOL}$ parallel zu $-Z_{WORLD}$ ausrichten. ($+X_{TOOL}$ = Stoßrichtung)

   Wenn **6D** gewählt wurde:

   Die Achsen des TOOL-Koordinatensystems folgendermaßen ausrichten.
   - $+X_{TOOL}$ parallel zu $-Z_{WORLD}$. ($+X_{TOOL}$= Stoßrichtung)
   - $+Y_{TOOL}$ parallel zu $+Y_{WORLD}$
   - $+Z_{TOOL}$ parallel zu $+X_{WORLD}$
5. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.

> Die beiden folgenden Schritte fallen weg, wenn man die Vorgehensweise nicht über das Hauptmenü aufgerufen hat, sondern nach der TCP-Vermessung über die Schaltfläche **ABC World**.

6. Die Traglastdaten eingeben. (Dieser Schritt kann übersprungen werden, wenn die Traglastdaten stattdessen gesondert eingegeben werden.)
   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
7. Mit **Weiter** bestätigen.
8. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-

Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.

9. **Speichern** drücken.

5.10.2.4 Orientierung festlegen: ABC 2-Punkt-Methode

Beschreibung

Der Robotersteuerung werden die Achsen des TOOL-Koordinatensystems bekanntgegeben, indem ein Punkt auf der X-Achse und ein Punkt in der XY-Ebene angefahren wird.

Diese Methode wird benutzt, wenn die Achsrichtungen besonders exakt festgelegt werden müssen.

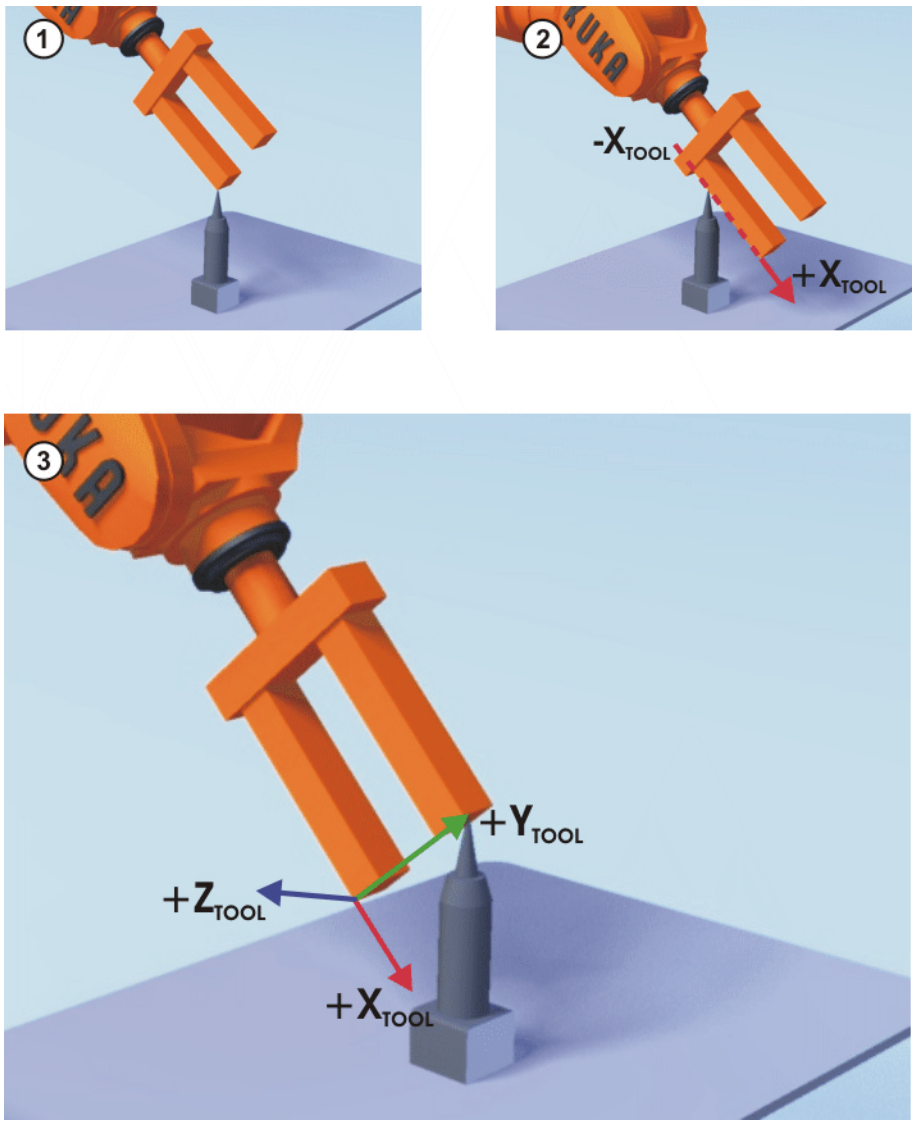


Abb. 5-24: ABC 2-Punkt-Methode

Voraussetzung

- Das zu vermessende Werkzeug ist am Anbauflansch montiert.
- Der TCP des Werkzeugs ist bereits vermessen.
- Betriebsart T1

Die folgende Vorgehensweise ist gültig, wenn die Werkzeug-Stoßrichtung die Default-Stoßrichtung (= X-Richtung) ist. Wenn die Stoßrichtung auf Y oder Z geändert wurde, muss auch die Vorgehensweise entsprechend geändert werden. (>>> 5.10.1 "Werkzeug-Stoßrichtung festlegen" Seite 128)

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **ABC 2-Punkt** wählen.
2. Die Nummer des montierten Werkzeugs eingeben. Mit **Weiter** bestätigen.
3. Mit dem TCP einen beliebigen Referenzpunkt anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
4. Das Werkzeug so verfahren, dass der Referenzpunkt auf der X-Achse auf einem Punkt mit negativem X-Wert (d. h. entgegen der Stoßrichtung) zu liegen kommt. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Das Werkzeug so verfahren, dass der Referenzpunkt auf der XY-Ebene auf einem Punkt mit positivem Y-Wert zu liegen kommt. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.

> Die beiden folgenden Schritte fallen weg, wenn man die Vorgehensweise nicht über das Hauptmenü aufgerufen hat, sondern nach der TCP-Vermessung über die Schaltfläche **ABC 2-Pkt**.

6. Die Traglastdaten eingeben. (Dieser Schritt kann übersprungen werden, wenn die Traglastdaten stattdessen gesondert eingegeben werden.)
   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
7. Mit **Weiter** bestätigen.
8. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
9. **Speichern** drücken.

5.10.2.5 Numerische Eingabe

Beschreibung

Die Daten des Werkzeugs können manuell eingegeben werden.

Mögliche Datenquellen:

- CAD
- Extern vermessenes Werkzeug
- Angaben des Werkzeug-Herstellers

> Bei Palettierrobotern mit 4 Achsen müssen die Werkzeugdaten numerisch eingegeben werden. Die XYZ- und ABC-Methoden können nicht verwendet werden, da bei diesen Robotern ein Umorientieren nur begrenzt möglich ist.

Voraussetzung

- Folgende Werte sind bekannt:
  - X, Y, Z in Bezug auf das FLANGE-Koordinatensystem
  - A, B, C in Bezug auf das FLANGE-Koordinatensystem
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **Numerische Eingabe** wählen.
2. Eine Nummer und einen Namen für das zu vermessende Werkzeug vergeben. Mit **Weiter** bestätigen.
3. Die Werkzeugdaten eingeben. Mit **Weiter** bestätigen.
4. Die Traglastdaten eingeben. (Dieser Schritt kann übersprungen werden, wenn die Traglastdaten stattdessen gesondert eingegeben werden.)
   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
5. Wenn die Online-Lastdatenprüfung zur Verfügung steht (dies ist abhängig vom Robotertyp): Nach Bedarf konfigurieren.

(>>> 5.11.5 "Online-Lastdatenprüfung (OLDC)" Seite 156)

6. Mit **Weiter** bestätigen.
7. **Speichern** drücken.

5.10.3 Basis vermessen

Beschreibung

Bei der Basisvermessung weist der Benutzer einer Arbeitsfläche oder dem Werkstück ein kartesisches Koordinatensystem (BASE-Koordinatensystem) zu. Das BASE-Koordinatensystem hat seinen Ursprung in einem vom Benutzer festgelegten Punkt.

Wenn das Werkstück am Anbauflansch angebracht ist, darf die hier beschriebene Vermessung nicht verwendet werden. Für Werkstücke am Anbauflansch muss eine eigene Art der Vermessung verwendet werden. (>>> 5.10.4 "Feststehendes Werkzeug vermessen" Seite 139)

Vorteile der Basisvermessung:

- Der TCP kann entlang der Kanten der Arbeitsfläche oder des Werkstücks manuell verfahren werden.
- Punkte können mit Bezug auf die Basis geteacht werden. Wenn die Basis verschoben werden muss, z. B. weil die Arbeitsfläche verschoben wurde, wandern die Punkte mit und müssen nicht neu geteacht werden.

Maximal 32 BASE-Koordinatensysteme können gespeichert werden. Variable: BASE_DATA[1…32].

Übersicht

Es gibt 2 Methoden, um eine Basis zu vermessen:

- 3-Punkt-Methode (>>> 5.10.3.1 "3-Punkt-Methode" Seite 136)
- Indirekte Methode (>>> 5.10.3.2 "Indirekte Methode" Seite 138)

Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.3.3 "Numerische Eingabe" Seite 139)

5.10.3.1 3-Punkt-Methode

Beschreibung

Der Ursprung und 2 weitere Punkte der neuen Basis werden angefahren. Diese 3 Punkte definieren die neue Basis.

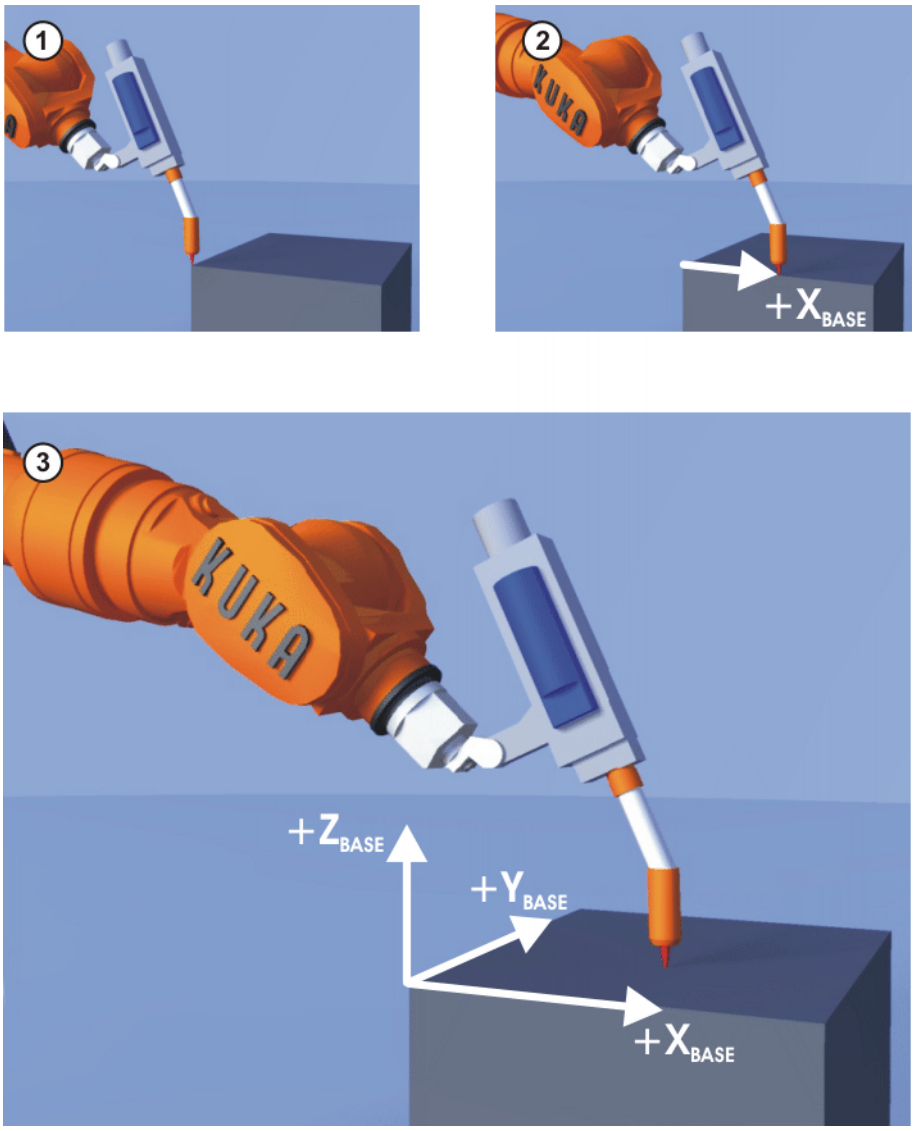


Abb. 5-25: 3-Punkt-Methode

Voraussetzung

- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Basis** > **3-Punkt** wählen.
2. Eine Nummer und einen Namen für die Basis vergeben. Mit **Weiter** bestätigen.
3. Die Nummer des montierten Werkzeugs eingeben. Mit **Weiter** bestätigen.
4. Mit dem TCP den Ursprung der neuen Basis anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Mit dem TCP einen Punkt auf der positiven X-Achse der neuen Basis anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
6. Mit dem TCP auf der XY-Ebene einen Punkt mit positivem Y-Wert anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
7. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
8. **Speichern** drücken.

5.10.3.2 Indirekte Methode

Beschreibung

Die indirekte Methode wird verwendet, wenn der Ursprung der Basis nicht angefahren werden kann, z. B. weil er im Innern eines Werkstücks oder außerhalb des Arbeitsraums des Roboters liegt.

4 Punkte der Basis, deren Koordinaten bekannt sein müssen, werden angefahren. Die Robotersteuerung berechnet die Basis auf Grundlage dieser Punkte.

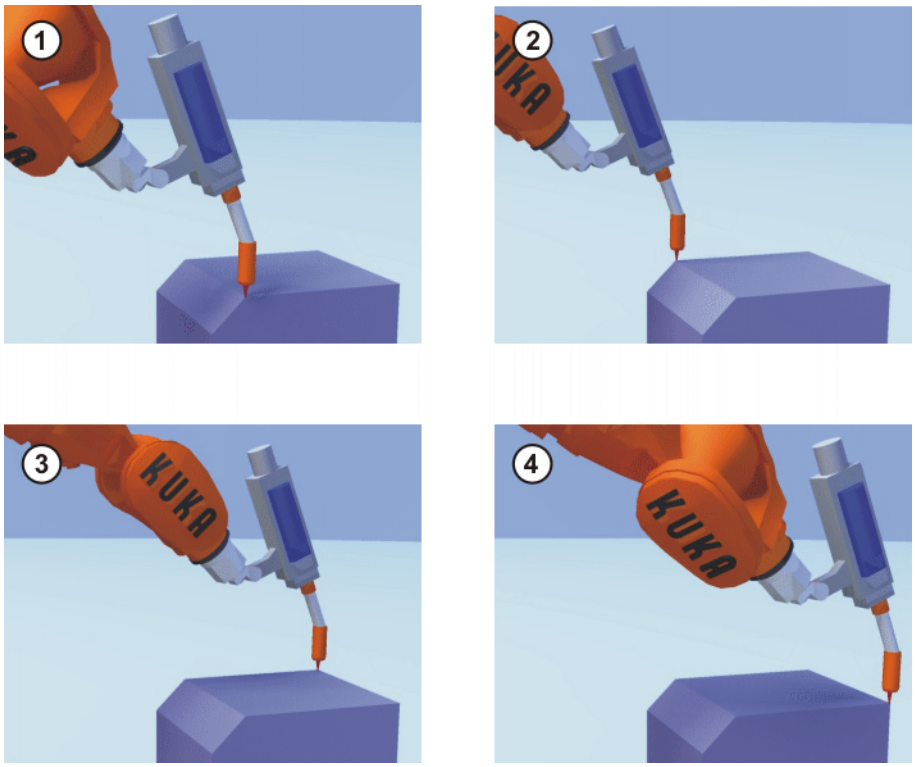


Abb. 5-26: Indirekte Methode

Voraussetzung

- Ein vermessenes Werkzeug ist am Anbauflansch montiert.
- Die Koordinaten von 4 Punkten der neuen Basis sind bekannt, z. B. aus CAD. Die 4 Punkte sind für den TCP erreichbar.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Basis** > **Indirekt** wählen.
2. Eine Nummer und einen Namen für die Basis vergeben. Mit **Weiter** bestätigen.
3. Die Nummer des montierten Werkzeugs eingeben. Mit **Weiter** bestätigen.
4. Die Koordinaten eines bekannten Punktes der neuen Basis eingeben und den Punkt mit dem TCP anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Schritt 4 dreimal wiederholen.
6. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
7. **Speichern** drücken.

5.10.3.3 Numerische Eingabe

Voraussetzung

- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  Entfernung des Ursprungs der Basis vom Ursprung des WORLD-Koordinatensystems
  Verdrehung der Achsen der Basis, bezogen auf das WORLD-Koordinatensystem
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Basis** > **Numerische Eingabe** wählen.
2. Eine Nummer und einen Namen für die Basis vergeben. Mit **Weiter** bestätigen.
3. Daten eingeben. Mit **Weiter** bestätigen.
4. **Speichern** drücken.

5.10.4 Feststehendes Werkzeug vermessen

Übersicht

Die Vermessung eines feststehenden Werkzeugs besteht aus 2 Schritten:

Schritt	Beschreibung
1	**TCP des feststehenden Werkzeugs vermessen** Der TCP eines feststehenden Werkzeugs wird externer TCP genannt. (>>> 5.10.4.1 "Externen TCP vermessen" Seite 139) Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.4.2 "Externen TCP numerisch eingeben" Seite 141)
2	**Werkstück vermessen** Folgende Methoden stehen zur Auswahl: ■ Direkte Methode (>>> 5.10.4.3 "Werkstück vermessen: Direkte Methode" Seite 142) ■ Indirekte Methode (>>> 5.10.4.4 "Werkstück vermessen: Indirekte Methode" Seite 143)

Die Robotersteuerung speichert den externen TCP als BASE-Koordinatensystem und das Werkstück als TOOL-Koordinatensystem. Insgesamt können maximal 32 BASE-Koordinatensysteme und 16 TOOL-Koordinatensysteme gespeichert werden.

5.10.4.1 Externen TCP vermessen

Beschreibung

Zuerst gibt der Benutzer der Robotersteuerung den TCP des feststehenden Werkzeugs bekannt. Dazu wird der TCP mit einem bereits vermessenen Werkzeug angefahren.

Dann wird der Robotersteuerung die Orientierung des Koordinatensystems des feststehenden Werkzeugs bekanntgegeben. Dazu richtet der Benutzer das Koordinatensystem des bereits vermessenen Werkzeugs parallel zum neuen Koordinatensystem aus. Es gibt 2 Varianten:

- **5D**: Der Benutzer gibt der Robotersteuerung die Stoßrichtung des Werkzeugs bekannt. Die Stoßrichtung ist defaultmäßig die X-Achse. Die Orientierung der anderen Achsen wird vom System festgelegt und kann vom Benutzer nicht beeinflusst werden.

  Das System legt die Orientierung der anderen Achsen immer gleich fest. Falls das Werkzeug später ein weiteres Mal vermessen werden muss, z. B. nach einem Crash, reicht es deshalb, die Stoßrichtung erneut festzulegen. Auf die Verdrehung um die Stoßrichtung muss nicht geachtet werden.
- **6D**: Der Benutzer gibt der Robotersteuerung die Orientierungen aller 3 Achsen bekannt.

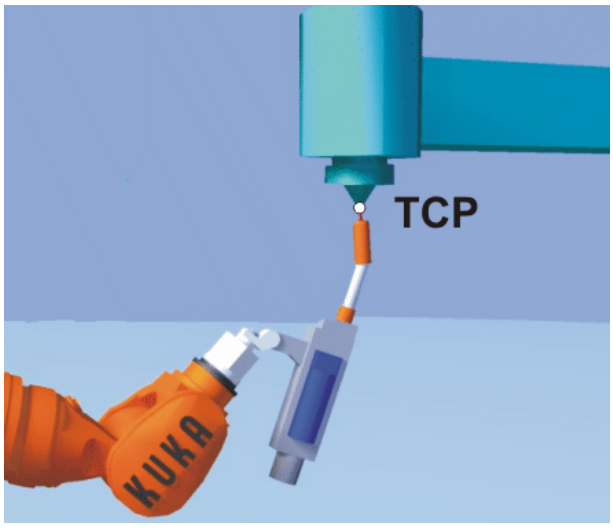


Abb. 5-27: Externen TCP anfahren

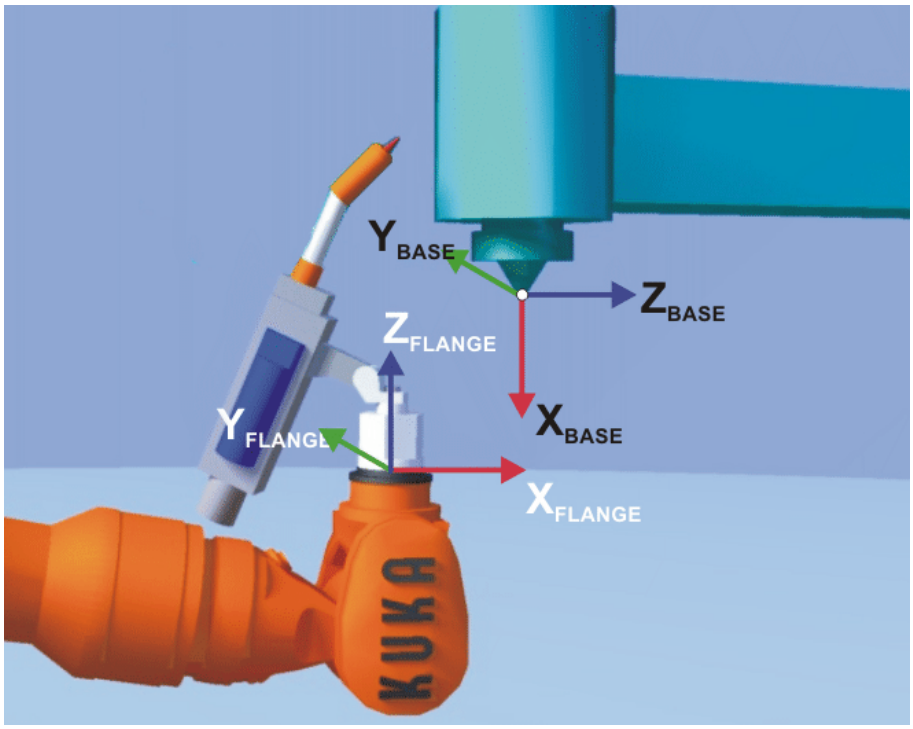


Abb. 5-28: Koordinatensysteme parallel ausrichten

Voraussetzung

- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Betriebsart T1

Die folgende Vorgehensweise ist gültig, wenn die Werkzeug-Stoßrichtung die Default-Stoßrichtung (= X-Richtung) ist. Wenn die Stoßrichtung auf Y oder Z geändert wurde, muss auch die Vorgehensweise entsprechend geändert werden. (>>> 5.10.1 "Werkzeug-Stoßrichtung festlegen" Seite 128)

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Werkzeug** wählen.
2. Eine Nummer und einen Namen für das feststehende Werkzeug vergeben. Mit **Weiter** bestätigen.
3. Nummer des bereits vermessenen Werkzeugs eingeben. Mit **Weiter** bestätigen.
4. Im Feld **5D/6D** eine Variante auswählen. Mit **Weiter** bestätigen.
5. Mit dem TCP des bereits vermessenen Werkzeugs den TCP des feststehenden Werkzeugs anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
6. Wenn **5D** gewählt wurde:

   $+X_{BASE}$ parallel zu $-Z_{FLANGE}$ ausrichten.

   (D. h. den Anbauflansch senkrecht zur Stoßrichtung des feststehenden Werkzeugs ausrichten.)

   Wenn **6D** gewählt wurde:

   Den Anbauflansch so ausrichten, dass seine Achsen parallel zu den Achsen des feststehenden Werkzeugs sind:

   - $+X_{BASE}$ parallel zu $-Z_{FLANGE}$

     (D. h. den Anbauflansch senkrecht zur Stoßrichtung des Werkzeugs ausrichten.)
   - $+Y_{BASE}$ parallel zu $+Y_{FLANGE}$
   - $+Z_{BASE}$ parallel zu $+X_{FLANGE}$
7. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
8. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
9. **Speichern** drücken.

5.10.4.2 Externen TCP numerisch eingeben

Voraussetzung

- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  - Entfernung des TCP des feststehenden Werkzeugs vom Ursprung des WORLD-Koordinatensystems (X, Y, Z)
  - Verdrehung der Achsen des feststehenden Werkzeugs, bezogen auf das WORLD-Koordinatensystem (A, B, C)
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Numerische Eingabe** wählen.
2. Eine Nummer und einen Namen für das feststehende Werkzeug vergeben. Mit **Weiter** bestätigen.
3. Daten eingeben. Mit **Weiter** bestätigen.
4. **Speichern** drücken.

5.10.4.3 Werkstück vermessen: Direkte Methode

Beschreibung

Der Robotersteuerung werden der Ursprung und 2 weitere Punkte des Werkstücks bekanntgegeben. Diese 3 Punkte definieren das Werkstück eindeutig.

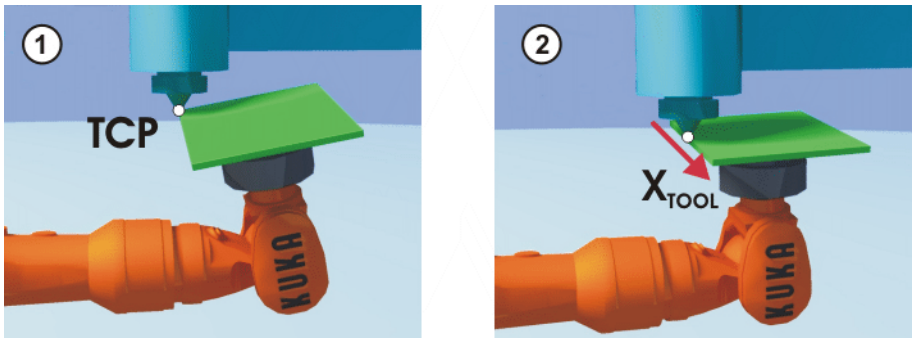


Abb. 5-29

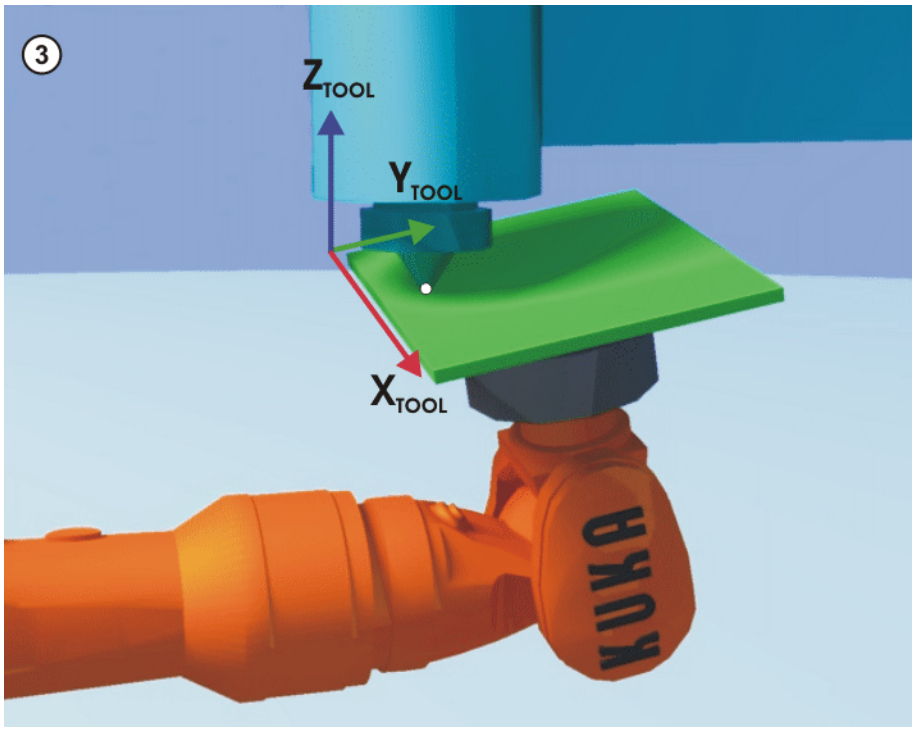


Abb. 5-30: Werkstück vermessen: Direkte Methode

Voraussetzung

- Das Werkstück ist am Anbauflansch montiert.
- Ein bereits vermessenes feststehendes Werkzeug ist montiert.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Werkstück** > **Direkte Vermessung** wählen.
2. Eine Nummer und einen Namen für das Werkstück vergeben. Mit **Weiter** bestätigen.
3. Die Nummer des feststehenden Werkzeugs eingeben. Mit **Weiter** bestätigen.
4. Den Ursprung des Werkstück-Koordinatensystems an den TCP des feststehenden Werkzeugs verfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Einen Punkt auf der positiven X-Achse des Werkstück-Koordinatensystems an den TCP des feststehenden Werkzeugs verfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.

6. Einen Punkt, der auf der XY-Ebene des Werkstück-Koordinatensystems einen positivem Y-Wert hat, an den TCP des feststehenden Werkzeugs verfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
7. Die Lastdaten des Werkstücks eingeben. (Dieser Schritt kann übersprungen werden, wenn die Lastdaten stattdessen gesondert eingegeben werden.)

   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
8. Mit **Weiter** bestätigen.
9. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
10. **Speichern** drücken.

5.10.4.4 Werkstück vermessen: Indirekte Methode

Beschreibung

Die Robotersteuerung berechnet das Werkstück auf Grundlage von 4 Punkten, deren Koordinaten bekannt sein müssen. Der Ursprung des Werkstücks wird nicht angefahren.

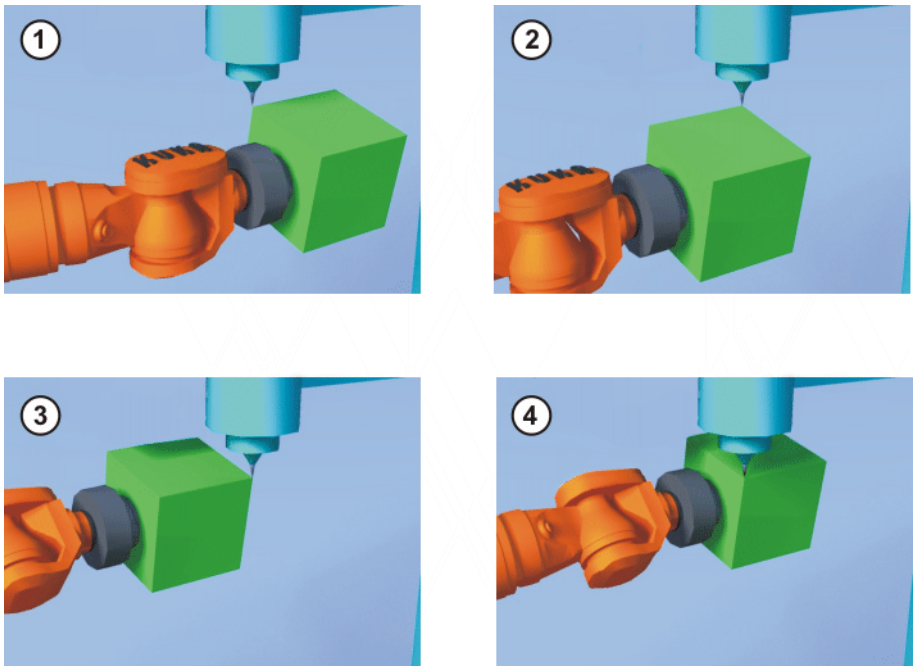


Abb. 5-31: Werkstück vermessen: Indirekte Methode

Voraussetzung

- Ein bereits vermessenes feststehendes Werkzeug ist montiert.
- Das zu vermessende Werkstück ist am Anbauflansch montiert.
- Die Koordinaten von 4 Punkten des neuen Werkstücks sind bekannt, z. B. aus CAD. Die 4 Punkte sind für den TCP erreichbar.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Werkstück** > **Indirekte Vermessung** wählen.
2. Eine Nummer und einen Namen für das Werkstück vergeben. Mit **Weiter** bestätigen.
3. Die Nummer des feststehenden Werkzeugs eingeben. Mit **Weiter** bestätigen.

4. Die Koordinaten eines bekannten Punktes des Werkstücks eingeben und mit diesem Punkt den TCP des feststehenden Werkzeugs anfahren. **Vermessen** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
5. Schritt 4 dreimal wiederholen.
6. Die Lastdaten des Werkstücks eingeben. (Dieser Schritt kann übersprungen werden, wenn die Lastdaten stattdessen gesondert eingegeben werden.)

   (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)
7. Mit **Weiter** bestätigen.
8. Bei Bedarf können Koordinaten und Orientierung der vermessenen Punkte in Inkrementen und Grad angezeigt werden (bezogen auf das FLANGE-Koordinatensystem). Hierzu auf **Meßpunkte** drücken. Danach über **Zurück** zur vorherigen Ansicht zurückkehren.
9. **Speichern** drücken.

5.10.5 Werkzeug/Basis umbenennen

Voraussetzung

- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** oder **Basis** > **Name ändern** wählen.
2. Werkzeug oder Basis markieren und **Name** drücken.
3. Neuen Namen eingeben und mit **Speichern** bestätigen.

5.10.6 Lineareinheit

Die KUKA Lineareinheit ist eine am Boden oder an der Decke montierte eigenständige 1-achsige Lineareinheit. Sie dient dem linearen Verfahren des Roboters und wird von der Robotersteuerung wie eine Zusatzachse gesteuert.

Die Lineareinheit ist eine ROBROOT-Kinematik. Beim Verfahren der Lineareinheit verändert sich die Position des Roboters im WORLD-Koordinatensystem. Die aktuelle Position des Roboters im WORLD-Koordinatensystem beschreibt der Vektor $ROBROOT_C.

$ROBROOT_C setzt sich zusammen aus:

- $ERSYSROOT (statischer Anteil)

  Fußpunkt der Lineareinheit, bezogen auf $WORLD. Der Fußpunkt liegt defaultmäßig in der Nullposition der Lineareinheit und ist abhängig von $MAMES.
- #ERSYS (dynamischer Anteil)

  Aktuelle Position des Roboters auf der Lineareinheit, bezogen auf $ERSYSROOT

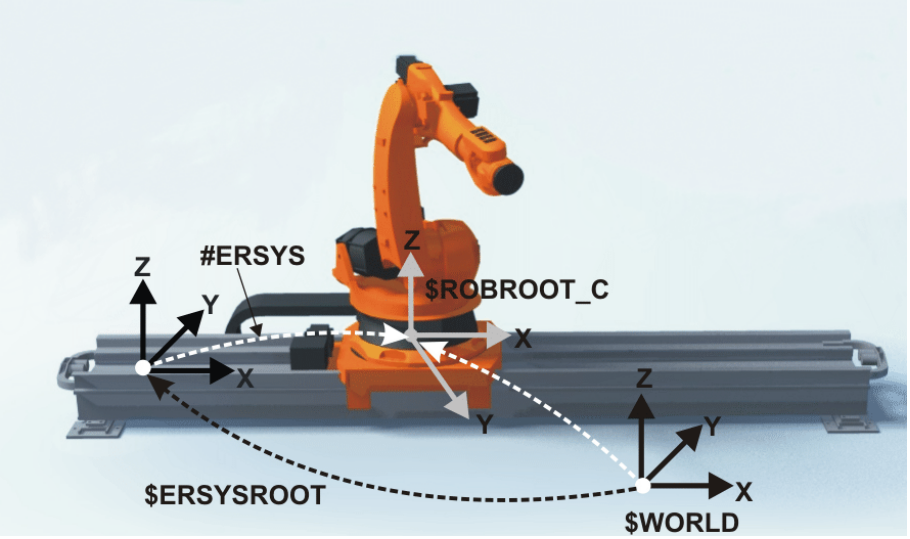


Abb. 5-32: ROBROOT-Kinematik - Lineareinheit

5.10.6.1 Prüfen, ob die Lineareinheit vermessen werden muss

Beschreibung

Der Roboter steht auf dem Flansch der Lineareinheit. Im Idealfall ist das ROBROOT-Koordinatensystem des Roboters deckungsgleich mit dem FLANGE-Koordinatensystem der Lineareinheit. In der Realität gibt es hier oft kleine Abweichungen, die dazu führen, dass Positionen nicht korrekt angefahren werden können. Das Vermessen dient dazu, diese Abweichungen rechnerisch zu korrigieren. (Verdrehungen um die Bewegungsrichtung der Lineareinheit können nicht korrigiert werden. Sie führen jedoch auch nicht zu Fehlern beim Anfahren von Positionen.)

Wenn keine Abweichungen vorhanden sind, muss die Lineareinheit nicht vermessen werden. Mit der folgenden Vorgehensweise kann man ermitteln, ob sie vermessen werden muss.

Voraussetzung

- Die Maschinendaten der Lineareinheit sind konfiguriert und auf die Robotersteuerung gespielt.
- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Es ist kein Programm geöffnet oder angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP auf einen beliebigen Punkt ausrichten und beobachten.
2. Die Lineareinheit kartesisch verfahren. (Nicht achsspezifisch!)
   - Wenn der TCP stehenbleibt: Die Linearachse muss nicht vermessen werden.
   - Wenn sich der TCP bewegt: Die Linearachse muss vermessen werden.

     (>>> 5.10.6.2 "Lineareinheit vermessen" Seite 145)

Wenn die Vermessungsdaten bereits bekannt sind (z. B. aus CAD), können sie direkt eingegeben werden. (>>> 5.10.6.3 "Lineareinheit numerisch eingeben" Seite 146)

5.10.6.2 Lineareinheit vermessen

Beschreibung

Bei der Vermessung wird ein Referenzpunkt mit dem TCP eines bereits vermessenen Werkzeugs 3-mal angefahren.

- Der Referenzpunkt kann beliebig gewählt werden.

- Die Position des Roboters auf der Lineareinheit, von der aus der Referenzpunkt angefahren wird, muss 3-mal unterschiedlich sein. Die 3 Positionen müssen ausreichend weit auseinanderliegen.

Die Korrekturwerte, die durch die Vermessung ermittelt werden, fließen in die Systemvariable $ET*x*_TFLA3 ein.

Voraussetzung

- Die Maschinendaten der Lineareinheit sind konfiguriert und auf die Robotersteuerung gespielt.
- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Es ist kein Programm geöffnet oder angewählt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Lineareinheit** wählen.
   Die Robotersteuerung detektiert die Lineareinheit automatisch und zeigt folgende Daten an:
   - **Ext. Kinematik Nr.**: Nummer der Externen Kinematik (1 … 6) ($EX_KIN)
   - **Achse**: Nummer der Zusatzachse (1 … 6) ($ET*x*_AX)
   - **Name der ext. Kinematik** ($ET*x*_NAME)

   (Wenn die Robotersteuerung diese Werte nicht ermitteln kann, z. B. weil die Lineareinheit noch nicht konfiguriert ist, kann die Vermessung nicht fortgesetzt werden.)
2. Lineareinheit mit der Verfahrtaste "+" verfahren.
3. Angeben, ob die Lineareinheit nach "+" oder nach "-" gefahren ist. Mit **Weiter** bestätigen.
4. Mit dem TCP den Referenzpunkt anfahren.
5. **Vermessen** drücken.
6. Schritt 4 und 5 zweimal wiederholen, jedoch jedes Mal vorher die Lineareinheit verfahren, um den Referenzpunkt aus verschiedenen Position anzufahren.
7. **Speichern** drücken. Die Vermessungsdaten werden gespeichert.
8. Es wird eine Abfrage angezeigt, ob bereits geteachte Positionen korrigiert werden sollen.
   - Wenn vor dem Vermessen noch keine Positionen geteacht worden sind, ist es gleichgültig, ob die Abfrage mit **Ja** oder **Nein** beantwortet wird.
   - Wenn vor dem Vermessen Positionen geteacht worden sind:

     Wenn die Abfrage mit **Ja** beantwortet wird, werden Positionen mit Basis 0 automatisch korrigiert. Andere Positionen werden nicht korrigiert!

     Wenn die Abfrage mit **Nein** beantwortet wird, werden keine Positionen korrigiert.

HINWEIS Nach dem Vermessen einer Lineareinheit müssen folgende Sicherheitsmaßnahmen durchgeführt werden:

1. Software-Endschalter der Lineareinheit prüfen und gegebenenfalls anpassen.
2. Programme in T1 testen.

Sachschäden können sonst die Folge sein.

5.10.6.3 Lineareinheit numerisch eingeben

Voraussetzung

- Die Maschinendaten der Lineareinheit sind konfiguriert und auf die Robotersteuerung gespielt.

- Es ist kein Programm geöffnet oder angewählt.
- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  - Entfernung des Roboterfuß-Flansches vom Ursprung des ERSYS-ROOT-Koordinatensystems (X, Y, Z)
  - Orientierung des Roboterfuß-Flansches bezogen auf das ERSYS-ROOT-Koordinatensystem (A, B, C)
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Lineareinheit (numerisch)** wählen.
   Die Robotersteuerung detektiert die Lineareinheit automatisch und zeigt folgende Daten an:
   - **Ext. Kinematik Nr.**: Nummer der Externen Kinematik (1 … 6)
   - **Achse**: Nummer der Zusatzachse (1 … 6)
   - **Name der ext. Kinematik**

   (Wenn die Robotersteuerung diese Werte nicht ermitteln kann, z. B. weil die Lineareinheit noch nicht konfiguriert ist, kann die Vermessung nicht fortgesetzt werden.)
2. Lineareinheit mit der Verfahrtaste "+" verfahren.
3. Angeben, ob die Lineareinheit nach "+" oder nach "-" gefahren ist. Mit **Weiter** bestätigen.
4. Daten eingeben. Mit **Weiter** bestätigen.
5. **Speichern** drücken. Die Vermessungsdaten werden gespeichert.
6. Es wird eine Abfrage angezeigt, ob bereits geteachte Positionen korrigiert werden sollen.
   - Wenn vor dem Vermessen noch keine Positionen geteacht worden sind, ist es gleichgültig, ob die Abfrage mit **Ja** oder **Nein** beantwortet wird.
   - Wenn vor dem Vermessen Positionen geteacht worden sind:
     Wenn die Abfrage mit **Ja** beantwortet wird, werden Positionen mit Basis 0 automatisch korrigiert. Andere Positionen werden nicht korrigiert!
     Wenn die Abfrage mit **Nein** beantwortet wird, werden keine Positionen korrigiert.

HINWEIS Nach dem Vermessen einer Lineareinheit müssen folgende Sicherheitsmaßnahmen durchgeführt werden:

1. Software-Endschalter der Lineareinheit prüfen und gegebenenfalls anpassen.
2. Programme in T1 testen.

Sachschäden können sonst die Folge sein.

5.10.7 Externe Kinematik vermessen

Beschreibung

Die Vermessung der externen Kinematik ist notwendig, damit die Achsen der Kinematik synchron und mathematisch gekoppelt mit den Roboterachsen bewegt werden können. Eine externe Kinematik kann z. B. ein Drehkipptisch oder ein Positionierer sein.

Für Lineareinheiten darf die hier beschriebene Vermessung nicht verwendet werden. Für Lineareinheiten muss eine eigene Art der Vermessung verwendet werden.
(>>> 5.10.6 "Lineareinheit" Seite 144)

Übersicht

Die Vermessung einer externen Kinematik besteht aus 2 Schritten:

Schritt	Beschreibung
1	Fußpunkt der externen Kinematik vermessen. (>>> 5.10.7.1 "Fußpunkt vermessen" Seite 148) Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.7.2 "Fußpunkt numerisch eingeben" Seite 149)
2	Wenn sich auf der externen Kinematik ein Werkstück befindet: Basis des Werkstücks vermessen. (>>> 5.10.7.3 "Werkstück-Basis vermessen" Seite 150) Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.7.4 "Werkstück-Basis numerisch eingeben" Seite 152)
	Wenn auf der externen Kinematik ein Werkzeug montiert ist: Externes Werkzeug vermessen. (>>> 5.10.7.5 "Externes Werkzeug vermessen" Seite 152) Wenn die Vermessungsdaten bereits bekannt sind, können sie direkt eingegeben werden. (>>> 5.10.7.6 "Externes Werkzeug numerisch eingeben" Seite 154)

5.10.7.1 Fußpunkt vermessen

Beschreibung

Um den Roboter mathematisch gekoppelt mit einer Kinematik verfahren zu können, muss der Roboter den genauen Standort der Kinematik kennen. Dieser Standort wird mit der Fußpunkt-Vermessung bekanntgegeben.

Mit dem TCP eines bereits vermessenen Werkzeugs wird ein Referenzpunkt auf der Kinematik 4-mal angefahren. Die Position des Referenzpunktes muss dabei jedesmal verschieden sein. Dies wird erreicht, indem die Achsen der Kinematik verfahren werden. Aus den unterschiedlichen Positionen des Referenzpunkts berechnet die Robotersteuerung den Fußpunkt der Kinematik.

Bei externen Kinematiken von KUKA ist der Referenzpunkt in den Maschinendaten in der Systemvariable $ET*x*_TPINFL konfiguriert. Diese enthält die Lage des Referenzpunkts relativ zum FLANGE-Koordinatensystem der Kinematik. (*x* = Nummer der Kinematik.) Der Referenzpunkt ist außerdem auf der Kinematik markiert. Beim Vermessen muss dieser Referenzpunkt angefahren werden.

Bei externen Kinematiken, die nicht von KUKA stammen, muss der Referenzpunkt in den Maschinendaten konfiguriert werden.

Die Robotersteuerung speichert die Koordinaten des Fußpunkts als BASE-Koordinatensystem.

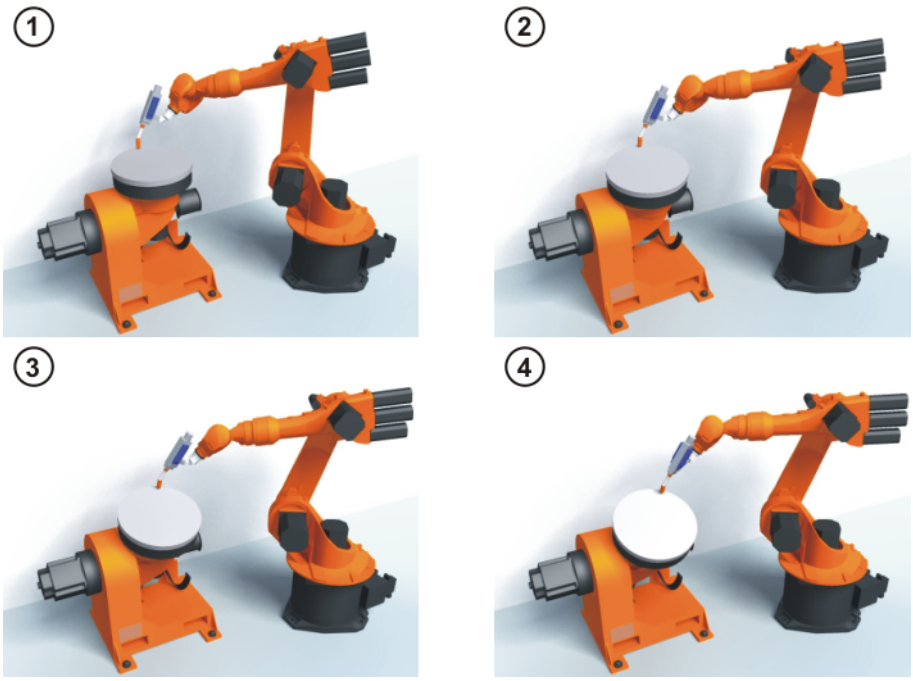


Abb. 5-33: Prinzip der Fußpunkt-Vermessung

Voraussetzung

- Die Maschinendaten der Kinematik sind konfiguriert und auf die Robotersteuerung gespielt.
- Die Nummer der externen Kinematik ist bekannt.
- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Falls $ET*x*_TPINFL geändert werden soll: Benutzergruppe Experte
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Fußpunkt** wählen.
2. Die Nummer des BASE-Koordinatensystems wählen, als das der Fußpunkt gespeichert werden soll. Mit **Weiter** bestätigen.
3. Die Nummer der externen Kinematik eingeben.
4. Einen Namen für die externe Kinematik vergeben. Mit **Weiter** bestätigen.
5. Die Nummer des Referenzwerkzeugs eingeben. Mit **Weiter** bestätigen.
6. Der Wert von $ET*x*_TPINFL wird angezeigt.
   - Wenn der Wert nicht korrekt ist: In der Benutzergruppe Experte kann der Wert hier geändert werden.
   - Wenn der Wert korrekt ist: Mit **Weiter** bestätigen.
7. Mit dem TCP den Referenzpunkt anfahren.
8. **Vermessen** drücken. Mit **Weiter** bestätigen.
9. Schritt 7 und 8 dreimal wiederholen. Jedes Mal vorher die Kinematik verfahren, um den Referenzpunkt aus verschiedenen Position anzufahren.
10. **Speichern** drücken.

5.10.7.2 Fußpunkt numerisch eingeben

Voraussetzung

- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  - Entfernung des Ursprungs des ROOT-Koordinatensystems vom Ursprung des WORLD-Koordinatensystems (X, Y, Z)
  - Orientierung des ROOT-Koordinatensystems, bezogen auf das WORLD-Koordinatensystem (A, B, C)

- Die Nummer der externen Kinematik ist bekannt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Fußpunkt (numerisch)** wählen.
2. Die Nummer des BASE-Koordinatensystems wählen, als das der Fußpunkt gespeichert werden soll. Mit **Weiter** bestätigen.
3. Die Nummer der externen Kinematik eingeben.
4. Einen Namen für die externe Kinematik vergeben. Mit **Weiter** bestätigen.
   (Der Name wird automatisch auch dem BASE-Koordinatensystem zugewiesen.)
5. Die Daten des ROOT-Koordinatensystems eingeben. Mit **Weiter** bestätigen.
6. **Speichern** drücken.

5.10.7.3 Werkstück-Basis vermessen

Beschreibung

Bei dieser Vermessung weist der Benutzer einem Werkstück, das sich auf der Kinematik befindet, ein BASE-Koordinatensystem zu. Dieses BASE-Koordinatensystem bezieht sich auf das FLANGE-Koordinatensystem der Kinematik. Die Basis ist somit eine bewegliche Basis und bewegt sich in derselben Art wie die Kinematik.

Es ist nicht zwingend erforderlich, eine Basis zu vermessen. Wenn keine vermessen wird, gilt das FLANGE-Koordinatensystem der Kinematik als Basis.

Bei der Vermessung werden mit dem TCP eines bereits vermessenen Werkzeugs der Ursprung und 2 weitere Punkte der gewünschten Basis angefahren. Diese 3 Punkte definieren die Basis. Pro Kinematik kann nur eine Basis vermessen werden.

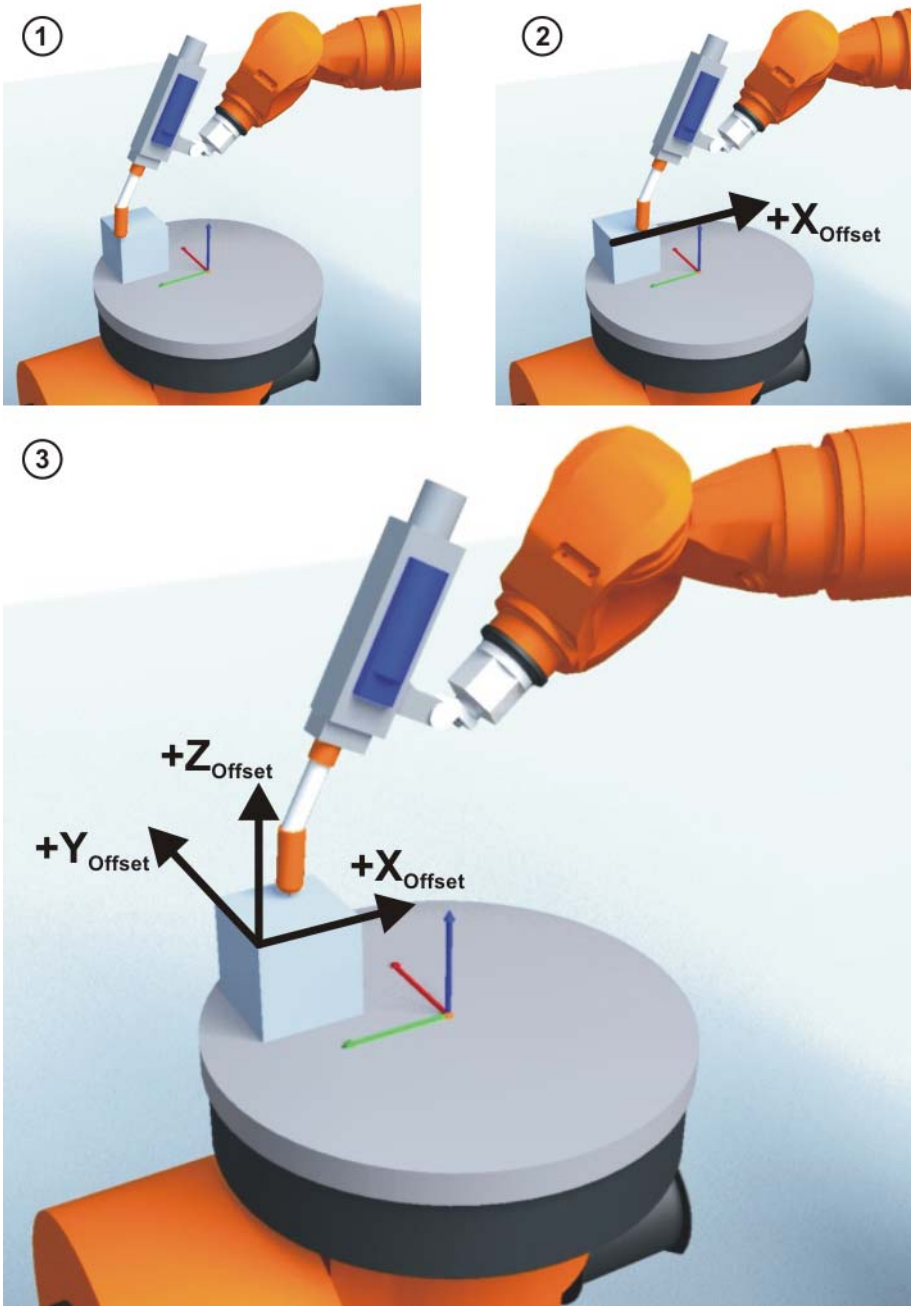


Abb. 5-34: Prinzip der Basis-Vermessung

Voraussetzung

- Die Maschinendaten der Kinematik sind konfiguriert und auf die Robotersteuerung gespielt.
- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Der Fußpunkt der externen Kinematik ist vermessen.
- Die Nummer der externen Kinematik ist bekannt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Offset** wählen.
2. Die Nummer des BASE-Koordinatensystems eingeben, als das der Fußpunkt gespeichert wurde. Der Name des BASE-Koordinatensystems wird angezeigt.

   Mit **Weiter** bestätigen.
3. Die Nummer der externen Kinematik eingeben. Der Name der externen Kinematik wird angezeigt.

Mit **Weiter** bestätigen.

4. Die Nummer des Referenzwerkzeugs eingeben. Mit **Weiter** bestätigen.
5. Mit dem TCP den Ursprung der Werkstück-Basis anfahren. **Vermessen** drücken und mit **Weiter** bestätigen.
6. Mit dem TCP einen Punkt auf der positiven X-Achse der Werkstück-Basis anfahren. **Vermessen** drücken und mit **Weiter** bestätigen.
7. Mit dem TCP auf der XY-Ebene einen Punkt mit positivem Y-Wert anfahren. **Vermessen** drücken und mit **Weiter** bestätigen.
8. **Speichern** drücken.

5.10.7.4 Werkstück-Basis numerisch eingeben

Voraussetzung

- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  - Entfernung des Ursprungs der Werkstück-Basis vom Ursprung des FLANGE-Koordinatensystems der Kinematik (X, Y, Z)
  - Verdrehung der Achsen der Werkstück-Basis, bezogen auf das FLANGE-Koordinatensystem der Kinematik (A, B, C)
- Der Fußpunkt der externen Kinematik ist vermessen.
- Die Nummer der externen Kinematik ist bekannt.
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Externe Kinematik** > **Offset (numerisch)** wählen.
2. Die Nummer des BASE-Koordinatensystems eingeben, als das der Fußpunkt gespeichert wurde. Der Name des BASE-Koordinatensystems wird angezeigt.

   Mit **Weiter** bestätigen.
3. Die Nummer der externen Kinematik eingeben. Der Name der externen Kinematik wird angezeigt.

   Mit **Weiter** bestätigen.
4. Daten eingeben. Mit **Weiter** bestätigen.
5. **Speichern** drücken.

5.10.7.5 Externes Werkzeug vermessen

Beschreibung

Bei der Vermessung des externen Werkzeugs weist der Benutzer einem Werkzeug, das auf der Kinematik angebracht ist, ein Koordinatensystem zu. Dieses Koordinatensystem hat seinen Ursprung im TCP des externen Werkzeugs und bezieht sich auf das FLANGE-Koordinatensystem der Kinematik.

Zuerst gibt der Benutzer der Robotersteuerung den TCP des Werkzeugs, das auf der Kinematik angebracht ist, bekannt. Dazu wird der TCP mit einem bereits vermessenen Werkzeug angefahren.

Dann wird der Robotersteuerung die Orientierung des Koordinatensystems des Werkzeugs bekanntgegeben. Dazu richtet der Benutzer das Koordinatensystem des bereits vermessenen Werkzeugs parallel zum neuen Koordinatensystem aus. Es gibt 2 Varianten:

- **5D**: Der Benutzer gibt der Robotersteuerung die Stoßrichtung des Werkzeugs bekannt. Die Stoßrichtung ist defaultmäßig die X-Achse. Die Orientierung der anderen Achsen wird vom System festgelegt und kann vom Benutzer nicht beeinflusst werden.

  Das System legt die Orientierung der anderen Achsen immer gleich fest. Falls das Werkzeug später ein weiteres Mal vermessen werden muss, z. B. nach einem Crash, reicht es deshalb, die Stoßrichtung erneut festzule-

gen. Auf die Verdrehung um die Stoßrichtung muss nicht geachtet werden.

- **6D**: Der Benutzer gibt der Robotersteuerung die Richtung aller 3 Achsen bekannt.

> Wenn **6D** verwendet wird: Es empfiehlt sich, die Ausrichtung aller Achsen zu dokumentieren. Falls das Werkzeug später ein weiteres Mal vermessen werden muss, z. B. nach einem Crash, müssen die Achsen wie beim ersten Mal ausgerichtet werden, um bestehende Punkte weiterhin korrekt anfahren zu können.

Die Robotersteuerung speichert die Koordinaten des externen Werkzeugs als BASE-Koordinatensystem.

Voraussetzung

- Die Maschinendaten der Kinematik sind konfiguriert und auf die Robotersteuerung gespielt.
- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Der Fußpunkt der externen Kinematik ist vermessen.
- Die Nummer der externen Kinematik ist bekannt.
- Betriebsart T1

> Die folgende Vorgehensweise ist gültig, wenn die Werkzeug-Stoßrichtung die Default-Stoßrichtung (= X-Richtung) ist. Wenn die Stoßrichtung auf Y oder Z geändert wurde, muss auch die Vorgehensweise entsprechend geändert werden. (>>> 5.10.1 "Werkzeug-Stoßrichtung festlegen" Seite 128)

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Offset externer Kinematik** wählen.
2. Die Nummer des BASE-Koordinatensystems eingeben, als das der Fußpunkt gespeichert wurde. Der Name des BASE-Koordinatensystems wird angezeigt.
   Mit **Weiter** bestätigen.
3. Die Nummer der externen Kinematik eingeben. Der Name der externen Kinematik wird angezeigt.
   Mit **Weiter** bestätigen.
4. Die Nummer des Referenzwerkzeugs eingeben. Mit **Weiter** bestätigen.
5. Im Feld **5D/6D** eine Variante auswählen. Mit **Weiter** bestätigen.
6. Mit dem TCP des bereits vermessenen Werkzeugs den TCP des externen Werkzeugs anfahren. **Vermessen** drücken und mit **Weiter** bestätigen.
7. Wenn **5D** gewählt wurde:
   $+X_{BASE}$ parallel zu $-Z_{FLANGE}$ ausrichten.
   (D. h. den Anbauflansch senkrecht zur Stoßrichtung des externen Werkzeugs ausrichten.)
   Wenn **6D** gewählt wurde:
   Den Anbauflansch so ausrichten, dass seine Achsen parallel zu den Achsen des externen Werkzeugs sind:
   - $+X_{BASE}$ parallel zu $-Z_{FLANGE}$
     (D. h. den Anbauflansch senkrecht zur Stoßrichtung des externen Werkzeugs ausrichten.)
   - $+Y_{BASE}$ parallel zu $+Y_{FLANGE}$
   - $+Z_{BASE}$ parallel zu $+X_{FLANGE}$
8. **Vermessen** drücken und mit **Weiter** bestätigen.
9. **Speichern** drücken.

5.10.7.6 Externes Werkzeug numerisch eingeben

Voraussetzung

- Folgende numerische Werte sind bekannt, z. B. aus CAD:
  - Entfernung des TCP des externen Werkzeugs vom Ursprung des FLANGE-Koordinatensystems der Kinematik (X, Y, Z)
  - Verdrehung der Achsen des externen Werkzeugs, bezogen auf das FLANGE-Koordinatensystem der Kinematik (A, B, C)
- Betriebsart T1

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Feststehendes Werkzeug** > **Numerische Eingabe** wählen.
2. Eine Nummer und einen Namen für das externe Werkzeug vergeben. Mit **Weiter** bestätigen.
3. Daten eingeben. Mit **Weiter** bestätigen.
4. **Speichern** drücken.

5.11 Lastdaten

Die Lastdaten fließen in die Berechnung der Bahnen und Beschleunigungen ein und tragen zur Optimierung der Taktzeiten bei. Die Lastdaten müssen in die Robotersteuerung eingegeben werden.

Quellen

Lastdaten können folgenden Quellen entnommen werden:

- Software-Option KUKA.LoadDataDetermination (nur für Traglasten am Flansch)
- Herstellerangaben
- Manuelle Berechnung
- CAD-Programme

5.11.1 Lasten prüfen mit KUKA.Load

Alle Lastdaten (Traglast und Zusatzlasten) müssen mit der Software KUKA.Load geprüft werden. Ausnahme: Wenn die Traglast mit KUKA.LoadDataDetermination geprüft wird, ist eine Prüfung mit KUKA.Load nicht notwendig.

Mit KUKA.Load kann ein Abnahmeprotokoll (Sign Off Sheet) für die Lasten erzeugt werden. KUKA.Load kann inklusive Dokumentation kostenlos von der KUKA Website www.kuka.com heruntergeladen werden.

Weitere Informationen sind in der Dokumentation **KUKA.Load** zu finden.

5.11.2 Traglasten ermitteln mit KUKA.LoadDataDetermination

Beschreibung

Mit KUKA.LoadDataDetermination können Traglasten exakt bestimmt werden und in die Robotersteuerung übertragen werden.

Voraussetzung

- Betriebsart T1 oder T2
- Kein Programm ist angewählt.

Vorgehensweise

- Im Hauptmenü **Inbetriebnahme** > **Service** > **Lastdatenermittlung** wählen.

Weitere Informationen sind in der Dokumentation **KUKA.LoadDataDetermination** zu finden.

5.11.3 Traglastdaten eingeben

Beschreibung

Die Traglastdaten müssen in die Robotersteuerung eingegeben werden und dem richtigen Werkzeug zugewiesen werden.

Ausnahme: Wenn die Traglastdaten bereits mit KUKA.LoadDataDetermination in die Robotersteuerung übertragen wurden, ist keine manuelle Eingabe mehr notwendig.

Voraussetzung

- Die Traglastdaten wurden mit KUKA.Load oder KUKA.LoadDataDetermination geprüft und der Roboter ist für diese Traglasten geeignet.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Werkzeug** > **Werkzeuglastdaten** wählen.
2. Im Feld **Werkzeug Nr.** die Nummer des Werkzeugs eingeben. Mit **Weiter** bestätigen.
3. Die Traglastdaten eingeben:
   - Feld **M**: Masse
   - Felder **X**, **Y**, **Z**: Lage des Schwerpunkts relativ zum Flansch
   - Felder **A**, **B**, **C**: Orientierung der Haupt-Trägheitsachsen relativ zum Flansch
   - Felder **JX**, **JY**, **JZ**: Massen-Trägheitsmomente

     (JX ist die Trägheit um die X-Achse des Koordinatensystems, das durch A, B und C relativ zum Flansch verdreht ist. JY und JZ analog die Trägheiten um die Y- und Z-Achse.)

   Oder, wenn die Default-Werte für diesen Robotertyp verwendet werden sollen: Auf **Default** drücken.
4. Wenn die Online-Lastdatenprüfung zur Verfügung steht (dies ist abhängig vom Robotertyp): Nach Bedarf konfigurieren.
   (>>> 5.11.5 "Online-Lastdatenprüfung (OLDC)" Seite 156)
5. Mit **Weiter** bestätigen.
6. **Speichern** drücken.

5.11.4 Zusatzlastdaten eingeben

Beschreibung

Die Zusatzlastdaten müssen in die Robotersteuerung eingegeben werden.

Bezugssysteme der X-, Y-, Z-Werte je Zusatzlast:

Last	Bezugssystem
Zusatzlast A1	ROBROOT-Koordinatensystem A1 = 0°
Zusatzlast A2	ROBROOT-Koordinatensystem A2 = -90°
Zusatzlast A3	FLANGE-Koordinatensystem A4 = 0°, A5 = 0°, A6 = 0°

Voraussetzung

- Die Zusatzlastdaten wurden mit KUKA.Load geprüft und sind für diesen Robotertyp geeignet.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Zusatzlastdaten** wählen.
2. Die Nummer der Achse eingeben, an der die Zusatzlast befestigt wird. Mit **Weiter** bestätigen.
3. Die Lastdaten eingeben. Mit **Weiter** bestätigen.

4. **Speichern** drücken.

5.11.5 Online-Lastdatenprüfung (OLDC)

Beschreibung

Bei zahlreichen Robotertypen überwacht die Robotersteuerung während des Betriebs, ob eine Über- oder Unterlast vorliegt. Diese Überwachung heißt "Online-Lastdatenprüfung" (= "OLDC" / "Online Load Data Check").

Wenn die OLDC z. B. eine Unterlast feststellt, zeigt die Robotersteuerung als Reaktion darauf z. B. eine Meldung an. Die Reaktionen sind konfigurierbar.

Das Ergebnis der Prüfung kann auch über die Systemvariable $LDC_RESULT abgefragt werden. (>>> "$LDC_RESULT" Seite 158)

Die OLDC steht für die Robotertypen zur Verfügung, für die auch KUKA.LoadDataDetermination verwendet werden kann. Ob die OLDC für den aktuellen Robotertyp zur Verfügung steht, kann über $LDC_LOADED abgefragt werden (TRUE = ja).

Überlast	Eine Überlast liegt vor, wenn die tatsächliche Last höher ist als die konfigurierte Last.
Unterlast	Eine Unterlast liegt vor, wenn die tatsächliche Last geringer ist als die konfigurierte Last.

Konfiguration

Die OLDC kann an folgenden Stellen konfiguriert werden:

- Bei der manuellen Eingabe der Werkzeugdaten
  (>>> 5.10.2.5 "Numerische Eingabe" Seite 135)
- Bei der gesonderten Eingabe der Traglastdaten
  (>>> 5.11.3 "Traglastdaten eingeben" Seite 155)

In demselben Fenster, in dem auch die Traglastdaten eingegeben werden, werden folgende Felder angezeigt:

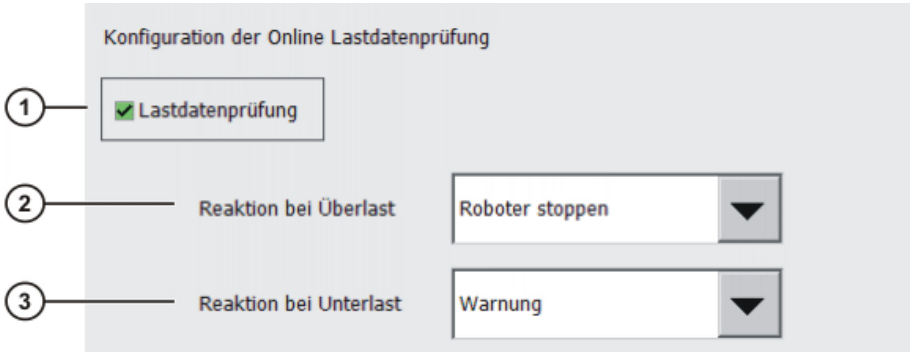


Abb. 5-35: Online-Lastdatenprüfung

Pos.	Beschreibung
1	**TRUE**: Die OLDC für das im gleichen Fenster angezeigte Werkzeug ist aktiv. Bei Über- oder Unterlast erfolgen die definierten Reaktionen. **FALSE**: Die OLDC für das im gleichen Fenster angezeigte Werkzeug ist inaktiv. Bei Über- oder Unterlast erfolgt keine Reaktion.
2	Hier kann definiert werden, welche Reaktion bei Überlast erfolgen soll. ■ **Keine**: Keine Reaktion. ■ **Warnung**: Die Robotersteuerung gibt folgende Zustandsmeldung aus: *Beim Überprüfen der Roboterlast (Tool {Nr.}) ist Überlast ermittelt worden.* ■ **Roboter stoppen**: Die Robotersteuerung gibt eine Quittiermeldung aus mit dem gleichen Inhalt wie bei **Warnung**. Der Roboter stoppt mit einem STOP 2.
3	Hier kann definiert werden, welche Reaktion bei Unterlast erfolgen soll. Die möglichen Reaktionen sind analog zu denen bei Überlast.

Die Reaktionen können im KRL-Programm über die Systemvariable $LDC_CONFIG geändert werden. (>>> "$LDC_CONFIG" Seite 157)

NULLFRAME

Für Bewegungen, denen als Werkzeug **NULLFRAME** zugeordnet ist, kann die OLDC nicht konfiguriert werden. Die Reaktionen für diesen Fall sind festgelegt und können vom Benutzer nicht beeinflusst werden.

- Reaktion bei Überlast: **Roboter stoppen**

  Folgende Quittiermeldung wird ausgegeben: *Beim Überprüfen der Roboterlast (kein Tool definiert) und den gesetzten Lastdaten ist Überlast ermittelt worden.* Der Roboter stoppt mit einem STOP 2.
- Reaktion bei Unterlast: **Warnung**

  Folgende Zustandsmeldung wird ausgegeben: *Beim Überprüfen der Roboterlast (kein Tool definiert) und den gesetzten Lastdaten ist Unterlast ermittelt worden.*

$LDC_CONFIG

`$LDC_CONFIG[`*Index*`]= {UNDERLOAD` *Reaktion*`, OVERLOAD` *Reaktion*`}`

Element	Beschreibung
Index	Typ: INT Werkzeugnummer ■ **1 … 32**
Reaktion	Typ: CHAR ■ #NONE (= **Keine**) ■ #WARNONLY (= **Warnung**) ■ #STOPROBOT (= **Roboter stoppen**)

Beispiel:

```
...
$LDC_CONFIG[1]={UNDERLOAD #NONE, OVERLOAD #NONE}
...
$LDC_CONFIG[1]={UNDERLOAD #WARNONLY, OVERLOAD #WARNONLY}
...
```

Zeile	Beschreibung
2	Die Reaktion sowohl für Unter- als auch für Überlast wird auf **Keine** gesetzt.
4	Die Reaktion wird auf **Warnung** gesetzt. Falls eine Unter- oder Überlast vorliegt, werden im Meldungsfenster die entsprechenden Zustandmeldungen angezeigt.

$LDC_RESULT

`$LDC_RESULT[`*Index*`]=` *Ergebnis*

Element	Beschreibung
Index	Typ: INT Werkzeugnummer ■ **1 … 32**
Ergebnis	Typ: CHAR ■ #OK: Die Traglast ist in Ordnung. (Weder Über- noch Unterlast.) ■ #OVERLOAD: Es liegt eine Überlast vor. ■ #UNDERLOAD: Es liegt eine Unterlast vor. ■ #CHECKING ■ #NONE: Es liegt aktuell kein Ergebnis vor, z. B. weil das Werkzeug gewechselt worden ist.

5.12 Langtexte exportieren/importieren

Beschreibung

Wenn Ein-/Ausgängen, Flags usw. Namen zugeordnet wurden, können diese Namen (die sogenannten "Langtexte") in eine Datei exportiert werden. Ebenso ist es möglich, eine Datei mit Langtext-Namen zu importieren. Auf diese Weise müssen die Langtexte nach einer Neuinstallation nicht bei jedem Roboter von Hand eingegeben werden.

Die Langtexte können auf einen USB-Stick exportiert werden oder in das Verzeichnis, das im Fenster **Roboterdaten** im Feld **Netzwerkarchivpfad** festgelegt ist. Die gleichen Verzeichnisse stehen auch als Quellen für den Import zur Verfügung.

Voraussetzung

- Entweder: USB-Stick
- Oder: Das Ziel ist im Fenster **Roboterdaten** im Feld **Netzwerkarchivpfad** konfiguriert.

Nur für den Import:

- Die Langtext-Namen liegen in einer TXT- oder CSV-Datei vor.
- Die Datei ist so aufgebaut, dass sie importiert werden kann.

Eine Datei, die bereits aus einem Langtext-Export stammt, ist automatisch so aufgebaut, dass sie auch wieder importiert werden kann. Wenn eine Datei manuell mit Namen befüllt werden soll, wird empfohlen, zuerst in der Robotersteuerung wenige Dummy-Langtexte zu vergeben, dann einen Export durchzuführen und die exportierte Datei zu befüllen.

Vorgehensweise

1. Wenn ein USB-Stick verwendet wird, diesen am Schrank oder am smartPAD anstecken.
2. Im Hauptmenü **Inbetriebnahme** > **Service** > **Langtexte** wählen. Das Fenster **Langtexte** öffnet sich.
3. Nach Bedarf die Registerkarte **Exportieren** oder **Importieren** wählen. Die benötigten Einstellungen vornehmen.
4. Auf die Schaltfläche **Exportieren** bzw. **Importieren** drücken.

Wenn der Import abgeschlossen ist, wird die Meldung *Import erfolgreich.* angezeigt.

Wenn der Export abgeschlossen ist, wird die Meldung *Export erfolgreich.* angezeigt.

Registerkarte Exportieren

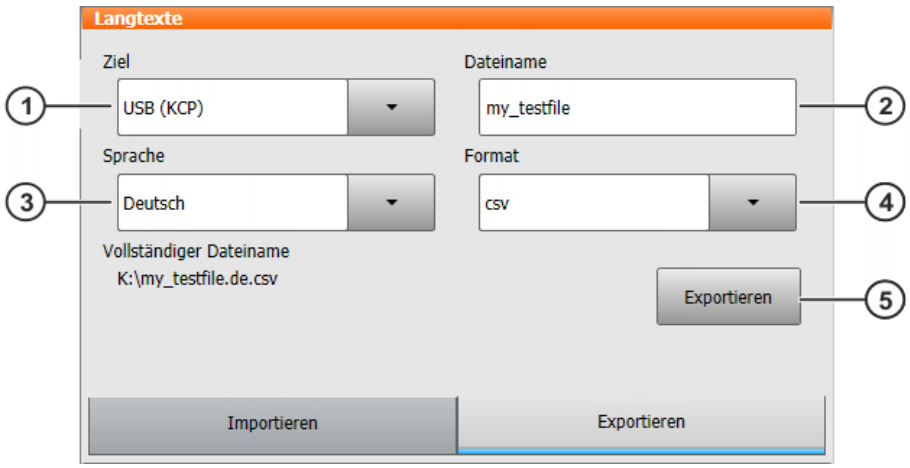


Abb. 5-36: Langtexte exportieren

Pos.	Beschreibung
1	Auswählen, wohin die Datei exportiert werden soll. Der Eintrag **Netzwerk** steht hier nur zur Auswahl, wenn im Fenster **Roboterdaten** ein Pfad konfiguriert ist.
2	Den gewünschten Dateinamen angeben. Wenn unter Pos. 1 **Netzwerk** ausgewählt wurde, wird der Archivname angezeigt, der im Fenster **Roboterdaten** konfiguriert ist. Der Name kann hier geändert werden. Im Fenster **Roboterdaten** ändert er sich dadurch nicht. Der Name wird noch mit einem automatischen Zusatz versehen, je nach gewählter Sprache.
3	Wählen, aus welcher Sprache die Langtexte exportiert werden sollen. Wenn z. B. die smartHMI auf "English" eingestellt ist und man wählt hier "Italiano", wird eine Datei mit dem Zusatz "it" erstellt. Sie enthält die Langtexte, die auf der italienischen smartHMI hinterlegt wurden. Auch **Alle Sprachen** kann ausgewählt werden.
4	Das gewünschte Dateiformat wählen.
5	Startet den Export.

Registerkarte Importieren

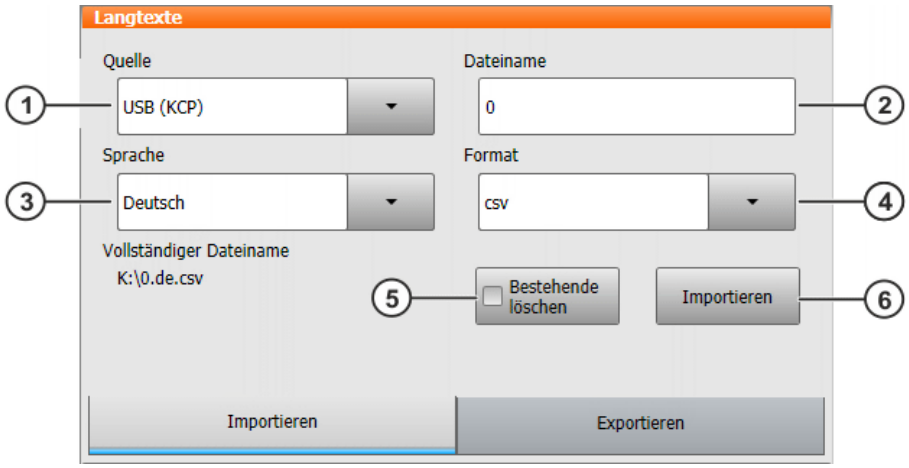


Abb. 5-37: Langtexte importieren

Pos.	Beschreibung
1	Angeben, von wo importiert werden soll. Der Eintrag **Netzwerk** steht hier nur zur Auswahl, wenn im Fenster **Roboterdaten** ein Pfad konfiguriert ist.
2	Den Namen der zu importierenden Datei angeben, jedoch ohne Sprachkürzel. Wenn unter Pos. 1 **Netzwerk** ausgewählt wurde, wird der Archivname angezeigt, der im Fenster **Roboterdaten** konfiguriert ist. Der Name kann hier geändert werden. Im Fenster **Roboterdaten** ändert er sich dadurch nicht.
3	Die Sprache angeben, die zum Sprachkürzel der Datei passt.
4	Das Format der Datei angeben.
5	■ **Aktiv:** Alle bestehenden Langtexte werden gelöscht. Der Inhalt der Datei wird übernommen. ■ **Inaktiv:** Einträge in der Datei überschreiben bestehende Langtexte. Bestehende Langtexte, zu denen in der Datei kein Eintrag existiert, bleiben erhalten.
6	Startet den Import.

5.13 Wartungshandbuch

In der KUKA System Software steht die Funktionalität **Wartungshandbuch** zur Verfügung. Das Wartungshandbuch ermöglicht es, Wartungen zu protokollieren. Die protokollierten Wartungen können in einer Übersicht angezeigt werden.

Die Robotersteuerung macht mit Meldungen darauf aufmerksam, wenn eine Wartung fällig wird:

- Einen Monat vor der Fälligkeit wird eine Meldung ausgegeben. Diese Meldung kann quittiert werden.
- Nach Ablauf des Monats gibt die Robotersteuerung eine Meldung aus, dass die Wartung fällig ist. Diese Meldung kann nicht quittiert werden. Zusätzlich blinkt die LED4 am Controller System Panel (= erste LED von links in der unteren Reihe).

  Erst wenn die entsprechende Wartung protokolliert wurde, blendet die Robotersteuerung die Meldung aus und die LED hört auf zu blinken.

Die Steuerungsvariante "KR C4 compact" besitzt kein Controller System Panel und keine Blinkanzeige für fällige Wartungen.

Die Fälligkeiten richten sich nach den Wartungsintervallen in den KUKA-Wartungsverträgen. Die Intervalle werden ab dem Zeitpunkt der Erstinbetriebnahme der Robotersteuerung gezählt. Gezählt wird die Betriebsdauer des Roboters.

5.13.1 Wartung protokollieren

Beschreibung

Es ist nicht möglich, an einem Tag mehrere Wartungen der gleichen Art zu protokollieren.

> Nach dem Speichern sind keine Änderungen mehr möglich.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **Wartungshandbuch** wählen. Das Fenster **Wartungshandbuch** öffnet sich.
2. Die Registerkarte **Wartungseingabe** wählen und die Angaben zur Wartung eintragen. Es muss in alle Felder eine Eintragung gemacht werden.
3. **Speichern** drücken. Eine Sicherheitsabfrage wird angezeigt.
4. Wenn alle Angaben korrekt sind, die Sicherheitsabfrage mit **Ja** beantworten.

Die Angaben sind jetzt gespeichert. Wenn man zur Registerkarte **Wartungsübersicht** wechselt, wird die Wartung dort angezeigt.

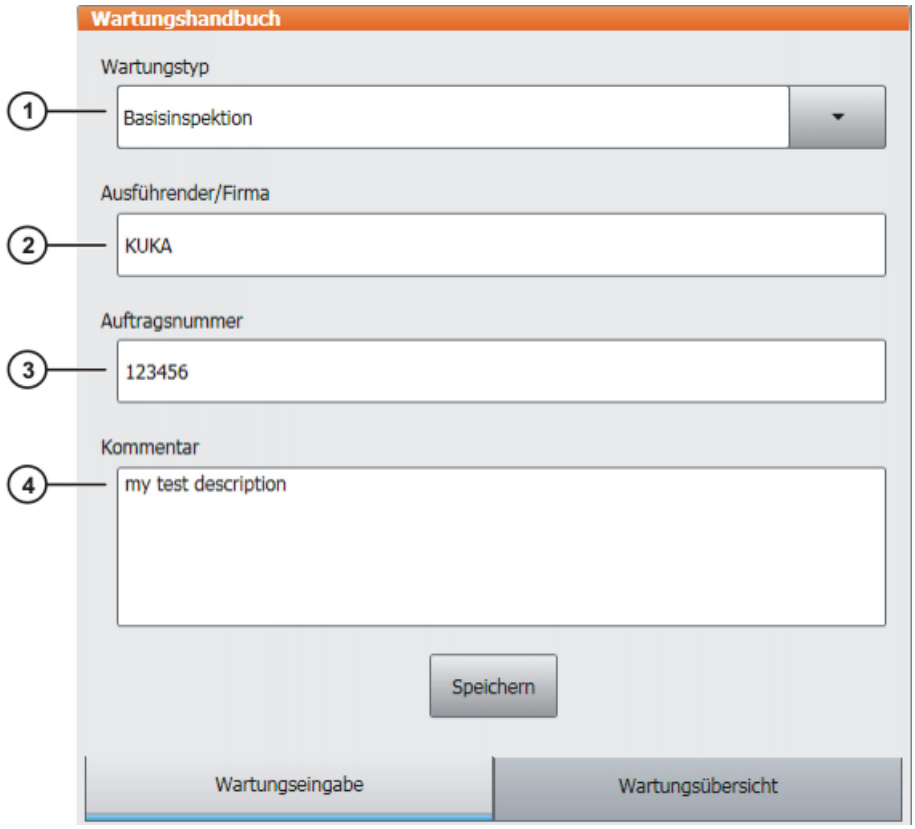


Abb. 5-38: Wartungseingabe

Pos.	Beschreibung
1	Auswählen, welche Art Wartung durchgeführt wurde.
2	Eintragen, wer die Wartung durchgeführt hat.

Pos.	Beschreibung
3	Bei Wartungen, die von KUKA-Mitarbeitern durchgeführt und protokolliert werden: Die Auftragsnummer eintragen. Bei anderen Wartungen: Eine beliebige Nummer eintragen.
4	Einen Kommentar eintragen.

Wartungsarten

Defaultmäßig können folgenden Wartungsarten ausgewählt werden:

- **Basisinspektion**
- **Zentralhandwartung**
- **Grundachswartung**
- **Getriebespielmessung**
- **Kleine Elektrowartung**
- **Große Elektrowartung**
- **Datensicherung mit Ersatzfestplatte**
- **Reparatur**

Diese Wartungsarten entsprechen denjenigen in den KUKA-Wartungsverträgen. Abhängig davon, welche Optionen verwendet werden (z. Bsp. eine Linearachse oder Technologiepakete), können weitere Wartungsarten zur Auswahl stehen.

5.13.2 Wartungsprotokoll anzeigen

Beschreibung

Die protokollierten Wartungen können in einer Übersicht angezeigt werden. Wenn die KUKA System Software upgedatet wird, bleibt diese Übersicht erhalten.

Wenn eine Archivierung durchgeführt wird, werden die protokollierten Wartungen immer mitarchiviert. Wenn die Daten wiederhergestellt werden und auf der Robotersteuerung inzwischen weitere Wartungen protokolliert wurden, werden diese nicht überschrieben, sondern die Übersicht wird mit den wiederhergestellten Protokollen ergänzt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Service** > **Wartungshandbuch** wählen. Das Fenster **Wartungshandbuch** öffnet sich.
2. Die Registerkarte **Wartungsübersicht** wählen.

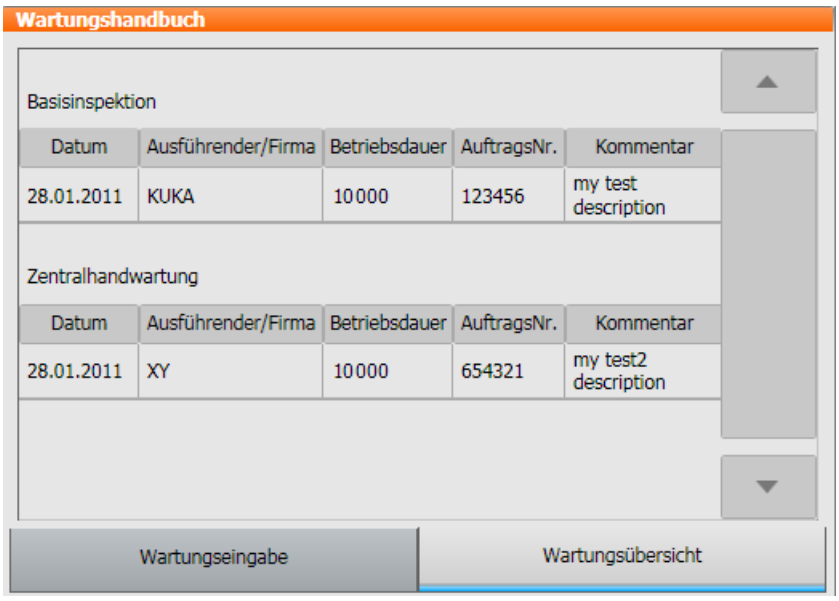


Abb. 5-39: Wartungsübersicht

6 Konfiguration

6.1 KUKA Line Interface (KLI) konfigurieren

Das KLI ist die Ethernet-Schnittstelle der Robotersteuerung für die externe Kommunikation. Es ist eine physikalische Schnittstelle und kann mehrere virtuelle Schnittstellen enthalten.

Damit externe PCs sich über Netzwerk mit der Robotersteuerung verbinden können, muss das KLI dementsprechend konfiguriert werden. Dies ist z. B. die Voraussetzung dafür, dass WorkVisual-Projekte über Netzwerk auf die Robotersteuerung übertragen werden können.

Folgende Adress-Bereiche werden defaultmäßig von der Robotersteuerung für interne Zwecke genutzt. IP-Adressen aus diesen Bereichen können deshalb vom Benutzer über die smartHMI nicht vergeben werden. Das System zeigt einen Fehler an.

- 192.168.0.0 … 192.168.0.255
- 172.16.0.0 … 172.16.255.255
- 172.17.0.0 … 172.17.255.255

6.1.1 Windows-Schnittstelle konfigurieren (ohne PROFINET)

Beschreibung

Die Windows-Schnittstelle ist eine virtuelle Schnittstelle des KLI. Sie hat eine vorkonfigurierte statische IP-Adresse:

- IP-Adresse: 172.31.1.147
- Subnetzmaske: 255.255.0.0

Diese Angaben kann der Benutzer ändern. Bei Bedarf kann auch auf dynamische Adressvergabe umgestellt werden. Ebenso kann von einer dynamischen Adresse wieder auf eine statische Adresse zurückgestellt werden.

DNS-Server-Adresse:

Bei Bedarf kann eine DNS-Server-Adresse angegeben werden.

Wenn die Robotersteuerung an das IT-Netz angeschlossen ist, können über die DNS-Server-Adresse die Namen von Teilnehmern aufgelöst werden. (D. h. der DNS-Server erhält eine Anfrage mit einem Teilnehmer-Namen und sendet als Antwort die zugehörige IP-Adresse.)

"0.0.0.0" ist zulässig und wird vom System ignoriert.

Voraussetzung

- PROFINET wird nicht verwendet.
- Wenn der Adresstyp **Dynamische IP-Adresse** gewählt werden soll: Im Netzwerk ist ein DHCP-Server vorhanden.
- Es ist kein Programm angewählt.
- Betriebsart T1 oder T2
- Benutzergruppe Experte

Vorgehensweise

Adress- und Subnetz-Felder können rot umrandet angezeigt werden. Dies bedeutet, dass ein Fehler vorliegt.
(>>> 6.1.6 "Fehleranzeige bei Adress- und Subnetz-Feldern" Seite 168)

1. Im Hauptmenü **Inbetriebnahme** > **Netzwerkkonfiguration** wählen. Das Fenster **Netzwerkkonfiguration** öffnet sich. Die aktive Windows-Schnittstelle wird angezeigt. (Default: "virtual5")

2. Im Feld **Adresstyp** den gewünschten Typ auswählen: **Dynamische IP-Adresse** oder **Feste IP-Adresse**

> Die weiteren Typen in diesem Feld (z. B. **Echtzeit-IP-Adresse**) dürfen nicht ausgewählt werden.

3. Nur bei **Dynamische IP-Adresse**:
   - Folgende Felder werden ausgeblendet, da die Angaben automatisch vom DHCP-Server vergeben werden: **IP-Adresse**, **Subnetzmaske**, **Standardgateway**
   - Das Feld **DNS Server** ausfüllen (bei Bedarf).
4. Nur bei **Feste IP-Adresse**: Folgende Felder ausfüllen:
   - **IP-Adresse**: IP-Adresse der Robotersteuerung eintragen.
   - **Subnetzmaske**: Die Subnetz-Maske muss passend zum IP-Netz gewählt werden.
   - **Standardgateway** (bei Bedarf) : Gibt an, über welche IP-Adresse das Netzwerk verlassen werden kann.

     "0.0.0.0" ist zulässig und wird vom System ignoriert.
   - **DNS Server** (bei Bedarf)
5. Auf **Speichern** drücken.
6. Die Robotersteuerung neu starten, um die Änderung zu übernehmen.

Abb. 6-1: Beispiel: Feste IP-Adresse

6.1.2 PROFINET-Schnittstelle konfigurieren und Windows-Schnittstelle anlegen

Beschreibung PROFINET belegt automatisch die virtuelle Schnittstelle "virtual5". Diese muss an PROFINET angepasst werden.

Die "virtual5" kann auch als Windows-Schnittstelle verwendet werden, jedoch muss dann für den Windows-Zugang die statische IP-Adresse von PROFINET verwendet werden.

Wenn für die Windows-Schnittstelle eine andere IP-Konfiguration gelten soll, muss eine zusätzliche virtuelle Schnittstelle, z. B. "virtual6" hinzugefügt und für diesen Zweck konfiguriert werden.

Voraussetzung

- PROFINET wird verwendet.
- PROFINET-Gerätetaufe ist abgeschlossen.

> Informationen zur Gerätetaufe sind in der Dokumentation **KR C4 PROFINET** zu finden.

- Es ist kein Programm angewählt.
- Betriebsart T1 oder T2
- Benutzergruppe Experte

Vorgehensweise

> Adress- und Subnetz-Felder können rot umrandet angezeigt werden. Dies bedeutet, dass ein Fehler vorliegt.
> (>>> 6.1.6 "Fehleranzeige bei Adress- und Subnetz-Feldern" Seite 168)

1. Im Hauptmenü **Inbetriebnahme** > **Netzwerkkonfiguration** wählen. Das Fenster **Netzwerkkonfiguration** öffnet sich. Die Schnittstelle "virtual5" wird angezeigt.
2. Falls nicht bereits ausgewählt: Im Feld **Adresstyp** den Typ **Feste IP-Adresse** auswählen.
3. Die Felder **IP-Adresse** und **Subnetzmaske** ausfüllen. Hierbei die Adresse und Maske eingeben, die auch von der PROFINET-SPS vergeben werden.
4. Auf **Erweitert…** drücken. Das Fenster für die erweiterte Netzwerkkonfiguration öffnet sich.
5. Die Registerkarte **Interfaces** wählen.
6. Im Bereich **Konfigurierte Interfaces** den Eintrag "virtual5" markieren und im Feld **Interfacebezeichnung** "PROFINET" eingeben.
7. Unterhalb des Eintrags "virtual5" werden mehrere Einträge "Empfangstask" angezeigt. Den untersten markieren.
8. Das Feld **Empfangsfilter** zeigt **Alles annehmen** an. Den Eintrag auf **Ziel-IP-Adresse** ändern.

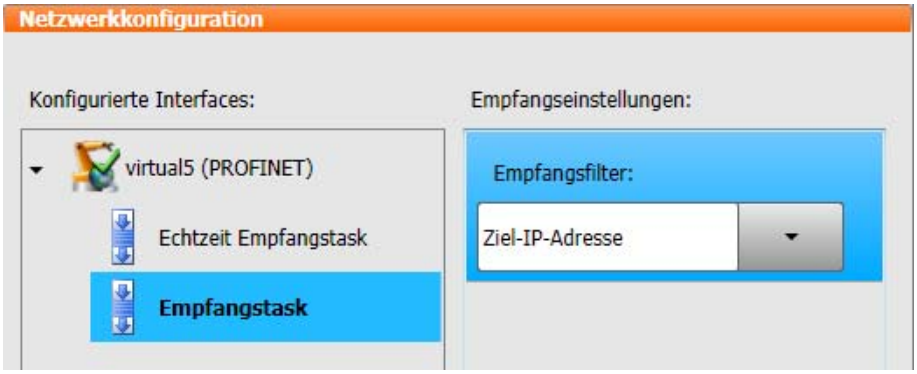


Abb. 6-2: Empfangsfilter

9. Den Eintrag "virtual5" markieren und auf die Schaltfläche **Interface hinzufügen** drücken. Der Eintrag "virtual6" wird automatisch angelegt.
10. Den Eintrag "virtual6" markieren und im Feld **Interfacebezeichnung** "WINDOWS" eingeben.

11. Im Feld **Adresstyp** den Typ **Dynamische IP-Adresse** auswählen.

> Hier kann auch **Feste IP-Adresse** ausgewählt werden, wenn eine statische IP-Adresse gewünscht ist. Die statische Adresse muss jedoch in einem anderen IP-Bereich liegen als die Adresse von "virtual5".

12. In der Checkbox **Windows-Schnittstelle** das Häkchen setzen.

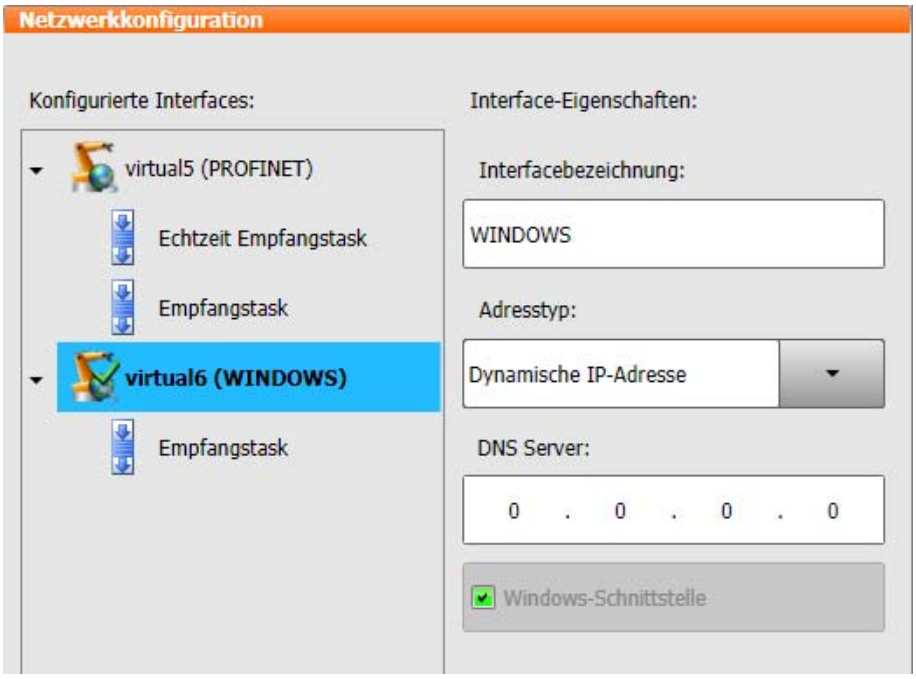


Abb. 6-3: Schnittstelle "virtual6"

13. Unterhalb des Eintrags "virtual6" den Eintrag "Empfangstask" markieren.
14. Wenn nicht bereits der Fall: Das Feld **Empfangsfilter** auf **Alles annehmen** ändern.
15. Auf **Speichern** drücken.
16. Das Fenster **Netzwerkkonfiguration** über das Schließen-Symbol schließen.
17. Die Robotersteuerung neu starten. Hierzu im Hauptmenü **Herunterfahren** wählen und die Option **Dateien neu einlesen** wählen.

Das Häkchen in der Checkbox **Windows-Schnittstelle** kann nicht wieder entfernt werden. Eine Abwahl ist nur möglich, indem eine andere Schnittstelle als Windows-Schnittstelle definiert wird.

6.1.3 Ports der Windows-Schnittstelle anzeigen oder weiteren Port freigeben

> In der Regel ist es nicht notwendig, zusätzliche Ports freizugeben. Wenn dies dennoch geschehen soll, muss vorher Kontakt zur KUKA Roboter GmbH aufgenommen werden.

> **HINWEIS** Die von KUKA defaultmäßig freigegebenen Ports dürfen nicht entfernt werden. Wenn dies dennoch geschieht, kann dies den Verlust von Funktionalitäten der Robotersteuerung zur Folge haben.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

- Es ist kein Programm angewählt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Netzwerkkonfiguration** wählen. Das Fenster **Netzwerkkonfiguration** öffnet sich.
2. Auf **Aktivieren** drücken. Das Fenster für die erweiterte Netzwerkkonfiguration öffnet sich.
3. Die Registerkarte **NAT** wählen. Im Bereich **Vorhandene Ports** wird eine Liste mit allen freigegebenen Ports der Windows-Schnittstelle angezeigt.
4. Nur, wenn ein Port freigegeben werden soll:
   a. Auf **Port hinzufügen** drücken. Der Liste wird ein neuer Port mit der Nummer "0" hinzugefügt.
   b. Die Felder **Portnummer** und **Zugelassene Protokolle** füllen.
   c. Auf **Speichern** drücken.

   Insgesamt können maximal 40 Ports freigegeben sein.
5. Das Fenster **Netzwerkkonfiguration** über das Schließen-Symbol schließen.
6. Nur, wenn Änderungen durchgeführt wurden:

   Die Robotersteuerung neu starten, um die Änderungen zu übernehmen. Wenn PROFINET verwendet wird, hierzu im Hauptmenü **Herunterfahren** wählen und die Option **Dateien neu einlesen** wählen.

6.1.4 Filter anzeigen oder ändern

In der Regel ist es nicht notwendig, die Filter zu ändern. Wenn dies dennoch geschehen soll, muss vorher Kontakt zur KUKA Roboter GmbH aufgenommen werden.

HINWEIS Die von KUKA defaultmäßig gesetzten Filter dürfen nicht geändert oder entfernt werden. Wenn dies dennoch geschieht, kann dies den Verlust von Funktionalitäten der Robotersteuerung zur Folge haben.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Netzwerkkonfiguration** wählen. Das Fenster **Netzwerkkonfiguration** öffnet sich.
2. Auf **Aktivieren** drücken. Das Fenster für die erweiterte Netzwerkkonfiguration öffnet sich.
3. Die Registerkarte **Interne Subnetze** wählen. Die Filter des KUKA Line Interface und ihre Eigenschaften werden hier angezeigt.
4. Nur, wenn Änderungen erforderlich sind: Diese durchführen und auf **Speichern** drücken.
5. Das Fenster **Netzwerkkonfiguration** über das Schließen-Symbol schließen.
6. Nur, wenn Änderungen durchgeführt wurden:

   Die Robotersteuerung neu starten, um die Änderungen zu übernehmen. Wenn PROFINET verwendet wird, hierzu im Hauptmenü **Herunterfahren** wählen und die Option **Dateien neu einlesen** wählen.

6.1.5 Subnetz-Konfiguration der Robotersteuerung anzeigen

Beschreibung

Die Subnetz-Konfiguration der Robotersteuerung kann angezeigt werden. Dies ermöglicht es, sie mit den Subnetzen des Kunden-Netzwerks zu vergleichen.

HINWEIS Die Subnetz-Konfiguration der Robotersteuerung darf nur nach Rücksprache mit der KUKA Roboter GmbH geändert werden. Änderungen ohne Rücksprache können den Verlust von Funktionalitäten der Robotersteuerung zur Folge haben.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Netzwerkkonfiguration** wählen. Das Fenster **Netzwerkkonfiguration** öffnet sich.
2. Auf **Aktivieren** drücken. Das Fenster für die erweiterte Netzwerkkonfiguration öffnet sich.
3. Die Registerkarte **Interne Subnetze** wählen. Die Subnetz-Konfiguration der Robotersteuerung wird angezeigt.

   Hierbei wird nur die Netzadresse angezeigt. Der Bereich, der die Adresse des jeweiligen Teilnehmers enthält, ist mit "x" gekennzeichnet.

Adress- und Subnetz-Felder können rot umrandet angezeigt werden. Dies bedeutet, dass ein Fehler vorliegt.
(>>> 6.1.6 "Fehleranzeige bei Adress- und Subnetz-Feldern" Seite 168)

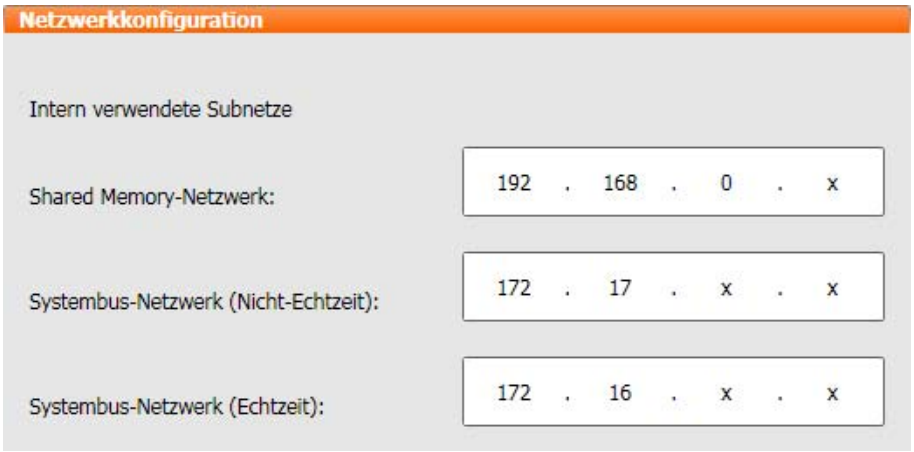


Abb. 6-4: Beispiel: Interne Subnetze

6.1.6 Fehleranzeige bei Adress- und Subnetz-Feldern

Beschreibung

Adress- und Subnetz-Felder können rot umrandet sein. Dies zeigt einen Fehler an und kann folgende Ursachen haben:

Ursache	Mögliche Abhilfe
Die Eingabe ist nicht konform zur IP-Systematik Beispiel: Die Zahlenbereiche der IP-Adresse und des Subnetzes passen nicht zueinander.	Das rot umrandete Feld prüfen und korrigieren. Die rote Umrandung verschwindet.
Diese Adresse wird bereits in einem der internen Subnetze verwendet. In diesem Fall wird sowohl die Adresse rot umrandet angezeigt als auch das zugehörige Feld in der Registerkarte **Interne Subnetze**.	■ Die eigentliche Adresse ändern. ■ Oder, nur nach Rücksprache mit KUKA: Die Adresse des internen Subnetzes ändern. Die rote Umrandung verschwindet.

HINWEIS Die Subnetz-Konfiguration der Robotersteuerung darf nur nach Rücksprache mit der KUKA Roboter GmbH geändert werden. Änderungen ohne Rücksprache können den Verlust von Funktionalitäten der Robotersteuerung zur Folge haben.

Beispiel

Beispiel für eine nicht-Systematik-konforme Eingabe:

Subnetz-Masken dürfen in der Binärdarstellung nur geschlossene Gruppen von führenden Einsen enthalten.

- Die Subnetz-Maske "255.255.208.0" wird eingegeben.
- Das Feld wird nun rot umrandet angezeigt.
  Grund: Die binäre Darstellung von "208" entspricht "11010000" und ist somit keine gültige Eingabe.
- Mögliche Abhilfe: 255.255.240.0 eingeben.
  Die binäre Darstellung von "240" entspricht "11110000" und ist somit eine gültige Eingabe.

6.2 E/A-Treiber rekonfigurieren

Beschreibung

Der Befehl **E/A Treiber** > **Rekonfigurieren** bewirkt, dass sämtliche Dateien im Verzeichnis C:\KRC\ROBOTER\Config\User neu eingelesen werden. Änderungen, die in diesen Dateien gemacht wurden, werden übernommen.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Während der Rekonfiguration gehen alle Ausgänge kurzzeitig auf Null und kehren danach wieder in ihren ursprünglichen Zustand zurück.

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Ein-/Ausgänge** > **E/A Treiber** wählen.
2. Die Schaltfläche **Rekonfigurieren** berühren.
   Es spielt keine Rolle, ob man sich dazu in der Registerkarte **Zustand** oder **Konfiguration** befindet.
3. Die Sicherheitsabfrage *Wollen Sie wirklich alle E/A-Treiber rekonfigurieren?* mit **Ja** beantworten.
   Die Meldung *Rekonfiguration wird durchgeführt ...* wird angezeigt. Wenn sich die Meldung wieder ausblendet, ist der Rekonfiguration abgeschlossen.

6.3 Sichere Achsüberwachungen konfigurieren

Voraussetzung

- Benutzergruppe Sicherheitsinstandhalter
- Betriebsart T1 oder T2

Vorgehensweise

Um die Konfigurationswerte zu speichern, muss eine Rekonfiguration durchgeführt werden. Während der Rekonfiguration gehen alle Ausgänge kurzzeitig auf Null und kehren danach wieder in ihren ursprünglichen Zustand zurück.

1. Im Hauptmenü **Konfiguration** > **Sicherheitskonfiguration** wählen.
   Das Fenster **Sicherheitskonfiguration** öffnet sich.
2. Die Registerkarte **Achsüberwachung** wählen.
3. Die Parameter nach Bedarf bearbeiten und auf **Speichern** drücken.
4. Die Sicherheitsabfrage mit **Ja** beantworten. Die Steuerung wird rekonfiguriert.
5. Wenn die Rekonfiguration abgeschlossen ist, wird folgende Meldung angezeigt: *Die Änderungen wurden erfolgreich gespeichert.*
   Die Meldung mit **OK** bestätigen.

⚠ WARNUNG Nach Änderungen an der Sicherheitskonfiguration müssen die sicheren Achsüberwachungen geprüft werden.
(>>> 6.4 "Sichere Achsüberwachungen prüfen" Seite 172)

Editierbare Parameter

Die folgenden Parameter können pro Achse eingestellt werden. In der Regel ist es jedoch nicht notwendig, die Default-Werte zu ändern.

Parameter	Beschreibung
Bremszeit	Dauer der überwachten achsspezifischen Bremsrampe für Sicherheitshalt 1 und Sicherheitshalt 2 Default: 1 500 ms (>>> 6.3.1 "Parameter Bremszeit" Seite 171)
Maximale Geschwindigkeit T1	Maximale Geschwindigkeit in T1 ■ Rotatorische Achsen: **1.00 … 100.00 °/s** Default: 30 °/s° ■ Lineare Achsen: **1.00 … 1 500.00 mm/s** Default: 250 mm/s Dieser Parameter ermöglicht es z. B., eine Servozange in T1 mit höherer Geschwindigkeit als 250 mm/s zu kalibrieren. **Hinweis:** Die kartesischen Geschwindigkeiten am Flansch und am TCP werden unabhängig von diesem Parameter überwacht und können 250 mm/s nicht überschreiten.
Positionstoleranz	Toleranz für die Stillstandsüberwachung beim Sicheren Betriebshalt. Innerhalb dieser Toleranz darf sich die Achse bei einem aktiven Sicheren Betriebshalt noch bewegen. ■ Rotatorische Achsen: **0.001 … 1 °** Default: 0.01 ° ■ Lineare Achsen: **0.003 … 3 mm** Default: 0.1 mm

6.3.1 Parameter Bremszeit

Beschreibung

Wenn ein Sicherheitshalt 1 oder 2 eintritt, überwacht die Sicherheitssteuerung den Bremsvorgang. Sie überwacht unter anderem, ob die achsspezifische Geschwindigkeit unterhalb der Bremsrampe bleibt. Wenn die Geschwindigkeit zu hoch ist, wenn also die Bremsrampe verletzt wird, dann löst die Sicherheitssteuerung einen Sicherheitshalt 0 aus.

Die Bremsrampe ergibt sich aus einem internen Faktor für die Rampenneigung und aus dem Wert **Bremszeit**.

Das bedeutet: Über den Parameter **Bremszeit** beeinflusst man eine Überwachungsfunktion. **Bremszeit** beeinflusst jedoch nicht das eigentliche Bewegungsverhalten der Kinematik.

Der Parameter **Bremszeit** wirkt sich in T1 nicht aus, da sich der Wert auf die achsspezifische Überwachung bezieht. In T1 gilt jedoch außerdem eine (nicht konfigurierbare) Überwachung für die kartesische Geschwindigkeit am Flansch. Diese ist strenger, deshalb kommt die achsspezifische Überwachung nicht zum Tragen.

⚠ WARNUNG Die Default-Zeit nur ändern, wenn es notwendig ist. Dies kann z. B. der Fall sein bei sehr schweren Maschinen und/oder sehr schweren Lasten, da diese nicht innerhalb der Default-Zeit zum Stehen kommen können.
Der Sicherheitsinbetriebnehmer muss prüfen, ob und inwieweit der Wert **Bremszeit** im konkreten Anwendungsfall geändert werden muss. Er muss auch prüfen, ob durch die Änderung zusätzliche Sicherheitsmaßnahmen erforderlich werden, z. B. ob eine Türzuhaltung installiert werden muss.

Bremszeit kann zwar pro Achse eingetragen werden. Zum Bremszeitpunkt wird jedoch immer für alle Achsen der gleiche Wert verwendet, und zwar der höchste der eingetragenen Werte.
Empfehlung: Für eine bessere Übersichtlichkeit für alle Achsen den identischen Wert eintragen.

Bremszeit geändert

Wenn der Wert Bremszeit erhöht wird, hat dies folgende Konsequenzen:

Die Bremsrampe wird länger und flacher, d. h. die Überwachung ist nun weniger streng. Ein und derselbe Bremsvorgang verletzt die Bremsrampe nun weniger wahrscheinlich als vorher.

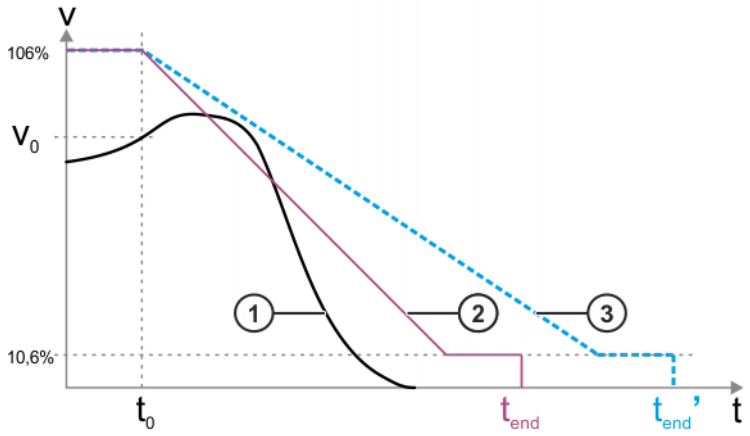


Abb. 6-5: Beispiel: Wert wird erhöht

1 Geschwindigkeitsverlauf beim Bremsvorgang (Beispiel)
2 Bremsrampe (ursprünglicher Wert **Bremszeit**)
3 Bremsrampe (höherer Wert **Bremszeit**)

v_0 Geschwindigkeit der Kinematik in dem Moment, wenn der Bremsvorgang einsetzt

t_0 Startzeitpunkt der Bremsrampe

t_{end} Ende der Bremsrampe

t_{end}' Ende der Bremsrampe bei höherem Wert für **Bremszeit**

Die Startgeschwindigkeit der achsspezifischen Bremsrampe liegt immer bei 106 % der Nenn-Drehzahl der Achse. Die Rampe senkt sich bis auf 10,6 %. Die Geschwindigkeit bleibt dann 300 ms so und fällt danach auf 0 %.

Wenn der Wert Bremszeit verringert wird, hat dies folgende Konsequenzen:

Die Bremsrampe wird kürzer und steiler, d. h. die Überwachung ist nun strenger. Ein und derselbe Bremsvorgang verletzt die Bremsrampe nun wahrscheinlicher als vorher.

6.4 Sichere Achsüberwachungen prüfen

Beschreibung

Wenn die Sicherheitskonfiguration gespeichert wird, können systemseitig zufällige Fehler auftreten, die dazu führen, dass die Sicherheitskonfiguration letztlich andere Werte enthält als die, die der Benutzer gespeichert hat. Dies stellt den Ausnahmefall dar, kann aber nicht vollends ausgeschlossen werden.

Um auszuschließen, dass bei den Parametern **Bremszeit** und **Positionstoleranz** ein solcher Fehler aufgetreten ist, müssen deren Werte im Diagnosemonitor verifiziert werden. Eine andere Art der Verifizierung ist für diese Parameter nicht möglich.

⚠ WARNUNG Die Werte müssen immer dann geprüft werden, wenn sich im Fenster **Sicherheitskonfiguration** in der Registerkarte **Allgemein** die Prüfsumme geändert hat. D. h. also, nicht nur dann, wenn die Werte selber geändert wurden, sondern generell nach Änderungen, die die Sicherheitskonfiguration betreffen.
Wenn die Prüfung unterlassen wird, kann die Sicherheitskonfiguration falsche Daten enthalten. Tod von Personen, schwere Verletzungen oder erhebliche Sachschäden können die Folge sein.

Voraussetzung

- Die zuletzt gespeicherten Werte für die Parameter **Bremszeit** und **Positionstoleranz** sind bekannt.
  In der Regel können die zuletzt gespeicherten Werte einer Checkliste, einem Abnahmeprotokoll o.ä. entnommen werden.

Vorgehensweise

SICHERHEITS-ANWEISUNGEN Die folgende Vorgehensweise genau einhalten!

1. Im Hauptmenü **Diagnose** > **Diagnosemonitor** wählen.
   Das Fenster **Diagnosemonitor** öffnet sich.
2. Im Feld **Modul** den Bereich **Sicherheitssteuerung (HnfHlp)** auswählen.
   Zu dem Bereich werden nun Daten angezeigt.
3. Die angezeigten Werte für **Bremszeit** und **Positionstoleranz** mit denen vergleichen, die zuletzt gespeichert wurden.
4. Ergebnis:
   - Wenn sich die Werte entsprechen: OK.
     Das Fenster **Diagnosemonitor** schließen. Es sind keine weiteren Aktionen notwendig.
   - Wenn sich die Werte nicht entsprechen:

Die Werte nochmal eingeben und speichern. Gegebenenfalls das WorkVisual-Projekt nochmal auf die Robotersteuerung übertragen.

Dann die Prüfung nochmal durchführen. Wenn sich die Werte immer noch nicht entsprechen, muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

6.5 Sicherheitskonfiguration der Robotersteuerung prüfen

Beschreibung

In folgenden Fällen muss die Sicherheitskonfiguration der Robotersteuerung geprüft werden:

- Nach der Aktivierung eines WorkVisual-Projekts auf der Robotersteuerung
- Allgemein nach Änderungen an den Maschinendaten (unabhängig von WorkVisual).

⚠ WARNUNG Wenn die Sicherheitskonfiguration nicht geprüft und gegebenenfalls aktualisiert wird, kann sie falsche Daten enthalten. Tod, Verletzungen oder Sachschäden können die Folge sein.

Voraussetzung

- Benutzergruppe Sicherheitsinstandhalter

Vorgehensweise

SICHERHEITS-ANWEISUNGEN Die folgende Vorgehensweise genau einhalten!

1. Im Hauptmenü **Konfiguration** > **Sicherheitskonfiguration** wählen.
2. Die Sicherheitskonfiguration prüft, ob relevante Abweichungen zwischen den Daten der Robotersteuerung und der Sicherheitssteuerung vorliegen.
3. Folgende Fälle können nun auftreten:
   a. Wenn keine Abweichungen vorliegen, öffnet sich das Fenster **Sicherheitskonfiguration**. Es wird keine Meldung angezeigt. Es sind keine weiteren Aktionen notwendig.
   b. Wenn Abweichungen bezüglich der Maschinendaten vorliegen, wird eine Dialogmeldung angezeigt. Die Abweichungen können nun synchronisiert werden oder nicht. In jedem Fall sind sicherheitsrelevante Maßnahmen zu beachten.
      (>>> "Abweichungen" Seite 173)
   c. Die Sicherheitskonfiguration prüft außerdem, ob (außer bei den Maschinendaten) weitere Abweichungen zwischen der Robotersteuerung und der Sicherheitssteuerung vorliegen.
      Wenn ja, öffnet sich das Fenster **Problembehebungsassistent**. Eine Beschreibung des Problems und eine Liste mit möglichen Ursachen wird angezeigt. Der Benutzer kann die zutreffende Ursache auswählen. Der Assistent schlägt dann eine Lösung vor.

Abweichungen

Die Dialogmeldung zeigt an, welche Maschinendaten der Robotersteuerung von denen der Sicherheitssteuerung abweichen.

Die Meldung fragt, ob die Sicherheitskonfiguration aktualisiert werden soll, d. h. ob die Maschinendaten der Robotersteuerung in die Sicherheitskonfiguration übernommen werden sollen.

- Wenn ja: Die Abfrage mit **Ja** bestätigen.

⚠ WARNUNG Wenn Abweichungen übernommen wurden, müssen danach die Sicherheitsmaßnahmen für die Inbetriebnahme und Wiederinbetriebnahme durchgeführt werden.

- Wenn nein: Die Abfrage mit **Nein** beantworten.

⚠ WARNUNG In diesem Fall darf der Roboter nicht betrieben werden. Die Maschinendaten müssen geprüft und korrigiert werden, so dass ein Zustand erreicht wird, in dem die Sicherheitskonfiguration entweder keine Abweichungen mehr feststellt oder die festgestellten Abweichungen übernommen werden können.

6.6 Prüfsumme der Sicherheitskonfiguration

Beschreibung

Bei jedem Speichern der Sicherheitskonfiguration aktualisiert sich die Prüfsumme. Die Robotersteuerung zeigt die Prüfsumme an folgenden Stellen an:

- Im Fenster **Sicherheitskonfiguration** in der Registerkarte **Allgemein**
  Das Fenster kann über das Hauptmenü geöffnet werden, über **Konfiguration** > **Sicherheitskonfiguration**.
- Für Benutzergruppe Experte oder höher:
  In der Datei SGTLCRC.XML im Verzeichnis C:\KRC\ROBOTER\Config\User\Common

Die SCTLCRC.XML steht zur Verfügung, damit der Systemintegrator bei Bedarf die Prüfsumme dort durch die übergeordnete Steuerung auslesen kann.

Es wird empfohlen, bei der Inbetriebnahme und Wiederinbetriebnahme des Industrieroboters zu prüfen, ob der Wert korrekt aus der SCTLCRC.XML ausgelesen und an die übergeordnete Steuerung übertragen wird.

SCTLCRC.XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <SafetyInfo ParameterCrc="3702332E" />
```

Abb. 6-6: Beispiel: Prüfsumme in der SCTLCRC.XML

6.7 Sicherheitskonfiguration exportieren (XML-Export)

Beschreibung

Teile der Sicherheitskonfiguration können exportiert werden. Der Export erzeugt eine XML-Datei. Diese enthält ausschließlich die Parameter, die in Zusammenhang mit Sicherheitsoptionen, z. B. SafeOperation, relevant sind.

- Der Export ist immer möglich, unabhängig davon, ob eine Sicherheitsoption installiert ist oder nicht. Sinnvoll ist der Export jedoch nur mit einer Sicherheitsoption.
- Wenn auf der Robotersteuerung keine Sicherheitsoption installiert ist, sind die Parameter in der XML-Datei mit Default-Werten befüllt (häufig "0").

Wenn eine Sicherheitsoption installiert ist, besteht zusätzlich zum Export auch die Möglichkeit, eine Sicherheitskonfiguration zu importieren. Genauere Informationen zum Export und Import sind in den Dokumentationen zu den Sicherheitsoptionen zu finden.
In WorkVisual besteht ebenfalls die Möglichkeit, Sicherheitskonfigurationen zu importieren oder zu exportieren. Informationen dazu sind in der Dokumentation zu WorkVisual zu finden.

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Sicherheitskonfiguration** wählen.
   Das Fenster **Sicherheitskonfiguration** öffnet sich.
2. Auf **Exportieren** drücken. Die vorhandenen Laufwerke werden angezeigt.
3. Gewünschten Speicherort auswählen und auf **Exportieren** drücken.

Die Sicherheitskonfiguration wird in einer XML-Datei gespeichert. Der Dateiname wird automatisch erzeugt.

6.8 Variablenübersicht konfigurieren

Hier wird definiert, welche Variablen in der Variablenübersicht angezeigt werden und die Anzahl der Gruppen. Es sind maximal 10 Gruppen möglich. Pro Gruppe sind maximal 25 Variablen möglich. Es können Systemvariablen und benutzerdefinierte Variablen angezeigt werden.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Übersicht** > **Konfiguration** wählen.
   Das Fenster **Variablenübersicht – Konfiguration** öffnet sich.
2. Gewünschte Einstellungen vornehmen. Um eine Zelle zu editieren, die Zelle markieren.
3. **OK** drücken, um die Konfiguration zu speichern und das Fenster zu schließen.

Beschreibung

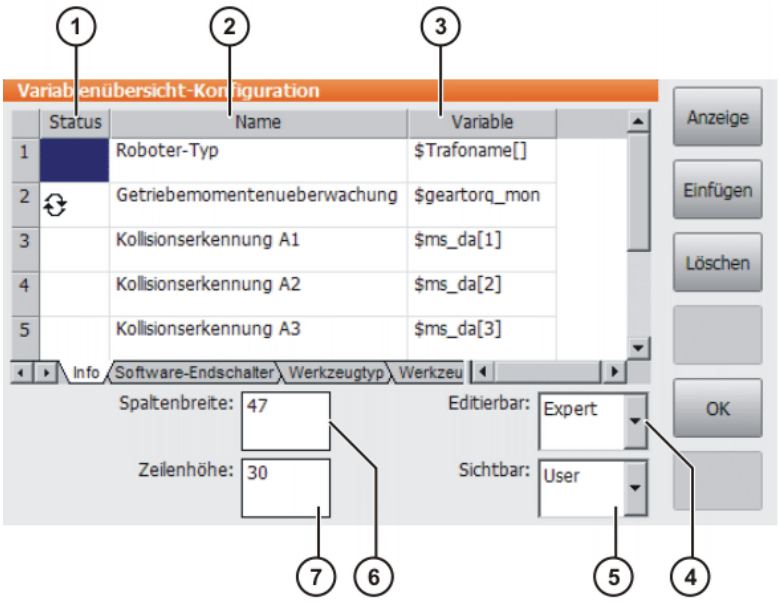


Abb. 6-7: Variablenübersicht Konfiguration

Pos.	Beschreibung
1	Pfeilsymbol: Wenn sich der Wert der Variablen ändert, wird die Anzeige automatisch aktualisiert. Kein Pfeilsymbol: Die Anzeige wird nicht automatisch aktualisiert.
2	Beschreibender Name
3	Pfad und Name der Variablen **Hinweis:** Bei Systemvariablen genügt der Name. Andere Variablen müssen folgendermaßen angegeben werden: /R1/*Programmname*/*Variablenname* Zwischen /R1/ und dem Programmnamen keine Ordner angeben. Beim Programmnamen keine Dateiendung angeben.
4	Benutzergruppe, ab der die aktuelle Gruppe geändert werden kann
5	Benutzergruppe, ab der die aktuelle Gruppe angezeigt werden kann

Pos.	Beschreibung
6	Spaltenbreite in mm. Den gewünschten Wert über die Tastatur eingeben und mit der Eingabe-Taste bestätigen.
7	Zeilenhöhe in mm. Den gewünschten Wert über die Tastatur eingeben und mit der Eingabe-Taste bestätigen.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Anzeige	Schaltet zur Variablenübersicht um. (>>> 4.17.8 "Variablenübersicht anzeigen und Variable ändern" Seite 88)
Einfügen	Zeigt weitere Schaltflächen an: ■ **Zeile davor**: Fügt über der markierten Zeile eine neue Zeile ein. ■ **Zeile darunter**: Fügt unter der markierten Zeile eine neue Zeile ein. ■ **Gruppe davor**: Fügt links von der aktuellen Gruppe eine neue Gruppe ein. ■ **Gruppe dahinter**: Fügt rechts von der aktuellen Gruppe eine neue Gruppe ein.
Löschen	Zeigt weitere Schaltflächen an: ■ **Zeile**: Die markierte Zeile wird gelöscht. ■ **Gruppe**: Die aktuelle Gruppe wird gelöscht.

6.9 Passwort ändern

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Benutzergruppe** wählen. Die aktuelle Benutzergruppe wird angezeigt.
2. **Anmelden…** drücken.
3. Die Benutzergruppe, deren Passwort geändert werden soll, markieren.
4. **Kennwort…** drücken.
5. Das alte Passwort eingeben. Das neue Passwort zweimal eingeben.
   Die Eingaben werden aus Sicherheitsgründen verschlüsselt angezeigt. Die Groß- und Kleinschreibung wird berücksichtigt.
6. **OK** drücken. Das neue Passwort ist sofort gültig.

6.10 Energiespar-Modus ($ECO_LEVEL)

Beschreibung

Über die Systemvariable $ECO_LEVEL kann der Benutzer den Roboter energiesparend verfahren. Der Grad der Einsparung kann auf "gering", "mittel" oder "hoch" eingestellt werden. Die Energiespar-Modus bewirkt, dass die Roboterachsen und die Zusatzachsen langsamer verfahren. Je höher die Einsparung ist, desto geringer ist die Geschwindigkeit. Wieviel Energie im Vergleich zur vollen Leistung gespart wird, hängt hauptsächlich von den Achsstellungen ab und ist nicht vorhersehbar.

$ECO_LEVEL wirkt sich nicht auf alle Bewegungen aus. Die folgende Tabelle gibt an, auf welche Bewegungen sie sich auswirkt und auf welche nicht:

Bewegung	Auswirkung?
PTP	Ja
LIN	Nein

Bewegung	Auswirkung?
CIRC	Nein
CP-Spline-Bewegungen (Block und Einzelsatz) Mit höherem Fahrprofil	Ja
CP-Spline-Bewegungen (Block und Einzelsatz) Ohne höheres Fahrprofil	Nein
PTP-Spline-Bewegungen (Block und Einzelsatz)	Ja

Wenn ein Programm zurückgesetzt oder abgewählt wird, wird der Energiespar-Modus automatisch ausgeschaltet.

In folgenden Fällen ist der Energiespar-Modus inaktiv, auch wenn er eingeschaltet ist:

- Bei einer SAK-Fahrt
- In einem Konstantfahrbereich beim Spline
- In einem Zeitblock beim Spline

Wenn für Geschwindigkeit und Beschleunigung bereits niedrige Werte programmiert sind, hat $ECO_LEVEL keine oder nahezu keine Auswirkung.

Je nach Robotertyp kann es sein, dass bei bestimmten Konstellationen die Einsparung bei "mittel" und "hoch" gleich oder nahezu gleich ist. (Z. Bsp. bei einer Traglast unter 30 % der Default-Traglast.)

Voraussetzung

- $ADAP_ACC <> #NONE
- $OPT_MOVE <> #NONE

Bei beiden Systemvariablen ist **<> #NONE** bereits die Default-Einstellung.

Syntax

`$ECO_LEVEL=`*Stufe*

Erläuterung der Syntax

Element	Beschreibung
Stufe	Typ: ENUM ■ **#OFF**: Der Energiespar-Modus ist ausgeschaltet. ■ **#LOW**: Geringe Einsparung ■ **#MIDDLE**: Mittlere Einsparung ■ **#HIGH**: Hohe Einsparung

6.11 Arbeitsräume konfigurieren

Für einen Roboter können Arbeitsräume konfiguriert werden. Diese dienen dem Anlagenschutz.

Maximal 8 kartesische (=kubische) und 8 achsspezifische Arbeitsräume können gleichzeitig konfiguriert sein. Die Arbeitsräume dürfen sich überlappen.

(>>> 6.11.1 "Kartesische Arbeitsräume konfigurieren" Seite 178)

(>>> 6.11.2 "Achsspezifische Arbeitsräume konfigurieren" Seite 180)

Es gibt 2 Typen von Arbeitsräumen:

- Nicht erlaubte Räume. Der Roboter darf sich nur außerhalb eines solchen Raums bewegen.
- Erlaubte Räume. Der Roboter darf sich nicht außerhalb eines solchen Raums bewegen.

Welche Reaktionen auftreten, wenn der Roboter einen Arbeitsraum verletzt, ist abhängig von der Konfiguration.

6.11.1 Kartesische Arbeitsräume konfigurieren

Beschreibung

Folgende Parameter definieren die Position und Größe eines kartesischen Arbeitsraums:

- Ursprung des Arbeitsraums in Bezug auf das WORLD-Koordinatensystem
- Abmessungen des Arbeitsraums, ausgehend vom Ursprung

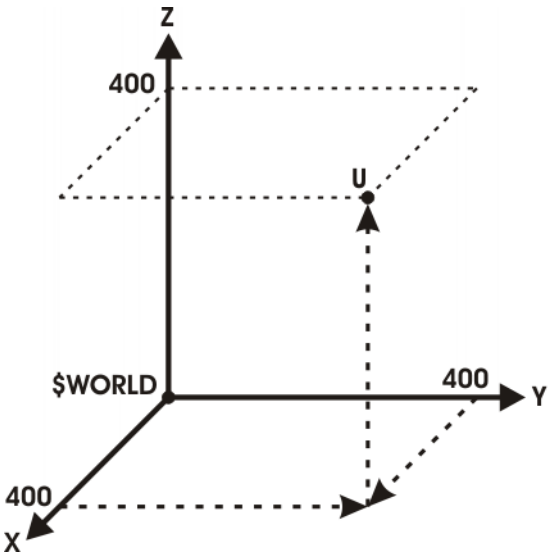


Abb. 6-8: Kartesischer Arbeitsraum, Ursprung U

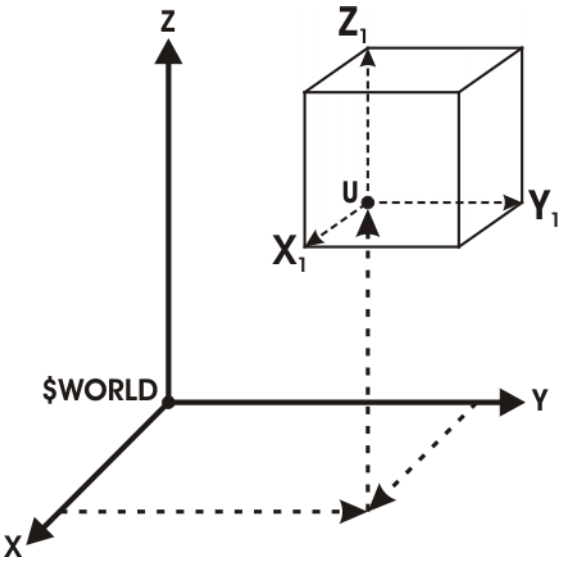


Abb. 6-9: Kartesischer Arbeitsraum, Abmessungen

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Arbeitsraumüberwachung** > **Konfiguration** wählen.
   Das Fenster **Kartesische Arbeitsräume** öffnet sich.
2. Werte eingeben und **Speichern** drücken.
3. **Signal** drücken. Das Fenster **Signale** öffnet sich.
4. In der Gruppe **Kartesisch**: Bei der Nummer des Arbeitsraums den Ausgang eintragen, der gesetzt werden soll, wenn der Arbeitsraum verletzt wird.
5. **Speichern** drücken.

6. Fenster schließen.

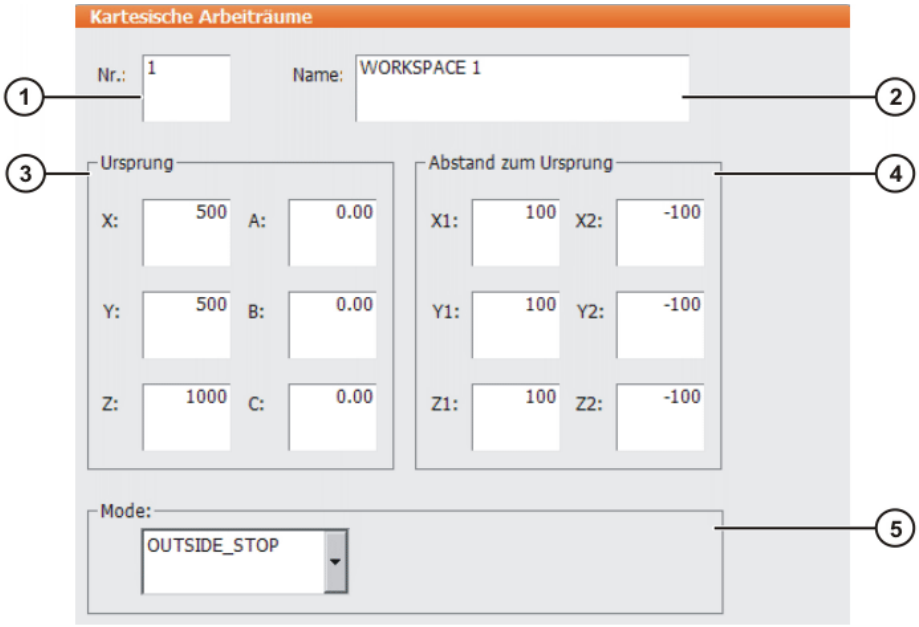


Abb. 6-10: Kartesischen Arbeitsraum konfigurieren

Pos.	Beschreibung
1	Nummer des Arbeitsraums (max. 8)
2	Bezeichnung des Arbeitsraums
3	Ursprung und Orientierung des Arbeitsraums, bezogen auf das WORLD-Koordinatensystem
4	Abmessung des Arbeitsraums in mm
5	Modus (>>> 6.11.3 "Modus für Arbeitsräume" Seite 182)

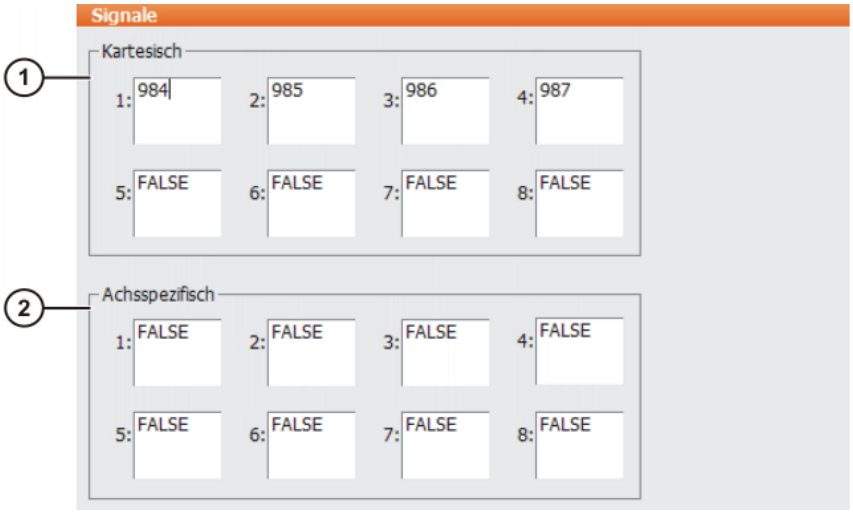


Abb. 6-11: Arbeitsraum Signale

Pos.	Beschreibung
1	Ausgänge für die Überwachung der kartesischen Arbeitsräume
2	Ausgänge für die Überwachung der achsspezifischen Arbeitsräume

Wenn beim Verletzen eines Arbeitsraums kein Ausgang gesetzt werden soll, muss FALSE eingetragen werden.

6.11.2 Achsspezifische Arbeitsräume konfigurieren

Beschreibung

Mit achsspezifischen Arbeitsräumen können die durch die Software-Endschalter festgelegten Bereiche weiter eingeschränkt werden, um den Roboter oder das Werkzeug bzw. Werkstück zu schützen.

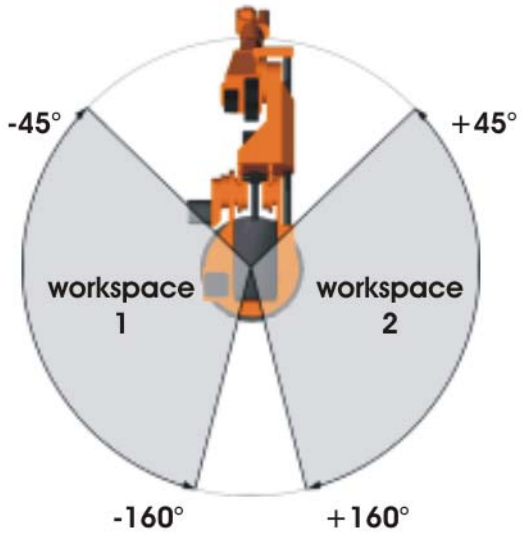


Abb. 6-12: Beispiel achsspezifische Arbeitsräume für A1

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Arbeitsraumüberwachung** > **Konfiguration** wählen.
   Das Fenster **Kartesische Arbeitsräume** öffnet sich.
2. **Achsspez.** drücken, um zum Fenster **Achsspezifische Arbeitsräume** zu wechseln.
3. Werte eingeben und **Speichern** drücken.
4. **Signal** drücken. Das Fenster **Signale** öffnet sich.
5. In der Gruppe **Achsspezifisch**: Bei der Nummer des Arbeitsraums den Ausgang eintragen, der gesetzt werden soll, wenn der Arbeitsraum verletzt wird.
6. **Speichern** drücken.
7. Fenster schließen.

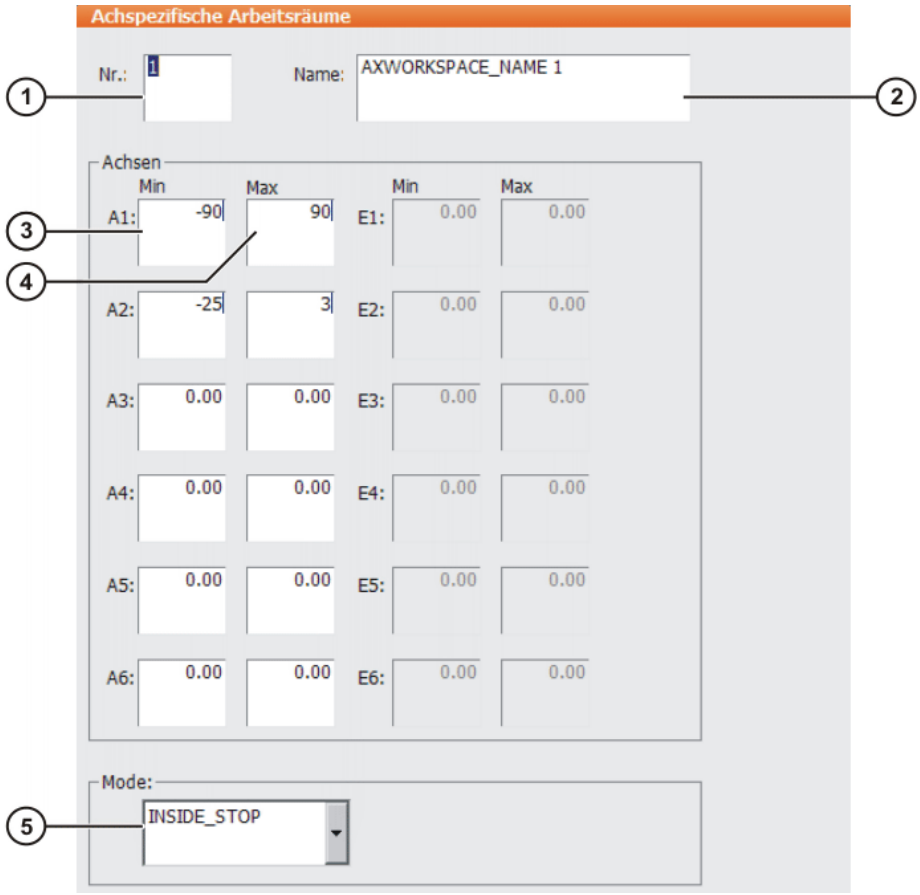


Abb. 6-13: Achsspezifischen Arbeitsraum konfigurieren

Pos.	Beschreibung
1	Nummer des Arbeitsraums (max. 8)
2	Bezeichnung des Arbeitsraums
3	Untergrenze für Achswinkel
4	Obergrenze für Achswinkel
5	Modus (>>> 6.11.3 "Modus für Arbeitsräume" Seite 182)

Wenn bei Pos. 3 und Pos. 4 einer Achse 0 steht, wird die Achse unabhängig vom Modus nicht überwacht.

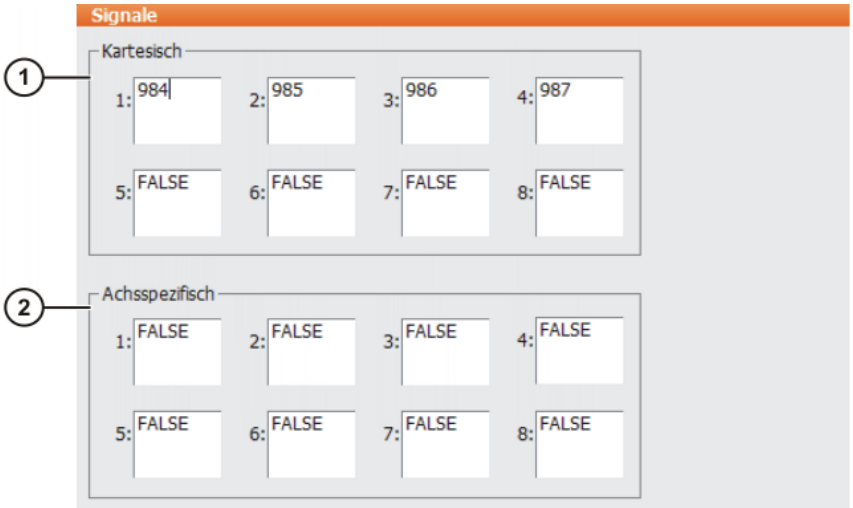


Abb. 6-14: Arbeitsraum Signale

Pos.	Beschreibung
1	Ausgänge für die Überwachung der kartesischen Arbeitsräume
2	Ausgänge für die Überwachung der achsspezifischen Arbeitsräume

Wenn beim Verletzen eines Arbeitsraums kein Ausgang gesetzt werden soll, muss FALSE eingetragen werden.

6.11.3 Modus für Arbeitsräume

Mode	Beschreibung
#OFF	Die Arbeitsraumüberwachung ist ausgeschaltet.
#INSIDE	■ Kartesischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich TCP oder Flansch innerhalb des Arbeitsraums befinden. ■ Achsspezifischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich die Achse innerhalb des Arbeitsraums befindet.
#OUTSIDE	■ Kartesischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich TCP oder Flansch außerhalb des Arbeitsraums befinden. ■ Achsspezifischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich die Achse außerhalb des Arbeitsraums befindet.
#INSIDE_STOP	■ Kartesischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich TCP, Flansch oder Handwurzelpunkt innerhalb des Arbeitsraums befinden. (Handwurzelpunkt = Mittelpunkt der Achse A5) ■ Achsspezifischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich die Achse innerhalb des Arbeitsraums befindet. Zusätzlich wird der Roboter gestoppt und Meldungen werden angezeigt. Der Roboter kann erst wieder verfahren werden, wenn die Arbeitsraumüberwachung ausgeschaltet oder überbrückt wurde. (>>> 4.16 "Arbeitsraumüberwachung überbrücken" Seite 80)
#OUTSIDE_STOP	■ Kartesischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich TCP oder Flansch außerhalb des Arbeitsraums befinden. ■ Achsspezifischer Arbeitsraum: Der definierte Ausgang wird gesetzt, wenn sich die Achse außerhalb des Arbeitsraums befindet. Zusätzlich wird der Roboter gestoppt und Meldungen werden angezeigt. Der Roboter kann erst wieder verfahren werden, wenn die Arbeitsraumüberwachung ausgeschaltet oder überbrückt wurde. (>>> 4.16 "Arbeitsraumüberwachung überbrücken" Seite 80)

6.12 Grenzen für das Umteachen festlegen

Beschreibung Wenn diese Funktion aktiv ist, dürfen bestehende Punkte von den Benutzergruppen Anwender und Bediener nur noch innerhalb der festgelegten Gren-

zen umgeteacht werden. Wenn die Grenzen überschritten werden, erscheint eine Meldung, dass die Änderung nicht möglich ist. Globale Positionen, z. B. die HOME-Position, können von diesen Benutzergruppen dann überhaupt nicht mehr umgeteacht werden.

In der Benutzergruppe Experte können Punkte nach wie vor ohne Einschränkung umgeteacht werden.

Die Grenzen gelten nur für Änderungen, die über die Schaltflächen **Ändern** und **Touch Up** oder **Befehl OK** durchgeführt werden. Änderungen über andere Funktionalitäten, z. B. Vermessen oder Variablenkorrektur, sind für die Benutzergruppen Anwender und Bediener nach wie vor möglich.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Korrekturgrenze Punktkoordinaten** wählen. Das Fenster **Korrekturgrenze Punktkoordinaten** öffnet sich.
2. Die gewünschten Werte eingeben und bei **Korrekturgrenze aktiv** das Häkchen setzen.
3. Das **Schließen**-Symbol berühren. Eine Abfrage wird angezeigt, ob die Änderungen gespeichert werden sollen.
4. Die Abfrage mit **Ja** bestätigen. Die Eingaben werden gespeichert und das Fenster schließt sich.

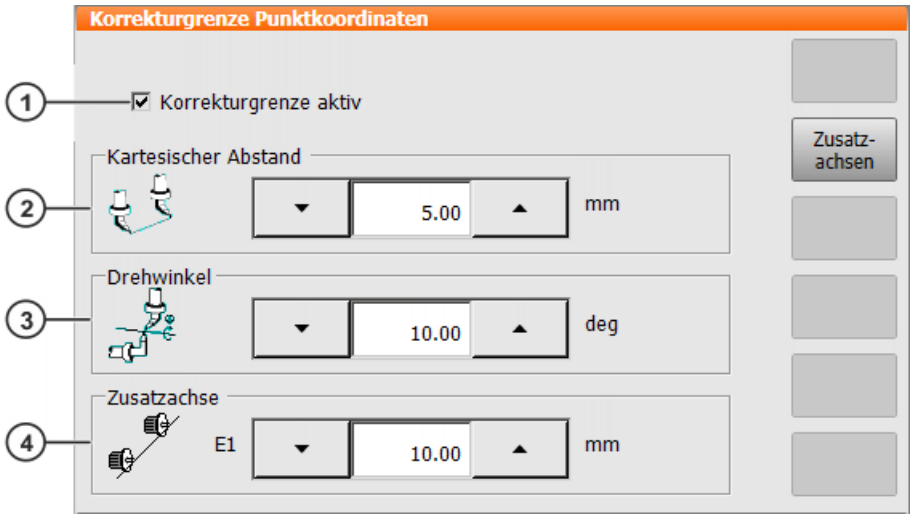


Abb. 6-15: Optionsfenster Korrekturgrenze Punktkoordinaten

Pos.	Beschreibung
1	Mit Häkchen: Die Grenzen sind aktiv. Ohne Häkchen: Die Grenzen sind nicht aktiv.
2	Maximal zulässige Änderung für den X-, Y- und Z-Wert ■ **1.00 … 100.00 mm** Der Wert stellt den Radius einer gedachten Kugel um den ursprünglichen Punkt dar.

Pos.	Beschreibung
3	Maximal zulässige Änderung für den A-, B- und C-Wert ■ **0.00 … 20.00 deg** Wert 0.00 bedeutet: Keine Änderung erlaubt.
4	Dieses Feld wird nur angezeigt, wenn mindestens eine Zusatzachse vorhanden ist. Maximal zulässige Änderung für die angezeigte Zusatzachse. ■ **0.00 … 100.00 mm**

Schaltfläche	Beschreibung
Zusatzachsen	Die Schaltfläche steht nur zur Verfügung, wenn mehr als eine Zusatzachse vorhanden ist. Zeigt die nächste Zusatzachse an.

6.13 Warmfahren

Beschreibung

Wenn ein Roboter bei niedrigen Umgebungstemperaturen startet, führt dies zu einer erhöhten Reibung im Getriebe. Dies kann zur Folge haben, dass der Motorstrom einer Achse (oder auch mehrerer Achsen) seinen Maximalwert erreicht. Der Roboter stoppt dadurch und die Robotersteuerung gibt die Fehlermeldung *Stellgröße <Achsnummer>* aus.

Um dies zu vermeiden, kann der Motorstrom während der Warmlaufphase überwacht werden: Wenn er einen definierten Wert erreicht, reduziert die Robotersteuerung die Verfahrgeschwindigkeit. Dadurch sinkt der Motorstrom.

Die Überwachung bezieht sich auf PTP-Bewegungen und PTP-CP-Überschleifsätze.
Andere Bewegungen werden nicht überwacht und ihre Geschwindigkeit wird nicht reduziert. Das sind: LIN, CIRC und alle Spline-Bewegungen (CP und PTP).

6.13.1 Warmfahren konfigurieren

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Datei R1\Mada\$machine.dat öffnen.
2. Die zugehörigen Systemvariablen auf die gewünschten Werte setzen.
   (>>> 6.13.3 "Systemvariablen für das Warmfahren" Seite 186)
3. Datei schließen. Die Sicherheitsabfrage, ob die Änderungen gespeichert werden sollen, mit **Ja** beantworten.

6.13.2 Ablauf Warmfahren

Voraussetzung

- $WARMUP_RED_VEL = TRUE
- Betriebsart AUT oder AUT EXT
- Der Roboter wird als kalt angesehen. Dies ist in folgenden Fällen der Fall:
  - Kaltstart
  - Oder $COOLDOWN_TIME ist abgelaufen.
  - Oder $WARMUP_RED_VEL wurde von FALSE auf TRUE gesetzt.

Beispiel

Ablauf am Beispiel folgender Werte in der $machine.dat:

```
BOOL $WARMUP_RED_VEL = TRUE
REAL $WARMUP_TIME = 30.0
REAL $COOLDOWN_TIME = 120.0
INT $WARMUP_CURR_LIMIT = 95
INT $WARMUP_MIN_FAC = 60
REAL $WARMUP_SLEW_RATE = 5.0
```

1. Der kalte Roboter startet. Die Motorströme werden 30 min lang ($WARMUP_TIME) überwacht.
2. Wenn der Motorstrom einer Achse 95 % ($WARMUP_CURR_LIMIT) des maximal zulässigen Motorstroms überschreitet, schlägt die Überwachung an. Die Robotersteuerung gibt daraufhin die Meldung *Warmfahren aktiv* aus und reduziert den internen Override. Der Roboter wird langsamer und der Motorstrom sinkt. Der auf der Bedienoberfläche angezeigte Programm-Override bleibt unverändert!

   Der interne Override wird maximal auf 60 % ($WARMUP_MIN_FAC) des programmierten Overrides reduziert. Wie schnell der interne Override reduziert wird, kann nicht beeinflusst werden.
3. Wenn die Überwachung nicht mehr anschlägt, erhöht die Robotersteuerung den internen Override wieder. Dies ist in der Regel der Fall, bevor das Minimum $WARMUP_MIN_FAC erreicht wurde! Der Roboter wird wieder schneller.

   Die Robotersteuerung tastet sich im Sekundentakt an den programmierten Override heran. $WARMUP_SLEW_RATE bestimmt die Steigerungsrate. Im Beispiel wird der interne Override um 5 % pro Sekunde erhöht.
4. Es ist möglich, dass der Roboter noch nicht warm genug ist und dass der Motorstrom deshalb noch einmal das Maximum $WARMUP_CURR_LIMIT überschreitet. Die Robotersteuerung reagiert (innerhalb der Zeit $WARMUP_TIME) wie beim ersten Mal.
5. Wenn der Roboter warm genug ist, dass die Robotersteuerung den internen Override so weit erhöhen kann, dass er wieder dem programmierten Override entspricht, blendet die Robotersteuerung die Meldung *Warmfahren aktiv* aus.
6. Nach 30 min ($WARMUP_TIME) wird der Roboter als warm angesehen und die Motorströme werden nicht mehr überwacht.

LOG-Datei

Folgende Ereignisse werden in der Datei Warmup.LOG (Verzeichnis KRC:\Roboter\Log) protokolliert:

Eintrag	Bedeutung
Monitoring active	Die Motorströme werden überwacht.
Monitoring inactive	Die Motorströme werden nicht überwacht.
Controlling active	Die Geschwindigkeit wird reduziert.
Controlling inactive	Die Geschwindigkeit entspricht wieder dem programmierten Override.

Beispiel:

```
...
Date: 21.08.08 Time: 14:46:57 State: Monitoring active
Date: 21.08.08 Time: 14:54:06 State: Controlling active
Date: 21.08.08 Time: 14:54:07 State: Controlling inactive
Date: 21.08.08 Time: 18:23:43 State: Monitoring inactive
...
```

6.13.3 Systemvariablen für das Warmfahren

Die Systemvariablen für das Warmfahren können ausschließlich in der Datei $machine.dat (Verzeichnis KRC:\R1\MADA) geändert werden.

Liegt einer der Werte außerhalb des zulässigen Bereichs, ist die Warmfahr-Funktionalität nicht aktiv und es erfolgt keine Geschwindigkeitsreduzierung.

Systemvariable	Beschreibung
$WARMUP_RED_VEL	■ **TRUE**: Warmfahr-Funktionalität ist aktiviert. ■ **FALSE**: Warmfahr-Funktionalität ist deaktiviert. (Default) Wenn $WARMUP_RED_VEL von FALSE auf TRUE gesetzt wird, setzt dies die Laufzeit des Roboters auf Null. Der Roboter wird als kalt angesehen, gleichgültig, wie lange er vorher in Regelung war.
$WARMUP_TIME	Zeit, während der die Motorströme durch die Warmfahr-Funktionalität überwacht werden Wenn der kalte Roboter startet, wird ein Laufzeitwert hochgezählt. Wenn der Roboter nicht in Regelung ist, wird der Wert heruntergezählt. Wenn die Laufzeit größer ist als $WARMUP_TIME, wird der Roboter als warm betrachtet und die Motorströme werden nicht mehr überwacht. ■ > 0,0 min
$COOLDOWN_TIME	Wenn der warme Roboter nicht in Regelung ist, wird ein Stillstandswert hochgezählt. Wenn der Roboter in Regelung ist, wird der Wert heruntergezählt. Wenn die Stillstandszeit größer ist als $COOLDOWN_TIME, wird der Roboter als kalt betrachtet und die Motorströme werden überwacht. (Vorausgesetzt $WARMUP_RED_VEL = TRUE.) Wenn die Steuerung eines warmen Roboters heruntergefahren und mit Warmstart neu gestartet wird, zählt die Zeit, während der die Steuerung ausgeschaltet war, als Stillstandszeit. ■ > 0,0 min
$WARMUP_CURR_LIMIT	Maximal zulässiger Motorstrom beim Warmfahren (bezogen auf den regulären maximal zulässigen Motorstrom) Regulärer maximal zulässiger Motorstrom = ($CURR_LIM * $CURR_MAX) / 100 ■ 0 … 100 %
$WARMUP_MIN_FAC	Minimum für die Override-Reduzierung durch die Warmfahr-Funktionalität Der interne Override wird maximal auf den hier definierten Faktor des programmierten Overrides reduziert. ■ 0 … 100 %
$WARMUP_SLEW_RATE	Steigerungsrate für die Geschwindigkeitserhöhung Wenn die Überwachung nicht mehr anschlägt, erhöht die Robotersteuerung den internen Override wieder. Die Steigerungsrate wird hier definiert. ■ > 0,0 %/s

6.14 Kollisionserkennung

Die Kollisionserkennung ist eine Erweiterung und Verbesserung der Momentenüberwachung.
Die Momentenüberwachung steht weiterhin zur Verfügung. Programme, in denen bereits eine Momentenüberwachung definiert ist, können weiterhin mit der bisherigen Vorgehensweise bearbeitet werden.
Alternativ kann man in solchen Programmen die Zeilen mit der Momentenüberwachung löschen und stattdessen die Kollisionserkennung verwenden.
Die Kollisionserkennung darf nicht zusammen mit der Momentenüberwachung in einem Programm verwendet werden.

Beschreibung

Wenn ein Roboter mit einem Objekt kollidiert, erhöht die Robotersteuerung die Achsmomente, um den Widerstand zu überwinden. Roboter, Werkzeug oder andere Teile können hierbei beschädigt werden.

Die Kollisionserkennung verringert das Risiko solcher Schäden. Sie überwacht die Achsmomente. Wenn diese von einem bestimmten Toleranzbereich abweichen, treten folgende Reaktionen ein:

- Der Roboter stoppt mit einem STOP 1.
- Die Robotersteuerung ruft das Programm **tm_useraction** auf. Es befindet sich im Ordner **Program** und enthält die Anweisung HALT. Alternativ kann der Benutzer im Programm **tm_useraction** andere Reaktionen programmieren.

  (>>> 6.14.4 "Programm tm_useraction bearbeiten" Seite 191)

Die Robotersteuerung ermittelt den Toleranzbereich automatisch. (Ausnahme: In der Betriebsart T1 werden keine Werte ermittelt.) In der Regel muss ein Programm 2- bis 3-mal durchlaufen werden, bis die Robotersteuerung einen praxistauglichen Toleranzbereich ermittelt hat. Für den von der Robotersteuerung ermittelten Toleranzbereich kann der Benutzer über die Bedienoberfläche einen Offset definieren.

Wenn der Roboter über einen längeren Zeitraum (z. B. Wochenende) nicht betrieben wird, kühlen sich Motoren, Getriebe etc. ab. Während der ersten Fahrten nach einer solchen Pause werden andere Achsmomente benötigt als bei einem betriebswarmen Roboter. Die Robotersteuerung passt die Kollisionserkennung automatisch an die veränderte Temperatur an.

Voraussetzung

- Um die Kollisionserkennung verwenden zu können, muss die Beschleunigungsanpassung eingeschaltet sein.

  Die Beschleunigungsanpassung ist eingeschaltet, wenn die Systemvariable $ADAP_ACC **ungleich** #NONE ist. (Dies ist die Defaulteinstellung.) Die Systemvariable ist in der Datei C:\KRC\Roboter\KRC\R1\MaDa\$ROBCOR.DAT zu finden.
- Der Toleranzbereich wird nur für Bewegungssätze ermittelt, die komplett abgefahren wurden.
- Um die Kollisionserkennung für eine Bewegung einzuschalten, muss bei der Programmierung der Parameter **Kollisionserkennung** auf TRUE gesetzt wurde. Dies ist im Programmcode am Zusatz `CD` zu erkennen:

```
PTP P2  Vel= 100 % PDAT1 Tool[1] Base[1] CD
```

Der Parameter **Kollisionserkennung** steht nur zur Verfügung, wenn die Bewegung über ein Inline-Formular programmiert wird.
Informationen zur Kollisionserkennung für Bewegungen, die ohne Inline-Formulare programmiert wurden, sind in der Dokumentation KUKA.ExpertTech zu finden.

Einschränkungen

- Für HOME-Positionen und andere globale Positionen ist keine Kollisionserkennung möglich.
- Für Zusatzachsen ist keine Kollisionserkennung möglich.
- Beim Rückwärtsfahren ist keine Kollisionserkennung möglich.
- In der Betriebsart T1 ist keine Kollisionserkennung möglich.
- Wenn der Roboter steht, so entstehen beim Anfahren sehr hohe Achsmomente. In der Anfahrphase (ca. 700 ms) werden die Achsmomente deswegen nicht überwacht.
- Nach einer Änderung des Programm-Overrides reagiert die Kollisionserkennung 2 bis 3 Programmläufe lang erheblich unsensibler. Danach hat die Robotersteuerung den Toleranzbereich an den neuen Programm-Override angepasst.

Systemvariablen

Systemvariable	Beschreibung
$TORQ_DIFF $TORQ_DIFF2	Beim Programmlauf werden automatisch die Werte von $TORQ_DIFF (Kraftmoment) und $TORQ_DIFF2 (Stoßmoment) ermittelt. Diese werden mit den Werten aus dem vorigen Programmlauf oder mit Default-Werten verglichen. Der größte Wert wird gespeichert. Die Werte werden immer ermittelt, auch wenn die Kollisionserkennung deaktiviert ist. Wenn die Kollisionserkennung aktiv ist, vergleicht das System während der Bewegung die Werte von $TORQ_DIFF und $TORQ_DIFF2 mit den gespeicherten Werten.
$TORQMON_DEF[1] … $TORQMON_DEF[6]	Werte für den Toleranzbereich im Programmbetrieb (pro Achse)* Datei KRC:\STEU\Mada\$custom.dat
$TORQMON_COM_DEF[1] … $TORQMON_COM_DEF[6]	Werte für den Toleranzbereich beim manuellen Verfahren (pro Achse)* Datei KRC:\STEU\Mada\$custom.dat
$COLL_ENABLE	Signalvereinbarung. Dieser Ausgang wird gesetzt, wenn der Wert einer der Variablen $TORQMON_DEF[…] kleiner als 200 ist. Datei: KRC:\STEU\Mada\$machine.dat
$COLL_ALARM	Signalvereinbarung. Dieser Ausgang wird gesetzt, wenn die Meldung 117, "Kollisionserkennung Achse <Achsnummer>", angezeigt wird. Der Ausgang bleibt solange gesetzt, wie $STOPMESS ansteht. Datei: KRC:\STEU\Mada\$machine.dat
$TORQMON_TIME	Ansprechzeit für die Kollisionserkennung. Einheit: Millisekunden. Defaultwert: 0.0 Datei: C:\KRC\Roboter\ KRC\Steu\MaDa\$CUSTOM.DAT.

*Die Breite des Toleranzbereichs ergibt sich aus dem maximalen Moment in [Nm] multipliziert mit dem Wert von $TORQMON_… . Der Defaultwert ist 200. Einheit: Prozent.

6.14.1 Toleranzbereich ermitteln und Kollisionserkennung aktivieren

Voraussetzung

- Die Beschleunigungsanpassung ist eingeschaltet.
- Die Lastdaten wurden korrekt eingegeben.
- Im Programm ist bei allen Bewegungen, die überwacht werden sollen, der Parameter **Kollisionserkennung** auf TRUE.

- Bei Bedarf: Im Programm **tm_useraction** wurde die gewünschte Reaktion auf eine Kollision programmiert.

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Kollisionserkennung** wählen.
   (>>> 6.14.3 "Optionsfenster Kollisionserkennung" Seite 190)
2. Im Feld **KCP** muss der Eintrag **MonOff** stehen. Wenn dies nicht der Fall ist, **Deaktivieren** drücken.
3. Programm starten und mehrmals durchlaufen lassen. Nach 2 bis 3 Programmläufen hat die Robotersteuerung einen praxistauglichen Toleranzbereich ermittelt.
4. **Aktivieren** drücken. Im Fenster **Kollisionserkennung** steht jetzt im Feld **KCP** der Eintrag **MonOn**.
   Die Konfiguration mit **Schließen** sichern.

Bei Bedarf kann der Benutzer für den Toleranzbereich einen Offset definieren. (>>> 6.14.2 "Offset für Toleranzbereich definieren" Seite 189)

In folgenden Fällen muss der Toleranzbereich neu ermittelt werden:
- Die Geschwindigkeit wurde verändert.
- Punkte wurden verändert, hinzugefügt oder entfernt.

6.14.2 Offset für Toleranzbereich definieren

Beschreibung

Für den Toleranzbereich kann ein Offset für das Kraftmoment und für das Stoßmoment definiert werden. Je kleiner der Offset ist, desto sensibler reagiert die Kollisionserkennung. Je größer der Offset ist, desto unsensibler reagiert die Kollisionserkennung.

Kraftmoment: Das Kraftmoment ist wirksam, wenn dem Roboter ein andauernder Widerstand entgegensetzt wird. Beispiele:

- Der Roboter kollidiert mit einer Wand und drückt gegen die Wand.
- Der Roboter kollidiert mit einem Container. Der Roboter drückt gegen den Container und bewegt diesen.

Stoßmoment: Das Stoßmoment ist wirksam, wenn dem Roboter ein kurzer Widerstand entgegensetzt wird. Beispiel:

- Der Roboter kollidiert mit einem Schild, das durch den Aufprall weggeschleudert wird.

Wenn die Kollisionserkennung zu sensibel reagiert, nicht sofort den Offset erhöhen. Stattdessen zuerst den Toleranzbereich neu ermitteln und testen, ob die Kollisionserkennung nun wie gewünscht reagiert.
(>>> 6.14.1 "Toleranzbereich ermitteln und Kollisionserkennung aktivieren" Seite 188)

Vorgehensweise

1. Programm anwählen.
2. Im Hauptmenü **Konfiguration** > **Extras** > **Kollisionserkennung** wählen.
   (>>> 6.14.3 "Optionsfenster Kollisionserkennung" Seite 190)
3. Der Offset für eine Bewegung kann bei laufendem Programm geändert werden:
   Wenn die gewünschte Bewegung im Fenster **Kollisionserkennung** angezeigt wird, auf eine Pfeiltaste im Feld **Kraftmoment** oder **Stoßmoment** drücken. Das Fenster bleibt auf der Bewegung stehen und der Offset kann geändert werden.
   Alternativ kann eine Satzanwahl auf die gewünschte Bewegung ausgeführt werden.

4. Die Änderung mit **Speichern** übernehmen.
5. Die Konfiguration mit **Schließen** sichern.
6. Ursprüngliche Betriebsart und Programmablaufart einstellen.

6.14.3 Optionsfenster Kollisionserkennung

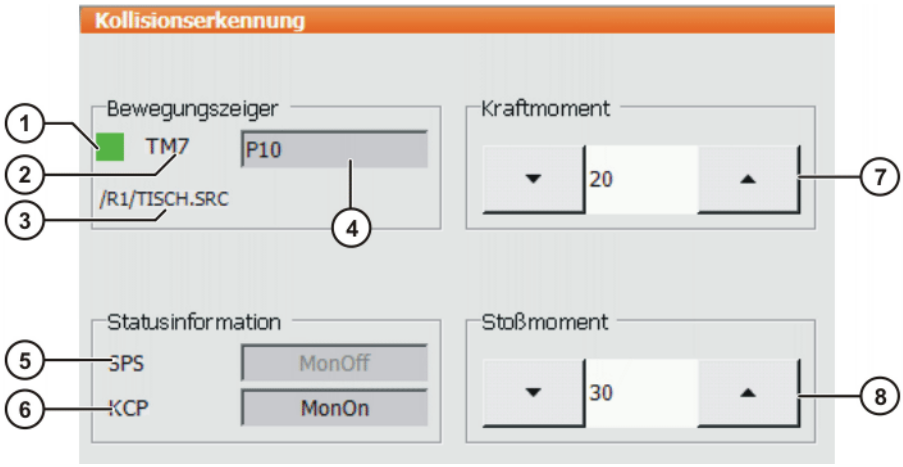


Abb. 6-16: Optionsfenster Kollisionserkennung

Die Angaben im Optionsfenster **Kollisionserkennung** beziehen sich nicht immer auf die aktuelle Bewegung. Insbesondere bei kurzen Punktabständen und überschliffenen Bewegungen sind Abweichungen möglich.

Pos.	Beschreibung
1	Zeigt den Zustand der aktuellen Bewegung an: ■ Rot: Die aktuelle Bewegung wird nicht überwacht. ■ Grün: Die aktuelle Bewegung wird überwacht. ■ Orange: Eine Pfeiltaste im Feld **Kraftmoment** oder **Stoßmoment** wurde gedrückt. Das Fenster bleibt auf der Bewegung stehen und der Offset kann geändert werden. Die Änderung kann dann mit **Speichern** übernommen werden. ■ Gerastert: In der Regel muss ein Programm 2- bis 3-mal durchlaufen werden, bis die Robotersteuerung einen praxistauglichen Toleranzbereich ermittelt hat. Während dieser Lernphase ist diese Anzeige gerastert.
2	Nummer der Variablen TMx Für jeden Bewegungssatz, bei dem der Parameter **Kollisionserkennung** auf TRUE ist, legt die Robotersteuerung eine Variable TMx an. TMx enthält alle Werte für den Toleranzbereich dieses Bewegungssatzes. Wenn sich 2 Bewegungssätze auf den gleichen Punkt Px beziehen, legt die Robotersteuerung 2 Variablen TMx an.
3	Pfad und Name des angewählten Programms
4	Punktname

Pos.	Beschreibung
5	Dieses Feld ist nur in der Betriebsart "Automatik Extern" aktiv. Ansonsten ist es gegraut. **MonOn**: Die Kollisionserkennung wurde von der SPS aktiviert. Wenn die Kollisionserkennung über die SPS aktiviert wird, sendet die SPS das Eingangssignal **sTQM_SPSACTIVE** an die Robotersteuerung. Die Robotersteuerung antwortet mit dem Ausgangssignal **sTQM_SPSSTATUS**. Die Signale sind in der Datei $config.dat definiert. **Hinweis:** Im Automatik Extern-Betrieb ist die Kollisionserkennung nur aktiv, wenn sowohl das Feld **SPS** als auch das Feld **KCP** den Eintrag **MonOn** anzeigen.
6	**MonOn**: Die Kollisionserkennung wurde vom smartPAD aus aktiviert. **Hinweis:** Im Automatik Extern-Betrieb ist die Kollisionserkennung nur aktiv, wenn sowohl das Feld **SPS** als auch das Feld **KCP** den Eintrag **MonOn** anzeigen.
7	Offset für das Kraftmoment. Je kleiner der Offset ist, desto sensibler reagiert die Kollisionserkennung. Defaultwert: 20. Wenn eine Pfeiltaste gedrückt wird, bleibt das Fenster auf der Bewegung stehen und der Offset kann geändert werden. Die Änderung kann dann mit **Speichern** übernommen werden. ■ **N.A.**: Für diese Bewegung ist die Option **Kollisionserkennung** im Inline-Formular auf FALSE.
8	Offset für das Stoßmoment. Je kleiner der Offset ist, desto sensibler reagiert die Kollisionserkennung. Defaultwert: 30. Wenn eine Pfeiltaste gedrückt wird, bleibt das Fenster auf der Bewegung stehen und der Offset kann geändert werden. Die Änderung kann dann mit **Speichern** übernommen werden. ■ **N.A.**: Für diese Bewegung ist die Option **Kollisionserkennung** im Inline-Formular auf FALSE.

Schaltfläche	Beschreibung
Aktivieren	Aktiviert die Kollisionserkennung. Diese Schaltfläche wird nicht angezeigt, wenn das Kraft- oder Stoßmoment geändert wurde, aber die Änderungen noch nicht gespeichert wurden.
Deaktivieren	Deaktiviert die Kollisionserkennung. Diese Schaltfläche wird nicht angezeigt, wenn das Kraft- oder Stoßmoment geändert wurde, aber die Änderungen noch nicht gespeichert wurden.
Speichern	Übernimmt Änderungen am Kraft- und/oder Stoßmoment.
Abbrechen	Verwirft Änderungen am Kraft- und/oder Stoßmoment.

6.14.4 Programm tm_useraction bearbeiten

Beschreibung Defaultmäßig enthält das Programm **tm_useraction** die Anweisung HALT. Bei Bedarf kann der Benutzer andere Anweisungen programmieren.

Vorbereitung

- Bei den Eigenschaften des Programms **tm_useraction** die Eigenschaft **Sichtbar** aktivieren:

  Hierzu in der Registerkarte **Modul Info** bei der Checkbox **Sichtbar** das Häkchen setzen.

  (>>> 7.4.2 "Eigenschaften von Dateien und Ordnern anzeigen oder ändern" Seite 241)

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1, T2 oder AUT
- Submit-Interpreter ist abgewählt.

Vorgehensweise

1. **tm_useraction.src** in der Dateiliste (= rechter Bereich des Navigators) markieren.
2. Auf die Schaltfläche **Öffnen** drücken. Das Programm wird im Editor angezeigt.
3. Die gewünschten Änderungen vornehmen.

> **tm_useraction.src** wird von der Robotersteuerung über einen Interrupt aufgerufen. Bei der Programmierung müssen deshalb die Einschränkungen, die für Interrupt-Programme gelten, beachtet werden.

4. Das Programm schließen.

   Um die Änderungen zu übernehmen, die Sicherheitsabfrage mit **Ja** beantworten.
5. Empfehlung: Die Eigenschaft **Sichtbar** wieder deaktivieren.

   Grund:
   - **Sichtbar** aktiv: Wenn die Robotersteuerung **tm_useraction** aufruft, zeigt der Satzzeiger dieses Programm an.
   - **Sichtbar** inaktiv: Wenn die Robotersteuerung **tm_useraction** aufruft, zeigt der Satzzeiger die Stelle an, an der das Hauptprogramm unterbrochen wurde. Dies ist in der Regel hilfreicher bei der Fehlersuche.

6.14.5 Momentenüberwachung

> Die Kollisionserkennung ist eine Erweiterung und Verbesserung der Momentenüberwachung.
> Die Momentenüberwachung steht weiterhin zur Verfügung. Programme, in denen bereits eine Momentenüberwachung definiert ist, können weiterhin mit der bisherigen Vorgehensweise bearbeitet werden.
> Alternativ kann man in solchen Programmen die Zeilen mit der Momentenüberwachung löschen und stattdessen die Kollisionserkennung verwenden.
> Die Kollisionserkennung darf nicht zusammen mit der Momentenüberwachung in einem Programm verwendet werden.

Beschreibung

Unterschiede der Momentenüberwachung zur Kollisionserkennung:

- Der Toleranzbereich wird nicht automatisch von der Robotersteuerung ermittelt, sondern muss vom Benutzer definiert werden.
- Der Toleranzbereich bezieht sich nur auf das Kraftmoment. Für das Stoßmoment können keine Werte definiert werden.
- Die Robotersteuerung kann den Toleranzbereich nicht automatisch an veränderte Temperaturen anpassen.
- Wenn eine Kollision erkannt wird, stoppt der Roboter mit einem STOP 1. Es ist nicht möglich, ein benutzerdefiniertes Programm aufzurufen.

Übersicht

Schritt	Beschreibung
1	Geeignete Werte für die Momentenüberwachung ermitteln. (>>> 6.14.5.1 "Werte für Momentenüberwachung ermitteln" Seite 193)
2	Momentenüberwachung programmieren. (>>> 6.14.5.2 "Momentenüberwachung programmieren" Seite 193)

6.14.5.1 Werte für Momentenüberwachung ermitteln

Beschreibung

Über die Systemvariable $TORQ_DIFF[…] kann die maximale aufgetretene Momentenabweichung in Prozent ermittelt werden.

Vorgehensweise

1. Im Hauptmenü **Anzeige** > **Variable** > **Einzeln** wählen.
2. Variable $TORQ_DIFF[…] auf den Wert '0' setzen.
3. Bewegungssatz ausführen und Variable erneut auslesen. Der Wert entspricht der maximalen Momentenabweichung.
4. Die Variable für die Überwachung der Achse auf diesen Wert setzen, zuzüglich einer Sicherheit von 5 - 10%.

Der Variablen $TORQ_DIFF[…] kann nur der Wert '0' zugewiesen werden.

6.14.5.2 Momentenüberwachung programmieren

Voraussetzung

- Um die Momentenüberwachung verwenden zu können, muss die Beschleunigungsanpassung eingeschaltet sein. Die Beschleunigungsanpassung ist eingeschaltet, wenn die Systemvariable $ADAP_ACC **ungleich** #NONE ist. (Dies ist die Defaulteinstellung.) Die Systemvariable ist in der Datei C:\KRC\Roboter\KRC\R1\MaDa\$ROBCOR.DAT zu finden.
- Programm ist angewählt.

Vorgehensweise

1. Cursor in die Zeile vor der Bewegung setzen, für die die Momentenüberwachung programmiert werden soll.
2. Menüfolge **Befehle** > **Bewegungsparameter** > **Momentüberwachung** wählen. Ein Inline-Formular öffnet sich.

Abb. 6-17

3. Im Feld **TORQMON** den Eintrag **SetLimits** auswählen.

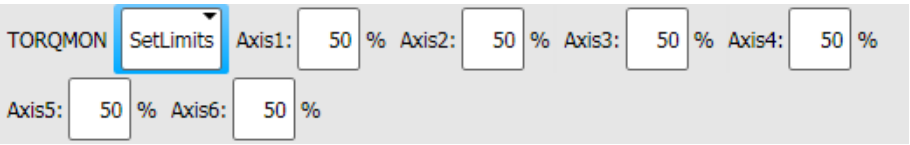


Abb. 6-18

4. Für jede Achse eingeben, um wieviel das Soll- vom Istmoment abweichen darf.
5. **Befehl OK** drücken.
6. Wenn für die Momentenüberwachung eine Ansprechzeit definiert werden soll:

   Die Variable $TORQMON_TIME auf den gewünschten Wert setzen. Einheit: Millisekunden. Defaultwert: 0.

In folgenden Fällen werden die Werte automatisch auf den Defaultwert 200 zurückgesetzt:

- Reset
- Satzanwahl
- Programmabwahl

6.15 Toleranzen für das Vermessen festlegen

> Die Default-Werte nur in Ausnahmefällen ändern. Andernfalls können vermehrt Fehlermeldungen auftreten und Ungenauigkeiten die Folge sein.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

- Im Hauptmenü **Inbetriebnahme** > **Vermessen** > **Toleranzen** wählen.

Beschreibung

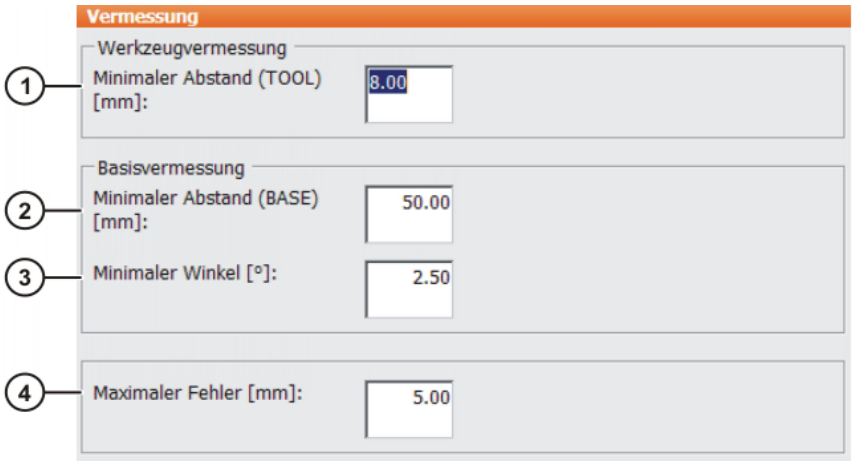


Abb. 6-19: Fehlertoleranzen - Default

Pos.	Beschreibung
1	Kleinster Abstand bei der Werkzeugvermessung ■ **0 … 200 mm**
2	Kleinster Abstand bei der Basisvermessung ■ **0 … 200 mm**
3	Kleinster Winkel zwischen den Geraden durch die 3 Messpunkte bei der Basisvermessung ■ **0 … 360°**
4	Größter Fehler bei der Berechnung ■ **0 … 200 mm**

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Default	Stellt die Default-Einstellungen wieder her. Die Daten müssen anschließend mit **OK** gespeichert werden.

6.16 Rückwärtsfahren konfigurieren

Beschreibung

Die im Folgenden beschriebenen Konfigurationsmöglichkeiten gelten für das Rückwärtsfahren über die Start-Rückwärts-Taste. Sie gelten nicht für andere

Rückwärts-Funktionalitäten, z. B. für Rückwärtsbewegungen im Rahmen von Fehlerstrategien in Technologiepaketen.

Für das Rückwärtsfahren über die Start-Rückwärts-Taste gelten folgende Default-Einstellungen:

- Das Rückwärtsfahren ist aktiv.
- Der Puffer kann maximal 30 Bewegungen speichern.
- Wenn man das Rückwärtsfahren startet, zeigt die Robotersteuerung dies nicht durch eine Meldung an.

Wenn andere Einstellungen gewünscht sind, müssen diese in die KrcConfig.XML eingetragen werden.

In dieser Dokumentation sind weiterführende Informationen zum Rückwärtsfahren zu finden.
(>>> 8.12 "Rückwärtsfahren über die Start-Rückwärts-Taste" Seite 280)

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1, T2 oder AUT

Vorgehensweise

1. Im Verzeichnis C:\KRC\ROBOTER\Config\User\Common die Datei KrcConfig.XML öffnen.
2. Vor der letzten Zeile, d. h. vor `</KrcConfig>`, den Eintrag BACKWARD_STEP einfügen.
3. Die Parameter auf die gewünschten Werte setzen.
4. Die KrcConfig.XML schließen. Die Sicherheitsabfrage, ob die Änderungen gespeichert werden sollen, mit **Ja** beantworten.
5. Die Robotersteuerung neu starten, mit den Einstellungen **Kaltstart** und **Dateien neu einlesen**.

BACKWARD_STEP

```
176    </KRLDIAG>

177  <BACKWARD_STEP ENABLE="true" MOVEMENTS="20"
    ↳ BACKWARD_WARNING="true"/>

178  </KrcConfig>
```

Abb. 6-20: Beispiel: BACKWARD_STEP

Was in der Zeile vor BACKWARD_STEP steht, ist für das Rückwärtsfahren nicht relevant und kann von diesem Beispiel abweichen.

Wenn man nicht alle Parameter von BACKWARD_STEP auflistet, gelten für die nicht gelisteten die Default-Werte.

Wenn man BACKWARD_STEP wieder aus der KrcConfig.XML entfernt, gelten für alle Parameter wieder die Default-Werte.

Parameter	Beschreibung / Default
ENABLE	Typ: BOOL ■ **TRUE** (= Default): Aktiv, d. h. Rückwärtsfahren über die Start-Rückwärts-Taste ist möglich. ■ **FALSE**: Inaktiv, d. h. Rückwärtsfahren über die Start-Rückwärts-Taste ist nicht möglich.
MOVE-MENTS	Typ: INT Maximale Anzahl der Bewegungen, die für das Rückwärtsfahren aufgezeichnet werden ■ **0 … 60** (Default = 30)
BACKWARD_WARNING	Typ: BOOL ■ **TRUE**: Beim ersten Drücken der Start-Rückwärts-Taste nach dem Vorwärtsfahren gibt die Robotersteuerung folgende Meldung aus: *Achtung ! Roboter fährt rückwärts*. Der Benutzer muss die Meldung quittieren und erneut die Start-Rückwärts-Taste drücken. ■ **FALSE** (= Default): Keine Meldung

6.17 Automatik Extern konfigurieren

Beschreibung

Wenn Roboterprozesse von einer übergeordneten Steuerung gesteuert werden sollen (z. B. von einer SPS), dann geschieht dies über die Schnittstelle Automatik Extern.

Über die Schnittstelle Automatik Extern übermittelt die übergeordnete Steuerung an die Robotersteuerung die Signale für die Roboterprozesse (z. B. Fahrfreigabe, Fehlerquittung, Programmstart etc.). Die Robotersteuerung übermittelt an die übergeordnete Steuerung Informationen über Betriebs- und Störzustände.

Übersicht

Um die Schnittstelle Automatik Extern nutzen zu können, müssen folgende Konfigurationen vorgenommen werden:

Schritt	Beschreibung
1	Programm CELL.SRC konfigurieren. (>>> 6.17.1 "CELL.SRC konfigurieren" Seite 196)
2	Ein-/Ausgänge der Schnittstelle Automatik Extern konfigurieren. (>>> 6.17.2 "Automatik Extern Ein-/Ausgänge konfigurieren" Seite 198)
3	Nur, wenn Fehlernummern an die übergeordnete Steuerung übertragen werden sollen: Datei P00.DAT konfigurieren. (>>> 6.17.3 "Fehlernummern an übergeordnete Steuerung übertragen" Seite 204)

6.17.1 CELL.SRC konfigurieren

Beschreibung

Im Automatik Extern-Betrieb werden Programme über das Programm CELL.SRC aufgerufen.

Programm

```
1   DEF   CELL ( )
      …
6     INIT
```

```
7    BASISTECH INI
8    CHECK HOME
9    PTP HOME  Vel= 100 % DEFAULT
10    AUTOEXT INI
11    LOOP
12      P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
13      SWITCH  PGNO ; Select with Programnumber
14
15      CASE 1
16        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17        ;EXAMPLE1 ( ) ; Call User-Program
18
19      CASE 2
20        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
21        ;EXAMPLE2 ( ) ; Call User-Program
22
23      CASE 3
24        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
25        ;EXAMPLE3 ( ) ; Call User-Program
26
27      DEFAULT
28        P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
29      ENDSWITCH
30    ENDLOOP
31  END
```

Zeile	Beschreibung
12	Die Robotersteuerung ruft die Programmnummer von der übergeordneten Steuerung ab.
15	CASE-Zweig für Programmnummer = 1
16	Der übergeordneten Steuerung wird der Empfang der Programmnummer 1 mitgeteilt.
17	Das benutzerdefinierte Programm EXAMPLE1 wird aufgerufen.
27	DEFAULT = Die Programmnummer ist ungültig.
28	Fehlerbehandlung bei ungültiger Programmnummer

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Navigator das Programm CELL.SRC öffnen. (Befindet sich im Ordner "R1".)
2. Im Abschnitt CASE 1 die Bezeichnung EXAMPLE1 durch den Namen des Programms ersetzen, das von der Programmnummer 1 aufgerufen werden soll. Das Semikolon vor dem Namen löschen.

```
 …
15      CASE 1
16        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17        MY_PROGRAM ( ) ; Call User-Program
…
```

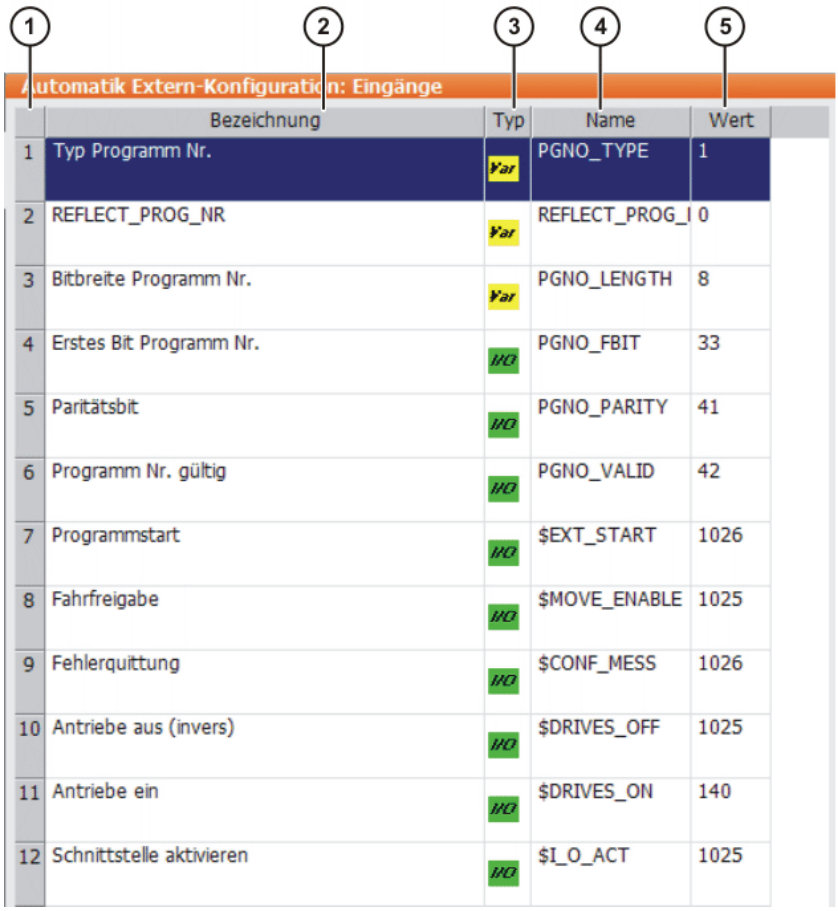
3. Für alle weiteren Programme analog zu Schritt 2. verfahren. Bei Bedarf weitere CASE-Zweige hinzufügen.
4. Das Programm CELL.SRC schließen. Die Sicherheitsabfrage, ob die Änderungen gespeichert werden sollen, mit **Ja** bestätigen.

6.17.2 Automatik Extern Ein-/Ausgänge konfigurieren

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Ein-/Ausgänge** > **Automatik Extern** wählen.
2. In der Spalte **Wert** die Zelle markieren, die bearbeitet werden soll, und auf **Bearb.** drücken.
3. Den gewünschten Wert eingeben und mit **OK** speichern.
4. Schritte 2. und 3. für alle zu bearbeitenden Werte wiederholen.
5. Das Fenster schließen. Die Änderungen werden übernommen.

Beschreibung



	Bezeichnung	Typ	Name	Wert
1	Typ Programm Nr.	Var	PGNO_TYPE	1
2	REFLECT_PROG_NR	Var	REFLECT_PROG_I	0
3	Bitbreite Programm Nr.	Var	PGNO_LENGTH	8
4	Erstes Bit Programm Nr.	I/O	PGNO_FBIT	33
5	Paritätsbit	I/O	PGNO_PARITY	41
6	Programm Nr. gültig	I/O	PGNO_VALID	42
7	Programmstart	I/O	$EXT_START	1026
8	Fahrfreigabe	I/O	$MOVE_ENABLE	1025
9	Fehlerquittung	I/O	$CONF_MESS	1026
10	Antriebe aus (invers)	I/O	$DRIVES_OFF	1025
11	Antriebe ein	I/O	$DRIVES_ON	140
12	Schnittstelle aktivieren	I/O	$I_O_ACT	1025

Abb. 6-21: Konfiguration Automatik Extern Eingänge

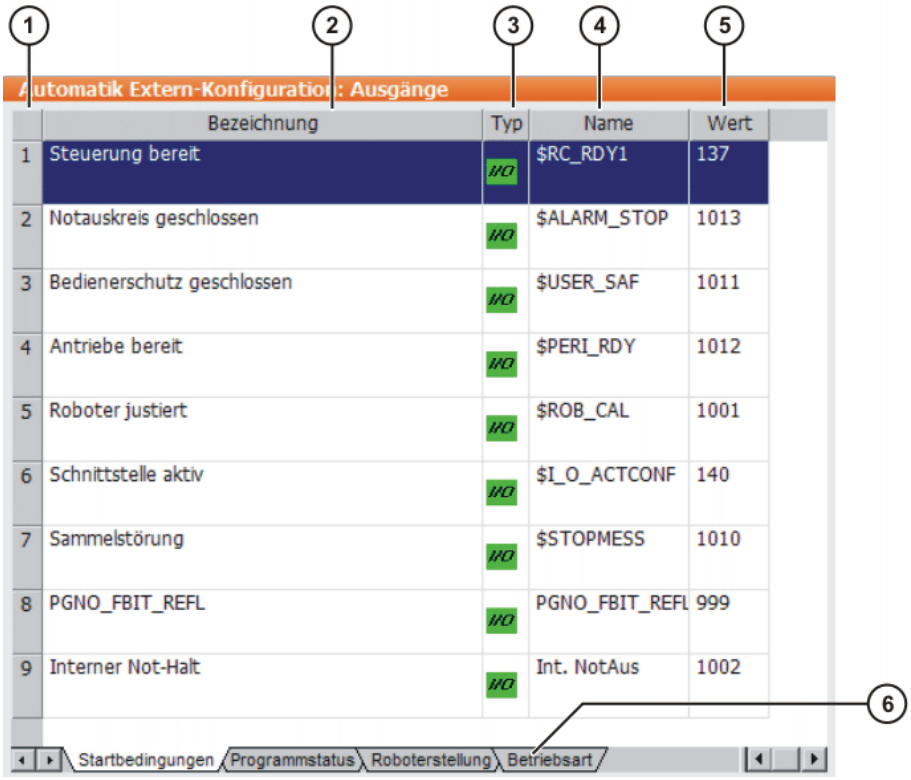


Abb. 6-22: Konfiguration Automatik Extern Ausgänge

Pos.	Beschreibung
1	Nummer
2	Langtext-Name des Ein-/Ausgangs
3	Typ ■ **Grün**: Ein-/Ausgang ■ **Gelb**: Variable oder Systemvariable ($...)
4	Name des Signals oder der Variable
5	Ein-/Ausgangsnummer oder Kanalnummer
6	Die Ausgänge sind thematisch in Registerkarten eingeteilt.

6.17.2.1 Automatik Extern Eingänge

PGNO_TYPE Typ: Variable

Diese Variable legt fest, in welchem Format die von der übergeordneten Steuerung übermittelte Programmnummer eingelesen wird.

Wert	Beschreibung	Beispiel
1	Einlesen als Binärzahl. Die Programmnummer wird von der übergeordneten Steuerung als binär codierter Integerwert übergeben.	0 0 1 0 0 1 1 1 => PGNO = 39

Wert	Beschreibung	Beispiel
2	Einlesen als BCD-Wert. Die Programmnummer wird von der übergeordneten Steuerung als binär codierter Dezimalwert übergeben.	0 0 1 0 0 1 1 1 => PGNO = 27
3	Einlesen als "1 aus N"*. Die Programmnummer wird von der übergeordneten Steuerung oder der Peripherie als "1 aus N" codierter Wert übergeben.	0 0 0 0 0 0 0 1 => PGNO = 1 0 0 0 0 1 0 0 0 => PGNO = 4

* Bei diesem Übergabeformat werden die Werte von PGNO_REQ, PGNO_PARITY sowie PGNO_VALID nicht ausgewertet und sind somit ohne Bedeutung.

REFLECT_PROG_NR

Typ: Variable

Diese Variable legt fest, ob die Programmnummer auf einen Bereich von Ausgängen gespiegelt wird. Die Ausgabe erfolgt ab dem Ausgang, der mit PGNO_FBIT_REFL festgelegt wird.

Wert	Beschreibung
0	Funktion ausgeschaltet
1	Funktion eingeschaltet

PGNO_LENGTH

Typ: Variable

Diese Variable legt die Bitbreite der von der übergeordneten Steuerung übermittelten Programmnummer fest. Wertebereich: 1 … 16.

Beispiel: PGNO_LENGTH = 4 => Die externe Programmnummer ist 4 Bit breit.

Wenn PGNO_TYPE den Wert 2 hat, sind nur die Bitbreiten 4, 8, 12 und 16 zugelassen.

PGNO_FBIT

Eingang, der das erste Bit der Programmnummer darstellt. Wertebereich: 1 … 8192.

Beispiel: PGNO_FBIT = 5 => Die externe Programmnummer beginnt mit dem Eingang $IN[5].

PGNO_PARITY

Eingang, auf den das Paritäts-Bit von der übergeordneten Steuerung übertragen wird.

Eingang	Funktion
Negativer Wert	Ungerade Parität
0	Keine Auswertung
Positiver Wert	Gerade Parität

(>>> 6.17.2.2 "Gerade / Ungerade Parität" Seite 202)

Wenn PGNO_TYPE den Wert 3 hat, wird PGNO_PARITY nicht ausgewertet.

PGNO_VALID

Eingang, auf den das Kommando zum Einlesen der Programmnummer von der übergeordneten Steuerung übertragen wird.

Eingang	Funktion
Negativer Wert	Nummer wird mit der abfallenden Flanke des Signals übernommen.

Eingang	Funktion
0	Nummer wird mit der ansteigenden Flanke des Signals an der Leitung EXT_START übernommen.
Positiver Wert	Nummer wird mit der ansteigenden Flanke des Signals übernommen.

Wenn PGNO_TYPE den Wert 3 hat, wird PGNO_VALID nicht ausgewertet.

$EXT_START Mit dem Setzen dieses Eingangs kann bei aktiver E/A-Schnittstelle ein Programm gestartet oder fortgesetzt werden.

Es wird nur die ansteigende Flanke des Signals ausgewertet.

HINWEIS Im Automatik Extern-Betrieb gibt es keine SAK-Fahrt. Dies bedeutet, dass der Roboter die erste programmierte Position nach dem Start mit programmierter (nicht reduzierter) Geschwindigkeit anfährt und dort nicht stoppt.

$MOVE_ENABLE Dieser Eingang wird zur Kontrolle der Roboterantriebe durch die übergeordnete Steuerung verwendet.

Signal	Funktion
TRUE	Handverfahren und Programmausführung möglich
FALSE	Stillsetzen aller Antriebe und Verriegelung aller aktiven Kommandos

Wenn die Antriebe von der übergeordneten Steuerung stillgesetzt worden sind, dann wird die Meldung "FAHRFREIGABE GESAMT" angezeigt. Das Bewegen des Roboters ist erst nach dem Löschen dieser Meldung und einem erneuten externen Startsignal wieder möglich.

Während der Inbetriebnahme wird die Variable $MOVE_ENABLE häufig auf den Wert $IN[1025] projektiert. Wenn danach vergessen wird, einen anderen Eingang zu projektieren, ist kein externer Start möglich.

$CHCK_MOVENA Typ: Variable

Wenn die Variable $CHCK_MOVENA den Wert FALSE besitzt, kann $MOVE_ENABLE umgangen werden. Der Wert der Variablen kann nur in der Datei C:\KRC\ROBOTER\KRC\STEU\Mada\$OPTION.DAT geändert werden.

Signal	Funktion
TRUE	Überwachung für MOVE_ENABLE ist wirksam.
FALSE	Überwachung für MOVE_ENABLE ist deaktiviert.

Um die Überwachung für MOVE_ENABLE verwenden zu können, muss $MOVE_ENABLE auf den Eingang $IN[1025] projektiert worden sein. Andernfalls hat $CHCK_MOVENA keine Wirkung.

$CONF_MESS Durch Setzen dieses Eingangs quittiert die übergeordnete Steuerung Fehlermeldungen selber, sobald die Störungsursache beseitigt wurde.

Es wird nur die ansteigende Flanke des Signals ausgewertet.

$DRIVES_OFF Wenn an diesem Eingang ein Low-Impuls von mindestens 20 ms Dauer anliegt, schaltet die übergeordnete Steuerung die Roboterantriebe ab.

$DRIVES_ON Wenn an diesem Eingang ein High-Impuls von mindestens 20 ms Dauer anliegt, schaltet die übergeordnete Steuerung die Roboterantriebe ein.

$I_O_ACT Wenn dieser Eingang TRUE ist, ist die Schnittstelle Automatik Extern aktiv. Defaulteinstellung: $IN[1025].

6.17.2.2 Gerade / Ungerade Parität

Beschreibung Ein Paritäts-Bit ist ein Bit, das zur Kontrolle an eine Bit-Folge angehängt wird. Es sagt aus, ob die Bit-Folge eine gerade oder ungerade Summe von Einsen enthält.

Welcher Bit-Wert was aussagt ("gerade" oder "ungerade"), hängt davon ab, welches Paritätsprotokoll gilt: "Gerade Parität" oder "Ungerade Parität".

Wenn der gesamte Datenblock, bestehend aus Bit-Folge und Paritäts-Bit, übertragen wird und danach das Paritäts-Bit nicht mehr zur Bit-Folge passt, bedeutet dies, dass bei der Übertragung ein Fehler passiert ist.

Gerade Parität **Gerade Parität / Even Parity**

Die Summe der Einsen im gesamten Datenblock (Bit-Folge + Paritäts-Bit) muss gerade sein.

Summe der Einsen in Bit-Folge	Paritäts-Bit
Gerade	0
Ungerade	1

Beispiel:

Bit-Folge	Paritäts-Bit
0011.1010	0
1010.0100	1

Ungerade Parität **Ungerade Parität / Odd Parity**

Die Summe der Einsen im gesamten Datenblock (Bit-Folge + Paritäts-Bit) muss ungerade sein.

Summe der Einsen in Bit-Folge	Paritäts-Bit
Gerade	1
Ungerade	0

Beispiel:

Bit-Folge	Paritäts-Bit
0011.1010	1
1010.0100	0

6.17.2.3 Automatik Extern Ausgänge

$RC_RDY1 Bereit für Programmstart.

$ALARM_STOP Dieser Ausgang wird bei folgenden NOT-HALT-Situationen zurückgesetzt:

- Die NOT-HALT-Einrichtung am smartPAD wird gedrückt.
- Externer NOT-HALT

Bei einem NOT-HALT ist an den Zuständen der Ausgänge **$ALARM_STOP** und **$ALARM_STOP_INTERN** erkennbar, um welche Art von NOT-HALT es sich handelt:

- Beide Ausgänge sind FALSE: Der NOT-HALT wurde am smartPAD ausgelöst
- **$ALARM_STOP** ist FALSE, **$ALARM_STOP_INTERN** ist TRUE: Externer NOT-HALT

$USER_SAF Dieser Ausgang wird beim Öffnen des Schutzgitter-Abfrageschalters (Betriebsart AUT) oder beim Loslassen eines Zustimmungsschalters (Betriebsart T1 oder T2) zurückgesetzt.

$PERI_RDY Mit Setzen dieses Ausgangs teilt die Robotersteuerung der übergeordneten Steuerung mit, dass die Roboterantriebe eingeschaltet sind.

$ROB_CAL Das Signal ist FALSE, sobald eine Achse des Roboters dejustiert ist.

$I_O_ACTCONF Dieser Ausgang ist TRUE, wenn die Betriebsart Automatik Extern ausgewählt ist und der Eingang $I_O_ACT TRUE ist.

$STOPMESS Dieser Ausgang wird von der Robotersteuerung gesetzt, um der übergeordneten Steuerung das Auftreten einer Meldung anzuzeigen, die das Anhalten des Roboters erforderlich machte. (Beispiele: NOT-HALT, Fahrfreigabe oder Bedienerschutz)

PGNO_FBIT_REFL Ausgang, der das erste Bit der Programmnummer darstellt. Voraussetzung: Der Eingang REFLECT_PROG_NR hat den Wert 1.

Die Größe des Ausgangsbereichs hängt von der Bitbreite der Programmnummer ab (PGNO_LENGTH).

Wenn ein Programm, welches von der SPS angewählt wurde, vom Bediener abgewählt wird, wird der Ausgabebereich beginnend mit PGNO_FBIT_REFL auf FALSE gesetzt. Auf diese Weise kann die SPS den manuellen Neustart eines Programmes unterbinden.

PGNO_FBIT_REFL wird ebenfalls auf FALSE gesetzt, wenn sich der Interpreter im CELL-Programm befindet.

$ALARM_STOP_INTERN Früherer Name: **Int. NotAus**

Dieser Ausgang wird auf FALSE gesetzt, wenn die NOT-HALT-Einrichtung am smartPAD gedrückt wird.

Bei einem NOT-HALT ist an den Zuständen der Ausgänge **$ALARM_STOP** und **$ALARM_STOP_INTERN** erkennbar, um welche Art von NOT-HALT es sich handelt:

- Beide Ausgänge sind FALSE: Der NOT-HALT wurde am smartPAD ausgelöst
- **$ALARM_STOP** ist FALSE, **$ALARM_STOP_INTERN** ist TRUE: Externer NOT-HALT

$PRO_ACT Dieser Ausgang ist immer dann gesetzt, wenn ein Prozess auf Roboterebene aktiv ist. Der Prozess ist aktiv, solange ein Programm oder ein Interrupt bearbeitet wird. Die Programmbearbeitung am Ende des Programms wird erst dann inaktiv, wenn alle Impulsausgänge und Trigger abgearbeitet sind.

Im Fall eines Fehlerstopps sind folgende Möglichkeiten zu unterscheiden:

- Wenn Interrupts aktiviert wurden, aber zum Zeitpunkt des Fehlerstopps nicht bearbeitet wurden, gilt der Prozess als inaktiv ($PRO_ACT=FALSE).

- Wenn Interrupts aktiviert wurden und zum Zeitpunkt des Fehlerstopps bearbeitet wurden, gilt der Prozess so lange als aktiv ($PRO_ACT=TRUE), bis das Interruptprogramm abgearbeitet ist oder auf einen HALT läuft ($PRO_ACT=FALSE).
- Wenn Interrupts aktiviert wurden und das Programm läuft auf einen HALT, gilt der Prozess als inaktiv ($PRO_ACT=FALSE). Wenn nach diesem Zeitpunkt eine Interruptbedingung erfüllt ist, gilt der Prozess so lange als aktiv ($PRO_ACT=TRUE), bis das Interruptprogramm abgearbeitet ist oder auf einen HALT läuft ($PRO_ACT=FALSE).

PGNO_REQ Mit einem Signalwechsel an diesem Ausgang wird die übergeordnete Steuerung aufgefordert, eine Programmnummer zu übermitteln.

Wenn PGNO_TYPE den Wert 3 hat, wird PGNO_REQ nicht ausgewertet.

APPL_RUN Mit dem Setzen dieses Ausgangs teilt die Robotersteuerung der übergeordneten Steuerung mit, dass gerade ein Programm abgearbeitet wird.

$PRO_MOVE Bedeutet, dass sich eine synchrone Achse bewegt, auch bei Handverfahren. Das Signal ist die Invertierung von $ROB_STOPPED.

$IN_HOME Dieser Ausgang teilt der übergeordneten Steuerung mit, ob sich der Roboter in seiner HOME-Position befindet.

$ON_PATH Dieser Ausgang ist gesetzt, solange sich der Roboter auf seiner programmierten Bahn befindet. Nach der SAK-Fahrt wird der Ausgang ON_PATH gesetzt. Dieser Ausgang bleibt solange gesetzt, bis der Roboter die Bahn verlässt, das Programm zurückgesetzt wird oder eine Satzanwahl durchgeführt wird. Das Signal ON_PATH hat aber kein Toleranzfenster; sobald der Roboter die Bahn verlässt, wird dieses Signal zurückgesetzt.

$NEAR_POSRET Über dieses Signal kann die übergeordnete Steuerung feststellen, ob der Roboter innerhalb einer Kugel um die in $POS_RET gespeicherte Position steht. Mit dieser Information kann die übergeordnete Steuerung entscheiden, ob das Programm wieder gestartet werden darf oder nicht.

Der Radius der Kugel kann vom Benutzer in der Datei $CUSTOM.DAT über die Systemvariable $NEARPATHTOL definiert werden.

$ROB_STOPPED Das Signal wird gesetzt, wenn der Roboter steht. Auch bei einer WAIT-Anweisung wird dieser Ausgang während des Wartens gesetzt.

Das Signal ist die Invertierung von $PRO_MOVE.

$T1, $T2, $AUT, $EXT Diese Ausgänge werden gesetzt, wenn die entsprechende Betriebsart angewählt ist.

6.17.3 Fehlernummern an übergeordnete Steuerung übertragen

Fehlernummern der Robotersteuerung im Bereich 1 … 255 können an die übergeordnete Steuerung übertragen werden. Um die Fehlernummern zu übertragen, muss die Datei P00.DAT im Verzeichnis C:\KRC\ROBOTER\KRC\R1\TP folgendermaßen konfiguriert werden:

```
1  DEFDAT  P00
2
3  BOOL PLC_ENABLE=TRUE ; Enable error-code transmission to plc
4  INT I
5  INT F_NO=1
6  INT MAXERR_C=1 ; maximum messages for $STOPMESS
7  INT MAXERR_A=1 ; maximum messages for APPLICATION
8  DECL STOPMESS MLD
9  SIGNAL ERR $OUT[25]  TO $OUT[32]
```

```
10  BOOL FOUND
11
12  STRUC PRESET INT OUT,CHAR PKG[3],INT ERR
13  DECL PRESET P[255]
    …
26  P[1]={OUT 2,PKG[] "P00",ERR 10}
    …
30  P[128]={OUT 128,PKG[] "CTL",ERR 1}
    …
35  STRUC ERR_MESS CHAR P[3],INT E
36  DECL ERR_MESS ERR_FILE[64]
37  ERR_FILE[1]={P[] "XXX",E 0}
    …
96  ERR_FILE[64]={P[] "XXX",E 0}
97  ENDDAT
```

Zeile	Beschreibung
3	PLC_ENABLE muss TRUE sein.
6	Anzahl der Steuerungsfehler eintragen, für deren Übertragung Parameter festgelegt werden.
7	Anzahl der Applikationsfehler eintragen, für deren Übertragung Parameter festgelegt werden.
9	Festlegen, über welche Ausgänge der Robotersteuerung die übergeordnete Steuerung die Fehlernummer auslesen soll. Es müssen 8 Ausgänge sein.
13	Im folgenden Bereich die Parameter der Fehler eintragen. P[1] … P[127]: Bereich für Applikationsfehler P[128] … P[255]: Bereich für Steuerungsfehler
26	Beispiel Parameter für Applikationsfehler: ■ OUT 2 = Fehlernummer 2 ■ PKG[] "P00" = Technologiepaket ■ ERR 10 = Fehlernummer im ausgewählten Technologiepaket
30	Beispiel Parameter für Steuerungsfehler: ■ OUT 128 = Fehlernummer 128 ■ PKG[] "CTL" = Technologiepaket ■ ERR 1 = Fehlernummer im ausgewählten Technologiepaket
37 … 96	Im Speicher ERR_FILE werden die letzten 64 aufgetretenen Fehler aufbewahrt.

6.17.4 Signaldiagramme

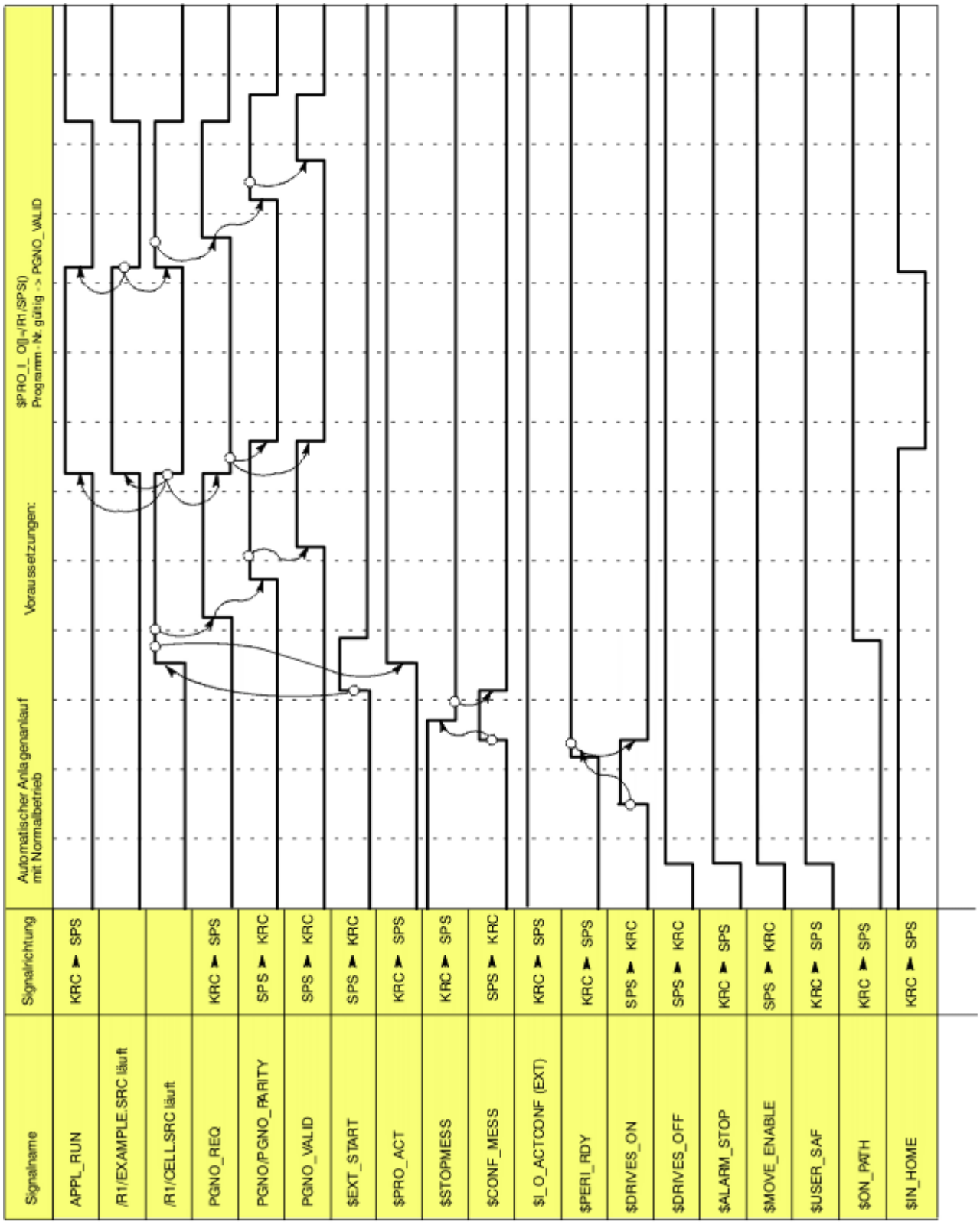


Abb. 6-23: Automatischer Anlagenanlauf und Normalbetrieb mit Quittierung der Programmnummer durch PGNO_VALID

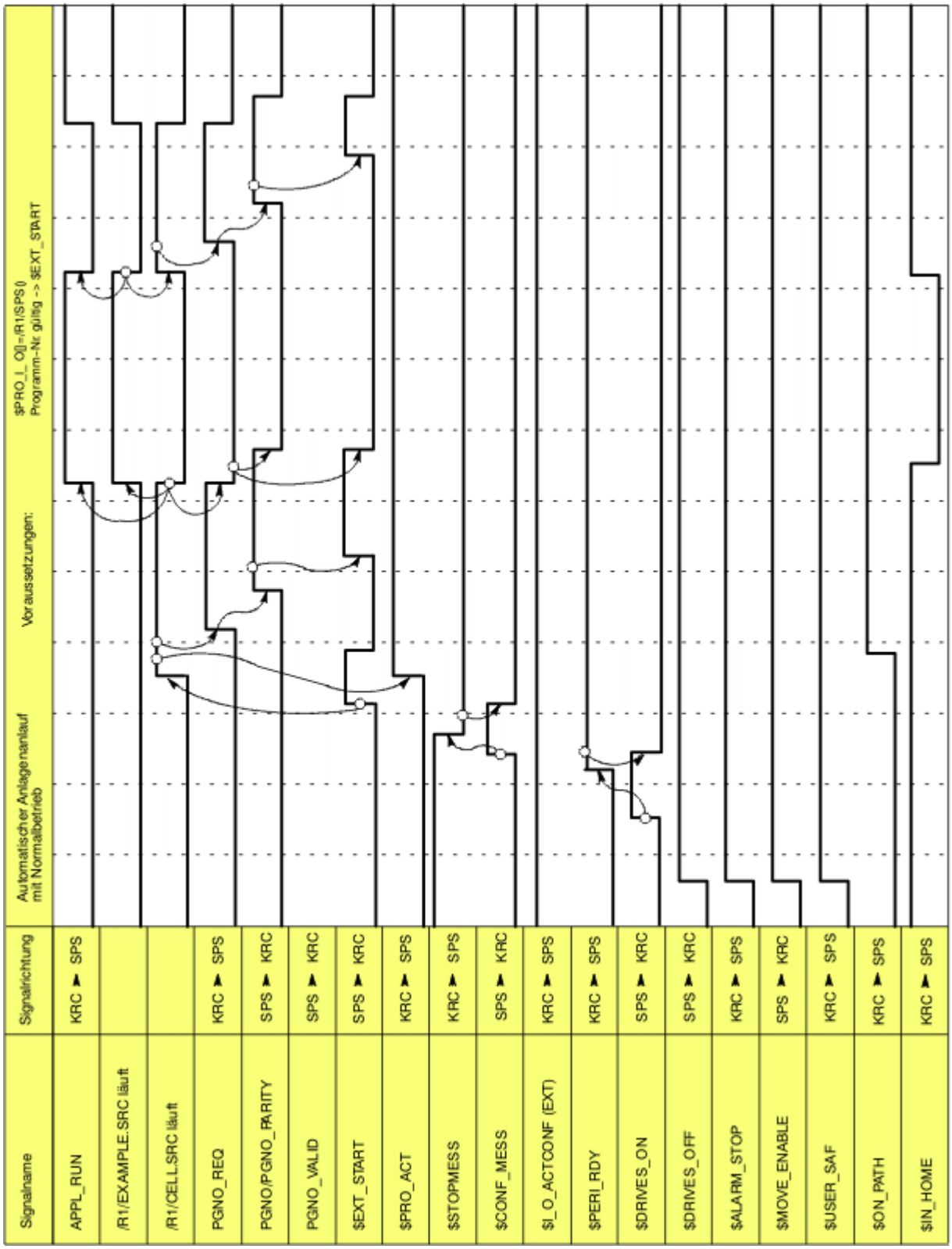


Abb. 6-24: Automatischer Anlagenanlauf und Normalbetrieb mit Quittierung der Programmnummer durch $EXT_START

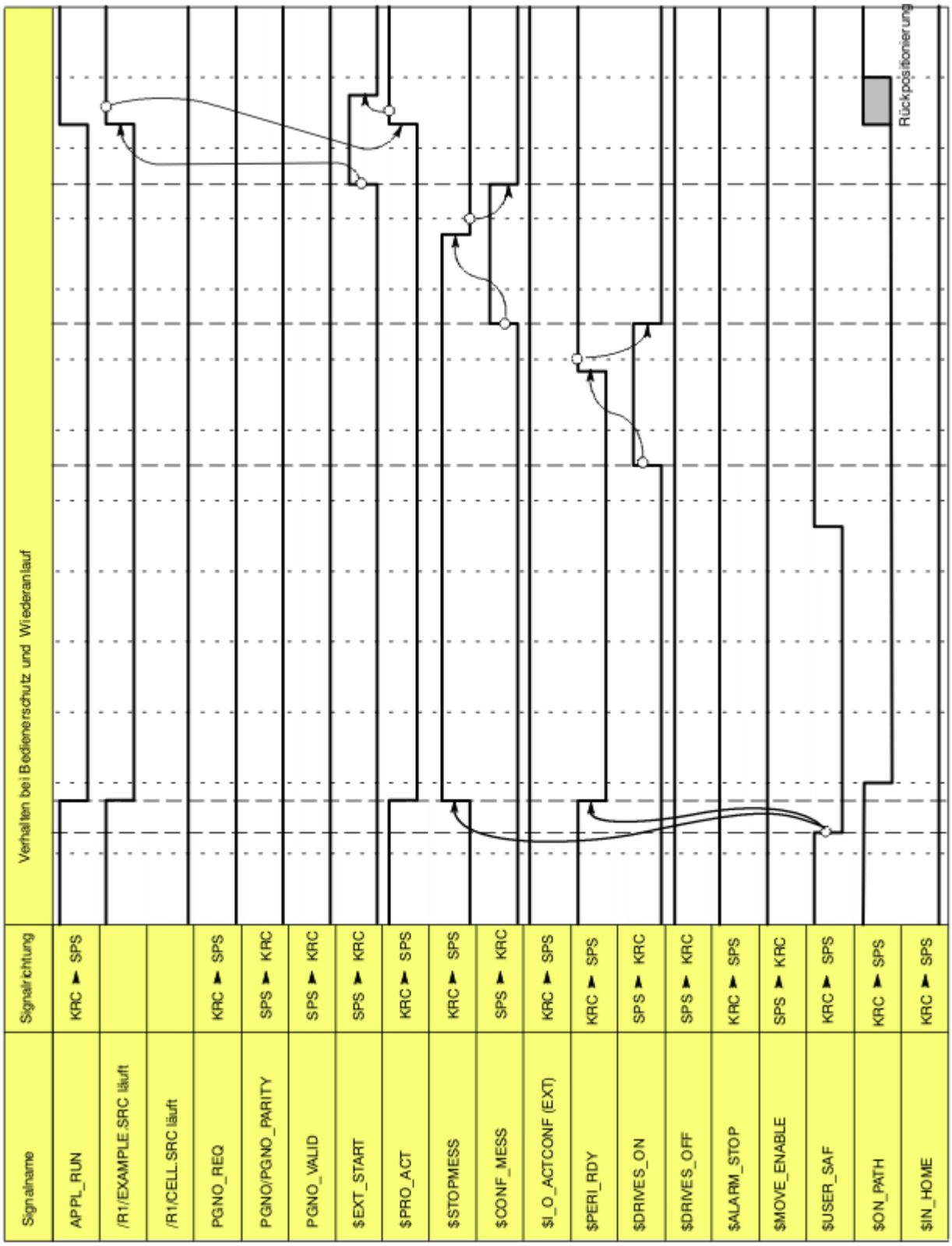


Abb. 6-25: Wiederanlauf nach generatorischem Stopp (Bedienerschutz und Wiederanlauf)

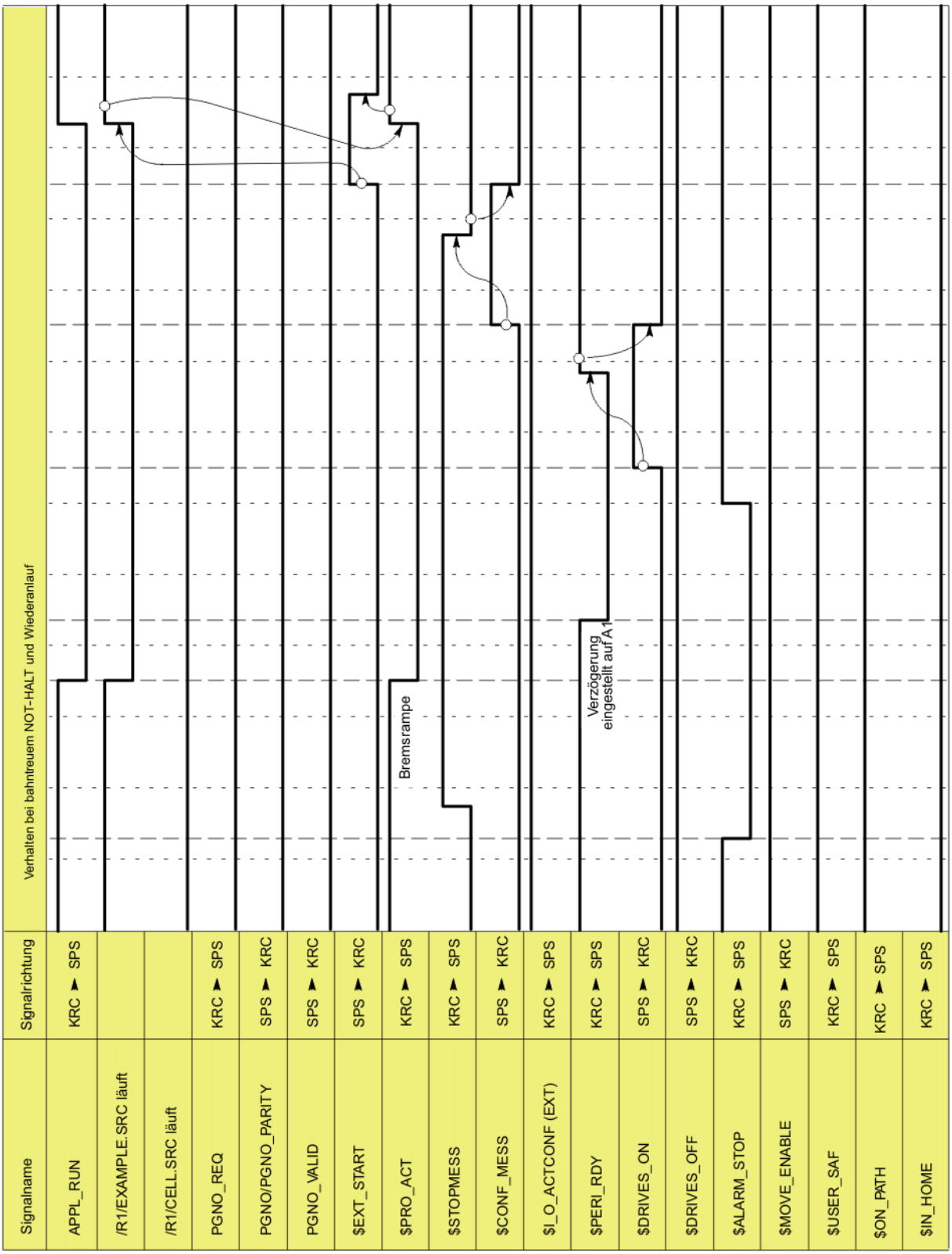


Abb. 6-26: Wiederanlauf nach bahntreuem NOT-HALT

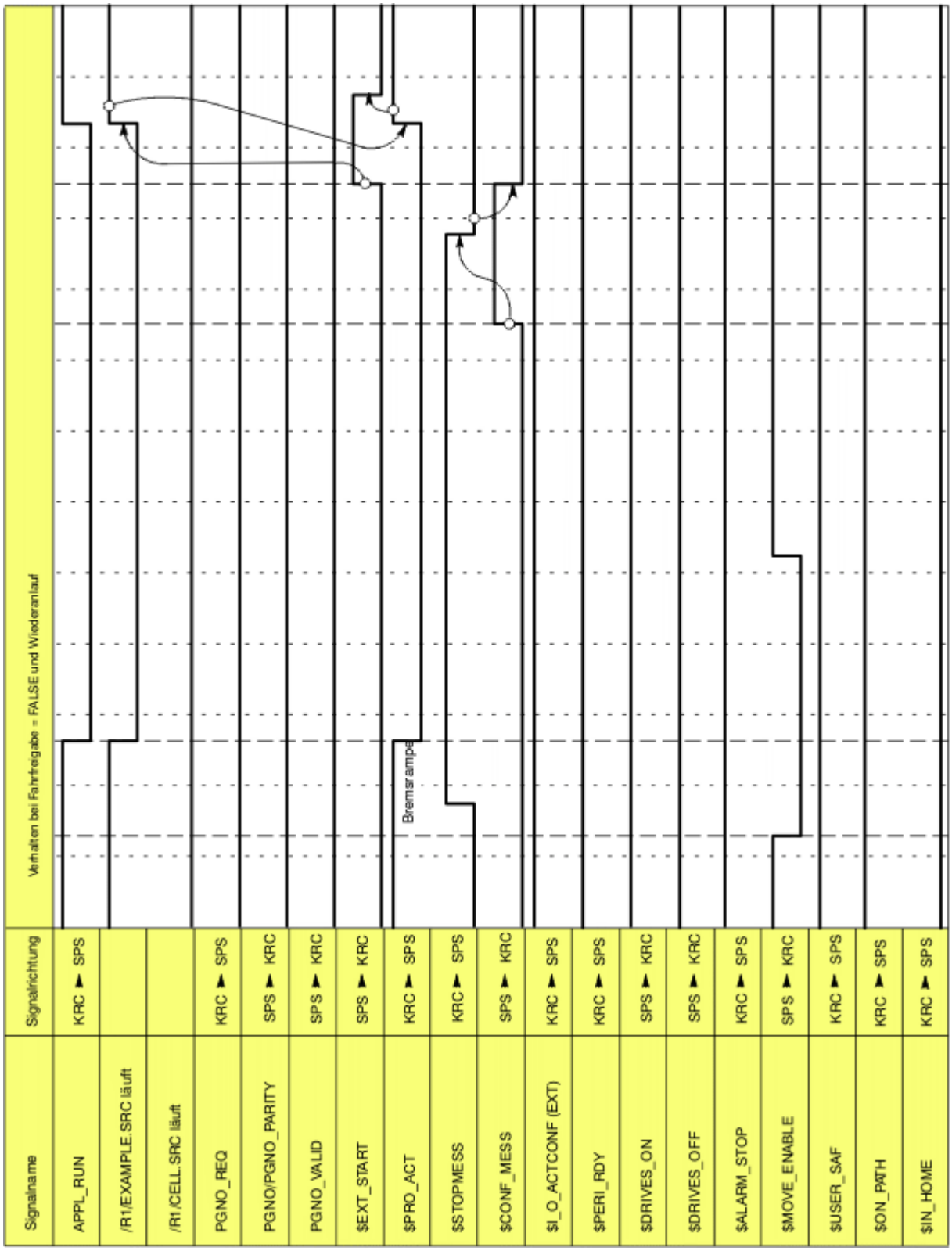


Abb. 6-27: Wiederanlauf nach Fahrfreigabe

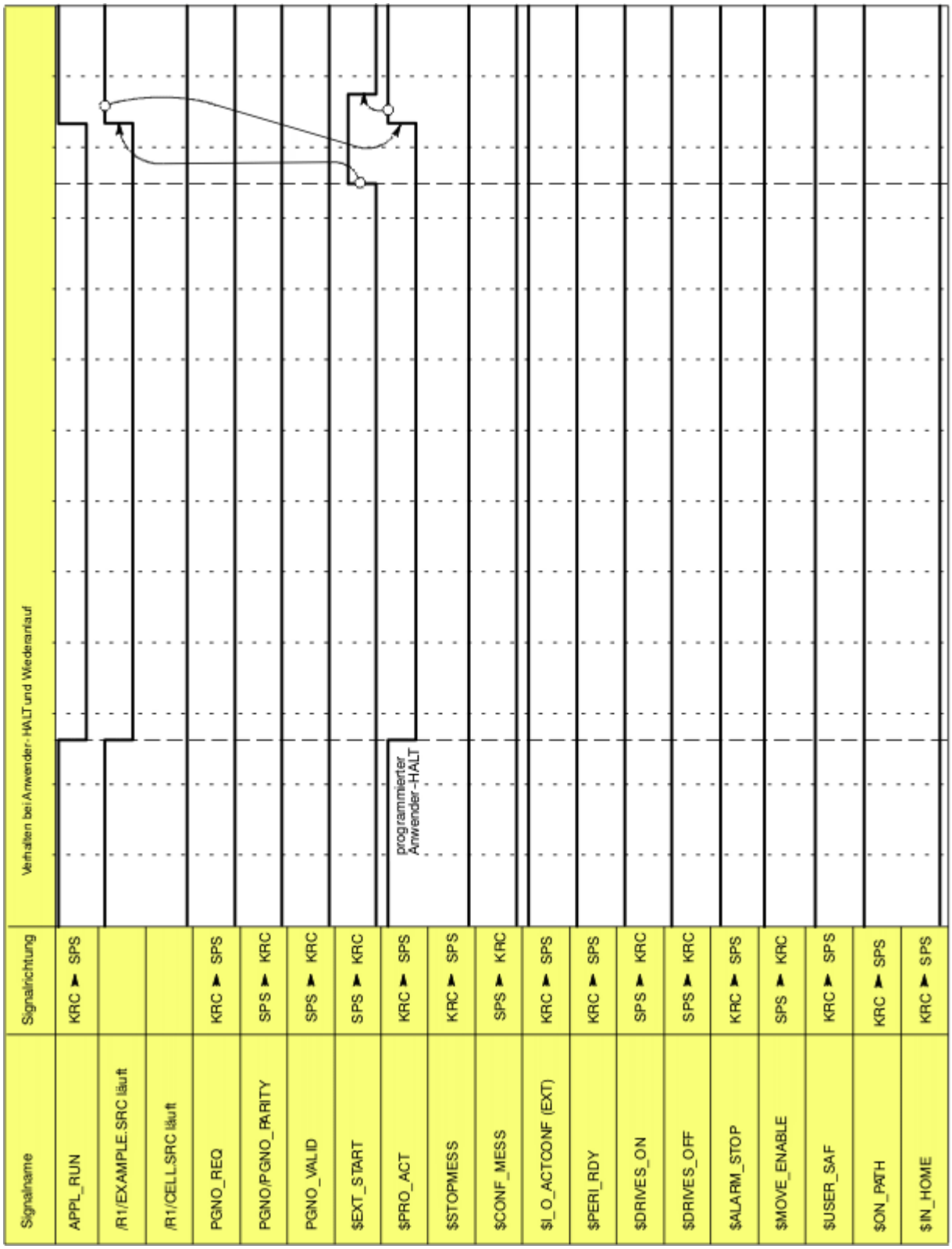


Abb. 6-28: Wiederanlauf nach Anwender-HALT

6.18 Momentenbetrieb

6.18.1 Übersicht: Momentenbetrieb

Beschreibung Die Funktionalität "Momentenbetrieb" umfasst die Teilfunktionalitäten "Momentenbegrenzung" und "Abschalten von Überwachungen".

Momentenbegrenzung:

Für einzelne oder mehrere Achsen können die Momente, d. h. der Motorstrom, begrenzt werden. Die Momentenbegrenzung ermöglicht folgende Anwendungsfälle:

- Die Achse kann mit einem definierten Moment gegen einen Widerstand drücken oder ziehen.

  Beispiel:

  Mit einer elektromotorischen Punktschweißzange einen definierten Druck auf das Werkstück aufbauen.
- Die Achse kann "weichgeschaltet" werden. Sie kann dann durch Krafteinwirkung von außen bewegt werden. Man kann sie z. B. wegschieben.

  Beispiele:

  Der Roboter muss ein Werkstück in einer Presse greifen, das dann von der Presse ausgestoßen wird. Damit der Roboter nachgeben und den Ausstoß abfangen kann, wird die betroffene Achse weichgeschaltet.

  Der Roboter muss ein Werkstück an eine Stelle legen, an der es von Spannern in die exakt richtige Position gezogen wird. Hierfür muss der Roboter nachgiebig sein.

Abschaltung von Überwachungen:

Durch die Momentenbegrenzung entsteht in der Regel eine relativ große Abweichung zwischen Soll- und Istposition. Bestimmte Überwachungen sprechen auf diese Abweichung an, was jedoch bei der Momentenbegrenzung nicht erwünscht ist. Deshalb können diese regulären Überwachungen abgeschaltet werden.

Einschränkung

Folgende Einschränkung ist zu beachten, wenn Achsen Austoßbewegungen abfangen sollen:

Eine schräg in den Raum wirkende Ausstoßbewegung kann durch das Weichschalten einer einzelnen Achse in der Regel nicht abgefangen werden. Abhilfe:

- Bei leicht schrägen Ausstoßbewegungen kann man dem dadurch abgeholfen werden, dass man den Roboter leicht schräg montiert.
- Oder die KUKA Roboter GmbH kontaktieren.

Die schräge Montage des Roboters ist nur bis zu einem bestimmten Neigungswinkel zulässig. Weitere Informationen sind in der Betriebsanleitung oder der Montageanleitung für den Roboter zu finden.

6.18.1.1 Momentenbetrieb verwenden

Momentenbetrieb ist nur im Programmbetrieb möglich, nicht im Handbetrieb.

⚠ WARNUNG Die Robotersteuerung ist defaultmäßig so konfiguriert, dass nur Grenzen gesetzt werden können, die das Haltemoment der Achse ($HOLDING_TORQUE) noch beinhalten. Dennoch kann es sein, dass die Achse bei begrenzten Momenten nicht mehr das erforderliche Moment zum Bremsen, Halten oder Fahren aufbringen kann. Dies kann z. B. der Fall sein, wenn die Default-Konfiguration der Robotersteuerung verändert wurde oder bei inkorrekten Lastdaten.
Falsch gesetzte Werte können zu unerwartetem Verhalten der Robotersteuerung führen, z. B. zu einer anderen Richtung der Bewegung oder veränderten Beschleunigungen.
Deshalb:

- Die Momente immer nur in kleinen Schritten begrenzen und sich so an die erforderliche Grenze herantasten.
- Die Momente nicht weiter begrenzen als erforderlich.

Wenn dies nicht berücksichtigt wird, können Tod, Verletzungen oder Sachschaden die Folge sein.

Wenn eine Applikation Momentengrenzen erfordert, die das Haltemoment der Achse nicht mehr beinhalten, muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

Vorgehensweise

1. Für die gewünschte Achse die Momentengrenzen setzen und/oder die regulären Überwachungen ausschalten.
   (>>> 6.18.2 "Momentenbetrieb aktivieren: SET_TORQUE_LIMITS()" Seite 215)
   Wenn die regulären Überwachungen ausgeschaltet werden, sind automatisch Überwachungen aktiv, die speziell auf den Momentenbetrieb abgestimmt sind.
   (>>> "Überwachungen" Seite 214)
2. Wenn die Achse weichgeschaltet werden soll: Die Achse bewegen, damit die Momentenbegrenzung aktiv wird. Nach dem Ende der Bewegung bleiben die Bremsen dieser Achse offen.
   Alternativ kann eine "Bewegung" auf die aktuelle Position durchgeführt werden. Dabei bewegt sich der Roboter nicht, aber die Bremsen öffnen sich.
3. Optional: Signal ausgeben, dass die Achse steht (z. Bsp. Signal an eine Spritzgiessmaschine).
4. Gewünschte Aktion durchführen, z. B. Werkstück anfahren und Druck aufbauen oder die Achse wegschieben.
5. Optional: Signal abwarten, um den Momentenbetrieb wieder zu beenden.
6. Den Momentenbetrieb wieder deaktivieren.
   (>>> 6.18.3 "Momentenbetrieb deaktivieren: RESET_TORQUE_LIMITS()" Seite 218)
   Die Momentengrenzen werden hierbei aufgehoben und die regulären Überwachungen werden wieder eingeschaltet. Außerdem wird die Sollposition an die Istposition angeglichen.

Aktiv/inaktiv

Der Momentenbetrieb gilt in folgendem Fall als aktiv:

- Wenn die obere Momentengrenze kleiner oder gleich dem oberen Wert des Intervalls von $TORQUE_AXIS_MAX ist.
  (>>> 6.18.5.3 "$TORQUE_AXIS_MAX" Seite 220)
- Und/oder: Wenn die untere Momentengrenze größer oder gleich dem unteren Wert des Intervalls von $TORQUE_AXIS_MAX ist.
- Und/oder: Wenn die regulären Überwachungen abgeschaltet sind.

Der Momentenbetrieb gilt in folgendem Fall als inaktiv:

- Wenn keine Grenzen gesetzt sind oder die Grenzen unwirksam sind. Eine Grenze ist unwirksam, wenn sie außerhalb des Intervalls von $TORQUE_AXIS_MAX liegt.
- Und: Wenn die regulären Überwachungen abgeschaltet sind.

Automatische Deaktivierung

In folgenden Fällen wird der Momentenbetrieb automatisch deaktiviert:

- Programmende
- Programm-Reset
- Programmabwahl
- Satzanwahl (jedoch keine Deaktivierung, wenn sich das Ziel der Satzanwahl innerhalb eines Interrupt-Programms befindet)
- RESUME (jedoch keine Deaktivierung, wenn man mit RESUME in ein Interrupt-Programm zurückkehrt)
- Der Handbetrieb wird aufgenommen. Es wird von der aktuellen Istposition aus geplant und mit vollem Moment weitergefahren.

Überwachungen

Im Momentenbetrieb entsteht in der Regel eine relativ große Abweichung zwischen Soll- und Istposition. Bestimmte Überwachungen sprechen auf diese Abweichung an, was jedoch im Momentenbetrieb nicht erwünscht ist. Deshalb können diese regulären Überwachungen mit SET_TORQUE_LIMITS() abgeschaltet werden.

Wenn die regulären Überwachungen abgeschaltet sind, sind für Istgeschwindigkeit und Schleppfehler automatisch andere Überwachungen aktiv, die speziell auf den Momentenbetrieb abgestimmt sind. Bei Bedarf können für diese Spezialüberwachungen mit SET_TORQUE_LIMITS() benutzerspezifische Werte gesetzt werden.

Die folgenden Meldungen gehören zu den regulären Überwachungen. Sie werden nicht mehr angezeigt, wenn die regulären Überwachungen abgeschaltet sind:

Überwachung	Meldungsnr. / Meldung
Schleppfehler-Überwachung	26024: *Quitt. max. Schleppfehler überschritten ({Antrieb}).*
Stillstands-Überwachung	1100: *Stillstand {(Achsnummer)}*
Positionier-Überwachung	1105: *Positionierüberwachung {(Achsnummer)}*
Überwachung, ob Motor blockiert	26009: *Motor blockiert ({Antrieb}).*

Es ist jedoch auch möglich, die regulären Überwachungen im Momentenbetrieb beizubehalten. Dies kann z. B. sinnvoll sein, wenn der Momentenbetrieb verwendet wird, um bei Kollisionen Schäden zu vermeiden.

(>>> 6.18.6.2 "Roboterprogramm: Bei Kollisionen Schäden vermeiden" Seite 223)

6.18.1.2 Beispiel Roboterprogramm: A1 in beide Richtungen weichschalten

Beschreibung

Dieses einfache Beispiel verdeutlicht das Grundprinzip des Momentenbetriebs.

In dem Beispiel soll A1 in beide Richtungen weichgeschaltet werden. Hierfür werden sowohl die positive als auch die negative Stromgrenze auf 0 Nm gesetzt. Dadurch kann die A1 durch Krafteinwirkung von außen bewegt werden.

Programm

```
   ...
1  PTP {A1 10}
2  SET_TORQUE_LIMITS(1, {lower 0, upper 0, monitor #off})
3  PTP {A1 11}
   ...
4  RESET_TORQUE_LIMITS(1)
5  PTP {A1 -20}
   ...
```

Zeile	Beschreibung
2	Negative und positive Stromgrenze der A1 werden auf 0 gesetzt; die regulären Überwachungen werden ausgeschaltet. (Die Istgeschwindigkeit und der Schleppfehler werden jetzt mit Spezialüberwachungen überwacht.)
3	Eine "Bewegung" wird durchgeführt, damit die Momentenbegrenzung aktiv wird. (Weil beide Stromgrenzen auf 0 sind, bewegt sich der Roboter in Wirklichkeit nicht.) A1 kann nun durch Krafteinwirkung von außen bewegt werden.
4	Den Momentenbetrieb für A1 wieder deaktivieren. Die Momentenbegrenzung wird hierbei aufgehoben und die regulären Überwachungen werden wieder eingeschaltet. Außerdem wird automatisch die Sollposition an die Istposition angeglichen.
5	Der Roboter fährt zur nächsten Position. (Die Bewegung aus Zeile 4 wird nicht nachgeholt, weil in Zeile 5 der Soll-Ist-Abgleich stattgefunden hat.)

Das Haltemoment ist nur bei A1 = 0 Nm. Zum Weichschalten können also die Grenzen nicht generell auf 0 gesetzt werden.

Diese Dokumentation enthält ein detaillierteres Beispiel zum Weichschalten, das auch auf andere Achsen übertragbar ist.

(>>> 6.18.6.1 "Roboterprogramm: Achse in beide Richtungen weichschalten" Seite 222)

6.18.2 Momentenbetrieb aktivieren: SET_TORQUE_LIMITS()

Beschreibung

Mit dieser Funktion kann man für eine bestimmte Achse folgende Aktionen durchführen:

- Die Momente in positive und/oder negative Richtung begrenzen.
- Die regulären Überwachungen, die bei erhöhtem Schleppfehler ansprechen würden, ausschalten.
- Wenn die regulären Überwachungen ausgeschaltet sind: Die Werte für die Spezialüberwachungen ändern.

Funktion

`SET_TORQUE_LIMITS (`*axis*`: in, `*values*`: in)`

Element	Beschreibung
axis	Typ: INT Achse, für die die Anweisung gilt
values	Typ: TorqLimitParam Werte, die für die Achse gesetzt werden

TorqLimitParam

```
STRUC TorqLimitParam REAL lower, upper, SW_ONOFF monitor,
REAL max_vel, max_lag
```

Element	Beschreibung
lower	Untere Momentengrenze Einheit: Nm (für Linearachsen: N) Default-Wert: -1E10 (d. h. unbegrenzt)
upper	Obere Momentengrenze Einheit: Nm (für Linearachsen: N) Default-Wert: 1E10 (d. h. unbegrenzt)
monitor	■ **#ON** (Default): Schaltet die regulären Überwachungen ein. ■ **#OFF**: Schaltet die regulären Überwachungen aus. Stattdessen sind die Überwachungen max_vel und max_lag aktiv.
max_vel	Maximale im Momentenbetrieb zulässige Istgeschwindigkeit (nur relevant, wenn die regulären Überwachungen abgeschaltet sind) Es darf nur ein positiver Wert programmiert werden. Einheit: Grad (für Linearachsen: mm) Default-Wert (gilt in allen Betriebsarten): T1-Handverfahr-Geschwindigkeit * interner Sicherheitsfaktor In T1 kann maximal mit dem Default-Wert verfahren werden, auch wenn ein höherer Wert programmiert ist. **Hinweis:** Einen höheren Wert als den Default-Wert nur setzen, wenn unbedingt notwendig.
max_lag	Maximaler im Momentenbetrieb zulässiger Schleppfehler (nur relevant, wenn die regulären Überwachungen abgeschaltet sind) Es darf nur ein positiver Wert programmiert werden. Einheit: Grad (für Linearachsen: mm) Default-Wert: 5 Grad (für Linearachsen: 100 mm) **Hinweis:** Einen höheren Wert als den Default-Wert nur setzen, wenn unbedingt notwendig.

lower/upper

Wann muss das obere, wann das untere Moment begrenzt werden?

Allgemein: Es muss immer die Richtung begrenzt werden, in die sich der Schleppfehler aufbaut.

Beispiel: Man will gegen ein Hindernis fahren und dort stehen bleiben. Hierbei soll das Moment, dass sich aufbaut, begrenzt werden.

- Wenn das Hindernis in positiver Bewegungsrichtung auftaucht, muss `upper` gesetzt werden.
- Wenn das Hindernis in negativer Bewegungsrichtung auftaucht, muss `lower` gesetzt werden.

Eigenschaften

- SET_TORQUE_LIMITS() kann in Roboterprogrammen und in Submit-Programmen verwendet werden.
- **Vorlaufstopp:** Im Roboterprogramm löst die Anweisung einen Vorlaufstopp aus.
- *Values* darf teilweise uninitialisiert bleiben. Die uninitialisierten Komponenten bedeuten, dass die bestehenden Werte unverändert bleiben sollen.

- Wenn beide Grenzen gesetzt werden, muss `upper` >= `lower` sein
- Wenn eine Grenze bereits gesetzt ist (oder beide) und die andere Grenze danach gesetzt wird, und wenn sich nun durch die neue Grenze ein leeres Intervall ergeben würde, dann wird der neue Grenzwert der Wert für beide Grenzen. Beispiel:
  - Bereits gesetzt: `{lower 1, upper 2}`
  - Neu gesetzt: `{lower 3}`
  - Dadurch gilt: `{lower 3, upper 3}`
- Es ist zulässig, eine positives `lower` oder ein negatives `upper` zu setzen.
- Die Grenzen müssen so gesetzt werden, dass sie das aktuelle Haltemoment $HOLDING_TORQUE beinhalten. Wenn sie anders gesetzt werden, gibt die Robotersteuerung eine Fehlermeldung aus, die der Benutzer quittieren muss.

Wenn eine Applikation Momentengrenzen erfordert, die das Haltemoment der Achse nicht mehr beinhalten, muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

- `lower` muss kleiner oder gleich dem oberen Wert des Intervalls von $TORQUE_AXIS_MAX_0 sein.

  `upper` muss größer oder gleich dem unteren Wert des Intervalls von $TORQUE_AXIS_MAX_0 sein.

  Wenn die Grenzen anders gesetzt werden, gibt die Robotersteuerung eine Fehlermeldung aus, die der Benutzer quittieren muss.

Beispiele

Beispiel 1:

Bei A1 wird der zulässige Momentenbereich auf das Intervall 800 … 1 400 Nm begrenzt.

```
SET_TORQUE_LIMITS(1, {lower 800, upper 1400} )
```

Beispiel 2:

Bei A3 wird die obere Momentengrenze auf 1 200 Nm gesetzt.

```
SET_TORQUE_LIMITS(3, {upper 1200} )
```

Beispiel 3:

Für A1 werden die regulären Überwachungen (wieder) eingeschaltet.

```
SET_TORQUE_LIMITS(1, {monitor #on} )
```

Beispiel 4:

Für A1 wird der zulässige Momentenbereich auf das Intervall -1 000 … 1 000 Nm begrenzt. Außerdem werden die regulären Überwachungen abgeschaltet und die Spezialüberwachungen werden auf benutzerdefinierte Werte gesetzt.

```
SET_TORQUE_LIMITS(1, {lower -1000, upper 1000, monitor #off, max_vel
10, max_lag 20} )
```

Beispiel 5:

Für A1 wird der zulässige Momentenbereich auf -1E10 … 1E10 gesetzt, d. h., der Bereich ist unbeschränkt. Die regulären Überwachungen werden (wieder) eingeschaltet.

```
SET_TORQUE_LIMITS(1, {lower -1E10, upper 1E10, monitor #on} )
```

Das Ganze entspricht `RESET_TORQUE_LIMITS(1)`, mit dem Unterschied, dass hier bei Beispiel 5 die Sollposition nicht an die Istpositon angeglichen wird.

Beispiel 6:

Bei A1 wird die untere Momentengrenze auf einen berechneten Wert gesetzt.

Der Wert wurde mit der Funktion myCalc() berechnet und mit der Variablen myLimits übergeben. (Im konkreten Anwendungsfall muss der Benutzer hierfür selber eine Funktion schreiben.)

Damit die anderen Komponenten uninitialisiert sind, wird der Wert mit einem teilinitialisierten Aggregat vorinitialisiert.

```
DECL TorqLimitParam myParams
...
myParams = {lower 0}
myParams.lower = myCalc()
SET_TORQUE_LIMITS(1, myParams)
```

Beispiel 7:

Hier werden die Grenzen ebenfalls auf einen Wert gesetzt, der mit einer Funktion berechnet wurde. (Im konkreten Anwendungsfall muss der Benutzer hierfür selber eine Funktion schreiben.)

Allerdings wird hier der Rückgabewert der Funktion direkt übergeben.

```
DEFFCT TorqLimitParam myCalcLimits()
DECL TorqLimitParam myLimits
...
RETURN myLimits
ENDFCT
...
SET_TORQUE_LIMITS(1, myCalcLimits())
```

6.18.3 Momentenbetrieb deaktivieren: RESET_TORQUE_LIMITS()

Beschreibung

Diese Funktion bewirkt Folgendes für die gewählte Achse:

- Sie hebt die Begrenzung der Momente auf, falls diese begrenzt waren.
- Sie schaltet die regulären Überwachungen wieder ein, falls diese ausgeschaltet waren.
- Sie gleicht die Sollposition an die Istpositon an.

Funktion

`RESET_TORQUE_LIMITS (`*axis*`: in)`

Element	Beschreibung
axis	Typ: INT Achse, für die die Anweisung gilt

Eigenschaften

- Die Anweisung kann in Roboterprogrammen und in Submit-Programmen verwendet werden.
- **Vorlaufstopp:** Im Roboterprogramm löst die Anweisung einen Vorlaufstopp aus. Dieser ist nicht über CONTINUE maskierbar!

Alternative

Wenn kein Soll-Ist-Abgleich notwendig ist, kann man den Momentenbetrieb statt mit RESET_TORQUE_LIMITS auch mit SET_TORQUE_LIMITS deaktivieren:

```
SET_TORQUE_LIMITS(1, {lower -1E10, upper 1E10, monitor #on} )
```

- Vorteil: Kann während einer Bewegung verwendet werden ("fliegend").
- Nachteil: Wenn durch die Momentenbegrenzung ein relativ großer Schleppfehler entstanden ist, beschleunigt der Roboter sehr stark. Hierdurch können Überwachungen anschlagen und das Programm stoppen.

Die Deaktivierung über SET_TORQUE_LIMITS ist in den meisten Fällen nicht geeignet.

6.18.4 Interpreterspezifika

Beschreibung

- SET_TORQUE_LIMITS() und RESET_TORQUE_LIMITS() können in Roboterprogrammen und in Submit-Programmen verwendet werden.
- Die Anweisungen sind interpreterspezifisch, d. h. sie wirken nur in dem Interpreter, in dem sie verwendet wurden.
- SET_TORQUE_LIMITS() kommt erstmals zur Wirkung, wenn die Achse für den Interpreter bewegt wird, der die Anweisung absetzt. Beispiel:
  a. Für eine Zusatzachse wird im Roboterprogramm der Momentenbetrieb aktiviert.
  b. Die Zusatzachse wird über ein Submit-Programm bewegt. Der Momentenbetrieb wirkt nicht.
  c. Die Zusatzachse wird über ein Roboterpogramm bewegt. Der Momentenbetrieb wirkt.
- Wenn der Momentenbetrieb bereits aktiv ist, wirkt SET_TORQUE_LIMITS() sofort.
- SET_TORQUE_LIMITS() wirkt sofort, wenn eine Bewegung aktiv ist. Deshalb können in Roboterprogrammen sowohl innerhalb als auch außerhalb eines Interrupts zu beliebiger Zeit die Momentengrenzen umgesetzt werden sowie die Überwachungen aus- und eingeschaltet werden.

  Es ist also möglich, den Momentenbetrieb rein innerhalb eines Interrupt-Programms zu verwenden. (Wenn RESET_TORQUE_LIMITS() verwendet wird, kann es notwendig sein, danach mit PTP $AXIS_RET wieder zur Unterbrechungsposition zurückzukehren.)

  (>>> 6.18.6.3 "Roboterprogramm: Momentenbetrieb im Interrupt" Seite 224)
- Beim "Besitzerwechsel" einer momentenbetriebenen Achse wird die Sollposition an die Istposition angeglichen.

  "Besitzerwechsel" bedeutet: Ein Interpreter hat die Achse im Momentenbetrieb bewegt (und "besitzt" sie somit). Während der Momentenbetrieb aktiv ist, wird die Achse von einem anderen Interpreter bewegt.

  Hauptanwendungsfall ist hierbei: Handverfahren, nachdem ein Programm im Momentenbetrieb unterbrochen wurde

Beispiel

Das folgende Beispiel zeigt, wann SET_TORQUE_LIMITS() wirkt, je nachdem, ob der Momentenbetrieb bereits aktiv ist oder nicht.

Ausgangssituation (Default): Die Überwachungen sind eingeschaltet.

```
1 SET_TORQUE_LIMITS(1, {monitor #off})
2 HALT
3 PTP_REL {A1 10}
4 HALT
5 SET_TORQUE_LIMITS(1, {monitor #on})
6 HALT
7 PTP_REL {A1 15}
```

Zeile	Beschreibung
1	Die Überwachungen für A1 werden ausgeschaltet.
2	Hier sind die Überwachungen noch eingeschaltet.
3	Die Achse wird bewegt. Ab jetzt wirkt die Anweisung SET_TORQUE_LIMITS.

Zeile	Beschreibung
4	Die Überwachungen sind ausgeschaltet.
5	Die Überwachungen werden eingeschaltet.
6	Hier sind die Überwachungen bereits eingeschaltet. Weil die Momentenbegrenzung bereits aktiv war, wirkte die Anweisung sofort und nicht erst ab der nächsten Bewegung dieser Achse.

6.18.5 Diagnosevariablen für den Momentenbetrieb

Alle diese Variablen bzw. Konstanten sind schreibgeschützt.

Ihr Wert ist unabhängig vom Interpreter.

6.18.5.1 $TORQUE_AXIS_ACT

Variable $TORQUE_AXIS_ACT[*Achsnummer*]

Datentyp: REAL

Beschreibung Aktuelles Motormoment für Achse [*Achsnummer*]

Einheit: Nm (für Linearachsen: N)

Der Wert hat nur Aussagekraft, wenn die Bremsen offen sind. Wenn die Bremsen geschlossen sind, ist er nahe Null.

Vorlaufstopp: Im Roboterprogramm löst die Variable einen Vorlaufstopp aus.

(>>> 6.18.5.6 "Vergleich: $TORQUE_AXIS_ACT und $HOLDING_TORQUE " Seite 222)

$BRAKE_SIG Der Zustand der Bremsen kann über die Systemvariable $BRAKE_SIG angezeigt werden. Der Wert von $BRAKE_SIG ist ein Bitfeld: Bit 0 entspricht A1, Bit 6 entspricht E1.

- Bit n = 0: Bremse ist geschlossen.
- Bit n = 1: Bremse ist offen.

6.18.5.2 $TORQUE_AXIS_MAX_0

Konstante $TORQUE_AXIS_MAX_0[*Achsnummer*]

Datentyp: REAL

Beschreibung Maximales Dauer-Motormoment für Achse [*Achsnummer*] bei Geschwindigkeit 0

Der Wert gibt ein Intervall an: Von -*Wert* bis +*Wert*.

Einheit: Nm (für Linearachsen: N)

Vorlaufstopp: Löst keinen Vorlaufstopp aus.

`lower` muss kleiner oder gleich dem oberen Wert des Intervalls von $TORQUE_AXIS_MAX_0 sein.
`upper` muss größer oder gleich dem unteren Wert des Intervalls von $TORQUE_AXIS_MAX_0 sein.
Wenn die Grenzen anders gesetzt werden, gibt die Robotersteuerung eine Fehlermeldung aus, die der Benutzer quittieren muss.

6.18.5.3 $TORQUE_AXIS_MAX

Konstante $TORQUE_AXIS_MAX[*Achsnummer*]

Datentyp: REAL

Beschreibung Absolut maximales Motormoment für Achse [*Achsnummer*]

Der Wert gibt ein Intervall an: Von -*Wert* bis +*Wert*.

Einheit: Nm (für Linearachsen: N)

Vorlaufstopp: Löst keinen Vorlaufstopp aus.

6.18.5.4 $TORQUE_AXIS_LIMITS

Variable $TORQUE_AXIS_LIMITS[*Achsnummer*]

Datentyp: TorqLimitParam

Beschreibung Aktuell wirkende Momentenbegrenzungen für Achse [*Achsnummer*]

Einheit: Nm (für Linearachsen: N)

Die Variable ist vornehmlich für die Diagnose über die Variablenkorrektur oder Variablenübersicht gedacht.

Eigenschaften:

- Wenn aktuell keine Begrenzungen gelten, bleiben `upper` und `lower` uninitialisiert.
- Die Komponente `monitor` ist immer initialisiert, außer die Achse existiert nicht.

  Dies ist z. B. relevant für 4-Achs- und 5-Achs-Roboter: Wenn man das ganze Feld anzeigt, können die nicht existierenden Achsen leicht identifiziert werden.

  Nicht existierende Zusatzachsen werden bei der Anzeige des gesamten Felds gar nicht angezeigt.
- `Max_vel` und `max_lag` sind uninitialisiert, wenn `monitor` = #ON ist, da in diesem Fall die regulären Überwachungen aktiv sind.

  Wenn monitor = #OFF ist, werden die Werte von `max_vel` und `max_lag` angezeigt, unabhängig davon, ob diese im aktuellen Programm explizit gesetzt wurden oder ob die Default-Werte verwendet werden.

Dass hier ggf. Komponenten uninitialisiert bleiben, erleichtert dem Benutzer die Diagnose.
Wenn jedoch über KRL auf die Variable zugegriffen wird, kann es sein, dass die Robotersteuerung den Zugriff als "ungültig" betrachtet. Empfehlung: Vor dem Zugriff den Zustand der Variablen mit VARSTATE() prüfen.

Vorlaufstopp: Im Roboterprogramm löst die Variable einen Vorlaufstopp aus.

6.18.5.5 $HOLDING_TORQUE

Variable $HOLDING_TORQUE[*Achsnummer*]

Datentyp: REAL

Beschreibung Haltemoment für die Roboterachse [*Achsnummer*]

Einheit: Nm

Das Haltemoment bezieht sich auf die aktuelle Ist-Achsposition und die aktuelle Last.

(Für Zusatzachsen wird immer der Wert 0 N zurückgegeben.)

Vorlaufstopp: Im Roboterprogramm löst die Variable einen Vorlaufstopp aus.

(>>> 6.18.5.6 "Vergleich: $TORQUE_AXIS_ACT und $HOLDING_TORQUE " Seite 222)

Wenn die obere und untere Momentengrenze für alle Achsen auf $HOLDING_TORQUE gesetzt ist, muss der Roboter bei offenen Bremsen stehenbleiben.
Wenn dies nicht der Fall ist, d. h. wenn der Roboter driftet, ist die Last nicht korrekt konfiguriert.

6.18.5.6 Vergleich: $TORQUE_AXIS_ACT und $HOLDING_TORQUE

Bei einem Roboter, der mit offenen Bremsen steht, ist $TORQUE_AXIS_ACT nicht gleich $HOLDING_TORQUE, obwohl man dies vermuten könnte.

Eigenschaften **$HOLDING_TORQUE**:

- Ist ein berechneter Wert, der die Reibung nicht berücksichtigt. Regelungseffekte haben keinen Einfluss auf den Wert.
- Ist nur abhängig von der aktuellen Position der Achse. Ist somit bei gleichbleibender Position unveränderlich.

Eigenschaften **$TORQUE_AXIS_ACT**:

- Wird aus den Ist-Strömen ermittelt. Reibung und Regelungseffekte wirken sich deshalb auf den Wert aus.
- Kann sich bei gleichbleibender Position der Achse ändern.

Die Abweichung zwischen $HOLDING_TORQUE und $TORQUE_AXIS_ACT ist kleiner oder gleich der aktuellen Reibung, wenn der Roboter seit mindestens 1 sec steht. Voraussetzung ist, dass die Lastdaten korrekt sind.

6.18.6 Weitere Beispiele

6.18.6.1 Roboterprogramm: Achse in beide Richtungen weichschalten

Beschreibung

Der Roboter muss ein Werkstück in einer Presse greifen, das dann von der Presse ausgestoßen wird. Damit der Roboter nachgeben und den Ausstoß abfangen kann, schaltet man die Achse weich.

Die Momentengrenzen müssen dazu auf ein sehr kleines Intervall um das Haltemoment der Achse gelegt werden. (Das Haltemoment ist nur bei A1 = 0 Nm. Zum Weichschalten können also die Grenzen nicht generell auf 0 gesetzt werden.)

Annahme für dieses Beispiel: Eine Drehung um die Achse [*ideal_axis*] bewegt den Greifer nahezu exakt in Ausstoßrichtung.

Programm

```
 1 DECL TorqLimitParam myLimits
 2 DECL INT ideal_axis
   ...
 3 myLimits.monitor = #off
 4 myLimits.lower = $holding_torque[ideal_axis] - 10
 5 myLimits.upper = $holding_torque[ideal_axis] + 10
 6 SET_TORQUE_LIMITS(ideal_axis, myLimits)
 7 PTP $AXIS_ACT
 8 OUT_SIGNAL_SOFT = TRUE
 9 WAIT FOR IN_SIGNAL_EJECTED
10 RESET_TORQUE_LIMITS(ideal_axis)
11 OUT_SIGNAL_SOFT = FALSE
12 WAIT FOR IN_SIGNAL_NEXTMOVE
   ...
```

Zeile	Beschreibung
3 …6	Für die Achse[*ideal_axis*] werden die regulären Überwachungen ausgeschaltet und die Momente auf ein sehr kleines Intervall um das Haltemoment begrenzt.
7	Eine "Bewegung" auf die aktuelle Position durchführen, damit die Reduzierung der Stromgrenzen aktiv wird. Die Achse[idealaxis] kann nun durch den Ausstoßer der Presse bewegt werden.
8	Signal an die Pressensteuerung, dass die Achse für den Ausstoß bereit ist
9	Warten auf Signal der Pressensteuerung, dass das Werkstück ausgestoßen wurde
10	Die Momentenbegrenzung aufheben und die regulären Überwachungen wieder einschalten. Außerdem die Sollposition an die Istposition angleichen.
11	Signal an die Pressensteuerung, dass die Achse nicht mehr für einen Ausstoß bereit ist
12	Eventuell: Warten auf Signal, dass das Wegfahren aus der Position erlaubt ist

6.18.6.2 Roboterprogramm: Bei Kollisionen Schäden vermeiden

Beschreibung

Die Momentenbegrenzung kann man nutzen, um bei Kollisionen Schäden zu vermeiden.

- Vorteil: Es ist sichergestellt, dass der Roboter nur mit einer bestimmten, begrenzten Kraft gegen das Hindernis drückt.
- Nachteil: Der Roboter wird träge. Es sind keinen hohen Beschleunigungen mehr möglich.

Programm

Der Roboter holt Werkstücke aus einer Kiste. Bei der Bewegung zu den Punkten P7, P8 und P9 kann nicht ausgeschlossen werden, dass er mit dem Werkstück an der Kiste hängenbleibt. Es soll sichergestellt werden, dass der Roboter nicht so stark drückt, dass ein Schaden entstehen kann. Hierzu werden vor den kritischen Punkten die Kräfte begrenzt.

Die regulären Überwachungen werden abgeschaltet. Nicht, weil sie sonst unnötig anschlagen würden, sondern weil sie im Gegenteil für dieses Beispiel nicht streng genug sind. Stattdessen wird eine der Spezialüberwachungen auf einen sehr geringen Wert gesetzt. (Abhängig vom konkreten Anwendungsfall kann es auch sinnvoll sein, die regulären Überwachungen zu verwenden.)

```
   ...
 1 DECL TorqLimitParam myParams
   ...
 2 FOR i = 1 to 6
 3   myParams.lower = $holding_torque[i] - 500
 4   myParams.upper = $holding_torque[i] + 500
 5   myParams.monitor = #off
 6   myParams.max_lag = 0.1
 7 SET_TORQUE_LIMITS(i, myParams)
 8 ENDFOR
 9 $acc.cp = my_low_acceleration
10 $vel.cp = my_low_velocity
11 LIN P7
12 LIN P8
13 LIN P9
14 FOR i = 1 to 6
```

```
15   myParams.lower = -1E10
16   myParams.upper = 1E10
17   myParams.monitor = #on
18 SET_TORQUE_LIMITS(i, myParams)
19 ENDFOR
20 $acc.cp = my_high_acceleration
21 $vel.cp = my_high_velocity
22 LIN P10
   ...
```

Zeile	Beschreibung
2 … 7	Die Momente für A1 … A6 werden begrenzt.
3, 4	Die Grenzen werden auf ein eher kleines Intervall gesetzt, mit dem Haltemoment in der Mitte.
5, 6	Die regulären Überwachungen werden abgeschaltet. `Max_lag = 0.1` bewirkt, dass bereits bei einem Schleppfehler von 0,1 ° ein Stopp ausgelöst wird.
9, 10	Beschleunigung und Geschwindkeit werden verringert, damit der Roboter langsam auf die kritischen Punkte zufährt.
11 …13	Punkte, an denen eine Kollision auftreten könnte Wenn eine Kollision auftritt, spricht die Überwachung `max_lag` an und der Bediener der Anlage kann eingreifen.
Nach dem kritischen Abschnitt:	
14 …19	Die Momentenbetrieb wird deaktiviert. Hier kann SET_TORQUE_LIMITS verwendet werden: Der Roboter gelangt nur an diese Stelle, wenn er die kritischen Punkte ohne Kollision passiert hat. In diesem Fall hat sich kein Schleppfehler aufgebaut und ein Soll-Ist-Abgleich ist nicht notwendig.
20, 21	Beschleunigung und Geschwindkeit werden wieder hochgesetzt.
22	Unkritischer Punkt

6.18.6.3 Roboterprogramm: Momentenbetrieb im Interrupt

Beschreibung

Hier wird gezeigt, dass der Momentenbetrieb komplett innerhalb eines Interrupt-Programms verwendet werden kann. Das Beispiel ist nicht in erster Linie für die Praxis gedacht, sondern soll zeigen, dass diese Verwendung grundsätzlich möglich ist.

Programm

Der Roboter soll die Position eines Werkstück ermitteln, von dem man recht genau weiß, wo es liegen kann, aber nicht die exakte Position. Zuerst teilt ein Näherungssensor der Robotersteuerung mit, wenn der Roboter in die Nähe des Werkstücks kommt. Aufgrund dieses Signals vom Sensor wird dann ein Interrupt-Programm aufgerufen.

Im Interrupt-Programm findet die eigentliche "Suche" statt: Die A1 bewegt sich auf das Werkstück zu. Vorher wird für A1 der Momentenbetrieb aktiviert. Dort, wo der Roboter auf das Werkstück stößt, bleibt er nun wegen der reduzierten Momente daran hängen: Das Werkstück wurde "gefunden". Diese Position kann nun als Position des Werkstücks gespeichert werden.

```
   ...
1 LIN P1
2 INTERRUPT DECL 1 WHEN sensor_signal==true DO search()
3 INTERRUPT ON 1
4 LIN P2
5 INTERRUPT OFF 1
```

```
   ...
-------------------------
 6 DEF search()
 7 ...
 8 BRAKE
 9 SET_TORQUE_LIMITS(1,{lower 1000, upper 1000, monitor #off})
10 PTP_REL {A1 10}
11 RESET_TORQ_LIMITS(1)
12 piece_found = $POS_ACT_MES
13 PTP $AXIS_RET
14 END
```

Zeile	Beschreibung
1 … 4	Auf dem Weg nach P2 soll der Sensor signalisieren, wenn sich der Roboter in der Nähe des Werkstücks befindet.
2	Wenn das Sensorsignal kommt, wird das Unterprogramm search() aufgerufen.
Im Unterprogramm:	
8	Die aktuelle Bewegung wird gestoppt, sobald search() ausgeführt wird.
9	Momentengrenzen für A1 setzen, reguläre Überwachungen ausschalten.
10	A1 bewegt sich. Dort, wo der Roboter auf das Werkstück stößt, bleibt er wegen der reduzierten Momente daran hängen. Wenn der Roboter seine Sollposition {A1 10} erreicht hat, wird die nächste Programmzeile abgearbeitet. (Dass die Istposition von der Sollposition abweicht, hat keine Auswirkung, da ja die regulären Überwachungen abgeschaltet sind.)
11	Die Momentenbegrenzung aufheben und die regulären Überwachungen wieder einschalten. Außerdem die Sollposition an die Istposition angleichen.
12	Die aktuelle Position des Roboters gibt nun die Position des Werkstücks an. Diese wird hier in eine Variable gespeichert.
13	An die Position zurückkehren, an der der Roboter die Bahn im Hauptprogramm verlassen hat.

6.18.6.4 Roboterprogramm: Servozange baut Druck auf

Beschreibung Dieses Programm zeigt, wie der Momentenbetrieb über Trigger aktiviert werden kann. (Vergleichbare Programme werden in den Technologiepaketen KUKA.ServoGun im Hintergrund verwendet. Sie müssen also nicht vom Benutzer programmiert werden.)

Programm **Hauptprogramm:**

```
 1 DEF SPOT()
 2 DECL BOOL error_occurred
   ...
 3 Interrupt DECL 1 WHEN $stopmess DO resume_subprog()
 4 Interrupt ON 1
 5 REPEAT
 6  error_occurred = false
 7  SPOT_MOVE()
 8 UNTIL error_occurred == false
   ...
```

Zeile	Beschreibung
3	Wenn ein Fehler auftritt, soll resume_subprog() aufgerufen werden.
7	Das Schweißprogramm SPOT_MOVE() wird aufgerufen.
5 … 8	Wenn ein Fehler aufgetreten ist (also wenn error_occurred == true), wird SPOT_MOVE() wiederholt.

Schweißprogramm:

```
 1 DEF SPOT_MOVE()
   ...
 2 TorqLimWeld = {lower -1000, upper 1000 , monitor #off}
 3 i = 6+EG_EXTAX_ACTIVE
   ...
 4 LIN P_APPROX C_DIS
 5 $VEL_EXTAX[EG_EXTAX_ACTIVE]=EG_MAX_CONST_VEL[EG_EXTAX_ACTIVE]
 6 LIN P_APPROX C_DIS
 7 TRIGGER WHEN DISTANCE=0 DELAY=50 DO SET_TORQUE_LIMITS(i,
TorqLimWeld) PRIO = -1
 8 LIN P_PART C_DIS
 9 TRIGGER WHEN DISTANCE=0 DELAY=50 DO START_TIMER_SPOT() PRIO=82
10 LIN P_PRESSURE C_DIS
11 LIN P_WELD
12 WAIT FOR EG_TRIGGER_END
13 RESET_TORQUE_LIMITS(i)
14 Interrupt OFF 1
15 LIN P_PART C_DIS
16 END
```

Zeile	Beschreibung
7	Kurz bevor die Zange das Werkstück berührt, werden die Momente reduziert.
9 … 11	Druck aufbauen und dann schweißen.
12	Der Schweißtimer signalisiert das Schweißende.
13	Die Momentenbegrenzung aufheben und die regulären Überwachungen wieder einschalten. Außerdem die Sollposition an die Istposition angleichen.

Interrupt-Programm für den Fehlerfall:

```
 1 DEF resume_subprog()
 2 BRAKE
 3 Suppress_repositioning()
 4 HALT
 5 error_occurred = true
 6 RESUME
 7 END
```

Zeile	Beschreibung
3	Normalerweise würde nach HALT beim Wieder-Anfahren rückpositioniert auf die Position, an der der Interrupt aufgetreten ist. (Wegen $STOPMESS.) Suppress_repositioning() verhindert diese Rückpositionierung. Suppress_repositioning() kann je nach Applikation nützlich sein, muss aber nicht.
5	error_occurred auf TRUE setzen, damit die REPEAT-Schleife im Hauptprogramm wiederholt wird.
6	RESUME deaktiviert den Momentenbetrieb. Rücksprung in Zeile 8 ins Hauptprogramm.

6.18.6.5 Submit-Programm: Servozange baut Druck auf

Voraussetzung

- E1 ist bei `ASYPTP {E1 10}` bereits asychron.
- Oder: $ASYNC_MODE ist so konfiguriert (Bit 0 = 1), dass bei ASYPTP im Submit-Programm die Achse implizit synchron gesetzt wird.

Programm

```
   ...
1  IF $PRO_STATE1==#P_FREE
2    SET_TORQUE_LIMITS(7,{upper 1000, monitor #off })
3    ASYPTP {E1 10}
   ...
4    RESET_TORQUE_LIMITS(7)
5    ASYPTP {E1 -10}
6  ENDIF
   ...
```

Zeile	Beschreibung
1	Sicherstellen, dass kein Roboterprogramm angewählt ist.
2	Das positive Moment begrenzen und die regulären Überwachungen ausschalten.
3	Bewegung in Richtung des Zielpunkts {E1 10} hinter dem Werkstück. Der Druck auf das Werkstück baut sich auf.
4	Die Momentenbegrenzung aufheben und die regulären Überwachungen wieder einschalten. Außerdem die Sollposition an die Istposition angleichen. Der Interpreter wartet in RESET_TORQUE_LIMITS(7), bis die asynchrone Bewegung beendet ist. Erst dann führt er den Soll-Ist-Abgleich durch. Daher ist es nicht notwendig, vor RESET… `WAIT FOR $ASYNC_STATE == #IDLE` zu programmieren.
5	Zange wieder öffnen.

6.19 Aktionsplaner

Mit dieser Funktion kann der Datenabgleich zwischen Grundsystem und Festplatte zeitlich oder aktionsbezogen gesteuert werden. Beim Datenabgleich werden die Daten des Grundsystems auf die Festplatte geschrieben.

6.19.1 Datenabgleich konfigurieren

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Konfiguration** > **Extras** > **Aktionsplaner** wählen.
2. Im linken Teil des Fensters die Baumstruktur aufklappen.
3. Die gewünschte Aktion markieren:
   - **T1 und T2 Konsistenz**: Daten in den Betriebsarten T1 und T2 in regelmäßigen Zeitintervallen abgleichen.
   - **AUT und EXT Konsistenz**: Daten in der Betriebsart Automatik Extern in regelmäßigen Zeitintervallen abgleichen.
   - **Logische Konsistenz**: Daten nach einem Wechsel der Betriebsart oder nach Online-Optimierung abgleichen.

     Eine Online-Optimierung liegt vor, wenn im laufenden Betrieb die Parameter eines Programms geändert werden.
4. Im rechten Teil des Fensters gewünschte Einstellungen vornehmen.

(>>> 6.19.2 "Konfiguration T1 und T2 Konsistenz, AUT und EXT Konsistenz" Seite 228)

(>>> 6.19.3 "Konfiguration Logische Konsistenz" Seite 229)

5. **Speichern** drücken.

6.19.2 Konfiguration T1 und T2 Konsistenz, AUT und EXT Konsistenz

Hier sind folgende Einstellungen möglich:

- Datum und Uhrzeit festlegen, wann die Daten zwischen Grundsystem und Festplatte erstmals abgeglichen werden.
- Zeitintervall festlegen, in dem dieser Vorgang wiederholt wird.

Beschreibung

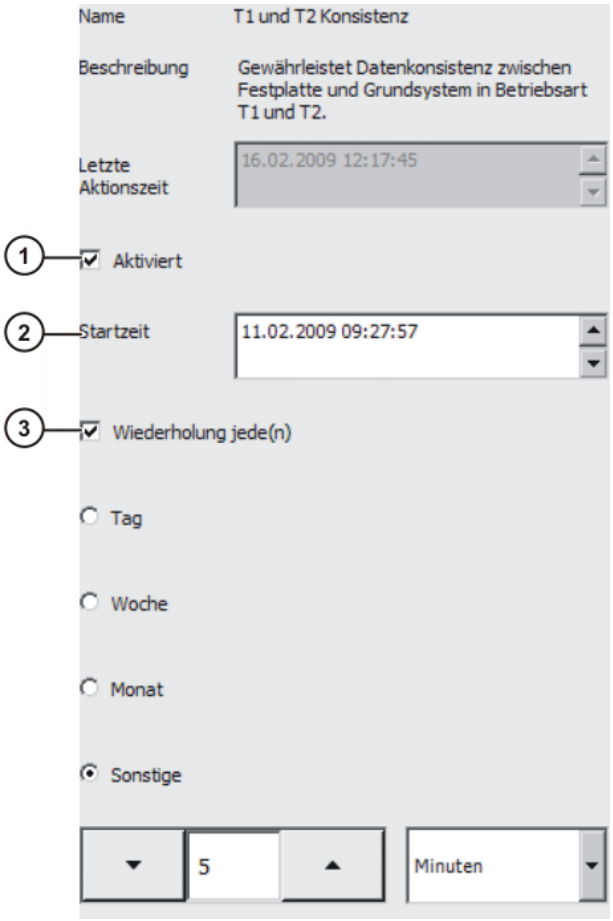


Abb. 6-29: Konfiguration T1 und T2 Konsistenz

Pos.	Beschreibung
1	■ **Checkbox aktiv**: Datenabgleich ist aktiviert. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
2	Datum und Uhrzeit im angezeigten Format eingeben.
3	■ **Checkbox aktiv**: Zeitintervall ist aktiviert. ■ **Checkbox inaktiv**: Zeitintervall ist deaktiviert.

HINWEIS Zu klein gewählte Zeitintervalle können zu Schäden an der Festplatte führen. Ein Zeitintervall im Minutenbereich wird empfohlen.

6.19.3 Konfiguration Logische Konsistenz

Beschreibung Hier sind folgende Einstellungen möglich:

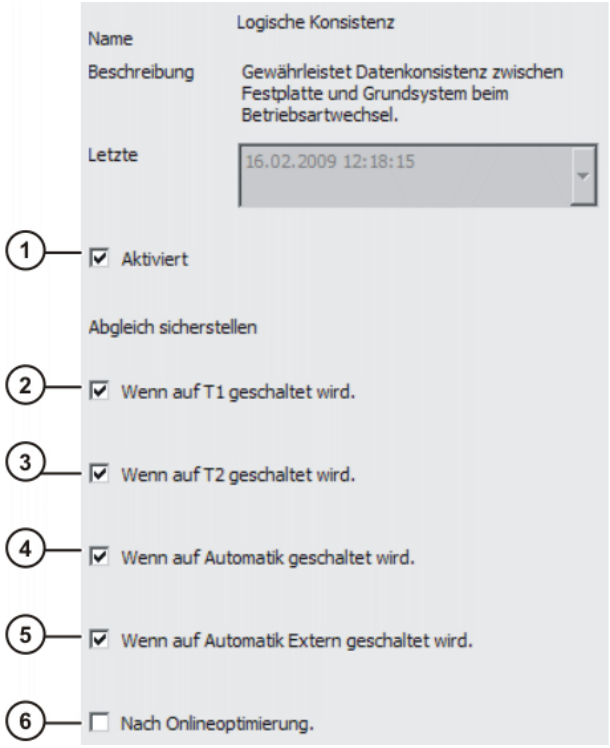


Abb. 6-30: Konfiguration Logische Konsistenz

Pos.	Beschreibung
1	■ **Checkbox aktiv**: Datenabgleich ist aktiviert. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
2	■ **Checkbox aktiv**: Daten werden abgeglichen, wenn in die Betriebsart T1 gewechselt wird. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
3	■ Checkbox aktiv: Daten werden abgeglichen, wenn in die Betriebsart T2 gewechselt wird. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
4	■ **Checkbox aktiv**: Daten werden abgeglichen, wenn in die Betriebsart Automatik gewechselt wird. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
5	■ **Checkbox aktiv**: Daten werden abgeglichen, wenn in die Betriebsart Automatik Extern gewechselt wird. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.
6	■ **Checkbox aktiv**: Daten werden nach einer Online-Optimierung abgeglichen. ■ **Checkbox inaktiv**: Datenabgleich ist deaktiviert.

6.20 Bremsentest

6.20.1 Übersicht Bremsentest

Beschreibung Jede Roboterachse hat eine in den Motor integrierte Haltebremse. Der Bremsentest prüft für jede Bremse bei niedriger Drehzahl und aktueller Temperatur, ob das Bremsmoment ausreichend hoch ist, d. h. ob es einen bestimmten Mi-

nimalwert überschreitet. Der Minimalwert für die einzelnen Achsen ist in den Maschinendaten hinterlegt. (Der Bremsentest ermittelt nicht den Absolutwert des Bremsmoments.)

Anforderung

Wenn der Bremsentest aktiv ist, dann fordern folgende Ereignisse einen Bremsentest an:

- Eingang $BRAKETEST_REQ_EX von extern, z. B. von einer SPS (externe Anforderung)
- Robotersteuerung startet mit Kaltstart (interne Anforderung)
- Funktionstest des Bremsentests (interne Anforderung)
- Bremsentest-Zykluszeit ist abgelaufen (interne Anforderung)

Zykluszeit

Die Zykluszeit beträgt 46 h. Sie ist abgelaufen, wenn die Antriebe insgesamt 46 h in Regelung waren. Die Robotersteuerung fordert dann einen Bremsentest an und gibt folgende Meldung aus: *Bremsentest erforderlich*. Der Roboter lässt sich weitere 2 Stunden verfahren. Danach stoppt er und die Robotersteuerung gibt folgende Quittiermeldung aus: *Prüfzyklus für Bremsentestanforderung nicht eingehalten*. Nach dem Quittieren kann der Roboter jeweils weitere 2 Stunden verfahren werden.

Durchführung

Voraussetzung für den Bremsentest ist, dass der Roboter Betriebstemperatur hat. Dies ist nach ca. 1 h im normalen Betrieb der Fall.

Der Bremsentest wird mit dem Programm BrakeTestReq.SRC durchgeführt. Er kann auf folgende Arten gestartet werden:

- **Automatisch**

  Hierfür das Programm BrakeTestReq.SRC so in das Applikationsprogramm einbinden, dass es zyklisch als Unterprogramm aufgerufen wird. Wenn ein Bremsentest angefordert wird, erkennt das Programm dies und startet den Bremsentest.
- **Manuell**

  Das Programm BrakeTestReq.SRC manuell starten.

Ablauf

Der Bremsentest prüft nacheinander alle Bremsen.

1. Der Roboter beschleunigt auf eine definierte Geschwindigkeit. (Die Geschwindigkeit kann vom Benutzer nicht beeinflusst werden.)
2. Wenn der Roboter die Geschwindigkeit erreicht hat, fällt die Bremse ein und das Ergebnis für diesen Bremsvorgang wird im Meldungsfenster angezeigt.
3. Wenn eine Bremse als defekt erkannt wurde, kann der Bremsentest zur Kontrolle wiederholt werden oder der Roboter in die Parkposition gefahren werden.

   Wenn eine Bremse die Verschleißgrenze erreicht hat, zeigt die Robotersteuerung dies mit einer Meldung an. Eine verschlissene Bremse wird in absehbarer Zeit als defekt erkannt werden. Der Roboter kann bis dahin ohne Einschränkungen verfahren werden.

Wenn eine Bremse als defekt erkannt wird, sind die Antriebe nach Beginn des Bremsentests noch 2 h in Regelung (= Monitoring-Zeit). Danach schaltet die Robotersteuerung die Antriebe ab.

Übersicht

Schritt	Beschreibung
In WorkVisual:	
1	Bei Bedarf: Den Bremsentest in WorkVisual aktivieren. (>>> 6.20.2 "Bremsentest aktivieren" Seite 231)
Auf der Robotersteuerung:	

Schritt	Beschreibung
2	Ein- und Ausgangssignale für den Bremsentest konfigurieren. (>>> 6.20.4 "Ein- und Ausgangssignale für den Bremsentest konfigurieren" Seite 232)
3	Positionen für Bremsentest teachen. Die Parkposition muss geteacht werden. Start- und Endposition können geteacht werden. (>>> 6.20.5 "Positionen für Bremsentest teachen" Seite 235)
4	Wenn der Bremsentest automatisch durchgeführt werden soll: Programm BrakeTestReq.SRC so in das Applikationsprogramm einbinden, dass es zyklisch als Unterprogramm aufgerufen wird.
5	Wenn der Bremsentest manuell durchgeführt werden soll: Programm BrakeTestReq.SRC manuell starten. (>>> 6.20.6 "Bremsentest manuell durchführen" Seite 236)
6	Bei Bedarf: Bremsentest auf Funktion testen (>>> 6.20.7 "Bremsentest auf Funktion testen" Seite 237)

6.20.2 Bremsentest aktivieren

- Wenn eine Sicherheitsoption installiert ist und die sichere Überwachung aktiv ist, ist der Bremsentest automatisch aktiv.
- Wenn der Bremsentest nicht automatisch aktiv ist, hat der Benutzer die Möglichkeit, ihn gezielt zu aktivieren. Dies muss in WorkVisual durchgeführt werden.

Wenn der Bremsentest nicht automatisch aktiv ist, muss der Benutzer durch eine Risikobetrachtung ermitteln, ob es in seinem konkreten Anwendungsfall erforderlich ist, den Bremsentest zu aktivieren.

Weitere Informationen zur Aktivierung des Bremsentests sind in der Dokumentation zu WorkVisual zu finden.

6.20.3 Programme für den Bremsentest

Die Programme befinden sich im Verzeichnis C:\KRC\ROBOTER\KRC\R1\TP\BrakeTest.

Programm	Beschreibung
BrakeTestReq.SRC	Dieses Programm führt den Bremsentest durch. Es gibt folgende Möglichkeiten für die Durchführung: ■ Das Programm so in das Applikationsprogramm einbinden, dass es zyklisch als Unterprogramm aufgerufen wird. Wenn ein Bremsentest angefordert wird, erkennt das Programm dies und führt den Bremsentest sofort aus. ■ Das Programm manuell ausführen. ■ Den Bremsentest auf Funktion testen. Hierbei führt die Robotersteuerung BrakeTestReq.SRC mit einer speziellen Parametrierung aus.
BrakeTestPark.SRC	In diesem Programm muss die Parkposition des Roboters geteacht werden. Die Parkposition kann man anfahren, wenn eine Bremse als defekt erkannt wurde. Alternativ kann man den Bremsentest zur Kontrolle wiederholen.
BrakeTestStart.SRC	In diesem Programm kann die Startposition des Bremsentests geteacht werden. Von dieser Position aus führt der Roboter den Bremsentest aus. Wenn die Startposition nicht geteacht wird, führt der Roboter den Bremsentest an der Istposition aus.
BrakeTestBack.SRC	In diesem Programm kann die Endposition des Bremsentests geteacht werden. Der Roboter fährt diese Position nach dem Bremsentest an. Wenn die Endposition nicht geteacht wird, bleibt der Roboter nach dem Bremsentest an der Istposition.
BrakeTestSelf-Test.SRC	Das Programm prüft, ob der Bremsentest eine defekte Bremse korrekt erkennt. Zu diesem Zweck führt die Robotersteuerung BrakeTestReq.SRC mit einer speziellen Parametrierung aus.

6.20.4 Ein- und Ausgangssignale für den Bremsentest konfigurieren

Beschreibung

Alle Signale für den Bremsentest sind im Verzeichnis KRC:\STEU\MADA in der Datei $machine.dat deklariert.

⚠ WARNUNG Diese Signale sind nicht redundant aufgebaut und können falsche Informationen liefern. Diese Signale nicht für sicherheitsrelevante Applikationen verwenden.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Navigator im Verzeichnis KRC:\STEU\MADA die Datei $machine.dat öffnen.
2. Ein- und Ausgänge zuordnen.
3. Datei speichern und schließen.

$machine.dat

Auszug aus der Datei $machine.dat (mit Default-Einstellungen, ohne Kommentare):

```
...
SIGNAL $BRAKETEST_REQ_EX $IN[1026]
SIGNAL $BRAKETEST_MONTIME FALSE
...
SIGNAL $BRAKETEST_REQ_INT FALSE
SIGNAL $BRAKETEST_WORK FALSE
SIGNAL $BRAKES_OK FALSE
```

```
SIGNAL $BRAKETEST_WARN FALSE
...
```

Signale

Es gibt 1 Eingangssignal. Dieses liegt defaultmäßig auf $IN[1026].

Die Ausgangssignale sind mit FALSE vorbelegt. Ihnen müssen nicht zwingend Ausgangsnummern zugeordnet werden. Es müssen nur Nummern zugeordnet werden, wenn gewünscht ist, die Signale lesen zu können. (Z. B. über die Variablenkorrektur oder den Programmlauf.)

Signal	Beschreibung
$BRAKETEST_REQ_EX	Eingang ■ **TRUE** = Bremsentest wird extern angefordert (z. B. von SPS). Die Robotersteuerung bestätigt das Signal mit $BRAKETEST_REQ_INT = TRUE und gibt die Meldung 27004 aus. ■ **FALSE** = Bremsentest wird nicht extern angefordert.
$BRAKETEST_MONTIME	Ausgang ■ **TRUE** = Roboter wurde aufgrund abgelaufener Monitoring-Zeit angehalten. Die Quittiermeldung 27002 wird ausgegeben. ■ **FALSE** = Die Quittiermeldung 27002 steht nicht an. (Wurde nicht ausgegeben oder wurde quittiert.)
$BRAKETEST_REQ_INT	Ausgang ■ **TRUE** = Die Meldung 27004 steht an. Das Signal wird erst wieder FALSE, wenn ein Bremsentest mit positivem Ergebnis durchgeführt wurde, also mit der Meldung 27012. ■ **FALSE** = Bremsentest wird nicht angefordert (weder intern noch extern).
$BRAKETEST_WORK	Ausgang ■ **TRUE** = Bremsentest wird gerade durchgeführt. ■ **FALSE** = Bremsentest wird nicht durchgeführt. Wenn keine defekte Bremse ermittelt wurde, wird die Meldung 27012 ausgegeben. Flanke **TRUE** → **FALSE**: ■ Der Test wurde erfolgreich durchgeführt. Keine Bremse ist defekt. Die Meldung 27012 wird ausgegeben. ■ Oder mindestens 1 defekte Bremse wurde ermittelt und die Parkposition wurde angefahren. ■ Oder das Programm wurde während der Ausführung des Bremsentests abgewählt.

Signal	Beschreibung
$BRAKES_OK	Ausgang ■ Flanke **FALSE** → **TRUE**: Aus dem vorhergehenden Bremsentest stand FALSE an. Der Bremsentest wurde erneut durchgeführt, und es wurde keine defekte Bremse ermittelt. ■ Flanke **TRUE** → **FALSE**: Eine Bremse wurde gerade als defekt erkannt. Die Meldung 27007 wird ausgegeben.
$BRAKETEST_WARN	Ausgang ■ Flanke **FALSE** → **TRUE**: Für mindestens 1 Bremse wurde ermittelt, dass die Verschleißgrenze erreicht ist. Gleichzeitig wird die Meldung 27001 ausgegeben. ■ Flanke **TRUE** → **FALSE**: Aus dem vorhergehenden Bremsentest stand TRUE an. Der Bremsentest wurde erneut durchgeführt, und es wurde keine verschlissene Bremse ermittelt.

Meldungen

Nr.	Meldung
27001	*Bremse {Bremse Nr.}{Achsnr.} hat die Verschleißgrenze erreicht*
27002	*Prüfzyklus für Bremsentestanforderung nicht eingehalten*
27004	*Bremsentest erforderlich*
27007	*Unzureichendes Haltemoment der Bremse {Bremse Nr.}{Achsnr.}*
27012	*Bremsentest erfolgreich durchgeführt*

6.20.4.1 Signalverlauf des Bremsentests – Beispiele

Beispiel 1

Der Signalverlauf des Bremsentests wird für folgenden Fall dargestellt:

- Keine Bremse hat die Verschleißgrenze erreicht.
- Keine Bremse ist defekt.

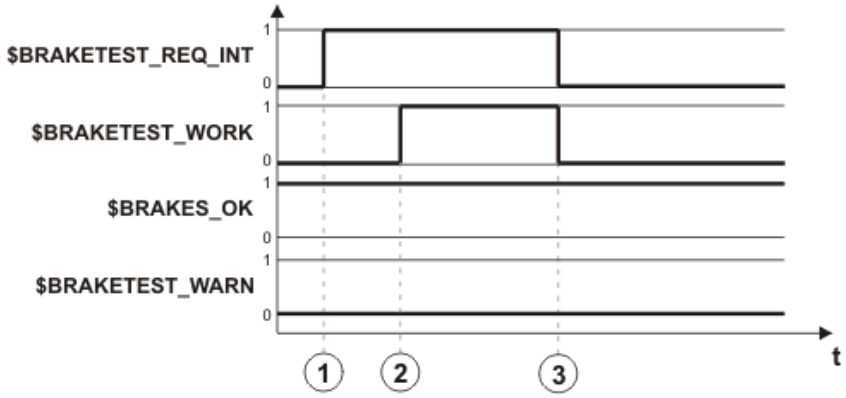


Abb. 6-31: Signalverlauf: Bremsen in Ordnung

Pos.	Beschreibung
1	Der Bremsentest wird angefordert.
2	Automatischer Aufruf des Programms BrakeTestReq.SRC Start des Bremsentests
3	Der Bremsentest ist beendet.

Beispiel 2

Der Signalverlauf des Bremsentests wird für folgenden Fall dargestellt:

- Bremse A2 ist verschlissen.

- Bremse A4 ist defekt.

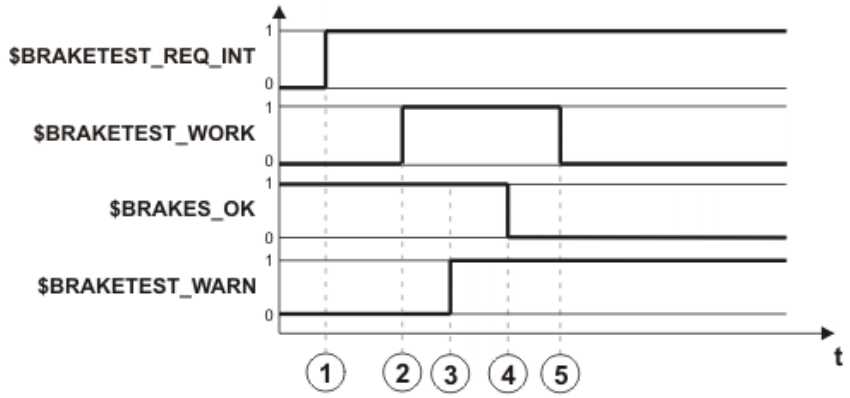


Abb. 6-32: Signalverlauf: Bremsen nicht in Ordnung

Pos.	Beschreibung
1	Der Bremsentest wird angefordert. $BRAKETEST_REQ_INT wird danach erst wieder FALSE, wenn ein Bremsentest mit positivem Ergebnis durchgeführt wurde.
2	Automatischer Aufruf des Programms BrakeTestReq.SRC Start des Bremsentests
3	Bremse A2 wird getestet: Bremse ist verschlissen.
4	Bremse A4 wird getestet: Bremse ist defekt.
5	Der Roboter wurde in die Parkposition gefahren, oder das Programm wurde abgewählt.

6.20.5 Positionen für Bremsentest teachen

Beschreibung

Die Parkposition muss geteacht werden.

Start- und Endposition können geteacht werden.

- Wenn die Startposition nicht geteacht wird, führt der Roboter den Bremsentest an der Istposition aus.
- Wenn die Endposition nicht geteacht wird, bleibt der Roboter nach dem Bremsentest an der Istposition.

Parkposition

Wenn eine Bremse als defekt erkannt wird, kann der Roboter in die Parkposition verfahren werden. Alternativ kann man den Bremsentest zur Kontrolle wiederholen.

⚠ WARNUNG Die Parkposition muss so gewählt werden, dass keine Personen gefährdet werden, falls der Roboter aufgrund der defekten Bremse zusammensackt. Als Parkposition kann z. B. die Transportstellung gewählt werden.
Weitere Informationen zur Transportstellung sind in der Betriebsanleitung oder der Montageanleitung für den Roboter zu finden.

Voraussetzung

- Alle Ausgangssignale sind Ausgängen zugeordnet.
- Benutzergruppe Experte
- Betriebsart T1

Vorgehensweise

1. Im Verzeichnis R1\TP\BrakeTest das Programm BrakeTestStart.SRC öffnen.
2. Die Bewegungen zur Startposition des Bremsentests teachen.

- Die Bewegungen müssen so geteacht werden, dass der Roboter auf dem Weg in die Startposition keine Kollision verursachen kann.
- In der Startposition muss jeder Roboterachse ein Bewegungsbereich von ±10° zur Verfügung stehen.

3. Programm speichern und schließen.
4. Im Verzeichnis R1\TP\BrakeTest das Programm BrakeTestBack.SRC öffnen.
5. Die Bewegungen von der Startposition zur Endposition des Bremsentests teachen.

   Start- und Endposition dürfen identisch sein.
6. Programm speichern und schließen.
7. Im Verzeichnis R1\TP\BrakeTest das Programm BrakeTestPark.SRC öffnen.
8. Die Bewegungen von der Endposition zur Parkposition des Roboters programmieren.
9. Programm speichern und schließen.

6.20.6 Bremsentest manuell durchführen

⚠ WARNUNG Wenn eine Bremse als defekt erkannt wird und die Antriebe abgeschaltet werden, kann der Roboter zusammensacken. Während der Bewegung zur Parkposition darf deshalb kein Stopp ausgelöst werden. Die Überwachungsfunktionen, die in diesem Bereich einen Stopp auslösen können (z. B. Überwachungsräume), müssen vorher deaktiviert werden. Es dürfen keine Sicherheitsfunktionen ausgeführt werden, die einen Stopp auslösen (z. B. NOT-HALT, Schutztür öffnen, Betriebsart wechseln).
Wenn eine Bremse als defekt erkannt wird, darf die Parkposition mit einer Geschwindigkeit von maximal 10 % der maximalen Geschwindigkeit angefahren werden.

⚠ WARNUNG Der Programm-Override wird beim Test automatisch auf 100 % gesetzt. Der Roboter verfährt mit hoher Geschwindigkeit. Darauf achten, dass der Roboter nicht kollidieren kann und dass sich keine Personen im Bewegungsbereich des Roboters befinden.

Voraussetzung

- Es befinden sich keine Personen oder Gegenstände im Bewegungsbereich des Roboters.
- Jeder Roboterachse steht in der Startposition ein Bewegungsbereich von ±10° zur Verfügung. (Oder, wenn keine Startposition geteacht wurde, in der Istposition.)
- Im Programm BrakeTestPark.SRC wurde die Parkposition geteacht.
- Benutzergruppe Experte
- Programm-Ablaufart GO
- Betriebsart AUT
- Der Roboter hat Betriebstemperatur (= nach ca. 1 h im normalen Betrieb).

Vorgehensweise

1. Im Verzeichnis R1\TP\BrakeTest das Programm BrakeTestReq.SRC anwählen und die Start-Taste drücken.
2. Folgende Meldung wird angezeigt: **Bremsentest wird händisch durchgeführt - bitte bestätigen**. Die Meldung quittieren.
3. Die Start-Taste drücken. Die Meldung *SAK erreicht* wird angezeigt.
4. Die Start-Taste drücken. Die Bremsen werden getestet, beginnend bei A1.
5. Mögliche Ergebnisse:

- Wenn eine Bremse in Ordnung ist, wird dies durch folgende Meldung angezeigt: *Wirkung der Bremse{Bremse Nr.}{Achsnr.} in Ordnung*.

  Wenn alle Bremsen in Ordnung sind, wird dies nach dem Bremsentest durch folgende Meldung angezeigt: *Bremsentest erfolgreich durchgeführt*. (Es ist möglich, dass eine oder mehrere Bremsen die Verschleißgrenze erreicht haben. Dies wird zusätzlich durch eine Meldung angezeigt.)

  Programm BrakeTestReq.SRC abwählen.
- Wenn eine Bremse defekt ist, wird dies durch folgende Meldung angezeigt: *Unzureichendes Haltemoment der Bremse {Bremse Nr.}{Achsnr.}*.

  Wenn alle Bremsen getestet wurden, entweder **Wiederh.** drücken, um den Bremsentest zur Kontrolle zu wiederholen.

  Oder **Parkpos.** drücken, um den Roboter in die Parkposition zu fahren.

Wenn eine Bremse als defekt erkannt wird, sind die Antriebe nach Beginn des Bremsentests noch 2 h in Regelung (= Monitoring-Zeit). Danach schaltet die Robotersteuerung die Antriebe ab.

6.20.7 Bremsentest auf Funktion testen

Beschreibung

Man kann prüfen, ob der Bremsentest eine defekte Bremse korrekt erkennt: Das Programm BrakeTestSelfTest.SRC simuliert einen Fehler in den Bremsen und löst einen Bremsentest aus. Wenn der Bremsentest den simulierten Fehler feststellt, arbeitet er korrekt.

⚠ WARNUNG Der Programm-Override wird beim Test automatisch auf 100 % gesetzt. Der Roboter verfährt mit hoher Geschwindigkeit. Darauf achten, dass der Roboter nicht kollidieren kann und dass sich keine Personen im Bewegungsbereich des Roboters befinden.

Voraussetzung

- Es befinden sich keine Personen oder Gegenstände im Bewegungsbereich des Roboters.
- Jeder Roboterachse steht in der Startposition ein Bewegungsbereich von ±10° zur Verfügung. (Oder, wenn keine Startposition geteacht wurde, in der Istposition.)
- Im Programm BrakeTestPark.SRC wurde die Parkposition geteacht.
- Benutzergruppe Experte
- Programm-Ablaufart GO
- Betriebsart AUT
- Der Roboter hat Betriebstemperatur (= nach ca. 1 h im normalen Betrieb).

Vorgehensweise

1. Im Verzeichnis R1\TP\BrakeTest das Programm BrakeTestSelfTest.SRC anwählen und die Start-Taste drücken.
2. Folgende Meldung wird angezeigt: *Selbsttest für Bremsentest wird durchgeführt - bitte bestätigen*. Die Meldung mit **Quitt** quittieren.
3. Die Start-Taste drücken.
4. Ergebnis des Funktionstests:
   - Meldung *Unzureichendes Haltemoment der Bremse 3*: Der Bremsentest hat den simulierten Fehler korrekt festgestellt. Der Bremsentest arbeitet korrekt.

     Das Programm BrakeTestSelfTest.SRC abwählen.

     Einen Bremsentest manuell durchführen. Dies sorgt dafür, dass der simulierte Fehler nicht stehen bleibt.

- Jede andere Meldung oder keine Meldung bedeutet: Der Bremsentest hat den simulierten Fehler nicht festgestellt. Der Bremsentest arbeitet nicht korrekt.

⚠ WARNUNG Wenn der Funktionstest ergibt, dass der Bremsentest nicht korrekt arbeitet:

- Der Roboter darf nicht mehr verfahren werden.
- Es muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

7 Programm- und Projektverwaltung

7.1 Neues Programm anlegen

Beschreibung In der Benutzergruppe Anwender kann kein Template ausgewählt werden. Es wird defaultmäßig ein Programm vom Typ "Modul" angelegt.

Voraussetzung
- Der Navigator wird angezeigt.

Vorgehensweise
1. In der Verzeichnisstruktur den Ordner markieren, in dem das Programm angelegt werden soll, z. Bsp. den Ordner **Program**. (Nicht in allen Ordnern können Programme angelegt werden.)
2. **Neu** drücken.
3. Nur in Benutzergruppe Experte:
   Das Fenster **Template Auswahl** öffnet sich. Das gewünschte Template markieren und mit **OK** bestätigen.
4. Einen Namen für das Programm eingeben und mit **OK** bestätigen.

7.2 Neuen Ordner anlegen

Voraussetzung
- Der Navigator wird angezeigt.

Vorgehensweise
1. In der Verzeichnisstruktur den Ordner markieren, in dem der neue Ordner angelegt werden soll, z. Bsp. den Ordner **R1**.
   Nicht in allen Ordnern können neue Ordner angelegt werden. In den Benutzergruppen Anwender und Bediener können ausschließlich im Ordner **R1** neue Ordner angelegt werden.
2. **Neu** drücken.
3. Einen Namen für den Ordner eingeben und mit **OK** bestätigen.

7.3 Datei oder Ordner umbenennen

Voraussetzung
- Der Navigator wird angezeigt.

Vorgehensweise
1. In der Verzeichnisstruktur den Ordner markieren, in dem sich die Datei oder der umzubenennende Ordner befindet.
2. In der Dateiliste die Datei oder den Ordner markieren.
3. **Bearbeiten** > **Umbenennen** wählen.
4. Den Namen mit dem neuen Namen überschreiben und mit **OK** bestätigen.

7.4 Dateimanager Navigator

Übersicht

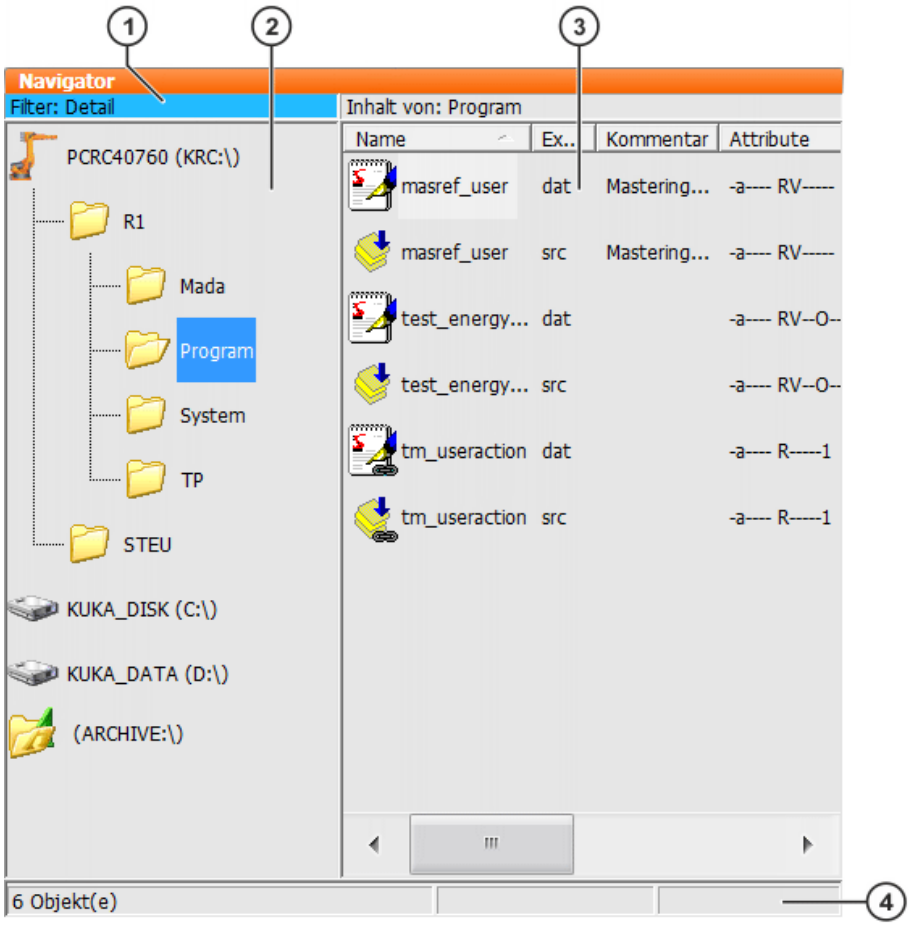


Abb. 7-1: Navigator

1	Kopfzeile	3	Dateiliste
2	Verzeichnisstruktur	4	Statuszeile

Beschreibung

Im Navigator verwaltet der Benutzer Programme und systemspezifische Dateien.

Kopfzeile

- Linker Bereich: Der gewählte Filter wird angezeigt.
  (>>> 7.4.1 "Filter auswählen" Seite 241)
- Rechter Bereich: Das Verzeichnis oder Laufwerk, das in der Verzeichnisstruktur markiert ist, wird angezeigt.

Verzeichnisstruktur

Übersicht über Verzeichnisse und Laufwerke. Welche Verzeichnisse und Laufwerke angezeigt werden, ist abhängig von der Benutzergruppe und von der Konfiguration.

Dateiliste

Der Inhalt des Verzeichnisses oder Laufwerks, das in der Verzeichnisstruktur markiert ist, wird angezeigt. In welcher Form Programme angezeigt werden, ist abhängig vom gewählten Filter.

Die Dateiliste hat folgende Spalten:

Spalte	Beschreibung
Name	Verzeichnis- oder Dateiname
Extension	Dateierweiterung Diese Spalte wird in der Benutzergruppe Anwender nicht angezeigt.
Kommentar	Kommentar
Attribute	Betriebssytem- und Grundsystem-Attribute Diese Spalte wird in der Benutzergruppe Anwender nicht angezeigt.
Größe	Dateigröße in KBytes Diese Spalte wird in der Benutzergruppe Anwender nicht angezeigt.
#	Anzahl der Änderungen an der Datei
Geändert	Datum und Uhrzeit der letzten Änderung
Erstellt	Datum und Uhrzeit der Erstellung Diese Spalte wird in der Benutzergruppe Anwender nicht angezeigt.

Statuszeile

Die Statuszeile kann folgende Informationen anzeigen:

- Markierte Objekte
- Laufende Aktionen
- Benutzer-Dialoge
- Aufforderungen für Benutzer-Eingaben
- Sicherheitsabfragen

7.4.1 Filter auswählen

Beschreibung

Diese Funktion steht nicht in der Benutzergruppe "Anwender" zur Verfügung.

Der Filter legt fest, wie Programme in der Dateiliste angezeigt werden. Folgende Filter stehen zur Auswahl:

- **Detail**
  Programme werden als SRC- und DAT-Dateien angezeigt. (Defaulteinstellung)
- **Module**
  Programme werden als Module angezeigt.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Menüfolge **Bearbeiten** > **Filter** wählen.
2. Im linken Bereich des Navigators den gewünschten Filter markieren.
3. Mit **OK** bestätigen.

7.4.2 Eigenschaften von Dateien und Ordnern anzeigen oder ändern

Voraussetzung

- Um Eigenschaften zu ändern: Benutzergruppe Experte

Vorgehensweise

1. Objekt in der Verzeichnisstruktur oder in der Dateiliste markieren.
2. Menüfolge **Bearbeiten** > **Eigenschaften** wählen.
   Ein Fenster öffnet sich. Das Fenster kann unterschiedlich viele Registerkarten haben, je nachdem, welches Objekt markiert wurde.

3. Bei Bedarf: Die Eigenschaften ändern und die Änderungen mit **OK** übernehmen.

Allgemein

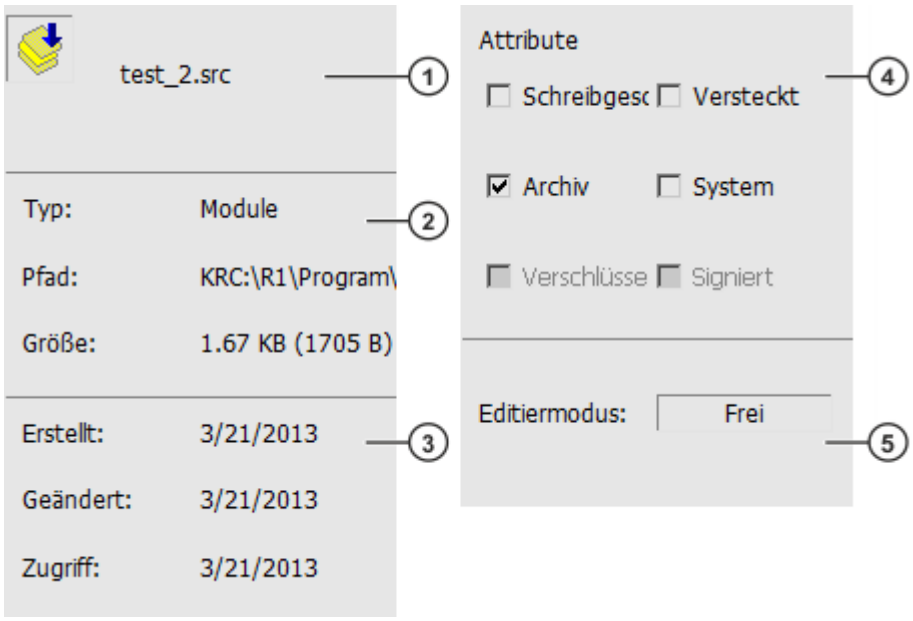


Abb. 7-2: Registerkarte Allgemein

Pos.	Beschreibung
1	Name des markierten Objekts
2	Objektart, Pfad und Größe. Objektarten: ■ **Module**: Modul ■ **Dir**: Ordner ■ **Archiv**: Archivdatei ■ **Bin**: Binärdatei ■ **Text**: Textdatei ■ **VirtualDir**: Virtueller Ordner ■ **Unknown**: Alle anderen Dateitypen
3	Windows-Objekteigenschaften
4	Windows-Objekteigenschaften. Die Eigenschaften können in der Benutzergruppe Experte geändert werden.
5	**Frei**: Die Datei ist auf der smart.HMI nicht angewählt und nicht geöffnet. **Full**: Die Datei ist auf der smart.HMI geöffnet. **ProKor**: Die Datei ist auf der smart.HMI angewählt.

Modul Info

Die Registerkarte **Modul Info** wird nur angezeigt, wenn das markierte Objekt eine Datei ist.

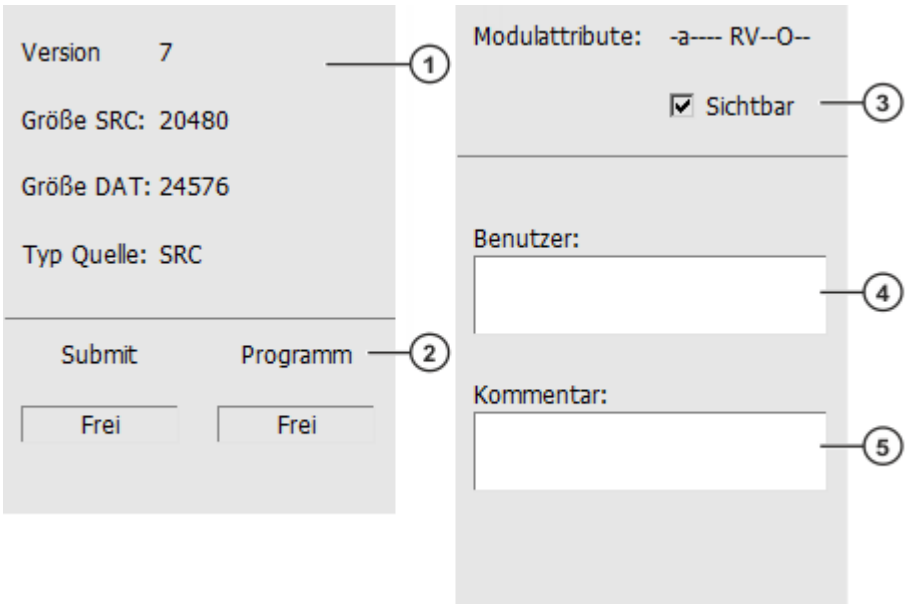


Abb. 7-3: Registerkarte Modul Info

Pos.	Beschreibung
1	**Version**: Interne Versionsnummer der Datei. Nach dem Anlegen hat die Datei noch keine Nummer. Nach der ersten Änderung erhält die Datei die Nummer 1. Wird nach jeder Änderung hochgezählt. **Größe SRC:**: Größe der SRC-Datei **Größe DAT:**: Größe der DAT-Datei **Typ Quelle:**: Dateityp ■ **SRC**: SRC-Datei ■ **SubmitSub**: SUB-Datei ■ **None**: Alle anderen Dateitypen, z. B. DAT-Datei
2	Zustand des Moduls im Submit-Interpreter und im Roboter-Interpreter **Frei**: Programm ist nicht angewählt. **angewählt**: Programm ist angewählt. **Active**: Nur relevant für das Feld **Submit**. Dieses Programm wird momentan vom Submit-Interpreter verwendet.
3	Checkbox aktiv: Wenn dieses Programm als Unterprogramm aufgerufen wird, wird es im Editor angezeigt. Checkbox inaktiv: Wenn dieses Programm als Unterprogramm aufgerufen wird, wird es im Editor nicht angezeigt. Dieses Programm kann nicht manuell angewählt werden.
4	Hier kann der Benutzer seinen Namen eintragen.
5	Hier kann der Benutzer einen Kommentar zum Modul eintragen. Der Kommentar wird im Navigator in der Spalte **Kommentar** angezeigt.

Parameter

Die Registerkarte **Parameter** wird nur angezeigt, wenn das markierte Objekt eine Datei ist.

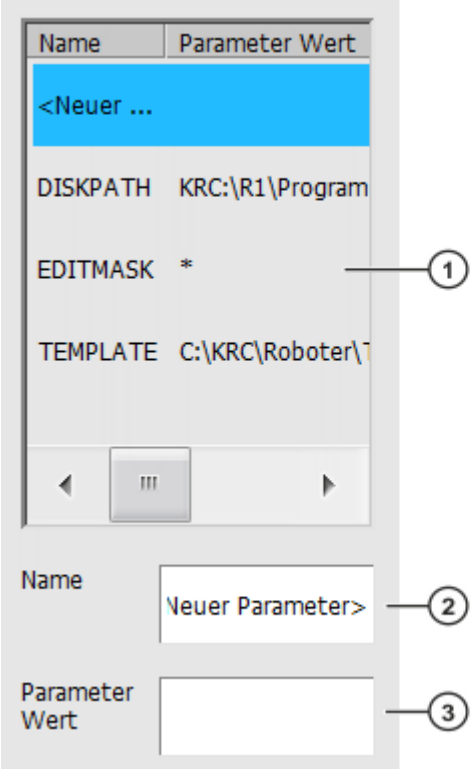


Abb. 7-4: Registerkarte Parameter

In KRL-Modulen können beliebige Informationen hinterlegt werden.

Pos.	Beschreibung
1	Hier werden die bestehenden Informationen angezeigt. Eine Information kann gelöscht werden, indem man die Zeile markiert und im Feld **Parameter Wert** den Inhalt löscht. Dann mit **OK** bestätigen.
2	Hier kann der Benutzer einen Namen für eine neue Information eintragen.
3	Hier kann der Benutzer eine Information eintragen.

Programm im Editor

Wenn eine SRC- oder DAT-Datei auf der Windows-Ebene in einem Editor (z. B. WordPad) geöffnet ist, werden oberhalb der DEF-Zeile einige der Dateieigenschaften angezeigt.

```
1  &ACCESS RV
2  &REL 2
3  &COMMENT test comment
4  &USER kuka
5  &PARAM test name = test param
6  &PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
7  &PARAM EDITMASK = *
8  DEF test( )
   ...
```

Zeile	Beschreibung
1	Registerkarte **Modul Info**, Checkbox **Sichtbar** ■ `&ACCESS RV` = Checkbox aktiv ■ `&ACCESS R` = Checkbox inaktiv
2	Registerkarte **Modul Info**, Feld **Version**
3	Registerkarte **Modul Info**, Feld **Kommentar**

Zeile	Beschreibung
4	Registerkarte **Modul Info**, Feld **Benutzer**
5	Registerkarte **Parameter**, Felder **Name** und **Parameter Wert**

7.5 Programm anwählen oder öffnen

Übersicht

Ein Programm kann angewählt oder geöffnet werden. Statt dem Navigator wird dann ein Editor mit dem Programm angezeigt.

(>>> 7.5.1 "Programm anwählen und abwählen" Seite 245)

(>>> 7.5.2 "Programm öffnen" Seite 246)

Zwischen der Programmanzeige und dem Navigator kann man hin- und herwechseln.

(>>> 7.5.3 "Zwischen Navigator und Programm wechseln" Seite 247)

Unterschiede

Programm ist angewählt:

- Der Satzzeiger wird angezeigt.
- Das Programm kann gestartet werden.
- Das Programm kann eingeschränkt bearbeitet werden.

  Angewählte Programme sind besonders für die Bearbeitung durch die Benutzergruppe Anwender geeignet.

  Beispiel: KRL-Anweisungen, die sich über mehrere Zeilen erstrecken (z. B. LOOP … ENDLOOP) sind nicht zulässig.
- Änderungen werden beim Abwählen ohne Sicherheitsabfrage übernommen. Wenn nicht zulässige Änderungen programmiert wurden, wird eine Fehlermeldung angezeigt.

Programm ist geöffnet:

- Das Programm kann nicht gestartet werden.
- Das Programm kann bearbeitet werden.

  Geöffnete Programme sind besonders für die Bearbeitung durch die Benutzergruppe Experte geeignet.
- Beim Schließen kommt eine Sicherheitsabfrage. Änderungen können übernommen oder verworfen werden.

7.5.1 Programm anwählen und abwählen

> Wenn ein angewähltes Programm in der Benutzergruppe Experte bearbeitet wird, muss danach der Cursor aus der bearbeiteten Zeile genommen werden und in eine beliebige andere Zeile gesetzt werden! Nur so ist gewährleistet, dass die Bearbeitung übernommen wird, wenn das Programm wieder abgewählt wird.

Voraussetzung

- Betriebsart T1, T2 oder AUT

Vorgehensweise

1. Programm im Navigator markieren und **Anwählen** drücken.

   Das Programm wird im Editor angezeigt. Es ist gleichgültig, ob ein Modul, eine SRC-Datei oder eine DAT-Datei markiert wurden. Im Editor wird immer die SRC-Datei angezeigt.
2. Programm starten oder bearbeiten.
3. Programm wieder abwählen:

   Bearbeiten > **Programm abwählen** wählen.

Oder: In der Statusleiste die Statusanzeige **Roboter-Interpreter** berühren. Ein Fenster öffnet sich. **Programm abwählen** wählen.

Änderungen werden beim Abwählen ohne Sicherheitsabfrage übernommen!

Wenn das Programm läuft, muss es gestoppt werden, bevor man es wieder abwählen kann.

Beschreibung

Wenn ein Programm angewählt ist, wird dies von der Statusanzeige **Roboter-Interpreter** angezeigt.

(>>> 8.6 "Statusanzeige Roboter-Interpreter" Seite 277)

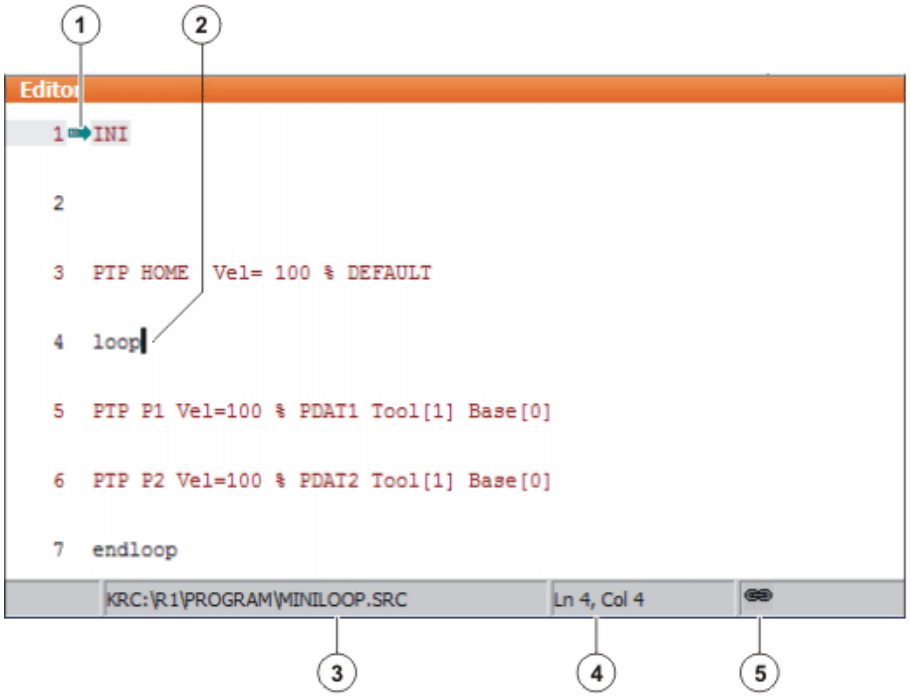


Abb. 7-5: Programm ist angewählt

1 Satzzeiger
2 Cursor
3 Pfad des Programms und Dateiname
4 Position des Cursors im Programm
5 Das Symbol zeigt, dass das Programm angewählt ist.

7.5.2 Programm öffnen

Voraussetzung

- Betriebsart T1, T2 oder AUT

In der Betriebsart AUT EXT kann man ein Programm zwar öffnen, aber nicht bearbeiten.

Vorgehensweise

1. Programm im Navigator markieren und **Öffnen** drücken. Das Programm wird im Editor angezeigt.
   Wenn ein Modul markiert wurde, wird im Editor die SRC-Datei angezeigt. Wenn eine SRC- oder DAT-Datei markiert wurde, wird im Editor die jeweilige Datei angezeigt.
2. Programm bearbeiten.
3. Programm schließen.

4. Um die Änderungen zu übernehmen, die Sicherheitsabfrage mit **Ja** beantworten.

Beschreibung

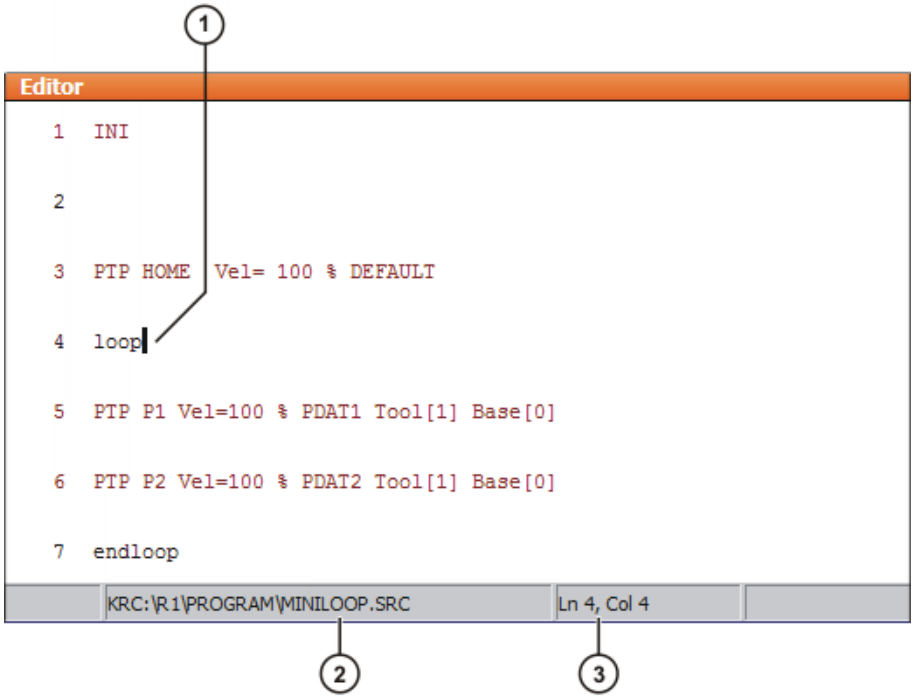


Abb. 7-6: Programm ist geöffnet

1 Cursor
2 Pfad des Programms und Dateiname
3 Position des Cursors im Programm

7.5.3 Zwischen Navigator und Programm wechseln

Beschreibung

Wenn ein Programm angewählt oder geöffnet ist, kann man den Navigator wieder anzeigen, ohne das Programm abwählen oder schließen zu müssen. Danach kann man wieder zum Programm zurückkehren.

Vorgehensweise

Programm ist angewählt:

- Vom Programm zum Navigator wechseln: Menüfolge **Bearbeiten** > **Navigator** wählen.
- Vom Navigator zum Programm wechseln: **PROGRAM** drücken.

Programm ist geöffnet:

- Vom Programm zum Navigator wechseln: Menüfolge **Bearbeiten** > **Navigator** wählen.
- Vom Navigator zum Programm wechseln: **EDITOR** drücken.

Laufende oder angehaltene Programme müssen erst gestoppt werden, damit die genannten Menüfolgen und Schaltflächen zur Verfügung stehen.

7.6 Aufbau eines KRL-Programms

```
1  DEF my_program( )
2  INI
3
4  PTP HOME  Vel= 100 % DEFAULT
```

```
   ...
8  LIN point_5 CONT Vel= 2 m/s CPDAT1 Tool[3] Base[4]
   ...
14 PTP point_1 CONT Vel= 100 % PDAT1 Tool[3] Base[4]
   ...
20 PTP HOME  Vel= 100 % DEFAULT
21
22 END
```

Zeile	Beschreibung
1	Die DEF-Zeile zeigt den Namen des Programms an. Wenn das Programm eine Funktion ist, beginnt die DEF-Zeile mit "DEFFCT" und enthält noch weitere Angaben. Die DEF-Zeile kann ein- oder ausgeblendet werden. (>>> 7.7.1 "DEF-Zeile ein-/ausblenden" Seite 249)
2	Die INI-Zeile enthält Initialisierungen für interne Variablen und Parameter.
4	HOME-Position (>>> 7.6.1 "HOME-Position" Seite 248)
8	LIN-Bewegung (>>> 10.2.3 "LIN-Bewegung programmieren" Seite 318)
14	PTP-Bewegung (>>> 10.2.1 "PTP-Bewegung programmieren" Seite 317)
20	HOME-Position
22	Die END-Zeile ist die letzte Zeile in jedem Programm. Wenn das Programm eine Funktion ist, lautet die END-Zeile "END-FCT". Die END-Zeile darf nicht gelöscht werden!

Die erste Bewegungsanweisung in einem KRL-Programm muss eine eindeutige Ausgangslage definieren. Bei der HOME-Position, die defaultmäßig in der Robotersteuerung angelegt ist, ist dies gewährleistet.

Wenn die erste Bewegungsanweisung nicht die Default-HOME-Position ist oder diese geändert wurde, muss eine der folgenden Anweisungen verwendet werden:

- Vollständige PTP-Anweisung vom Typ POS oder E6POS
- Vollständige PTP-Anweisung vom Typ AXIS oder E6AXIS

"Vollständig" bedeutet, dass alle Komponenten des Zielpunkts angegeben werden müssen.

⚠ WARNUNG Wenn die HOME-Position geändert wird, wirkt sich das auf alle Programme aus, in denen sie verwendet wird. Verletzungen und Sachschäden können die Folge sein.

In Programmen, die ausschließlich als Unterprogramme verwendet werden, können auch andere Anweisungen als erste Bewegungsanweisung verwendet werden.

7.6.1 HOME-Position

Die HOME-Position ist eine programmübergreifend gültige Position. Sie wird in der Regel als erste und letzte Position im Programm verwendet, weil sie eindeutig definiert und unkritisch ist.

Die HOME-Position ist defaultmäßig mit folgenden Werten in der Robotersteuerung angelegt:

Achse	A1	A2	A3	A4	A5	A6
Pos.	0°	- 90°	+ 90°	0°	0°	0°

Es können weitere HOME-Positionen geteacht werden. Eine HOME-Position muss folgende Bedingungen erfüllen:

- Günstige Ausgangsposition für den Programmlauf
- Günstige Stillstandsposition. Beispielsweise darf der Roboter im Stillstand kein Hindernis darstellen.

⚠ WARNUNG Wenn die HOME-Position geändert wird, wirkt sich das auf alle Programme aus, in denen sie verwendet wird. Verletzungen und Sachschäden können die Folge sein.

7.7 Programmteile ein-/ausblenden

7.7.1 DEF-Zeile ein-/ausblenden

Beschreibung

Die DEF-Zeile ist defaultmäßig ausgeblendet. In einem Programm können nur Deklarationen vorgenommen werden, wenn die DEF-Zeile eingeblendet ist.

Die DEF-Zeile wird für geöffnete und angewählte Programme getrennt ein- und ausgeblendet. Wenn die Detailansicht eingeschaltet ist, ist die DEF-Zeile sichtbar und braucht nicht gesondert eingeblendet werden.

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt oder geöffnet.

Vorgehensweise

1. Menüfolge **Bearbeiten** > **Ansicht** wählen. Der Unterpunkt **DEF-Zeile** zeigt den aktuellen Zustand an:
   - **Ohne Häkchen**: Die DEF-Zeile ist ausgeblendet.
   - **Mit Häkchen**: Die DEF-Zeile ist eingeblendet.

2. Um den Zustand zu ändern, den Menüpunkt **DEF-Zeile** berühren.
   Danach schließt sich das Menü automatisch.

7.7.2 Detailansicht anzeigen

Beschreibung

Die Detailansicht ist defaultmäßig ausgeschaltet, um das Programm übersichtlich zu halten. Wenn die Detailansicht eingeschaltet wird, werden verborgene Programmzeilen eingeblendet, beispielsweise FOLD- und ENDFOLD-Zeilen und die DEF-Zeile.

Die Detailansicht wird für geöffnete und angewählte Programme getrennt ein- und ausgeschaltet.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Menüfolge **Bearbeiten** > **Ansicht** wählen. Der Unterpunkt **Detailansicht (ASCII)** zeigt den aktuellen Zustand an:
   - **Ohne Häkchen**: Die Detailansicht ist ausgeschaltet.
   - **Mit Häkchen**: Die Detailansicht ist eingeschaltet.

2. Um den Zustand zu ändern, den Menüpunkt **Detailansicht (ASCII)** berühren.
   Danach schließt sich das Menü automatisch.

7.7.3 Zeilenumbruch ein-/ausschalten

Beschreibung

Wenn eine Zeile breiter ist als das Programmfenster, wird sie defaultmäßig umgebrochen. Der umgebrochene Teil der Zeile besitzt keine Zeilennummer und ist durch einen schwarzen, L-förmigen Pfeil markiert.

```
8  EXT  IBGN (IBGN_COMMAND  :IN,BOOL  :IN,REAL  :IN,REAL
  ↳ :IN,BOOL  :IN,E6POS  :OUT )
```

Abb. 7-7: Zeilenumbruch

Der Zeilenumbruch kann ausgeschaltet werden. Wenn eine Zeile breiter ist als das Programmfenster, ist sie nun nicht mehr auf einmal sichtbar. Unterhalb des Programmfensters blendet sich ein Scroll-Balken ein.

Der Zeilenumbruch wird für geöffnete und angewählte Programme getrennt ein- und ausgeschaltet.

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt oder geöffnet.

Vorgehensweise

1. Menüfolge **Bearbeiten** > **Ansicht** wählen. Der Unterpunkt **Zeilenumbruch** zeigt den aktuellen Zustand an:
   - **Ohne Häkchen**: Der Zeilenumbruch ist ausgeschaltet.
   - **Mit Häkchen**: Der Zeilenumbruch ist eingeschaltet.

     ✓ Zeilenumbruch
2. Um den Zustand zu ändern, den Menüpunkt **Zeilenumbruch** berühren.
   Danach schließt sich das Menü automatisch.

7.7.4 Folds anzeigen

Beschreibung

In Folds sind Programmteile verborgen. Folds machen Programme dadurch übersichtlicher. Die verborgenen Programmteile werden beim Programmlauf genauso abgearbeitet wie normale Programmteile.

- In der Benutzergruppe Anwender sind Folds immer geschlossen. D. h., die Inhalte der Folds sind nicht sichtbar und können nicht bearbeitet werden.
- In der Benutzergruppe Experte sind Folds defaultmäßig geschlossen. Sie können geöffnet und bearbeitet werden. Neue Folds können angelegt werden.
  (>>> 7.8.3 "Folds anlegen" Seite 253)

Wenn ein Programm abgewählt oder geschlossen wird, werden alle Folds automatisch geschlossen.

```
2
3  PTP HOME  Vel= 100 % DEFAULT
4
```

Abb. 7-8: Beispiel geschlossener Fold

```
2
3  PTP HOME  Vel= 100 % DEFAULT
4  $BWDSTART = FALSE
5  PDAT_ACT=PDEFAULT
6  FDAT_ACT=FHOME
7  BAS (#PTP_PARAMS,100 )
8  $H_POS=XHOME
9  PTP  XHOME
10
```

Abb. 7-9: Beispiel offener Fold

Farbkennzeichnung bei Folds:

Farbe	Beschreibung
Dunkelrot	Geschlossener Fold
Hellrot	Geöffneter Fold
Dunkelblau	Geschlossener Unterfold
Hellblau	Geöffneter Unterfold
Grün	Inhalt des Folds

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt oder geöffnet.

Vorgehensweise

1. Die Zeile markieren, die den Fold enthält.
2. **Fold öffn/schl** drücken. Der Fold öffnet sich.
3. Um den Fold zu schließen, **Fold öffn/schl** erneut drücken.

Alternativ können über die Menüfolge **Bearbeiten** > **FOLD** > **Alle FOLDs öffnen** oder **Alle FOLDs schließen** alle Folds eines Programms auf einmal geöffnet oder geschlossen werden.

7.8 Programme bearbeiten

Übersicht

- Ein laufendes Programm kann nicht bearbeitet werden.
- In der Betriebsart AUT EXT können Programme nicht bearbeitet werden.

Wenn ein angewähltes Programm in der Benutzergruppe Experte bearbeitet wird, muss danach der Cursor aus der bearbeiteten Zeile genommen werden und in eine beliebige andere Zeile gesetzt werden! Nur so ist gewährleistet, dass die Bearbeitung übernommen wird, wenn das Programm wieder abgewählt wird.

Aktion	Möglich in Benutzergruppe …?
Kommentar oder Stempel einfügen	**Anwender**: Ja **Experte**: Ja
Zeilen löschen	**Anwender**: Ja **Experte**: Ja
Folds anlegen	**Anwender**: Nein **Experte**: Ja
Kopieren	**Anwender**: Nein **Experte**: Ja
Einfügen	**Anwender**: Nein **Experte**: Ja
Leerzeilen einfügen (Eingabe-Taste drücken)	**Anwender**: Nein **Experte**: Ja
Ausschneiden	**Anwender**: Nein **Experte**: Ja
Suchen	**Anwender**: Ja **Experte**: Ja Für alle Benutzergruppen bei geöffnetem Programm auch in der Betriebsart AUT EXT möglich.
Ersetzen	**Anwender**: Nein **Experte**: Ja (Programm ist geöffnet, nicht angewählt)
Mit Inline-Formularen programmieren	**Anwender**: Ja **Experte**: Ja
KRL programmieren	**Anwender**: Eingeschränkt möglich. KRL-Anweisungen, die sich über mehrere Zeilen erstrecken (z. B. LOOP … ENDLOOP) sind nicht zulässig. **Experte**: Ja

7.8.1 Kommentar oder Stempel einfügen

Voraussetzung

- Programm ist angewählt oder geöffnet.
- Betriebsart T1

Vorgehensweise

1. Die Zeile markieren, nach der der Kommentar oder der Stempel eingefügt werden soll.
2. Menüfolge **Befehle** > **Kommentar** > **Normal** oder **Stempel** wählen.
3. Gewünschte Daten eingeben. Wenn bereits vorher ein Kommentar oder Stempel eingefügt worden sind, enthält das Inline-Formular noch die gleichen Angaben.
   - Beim Kommentar kann mit **Text NEU** das Feld geleert werden, um einen neuen Text einzugeben.

- Beim Stempel kann außerdem mit **Zeit NEU** die Systemzeit aktualisiert werden und mit **Name NEU** das Feld **NAME** geleert werden.

4. Mit **Befehl OK** speichern.

Beschreibung Kommentar

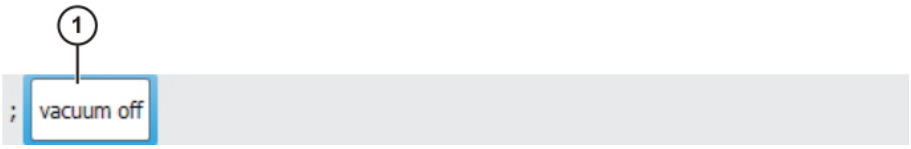


Abb. 7-10: Inline-Formular Kommentar

Pos.	Beschreibung
1	Beliebiger Text

Beschreibung Stempel

Ein Stempel ist ein Kommentar, der um Systemdatum, -zeit und Benutzerkennung erweitert ist.

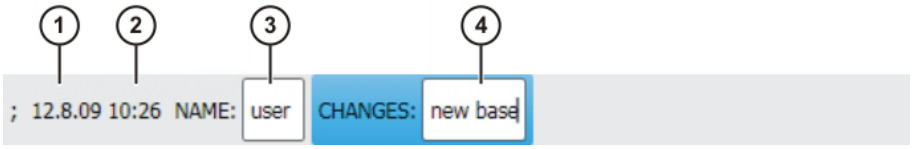


Abb. 7-11: Inline-Formular Stempel

Pos.	Beschreibung
1	Systemdatum (nicht editierbar)
2	Systemzeit
3	Name oder Kennung des Benutzers
4	Beliebiger Text

7.8.2 Programmzeilen löschen

Gelöschte Zeilen können nicht wiederhergestellt werden!

Beschreibung

Wenn eine Programmzeile gelöscht wird, die eine Bewegungsanweisung enthält, so bleiben Punktname und -koordinaten in der DAT-Datei gespeichert. Der Punkt kann in anderen Bewegungsanweisungen verwendet werden und muss nicht noch einmal geteacht werden.

Voraussetzung

- Programm ist angewählt oder geöffnet.
- Betriebsart T1

Vorgehensweise

1. Die Zeile markieren, die gelöscht werden soll. (Die Zeile muss nicht farbig hinterlegt sein. Es reicht, wenn der Cursor in der Zeile steht.)
   Wenn mehrere aufeinanderfolgende Zeilen gelöscht werden sollen: Den Finger oder Zeigestift über den gewünschten Bereich ziehen. (Der Bereich muss dann farbig hinterlegt sein.)
2. Menüfolge **Bearbeiten** > **Löschen** wählen.
3. Sicherheitsabfrage mit **Ja** bestätigen.

7.8.3 Folds anlegen

Syntax

```
;FOLD Name
Anweisungen
```

;ENDFOLD <*Name*>

Die ENDFOLD-Zeilen können leichter zugeordnet werden, wenn auch hier der Name des Folds eingetragen wird. Folds können geschachtelt werden.

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt oder geöffnet.
- Betriebsart T1

Vorgehensweise

1. Fold ins Programm eintragen. Ein doppeltes Semikolon verhindert, dass der Fold sich beim Bearbeiten schließt.

```
4
5  ;;FOLD outputs
6  $OUT[1] = TRUE
7  $OUT[2] = TRUE
8  ;;ENDFOLD outputs
9
```

Abb. 7-12: Beispiel Fold anlegen, Schritt 1

2. Das zweite Semikolon entfernen.

```
4
5  ;FOLD outputs
6  $OUT[1] = TRUE
7  $OUT[2] = TRUE
8  ;ENDFOLD outputs
9
```

Abb. 7-13: Beispiel Fold anlegen, Schritt 2

3. Den Cursor in eine Zeile außerhalb des Folds setzen. Der Fold schließt sich.

```
4
5  outputs
6
```

Abb. 7-14: Beispiel Fold anlegen, Schritt 3

7.8.4 Weitere Bearbeitungsfunktionen

Folgende weitere Funktionen zum Bearbeiten von Programmen können über **Bearbeiten** aufgerufen werden:

Kopieren

Voraussetzung:

- Programm ist angewählt oder geöffnet.
- Benutzergruppe Experte
- Betriebsart T1

Einfügen

Voraussetzung:

- Programm ist angewählt oder geöffnet.
- Benutzergruppe Experte

- Betriebsart T1

Ausschneiden

Voraussetzung:

- Programm ist angewählt oder geöffnet.
- Benutzergruppe Experte
- Betriebsart T1

Suchen

Voraussetzung:

- Programm ist angewählt oder geöffnet.

Ersetzen

Voraussetzung:

- Programm ist geöffnet.
- Benutzergruppe Experte
- Betriebsart T1

Markierter Bereich

(>>> 10.5.4 "Koordinaten blockweise verschieben" Seite 351)

7.9 Programm drucken

Vorgehensweise

1. Das Programm im Navigator markieren. Es können auch mehrere Programme markiert werden.
2. Menüfolge **Bearbeiten** > **Drucken** wählen.

7.10 Daten archivieren und wiederherstellen

7.10.1 Übersicht Archivierung

Zielorte

Auf folgende Zielorte kann archiviert werden:

- USB-Stick am smartPAD oder an der Robotersteuerung
- Netzwerk

Menüpunkte

Folgende Menüpunkte stehen zur Auswahl:

("*.*" bedeutet: Alle Dateien und Unterverzeichnisse.)

Menüpunkt	Archiviert die Verzeichnisse/Dateien
Alles	■ KRC:*.* ■ C:\KRC\Roboter\Config\User*.* ■ C:\KRC\Roboter\Config\System\Common\Mada*.* ■ C:\KRC\Roboter\Init*.* ■ C:\KRC\Roboter\Ir_Spec*.* ■ C:\KRC\Roboter\Template*.* ■ C:\KRC\Roboter\Rdc*.* ■ C:\KRC\User*.* ■ C:\KRC\Roboter\log\Mastery.log ■ Einige weitere Log-Daten
Anwendungen	■ KRC:\R1\Program*.* ■ KRC:\R1\System*.* ■ KRC:\R1\cell*.* ■ KRC:\Steu\$config*.*
Systemdaten	■ KRC:\R1\Mada*.* ■ KRC:\R1\System*.* ■ KRC:\R1\TP*.* ■ KRC:\Steu\Mada*.* ■ C:\KRC\Roboter\Config\User*.* ■ C:\KRC\Roboter\Config\System\Common\Mada*.* ■ C:\KRC\Roboter\Init*.* ■ C:\KRC\Roboter\Ir_Spec*.* ■ C:\KRC\Roboter\Template*.* ■ C:\KRC\Roboter\Rdc*.* ■ C:\KRC\User*.*
Log Daten	■ C:\KRC\Roboter\log*.* Außer: Poslog.xsl sowie Dateien mit der Endung DMP ■ Einige weitere Log-Daten
KrcDiag	Wenn ein Fehler von der KUKA Roboter GmbH analysiert werden muss, kann man über diesen Menüpunkt die benötigten Daten verpacken, um sie KUKA zukommen zu lassen. Für das Datenpaket wird automatisch ein Screenshot von der aktuellen Ansicht der smartHMI erzeugt. Deshalb vor dem Starten des Vorgangs auf der smartHMI fehlerrelevante Informationen einblenden: Zum Bsp. das Meldungsfenster expandieren oder das Logbuch anzeigen. Welche Informationen sinnvoll sind, ist vom abhängig vom Einzelfall. Außer über **Datei** > **Archivieren** können diese Daten auch mit anderen Vorgehensweisen verpackt werden. (>>> 13.5 "Daten automatisch verpacken für Fehleranalyse (KrcDiag)" Seite 482)

Wenn über den Menüpunkt **Alles** archiviert wird und bereits ein Archiv vorhanden ist, wird dieses überschrieben.

Wenn über einen anderen Menüpunkt als **Alles** oder **KrcDiag** archiviert wird und bereits ein Archiv vorhanden ist, vergleicht die Robotersteuerung ihren

Robotername mit dem des Archivs. Wenn die Namen unterschiedlich sind, kommt eine Sicherheitsabfrage.

Wenn mehrmals über **KrcDiag** archiviert wird, können maximal 10 Archive erstellt werden. Weitere Archive überschreiben das älteste vorhandene Archiv.

Außerdem kann das Logbuch archiviert werden. (>>> 7.10.4 "Logbuch archivieren" Seite 258)

7.10.2 Archivieren auf USB-Stick

Beschreibung

Diese Vorgehensweise erzeugt auf dem Stick eine ZIP-Datei. Diese hat defaultmäßig den gleichen Namen wie der Roboter. Unter **Roboterdaten** kann jedoch auch ein eigener Name für die Datei festgelegt werden.

(>>> 4.17.15 "Roboterdaten anzeigen/bearbeiten" Seite 94)

Das Archiv wird im Navigator im Verzeichnis ARCHIVE:\ angezeigt. Zusätzlich zum Stick wird automatisch auch auf D:\ archiviert. Hier wird die Datei INTERN.ZIP erzeugt.

Sonderfall **KrcDiag**:

Dieser Menüpunkt erzeugt auf dem Stick den Ordner **KRCDiag**. Dieser enthält eine ZIP-Datei. Zusätzlich wird die ZIP-Datei automatisch auch auf C:\KUKA\KRCDiag archiviert.

HINWEIS Es muss ein nicht-bootfähiger USB-Stick verwendet werden.
Es wird empfohlen, einen nicht-bootfähigen KUKA-Stick zu verwenden. Wenn ein Stick eines anderen Herstellers verwendet wird, können Daten verloren gehen.

Vorgehensweise

1. Einen USB-Stick anstecken (an smartPAD oder Schrank).
2. Im Hauptmenü **Datei** > **Archivieren** > **USB (KCP)** oder **USB (Schrank)** wählen und dann den gewünschten Unterpunkt wählen.
3. Die Sicherheitsabfrage mit **Ja** bestätigen. Das Archiv wird erstellt.
   Im Meldungsfenster wird angezeigt, wenn die Archivierung abgeschlossen ist.
   Sonderfall **KrcDiag**: Wenn über diesen Menüpunkt archiviert wird, wird in einem gesonderten Fenster angezeigt, wenn die Archivierung abgeschlossen ist. Das Fenster blendet sich danach selbständig wieder aus.
4. Der Stick kann jetzt abgezogen werden.

7.10.3 Archivieren auf Netzwerk

Beschreibung

Diese Vorgehensweise erzeugt auf dem Netzwerk-Pfad eine ZIP-Datei. Diese hat defaultmäßig den gleichen Namen wie der Roboter. Unter **Inbetriebnahme** > **Roboterdaten** kann jedoch auch ein eigener Name für die Datei festgelegt werden.

Der Netzwerk-Pfad, auf den archiviert werden soll, muss im Fenster **Roboterdaten** konfiguriert werden. Wenn ein Benutzername und ein Passwort erforderlich sind, damit auf diesen Pfad archiviert werden kann, können diese dort ebenfalls eingegeben werden.

(>>> 4.17.15 "Roboterdaten anzeigen/bearbeiten" Seite 94)

Das Archiv wird im Navigator im Verzeichnis ARCHIVE:\ angezeigt. Zusätzlich zum Netzwerk-Pfad wird automatisch auch auf D:\ archiviert. Hier wird die Datei INTERN.ZIP erzeugt.

Sonderfall **KrcDiag**:

Dieser Menüpunkt erzeugt auf dem Netzwerk-Pfad den Ordner **KRCDiag**. Dieser enthält eine ZIP-Datei. Zusätzlich wird die ZIP-Datei automatisch auch auf C:\KUKA\KRCDiag archiviert.

Voraussetzung

- Der Netzwerk-Pfad, auf den archiviert werden soll, ist konfiguriert.

Vorgehensweise

1. Im Hauptmenü **Datei** > **Archivieren** > **Netzwerk** wählen und dann den gewünschten Unterpunkt wählen.
2. Die Sicherheitsabfrage mit **Ja** bestätigen. Das Archiv wird erstellt.
   Im Meldungsfenster wird angezeigt, wenn die Archivierung abgeschlossen ist. Sonderfall **KrcDiag**: Wenn über diesen Menüpunkt archiviert wird, wird in einem gesonderten Fenster angezeigt, wenn die Archivierung abgeschlossen ist. Das Fenster blendet sich danach selbständig wieder aus.

7.10.4 Logbuch archivieren

Beschreibung

Als Archiv wird im Verzeichnis C:\KRC\ROBOTER\LOG die Datei Logbuch.txt erzeugt.

Vorgehensweise

- Im Hauptmenü **Datei** > **Archivieren** > **Logbuch** wählen.
  Das Archiv wird erstellt. Im Meldungsfenster wird angezeigt, wenn die Archivierung abgeschlossen ist.

7.10.5 Daten wiederherstellen

Beschreibung

⚠ WARNUNG In die KSS 8.3 dürfen ausschließlich Archive der KSS 8.3 geladen werden. Wenn andere Archive geladen werden, können als Folgen auftreten:

- Fehlermeldungen
- Robotersteuerung ist nicht lauffähig.
- Personen- und Sachschäden

Beim Wiederherstellen stehen folgende Menüpunkte zur Auswahl:

- **Alles**
- **Anwendungen**
- **Systemdaten**

Wenn die archivierten Dateien nicht die gleiche Version haben wie die im System vorhandenen Dateien, wird beim Wiederherstellen eine Fehlermeldung ausgegeben.

Wenn die Version der archivierten Technologiepakete nicht mit der installierten Version übereinstimmt, wird ebenfalls eine Fehlermeldung ausgegeben.

Voraussetzung

- Wenn von USB-Stick wiederhergestellt werden soll: Ein USB-Stick mit dem Archiv ist angeschlossen.
  Der Stick kann am smartPAD oder an der Robotersteuerung angeschlossen werden.

HINWEIS Es muss ein nicht-bootfähiger USB-Stick verwendet werden.
Es wird empfohlen, einen nicht-bootfähigen KUKA-Stick zu verwenden. Wenn ein Stick eines anderen Herstellers verwendet wird, können Daten verloren gehen.

Vorgehensweise

1. Im Hauptmenü **Datei** > **Wiederherstellen** wählen und dann die gewünschten Unterpunkte wählen.

2. Die Sicherheitsabfrage mit **Ja** bestätigen. Die archivierten Dateien werden auf der Robotersteuerung wiederhergestellt. Eine Meldung zeigt an, wenn die Wiederherstellung abgeschlossen ist.
3. Wenn von USB-Stick wiederhergestellt wurde: Der Stick kann jetzt abgezogen werden.
4. Die Robotersteuerung neu starten.

7.11 Projektverwaltung

7.11.1 Projekt auf der Robotersteuerung pinnen

Beschreibung

Projekte, die sich auf der Robotersteuerung befinden, können gepinnt werden. Ein Projekt kann direkt auf der Robotersteuerung gepinnt werden oder von WorkVisual aus.

Gepinnte Projekte können nicht verändert, aktiviert oder gelöscht werden. Sie können jedoch kopiert oder entpinnt werden. Man kann also ein Projekt pinnen, um z. B. zu verhindern, dass es versehentlich gelöscht wird.

Informationen darüber, wie man Projekte von WorkVisual aus pinnen kann, sind in der Dokumentation **WorkVisual** zu finden.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Auf der smartHMI das WorkVisual-Symbol berühren, dann auf **Öffnen** gehen. Das Fenster **Projektverwaltung** öffnet sich.
2. Das gewünschte Projekt markieren und auf die Schaltfläche **Pinnen** drücken. Das Projekt wird gepinnt und in der Projektliste durch ein Pin-Symbol gekennzeichnet.
   (>>> 7.11.3 "Fenster Projektverwaltung" Seite 260)
3. Mit der Schaltfläche **Entpinnen** kann das Projekt wieder entpinnt werden.

7.11.2 Projekt aktivieren

Voraussetzung

- Benutzergruppe Experte oder höher

  Wenn die Aktivierung Änderungen im Bereich **Sicherheitsrelevante Kommunikationsparameter** bewirken würde, muss die Benutzergruppe Sicherheitsinstandhalter oder höher ausgewählt sein.
- In der Betriebsart AUT oder AUT EXT:

  Das Projekt kann nur aktiviert werden, wenn sich dadurch nur KRL-Programme ändern. Wenn das Projekt Einstellungen enthält, die andere Änderungen bewirken würden, kann es nicht aktiviert werden.

Wenn auf der Robotersteuerung eine der Optionen KUKA.SafeOperation oder KUKA.SafeRangeMonitoring installiert ist, können andere Benutzergruppen gelten. Informationen sind den Dokumentationen der genannten Optionen zu entnehmen.

Vorbereitung

Es gibt 2 Möglichkeiten, wie man zum ersten Schritt der untenstehenden Vorgehensweise gelangen kann.

- Die Projektaktivierung ist die direkte Fortsetzung eines anderen Ablaufs, z. B. der Wiederherstellung eines Projekts.

  In dem Fall ist diese Vorbereitung nicht notwendig.
- Oder: Man führt die Projektaktivierung als eigenständigen Ablauf durch.

  In dem Fall gelangt man über diese Vorbereitung zur Vorgehensweise.

Vorbereitung:

1. Auf der smartHMI das WorkVisual-Symbol berühren, dann auf **Öffnen** gehen. Das Fenster **Projektverwaltung** öffnet sich.
2. Das gewünschte Projekt markieren und mit der Schaltfläche **Aktivieren** aktivieren.

Vorgehensweise

1. Die KUKA smartHMI zeigt die Sicherheitsabfrage *Wollen Sie die Aktivierung des Projektes […] zulassen?* an. Zusätzlich wird angezeigt, ob durch die Aktivierung ein Projekt überschrieben würde, und wenn ja, welches.

   Wenn kein relevantes Projekt überschrieben wird: Die Abfrage innerhalb von 30 min mit **Ja** bestätigen.
2. Es wird eine Übersicht über die Änderungen angezeigt, die im Vergleich zum noch aktiven Projekt auf der Robotersteuerung vorgenommen werden. Über die Checkbox **Details** kann man Details zu den Änderungen anzeigen.

⚠ WARNUNG Wenn in der Übersicht unter der Überschrift **Sicherheitsrelevante Kommunikationsparameter** Änderungen genannt sind, bedeutet dies, dass das Verhalten des NOT-HALT und des Signals "Bedienerschutz" gegenüber dem bisherigen Projekt verändert sein kann.
Nach der Aktivierung des Projekts müssen deshalb der NOT-HALT und das Signal "Bedienerschutz" auf ihre sichere Funktion geprüft werden. Wenn das Projekt auf mehreren Robotersteuerungen aktiviert wird, ist diese Prüfung für jede Robotersteuerung durchzuführen. Wenn die Prüfung unterlassen wird, können Tod, Verletzungen oder Sachschäden die Folge sein.

3. Die Übersicht zeigt die Sicherheitsabfrage *Wollen Sie fortfahren?* an. Mit **Ja** beantworten. Das Projekt wird auf der Robotersteuerung aktiviert.

⚠ WARNUNG Nach der Aktivierung eines Projekts auf der Robotersteuerung muss dort die Sicherheitskonfiguration geprüft werden! Wenn dies nicht geschieht, wird der Roboter eventuell mit falschen Daten betrieben. Tod, Verletzungen oder Sachschäden können die Folge sein.
(>>> 6.5 "Sicherheitskonfiguration der Robotersteuerung prüfen" Seite 173)

⚠ WARNUNG Wenn die Aktivierung eines Projekts fehlschlägt, wird eine Fehlermeldung angezeigt. In diesem Fall muss eine der folgenden Maßnahmen durchgeführt werden:

- Entweder erneut ein Projekt aktivieren. (Dasselbe oder ein anderes).
- Oder die Robotersteuerung mit einem Kaltstart neu starten.

Bei einem KSS/VSS-Update werden Initialprojekt und Basisprojekt durch Kopien des aktiven Projekts überschrieben.

7.11.3 Fenster Projektverwaltung

Übersicht

Das Fenster **Projektverwaltung** öffnet man über das WorkVisual-Symbol auf der smartHMI.

Neben den regulären Projekten enthält das Fenster **Projektverwaltung** folgende spezielle Projekte:

Projekt	Beschreibung
Initialprojekt	Das Initialprojekt ist immer vorhanden. Es kann vom Benutzer nicht verändert werden. Es enthält den Zustand der Robotersteuerung bei der Auslieferung.
Basisprojekt	Der Benutzer kann das aktive Projekt als Basisprojekt speichern. Diese Funktionalität wird in der Regel dafür benutzt, um einen funktionstüchtigen, bewährten Projektzustand zu sichern. Das Basisprojekt kann nicht aktiviert, aber kopiert werden. Das Basisprojekt kann vom Benutzer nicht mehr verändert werden. Es kann aber durch das Speichern eines neuen Basisprojekts überschrieben werden (nach einer Sicherheitsabfrage). Wenn ein Projekt aktiviert wird, das nicht alle Konfigurations-Dateien enthält, werden die fehlenden Informationen aus dem Basisprojekt übernommen. Die kann z. B. der Fall sein, wenn ein Projekt aus einer früheren WorkVisual-Version aktiviert wird. (Zu den Konfigurations-Dateien zählen Maschinendaten-Dateien, Dateien der Sicherheitskonfiguration und zahlreiche weitere.)

Bei einem KSS/VSS-Update werden Initialprojekt und Basisprojekt durch Kopien des aktiven Projekts überschrieben.

Beschreibung

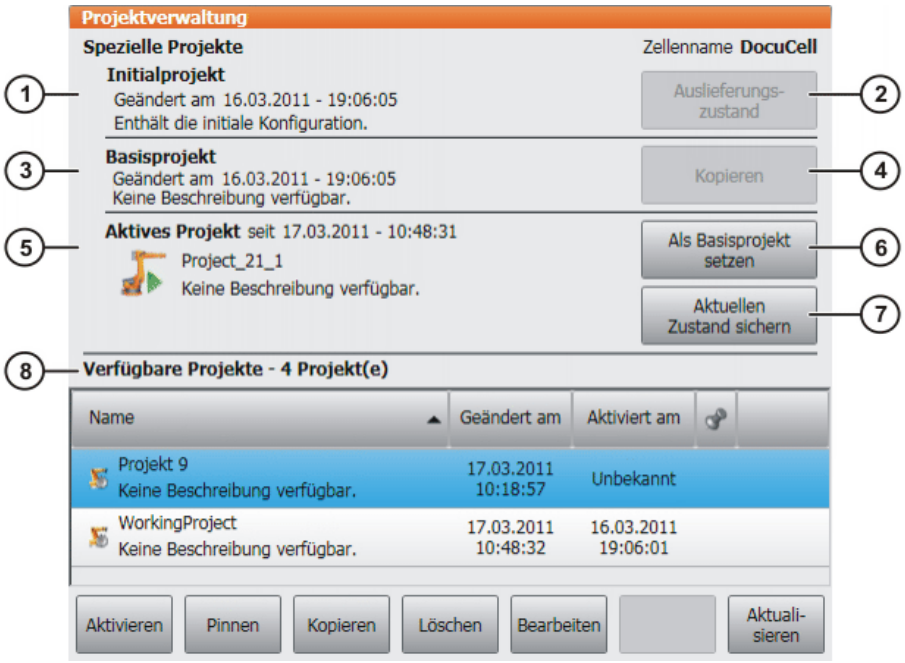


Abb. 7-15: Fenster Projektverwaltung

Pos.	Beschreibung
1	Das Initialprojekt wird angezeigt.
2	Stellt den Auslieferungszustand der Robotersteuerung wieder her. Steht ab der Benutzergruppe Experte zur Verfügung.
3	Das Basisprojekt wird angezeigt.
4	Erstellt eine Kopie des Basisprojekts. Steht ab der Benutzergruppe Experte zur Verfügung.

Pos.	Beschreibung
5	Das aktive Projekt wird angezeigt.
6	Speichert das aktive Projekt als Basisprojekt. Das aktive Projekt bleibt aktiv. Steht ab der Benutzergruppe Experte zur Verfügung.
7	Erstellt eine gepinnte Kopie des aktiven Projekts. Steht ab der Benutzergruppe Experte zur Verfügung.
8	Liste der inaktiven Projekte (außer Basis- und Initialprojekt)

Bei allen Kopiervorgängen öffnet sich ein Fenster, in dem ein Name und eine Beschreibung für die Kopie eingegeben werden können.

Schaltflächen

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Aktivieren	Aktiviert das markierte Projekt. Wenn das markierte Projekt gepinnt ist: Erstellt eine Kopie des markierten Projekts. (Ein gepinntes Projekt kann nicht selber aktiviert werden, nur eine Kopie davon.) Der Benutzer kann dann entscheiden, ob die Kopie gleich aktiviert werden soll oder das bisherige Projekt aktiv bleiben soll. Steht ab der Benutzergruppe Experte zur Verfügung.
Pinnen	(>>> 7.11.1 "Projekt auf der Robotersteuerung pinnen" Seite 259) Steht nur zur Verfügung, wenn ein ungepinntes Projekt markiert ist. Steht ab der Benutzergruppe Experte zur Verfügung.
Entpinnen	Entpinnt das Projekt. Steht nur zur Verfügung, wenn ein gepinntes Projekt markiert ist. Steht ab der Benutzergruppe Experte zur Verfügung.
Kopieren	Kopiert das markierte Projekt. Steht ab der Benutzergruppe Experte zur Verfügung.
Löschen	Löscht das markierte Projekt. Steht nur zur Verfügung, wenn ein nicht aktives, nicht gepinntes Projekt markiert ist. Steht ab der Benutzergruppe Experte zur Verfügung.
Bearbeiten	Öffnet ein Fenster, in dem der Name und/oder die Beschreibung des markierten Projekts geändert werden können. Steht nur zur Verfügung, wenn ein nicht gepinntes Projekt markiert ist. Steht ab der Benutzergruppe Experte zur Verfügung.
Aktualisieren	Aktualisiert die Projektliste. Auf diese Weise werden z. B. Projekte angezeigt, die seit dem Öffnen der Anzeige auf die Robotersteuerung übertragen wurden.

7.12 Backup-Manager

7.12.1 Übersicht Backup-Manager

Übersicht

Der Backup-Manager ermöglicht es, Projekte, Optionspakete und RDC-Daten zu sichern und wiederherzustellen.

Die Default-Einstellungen für den Backup-Manager sind im Fenster **Backup-Konfiguration** festgelegt. Die Einstellungen können geändert werden.

(>>> 7.12.5 "Backup-Manager konfigurieren" Seite 266)

Start-Arten

Es gibt mehrere Möglichkeiten, eine Sicherung oder eine Wiederherstellung zu starten:

	Sicherung	Wiederherstellung
Manuell	Ja	Ja
Automatisch, in Intervallen	Ja	Nein
Über Eingänge (nur möglich in AUT oder AUT EXT)	Ja	Ja (nur Projekte und Optionspakete, keine RDC-Daten)

Während eine Sicherung/Wiederherstellung läuft, kann keine weitere Sicherung/Wiederherstellung gestartet werden. Grundsätzlich schließen sich die Start-Arten jedoch nicht aus. Wenn z. B. eine automatische Sicherung konfiguriert ist, kann trotzdem eine manuelle Sicherung durchgeführt werden.

	Durchführung
Manuell	(>>> 7.12.2 "Projekte, Optionspakete und RDC-Daten manuell sichern" Seite 263) (>>> 7.12.3 "Projekte und Optionspakete manuell wiederherstellen" Seite 264) (>>> 7.12.4 "RDC-Daten manuell wiederherstellen" Seite 266)
Automatisch, in Intervallen	Wenn Sicherungen automatisch gestartet werden sollen, muss dies im Fenster **Backup-Konfiguration** konfiguriert werden.
Über Eingänge	Wenn Sicherungen/Wiederherstellungen über Eingänge gestartet werden sollen, muss dies im Fenster **Backup-Konfiguration** konfiguriert werden.

7.12.2 Projekte, Optionspakete und RDC-Daten manuell sichern

Beschreibung

Projekte

Defaultmäßig werden folgende Projekte gesichert:

- Aktives Projekt
- Initialprojekt
- Basisprojekt

Optionspakete

Optionspakete werden unter folgenden Voraussetzungen gesichert:

- Zum Optionspaket gehört eine KOP-Datei.

- Das Optionspaket wurde dem Projekt ursprünglich in WorkVisual hinzugefügt. Das Projekt ist jetzt auf der Robotersteuerung aktiv.

  Oder:

  Das Optionspaket wurde im aktiven Projekt über **Inbetriebnahme** > **Zusatzsoftware** installiert. Das Optionspaket lag bei der Installation als einzelne KOP-Datei vor. (Nicht als Verzeichnisstruktur!)

RDC-Daten

Bei der Sicherung wird immer eine Datei *[Roboterseriennummer]*.RDC erstellt. Sie enthält die CAL-, MAM- und PID-Datei. Es sind nicht immer alle Datein vorhanden (abhängig vom Roboter).

Vorgehensweise

- Im Hauptmenü **Datei** > **Backup-Manager** wählen, und dann einen der folgenden Menüpunkte wählen:
  - **Sichern**

    Das Zielverzeichnis ist D:\ProjectBackup, sofern kein anderes Verzeichnis konfiguriert wurde. Der Pfad wird automatisch angelegt, falls er noch nicht vorhanden ist.
  - Oder **Sichern unter ...**

    Hier kann ein Zielverzeichnis ausgewählt werden. Es gilt nur für diese Sicherung.

Die Sicherung wird durchgeführt. Die Robotersteuerung zeigt mit Meldung an, wenn die Sicherung erfolgreich verlaufen ist. Sie gibt pro Projekt und pro Optionspaket eine Meldung aus, sowie eine Meldung zu den RDC-Daten.

Optionspakete werden jedoch nicht gesichert, wenn die gleiche Paketversion im Zielverzeichnis bereits vorhanden ist.

7.12.3 Projekte und Optionspakete manuell wiederherstellen

Beschreibung

Projekte

Defaultmäßig werden folgende Projekte wiederhergestellt:

- Aktives Projekt (d. h. das Projekt, das bei der Sicherung das aktive Projekt war)
- Initialprojekt
- Basisprojekt

Nach einer manuellen Wiederherstellung aktiviert die Robotersteuerung das ehemals aktive Projekt wieder, wenn bestimmte Voraussetzungen erfüllt sind. Wenn nicht, bleibt das Projekt inaktiv. Der Benutzer kann es zum gewünschten Zeitpunkt selber aktivieren.

Nach einer Wiederherstellung über Eingänge bleibt das ehemals aktive Projekt ebenfalls inaktiv.

Optionspakete

Es werden nicht zwangsläufig alle Optionspakete wiederhergestellt, deren KOP-Datei im Quellverzeichnis liegt. Ein Optionspaket wird unter folgenden Voraussetzungen wiederhergestellt:

- Zum Zeitpunkt der Sicherung war das Optionspaket im aktiven Projekt vorhanden.
- Zum Zeitpunkt der Wiederherstellung ist diese Version des Optionspakets nicht auf der Robotersteuerung installiert.

Nach der Wiederherstellung installieren sich manche Optionspakete automatisch, wenn das zugehörige Projekt aktiviert wird.

Optionspakete, die sich nicht automatisch installieren, stehen unter **Inbetriebnahme** > **Zusatzsoftware** für die Installation zur Verfügung. Die Robotersteuerung weist mit einer Meldung darauf hin, dass das Optionspaket noch installiert werden muss.

Menüpunkte

Für die Wiederherstellung stehen 2 Menüpunkte zur Auswahl. Sie unterscheiden sich in Bezug auf das Quellverzeichnis, auf das zugegriffen wird.

- **Wiederherstellen** >

  Das Quellverzeichnis ist D:\ProjectBackup, sofern kein anderes Verzeichnis konfiguriert wurde.
- Oder **Wiederherstellen von ...** >

  Hier kann ein Quellverzeichnis ausgewählt werden. Es gilt nur für diese Wiederherstellung.

Voraussetzung

Für die Wiederherstellung an sich sind keine besonderen Voraussetzungen erforderlich. Wenn ein Projekt aktiviert werden soll, müssen die im Folgenden genannten Voraussetzungen erfüllt sein.

- Benutzergruppe Experte oder höher

  Wenn die Aktivierung Änderungen im Bereich **Sicherheitsrelevante Kommunikationsparameter** bewirken würde, muss die Benutzergruppe Sicherheitsinstandhalter oder höher ausgewählt sein.
- In der Betriebsart AUT oder AUT EXT:

  Das Projekt kann nur aktiviert werden, wenn sich dadurch nur KRL-Programme ändern. Wenn das Projekt Einstellungen enthält, die andere Änderungen bewirken würden, kann es nicht aktiviert werden.

Wenn auf der Robotersteuerung eine der Optionen KUKA.SafeOperation oder KUKA.SafeRangeMonitoring installiert ist, können andere Benutzergruppen gelten. Informationen sind den Dokumentationen der genannten Optionen zu entnehmen.

Vorgehensweise AUT / AUT EXT

1. Im Hauptmenü **Datei** > **Backup-Manager** > **Wiederherstellen** > **Projekte und Optionen** wählen.

   Oder:

   Datei > **Backup-Manager** > **Wiederherstellen von ...** > **Projekte und Optionen** wählen und dann das Quellverzeichnis auswählen.
2. Die Robotersteuerung zeigt mit je einer Meldung pro Projekt und pro Optionspaket an, wenn die Wiederherstellung erfolgreich verlaufen ist.
   - Wenn ein Optionspaket wiederhergestellt wurde, muss man erst das Paket installieren, bevor man das Projekt aktivieren kann. Die Robotersteuerung weist darauf mit einer Meldung hin.
   - Wenn kein Optionspaket wiederhergestellt wurde, beginnt die Robotersteuerung mit der Aktivierung des Projekts, ohne Sicherheitsabfrage.
3. Zu den weiteren Schritte bei der Aktivierung: (>>> 7.11.2 "Projekt aktivieren" Seite 259)

Die Sicherheitshinweise zur Projektaktivierung beachten!

Vorgehensweise T1 / T2

1. Im Hauptmenü **Datei** > **Backup-Manager** > **Wiederherstellen** > **Projekte und Optionen** wählen.

   Oder:

   Datei > **Backup-Manager** > **Wiederherstellen von ...** > **Projekte und Optionen** wählen und dann das Quellverzeichnis auswählen.

2. Die Robotersteuerung zeigt mit je einer Meldung pro Projekt und pro Optionspaket an, wenn die Wiederherstellung erfolgreich verlaufen ist.
   - Wenn ein Optionspaket wiederhergestellt wurde, muss man erst das Paket installieren, bevor man das Projekt aktivieren kann. Die Robotersteuerung weist darauf mit einer Meldung hin.
   - Wenn kein Optionspaket wiederhergestellt wurde, zeigt die Robotersteuerung folgende Sicherheitsabfrage an:

     Wollen Sie die Aktivierung des Projektes [...] zulassen? Zusätzlich wird angezeigt, ob durch die Aktivierung ein Projekt überschrieben würde, und wenn ja, welches.

     Wenn kein relevantes Projekt überschrieben wird: Die Abfrage innerhalb von 30 min mit **Ja** beantworten.
3. Zu den weiteren Schritte bei der Aktivierung: (>>> 7.11.2 "Projekt aktivieren" Seite 259)

Die Sicherheitshinweise zur Projektaktivierung beachten!

7.12.4 RDC-Daten manuell wiederherstellen

Menüpunkte

Für die Wiederherstellung stehen 2 Menüpunkte zur Auswahl. Sie unterscheiden sich in Bezug auf das Quellverzeichnis, auf das zugegriffen wird.

- **Wiederherstellen** >

  Das Quellverzeichnis ist D:\ProjectBackup, sofern kein anderes Verzeichnis konfiguriert wurde.
- Oder **Wiederherstellen von ...** >

  Hier kann ein Quellverzeichnis ausgewählt werden. Es gilt nur für diese Wiederherstellung.

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Datei** > **Backup-Manager** > **Wiederherstellen** > **RDC-Daten** wählen.

   Oder:

   Datei > **Backup-Manager** > **Wiederherstellen von ...** > **RDC-Daten** wählen und dann das Quellverzeichnis auswählen.
2. Ein Fenster öffnet sich. Bei den Daten, die wiederhergestellt werden sollen, ein Häkchen setzen:
   - **PID-Datei**, **MAM-Datei** und/oder **CAL-Datei**

   Wenn ein Eintrag ausgegraut ist und kein Häkchen gesetzt werden kann, bedeutet dies, dass diese Datei in der Sicherung nicht vorhanden ist.
3. Die Auswahl mit **Wiederherstellen** bestätigen.

   Wenn die Wiederherstellung abgeschlossen ist, wird folgende Meldung angezeigt: *RDC Daten wurde erfolgreich von {0} wiederhergestellt.*

7.12.5 Backup-Manager konfigurieren

Voraussetzung

- Benutzergruppe Experte
- Für Änderungen in der Registerkarte **Signalschnittstelle** zusätzlich:

  Betriebsart T1 oder T2

Vorgehensweise

1. Im Hauptmenü **Datei** > **Backup-Manager** > **Backup-Konfiguration** wählen.

   Das Fenster **Backup-Konfiguration** öffnet sich.

2. In den Registerkarten die gewünschten Änderungen vornehmen.
   - **Backup-Konfiguration** enthält die allgemeinen Einstellungen.
     Außerdem kann hier bei Bedarf die automatische Sicherung konfiguriert werden.
   - Unter **Signalschnittstelle** kann bei Bedarf die E/A-Ansteuerung konfiguriert werden.
3. Das Fenster schließen.
4. Die Sicherheitsabfrage, ob die Änderungen gespeichert werden sollen, mit **Ja** beantworten.

7.12.5.1 Registerkarte Backup-Konfiguration

Backup-Konfiguration

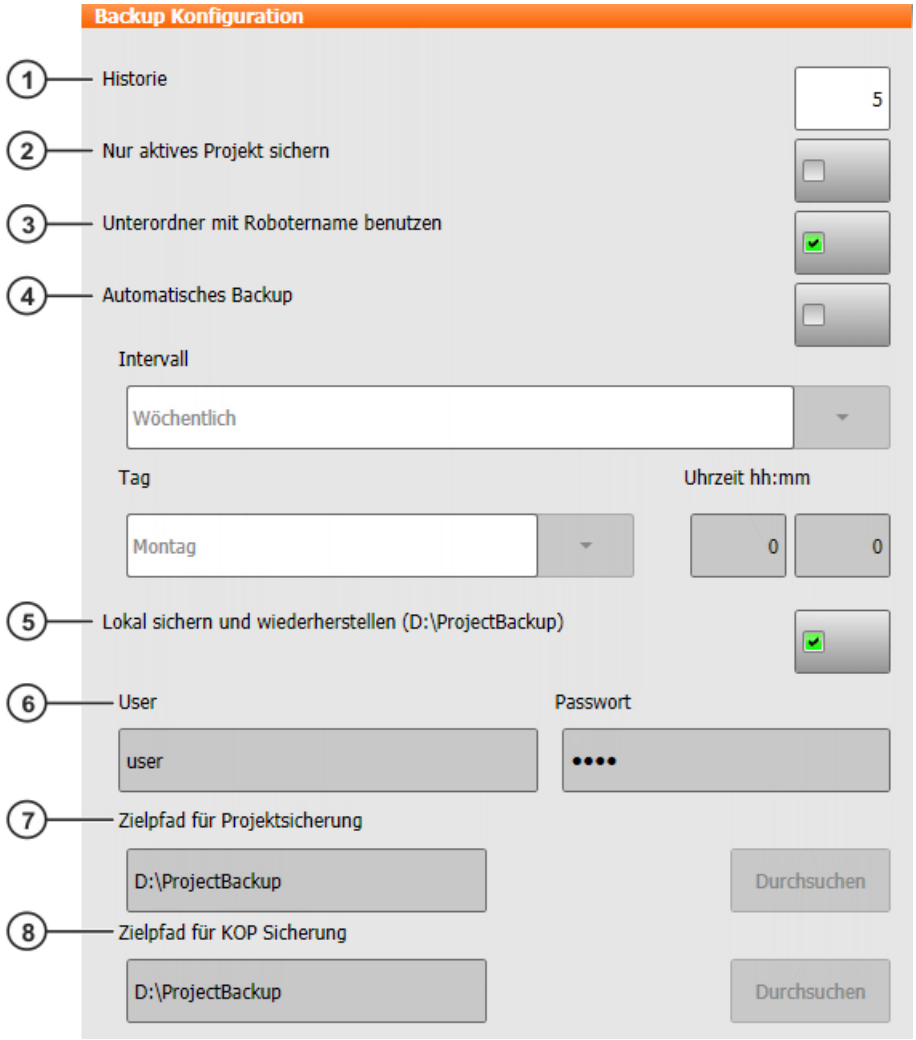


Abb. 7-16: Registerkarte Backup-Konfiguration

Pos.	Beschreibung
1	Maximale Anzahl von Unterordnern für ältere Sicherungen Eine Sicherung überschreibt bereits vorhandene Sicherungsdateien nicht, sondern für diese wird automatisch ein Unterordner angelegt, in den sie verschoben werden. Wenn die maximale Anzahl von Unterordnern errreicht ist, wird bei der nächsten Sicherung der älteste Unterordner gelöscht und ein neuer angelegt. ■ **0 … 50** Default: 5
2	■ **Aktiv**. Beim Sichern wird das aktive Projekt gesichert. Beim Wiederherstellen wird nur das Projekt wiederhergestellt, das bei der Sicherung das aktive Projekt war. ■ **Inaktiv** (Default): Beim Sichern werden folgende Projekte gesichert: Aktives Projekt, Basisprojekt, Initialprojekt Beim Wiederherstellen werden diese Projekte wiederhergestellt.
3	■ **Aktiv** (Default): Beim Sichern legt die Robotersteuerung Projekte und RDC-Daten im Verzeichnis [**Zielpfad für Projektsicherung**]\[Robotername]\ ab. Beim Wiederherstellen greift die Robotersteuerung auf dieses Verzeichnis zu. **Hinweis:** Diese Einstellung wählen, wenn mehrere Robotersteuerungen denselben Zielpfad nutzen. ■ **Inaktiv**: Beim Sichern legt die Robotersteuerung Projekte und RDC-Daten im Verzeichnis [**Zielpfad für Projektsicherung**]\ ab. Beim Wiederherstellen greift die Robotersteuerung auf dieses Verzeichnis zu. **Hinweis:** Optionspakete werden unabhängig von dieser Einstellung immer unter [**Zielpfad für KOP Sicherung**]\OptionPackages\ abgelegt.
4	■ **Aktiv**: Die Robotersteuerung führt automatisch Sicherungen durch. Folgende Parameter bestimmen den Zeitpunkt: **Intervall**, **Tag**, **Uhrzeit hh:mm** **Hinweis:** Nach einer automatischen Sicherung zeigt die Robotersteuerung keine Erfolgsmeldungen an. Wenn die Robotersteuerung zum konfigurierten Zeitpunkt ausgeschaltet war, führt sie eine Sicherung durch, sobald sie wieder eingeschaltet wird. Hierbei führt sie immer nur eine Sicherung durch, auch wenn der Zeitpunkt mehrmals verpasst wurde. ■ **Inaktiv** (Default): Keine automatische Sicherung.
5	■ **Aktiv** (Default): Das Zielverzeichnis für Sicherungen und das Quellverzeichnis für Wiederherstellungen ist D:\ProjectBackup. ■ **Inaktiv**: Das Zielverzeichnis für Sicherungen und das Quellverzeichnis für Wiederherstellungen wird über folgende Parameter bestimmt: **Zielpfad für Projektsicherung**, **Zielpfad für KOP-Sicherung**
Die folgenden Parameter sind nur editierbar, wenn die Option **Lokal sichern und wiederherstellen (D:\ProjectBackup)** auf **Inaktiv** gesetzt ist.	

Pos.	Beschreibung
6	Wenn die Robotersteuerung beim Speichern und/oder Wiederherstellen auf das Netzwerk zugreifen muss muss und dafür eine Authentifizierung erforderlich ist, werden die hier hinterlegten Daten verwendet. Wenn keine Authentifizierung erforderlich ist, haben die Daten keine Auswirkung. ■ **User**: Benutzername Default: user ■ **Passwort**: Passwort. Beim Eingeben wird nur die Anzahl der Zeichen angezeigt, nicht die Zeichen selber. Default: kuka
7	Für Projekte und RDC-Daten: Zielverzeichnis für Sicherungen und Quellverzeichnis für Wiederherstellungen Man kann zu dem gewünschten Verzeichnis navigieren oder es direkt eingeben. In letzterem Fall wird der Pfad automatisch angelegt, falls er noch nicht vorhanden ist.
8	Für KOP-Dateien: Zielverzeichnis für Sicherungen und Quellverzeichnis für Wiederherstellungen Man kann zu dem gewünschten Verzeichnis navigieren oder es direkt eingeben. In letzterem Fall wird der Pfad automatisch angelegt, falls er noch nicht vorhanden ist.

7.12.5.2 Registerkarte Signalschnittstelle

Wenn eine Wiederherstellung über Eingänge gestartet wird, aktiviert die Robotersteuerung – im Gegensatz zur manuellen Wiederherstellung – das ehemals aktive Projekt nicht. Der Benutzer kann es zum gewünschten Zeitpunkt selber aktivieren.

Signalschnitt-stelle

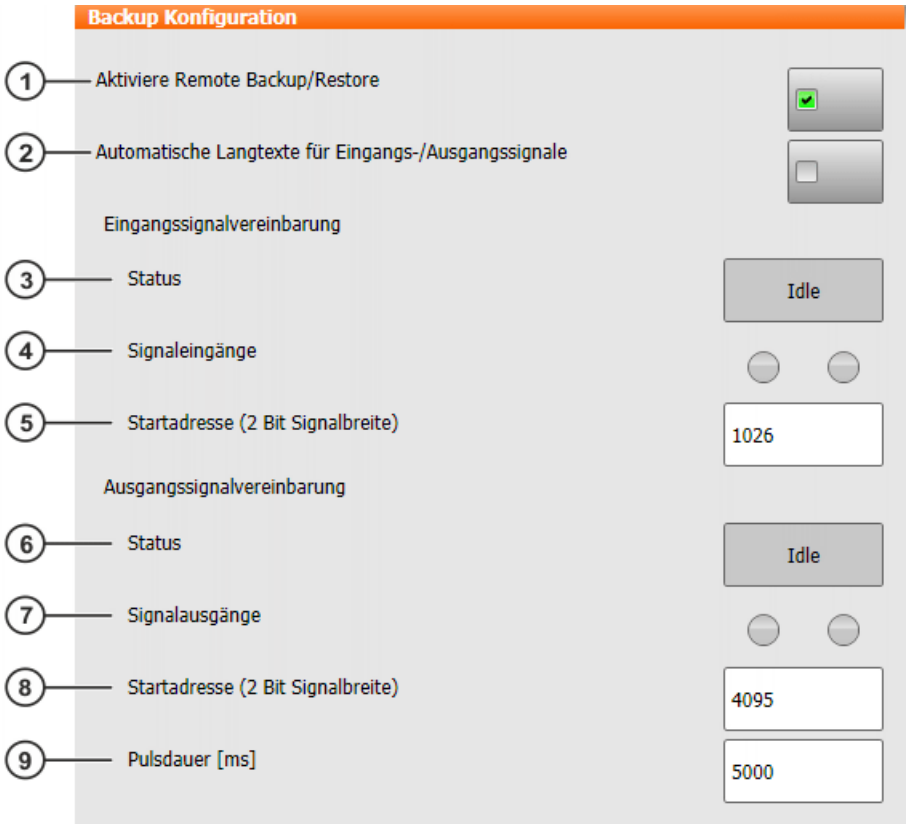


Abb. 7-17: Registerkarte Signalschnittstelle

Pos.	Beschreibung
1	■ **Aktiv**: Über das unter **Startadresse** konfigurierte Eingangssignal können Sicherungen und Wiederherstellungen gestartet werden. ■ **Inaktiv** (Default): Kein Start über Eingangssignal möglich.
Die folgenden Parameter sind nur editierbar, wenn die Option **Aktiviere Remote Backup/Restore** auf **Aktiv** gesetzt ist.	
2	■ **Aktiv**: Den unter **Startadresse** definierten Signalen werden folgende Langtext-Namen zugewiesen: **BM Eingangssignal**, **BM Ausgangssignal** Wenn die Signale bereits Langtext-Namen haben, werden diese nicht überschrieben. ■ **Inaktiv** (Default): Keine Langtext-Namen Wenn vorher **Aktiv** ausgewählt war, entfernt **Inaktiv** die Langtext-Namen wieder.
3	Zeigt an, wenn das Eingangssignal eine Sicherung oder Wiederherstellung startet. (Anzeige während der Signaldauer) (>>> "Eingangssignale" Seite 271)
4	■ Linkes Lämpchen: Zustand des Eingangs **Startadresse**. ■ Rechtes Lämpchen: Zustand des Eingangs **Startadresse** + 1. Zustand (nicht editierbar): Grün = 1; Grau = 0
5	Eingangssignal, bestehend aus folgenden Eingängen: **Startadresse** und **Startadresse** + 1. Default: 1 026
6	Zeigt die laufende Aktion an oder das Ergebnis. (>>> "Ausgangssignale" Seite 271)

Pos.	Beschreibung
7	■ Linkes Lämpchen: Zustand des Ausgangs **Startadresse**. ■ Rechtes Lämpchen: Zustand des Ausgangs **Startadresse** + 1. Zustand (nicht editierbar): Grün = 1; Grau = 0
8	Ausgangssignal, bestehend aus folgenden Ausgängen: **Startadresse** und **Startadresse** + 1. Default: 4 095
9	Dauer, wie lange das Ausgangssignal 01 oder 11 anliegt, bevor die Robotersteuerung wieder 00 setzt Einheit: ms Default: 5 000 ms

Eingangssignale

Signal	→ Erzeugte Aktion / Anzeige im Feld Status
00	Keine Aktion / **Idle**
01	Startet eine Sicherung / **Backup**
10	Startet eine Wiederherstellung / **Restore**
11	Nicht definierter Zustand

Ein dauerhaft anliegendes Eingangssignal führt nicht zur Wiederholung der Aktion.

Während eine Sicherung/Wiederherstellung läuft, kann keine weitere Sicherung/Wiederherstellung gestartet werden.

Ausgangssignale

Die Signale 01 und 11 liegen während der unter **Pulsdauer [ms]** konfigurierten Zeit an, danach wechselt das Signal zu 00.

Aktion / Ergebnis	→ Erzeugtes Signal / Anzeige im Feld Status
Keine Aktion läuft	00 / **Idle**
Sicherung läuft	10 / **Backup**
Wiederherstellung läuft	10 / **Restore**
Sicherung erfolgreich	01 / **BackupSuccess**
Wiederherstellung erfolgreich	01 / **RestoreSuccess**
Sicherung nicht erfolgreich	11 / **BackupError**
Wiederherstellung nicht erfolgreich	11 / **RestoreError**

8 Programmausführung

8.1 Programmablaufart auswählen

Vorgehensweise

1. Die Statusanzeige **Programmablaufart** berühren. Das Fenster **Programmablaufart** öffnet sich.
2. Die gewünschte Programmablaufart auswählen.
   (>>> 8.2 "Programmablaufarten" Seite 273)
   Das Fenster schließt sich und die gewählte Programmablaufart wird übernommen.

8.2 Programmablaufarten

Bezeichnung	Statusanzeige	Beschreibung
Go #GO		Das Programm läuft ohne Stopp bis zum Programmende ab.
Bewegung #MSTEP		Das Programm läuft mit einem Stopp an jedem Punkt ab, auch an Hilfspunkten und an Spline-Segmentpunkten. Die Start-Taste muss für jeden Punkt neu gedrückt werden. Das Programm läuft ohne Vorlauf ab.
Einzelschritt #ISTEP		Das Programm läuft mit einem Stopp nach jeder Programmzeile ab. Auch nach Programmzeilen, die nicht sichtbar sind, und nach Leerzeilen wird gestoppt. Die Start-Taste muss für jede Zeile neu gedrückt werden. Das Programm läuft ohne Vorlauf ab. **Einzelschritt** steht nur in der Benutzergruppe Experte zur Verfügung.
Rückwärts #BSTEP		Diese Programmablaufart wird automatisch angewählt, wenn die Start-Rückwärts-Taste gedrückt wird. Sie kann nicht auf eine andere Art angewählt werden. Das Verhalten ist wie bei **Bewegung**, mit folgender Ausnahme: CIRC-Bewegungen werden rückwärts so abgefahren wie beim letzten Vorwärtsfahren. D.h. wenn vorwärts am Hilfspunkt nicht gestoppt wurde, wird dort auch rückwärts nicht gestoppt. Für SCIRC-Bewegungen gilt die Ausnahme nicht. Hier wird rückwärts immer am Hilfspunkt gestoppt.

Für Systemintegratoren stehen zusätzlich die folgenden Programmablaufarten zur Verfügung.

Diese Programmablaufarten können nur über die Variablenkorrektur ausgewählt werden. Systemvariable für die Programmablaufart: $PRO_MODE.

Bezeichnung	Statusanzeige	Beschreibung
Program Step #PSTEP		Das Programm wird schrittweise und ohne Vorlauf abgefahren. Unterprogramme werden komplett durchlaufen.
Continuous Step #CSTEP		Überschleifpunkte werden mit Vorlauf bearbeitet, d. h. sie werden überschliffen. Genauhaltpunkte werden ohne Vorlauf und mit einem Stopp nach dem Bewegungssatz abgearbeitet.

8.3 Vorlauf

Der Vorlauf ist die maximale Anzahl der Bewegungssätze, die die Robotersteuerung beim Programmlauf im Voraus berechnet und plant. Die tatsächliche Anzahl ist abhängig von der Rechnerauslastung.

Der Vorlauf bezieht sich auf die aktuelle Position des Satzzeigers. Er wird über die Systemvariable $ADVANCE eingestellt:

- Defaultwert: 3
- Maximalwert: 5

Der Vorlauf ist unter anderem notwendig, um Überschleifbewegungen berechnen zu können. Bei $ADVANCE = 0 ist kein Überschleifen möglich.

Manche Anweisungen lösen einen Vorlaufstopp aus. Dazu gehören Anweisungen, die die Peripherie beeinflussen, z. B. OUT-Anweisungen.

8.4 Satzzeiger

Übersicht

Beim Programmlauf zeigt der Satzzeiger verschiedene Informationen an:

- Welche Bewegung der Roboter gerade abfährt oder abgeschlossen hat
- Ob gerade ein Hilfspunkt oder ein Zielpunkt angefahren wird
- Die Richtung, in der der Roboter das Programm abfährt

Zeiger	Richtung	Beschreibung
	Vorwärts	Der Zielpunkt wird angefahren.
	Rückwärts	
	Vorwärts	Der Zielpunkt wurde mit Genauhalt erreicht.
	Rückwärts	
	Vorwärts	Der Hilfspunkt wird angefahren.
	Rückwärts	
	Vorwärts	Der Hilfspunkt wurde mit Genauhalt erreicht.
	Rückwärts	

Beispiele für Vorwärts

```
5  PTP P3 Vel=100 % PDAT1 Tool[1] Base[0]
6  PTP P4 Vel=100 % PDAT2 Tool[1] Base[0]
7  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
```

Abb. 8-1: Der Roboter bewegt sich von P3 nach P4

```
5  PTP P3 Vel=100 % PDAT1 Tool[1] Base[0]
6  PTP P4 Vel=100 % PDAT2 Tool[1] Base[0]
7  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
```

Abb. 8-2: Der Roboter hat P4 mit Genauhalt erreicht

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-3: Der Roboter bewegt sich von P5 zum Hilfspunkt P6

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-4: Der Roboter hat den Hilfspunkt P6 mit Genauhalt erreicht

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-5: Der Roboter bewegt sich vom Hilfspunkt P6 nach P7

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-6: Der Roboter hat P7 mit Genauhalt erreicht

Beispiele für Rückwärts

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-7: Der Roboter bewegt sich von P8 nach P7

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-8: Der Roboter hat P7 mit Genauhalt erreicht

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-9: Der Roboter bewegt sich von P7 zum Hilfspunkt P6

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]
8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

Abb. 8-10: Der Roboter hat den Hilfspunkt P6 mit Genauhalt erreicht

Doppelpfeil nach oben/unten

Wenn das Programmfenster einen Abschnitt anzeigt, in dem der Satzzeiger gerade nicht ist, zeigt ein Doppelpfeil an, in welcher Richtung er sich befindet.

```
7  PTP P6 Vel=100 % PDAT4 Tool[1] Base[0]
8  PTP P7 Vel=100 % PDAT5 Tool[1] Base[0]
```

Abb. 8-11: Der Satzzeiger befindet sich weiter oben im Programm

```
14  PTP P13 Vel=100 % PDAT11 Tool[1] Base[0]
15  PTP P14 Vel=100 % PDAT12 Tool[1] Base[0]
```

Abb. 8-12: Der Satzzeiger befindet sich weiter unten im Programm

8.5 Programm-Override (POV) einstellen

Beschreibung

Der Programm-Override ist die Geschwindigkeit des Roboters beim Programmablauf. Der Programm-Override wird in Prozent angegeben und bezieht sich auf die programmierte Geschwindigkeit.

In der Betriebsart T1 ist die maximale Geschwindigkeit 250 mm/s, unabhängig vom eingestellten Wert.

Vorgehensweise

1. Die Statusanzeige **POV/HOV** berühren. Das Fenster **Overrides** öffnet sich.
2. Den gewünschten Programm-Override einstellen. Er kann entweder über die Plus-Minus-Tasten oder über den Regler eingestellt werden.
   - Plus-Minus-Tasten: Einstellung möglich in den Schritten 100%, 75%, 50%, 30%, 10%, 3%, 1%
   - Regler: Der Override kann in 1%-Schritten geändert werden.
3. Die Statusanzeige **POV/HOV** erneut berühren. (Oder den Bereich außerhalb des Fensters berühren.)

   Das Fenster schließt sich und der gewählte Override wird übernommen.

> Im Fenster **Overrides** kann über **Optionen** das Fenster **Handverfahroptionen** geöffnet werden.

Alternative Vorgehensweise

Alternativ kann der Override mit der Plus-Minus-Taste rechts am smartPAD eingestellt werden.

Einstellung möglich in den Schritten 100%, 75%, 50%, 30%, 10%, 3%, 1%.

8.6 Statusanzeige Roboter-Interpreter

Symbol	Farbe	Beschreibung
	grau	Kein Programm ist angewählt.
	gelb	Satzzeiger steht auf der ersten Zeile des angewählten Programms.
	grün	Programm ist angewählt und läuft ab.
	rot	Angewähltes und gestartetes Programm wurde angehalten.
	schwarz	Satzzeiger steht am Ende des angewählten Programms.

8.7 Programm vorwärts starten (manuell)

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1 oder T2

Vorgehensweise

1. Programmablaufart wählen.

2. Zustimmungsschalter gedrückt halten und warten, bis die Statusleiste "Antriebe bereit" anzeigt:

Abb. 8-13

3. SAK-Fahrt durchführen: Start-Taste drücken und halten, bis im Meldungsfenster "*SAK erreicht*" angezeigt wird. Der Roboter bleibt stehen.

⚠ VORSICHT Die SAK-Fahrt erfolgt als LIN- oder PTP-Bewegung von der Istposition zur Zielposition. Die Geschwindigkeit ist automatisch reduziert. Der Verlauf der Bewegung ist nicht sicher vorhersehbar. Die Bewegung während der SAK-Fahrt beobachten, damit der Roboter rechtzeitig gestoppt werden kann, falls sich eine Kollision abzeichnet.

4. Start-Taste drücken und gedrückt halten.
   Das Programm läuft ab, je nach Programmablaufart mit oder ohne Stopps.

Um ein manuell gestartetes Programm zu stoppen, die Start-Taste loslassen.

8.8 Programm vorwärts starten (automatisch)

Voraussetzung

- Programm ist angewählt.
- Betriebsart Automatik (nicht Automatik extern)

Vorgehensweise

1. Die Programmablaufart **Go** wählen.
2. Die Antriebe einschalten.
3. SAK-Fahrt durchführen:
   Start-Taste drücken und halten, bis im Meldungsfenster "SAK erreicht" angezeigt wird. Der Roboter bleibt stehen.

⚠ VORSICHT Die SAK-Fahrt erfolgt als LIN- oder PTP-Bewegung von der Istposition zur Zielposition. Die Geschwindigkeit ist automatisch reduziert. Der Verlauf der Bewegung ist nicht sicher vorhersehbar. Die Bewegung während der SAK-Fahrt beobachten, damit der Roboter rechtzeitig gestoppt werden kann, falls sich eine Kollision abzeichnet.

4. Start-Taste drücken. Das Programm läuft ab.

Um ein im Automatik-Betrieb gestartetes Programm zu stoppen, die STOP-Taste drücken.

8.9 Satzanwahl durchführen

Beschreibung

Ein Programm kann mit der Satzanwahl an einem beliebigen Punkt gestartet werden.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1 oder T2

Vorgehensweise

1. Programmablaufart wählen.
2. Den Bewegungssatz markieren, an dem das Programm gestartet werden soll.
3. **Satzanwahl** drücken. Der Satzzeiger zeigt auf den Bewegungssatz.

4. Zustimmungsschalter gedrückt halten und warten, bis die Statusleiste "Antriebe bereit" anzeigt:

5. SAK-Fahrt durchführen: Start-Taste drücken und halten, bis im Meldungsfenster "*SAK erreicht*" angezeigt wird. Der Roboter bleibt stehen.

⚠ VORSICHT Die SAK-Fahrt erfolgt als LIN- oder PTP-Bewegung von der Istposition zur Zielposition. Die Geschwindigkeit ist automatisch reduziert. Der Verlauf der Bewegung ist nicht sicher vorhersehbar. Die Bewegung während der SAK-Fahrt beobachten, damit der Roboter rechtzeitig gestoppt werden kann, falls sich eine Kollision abzeichnet.

6. Das Programm kann jetzt manuell oder automatisch gestartet werden. Es ist nicht notwendig, dabei die SAK-Fahrt nochmal durchzuführen.

8.10 Programm zurücksetzen

Beschreibung Um ein unterbrochenes Programm wieder von vorne zu starten, muss es zurückgesetzt werden. Dies bringt das Programm wieder in den Anfangszustand.

Voraussetzung
- Programm ist angewählt.

Vorgehensweise
- Menüfolge **Bearbeiten** > **Programm zurücksetzen** wählen.

Alternative Vorgehensweise
- In der Statusleiste die Statusanzeige **Roboter-Interpreter** berühren. Ein Fenster öffnet sich.
  Programm zurücksetzen wählen.

8.11 Automatik Extern-Betrieb starten

HINWEIS Im Automatik Extern-Betrieb gibt es keine SAK-Fahrt. Dies bedeutet, dass der Roboter die erste programmierte Position nach dem Start mit programmierter (nicht reduzierter) Geschwindigkeit anfährt und dort nicht stoppt.

Voraussetzung
- Betriebsart T1 oder T2
- Die Ein-/Ausgänge für Automatik Extern sind konfiguriert.
- Das Programm CELL.SRC ist konfiguriert.

Vorgehensweise
1. Im Navigator das Programm CELL.SRC anwählen. (Befindet sich im Ordner "R1".)
2. Programm-Override auf 100% einstellen. (Dies ist die empfohlene Einstellung. Bei Bedarf kann ein anderer Wert eingestellt werden.)
3. SAK-Fahrt durchführen:
   Zustimmungsschalter drücken und halten. Dann Start-Taste drücken und halten, bis im Meldungsfenster "SAK erreicht" angezeigt wird.

⚠ VORSICHT Die SAK-Fahrt erfolgt als LIN- oder PTP-Bewegung von der Istposition zur Zielposition. Die Geschwindigkeit ist automatisch reduziert. Der Verlauf der Bewegung ist nicht sicher vorhersehbar. Die Bewegung während der SAK-Fahrt beobachten, damit der Roboter rechtzeitig gestoppt werden kann, falls sich eine Kollision abzeichnet.

4. Betriebsart "Automatik Extern" wählen.
5. Das Programm von einer übergeordneten Steuerung (SPS) aus starten.

Um ein im Automatik-Betrieb gestartetes Programm zu stoppen, die STOP-Taste drücken.

8.12 Rückwärtsfahren über die Start-Rückwärts-Taste

Die folgenden Ausführungen gelten für das Rückwärtsfahren über die Start-Rückwärts-Taste. Sie gelten nicht für andere Rückwärts-Funktionalitäten, z. B. für Rückwärtsbewegungen im Rahmen von Fehlerstrategien in Technologiepaketen.

8.12.1 Bewegungen rückwärts abfahren

Beschreibung

Das Rückwärtsfahren wird häufig verwendet, wenn eine Abfolge von Bewegungen optimiert werden soll und zu diesem Zweck einzelne Punkte umgeteacht werden sollen. Der Benutzer fährt die Bahn rückwärts ab, bis der zu korrigierende Punkt erreicht ist. Wenn er den Punkt umgeteacht hat, fährt er bei Bedarf weiter rückwärts, um weitere Punkte zu korrigieren.

Beim Rückwärtsfahren gilt automatisch die Programmablaufart #BSTEP.

Beim Rückwärtsfahren ist es nicht möglich, zu überschleifen oder zu pendeln. Wenn also vorwärts Punkte überschliffen wurden oder gependelt wurde, unterscheidet sich die Rückwärtsbahn von der Vorwärtsbahn. Dadurch kann es nach dem Rückwärts-Start sein, dass der Roboter erst eine SAK-Fahrt durchführen muss, obwohl er die Bahn vorwärts gar nicht verlassen hatte.

⚠ VORSICHT Die SAK-Fahrt erfolgt als LIN- oder PTP-Bewegung von der Istposition zur Zielposition. Die Geschwindigkeit ist automatisch reduziert. Der Verlauf der Bewegung ist nicht sicher vorhersehbar. Die Bewegung während der SAK-Fahrt beobachten, damit der Roboter rechtzeitig gestoppt werden kann, falls sich eine Kollision abzeichnet.

Voraussetzung

- Programm ist angewählt.
- Die Bewegungen, die rückwärts gefahren werden sollen, wurden vorwärts abgefahren.
- Betriebsart T1 oder T2

Vorgehensweise

1. Zustimmungsschalter gedrückt halten und warten, bis die Statusleiste "Antriebe bereit" anzeigt:

2. Die Start-Rückwärts-Taste drücken und halten.
   - Wenn sich der Roboter bereits auf der Rückwärtsbahn befindet, fährt er nun rückwärts.
   - Wenn sich der Roboter nicht auf der Rückwärtsbahn befindet, fährt er nun dorthin. Wenn im Meldungsfenster "*SAK erreicht*" angezeigt wird, hat er die Bahn erreicht. Der Roboter bleibt stehen.

     Die Start-Rückwärts-Taste erneut drücken. Der Roboter fährt nun rückwärts.
3. Die Start-Rückwärts-Taste für jeden Bewegungssatz erneut drücken.

8.12.2 Funktionsweise und Eigenschaften des Rückwärtsfahrens

Funktionsweise

Beim Vorwärtsfahren speichert die Robotersteuerungen die abgearbeiteten Bewegungen in einem Ringpuffer. Beim Rückwärtsfahren werden die Bewegungen auf Basis der gespeicherten Informationen abgefahren.

Kein Rückwärtsfahren möglich nach Pufferlöschung:

In den folgenden Fällen wird der Inhalt des Puffers gelöscht. Danach ist das Rückwärtsfahren erst wieder möglich, wenn wieder Bewegungen vorwärts abgefahren wurden.

- Programm wird zurückgesetzt.
- Programm wird abgewählt.
- In das Programm werden Zeilen eingefügt oder gelöscht.
- KRL-Anweisung RESUME
- Eine Satzanwahl auf eine andere Bewegung als die aktuelle Bewegung.

  Ohne Einschränkung möglich ist jedoch eine Satzanwahl auf einen beliebigen Segmentpunkt innerhalb des aktuellen Spline-Blocks. Dies gilt als Satzanwahl auf die aktuelle Bewegung, da die Robotersteuerung den Spline-Block als 1 Bewegung plant und ausführt.

Die Robotersteuerung löscht den Puffer, ohne eine Meldung dazu auszugeben.

Eigenschaften

- Das Rückwärtsfahren ist nur möglich in T1 und T2.
- Beim Rückwärtsfahren werden ausschließlich Bewegungen ausgeführt, keine Kontrollstrukturen und Steueranweisungen.
- Ausgänge, Flags und Cycflag werden beim Vorwärtsfahren nicht aufgezeichnet. Beim Rückwärtsfahren werden deshalb ihre vorigen Zustände nicht wiederhergestellt.
- Die Geschwindigkeit ist wie beim Vorwärtsfahren.

  Es ist möglich, dass in T2 beim Rückwärtsfahren Überwachungen ansprechen, die in Vorwärtsrichtung nicht ansprechen. In diesem Fall muss der Programm-Override reduziert werden.

Das Rückwärtsfahren kann deaktiviert werden. Darüber hinaus stehen weitere Konfigurationsmöglichkeiten zur Verfügung.

(>>> 6.16 "Rückwärtsfahren konfigurieren" Seite 194)

Über `DELETE_BACKWARD_BUFFER()` kann man das Rückwärtsfahren für bestimmte Bewegungen gezielt verhindern.

(>>> 11.14.1 "DELETE_BACKWARD_BUFFER()" Seite 457)

Momenten-/Kraftbetrieb, Vectormove

Für Bewegungen mit Momenten- oder Kraftbetrieb oder Vectormove gilt:

- Bei herkömmlichen Bewegungen ist das Rückwärtsfahren möglich, jedoch wird der Momenten- oder Kraftbetrieb oder Vectormove dabei automatisch ausgeschaltet.
- Spline-Bewegungen können nicht rückwärts abgefahren werden.

HINWEIS Bei komplexeren Applikationen mit Momentenbetrieb (z.B. Pressenausstoßen) und/oder Vectormove ist allgemein davon abzuraten, rückwärts zu fahren, da die zugrunde liegenden Prozesse in der Regel nicht umkehrbar sind.
Es wird empfohlen, in solchen Fällen das Rückwärtsfahren über DELETE_BACKWARD_BUFFER() zu verhindern. Sachschäden können sonst die Folge sein.

8.12.2.1 Verhalten bei Unterprogrammen

- Bewegungen, die in einem Interrupt-Programm vorwärts gefahren werden, werden nicht aufgezeichnet. Sie können deshalb auch nicht rückwärts abgefahren werden.
- Wenn ein Unterprogramm beim Vorwärtsfahren vollständig durchfahren wurde, kann es nicht rückwärts durchfahren werden.

- Wenn die Vorwärts-Bewegung in einem Unterprogramm angehalten wurde, hängt das Verhalten von der Position des Vorlaufzeigers ab:

Position Vorlaufzeiger	Verhalten
Vorlaufzeiger steht innerhalb des Unterprogramms.	Rückwärtsfahren ist möglich.
Vorlaufzeiger hat Unterprogramm bereits verlassen.	Rückwärtsfahren ist nicht möglich. Vorbeugung: Vor dem END des Unterprogramms einen Vorlaufstop auslösen, z. B. mit WAIT SEC 0. Allerdings ist an dieser Stelle dann kein Überschleifen mehr möglich. Oder $ADVANCE auf "1" setzen. Verhindert die Fehlermeldung nicht in jedem Fall, verringert aber die Wahrscheinlichkeit. Ein Überschleifen ist weiterhin möglich.

8.12.2.2 Verhalten bei Überschleifen

Beschreibung

Beim Rückwärtsfahren ist es nicht möglich, zu überschleifen. Wenn also vorwärts Punkte überschliffen wurden, unterscheidet sich die Rückwärtsbahn von der Vorwärtsbahn. Dadurch kann es nach dem Rückwärts-Start sein, dass der Roboter erst eine SAK-Fahrt zur Rückwärtsbahn durchführen muss, obwohl er die Bahn vorwärts gar nicht verlassen hatte.

Beispiel 1

Rückwärts-Start außerhalb eines Überschleifbereichs:

Die Start-Rückwärts-Taste wird gedrückt, während sich der Roboter auf der Bahn befindet, jedoch nicht in einem Überschleifbereich. Der Roboter fährt nun auf der Bahn rückwärts zum Zielpunkt der vorigen Bewegung.

P_{BACK} = Position des Roboters zum Zeitpunkt, an dem die Start-Rückwärts-Taste gedrückt wird

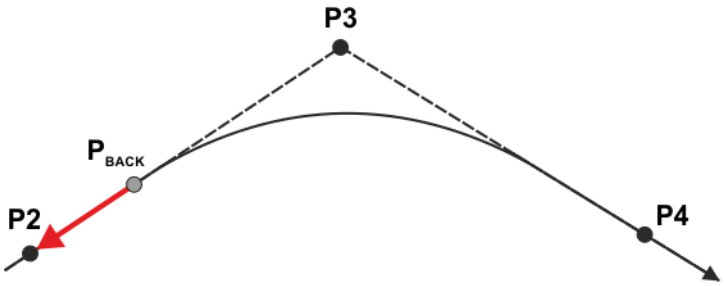


Abb. 8-14: Fall 1: Rückwärts-Start außerhalb eines Überschleifbereichs

Wenn der Zielpunkt der vorigen Bewegung überschliffen ist, wird er trotzdem genau angefahren.

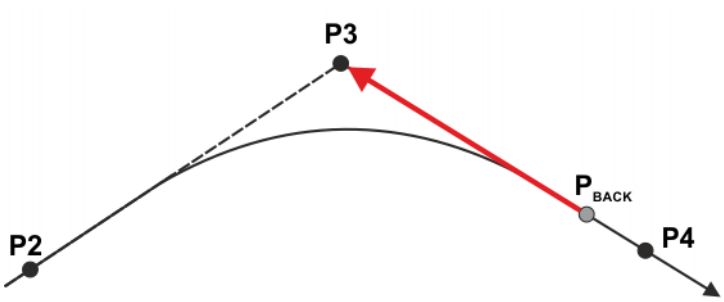


Abb. 8-15: Fall 2: Rückwärts-Start außerhalb eines Überschleifbereichs

Beispiel 2

Rückwärts-Start im Überschleifbereich:

Die Start-Rückwärts-Taste wird gedrückt, während sich der Roboter in einem Überschleifbereich befindet. Der Roboter führt nun eine SAK-Fahrt an den Beginn des Überschleifbereichs durch und stoppt dort. Wenn die Start-Rückwärts-Taste nun noch einmal gedrückt wird, beginnt das eigentliche Rückwärtsfahren, d. h. der Roboter fährt auf der Bahn rückwärts zum Zielpunkt der vorigen Bewegung.

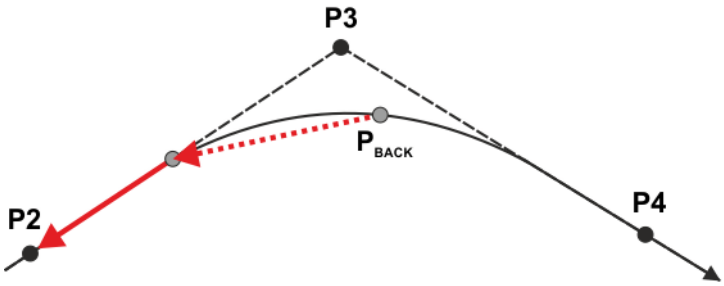


Abb. 8-16: Rückwärts-Start im Überschleifbereich

8.12.2.3 Verhalten bei Pendelbewegungen

Beschreibung

Beim Rückwärtsfahren ist kein Pendeln möglich. Wenn also vorwärts gependelt wurde, unterscheidet sich die Rückwärtsbahn von der Vorwärtsbahn. Nach dem Rückwärts-Start muss der Roboter deshalb erst eine SAK-Fahrt zur Rückwärtsbahn durchführen, obwohl er die Bahn vorwärts nicht verlassen hatte.

Beispiel

Rückwärts-Start auf Pendelbahn:

Die Start-Rückwärts-Taste wird gedrückt, während sich der Roboter pendelt. Der Roboter führt nun eine SAK-Fahrt zur geteachten Bahn durch und stoppt dort. Wenn die Start-Rückwärts-Taste nun noch einmal gedrückt wird, beginnt das eigentliche Rückwärtsfahren, d. h. der Roboter fährt auf der Bahn rückwärts zum Zielpunkt der vorigen Bewegung.

P_{BACK} = Position des Roboters zum Zeitpunkt, an dem die Start-Rückwärts-Taste gedrückt wird

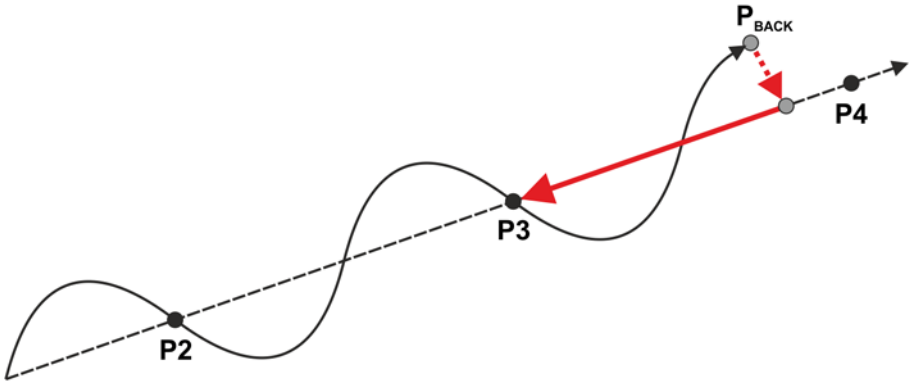


Abb. 8-17: Rückwärts-Start auf Pendelbahn

8.12.2.4 Wechsel von rückwärts auf vorwärts

Voraussetzung

Nach dem Rückwartsfahren wieder vorwärts zu fahren, ist nur unter folgenden Voraussetzungen möglich:

- Auf die Programmzeile, an der der Rückwärts-Satzzeiger gerade steht, ist eine Satzanwahl möglich.
- Wenn die erste Bewegung, die wieder vorwärts gefahren werden soll, eine herkömmliche Bewegung ist, muss sie vollständig programmiert sein.

  Es ist also z. B. nicht möglich, von rückwärts auf vorwärts zu wechseln, wenn die erste Bewegung ein PTP_REL ist.

  Für Spline-Bewegungen gilt diese Einschränkung bis auf wenige Ausnahmen nicht.

Verhalten

Wenn man nach dem Rückwärtsfahren zum ersten Mal die Start-Vorwärts-Taste drückt, ist das Verhalten folgendermaßen:

- Wenn SAK vorhanden ist, wird die zuletzt vorwärts gefahrene Programmablaufart automatisch wiederhergestellt und der Roboter fährt auf der Bahn vorwärts.
- Wenn keine SAK vorhanden ist, wird eine SAK-Fahrt ausgeführt. Währenddessen bleibt die Programmablaufart noch auf #BSTEP. Nach der SAK-Fahrt stoppt der Roboter. Nun muss nochmal die Start-Vorwärts-Taste gedrückt werden. Die zuletzt vorwärts gefahrene Programmablaufart wird automatisch wiederhergestellt und der Roboter fährt nun auf der Bahn vorwärts.

Wenn man innerhalb einer Kontrollstruktur von Rückwärts auf Vorwärts wechselt, so fährt der Roboter zunächst vorwärts bis zum Ende der Kontrollstruktur. Danach stoppt er mit der Meldung *Kontrollstruktur nächster Satz {Satznummer}*. Die Satznummer gibt den ersten Satz nach der Kontrollstruktur an.

8.12.3 Systemvariablen mit veränderter Bedeutung

Die Bedeutung einiger Systemvariablen, die zur Funktionalität "Rückwärtsfahren" gehören, hat sich ab der V8.3.6 verändert. Teilweise haben sie keine Auswirkung mehr, existieren aber noch aus Gründen der Kompatibilität zu älteren Versionen.

Die Systemvariablen, die mit "$VW_…" beginnen, existieren nicht nur in der VSS, sondern auch in der KSS.

Bezeichnung	Bedeutung in V8.3.6 und höher
$BWD_INFO	Beinhaltet die aktuelle Konfiguration zum Rückwärtsfahren als Bitmap. Zahlreiche Technologiepakete lesen diese Variable aus. Die Variable ist nur lesbar.
$BWDSTART	Die Variable existiert nur noch aus Kompatibilitätsgründen. Sie kann auf TRUE oder FALSE gesetzt werden, dies hat jedoch keine Auswirkung.
$VW_BACKWARD	Entspricht dem Attribut ENABLE des Konfigurationselements BACKWARD_STEP. Die Variable ist nur lesbar.
$VW_CYCFLAG	Liefert immer den Wert 0. Die Variable existiert nur noch aus Kompatibilitätsgründen und ist nur lesbar.
$VW_MOVEMENT	Entspricht dem Attribut MOVEMENTS des Konfigurationselements BACKWARD_STEP. Die Variable ist nur lesbar.
$VW_RETRACE_AMF	Liefert immer den Wert FALSE. Die Variable existiert nur noch aus Kompatibilitätsgründen und ist nur lesbar.
$LOAD_BWINI	Liefert immer den Wert FALSE. Die Variable existiert nur noch aus Kompatibilitätsgründen und ist nur lesbar.

9 Grundlagen der Bewegungsprogrammierung

9.1 Bewegungsarten Übersicht

Folgende Bewegungsarten können programmiert werden:

- **Point-to-Point-Bewegung (PTP)**
  (>>> 9.2 "Bewegungsart PTP" Seite 287)
- **Linear-Bewegung (LIN)**
  (>>> 9.3 "Bewegungsart LIN" Seite 288)
- **Circular-Bewegung (CIRC)**
  (>>> 9.4 "Bewegungsart CIRC" Seite 288)
- **Spline-Bewegungen**
  Spline-Bewegungen haben eine Reihe von Vorteilen gegenüber den herkömmlichen PTP-, LIN- und CIRC-Bewegungen.
  (>>> 9.7 "Bewegungsart Spline" Seite 293)

Der Startpunkt einer Bewegung ist immer der Zielpunkt der vorhergehenden Bewegung.

Folgende Bewegungen werden unter dem Begriff "CP-Bewegungen" ("Continuous Path") zusammengefasst:

- LIN, CIRC, CP-Spline-Blöcke, SLIN, SCIRC

9.2 Bewegungsart PTP

Der Roboter führt den TCP entlang der schnellsten Bahn zum Zielpunkt. Die schnellste Bahn ist in der Regel nicht die kürzeste Bahn und somit keine Gerade. Da sich die Roboterachsen rotatorisch bewegen, können geschwungene Bahnen schneller ausgeführt werden als gerade Bahnen.

Der exakte Verlauf der Bewegung ist nicht vorhersehbar.

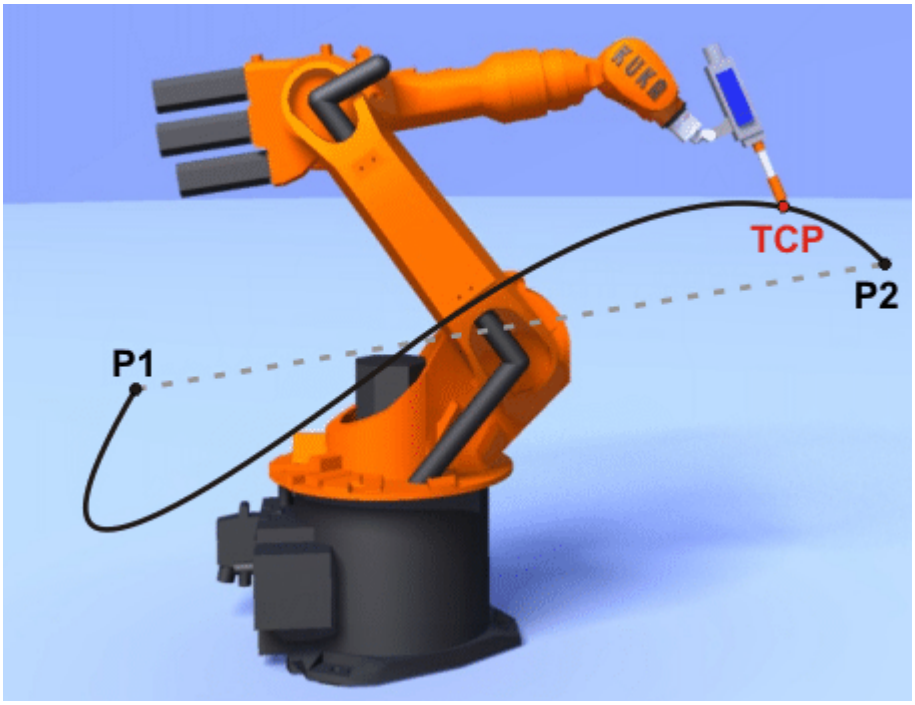


Abb. 9-1: PTP-Bewegung

9.3 Bewegungsart LIN

Der Roboter führt den TCP mit der definierten Geschwindigkeit entlang einer Geraden zum Zielpunkt.

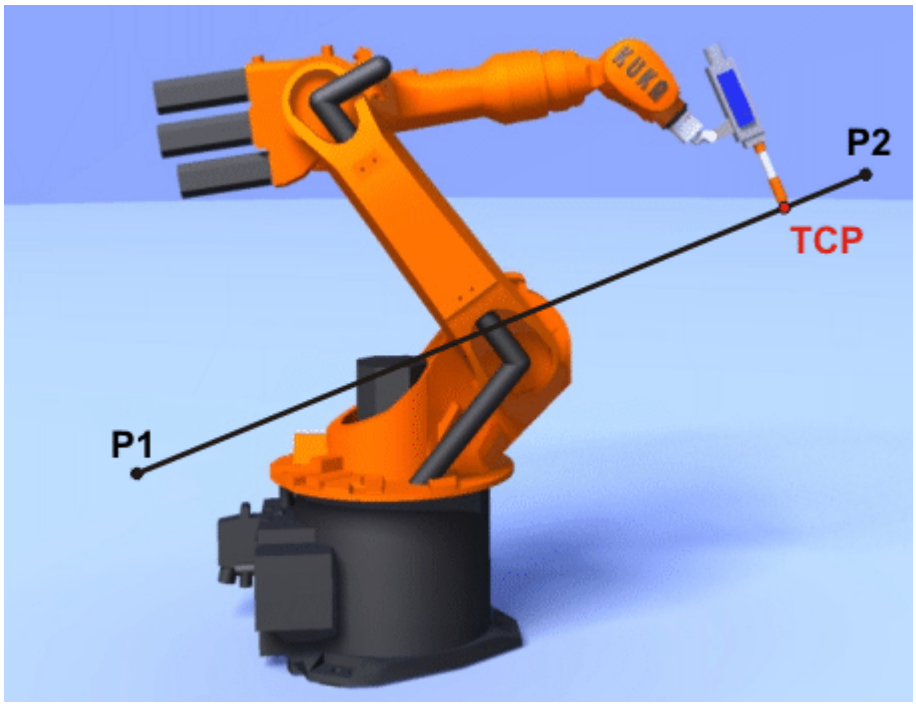


Abb. 9-2: LIN-Bewegung

9.4 Bewegungsart CIRC

Der Roboter führt den TCP mit der definierten Geschwindigkeit entlang einer Kreisbahn zum Zielpunkt. Die Kreisbahn ist definiert durch Startpunkt, Hilfspunkt und Zielpunkt.

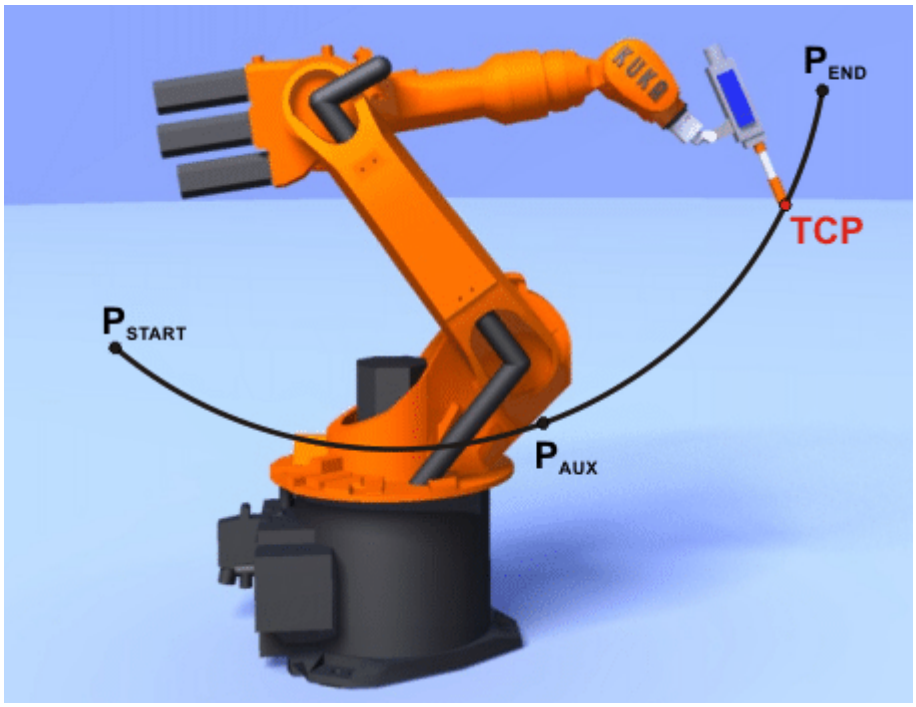


Abb. 9-3: CIRC-Bewegung

9.5 Überschleifen

Überschleifen bedeutet: Der programmierte Punkt wird nicht genau angefahren. Überschleifen ist eine Option, die bei der Bewegungsprogrammierung ausgewählt werden kann.

Überschleifen ist nicht möglich, wenn nach der Bewegungsanweisung eine Anweisung folgt, die einen Vorlaufstopp auslöst.

Überschleifen bei einer PTP-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt eine schnellere Bahn. Beim Programmieren der Bewegung wird die Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn abweichen darf.

Der Bahnverlauf bei einer überschliffenen PTP-Bewegung ist nicht vorhersehbar. Es ist auch nicht vorhersehbar, auf welcher Seite des überschliffenen Punkts die Bahn verlaufen wird.

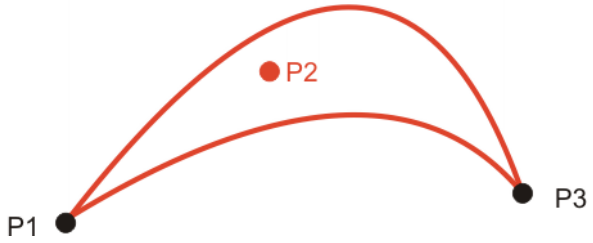


Abb. 9-4: PTP-Bewegung, P2 ist überschliffen

Überschleifen bei einer LIN-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt eine kürzere Bahn. Beim Programmieren der Bewegung wird die Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn abweichen darf.

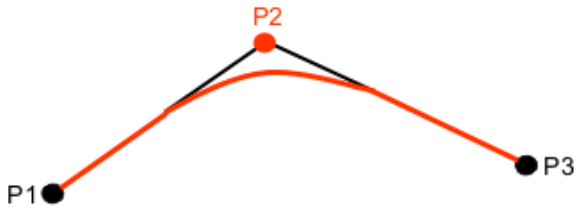


Abb. 9-5: LIN-Bewegung, P2 ist überschliffen

Überschleifen bei einer CIRC-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt eine kürzere Bahn. Beim Programmieren der Bewegung wird die Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn abweichen darf.

Der Hilfspunkt wird genau durchfahren.

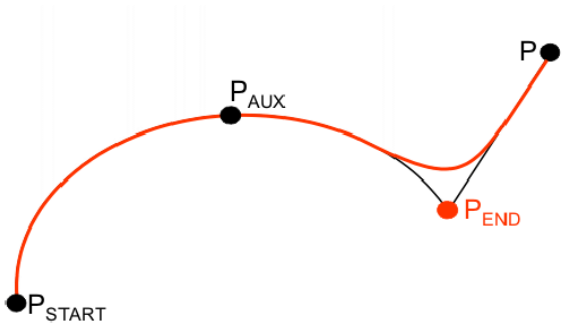


Abb. 9-6: CIRC-Bewegung, P_{END} ist überschliffen

9.6 Orientierungsführung LIN, CIRC

Beschreibung

Der TCP kann am Start- und am Zielpunkt einer Bewegung unterschiedliche Orientierungen haben. Die Start-Orientierung kann auf mehrere Arten in die Ziel-Orientierung übergehen. Beim Programmieren einer CP-Bewegung muss eine Art ausgewählt werden.

Die Orientierungsführung für LIN- und CIRC-Bewegungen wird folgendermaßen festgelegt:

- Im Optionsfenster **Bewegungsparameter**
  (>>> 10.2.8 "Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)" Seite 321)
- Oder über die Systemvariable $ORI_TYPE

LIN-Bewegung

Orientierungsführung	Beschreibung
■ Optionsfenster: **Konstante Orientierung** ■ $ORI_TYPE = #CONSTANT	Die Orientierung des TCP bleibt während der Bewegung konstant. Für den Zielpunkt wird die programmierte Orientierung ignoriert und die des Startpunkts beibehalten.
■ Optionsfenster: **Standard** ■ $ORI_TYPE = #VAR	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich. **Hinweis**: Wenn der Roboter mit **Standard** in eine Handachsen-Singularität gerät, stattdessen **Hand PTP** verwenden.
■ Optionsfenster: **Hand PTP** ■ $ORI_TYPE = #JOINT	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich. Dies geschieht durch lineare Überführung (achsspezifisches Verfahren) der Handachswinkel. **Hinweis**: **Hand PTP** dann verwenden, wenn der Roboter mit **Standard** in eine Handachsen-Singularität gerät. Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich, jedoch nicht ganz gleichmäßig. **Hand PTP** ist deshalb nicht geeignet, wenn ein bestimmter Verlauf der Orientierung exakt gehalten muss, z. B. beim Laserschweißen.

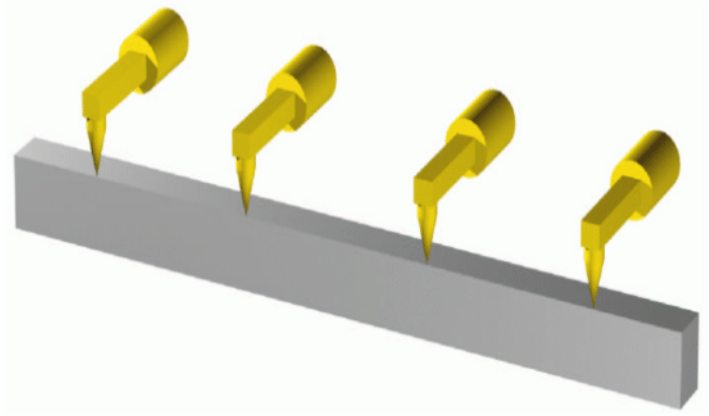


Abb. 9-7: Konstante Orientierung

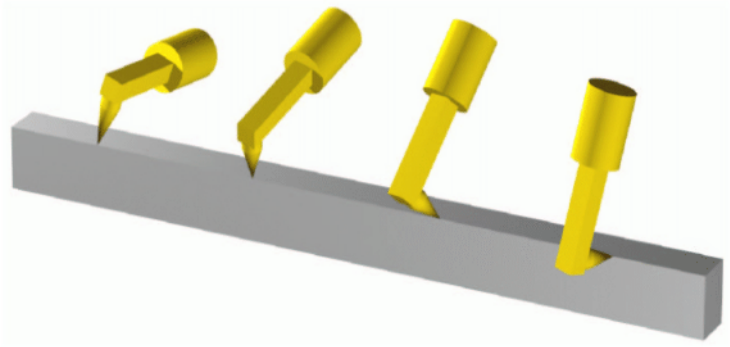


Abb. 9-8: Standard oder Hand PTP

CIRC-Bewegung

Bei CIRC-Bewegungen berücksichtigt die Robotersteuerung nur die programmierte Orientierung des Zielpunkts. Die programmierte Orientierung des Hilfspunkts wird ignoriert.

Für CIRC-Bewegungen stehen die gleichen Orientierungsführungen zur Auswahl wie für LIN-Bewegungen.

Zusätzlich kann für CIRC-Bewegungen festgelegt werden, ob die Orientierungsführung basisbezogen oder bahnbezogen sein soll. Dies wird über die Systemvariable $CIRC_TYPE festgelegt.

Orientierungsführung	Beschreibung
$CIRC_TYPE = #BASE	Basisbezogene Orientierungsführung während der Kreisbewegung
$CIRC_TYPE = #PATH	Bahnbezogene Orientierungsführung während der Kreisbewegung

$CIRC_TYPE ist bedeutungslos, wenn $ORI_TYPE = #JOINT.

(>>> 9.6.1 "Kombinationen von $ORI_TYPE und $CIRC_TYPE" Seite 291)

9.6.1 Kombinationen von $ORI_TYPE und $CIRC_TYPE

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #PATH :

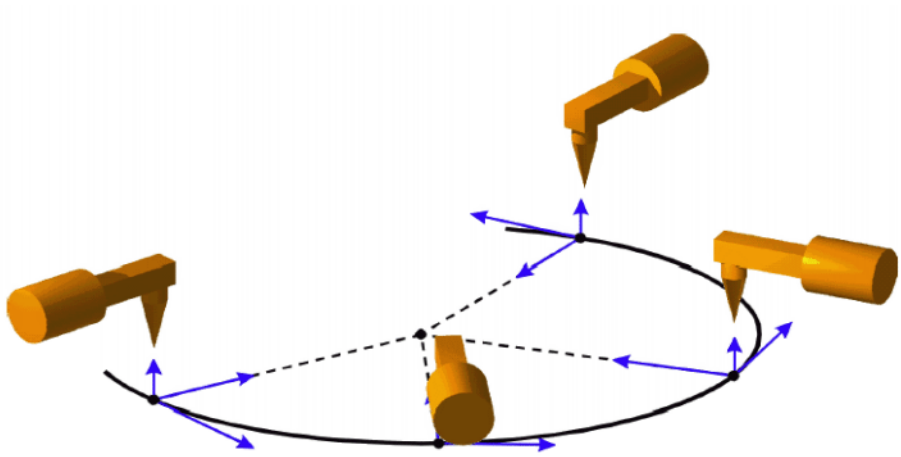


Abb. 9-9: Konstante Orientierung, bahnbezogen

$ORI_TYPE = #VAR, $CIRC_TYPE = #PATH :

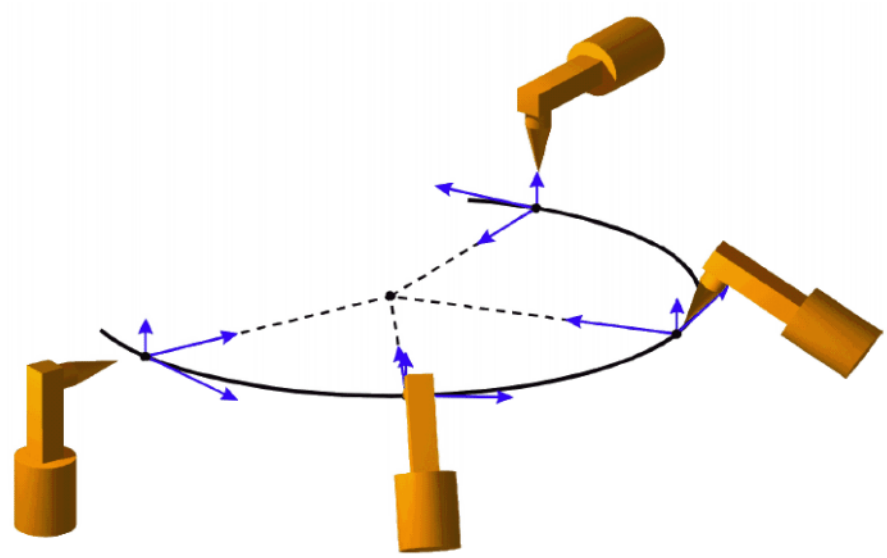


Abb. 9-10: Variable Orientierung, bahnbezogen

$ORI_TYPE = #CONSTANT, $CIRC_TYPE = #BASE :

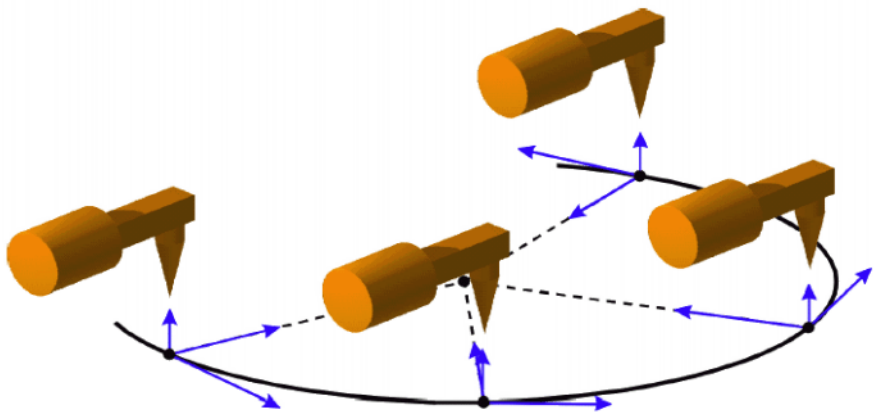


Abb. 9-11: Konstante Orientierung, basisbezogen

$ORI_TYPE = #VAR, $CIRC_TYPE = #BASE :

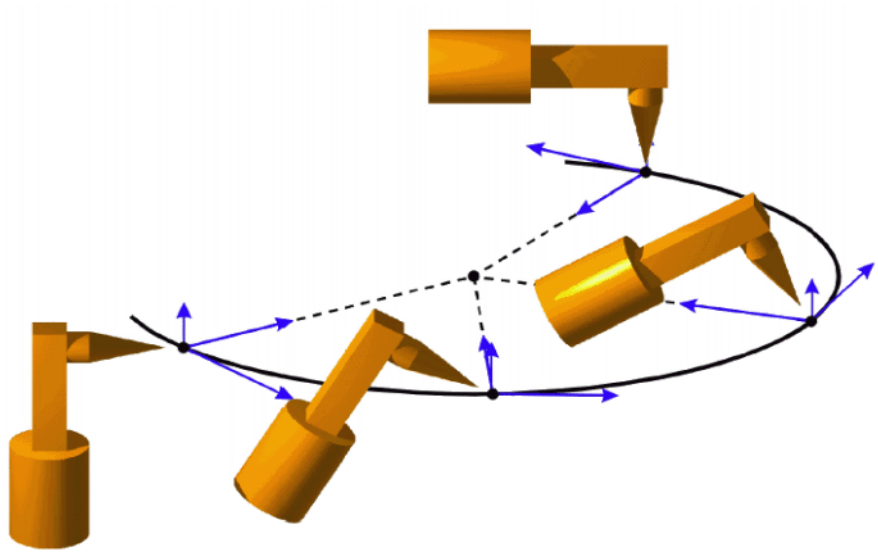


Abb. 9-12: Variable Orientierung, basisbezogen

9.7 Bewegungsart Spline

Der Spline ist eine Bewegungsart, die besonders für komplexe geschwungene Bahnen geeignet ist. Solche Bahnen können grundsätzlich auch mit überschliffenen LIN- und CIRC-Bewegungen erzeugt werden, der Spline hat jedoch Vorteile.

Die vielseitigste Spline-Bewegung ist der Spline-Block. Mit einem Spline-Block fasst man mehrere Bewegungen zu einer Gesamtbewegung zusammen. Der Spline-Block wird von der Robotersteuerung als 1 Bewegungssatz geplant und ausgeführt.

Die Bewegungen, die in einem Spline-Block stehen können, heißen Spline-Segmente. Sie werden einzeln geteacht.

- Ein CP-Spline-Block kann SPL-, SLIN- und SCIRC-Segmente enthalten.
- Ein PTP-Spline-Block kann SPTP-Segmente enthalten.

Außer Spline-Blöcken können auch Spline-Einzelbewegungen programmiert werden: SLIN, SCIRC und SPTP.

Vorteile Spline-Block

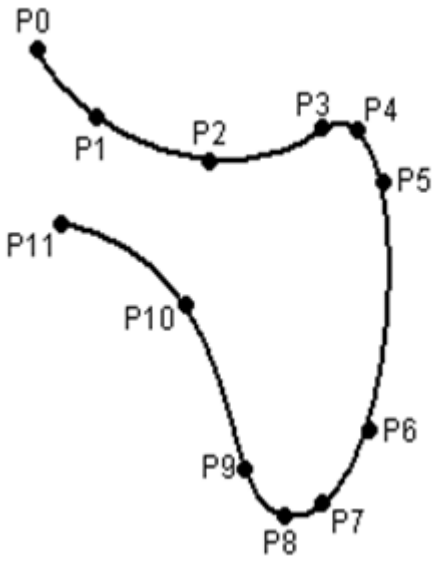


Abb. 9-13: Geschwungene Bahn mit Spline-Block

- Die Bahn wird definiert über Punkte, die auf der Bahn liegen. Die gewünschte Bahn kann einfach erzeugt werden.
- Die programmierte Geschwindigkeit wird besser gehalten als bei den herkömmlichen Bewegungsarten. Nur in wenigen Fällen kommt es zu einer Geschwindigkeitsreduzierung.

  (>>> 9.7.1 "Geschwindigkeitsprofil bei Spline-Bewegungen" Seite 295)

  In CP-Spline-Blöcken können darüber hinaus spezielle Konstantfahrbereiche definiert werden.
- Der Bahnverlauf ist immer gleich, unabhängig von Override, Geschwindigkeit oder Beschleunigung.
- Kreise und enge Radien werden mit hoher Präzision gefahren.

Nachteile LIN/ CIRC

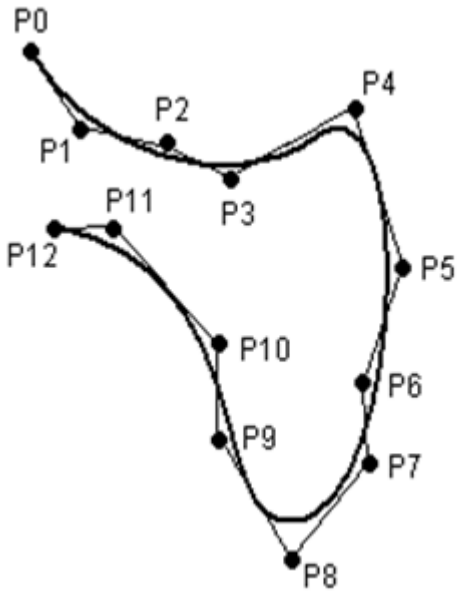


Abb. 9-14: Geschwungene Bahn mit überschliffenen LIN-Bewegungen

- Die Bahn wird definiert über überschliffene Punkte, die nicht auf der Bahn liegen. Die Überschleifbereiche sind schwer vorhersehbar. Es ist aufwendig, die gewünschte Bahn zu erzeugen.

- In zahlreichen Fällen kommt es zu schwer vorhersehbaren Geschwindigkeitsreduzierungen, z. B. in den Überschleifbereichen und bei nahe beieinanderliegenden Punkten.
- Der Bahnverlauf ändert sich, wenn das Überschleifen nicht möglich ist, z. B. aus Zeitgründen.
- Der Bahnverlauf ändert sich abhängig von Override, Geschwindigkeit oder Beschleunigung.

9.7.1 Geschwindigkeitsprofil bei Spline-Bewegungen

Die Bahn verläuft immer gleich, unabhängig von Override, Geschwindigkeit oder Beschleunigung.

Die Robotersteuerung berücksichtigt bereits bei der Planung die physikalischen Grenzen des Roboters. Der Roboter bewegt sich im Rahmen der programmierten Geschwindigkeit so schnell wie möglich, d. h. so, wie es seine physikalischen Grenzen erlauben. Dies ist ein Vorteil gegenüber den herkömmlichen LIN- und CIRC-Bewegungen, bei denen die physikalischen Grenzen bei der Planung nicht berücksichtigt werden. Sie wirken sich dort erst während der Bewegungsausführung aus und lösen gegebenenfalls Stopps aus.

Absenkung der Geschwindigkeit

Zu den Fällen, in denen die programmierte Geschwindigkeit unterschritten werden muss, gehören vor allem:

- Ausgeprägte Ecken
- Große Umorientierungen
- Große Bewegungen der Zusatzachsen
- In der Nähe von Singularitäten

Eine Absenkung der Geschwindigkeit aufgrund von großen Umorientierungen kann man bei Spline-Segmenten vermeiden, indem man die Orientierungsführung **Ohne Orientierung** auswählt.

Eine Absenkung der Geschwindigkeit aufgrund großer Zusatzachs-Bewegungen kann man bei Spline-Segmenten über $EX_AX_IGNORE vermeiden.

Absenkung der Geschwindigkeit auf 0

Dies ist der Fall bei:

- Aufeinanderfolgenden Punkten mit gleichen Koordinaten
- Aufeinanderfolgenden SLIN- und/oder SCIRC-Segmenten. Ursache: Unstetiger Verlauf der Geschwindigkeitsrichtung.

  Bei SLIN-SCIRC-Übergängen wird die Geschwindigkeit auch dann 0, wenn die Gerade tangential in den Kreis übergeht, da der Kreis im Gegensatz zur Geraden gekrümmt ist.

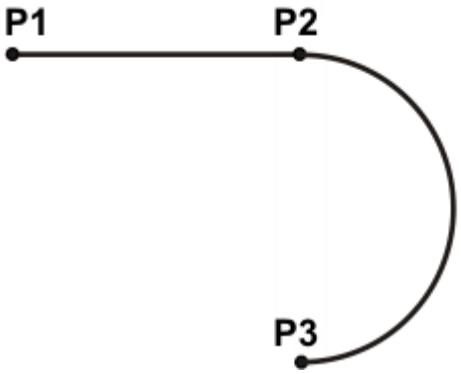


Abb. 9-15: Genauhalt in P2

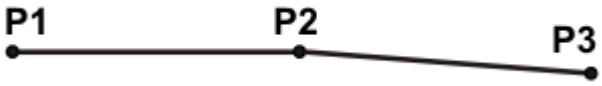


Abb. 9-16: Genauhalt in P2

Ausnahmen:

- Wenn SLIN-Segmente aufeinanderfolgen, die eine Gerade ergeben und bei denen sich die Orientierungen gleichmäßig ändern, wird die Geschwindigkeit nicht reduziert.

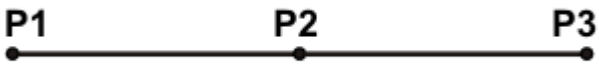


Abb. 9-17: P2 wird ohne Genauhalt durchfahren.

- Bei einem SCIRC-SCIRC-Übergang wird die Geschwindigkeit nicht reduziert, wenn beide Kreise den gleichen Mittelpunkt und den gleichen Radius haben, und wenn sich die Orientierungen gleichmäßig ändern. (Schwierig zu teachen, deshalb Punkte berechnen und programmieren.)

Kreise mit gleichem Mittelpunkt und gleichem Radius werden manchmal programmiert, um Kreise ≥ 360 ° zu erhalten. Eine einfachere Möglichkeit ist es, einen Kreiswinkel zu programmieren.

9.7.2 Satzanwahl bei Spline-Bewegungen

Spline-Block

Auf die Segmente eines Spline-Blocks kann eine Satzanwahl ausgeführt werden.

- CP-Spline-Block:

  Die SAK-Fahrt wird als herkömmliche LIN-Bewegung ausgeführt. Dies wird durch eine Meldung angekündigt, die quittiert werden muss.
- PTP-Spline-Block:

  Die SAK-Fahrt wird als herkömmliche PTP-Bewegung ausgeführt. Dies wird nicht durch eine Meldung angekündigt.

Nach einer Satzanwahl verläuft die Bahn in der Regel genauso, wie wenn der Spline im normalen Programmablauf abgefahren würde.

Ausnahmen sind möglich, falls der Spline vor der Satzanwahl noch nie abgefahren wurde und wenn in diesem Fall eine Satzanwahl auf den Anfang des Spline-Blocks durchgeführt wird:

Der Startpunkt der Spline-Bewegung ist der letzte Punkt vor dem Spline-Block, d. h. der Startpunkt liegt außerhalb des Blocks. Die Robotersteuerung speichert den Startpunkt beim normalen Abfahren eines Splines. Dadurch ist er bekannt, wenn zu einem späteren Zeitpunkt eine Satzanwahl durchgeführt wird. Wenn der Spline-Block jedoch noch nie abgefahren wurde, ist der Startpunkt nicht bekannt.

Wenn nach der SAK-Fahrt die Start-Taste gedrückt wird, wird die veränderte Bahn mit einer Meldung angekündigt, die quittiert werden muss.

Beispiel: Veränderte Bahn bei Satzanwahl auf P1

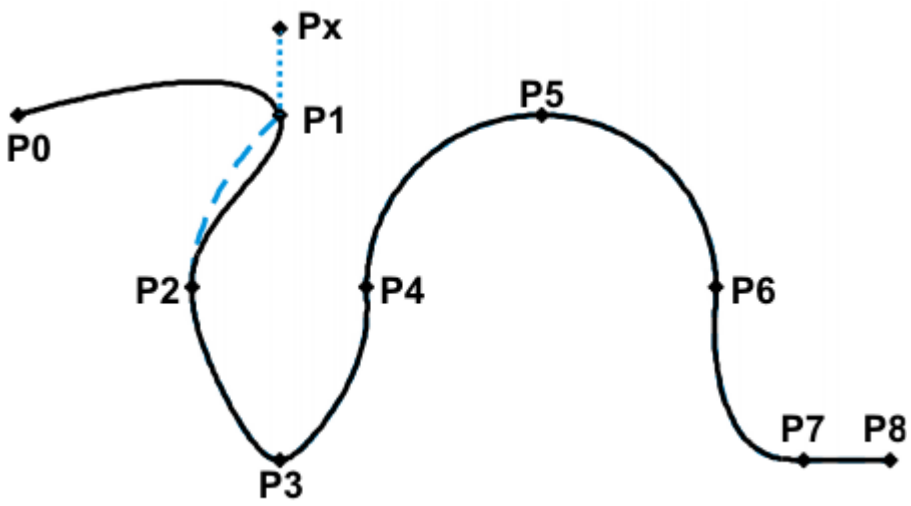


Abb. 9-18: Beispiel: Veränderte Bahn bei Satzanwahl auf P1

```
1  PTP P0
2  SPLINE
3   SPL P1
4   SPL P2
5   SPL P3
6   SPL P4
7   SCIRC P5, P6
8   SPL P7
9   SLIN P8
10 ENDSPLINE
```

Zeile	Beschreibung
2	Kopfzeile/Beginn des CP-Spline-Blocks
3 … 9	Spline-Segmente
10	Ende des CP-Spline-Blocks

SCIRC

Bei Satzanwahl auf ein SCIRC-Segment, für das ein Kreiswinkel programmiert ist, wird der Zielpunkt unter Berücksichtigung des Kreiswinkels angefahren, vorausgesetzt, dass die Robotersteuerung den Startpunkt kennt.

Wenn die Robotersteuerung den Startpunkt nicht kennt, wird der programmierte Zielpunkt angefahren. In diesem Fall zeigt eine Meldung an, dass der Kreiswinkel nicht berücksichtigt wird.

Bei einer Satzanwahl auf eine SCIRC-Einzelbewegung wird der Kreiswinkel nie berücksichtigt.

9.7.3 Änderungen an Spline-Blöcken

Beschreibung

- Änderung der Punktposition:

  Wenn ein Punkt innerhalb eines Spline-Blocks verschoben wird, ändert sich die Bahn maximal in den 2 Segmenten vor diesem Punkt und in den 2 Segmenten danach.

  Kleine Punktverschiebungen ergeben in der Regel kleine Bahnänderungen. Wenn jedoch sehr lange und sehr kurze Segmente aufeinanderfolgen, können kleine Änderungen sehr große Auswirkungen haben.
- Änderung des Segmenttyps:

  Wenn ein SPL-Segment in ein SLIN-Segment geändert wird oder umgekehrt, ändert sich die Bahn im vorhergehenden Segment und im folgenden Segment.

Beispiel 1

Ursprüngliche Bahn:

```
PTP P0
SPLINE
 SPL P1
 SPL P2
 SPL P3
 SPL P4
 SCIRC P5, P6
 SPL P7
 SLIN P8
ENDSPLINE
```

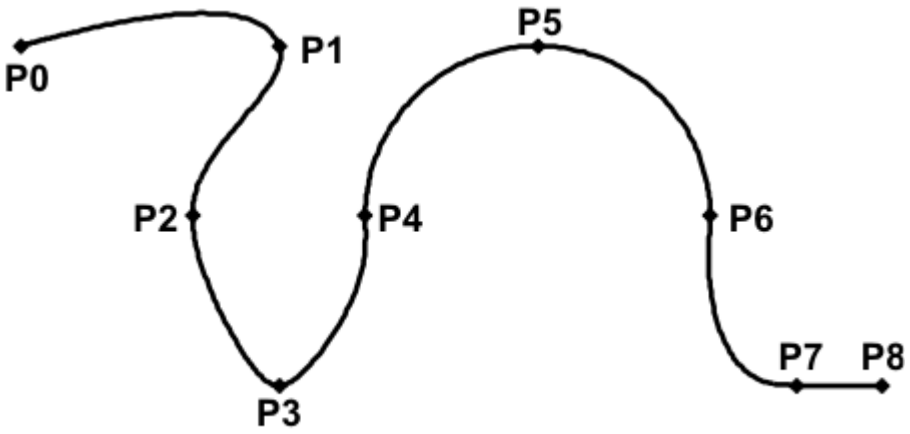


Abb. 9-19: Ursprüngliche Bahn

Gegenüber der ursprünglichen Bahn wird ein Punkt verschoben:

P3 wird verschoben. Dadurch ändert sich die Bahn in den Segmenten P1 - P2, P2 - P3 und P3 - P4. Das Segment P4 - P5 ändert sich in diesem Fall nicht, da es zu einem SCIRC gehört und dadurch eine Kreisbahn festgelegt ist.

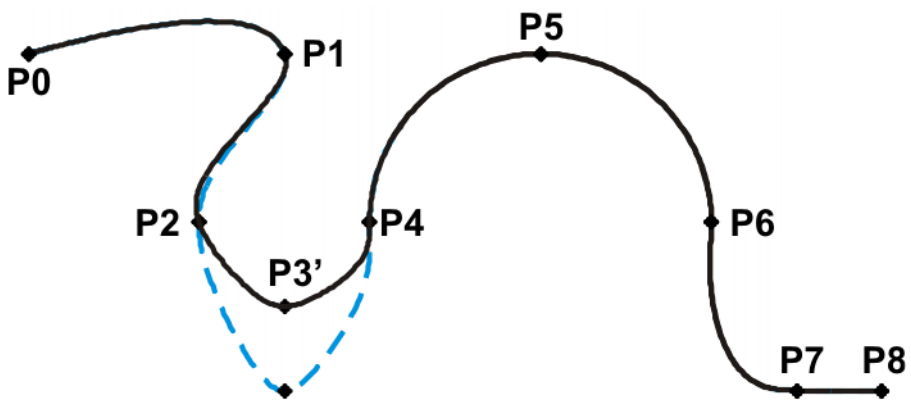


Abb. 9-20: Punkt wurde verschoben

Gegenüber der ursprünglichen Bahn wird der Typ eines Segments geändert:

Bei der ursprünglichen Bahn wird der Segmenttyp von P2 - P3 von SPL in SLIN geändert. Die Bahn ändert sich in den Segmenten P1 - P2, P2 - P3 und P3 - P4.

```
PTP P0
SPLINE
 SPL P1
 SPL P2
 SLIN P3
 SPL P4
 SCIRC P5, P6
 SPL P7
 SLIN P8
ENDSPLINE
```

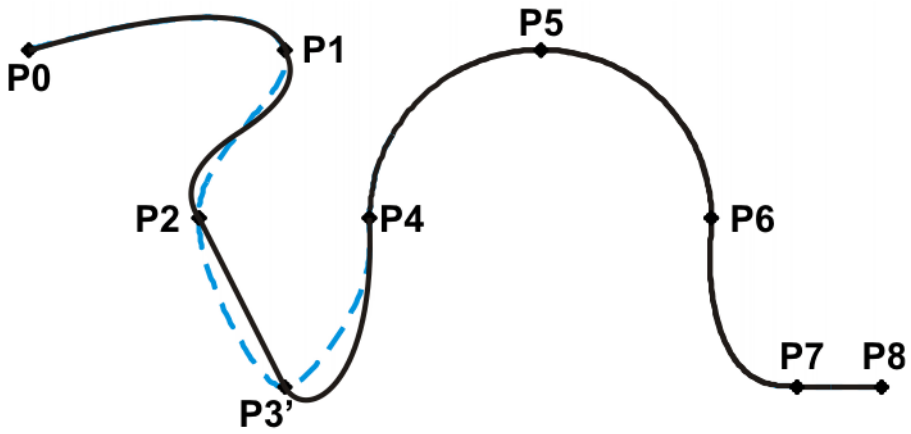


Abb. 9-21: Segmenttyp wurde geändert

Beispiel 2

Ursprüngliche Bahn:

```
...
SPLINE
 SPL {X 100, Y 0, ...}
 SPL {X 102, Y 0}
 SPL {X 104, Y 0}
```

```
 SPL {X 204, Y 0}
ENDSPLINE
```

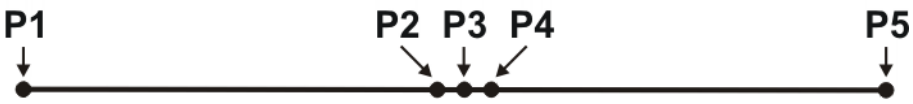


Abb. 9-22: Ursprüngliche Bahn

Gegenüber der ursprünglichen Bahn wird ein Punkt verschoben:

P3 wird verschoben. Dadurch ändert sich die Bahn in allen dargestellten Segmenten. Da P2 - P3 und P3 - P4 sehr kurze Segmente und P1 - P2 und P4 - P5 lange Segmente sind, bewirkt die kleine Verschiebung ein starke Änderung der Bahn.

```
...
SPLINE
 SPL {X 100, Y 0, ...}
 SPL {X 102, Y 1}
 SPL {X 104, Y 0}
 SPL {X 204, Y 0}
ENDSPLINE
```

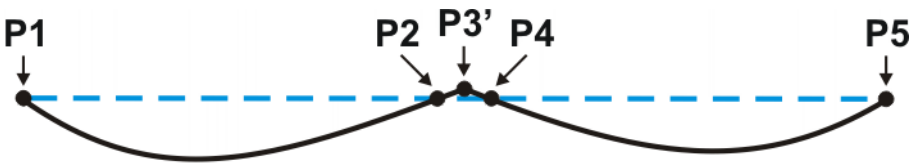


Abb. 9-23: Punkt wurde verschoben

Abhilfe:

- Punktabstände gleichmäßiger verteilen
- Geraden (außer sehr kurze Geraden) als SLIN-Segmente programmieren

9.7.4 Überschleifen von Spline-Bewegungen

Alle Spline-Blöcke und alle Spline-Einzelbewegungen können miteinander überschliffen werden. Es ist gleichgültig, ob es sich um CP- oder PTP-Spline-Blöcke handelt oder um welche Einzelbewegung es sich handelt.

Der Überschleifbogen entspricht vom Bewegungstyp her immer der zweiten Bewegung. Beim SPTP-SLIN-Überschleifen z. B. ist der Überschleifbogen vom Typ CP.

Spline-Bewegungen können nicht mit herkömmlichen Bewegungen (LIN, CIRC, PTP) überschliffen werden.

Überschleifen nicht möglich wegen Zeit oder Vorlaufstopps:

Wenn ein Überschleifen aus zeitlichen Gründen oder wegen Vorlaufstopps nicht möglich ist, wartet der Roboter am Beginn des Überschleifbogens.

- Bei zeitlichen Gründen: Der Roboter fährt weiter, sobald der nächste Satz geplant werden konnte.
- Bei einem Vorlaufstopp: Mit dem Beginn des Überschleifbogens ist das Ende des aktuellen Satzes erreicht. D. h., der Vorlaufstopp ist aufgehoben und die Robotersteuerung kann den nächsten Satz planen. Der Roboter fährt weiter.

In beiden Fällen fährt der Roboter nun den Überschleifbogen. Das Überschleifen ist also genaugenommen möglich, es verzögert sich nur.

Dieses Verhalten steht im Gegensatz zu LIN-, CIRC- oder PTP-Bewegungen. Wenn hier ein Überschleifen aus den genannten Gründen nicht möglich ist, wird der Zielpunkt genau angefahren.

Kein Überschleifen in MSTEP und ISTEP:

In den Programmablaufarten MSTEP und ISTEP wird auch bei überschliffenen Bewegungen der Zielpunkt genau angefahren.

Beim Überschleifen von Spline-Block zu Spline-Block ist als Folge dieses Genauhalts die Bahn im letzten Segment des ersten Blocks und im ersten Segment des zweiten Blocks anders als in der Programmablaufart GO.

Bei allen anderen Segmenten in den beiden Spline-Blöcken ist die Bahn in MSTEP, ISTEP und GO gleich.

9.7.5 Überschliffene CP-Bewegung durch Spline-Block ersetzen

Beschreibung

Um herkömmliche überschliffene CP-Bewegungen durch Spline-Blöcke zu ersetzen, muss das Programm folgendermaßen geändert werden:

- LIN - LIN ersetzen durch SLIN - SPL - SLIN.
- LIN - CIRC ersetzen durch SLIN - SPL - SCIRC.

  Empfehlung: Den SPL ein Stück in den ursprünglichen Kreis hineinragen lassen. Der SCIRC beginnt somit später als der ursprüngliche CIRC.

Bei überschliffenen Bewegungen wird der Eckpunkt programmiert. Im Spline-Block werden stattdessen Punkte am Überschleifbeginn und am Überschleifende programmiert.

Folgende überschliffene Bewegung soll nachgebildet werden:

```
LIN P1 C_DIS
LIN P2
```

Spline-Bewegung:

```
SPLINE
 SLIN P1A
 SPL P1B
 SLIN P2
ENDSPLINE
```

P1A = Überschleifbeginn, P1B = Überschleifende

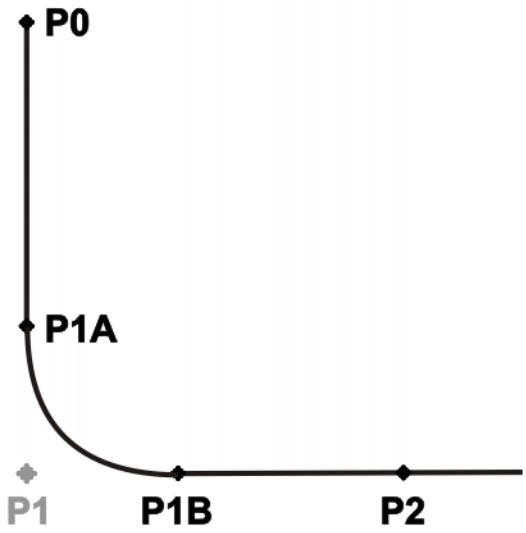


Abb. 9-24: Überschliffene Bewegung - Spline-Bewegung

Möglichkeiten, um P1A und P1B zu ermitteln:

- Die überschliffene Bahn abfahren und an der gewünschten Stelle die Positionen über Trigger speichern.
- Punkte im Programm mit KRL berechnen.
- Der Überschleifbeginn kann aus dem Überschleifkriterium ermittelt werden. Beispiel: Wenn als Überschleifkriterium C_DIS angegeben ist, entspricht der Abstand vom Überschleifbeginn bis zum Eckpunkt dem Wert von $APO.CDIS.

  Das Überschleifende ist abhängig von der programmierten Geschwindigkeit.

Die SPL-Bahn entspricht nicht exakt dem Überschleifbogen, selbst wenn P1A und P1B genau am Überschleifbeginn und am Überschleifende liegen. Um exakt den Überschleifbogen zu erhalten, müssen zusätzliche Punkte in den Spline eingefügt werden. In der Regel ist ein Punkt ausreichend.

Beispiel

Folgende überschliffene Bewegung soll nachgebildet werden:

```
$APO.CDIS=20
$VEL.CP=0.5
LIN {Z 10} C_DIS
LIN {Y 60}
```

Spline-Bewegung:

```
SPLINE WITH $VEL.CP=0.5
 SLIN {Z 30}
 SPL {Y 30, Z 10}
 SLIN {Y 60}
ENDSPLINE
```

Der Beginn des Überschleifbogens wurde aus dem Überschleifkriterium errechnet.

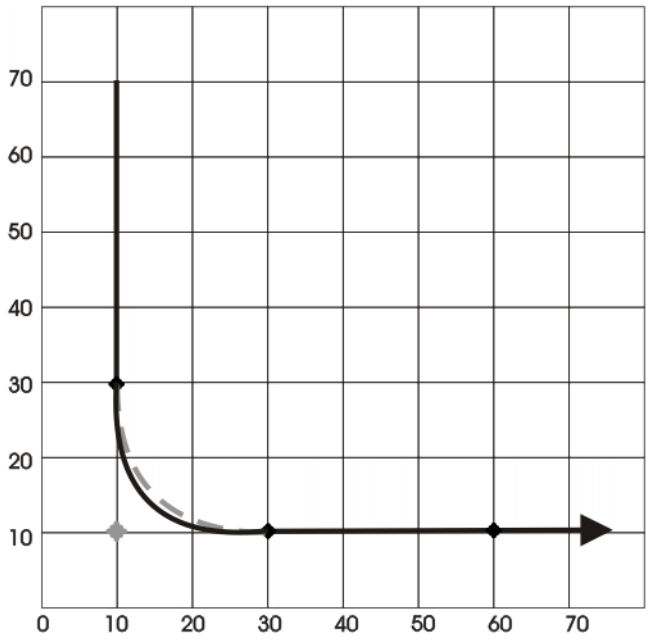


Abb. 9-25: Beispiel: Überschliffene Bewegung - Spline-Bewegung, 1

Die SPL-Bahn entspricht noch nicht exakt dem Überschleifbogen. Deswegen wird ein weiteres SPL-Segment in den Spline eingefügt.

```
SPLINE WITH $VEL.CP=0.5
 SLIN {Z 30}
 SPL {Y 15, Z 15}
 SPL {Y 30, Z 10}
 SLIN {Y 60}
ENDSPLINE
```

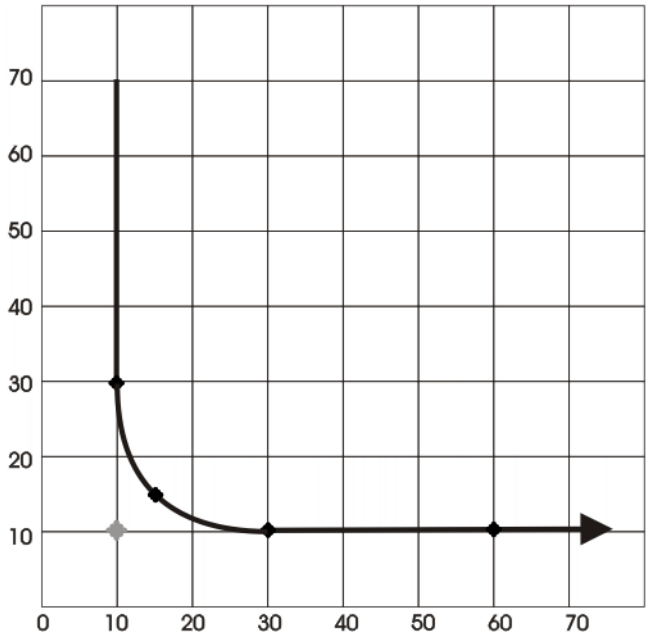


Abb. 9-26: Beispiel: Überschliffene Bewegung - Spline-Bewegung, 2

Durch den zusätzlichen Punkt entspricht die Bahn nun dem Überschleifbogen.

9.7.5.1 SLIN-SPL-SLIN-Übergang

Bei einer Segmentfolge SLIN-SPL-SLIN ist es in der Regel erwünscht, dass das SPL-Segment innerhalb des kleineren Winkels zwischen den beiden Geraden verläuft. Abhängig von Start- und Zielpunkt des SPL-Segments kann die Bahn jedoch auch außerhalb verlaufen.

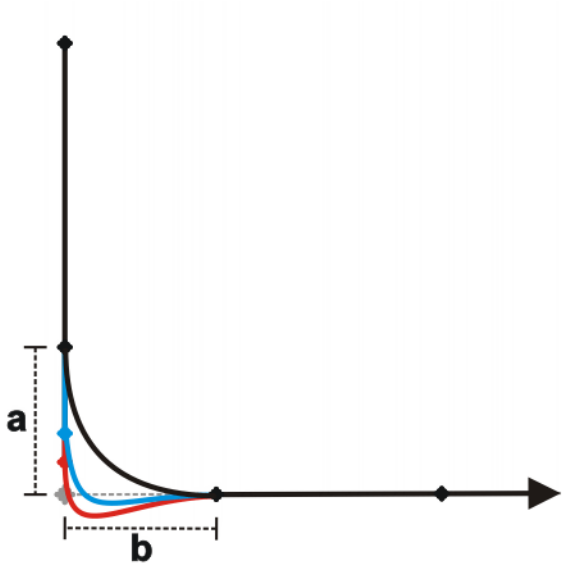


Abb. 9-27: SLIN-SPL-SLIN

Die Bahn verläuft innerhalb, wenn folgende Voraussetzungen erfüllt sind:

- Die beiden SLIN-Segmente schneiden sich in ihrer Verlängerung.
- $2/3 \leq a/b \leq 3/2$

  a = Abstand vom Startpunkt des SPL-Segments zum Schnittpunkt der SLIN-Segmente

  b = Abstand vom Schnittpunkt der SLIN-Segmente zum Zielpunkt des SPL-Segments

9.8 Orientierungsführung CP-Spline

Beschreibung

Der TCP kann am Start- und am Zielpunkt einer Bewegung unterschiedliche Orientierungen haben. Beim Programmieren einer CP-Spline-Bewegung muss ausgewählt werden, wie mit den unterschiedlichen Orientierungen umgegangen werden soll.

Die Orientierungsführung wird folgendermaßen festgelegt:

- Bei Programmierung mit KRL-Syntax: über die Systemvariable $ORI_TYPE
- Bei Programmierung über Inline-Formulare: im Optionsfenster **Bewegungsparameter**

Orientierungsführung	Beschreibung
■ Optionsfenster: **Konstante Orientierung** ■ $ORI_TYPE = #CONSTANT	Die Orientierung des TCP bleibt während der Bewegung konstant. Die Orientierung des Startpunkts wird beibehalten. Die programmierte Orientierung des Zielpunkts wird nicht berücksichtigt.
■ Optionsfenster: **Standard** ■ $ORI_TYPE = #VAR	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich. Im Zielpunkt hat der TCP die programmierte Orientierung.

Orientierungsführung	Beschreibung
■ Optionsfenster: **Hand PTP** ■ $ORI_TYPE = #JOINT	Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich. Dies geschieht durch lineare Überführung (achsspezifisches Verfahren) der Handachswinkel. **Hinweis**: **Hand PTP** dann verwenden, wenn der Roboter mit **Standard** in eine Handachsen-Singularität gerät. Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich, jedoch nicht ganz gleichmäßig. **Hand PTP** ist deshalb nicht geeignet, wenn ein bestimmter Verlauf der Orientierung exakt gehalten muss, z. B. beim Laserschweißen.
■ Optionsfenster: **Ohne Orientierung** ■ $ORI_TYPE = #IGNORE	Diese Option steht nur für CP-Spline-Segmente zur Verfügung. (Nicht für den Spline-Block oder für Spline-Einzelbewegungen.) Diese Option verwendet man, wenn an einem Punkt keine bestimmte Orientierung erforderlich ist. (>>> "#IGNORE" Seite 305)

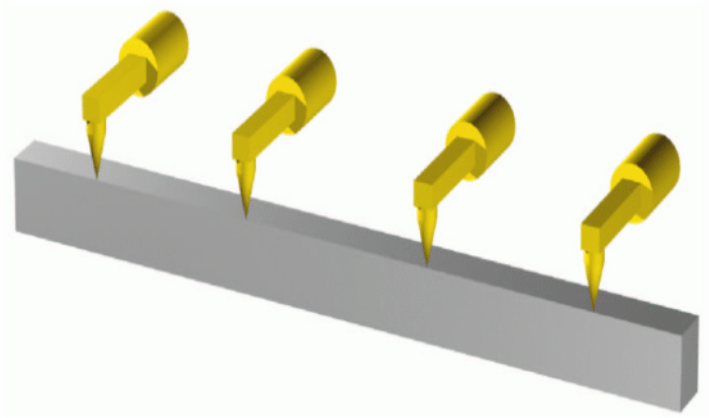


Abb. 9-28: Konstante Orientierung

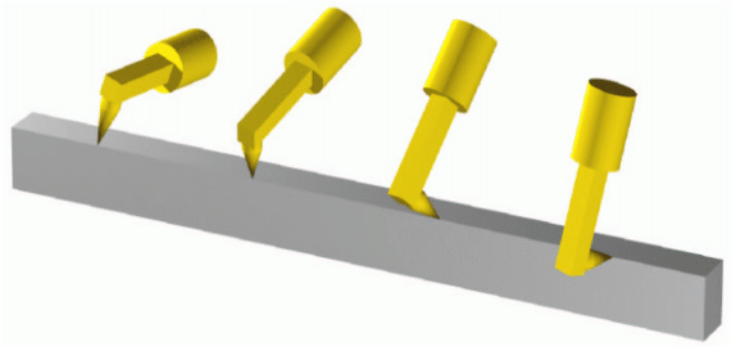


Abb. 9-29: Orientierungsführung Standard

#IGNORE

$ORI_TYPE = #IGNORE verwendet man, wenn an einem Punkt keine bestimmte Orientierung erforderlich ist. Wenn diese Option ausgewählt ist, ignoriert die Robotersteuerung die geteachte oder programmierte Orientierung des Punktes. Stattdessen errechnet sie auf Grundlage der Orientierungen der umgebenden Punkte die optimale Orientierung für diesen Punkt. Dies verringert die Taktzeit.

Beispiel:

```
SPLINE
  SPL XP1
```

```
  SPL XP2
  SPL XP3 WITH $ORI_TYPE=#IGNORE
  SPL XP4 WITH $ORI_TYPE=#IGNORE
  SPL XP5
  SPL XP6
ENDSPLINE
```

Die geteachte oder programmierte Orientierung von XP3 und XP4 wird ignoriert.

Eigenschaften von $ORI_TYPE = #IGNORE:

- In den Programmablaufarten MSTEP und ISTEP stoppt der Roboter mit den von der Robotersteuerung errechneten Orientierungen.
- Bei einer Satzanwahl auf einen Punkt mit #IGNORE nimmt der Roboter die von der Robotersteuerung errechnete Orientierung an.

Für folgende Segmente ist $ORI_TYPE = #IGNORE nicht erlaubt:

- Das letzte Segment in einem Spline-Block
- SCIRC-Segmente mit $CIRC_TYPE=#PATH
- Segmente, auf die ein SCIRC-Segment folgt mit $CIRC_TYPE=#PATH
- Segmente, auf die ein Segment folgt mit $ORI_TYPE=#CONSTANT

SCIRC

Für SCIRC-Bewegungen kann festgelegt werden, ob die Orientierungsführung raumbezogen oder bahnbezogen sein soll.

(>>> 9.8.1 "SCIRC: Bezugssystem der Orientierungsführung" Seite 306)

Für SCIRC-Bewegungen kann festgelegt werden, ob und inwieweit die Orientierung des Hilfspunkts berücksichtigt wird. Außerdem kann das Orientierungsverhalten am Zielpunkt festgelegt werden.

(>>> 9.8.2 "SCIRC: Orientierungsverhalten" Seite 307)

$SPL_ORI_JOINT_AUTO

$SPL_ORI_JOINT_AUTO dient zur Optimierung des Fahrverhaltens in Nähe von Handachsen-Singularitäten.

Wirkungsweise von $SPL_ORI_JOINT_AUTO = #ON:

- Für CP-Spline-Bewegungen, für die $ORI_TYPE = #VAR gilt, entscheidet die Robotersteuerung automatisch pro Bewegung (d. h. auch pro Segment), ob sie als #VAR oder als #JOINT gefahren werden.

$SPL_ORI_JOINT_AUTO = #ON ist eine Alternative zur Verwendung von $ORI_TYPE = #JOINT. Während $ORI_TYPE = #JOINT gezielt für einzelne Bewegungen verwendet werden kann, ermöglicht $SPL_ORI_JOINT_AUTO = #ON eine automatische Optimierung über beliebig große Programmsequenzen, bei sehr kleinem Änderungsaufwand.

$SPL_ORI_JOINT_AUTO kann nur über ein Roboterprogramm geändert werden. $SPL_ORI_JOINT_AUTO kann nicht in Spline-Segmenten gesetzt werden.

Default: $SPL_ORI_JOINT_AUTO = #OFF

9.8.1 SCIRC: Bezugssystem der Orientierungsführung

Für SCIRC-Bewegungen kann festgelegt werden, ob die Orientierungsführung raumbezogen oder bahnbezogen sein soll. Dies kann folgendermaßen festgelegt werden:

- Bei Programmierung über Inline-Formulare: im Optionsfenster **Bewegungsparameter**
- Bei Programmierung mit KRL-Syntax: über die Systemvariable $CIRC_TYPE

Orientierungsführung	Beschreibung
■ Optionsfenster: **basisbezogen** ■ $CIRC_TYPE = #BASE	Basisbezogene Orientierungsführung während der Kreisbewegung
■ Optionsfenster: **bahnbezogen** ■ $CIRC_TYPE = #PATH	Bahnbezogene Orientierungsführung während der Kreisbewegung

Für folgende Bewegungen ist $CIRC_TYPE = #PATH nicht erlaubt:

- SCIRC-Segmente, für die $ORI_TYPE = #IGNORE gilt
- SCIRC-Bewegungen, denen ein Spline-Segment vorausgeht, für das $ORI_TYPE = #IGNORE gilt

(>>> 9.6.1 "Kombinationen von $ORI_TYPE und $CIRC_TYPE" Seite 291)

9.8.2 SCIRC: Orientierungsverhalten

Beschreibung

Bei SCIRC-Bewegungen kann die Robotersteuerung die programmierte Orientierung des Hilfspunkts berücksichtigen. Ob und inwieweit sie tatsächlich berücksichtigt wird, kann der Benutzer festlegen:

- Bei Programmierung mit KRL-Syntax: über die Systemvariable $CIRC_MODE
- Bei Programmierung über Inline-Formulare: im Optionsfenster **Bewegungsparameter**, Registerkarte **Kreiskonfiguration**

Auf die gleiche Weise kann außerdem für SCIRC-Anweisungen mit Kreiswinkel festgelegt werden, ob der Zielpunkt die programmierte Orientierung haben soll oder ob die Orientierung entsprechend dem Kreiswinkel weitergeführt werden soll.

$CIRC_MODE ist nur über eine SCIRC-Anweisung beschreibbar. $CIRC_MODE kann nicht gelesen werden.

Syntax

Für Hilfspunkte:

```
$CIRC_MODE.AUX_PT.ORI = VerhaltenHP
```

Für Zielpunkte:

```
$CIRC_MODE.TARGET_PT.ORI = VerhaltenZP
```

Erläuterung der Syntax

Element	Beschreibung
VerhaltenHP	Datentyp: ENUM ■ **#INTERPOLATE**: Im Hilfspunkt nimmt der TCP die programmierte Orientierung an. ■ **#IGNORE**: Die Robotersteuerung ignoriert die programmierte Orientierung des Hilfspunkts. Die Start-Orientierung des TCP wird auf dem kürzesten Weg in die Ziel-Orientierung überführt. ■ **#CONSIDER** (Default): Grundsätzlich gibt es 2 Wege, wie die Start-Orientierung mit einer Drehung in die Ziel-Orientierung überführt werden kann: Einen kürzeren und einen längeren. Mit #CONSIDER wählt die Robotersteuerung denjenigen Weg, der der programmierten Orientierung des Hilfspunkts näher kommt. Es kann sein, dass der TCP irgendwo auf dem Weg die programmierte Orientierung des Hilfspunkts annimmt. Dies muss aber nicht der Fall sein.
VerhaltenZP	Datentyp: ENUM ■ **#INTERPOLATE**: Am tatsächlichen Zielpunkt wird die programmierte Orientierung des Zielpunkts angenommen. (Einzige Möglichkeit für SCIRC ohne Kreiswinkel-Angabe. Wenn #EXTRAPOLATE gesetzt wird, wird trotzdem #INTERPOLATE ausgeführt.) ■ **#EXTRAPOLATE**: Die Orientierung wird an den Kreiswinkel angepasst: Wenn der Kreiswinkel die Bewegung verlängert, wird am programmierten Zielpunkt die programmierte Orientierung angenommen. Bis zum tatsächlichen Zielpunkt wird die Orientierung dementsprechend weitergeführt. Wenn der Kreiswinkel die Bewegung verkürzt, wird die programmierte Orientierung nicht erreicht. (Default für SCIRC mit Kreiswinkel-Angabe)

Einschränkungen

- Wenn für ein SCIRC-Segment $ORI_TYPE = #IGNORE gilt, dann wird $CIRC_MODE nicht ausgewertet.
- Wenn einem SCIRC-Segment ein SCIRC- oder SLIN-Segment vorausgeht, für das $ORI_TYPE = #IGNORE gilt, dann kann #CONSIDER in diesem SCIRC-Segment nicht verwendet werden.

Für SCIRC mit Kreiswinkel:

- Für den Hilfspunkt darf nicht #INTERPOLATE gesetzt werden.
- Wenn $ORI_TYPE = #IGNORE gilt, dann darf für den Zielpunkt nicht #EXTRAPOLATE gesetzt werden.
- Wenn ein Spline-Segment vorausgeht, für das $ORI_TYPE = #IGNORE gilt, dann darf für den Zielpunkt nicht #EXTRAPOLATE gesetzt werden.

9.8.2.1 SCIRC: Orientierungsverhalten – Beispiel Hilfspunkt

Beschreibung

Für den TCP wurden folgende Orientierungen programmiert:

- Startpunkt: 0°
- Hilfspunkt: 98°
- Zielpunkt: 197°

Die Umorientierung beträgt also 197°. Wenn der Hilfspunkt ignoriert wird, kann die Zielorientierung auch über die kürzere Umorientierung von 360° - 197° = 163° erreicht werden.

- Die gestrichelten, orangen Pfeile zeigen die programmierte Orientierung an.
- Die grauen Pfeile deuten schematisch an, wie die tatsächliche Orientierung ausfallen würde, sofern sie von der programmierten Orientierung abweicht.

#INTERPOLATE

Der TCP nimmt am Hilfspunkt die programmierte Orientierung von 98° an. Die Umorientierung beträgt 197°.

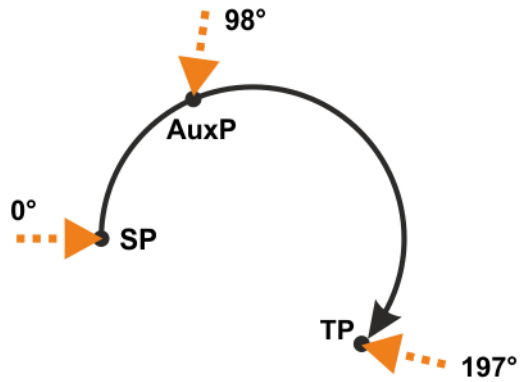


Abb. 9-30: #INTERPOLATE

SP	Startpunkt
AuxP	Hilfspunkt
TP	Zielpunkt

#IGNORE

Die kurze Umorientierung mit 163° wird gefahren. Die programmierte Orientierung des Hilfspunkts wird ignoriert.

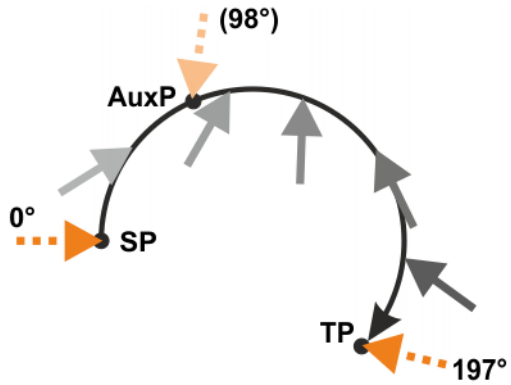


Abb. 9-31: #IGNORE

#CONSIDER

> #CONSIDER ist geeignet, wenn der Benutzer festlegen möchte, in welche Richtung der TCP umorientieren soll, ohne dass es auf eine bestimmte Orientierung im Hilfspunkt ankommt. Der Benutzer kann die Richtung über den Hilfspunkt vorgeben.

Die programmierte Orientierung des Hilfspunkts ist 98° und liegt somit auf dem längeren Weg. Die Robotersteuerung schlägt für die Umorientierung deshalb den längeren Weg ein.

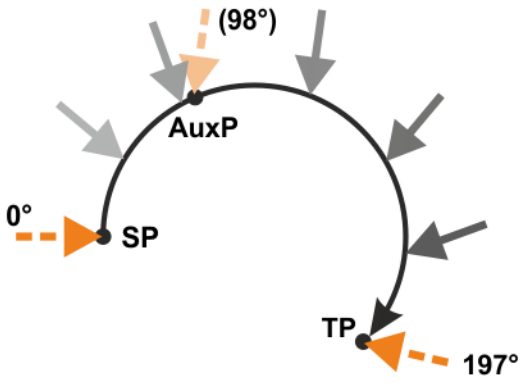


Abb. 9-32: #CONSIDER

Weiteres Beispiel für #CONSIDER:

Wenn der Hilfspunkt mit 262° programmiert wäre, würde er auf dem kürzeren Weg liegen. Die Robotersteuerung würde für die Umorientierung deshalb den kürzeren Weg einschlagen. Die grauen Pfeile zeigen, dass sie dabei keineswegs unbedingt die programmierte Orientierung des Hilfspunkts annimmt.

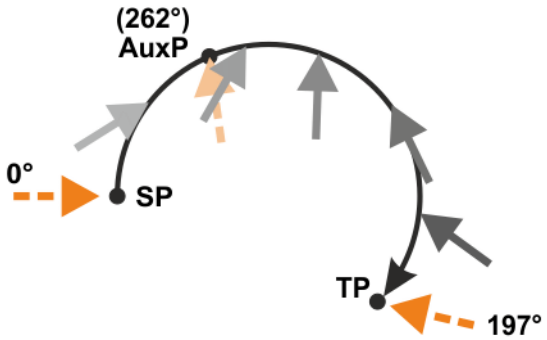


Abb. 9-33: #CONSIDER, weiteres Beispiel

9.8.2.2 SCIRC: Orientierungsverhalten – Beispiel Zielpunkt

Beschreibung

- Die gestrichelten, orangen Pfeile zeigen die programmierte Orientierung an.
- Die grauen Pfeile zeigen die tatsächliche Orientierung an, sofern sie von der programmierten Orientierung abweicht.

#INTERPOLATE

In **TP**, der sich vor **TP_CA** befindet, ist die programmierte Orientierung noch nicht erreicht. In **TP_CA** wird die programmierte Orientierung angenommen.

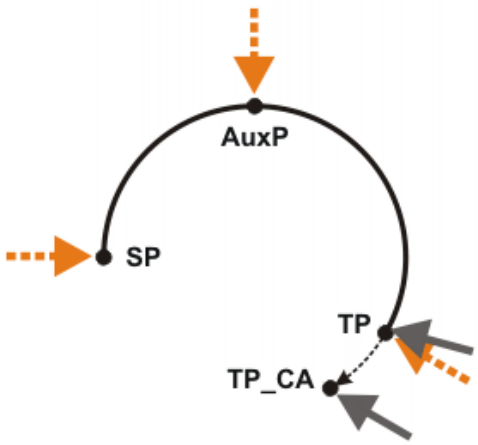


Abb. 9-34: #INTERPOLATE

SP	Startpunkt
AuxP	Hilfspunkt
TP	Programmierter Zielpunkt
TP_CA	Tatsächlicher Zielpunkt. Ergibt sich durch den Kreiswinkel.

#EXTRAPOLATE In **TP** wird die programmierte Orientierung angenommen. Für **TP_CA** wird diese Orientierung gemäß dem Kreiswinkel weitergeführt.

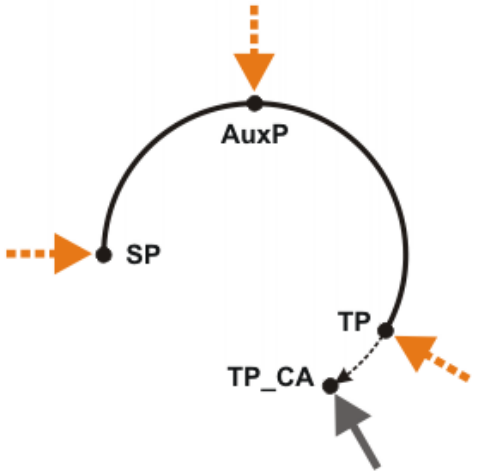


Abb. 9-35: #EXTRAPOLATE

9.9 Kreiswinkel

Für die meisten Kreisbewegungen kann ein Kreiswinkel programmiert werden.

Informationen dazu, ob dies für eine bestimmte Kreisbewegung möglich ist, können der Beschreibung zur Bewegung im Programmierteil dieser Dokumentation entnommen werden.

Der Kreiswinkel gibt den Gesamtwinkel der Bewegung an. Er ermöglicht dadurch eine Verlängerung der Bewegung über den programmierten Zielpunkt hinaus, oder auch eine Verkürzung. Der tatsächliche Zielpunkt entspricht dadurch nicht mehr dem programmierten Zielpunkt.

Einheit: Grad. Es können Kreiswinkel größer + 360° und kleiner - 360° programmiert werden.

Das Vorzeichen bestimmt, in welcher Richtung die Kreisbahn abgefahren wird:

- Positiv: Richtung **Startpunkt › Hilfspunkt › Zielpunkt**
- Negativ: Richtung **Startpunkt › Zielpunkt › Hilfspunkt**

9.10 Status und Turn

Übersicht

Die Werte von Position (X, Y, Z) und Orientierung (A, B, C) des TCP reichen nicht aus, um die Position eines Roboters eindeutig festzulegen, da bei gleichem TCP dennoch mehrere Achsstellungen möglich sind. Status und Turn dienen dazu, aus mehreren möglichen Achsstellungen eine eindeutige Stellung festzulegen.

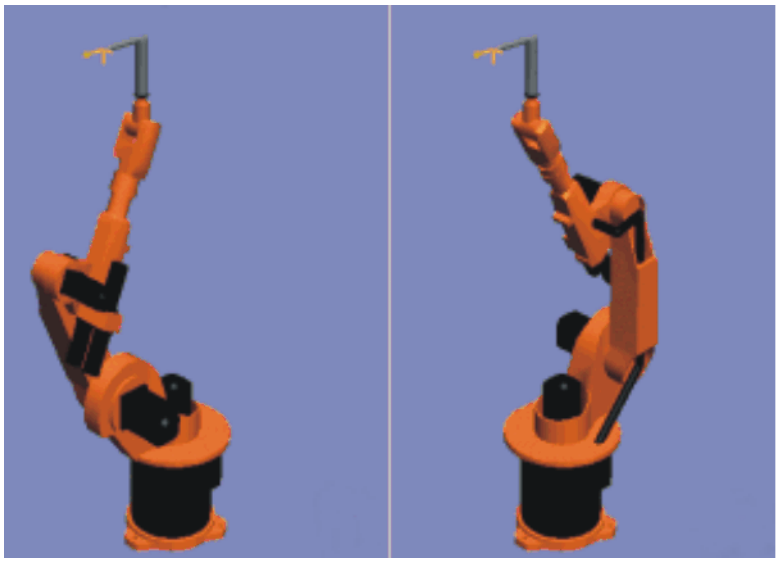


Abb. 9-36: Beispiel: TCP gleich, Achsstellung verschieden

Status (S) und Turn (T) sind Bestandteile der Datentypen POS und E6POS:

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

KRL-Programm

Die Robotersteuerung berücksichtigt programmierte Status- und Turn-Werte nur bei PTP-Bewegungen. Bei CP-Bewegungen werden sie ignoriert.

Die erste Bewegungsanweisung in einem KRL-Programm muss deshalb eine der folgenden Anweisungen sein, damit für den Roboter eine eindeutige Ausgangslage definiert ist:

- Eine vollständige PTP-Anweisung vom Typ POS oder E6POS
- Oder eine vollständige PTP-Anweisung vom Typ AXIS oder E6AXIS

"Vollständig" bedeutet, dass alle Komponenten des Zielpunkts angegeben werden müssen. Die Default-HOME-Position ist immer eine vollständige PTP-Anweisung.

In den weiteren Anweisungen können Status und Turn weggelassen werden:

- Die Robotersteuerung behält den bisherigen Status-Wert bei.
- Der Turn-Wert ergibt sich bei CP-Bewegungen aus der Bahn. Bei PTP-Bewegungen wählt die Robotersteuerung den Turn-Wert, der den kürzesten möglichen Weg ergibt.

9.10.1 Status

Die Status-Angabe verhindert Mehrdeutigkeiten bei der Achsstellung.

Bit 0

Bit 0 gibt die Position des Schnittpunkts der Handachsen (A4, A5, A6) an.

Position	Wert
Überkopfbereich Der Roboter befindet sich im Überkopfbereich, wenn der x-Wert des Schnittpunkts der Handachsen, bezogen auf das A1-Koordinatensystem, negativ ist.	Bit 0 = 1
Grundbereich Der Roboter befindet sich im Grundbereich, wenn der x-Wert des Schnittpunkts der Handachsen, bezogen auf das A1-Koordinatensystem, positiv ist.	Bit 0 = 0

Das A1-Koordinatensystem ist mit dem $ROBROOT-Koordinatensystem identisch, wenn die Achse 1 auf 0° steht. Bei Werten ungleich 0° wird es mit der Achse 1 mitbewegt.

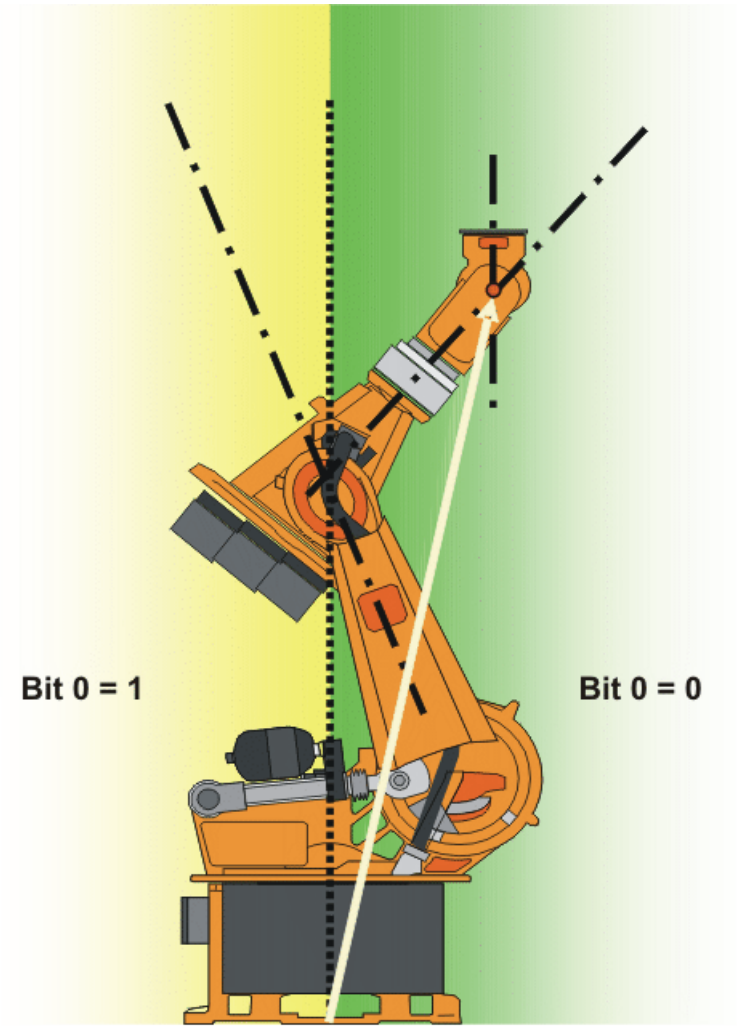


Abb. 9-37: Beispiel: Der Schnittpunkt der Handachsen (roter Punkt) liegt im Grundbereich.

Bit 1

Bit 1 gibt die Position von Achse 3 an. Der Winkel, bei dem sich der Wert von Bit 1 ändert, ist abhängig vom Robotertyp.

Für Roboter, deren Achsen 3 und 4 sich schneiden, gilt:

Position	Wert
A3 ≥ 0°	Bit 1 = 1
A3 < 0°	Bit 1 = 0

Bei Robotern mit einem Offset zwischen Achse 3 und 4 ist der Winkel, bei dem sich der Wert von Bit 1 ändert, von der Größe dieses Offsets abhängig.

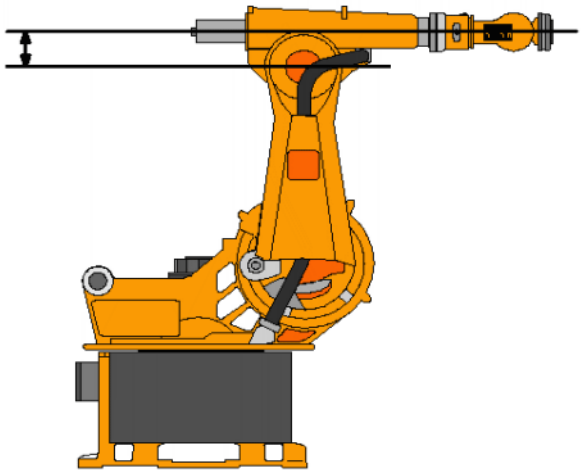


Abb. 9-38: Offset zwischen A3 und A4: Beispiel KR 30

Bit 2

Bit 2 gibt die Position von Achse 5 an.

Position (Bezug: A4 = 0°)	Wert
A5 ist nach oben gekippt.	Bit 2 = 1
A5 = 0°	Bit 2 = 0
A5 ist nach unten gekippt.	

Das Vorzeichen des Winkels von A5 ist nicht ausschlaggebend! Ausschlaggebend ist die Richtung, in die die A5 gekippt ist, also nach oben oder unten. Die Richtungsangabe wiederum bezieht sich auf die Nullstellung der A4.

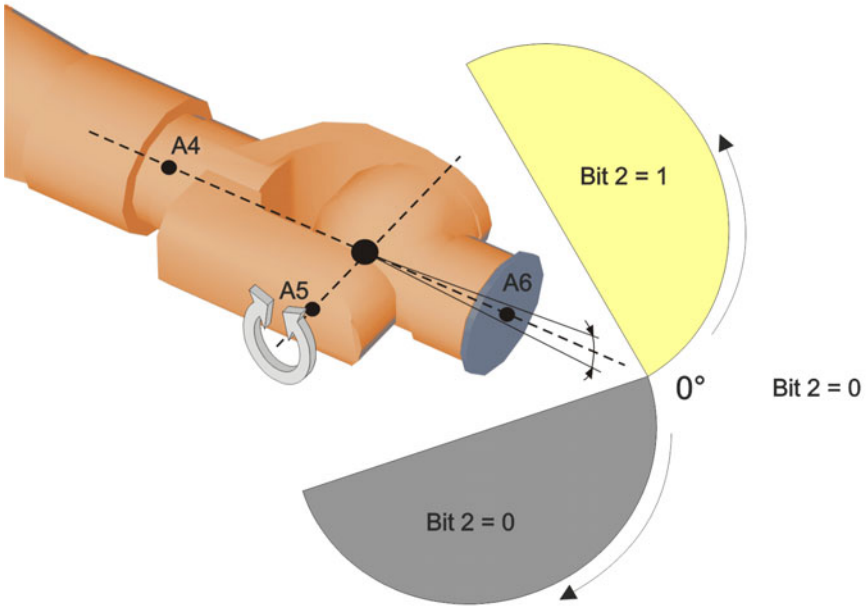


Abb. 9-39: Bit 2

Bit 3

Bit 3 wird nicht genutzt und ist immer 0.

Bit 4

Bit 4 gibt an, ob der Punkt mit einem absolutgenauen Roboter geteacht wurde oder nicht.

Der Punkt kann unabhängig vom Wert des Bits sowohl mit absolutgenauen als auch mit nicht absolutgenauen Robotern abgefahren werden. Das Bit 4 dient nur der Information und hat keinen Einfluss darauf, wie die Robotersteuerung den Punkt berechnet. Dies bedeutet auch, wenn ein Roboter offline programmiert wird, kann Bit 4 außer Acht gelassen werden.

Beschreibung	Wert
Der Punkt wurde nicht mit einem absolutgenauen Roboter geteacht.	Bit 4 = 0
Der Punkt wurde mit einem absolutgenauen Roboter geteacht.	Bit 4 = 1

9.10.2 Turn

Beschreibung

Die Turn-Angabe ermöglicht es, auch Achswinkel, die größer +180 ° oder kleiner -180 ° sind, ohne besondere Verfahrstrategie (z. B. Zwischenpunkte) anzufahren. Die einzelnen Bits bestimmen bei rotatorischen Achsen das Vorzeichen des Achswertes folgendermaßen:

Bit = 0: Winkel ≥ 0 °

Bit = 1: Winkel < 0 °

Wert	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A6 ≥ 0 °	A5 ≥ 0 °	A4 ≥ 0 °	A3 ≥ 0 °	A2 ≥ 0 °	A1 ≥ 0 °
1	A6 < 0 °	A5 < 0 °	A4 < 0 °	A3 < 0 °	A2 < 0 °	A1 < 0 °

Beispiel

```
DECL POS XP1 = {X 900, Y 0, Z 800, A 0, B 0, C 0, S 6, T 19}
```

T 19 entspricht T 'B010011'. Dies bedeutet:

Achse	Winkel
A 1	negativ
A 2	negativ
A 3	positiv
A 4	positiv
A 5	negativ
A 6	positiv

9.11 Singularitäten

Die KUKA Roboter mit 6 Freiheitsgraden haben 3 verschiedene singuläre Stellungen.

- Überkopf-Singularität
- Strecklagen-Singularität
- Handachsen-Singularität

Eine singuläre Stellung ist dadurch gekennzeichnet, dass eine Rückwärtstransformation (Umrechnung kartesischer Koordinaten in achsspezifische Werte) trotz vorgegebenem Status und Turn nicht eindeutig möglich ist. In diesem Fall oder wenn kleinste kartesische Änderungen zu sehr großen Achswinkel-Änderungen führen, spricht man von singulären Stellungen.

Überkopf

Bei der Überkopf-Singularität liegt der Handwurzelpunkt (= Mittelpunkt der Achse A5) senkrecht auf der Achse A1 des Roboters.

Die Position der Achse A1 ist durch Rückwärtstransformation nicht eindeutig bestimmbar und kann deshalb beliebige Werte annehmen.

Liegt der Zielpunkt eines PTP-Bewegungssatzes in dieser Überkopf-Singularität kann die Robotersteuerung durch die Systemvariable $SINGUL_POS[1] wie folgt reagieren:

- **0**: Der Winkel der Achse A1 wird auf Null Grad festgelegt. (Default-Einstellung)
- **1**: Der Winkel der Achse A1 bleibt vom Start- bis zum Zielpunkt gleich.

Strecklagen

Bei der Strecklagen-Singularität liegt der Handwurzelpunkt (= Mittelpunkt der Achse A5) in Verlängerung der Achse A2 und A3 des Roboters.

Der Roboter befindet sich am Rand seines Arbeitsbereichs.

Die Rückwärtstransformation liefert eindeutige Achswinkel, aber kleine kartesische Geschwindigkeiten haben große Achsgeschwindigkeiten der Achse A2 und A3 zur Folge.

Liegt der Zielpunkt eines PTP-Bewegungssatzes in dieser Strecklagen-Singularität kann die Robotersteuerung durch die Systemvariable $SINGUL_POS[2] wie folgt reagieren:

- **0**: Der Winkel der Achse A2 wird auf Null Grad festgelegt. (Default-Einstellung).
- **1**: Der Winkel der Achse A2 bleibt vom Start- bis zum Zielpunkt gleich.

Handachsen

Bei der Handachsen-Singularität stehen die Achsen A4 und A6 parallel zueinander und Achse A5 innerhalb des Bereichs von ±0,01812°.

Die Stellung der beiden Achsen ist durch eine Rückwärtstransformation nicht eindeutig bestimmbar. Es gibt aber beliebig viele Achsstellungen für Achse A4 und A6 deren Achswinkelsummen identisch sind.

Liegt der Zielpunkt eines PTP-Bewegungssatzes in dieser Handachsen-Singularität kann die Robotersteuerung durch die Systemvariable $SINGUL_POS[3] wie folgt reagieren:

- **0**: Der Winkel der Achse A4 wird auf Null Grad festgelegt. (Default-Einstellung)
- **1**: Der Winkel der Achse A4 bleibt vom Start- bis zum Zielpunkt gleich.

10 Programmierung für Benutzergruppe Anwender (Inline-Formulare)

HINWEIS Bei Programmen mit folgenden Achsbewegungen oder -positionen kann es an den Getrieben der Achsen zu einer Unterbrechung des Schmierfilms kommen:

- Bewegungen < 3°
- Oszillierende Bewegungen
- Dauerhaft oben liegende Getriebebereiche

Es muss sichergestellt werden, dass die Getriebe ausreichend mit Öl versorgt werden. Hierfür muss bei oszillierenden oder kurzen Bewegungen (< 3°) so programmiert werden, dass sich die betroffenen Achsen regelmäßig (z. Bsp. je Zyklus) um mehr als 40° bewegen.
Im Falle dauerhaft oben liegender Getriebebereiche muss eine ausreichende Ölversorgung erreicht werden, indem man Umorientierungen der Zentralhand programmiert. Auf diese Weise kann das Öl durch die Schwerkraft in alle Getriebebereiche gelangen. Erforderliche Häufigkeit der Umorientierungen:

- Bei geringer Belastung (Getriebetemperatur < +35 °C): 1-mal täglich
- Bei mittlerer Belastung (Getriebetemperatur +35 bis 55 °C): stündlich
- Bei starker Belastung (Getriebetemperatur > +55 °C): alle 10 min

Wenn dies nicht beachtet wird, können Schäden an den Getrieben entstehen.

10.1 Namen in Inline-Formularen

In Inline-Formularen können Namen für Datensätze eingegeben werden. Dazu gehören z. B. Punktnamen, Namen für Bewegungsdatensätze etc.

Für die Namen gelten folgende Einschränkungen:

- Maximale Länge 23 Zeichen
- Es sind keine Sonderzeichen zulässig außer $.
- An erster Stelle ist keine Ziffer zulässig.

Die Einschränkungen gelten nicht für Namen von Ausgängen.

Für Inline-Formulare in Technologiepaketen können andere Einschränkungen gelten.

10.2 PTP-, LIN-, CIRC-Bewegungen programmieren

10.2.1 PTP-Bewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an die Position verfahren, die als Zielpunkt geteacht werden soll.
2. Cursor in die Zeile setzen, nach der die Bewegungsanweisung eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **PTP** wählen.

4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.2.2 "Inline-Formular PTP" Seite 318)
5. Anweisung mit **Befehl OK** speichern.

10.2.2 Inline-Formular PTP

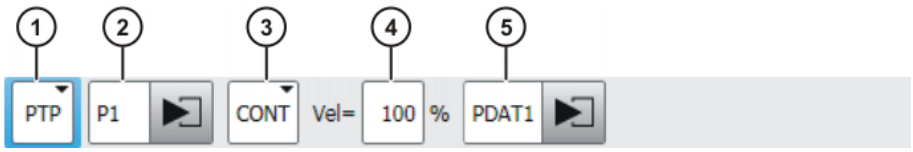


Abb. 10-1: Inline-Formular PTP-Bewegung

Pos.	Beschreibung
1	Bewegungsart **PTP**
2	Name des Zielpunkts Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
3	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
4	Geschwindigkeit ■ **1 … 100 %**
5	Name für den Bewegungsdatensatz Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.8 "Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)" Seite 321)

10.2.3 LIN-Bewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an die Position verfahren, die als Zielpunkt geteacht werden soll.
2. Cursor in die Zeile setzen, nach der die Bewegungsanweisung eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **LIN** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.2.4 "Inline-Formular LIN" Seite 319)
5. Anweisung mit **Befehl OK** speichern.

10.2.4 Inline-Formular LIN

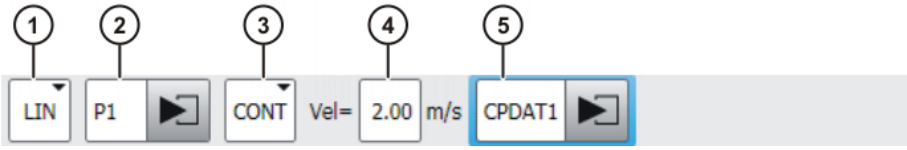


Abb. 10-2: Inline-Formular LIN-Bewegung

Pos.	Beschreibung
1	Bewegungsart **LIN**
2	Name des Zielpunkts Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
3	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
4	Geschwindigkeit ■ **0.001 … 2 m/s**
5	Name für den Bewegungsdatensatz Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.8 "Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)" Seite 321)

10.2.5 CIRC-Bewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an die Position verfahren, die als Hilfspunkt geteacht werden soll.
2. Cursor in die Zeile setzen, nach der die Bewegungsanweisung eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **CIRC** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.2.6 "Inline-Formular CIRC" Seite 320)
5. **Touchup HP** drücken.
6. Den TCP an die Position verfahren, die als Zielpunkt geteacht werden soll.
7. Anweisung mit **Befehl OK** speichern.

10.2.6 Inline-Formular CIRC

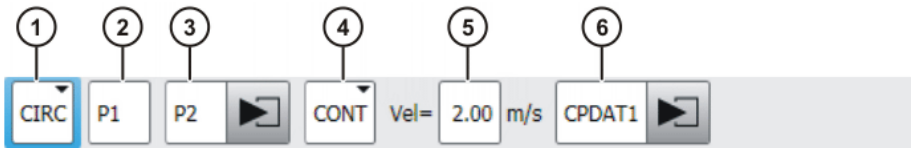


Abb. 10-3: Inline-Formular CIRC-Bewegung

Pos.	Beschreibung
1	Bewegungsart **CIRC**
2	Name des Hilfspunkts Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317)
3	Name des Zielpunkts Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
4	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
5	Geschwindigkeit ■ **0.001 … 2 m/s**
6	Name für den Bewegungsdatensatz Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.8 "Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)" Seite 321)

10.2.7 Optionsfenster Frames

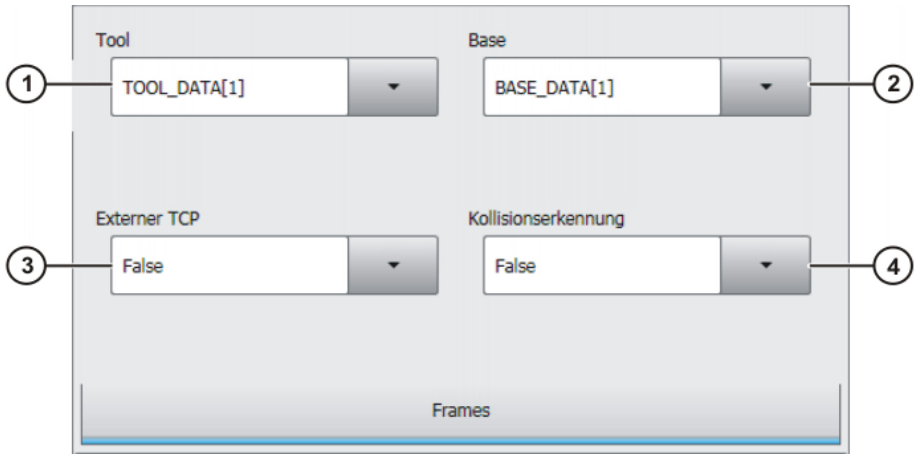


Abb. 10-4: Optionsfenster Frames

Pos.	Beschreibung
1	Werkzeug auswählen. Wenn **True** im Feld **Externer TCP**: Werkstück auswählen. Wertebereich: [1] … [16]
2	Basis auswählen. Wenn **True** im Feld **Externer TCP**: Feststehendes Werkzeug auswählen. Wertebereich: [1] … [32]
3	Interpolationsmodus ■ **False**: Das Werkzeug ist am Anbauflansch montiert. ■ **True**: Das Werkzeug ist ein feststehendes Werkzeug.
4	■ **True**: Für diese Bewegung ermittelt die Robotersteuerung die Achsmomente. Diese werden für die Kollisionserkennung benötigt. ■ **False**: Für diese Bewegung ermittelt die Robotersteuerung keine Achsmomente. Eine Kollisionserkennung ist für diese Bewegung daher nicht möglich.

10.2.8 Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)

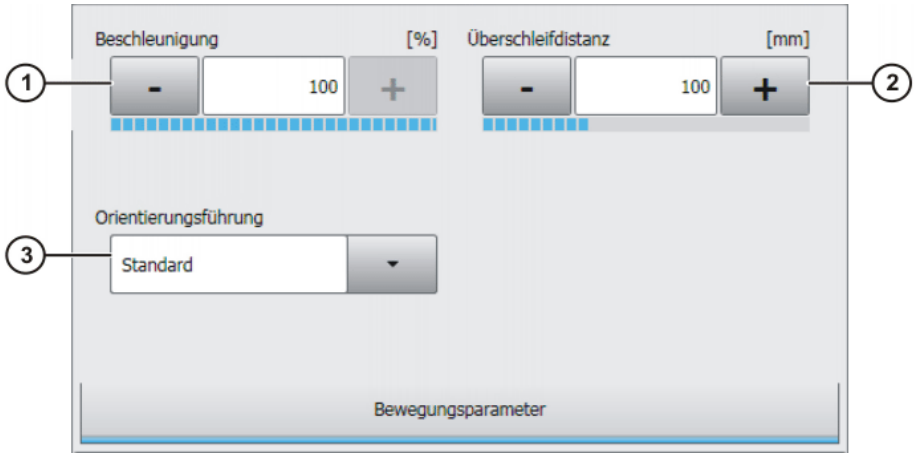


Abb. 10-5: Optionsfenster Bewegungsparameter (LIN, CIRC, PTP)

Pos.	Beschreibung
1	Beschleunigung Bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. Der Maximalwert ist abhängig vom Robotertyp und der eingestellten Betriebsart.

Pos.	Beschreibung
2	Dieses Feld wird nur angezeigt, wenn im Inline-Formular ausgewählt wurde, dass der Punkt überschliffen werden soll. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal die halbe Entfernung zwischen Startpunkt und Zielpunkt betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.
3	Dieses Feld wird nur für LIN- und CIRC-Bewegungen angezeigt. Orientierungsführung auswählen. ■ **Standard** ■ **Hand PTP** ■ **Konstante Orientierungsführung** (>>> 9.6 "Orientierungsführung LIN, CIRC" Seite 290)

10.3 Spline-Bewegungen programmieren

10.3.1 Programmiertipps für Spline-Bewegungen

- Die vollen Vorteile der Bewegungsart Spline können nur ausgeschöpft werden, wenn Spline-Blöcke verwendet werden.
- Interrupt-Programme dürfen keine Spline-Bewegungen enthalten.
- Ein Spline-Block soll nur einen Prozess umfassen (z. B. eine Klebenaht). Mehrere Prozesse in einem Spline-Block machen das Programm unübersichtlich und erschweren Änderungen.
- Wo durch das Werkstück Geraden und Kreisabschnitte vorgegeben sind, SLIN- und SCIRC-Segmente verwenden. (Ausnahme: Für sehr kurze Geraden SPL-Segmente verwenden.) Ansonsten SPL-Segmente verwenden, besonders bei kurzen Punktabständen.
- Vorgehensweise bei der Festlegung der Bahn:
  a. Zunächst wenige charakteristische Punkte teachen oder berechnen. Beispiel: Punkte, an denen die Krümmung umschlägt.
  b. Die Bahn testen. An den Stellen, an denen die Genauigkeit noch nicht ausreicht, weitere SPL-Punkte einfügen.
- Aufeinanderfolgende SLIN- und/oder SCIRC-Segmente vermeiden, da die Geschwindigkeit hierdurch häufig auf 0 reduziert wird.

  Zwischen SLIN- und SCIRC-Segmenten SPL-Segmente programmieren. Die Länge der SPL-Segmente muss mindestens > 0,5 mm sein. Abhängig vom konkreten Bahnverlauf können auch deutlich größere SPL-Segmente erforderlich sein.
- Aufeinanderfolgende Punkte mit gleichen kartesischen Koordinaten vermeiden, da die Geschwindigkeit hierdurch auf 0 reduziert wird.
- Die Parameter (Tool, Base, Geschwindigkeit etc.), die dem Spline-Block zugewiesen werden, wirken sich genauso aus wie Zuweisungen vor dem Spline-Block. Die Zuweisung zum Spline-Block hat jedoch den Vorteil, dass im Fall einer Satzanwahl die korrekten Parameter eingelesen werden.
- Wenn bei einem SLIN-, SCIRC- oder SPL-Segment keine bestimmte Orientierung erforderlich ist, die Option **Ohne Orientierung** verwenden. Die Robotersteuerung errechnet dann auf Grundlage der Orientierungen der umgebenden Punkte die optimale Orientierung für diesen Punkt. Dies verbessert die Taktzeit.

- Wenn Zusatzachsen vorhanden sind: Wenn bei einem Spline-Segment keine bestimmte Position der Zusatzachse erforderlich ist, $EX_AX_IGNORE für diese Zusatzachse auf "1" setzen. Die Robotersteuerung errechnet dann auf Grundlage der umgebenden Zusatzachs-Positionen die optimale Position für diesen Punkt. Dies verbessert die Taktzeit.
- Der Ruck kann verändert werden. Der Ruck ist die Beschleunigungsänderung. Vorgehensweise:
  a. Zunächst die Defaultwerte verwenden.
  b. Wenn Schwingungen an kleinen Ecken auftreten: Werte reduzieren.
     Wenn Geschwindigkeitseinbrüche auftreten oder die erwünschte Geschwindigkeit nicht erreicht wird: Werte erhöhen oder Beschleunigung erhöhen.
- Wenn der Roboter Punkte abfährt, die auf einer Arbeitsfläche liegen, kann es beim Anfahren des ersten Punkts zu einer Kollision mit der Arbeitsfläche kommen.

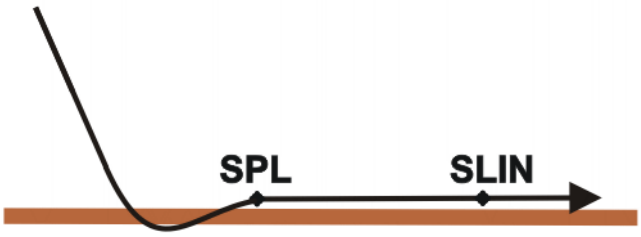


Abb. 10-6: Kollision mit Arbeitsfläche

Um eine Kollision zu vermeiden, die Empfehlungen für den SLIN-SPL-SLIN-Übergang beachten.
(>>> 9.7.5.1 "SLIN-SPL-SLIN-Übergang" Seite 304)

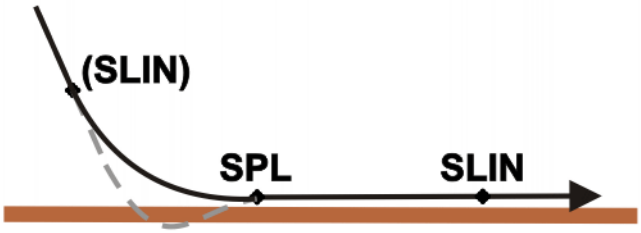


Abb. 10-7: Kollision mit Arbeitsfläche vermeiden

- Bei PTP-Spline-Blöcken mit mehreren SPTP-Segmenten kann es vorkommen, dass beim Programmlauf die Software-Endschalter verletzt werden, obwohl die Punkte innerhalb der Grenzen liegen!
  In diesem Fall müssen die Punkte umgeteacht werden, d. h. sie müssen weiter von den Software-Endschaltern entfernt werden. Alternativ können die Software-Endschalter geändert werden, unter der Voraussetzung, dass der erforderliche Maschinenschutz weiterhin gewährleistet ist.

10.3.2 Spline-Block programmieren

Beschreibung

Mit einem Spline-Block kann man mehrere Bewegungen zu einer Gesamtbewegung zusammenfassen. Die Bewegungen, die in einem Spline-Block stehen können, heißen Spline-Segmente. Sie werden einzeln geteacht.

Ein Spline-Block wird von der Robotersteuerung als 1 Bewegungssatz geplant und ausgeführt.

- Ein CP-Spline-Block darf SPL-, SLIN- und SCIRC-Segmente enthalten.
- Ein PTP-Spline-Block darf SPTP-Segmente enthalten.

Ein Spline-Block, der keine Segmente enthält, ist keine Bewegungsanweisung. Die Anzahl der Segmente im Block ist nur durch die Speicherkapazität begrenzt. Außer den Segmenten darf ein Spline-Block folgende Elemente enthalten:

- Inline-Befehle aus Technologiepaketen, die über die Spline-Funktionalität verfügen
- Kommentare und Leerzeilen

Ein Spline-Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.

Der Startpunkt eines Spline-Blocks ist der letzte Punkt vor dem Spline-Block.
Der Zielpunkt eines Spline-Blocks ist der letzte Punkt im Spline-Block.
Ein Spline-Block löst keinen Vorlaufstopp aus.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den Cursor in die Zeile setzen, nach der der Spline-Block eingefügt werden soll.
2. Menüfolge **Befehle** > **Bewegung** wählen.
   - Dann für einen CP-Spline-Block **SPLINE Block** wählen.
   - Oder für einen PTP-Spline-Block **PTP SPLINE Block** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.2.1 "Inline-Formular CP-Spline-Block" Seite 324)
   (>>> 10.3.2.2 "Inline-Formular PTP SPLINE Block" Seite 325)
4. **Befehl OK** drücken.
5. **Fold öffn/schl** drücken. Nun können Spline-Segmente in den Block eingefügt werden.

10.3.2.1 Inline-Formular CP-Spline-Block

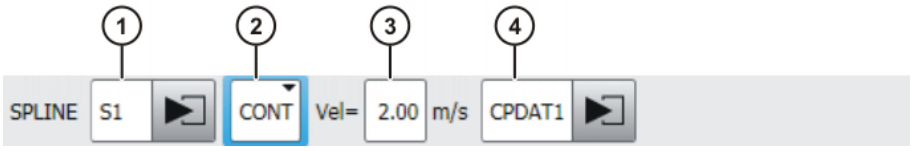


Abb. 10-8: Inline-Formular CP-Spline-Block

Pos.	Beschreibung
1	Name des Spline-Blocks. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Bewegungsdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.2.3 "Optionsfenster Frames (CP- und PTP-Spline-Block)" Seite 326)
2	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.

Pos.	Beschreibung
3	Kartesische Geschwindigkeit ■ **0.001 … 2 m/s**
4	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Bewegungsdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.2.4 "Optionsfenster Bewegungsparameter (CP-Spline-Block)" Seite 326)

10.3.2.2 Inline-Formular PTP SPLINE Block

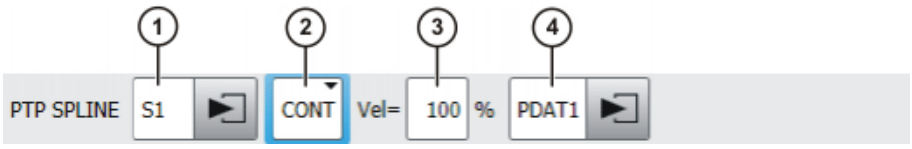


Abb. 10-9: Inline-Formular PTP SPLINE Block

Pos.	Beschreibung
1	Name des Spline-Blocks. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Bewegungsdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.2.3 "Optionsfenster Frames (CP- und PTP-Spline-Block)" Seite 326)
2	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
3	Achsgeschwindigkeit ■ **1 … 100 %**
4	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Bewegungsdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.2.5 "Optionsfenster Bewegungsparameter (PTP-Spline-Block)" Seite 327)

10.3.2.3 Optionsfenster Frames (CP- und PTP-Spline-Block)

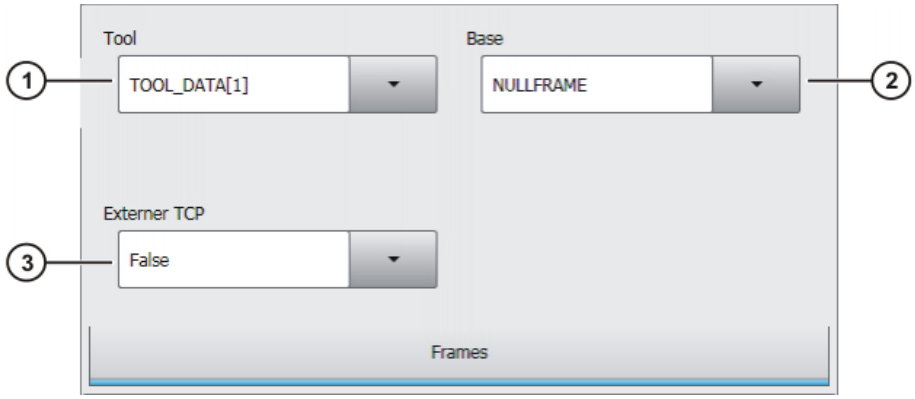


Abb. 10-10: Optionsfenster Frames (CP- und PTP-Spline-Block)

Pos.	Beschreibung
1	Werkzeug auswählen. Oder: Wenn **True** im Feld **Externer TCP**: Werkstück auswählen. ■ **[1]** … **[16]**
2	Basis auswählen. Oder: Wenn **True** im Feld **Externer TCP**: Feststehendes Werkzeug auswählen. ■ **[1]** … **[32]**
3	Interpolationsmodus ■ **False**: Das Werkzeug ist am Anbauflansch montiert. ■ **True**: Das Werkzeug ist ein feststehendes Werkzeug.

10.3.2.4 Optionsfenster Bewegungsparameter (CP-Spline-Block)

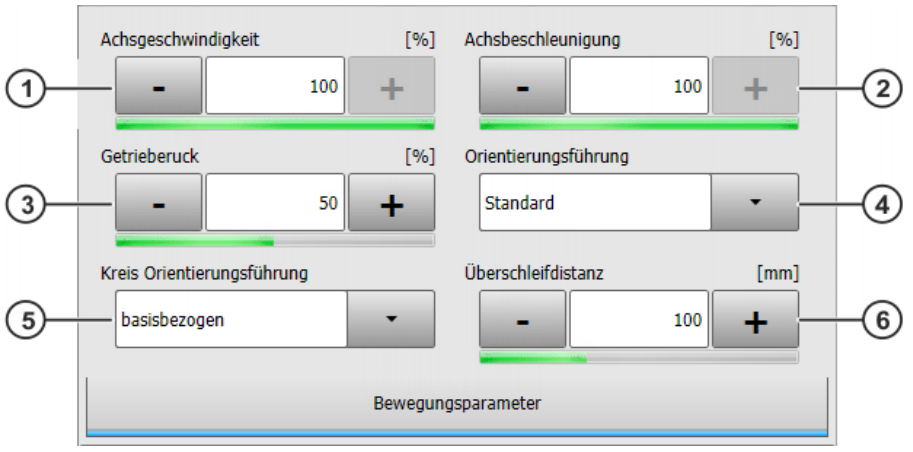


Abb. 10-11: Optionsfenster Bewegungsparameter (CP-Spline-Block)

Pos.	Beschreibung
1	Achsgeschwindigkeit. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
2	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
4	Orientierungsführung auswählen.
5	Bezugssystem der Orientierungsführung auswählen. Dieser Parameter wirkt sich nur auf SCIRC-Segmente aus (falls vorhanden).
6	Dieses Feld wird nur angezeigt, wenn im Inline-Formular **CONT** ausgewählt wurde. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal so groß sein wie das letzte Segment im Spline. Wenn nur ein Segment vorhanden ist, kann sie maximal die halbe Segmentlänge betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.

10.3.2.5 Optionsfenster Bewegungsparameter (PTP-Spline-Block)

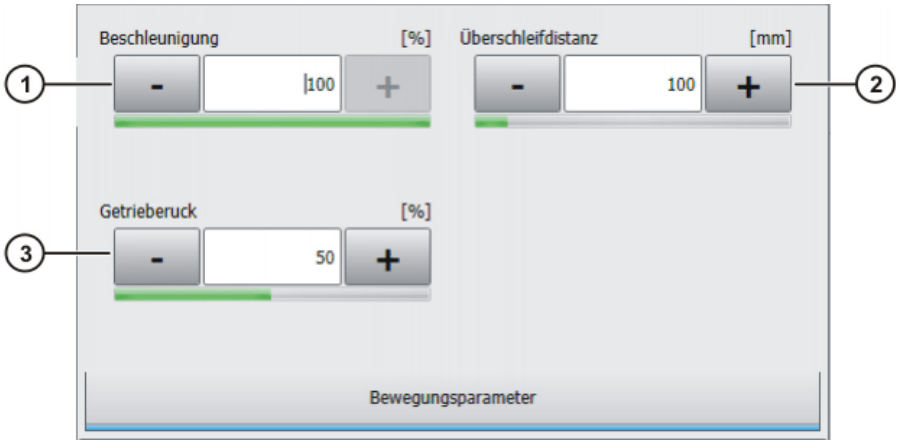


Abb. 10-12: Optionsfenster Bewegungsparameter (PTP-Spline-Block)

Pos.	Beschreibung
1	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
2	Dieses Feld wird nur angezeigt, wenn im Inline-Formular **CONT** ausgewählt wurde. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal so groß sein wie das letzte Segment im Spline. Wenn nur ein Segment vorhanden ist, kann sie maximal die halbe Segmentlänge betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**

10.3.3 Segmente für Spline-Block programmieren

10.3.3.1 SPL- oder SLIN-Segment programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1
- Der Fold des CP-Spline-Blocks ist geöffnet.

Vorgehensweise

1. Den TCP an den Zielpunkt verfahren.
2. Den Cursor in die Zeile im Spline-Block setzen, nach der das Segment eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **SPL** oder **SLIN** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.3.3 "Inline-Formular CP-Spline-Segment" Seite 329)
5. **Befehl OK** drücken.

10.3.3.2 SCIRC-Segment programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1
- Der Fold des CP-Spline-Blocks ist geöffnet.

Vorgehensweise

1. Den TCP an den Hilfspunkt verfahren.
2. Den Cursor in die Zeile im Spline-Block setzen, nach der das Segment eingefügt werden soll.

3. Menüfolge **Befehle** > **Bewegung** > **SCIRC** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.3.3 "Inline-Formular CP-Spline-Segment" Seite 329)
5. **Touchup HP** drücken.
6. Den TCP an den Zielpunkt verfahren.
7. **Befehl OK** drücken.

10.3.3.3 Inline-Formular CP-Spline-Segment

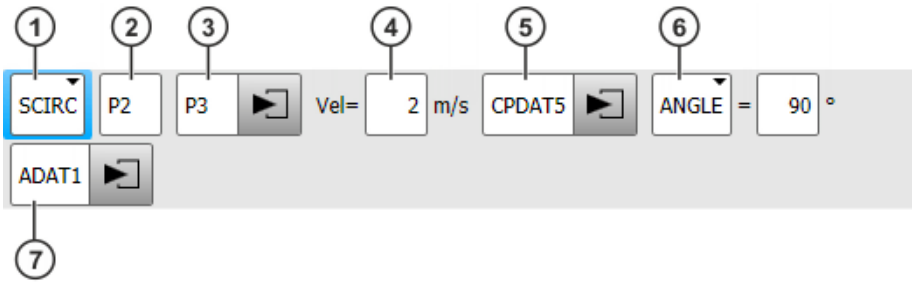


Abb. 10-13: Inline-Formular CP-Spline-Segment

Defaultmäßig werden nicht alle Felder des Inline-Formulars angezeigt. Über die Schaltfläche **Parameter wechseln** können die Felder ein- und ausgeblendet werden.

Pos.	Beschreibung
1	Bewegungsart ■ **SPL**, **SLIN** oder **SCIRC**
2	Nur bei **SCIRC**: Punktname für den Hilfspunkt Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317)
3	Punktname für den Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.6 "Optionsfenster Frames (CP- und PTP-Spline-Segmente)" Seite 331)
4	Kartesische Geschwindigkeit Defaultmäßig gilt für das Segment der für den Spline-Block gültige Wert. Bei Bedarf kann hier dem Segment gesondert ein Wert zugewiesen werden. Der Wert gilt nur für dieses Segment. ■ **0.001 … 2 m/s**
5	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Defaultmäßig gelten für das Segment die für den Spline-Block gültigen Werte. Bei Bedarf können hier dem Segment gesondert Werte zugewiesen werden. Die Werte gelten nur für dieses Segment. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.7 "Optionsfenster Bewegungsparameter (CP-Spline-Segment)" Seite 332)

Pos.	Beschreibung
6	Kreiswinkel Steht nur zur Verfügung, wenn die Bewegungsart **SCIRC** ausgewählt wurde. ■ **- 9 999° … + 9 999°** Wenn ein Wert kleiner - 400° oder größer + 400° eingegeben wird, öffnet sich beim Speichern des Inline-Formulars eine Abfrage, in der die Eingabe bestätigt oder verworfen werden muss.
7	Name für den Datensatz mit Logikparametern. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.9 "Optionsfenster Logikparameter" Seite 334)

10.3.3.4 SPTP-Segment programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1
- Der Fold des PTP-Spline-Blocks ist geöffnet.

Vorgehensweise

1. Den TCP an den Zielpunkt verfahren.
2. Den Cursor in die Zeile im Spline-Block setzen, nach der das Segment eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **SPTP** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.3.5 "Inline-Formular SPTP-Segment" Seite 330)
5. **Befehl OK** drücken.

10.3.3.5 Inline-Formular SPTP-Segment

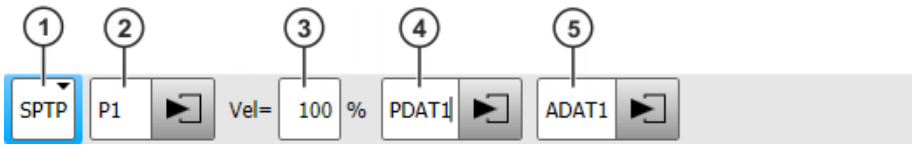


Abb. 10-14: Inline-Formular SPTP-Segment

Defaultmäßig werden nicht alle Felder des Inline-Formulars angezeigt. Über die Schaltfläche **Parameter wechseln** können die Felder ein- und ausgeblendet werden.

Pos.	Beschreibung
1	Bewegungsart **SPTP**
2	Punktname für Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.6 "Optionsfenster Frames (CP- und PTP-Spline-Segmente)" Seite 331)
3	Achsgeschwindigkeit Defaultmäßig gilt für das Segment der für den Spline-Block gültige Wert. Bei Bedarf kann hier dem Segment gesondert ein Wert zugewiesen werden. Der Wert gilt nur für dieses Segment. ■ **1 … 100 %**
4	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Defaultmäßig gelten für das Segment die für den Spline-Block gültigen Werte. Bei Bedarf können hier dem Segment gesondert Werte zugewiesen werden. Die Werte gelten nur für dieses Segment. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.8 "Optionsfenster Bewegungsparameter (SPTP)" Seite 333)
5	Name für den Datensatz mit Logikparametern. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.9 "Optionsfenster Logikparameter" Seite 334)

10.3.3.6 Optionsfenster Frames (CP- und PTP-Spline-Segmente)

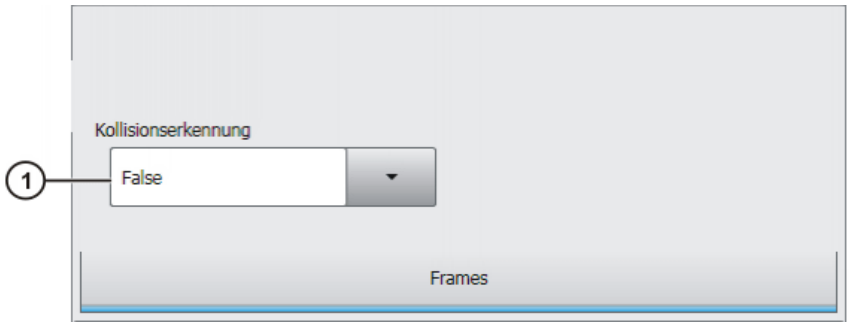


Abb. 10-15: Optionsfenster Frames (CP- und PTP-Spline-Segmente)

Pos.	Beschreibung
1	■ **True**: Für diese Bewegung ermittelt die Robotersteuerung die Achsmomente. Diese werden für die Kollisionserkennung benötigt. ■ **False**: Für diese Bewegung ermittelt die Robotersteuerung keine Achsmomente. Eine Kollisionserkennung ist für diese Bewegung daher nicht möglich.

10.3.3.7 Optionsfenster Bewegungsparameter (CP-Spline-Segment)

Bewegungsparameter

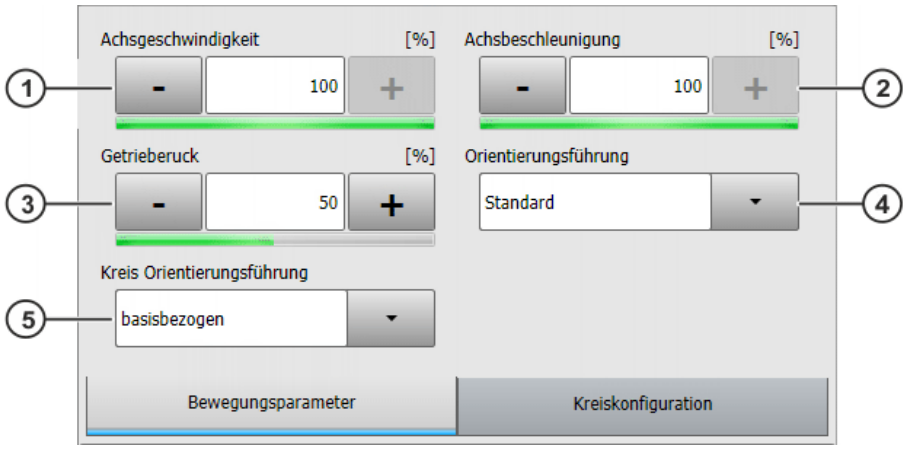


Abb. 10-16: Optionsfenster Bewegungsparameter (CP-Spline-Segment)

Pos.	Beschreibung
1	Achsgeschwindigkeit. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
2	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
4	Orientierungsführung auswählen
5	Nur bei SCIRC-Segmenten: Bezugssystem der Orientierungsführung auswählen.

Kreiskonfiguration

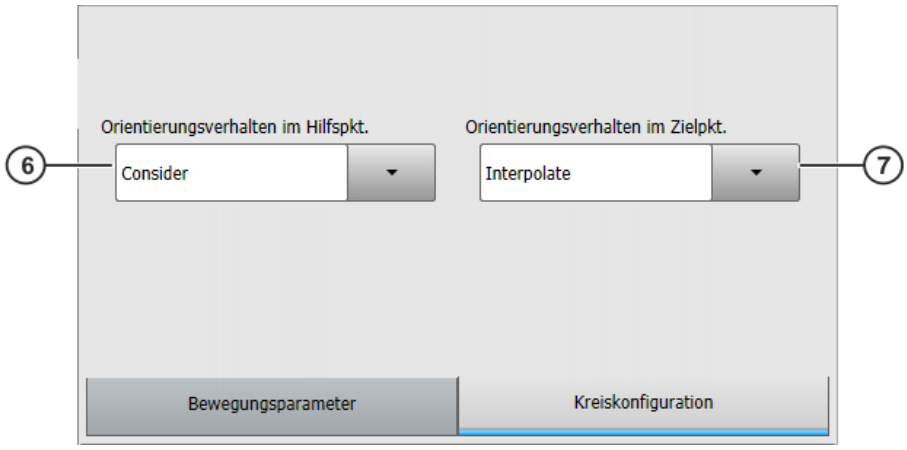


Abb. 10-17: Kreiskonfiguration (SCIRC-Segment)

Pos.	Beschreibung
6	Nur bei SCIRC-Segmenten: Orientierungsverhalten im Hilfspunkt auswählen
7	Nur bei SCIRC-Segmenten: Dieses Feld wird nur angezeigt, wenn im Inline-Formular **ANGLE** ausgewählt wurde. Orientierungsverhalten im Zielpunkt auswählen

10.3.3.8 Optionsfenster Bewegungsparameter (SPTP)

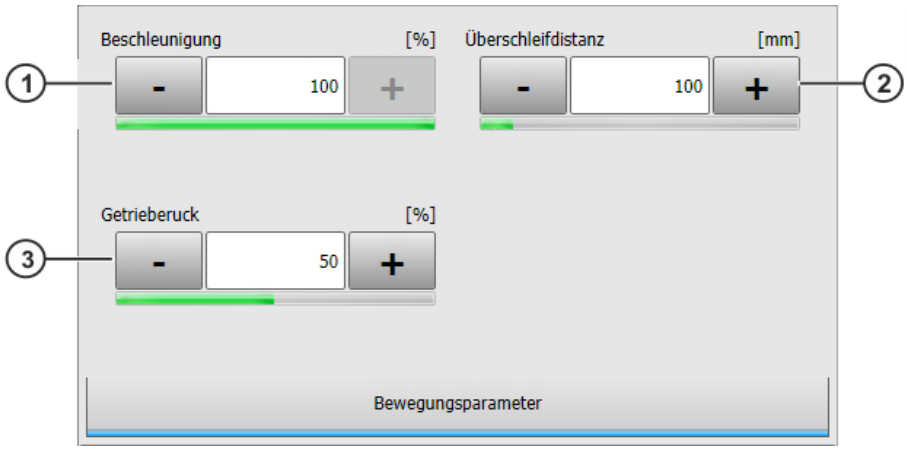


Abb. 10-18: Optionsfenster Bewegungsparameter (SPTP)

Pos.	Beschreibung
1	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**

Pos.	Beschreibung
2	Dieses Feld steht für SPTP-Segmente nicht zur Verfügung. Bei SPTP-Einzelbewegungen wird dieses Feld nur angezeigt, wenn im Inline-Formular **CONT** ausgewählt wurde. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal die halbe Entfernung zwischen Startpunkt und Zielpunkt betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**

10.3.3.9 Optionsfenster Logikparameter

Trigger **Trigger**

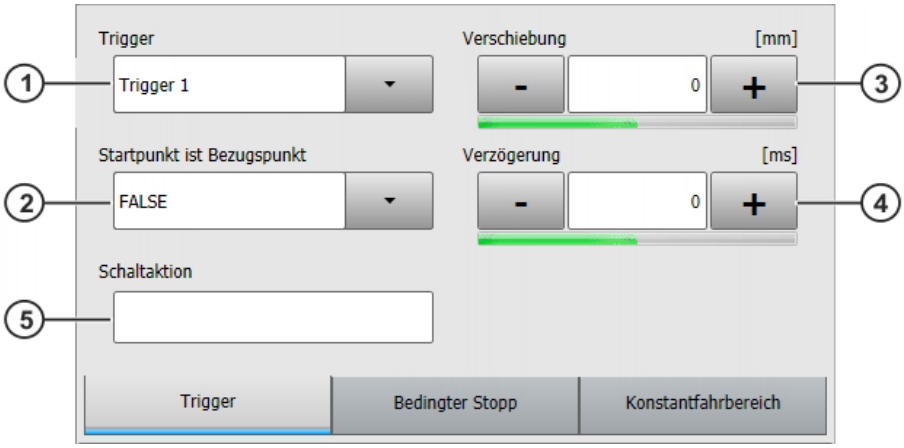


Abb. 10-19: Trigger

Pos.	Beschreibung
1	Über die Schaltfläche **Aktion wählen** > **Trigger hinzufügen** kann man hier der Bewegung einen (weiteren) Trigger zuordnen. Wenn es der erste Trigger für diese Bewegung ist, blendet dieser Befehl auch das Feld **Trigger** ein. Pro Bewegung sind max. 8 Trigger möglich. (Ein Trigger kann über **Aktion wählen** > **Trigger entfernen** wieder entfernt werden.)
2	Bezugspunkt des Triggers ■ **TRUE**: Startpunkt ■ **FALSE**: Zielpunkt (>>> 11.11.2.1 "Bezugspunkt beim Überschleifen – Übersicht" Seite 444)

Pos.	Beschreibung
3	Örtliche Verschiebung in Bezug auf den Ziel- oder Startpunkt ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang ■ Positiver Wert: Verschiebung in Richtung Bewegungsende (>>> "Max. Verschiebung" Seite 441) Die örtliche Verschiebung kann auch geteacht werden. Wenn dies geschieht, wird das Feld **Startpunkt ist Bezugspunkt** automatisch auf **FALSE** gesetzt. (>>> 10.3.3.10 "Örtliche Verschiebung für Logikparameter teachen" Seite 337)
4	Zeitliche Verschiebung in Bezug auf **Verschiebung** ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang. ■ Positiver Wert: Trigger wird nach Ablauf von *Zeit* geschaltet. (>>> "Max. Verschiebung" Seite 441)
5	Anweisung, die der Trigger auslösen soll. Möglich sind: ■ Wertzuweisung an eine Variable **Hinweis:** Auf der linken Seite der Zuweisung darf keine Laufzeit-Variable stehen. ■ OUT-Anweisung; PULSE-Anweisung; CYCFLAG-Anweisung ■ Aufruf eines Unterprogramms. In diesem Fall muss die Priorität angegeben werden. Beispiel: `my_subprogram() PRIO = 81` Zur Verfügung stehen die Prioriäten 1, 2, 4 - 39 sowie 81 - 128. Die Prioriäten 40 - 80 sind reserviert für Fälle, in denen die Priorität automatisch durch das System vergeben wird. Wenn die Priorität automatisch durch das System vergeben werden soll, programmiert man: `PRIO = -1`. Wenn mehrere Trigger gleichzeitig Unterprogramme aufrufen, wird zuerst der Trigger mit der höchsten Priorität bearbeitet, dann die Trigger mit niedrigerer Priorität. 1 = höchste Priorität.

Bedingter Stopp

Bedingter Stopp

In dieser Dokumentation sind weiterführende Informationen zum Bedingten Stopp zu finden.
(>>> 10.3.5 "Bedingter Stopp" Seite 344)

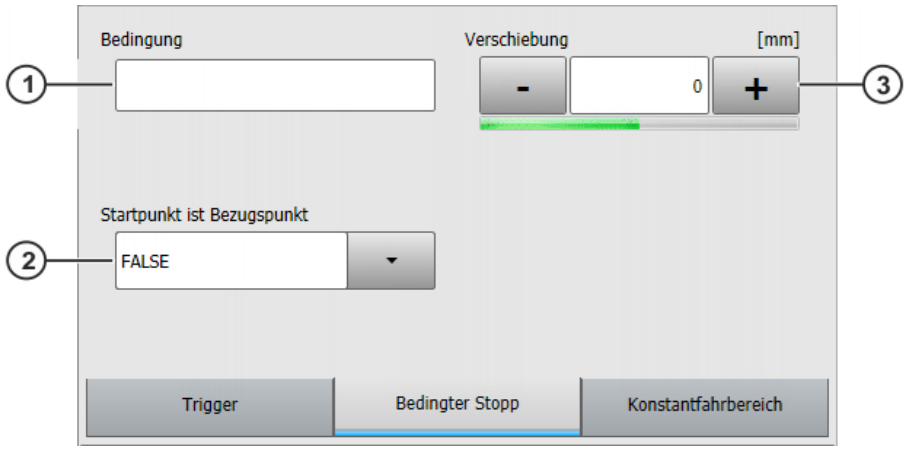


Abb. 10-20: Bedingter Stopp

Pos.	Beschreibung
1	Stopp-Bedingung. Zulässig sind: ■ eine globale boolesche Variable ■ ein Signalname ■ ein Vergleich ■ eine einfache logische Verknüpfung: NOT, OR, AND oder EXOR
2	Der Bedingte Stopp kann sich entweder auf den Startpunkt oder auf den Zielpunkt der Bewegung beziehen. ■ **TRUE**: Startpunkt ■ **FALSE**: Zielpunkt Wenn der Bezugspunkt überschliffen ist, gelten die gleichen Regeln wie beim PATH-Trigger. (>>> 11.11.2.1 "Bezugspunkt beim Überschleifen – Übersicht" Seite 444)
3	Der Stopp-Punkt kann örtlich verschoben werden. Dazu muss hier die gewünschte Entfernung zum Start- bzw. Zielpunkt angegeben werden. Wenn keine örtliche Verschiebung gewünscht ist, "0" eintragen. ■ Positiver Wert: Verschiebung in Richtung Bewegungsende ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang Der Stopp-Punkt kann nicht beliebig weit verschoben werden. Es gelten die gleichen Grenzen wie beim PATH-Trigger. (>>> "Max. Verschiebung" Seite 441) Die örtliche Verschiebung kann auch geteacht werden. Wenn dies geschieht, wird das Feld **Startpunkt ist Bezugspunkt** automatisch auf **FALSE** gesetzt. (>>> 10.3.3.10 "Örtliche Verschiebung für Logikparameter teachen" Seite 337)

Konstantfahrbereich

Konstantfahrbereich

Konstantfahrbereich steht nur für CP-Spline-Segmente zur Verfügung.

In dieser Dokumentation sind weiterführende Informationen zu Konstantfahrbereichen zu finden.
(>>> 10.3.6 "Konstantfahrbereich im CP-Spline-Block" Seite 347)

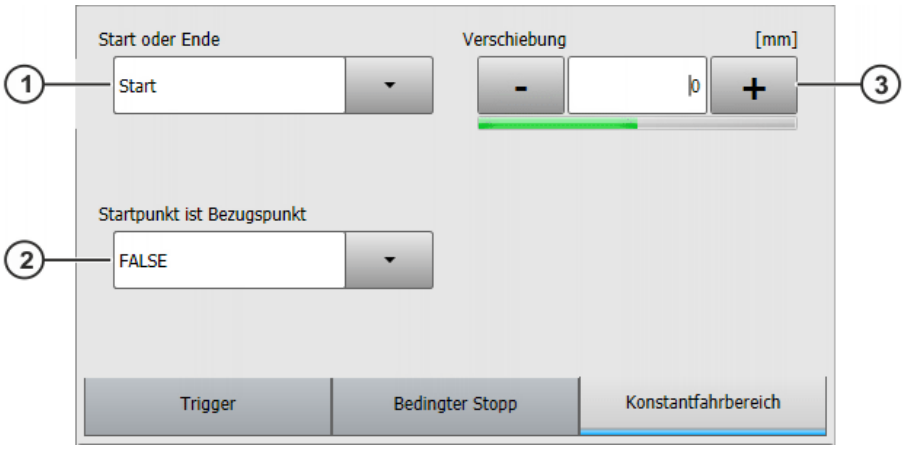


Abb. 10-21: Konstantfahrbereich

Pos.	Beschreibung
1	■ **Start**: Legt den Anfang des Konstantfahrbereiches fest. ■ **End**: Legt das Ende des Konstantfahrbereiches fest.
2	**Start** bzw. **End** kann sich entweder auf den Startpunkt oder auf den Zielpunkt der Bewegung beziehen. ■ **TRUE**: **Start** bzw. **End** bezieht sich auf den Startpunkt. Wenn der Startpunkt überschliffen ist, ergibt sich der Bezugspunkt auf die gleiche Weise wie beim homogenen Überschleifen beim PATH-Trigger. (>>> 11.11.2.2 "Bezugspunkt beim homogenen Überschleifen" Seite 444) ■ **FALSE**: **Start** bzw. **End** bezieht sich auf den Zielpunkt. Wenn der Zielpunkt überschliffen ist, bezieht sich **Start** bzw. **End** auf den Anfang des Überschleifbogens.
3	Der Anfang bzw. das Ende des Konstantfahrbereichs kann örtlich verschoben werden. Dazu muss hier die gewünschte Strecke angegeben werden. Wenn keine örtliche Verschiebung gewünscht ist, "0" eintragen. ■ Positiver Wert: Verschiebung in Richtung Bewegungsende ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang (>>> 10.3.6.2 "Maximale Grenzen" Seite 349) Die örtliche Verschiebung kann auch geteacht werden. Wenn dies geschieht, wird das Feld **Startpunkt ist Bezugspunkt** automatisch auf **FALSE** gesetzt. (>>> 10.3.3.10 "Örtliche Verschiebung für Logikparameter teachen" Seite 337)

10.3.3.10 Örtliche Verschiebung für Logikparameter teachen

Beschreibung

Im Optionsfenster **Logikparameter** können für Trigger, Bedingten Stopp und Konstantfahrbereich örtliche Verschiebungen angegeben werden. Statt diese Verschiebungen numerisch anzugeben, können sie auch geteacht werden.

Wenn eine Verschiebung geteacht wird, wird das Feld **Startpunkt ist Bezugspunkt** in der jeweiligen Registerkarte automatisch auf **FALSE** gesetzt, da sich die geteachte Entfernung auf den Zielpunkt der Bewegung bezieht.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1
- Der Punkt, für den die Verschiebung gelten soll, wurde bereits geteacht.

Vorgehensweise

1. Die gewünschte Position mit dem TCP anfahren.
2. Den Cursor in die Zeile mit der Bewegungsanweisung setzen, für die die Verschiebung geteacht werden soll.
3. Auf **Ändern** drücken. Das Inline-Formular zur Anweisung öffnet sich.
4. Das Optionsfenster **Logikparameter** öffnen und die benötigte Registerkarte wählen.
5. Auf **Aktion wählen** drücken, dann auf eine der folgenden Schaltflächen drücken, je nachdem, für was die Verschiebung geteacht werden soll:
   - **Trigger Path aufnehmen**
   - **Bedingter Stopp Path aufnehmen**
   - **Konstantfahr-Bereich Path aufnehmen**

   Die Entfernung vom Zielpunkt der aktuellen Bewegungsanweisung wird nun als Wert für die örtliche Verschiebung übernommen.
6. Die Änderung mit **Befehl OK** speichern.

10.3.4 Spline-Einzelbewegungen programmieren

10.3.4.1 SLIN-Einzelbewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an den Zielpunkt verfahren.
2. Den Cursor in die Zeile setzen, nach der die Bewegung eingefügt werden soll.
3. **Befehle** > **Bewegung** > **SLIN** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.4.2 "Inline-Formular SLIN" Seite 338)
5. **Befehl OK** drücken.

10.3.4.2 Inline-Formular SLIN

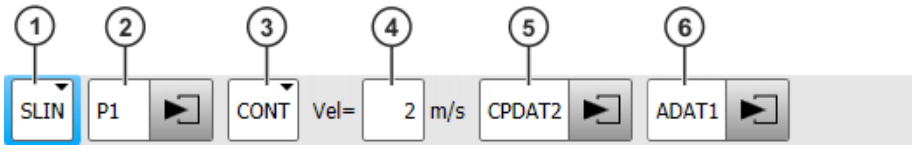


Abb. 10-22: Inline-Formular SLIN (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart **SLIN**
2	Punktname für Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
3	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
4	Geschwindigkeit ■ **0.001 … 2 m/s**
5	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.4.3 "Optionsfenster Bewegungsparameter (SLIN)" Seite 339)
6	Dieses Feld kann über **Parameter wechseln** ein- und ausgeblendet werden. Name für den Datensatz mit Logikparametern. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.9 "Optionsfenster Logikparameter" Seite 334)

10.3.4.3 Optionsfenster Bewegungsparameter (SLIN)

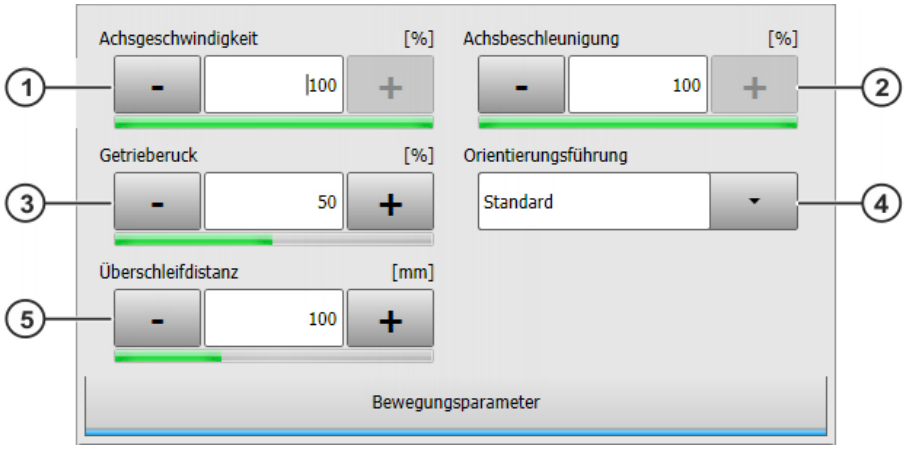


Abb. 10-23: Optionsfenster Bewegungsparameter (SLIN)

Pos.	Beschreibung
1	Achsgeschwindigkeit. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
2	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**

Pos.	Beschreibung
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
4	Orientierungsführung auswählen.
5	Dieses Feld wird nur angezeigt, wenn im Inline-Formular **CONT** ausgewählt wurde. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal die halbe Entfernung zwischen Startpunkt und Zielpunkt betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.

10.3.4.4 SCIRC-Einzelbewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an den Hilfspunkt verfahren.
2. Den Cursor in die Zeile setzen, nach der die Bewegung eingefügt werden soll.
3. Menüfolge **Befehle** > **Bewegung** > **SCIRC** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.4.5 "Inline-Formular SCIRC" Seite 340)
5. **Touchup HP** drücken.
6. Den TCP an den Zielpunkt verfahren.
7. **Befehl OK** drücken.

10.3.4.5 Inline-Formular SCIRC

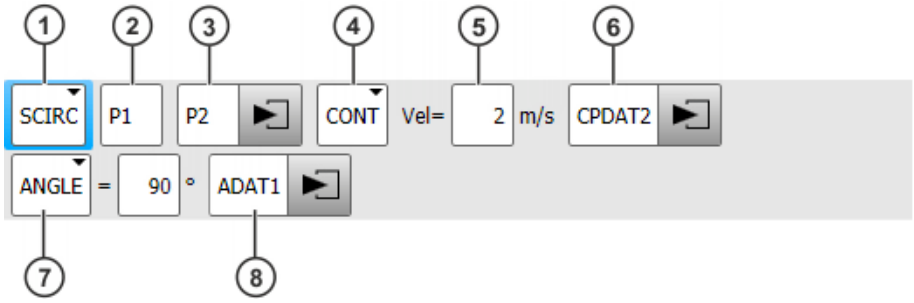


Abb. 10-24: Inline-Formular SCIRC (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart **SCIRC**
2	Punktname für den Hilfspunkt Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317)
3	Punktname für den Zielpunkt Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
4	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
5	Geschwindigkeit ■ **0.001 … 2 m/s**
6	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.4.6 "Optionsfenster Bewegungsparameter (SCIRC)" Seite 342)
7	Kreiswinkel ■ **- 9 999° … + 9 999°** Wenn ein Kreiswinkel kleiner - 400° oder größer + 400° eingegeben wird, öffnet sich beim Speichern des Inline-Formulars eine Abfrage, in der die Eingabe bestätigt oder verworfen werden muss.
8	Dieses Feld kann über **Parameter wechseln** ein- und ausgeblendet werden. Name für den Datensatz mit Logikparametern. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.9 "Optionsfenster Logikparameter" Seite 334)

10.3.4.6 Optionsfenster Bewegungsparameter (SCIRC)

Bewegungsparameter

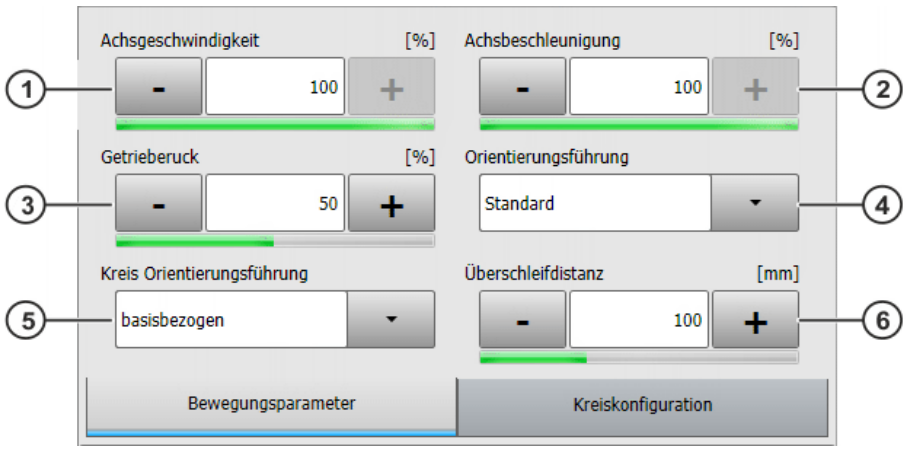


Abb. 10-25: Bewegungsparameter (SCIRC)

Pos.	Beschreibung
1	Achsgeschwindigkeit. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
2	Achsbeschleunigung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
3	Getrieberuck. Der Ruck ist die Beschleunigungsänderung. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. ■ **1 … 100 %**
4	Orientierungsführung auswählen
5	Bezugssystem der Orientierungsführung auswählen.
6	Dieses Feld wird nur angezeigt, wenn im Inline-Formular **CONT** ausgewählt wurde. Distanz vor dem Zielpunkt, bei der das Überschleifen frühestens beginnt Die Distanz kann maximal die halbe Entfernung zwischen Startpunkt und Zielpunkt betragen. Wenn hier ein höherer Wert eingetragen wird, wird dieser ignoriert und der Maximalwert verwendet.

Kreiskonfiguration

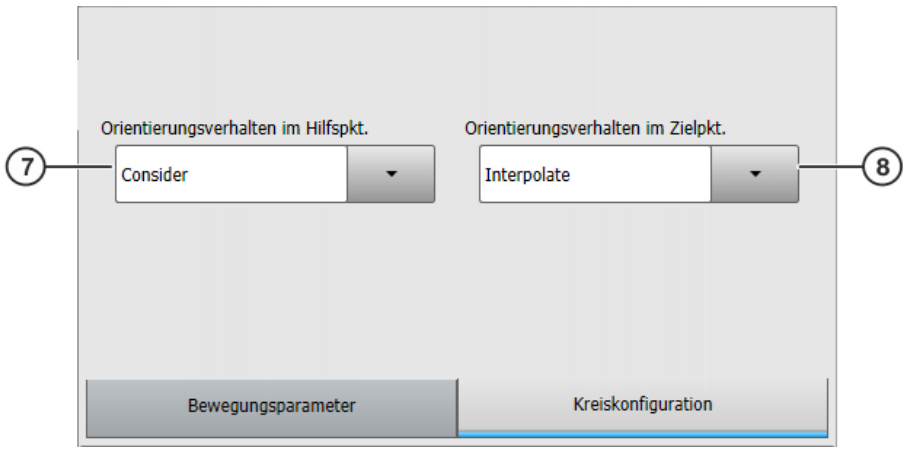


Abb. 10-26: Kreiskonfiguration (SCIRC)

Pos.	Beschreibung
7	Orientierungsverhalten im Hilfspunkt auswählen
8	Dieses Feld wird nur angezeigt, wenn im Inline-Formular **ANGLE** ausgewählt wurde. Orientierungsverhalten im Zielpunkt auswählen

10.3.4.7 SPTP-Einzelbewegung programmieren

HINWEIS Beim Programmieren von Bewegungen ist darauf zu achten, dass sich die Energiezuführung beim Programmablauf nicht aufwickelt oder beschädigt wird.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Den TCP an den Zielpunkt verfahren.
2. Den Cursor in die Zeile setzen, nach der die Bewegung eingefügt werden soll.
3. **Befehle** > **Bewegung** > **SPTP** wählen.
4. Im Inline-Formular die Parameter einstellen.
   (>>> 10.3.4.8 "Inline-Formular SPTP" Seite 343)
5. **Befehl OK** drücken.

10.3.4.8 Inline-Formular SPTP

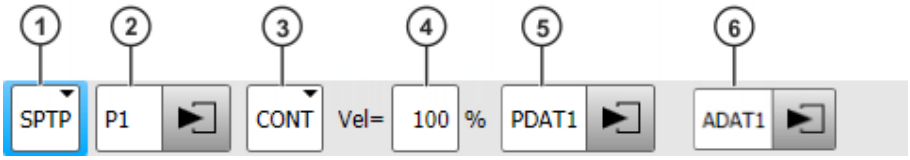


Abb. 10-27: Inline-Formular SPTP (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart **SPTP**
2	Punktname für Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. (>>> 10.1 "Namen in Inline-Formularen" Seite 317) Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.2.7 "Optionsfenster Frames" Seite 320)
3	■ **CONT**: Zielpunkt wird überschliffen. ■ **[leer]**: Zielpunkt wird genau angefahren.
4	Geschwindigkeit ■ **1 … 100 %**

Pos.	Beschreibung
5	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Punktdaten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.8 "Optionsfenster Bewegungsparameter (SPTP)" Seite 333)
6	Dieses Feld kann über **Parameter wechseln** ein- und ausgeblendet werden. Name für den Datensatz mit Logikparametern. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. (>>> 10.3.3.9 "Optionsfenster Logikparameter" Seite 334)

10.3.5 Bedingter Stopp

Beschreibung

Der "Bedingte Stopp" ermöglicht es dem Benutzer, eine Stelle auf der Bahn zu definieren, an der der Roboter stoppt, falls eine bestimmte Bedingung erfüllt ist. Die Stelle wird "Stopp-Punkt" genannt. Sobald die Bedingung nicht mehr erfüllt ist, fährt der Roboter wieder weiter.

Die Robotersteuerung errechnet während der Laufzeit den Punkt, an dem sie spätestens bremsen muss, um am Stopp-Punkt stoppen zu können. Ab diesem Punkt (= "Bremspunkt") wertet sie aus, ob die Bedingung erfüllt ist oder nicht.

- Wenn die Bedingung am Bremspunkt erfüllt ist, bremst der Roboter, um am Stopp-Punkt zu stoppen.
  Falls jedoch die Bedingung dann vor Erreichen des Stopp-Punkts wieder auf "nicht erfüllt" wechselt, beschleunigt der Roboter wieder und stoppt nicht.
- Wenn die Bedingung am Bremspunkt nicht erfüllt ist, fährt der Roboter weiter, ohne zu bremsen.

Grundsätzlich können beliebig viele Bedingte Stopps programmiert werden. Es dürfen sich jedoch maximal 10 Strecken "Bremspunkt → Stopp-Punkt" überschneiden.

Während eines Bremsvorgangs zeigt die Robotersteuerung in T1/T2 folgende Meldung an: *Bedingter Stopp aktiv (Zeile {Zeilennummer}).*

(>>> 10.3.5.2 "Stopp-Bedingung: Beispiel und Bremsverhalten" Seite 346)

Programmierung

Programmierung mit KRL-Syntax:

- über die Anweisung STOP WHEN PATH

Programmierung über Inline-Formulare:

- Im Spline-Block (CP und PTP) oder im Spline-Einzelsatz:
  im Optionsfenster **Logikparameter**
- Vor einem Spline-Block (CP und PTP):
  über das Inline-Formular **Spline Stop Condition**

10.3.5.1 Inline-Formular Spline Stop Condition

Dieses Inline-Formular darf nur vor einem Spline-Block verwendet werden. Zwischen dem Inline-Formular und dem Spline-Block dürfen weitere Anweisungen stehen, jedoch keine Bewegungsanweisungen.

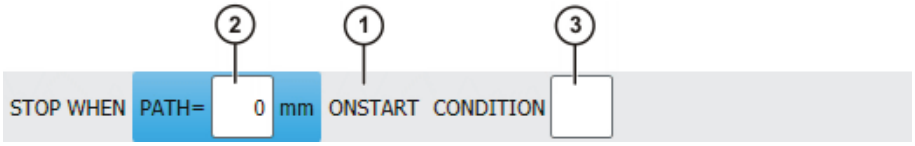


Abb. 10-28: Inline-Formular Spline Stop Condition

Pos.	Beschreibung
1	Punkt, auf den sich der Bedingte Stopp bezieht ■ mit ONSTART: letzter Punkt vor dem Spline-Block ■ ohne ONSTART: letzter Punkt im Spline-Block Wenn der Spline überschliffen ist, gelten die gleichen Regeln wie beim PATH-Trigger. **Hinweis:** Informationen zum Überschleifen beim PATH-Trigger sind in der Bedien-/Programmieranleitung für Systemintegratoren zu finden. ONSTART kann über die Schaltfläche **Umsch. OnStart** gesetzt oder entfernt werden.
2	Der Stopp-Punkt kann örtlich verschoben werden. Dazu muss hier die gewünschte Entfernung zum Bezugspunkt angegeben werden. Wenn keine örtliche Verschiebung gewünscht ist, "0" eintragen. ■ Positiver Wert: Verschiebung in Richtung Bewegungsende ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang Der Stopp-Punkt kann nicht beliebig weit verschoben werden. Es gelten die gleichen Grenzen wie beim PATH-Trigger. Die örtliche Verschiebung kann auch geteacht werden. (>>> "Path aufnehmen" Seite 346)
3	Stopp-Bedingung. Zulässig sind: ■ eine globale boolesche Variable ■ ein Signalname ■ ein Vergleich ■ eine einfache logische Verknüpfung: NOT, OR, AND oder EXOR

Path aufnehmen

Schaltfläche	Beschreibung
Path aufnehmen	Wenn eine Verschiebung gewünscht ist, muss der Wert nicht unbedingt numerisch ins Inline-Formular eingegeben werden, sondern die Verschiebung kann auch geteacht werden. Dies geschieht über **Path aufnehmen**. Wenn eine Verschiebung geteacht wird, wird ON-START, falls im Inline-Formular gesetzt, automatisch entfernt, da sich die geteachte Entfernung immer auf den Zielpunkt der Bewegung bezieht. Der Ablauf beim Teachen ist analog zum Ablauf für das Optionsfenster **Logikparameter**. (>>> 10.3.3.10 "Örtliche Verschiebung für Logikparameter teachen" Seite 337)

10.3.5.2 Stopp-Bedingung: Beispiel und Bremsverhalten

Beispiel

Die Einrückungen sind defaultmäßig nicht vorhanden und wurden hier der besseren Übersicht wegen eingefügt.

```
PTP P0 Vel=100 % PDAT1 Tool[1] Base[0]

SPLINE S1 Vel=2 m/s CPDAT1 Tool[1] Base[0]

SPL P1 ADAT1

  STOP WHEN PATH = 50 IF $in[77]==FALSE

  SPL XP1

SPL P2

SPL P3

ENDSPLINE
```

Abb. 10-29: Beispiel Inline-Programmierung (Folds aufgeklappt)

Zeile	Beschreibung
4	Wenn der Eingang $IN[77] FALSE ist, stoppt der Roboter 50 mm nach P2 und wartet, bis $IN[77] TRUE wird.

Abb. 10-30: STOP WHEN PATH Beispiel

Punkt	Beschreibung
BP	Bremspunkt (**B**rake **P**oint): Hier muss der Roboter anfangen zu bremsen, damit er am Stopp-Punkt zum Stehen kommt. Ab diesem Punkt wertet die Robotersteuerung aus, ob die Stopp-Bedingung erfüllt ist oder nicht. Die Position von **BP** ist abhängig von der Geschwindigkeit und vom Override und ist für den Benutzer nicht erkennbar.
SP	Stopp-Punkt (**S**top **P**oint) Die Strecke **P2** → **SP** ist 50 mm lang.

Bremsverhalten

Situation am BP	Verhalten des Roboters
$IN[77] == FALSE	Der Roboter bremst und stoppt am **SP**.
$IN[77] == TRUE	Der Roboter bremst nicht und stoppt am **SP** nicht. Das Programm wird abgefahren, wie wenn die Anweisung STOP WHEN PATH nicht vorhanden wäre.
1. Am **BP** ist $IN[77] == FALSE. 2. Zwischen **BP** und **SP** wechselt der Eingang auf TRUE.	1. Der Roboter bremst am **BP**. 2. Wenn der Eingang TRUE wird, beschleunigt der Roboter wieder und stoppt nicht am **SP**.
1. Am **BP** ist $IN[77] == TRUE. 2. Zwischen **BP** und **SP** wechselt der Eingang auf FALSE.	1. Der Roboter bremst am **BP** nicht. 2. Wenn der Eingang FALSE wird, stoppt der Roboter mit einem bahntreuen NOT-HALT und kommt an einem nicht vorhersehbaren Punkt zum Stehen.

Wenn sich die Bedingung für den Stopp erst erfüllt, wenn der Roboter den **BP** bereits passiert hat, ist es zu spät, um mit einer normalen Bremsrampe am **SP** anzuhalten. Der Roboter stoppt in diesem Fall mit einem bahntreuen NOT-HALT und kommt an einem nicht vorhersehbaren Punkt zum Stehen.

- Falls der Roboter durch den NOT-HALT nach dem **SP** zum Stehen kommt, kann das Programm erst fortgesetzt werden, wenn die Stopp-Bedingung nicht mehr erfüllt ist.
- Falls der Roboter durch den bahntreuen NOT-HALT vor dem **SP** zum Stehen kommt, geschieht Folgendes, wenn das Programm fortgesetzt wird:
  - Wenn die Stopp-Bedingung nicht mehr erfüllt ist: Der Roboter fährt weiter.
  - Wenn die Stopp-Bedingung noch erfüllt ist: Der Roboter fährt bis zum **SP** und bleibt dort stehen.

10.3.6 Konstantfahrbereich im CP-Spline-Block

Beschreibung

In einem CP-Spline-Block kann ein Bereich definiert werden, in dem der Roboter die programmierte Geschwindigkeit konstant hält, sofern möglich. Der Bereich wird "Konstantfahrbereich" genannt.

- Pro CP-Spline-Block kann 1 Konstantfahrbereich definiert werden.
- Ein Konstantfahrbereich ist definiert durch eine Start-Anweisung und eine Ende-Anweisung.
- Der Bereich kann sich nicht über den Spline-Block hinaus erstrecken.

- Der Bereich kann beliebig klein sein.

Wenn es nicht möglich ist, die programmierte Geschwindigkeit konstant zu halten, zeigt die Robotersteuerung dies beim Programmlauf durch eine Meldung an.

Konstantfahrbereich über mehrere Segmente:

Ein Konstantfahrbereich kann sich über mehrere Segmente mit verschiedenen programmierten Geschwindigkeiten erstrecken. In diesem Fall gilt die niedrigste der Geschwindigkeiten für den gesamten Bereich.

Auch in den Segmenten mit höherer programmierter Geschwindigkeit wird in diesem Fall mit der niedrigsten Geschwindigkeit verfahren. Hier wird keine Meldung wegen Geschwindigkeits-Unterschreitung ausgegeben. Dies geschieht nur, wenn die niedrigste Geschwindigkeit nicht gehalten werden kann.

Programmierung

Es gibt folgende Möglichkeiten, einen Konstantfahrbereich zu programmieren:

- Bei Programmierung mit KRL-Syntax: über die Anweisung CONST_VEL
- Bei Programmierung über Inline-Formulare:
  Start bzw. Ende des Bereichs hinterlegt man am jeweiligen CP-Segment, im Optionsfenster **Logikparameter**.

10.3.6.1 Satzanwahl in den Konstantfahrbereich

Beschreibung

Wenn eine Satzanwahl in einen Konstantfahrbereich durchgeführt wird, ignoriert die Robotersteuerung diesen und gibt diesbezüglich eine Meldung aus. Die Bewegungen werden ausgeführt, wie wenn kein Konstantfahrbereich programmiert wäre.

Als Satzanwahl in den Konstantfahrbereich gilt eine Satzanwahl in den Bahnabschnitt, der durch die Verschiebungs-Werte definiert ist. In welchen Bewegungssätzen Anfang und Ende des Bereichs programmiert sind, spielt dagegen keine Rolle.

Beispiel

```
SPLINE S1 Vel=2 m/s CPDAT1 Tool[1] Base[0]

SLIN P1 ADAT1

   CONST_VEL START = 50

   SLIN XP1

SLIN P2

SLIN P3

SLIN P4 ADAT2

   CONST_VEL END = -50 ONSTART

   SLIN XP4

SLIN P5

ENDSPLINE
```

Abb. 10-31: Beispiel Konstantfahrbereich (Inline-Programmierung)

Die Folds im Programm sind aufgeklappt. Die Einrückungen sind defaultmäßig nicht vorhanden und wurden hier der besseren Übersicht wegen eingefügt.

Der Anfang des Konstantfahrbereichs ist im Programm an P1 hinterlegt. Das Ende ist an P4 hinterlegt. Um zu beurteilen, was als Satzanwahl in den Konstantfahrbereich gilt, ist jedoch ausschlaggebend, wo sich der Bereich auf der Bahn befindet:

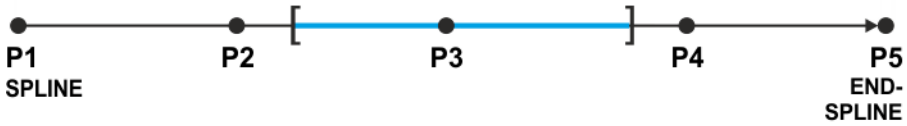


Abb. 10-32: Beispiel Konstantfahrbereich (Bahn)

Was gilt als Satzanwahl in den Konstantfahrbereich?

Satzanwahl auf Punkt …	P1	P2	P3	P4
= Im Konstantfahrbereich?	Nein	Nein	Ja	Nein

10.3.6.2 Maximale Grenzen

Wenn der Start- bzw. Zielpunkt des Spline-Blocks ein Genauhalt ist:

- Der Konstantfahrbereich beginnt frühestens am Startpunkt.
- Der Konstantfahrbereich endet spätestens am Zielpunkt.

Wenn der Verschiebewert so ist, dass diese Grenzen überschritten würden, dann reduziert die Robotersteuerung den Offset automatisch und gibt folgende Meldung aus: *CONST_VEL {Start/End} = {Offset} nicht realisierbar, {Neuer Offset} wird verwendet.*

Die Robotersteuerung reduziert den Offset soweit, dass ein Bereich entsteht, in dem sie die programmierte Geschwindigkeit konstant halten kann. Das heißt: Sie schiebt die Grenze nicht unbedingt genau auf den Start- oder Zielpunkt des Spline-Blocks, sondern eventuell weiter nach innen.

Die gleiche Meldung kommt, wenn der Bereich zwar von vorneherein innerhalb des Spline-Blocks liegt, aber die definierte Geschwindigkeit aufgrund des Offsets nicht gehalten werden kann. Auch dann reduziert die Robotersteuerung den Offset.

Wenn der Start- bzw. Zielpunkt des Spline-Blocks überschliffen ist:

- Der Konstantfahrbereich beginnt frühestens am Anfang des Überschleifbogens des Startpunkts.
- Der Konstantfahrbereich endet spätestens am Anfang des Überschleifbogens des Zielpunkts.

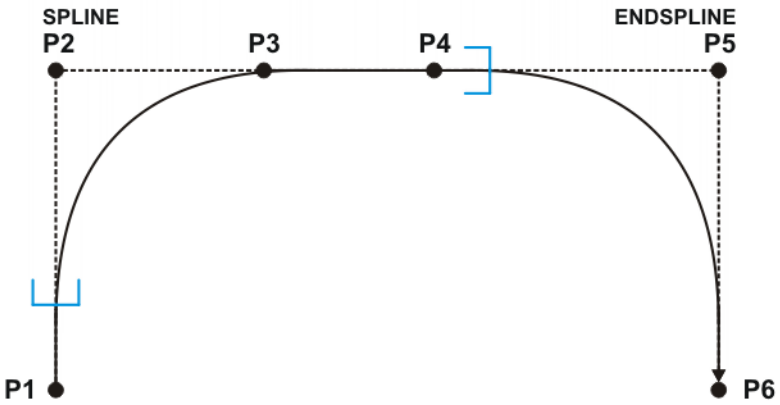


Abb. 10-33: Maximale Grenzen bei überschliffenem SPLINE/ENDSPLINE

Wenn der Offset so ist, dass diese Grenzen überschritten würden, dann setzt die Robotersteuerung die Grenze automatisch auf den Anfang des jeweiligen Überschleifbogens. Sie gibt keine Meldung aus.

10.4 Abstand zwischen Punkten anzeigen

Voraussetzung

- Programm ist angewählt oder geöffnet.
- Betriebsart T1 oder T2
- Benutzergruppe Experte

Vorgehensweise

1. Die Punkte (= die Bewegungssätze), deren Abstand angezeigt werden soll, markieren. Es können auch mehrere, aufeinanderfolgende Sätze markiert werden.
2. Menüfolge **Bearbeiten** > **Markierter Bereich** > **Kart. Abstand** wählen.
   Ein Fenster öffnet sich. Es zeigt folgende Informationen an:
   - Die kartesischen Koordinaten des ersten markierten Punkts
   - Die kartesischen Koordinaten des letzten markierten Punkts
   - Den Abstand zwischen den Koordinaten, in Millimeter und Grad
   - Den Abstand zwischen der TCP-Position am ersten und letzten Punkt, in Millimeter und Grad
3. Bei Bedarf noch andere Punkte markieren.
4. Auf **Aktualisieren** drücken, um die Anzeige zu aktualisieren.

10.5 Programmierte Bewegungen ändern

10.5.1 Bewegungsparameter ändern

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile mit der Anweisung setzen, die geändert werden soll.
2. **Ändern** drücken. Das Inline-Formular zur Anweisung öffnet sich.
3. Parameter ändern.
4. Änderungen mit **Befehl OK** speichern.

10.5.2 Bewegungsparameter blockweise ändern

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt.

- Betriebsart T1

Vorgehensweise

1. Die Bewegungssätze, die geändert werden sollen, markieren. (Nur aufeinanderfolgende Bewegungssätze können blockweise geändert werden.)
2. **Ändern** drücken. Das Inline-Formular des ersten markierten Bewegungssatzes öffnet sich.
3. Parameter ändern.
4. **Befehl OK** drücken. Die Änderungen werden in die markierten Bewegungssätze übernommen, soweit dies möglich ist.

   Manche Änderungen werden nicht in jeden Bewegungssatz übernommen. Z. B. ist es nicht möglich den PTP-Parameter **Geschwindigkeit** in einen LIN-Bewegungssatz zu übernehmen.

10.5.3 Punkt umteachen

Beschreibung

Die Koordinaten eines geteachten Punkts können geändert werden. Dazu fährt man die gewünschte neue Position an und überschreibt den alten Punkt mit der neuen Position.

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Gewünschte Position mit dem TCP anfahren.
2. Cursor in die Zeile mit der Bewegungsanweisung setzen, die geändert werden soll.
3. **Ändern** drücken. Das Inline-Formular zur Anweisung öffnet sich.
4. Für PTP- und LIN-Bewegungen: **Touch Up** drücken, um die aktuelle Position des TCP als neuen Zielpunkt zu übernehmen.

   Für CIRC-Bewegungen:

   - **Touchup HP** drücken, um die aktuelle Position des TCP als neuen Hilfspunkt zu übernehmen.
   - Oder **Touchup ZP** drücken, um die aktuelle Position des TCP als neuen Zielpunkt zu übernehmen.
5. Sicherheitsabfrage mit **Ja** bestätigen.
6. Änderung mit **Befehl OK** speichern.

10.5.4 Koordinaten blockweise verschieben

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Die Bewegungssätze, die geändert werden sollen, markieren. (Nur aufeinanderfolgende Bewegungssätze können blockweise geändert werden.)
2. Menüfolge **Bearbeiten** > **Markierter Bereich** wählen. Verschiebungstyp wählen.

   Das zugehörige Fenster öffnet sich.

   (>>> 10.5.4.1 "Fenster Achsspiegeln" Seite 355)

   (>>> 10.5.4.2 "Fenster Verschieben - achsspezifisch" Seite 356)

   (>>> 10.5.4.3 "Fenster Verschieben - kartesisch" Seite 357)
3. Werte für die Verschiebung eingeben und **Berechnen** drücken.

Übersicht

Folgende Verschiebungstypen stehen zur Auswahl:

- **Verschieben - kartesisch Base**

- **Verschieben - kartesisch Tool**
- **Verschieben - kartesisch World**
- **Verschieben - achsspezifisch**
- **Achsspiegeln**

Verschieben Base

Verschieben - kartesisch Base:

Die Verschiebung bezieht sich auf das aktuelle BASE-Koordinatensystem.

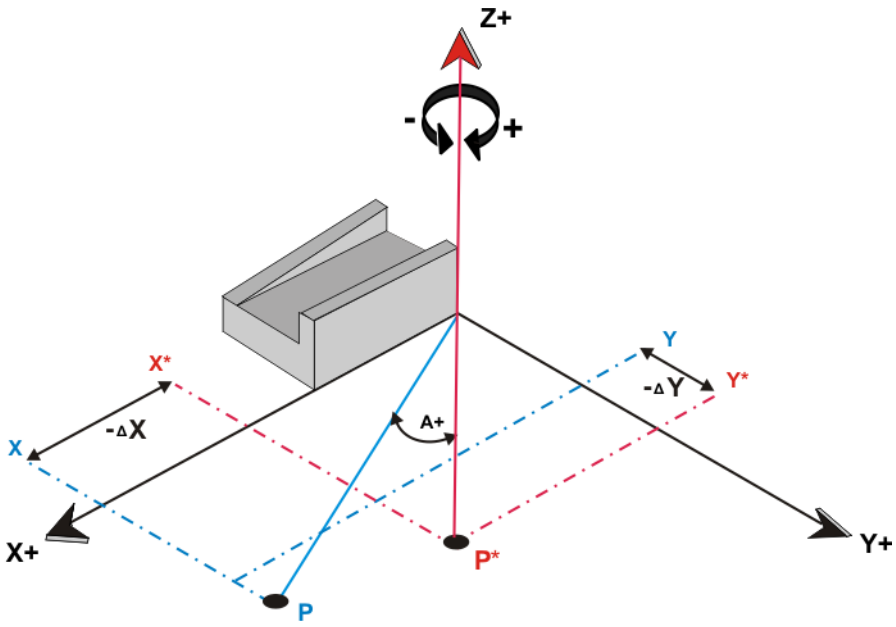


Abb. 10-34: Verschieben - kartesisch Base

Der Punkt P wird um Δ X und Δ Y negativ verschoben. Die neue Position des Punktes ist P*.

Verschieben Tool

Verschieben - kartesisch Tool:

Die Verschiebung bezieht sich auf das aktuelle TOOL-Koordinatensystem.

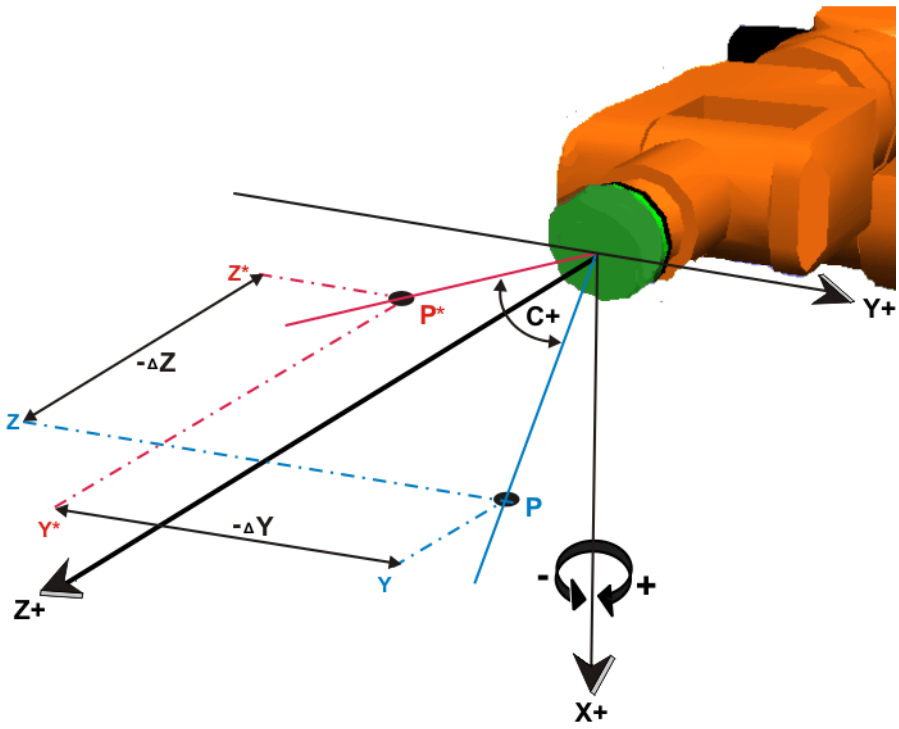


Abb. 10-35: Verschieben - kartesisch Tool

Der Punkt P wird um Δ Z und Δ Y negativ verschoben. Die neue Position des Punktes ist P*.

Verschieben World

Verschieben - kartesisch World:

Die Verschiebung bezieht sich auf das WORLD-Koordinatensystem.

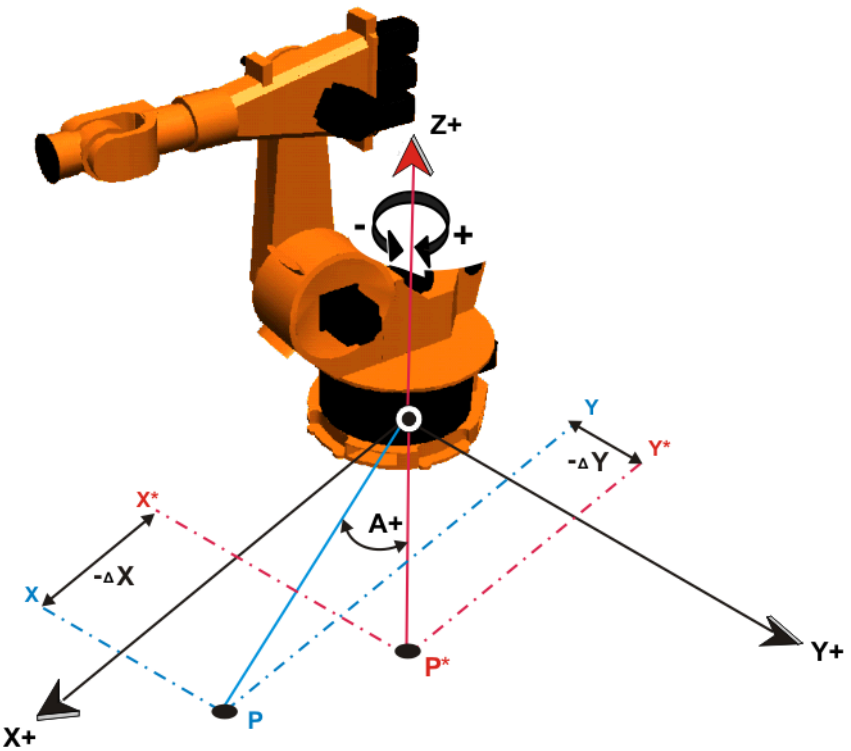


Abb. 10-36: Verschieben - kartesisch World

Der Punkt P wird um Δ X und Δ Y negativ verschoben. Die neue Position des Punktes ist P*.

Verschieben achsspezifisch

Verschieben - achspezifisch:

Die Verschiebung ist achsspezifisch.

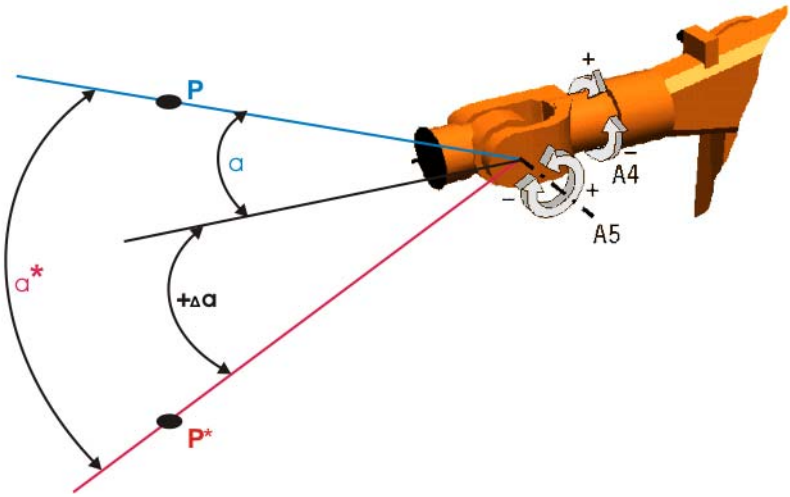


Abb. 10-37: Verschieben - achsspezifisch

Die Achse A5 wird um den Winkel Δα verdreht. Die neue Position des Punktes P ist P*.

Achsspiegeln

Achsspiegeln:

Spiegeln in der X-Y-Ebene des ROBROOT-Koordinatensystems.

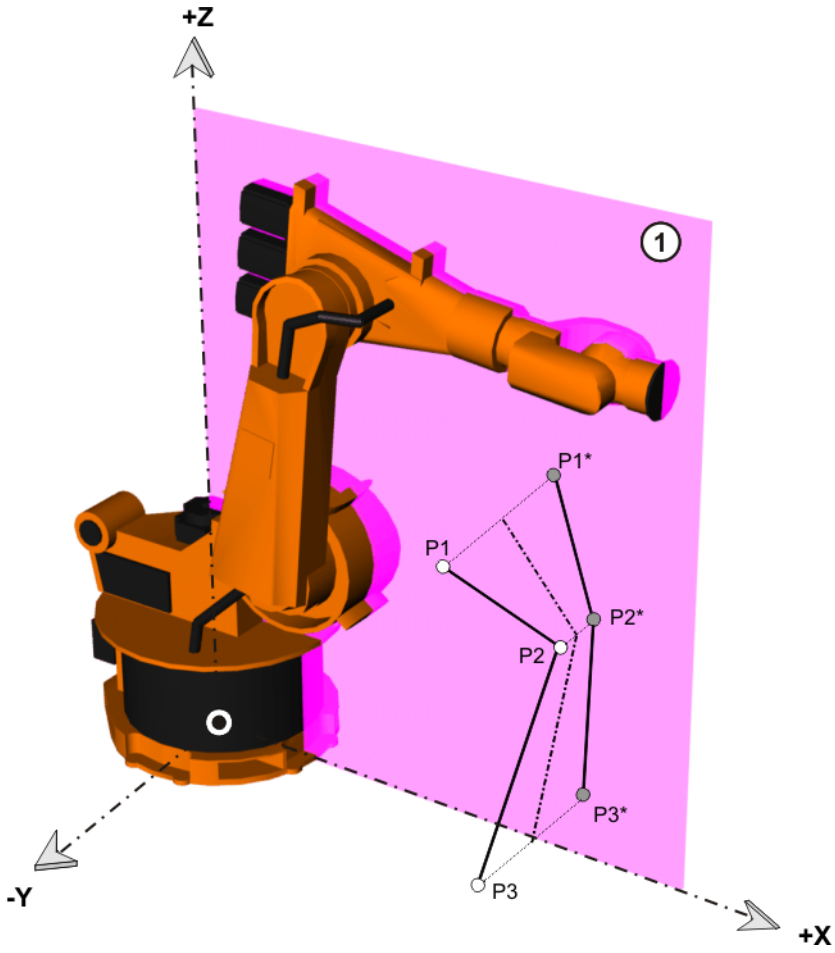


Abb. 10-38: Achsspiegeln

Die Punkte P1, P2, und P3 werden in der X-Y-Ebene (1) gespiegelt. Die neuen Positionen der Punkte sind P1*, P2* und P3*.

10.5.4.1 Fenster Achsspiegeln

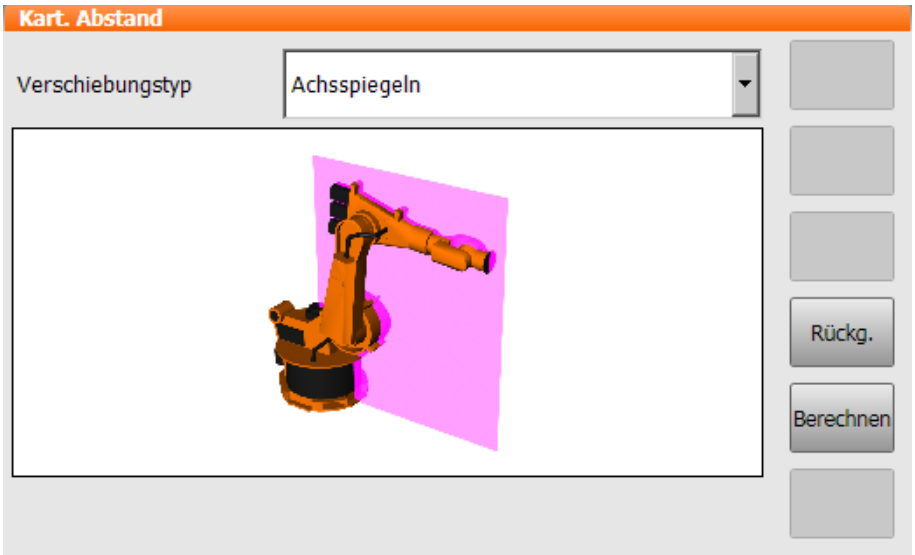


Abb. 10-39: Achsspiegeln

In dieses Fenster müssen keine Werte eingegeben werden. Mit der Schaltfläche **Berechnen** werden die Punktkoordinaten in der X-Z-Ebene des ROBROOT-Koordinatensystems gespiegelt.

Nach dem Achsspiegeln muss das verwendete Werkzeug ebenfalls in der X-Z-Ebene gespiegelt werden.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Berechnen	Spiegelt die Koordinaten der markierten Bahnpunkte in der X-Z-Ebene, rechnet die Koordinaten in Achswinkel um und übernimmt die neuen Werte.
Rückg.	Macht die Achsspiegelung rückgängig und stellt die alten Punktdaten wieder her.

Es werden nur markierte Punkte mit vollständiger E6POS-Beschreibung kopiert. Das sind z. B. alle, die bei der Programmierung über Inline-Formulare generiert werden. Punkte ohne vollständige E6POS-Beschreibung werden bei der Punktverschiebung ignoriert.

10.5.4.2 Fenster Verschieben - achsspezifisch

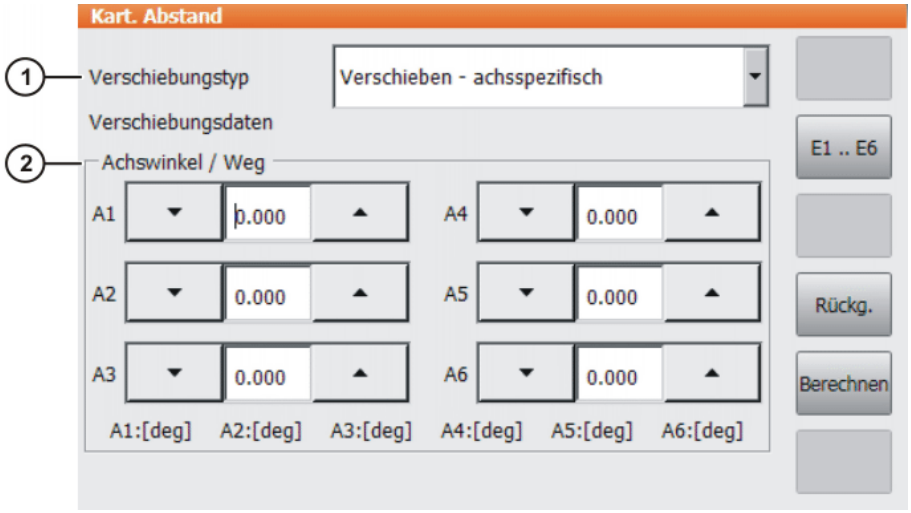


Abb. 10-40: Punktverschiebung - achsspezifisch

Pos.	Beschreibung
1	Auswahl des Verschiebungstyps
2	Gruppe **Achswinkel / Weg**: Eingabefelder für die Verschiebung der Position der Achsen A1 ... A6 ■ Wertebereich: Abhängig von der Konfiguration der achsspezifischen Arbeitsräume Mit **E1 .. E6** kann zur Gruppe **Zusatzachsen** umgeschaltet werden: Eingabefelder für die Verschiebung der Position der Achsen E1 ... E6 **Hinweis:** Nur bei konfigurierten Achsen können Werte eingegeben werden.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
E1 .. E6/A1 .. A6	Schaltet zwischen den Gruppen **Achswinkel / Weg** und **Zusatzachsen** um.
Rückg.	Macht die Punktverschiebung rückgängig und stellt die alten Punktdaten wieder her.
Berechnen	Berechnet die Punktverschiebung und übernimmt sie für alle markierten Bahnpunkte. Wenn ein Punkt durch die Verschiebung außerhalb des konfigurierten Arbeitsraumes liegen würde, wird der Punkt nicht verschoben.

Es werden nur markierte Punkte mit vollständiger E6POS-Beschreibung kopiert. Das sind z. B. alle, die bei der Programmierung über Inline-Formulare generiert werden. Punkte ohne vollständige E6POS-Beschreibung werden bei der Punktverschiebung ignoriert.

10.5.4.3 Fenster Verschieben - kartesisch

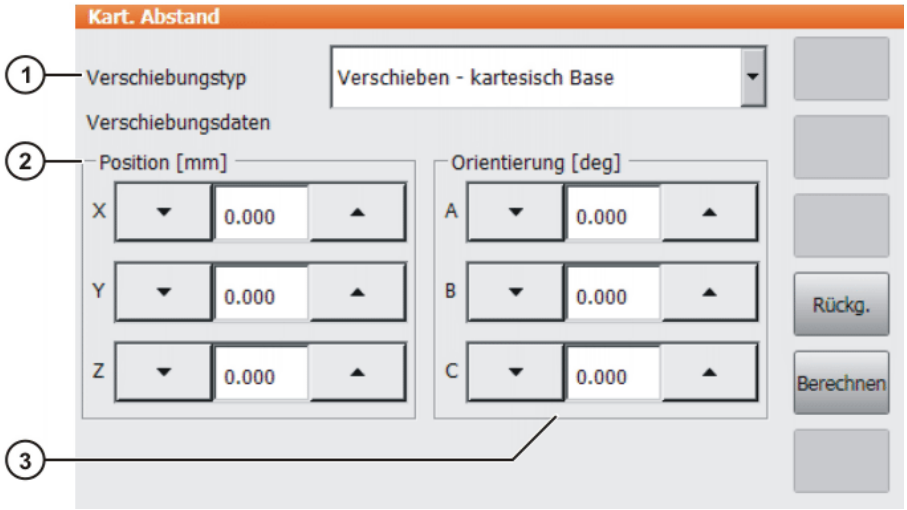


Abb. 10-41: Punktverschiebung - kartesisch

Pos.	Beschreibung
1	Auswahl des Verschiebungstyps
2	Gruppe **Position**: Eingabefelder für die Punktverschiebung in X-, Y-, Z-Richtung ■ Wertebereich: Abhängig von der Konfiguration der kartesischen Arbeitsräume
3	Gruppe **Orientierung**: Eingabefelder für die Verschiebung der A-, B-, C-Orientierung ■ Wertebereich: Abhängig von der Konfiguration der kartesischen Arbeitsräume

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Rückg.	Macht die Punktverschiebung rückgängig und stellt die alten Punktdaten wieder her.
Berechnen	Berechnet die Punktverschiebung und übernimmt sie für alle markierten Bahnpunkte. Wenn ein Punkt durch die Verschiebung außerhalb des konfigurierten Arbeitsraumes liegen würde, wird der Punkt nicht verschoben.

Es werden nur markierte Punkte mit vollständiger E6POS-Beschreibung kopiert. Das sind z. B. alle, die bei der Programmierung über Inline-Formulare generiert werden. Punkte ohne vollständige E6POS-Beschreibung werden bei der Punktverschiebung ignoriert.

10.6 Logikanweisungen programmieren

10.6.1 Ein-/Ausgänge

Digitale Ein-/Ausgänge

Die Robotersteuerung kann maximal 8192 digitale Eingänge und 8192 digitale Ausgänge verwalten. Defaultmäßig stehen 4096 Ein-/Ausgänge zur Verfügung.

Analoge Ein-/Ausgänge

Die Robotersteuerung kann 32 analoge Eingänge und 32 analoge Ausgänge verwalten.

Die Ein-/Ausgänge werden über folgende Systemvariablen verwaltet:

	Eingänge	Ausgänge
Digital	$IN[1] … $IN[8192]	$OUT[1] … $OUT[8192]
Analog	$ANIN[1] … $ANIN[32]	$ANOUT[1] … $ANOUT[32]

$ANIN[…] zeigt die Eingangsspannung an, angepasst auf den Bereich zwischen -1.0 und +1.0. Die tatsächliche Spannung hängt von den Einstellungen des Analogmoduls ab.

Über $ANOUT[…] kann eine Analogspannung gesetzt werden. $ANOUT[…] kann mit Werten von -1.0 bis +1.0 beschrieben werden. Die tatsächlich erzeugte Spannung hängt von den Einstellungen des Analogmoduls ab. Wenn man versucht, Spannungen außerhalb des Wertebereichs zu setzen, zeigt die Robotersteuerung folgende Meldung an: *Begrenzung {Signalname}*

10.6.2 Digitalen Ausgang setzen - OUT

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **OUT** > **OUT** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.3 "Inline-Formular OUT" Seite 359)
4. Anweisung mit **Befehl OK** speichern.

10.6.3 Inline-Formular OUT

Die Anweisung setzt einen digitalen Ausgang.

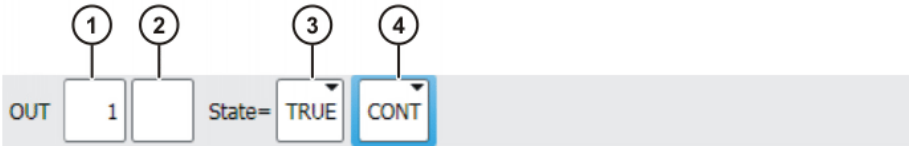


Abb. 10-42: Inline-Formular OUT

Pos.	Beschreibung
1	Nummer des Ausgangs
2	Wenn für den Ausgang ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.
3	Status, auf den der Ausgang geschaltet wird ■ **TRUE** ■ **FALSE**
4	■ **CONT**: Bearbeitung im Vorlauf ■ **[leer]**: Bearbeitung mit Vorlaufstopp

10.6.4 Impulsausgang setzen - PULSE

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **OUT** > **PULSE** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.5 "Inline-Formular PULSE" Seite 359)
4. Anweisung mit **Befehl OK** speichern.

10.6.5 Inline-Formular PULSE

Die Anweisung setzt einen Impuls mit der definierten Länge.

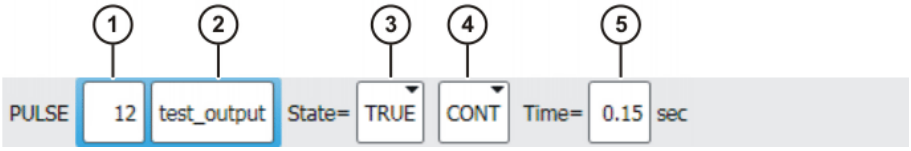


Abb. 10-43: Inline-Formular PULSE

Pos.	Beschreibung
1	Nummer des Ausgangs
2	Wenn für den Ausgang ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.

Pos.	Beschreibung
3	Status, auf den der Ausgang geschaltet wird ■ **TRUE**: Pegel "High" ■ **FALSE**: Pegel "Low"
4	■ **CONT**: Bearbeitung im Vorlauf ■ **[leer]**: Bearbeitung mit Vorlaufstopp
5	Länge des Impulses ■ **0.10 … 3.00 s**

10.6.6 Analogen Ausgang setzen - ANOUT

Voraussetzung
- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise
1. Cursor in die Zeile setzen, nach der die Anweisung eingefügt werden soll.
2. **Befehle** > **Analogausgabe** > **Statisch** oder **Dynamisch** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.7 "Inline-Formular ANOUT statisch" Seite 360)
   (>>> 10.6.8 "Inline-Formular ANOUT dynamisch" Seite 360)
4. Anweisung mit **Befehl OK** speichern.

10.6.7 Inline-Formular ANOUT statisch

Diese Anweisung setzt einen statischen analogen Ausgang. Die Spannung wird durch einen Faktor auf eine feste Höhe gesetzt. Wie hoch die Spannung tatsächlich ist, ist abhängig vom verwendeten Analogmodul. Zum Beispiel liefert ein 10 V-Modul bei einem Faktor von 0.5 eine Spannung von 5 V.

ANOUT löst einen Vorlaufstopp aus.

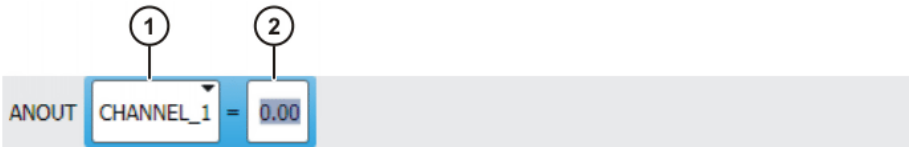


Abb. 10-44: Inline-Formular ANOUT statisch

Pos.	Beschreibung
1	Nummer des analogen Ausgangs ■ **CHANNEL_1 … CHANNEL_32**
2	Faktor für die Spannung ■ **0 … 1** (Abstufung: 0.01)

10.6.8 Inline-Formular ANOUT dynamisch

Diese Anweisung schaltet einen dynamischen analogen Ausgang ein oder aus.

Maximal 4 dynamische analoge Ausgänge können gleichzeitig eingeschaltet sein. ANOUT löst einen Vorlaufstopp aus.

Die Spannung wird durch einen Faktor festgelegt. Wie hoch die Spannung tatsächlich ist, ist abhängig von folgenden Werten:

- Geschwindigkeit oder Funktionsgenerator

  Zum Beispiel ergibt eine Geschwindigkeit von 1m/s bei einem Faktor von 0.5 eine Spannung von 5 V.
- Offset

  Zum Beispiel ergibt ein Offset von +0.15 auf eine Spannung von 0.5 V eine Spannung von 6.5 V.

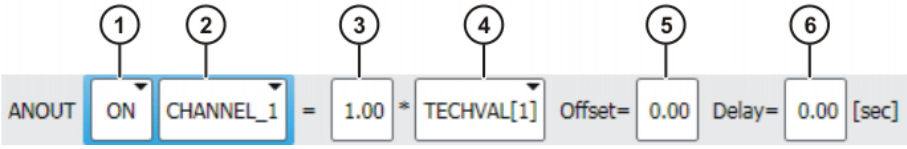


Abb. 10-45: Inline-Formular ANOUT dynamisch

Pos.	Beschreibung
1	Ein- oder Ausschalten des analogen Ausgangs ■ **ON** ■ **OFF**
2	Nummer des analogen Ausgangs ■ **CHANNEL_1 … CHANNEL_32**
3	Faktor für die Spannung ■ **0 … 10** (Abstufung: 0.01)
4	■ **VEL_ACT**: Die Spannung ist abhängig von der Geschwindigkeit. ■ **TECHVAL[1] … TECHVAL[6]**: Die Spannung wird über einen Funktionsgenerator gesteuert.
5	Wert, um den die Spannung erhöht oder verringert wird ■ **-1 … +1** (Abstufung: 0.01)
6	Zeit, um die das Ausgabesignal verzögert (+) oder vorzeitig (-) ausgegeben wird ■ **-0.2 … +0.5 s**

10.6.9 Wartezeit programmieren - WAIT

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **WAIT** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.10 "Inline-Formular WAIT" Seite 361)
4. Anweisung mit **Befehl OK** speichern.

10.6.10 Inline-Formular WAIT

Mit WAIT kann eine Wartezeit programmiert werden. Die Roboterbewegung wird für die programmierte Zeit angehalten. WAIT löst immer einen Vorlaufstopp aus.

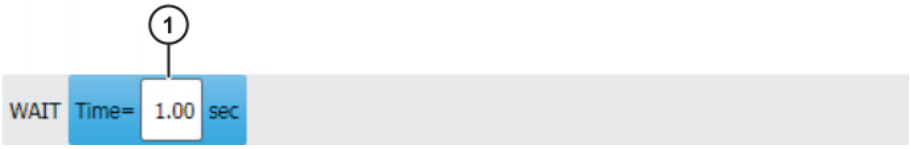


Abb. 10-46: Inline-Formular WAIT

Pos.	Beschreibung
1	Wartezeit ■ **≥ 0** s

10.6.11 Signalabhängige Wartefunktion programmieren - WAITFOR

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **WAITFOR** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.12 "Inline-Formular WAITFOR" Seite 362)
4. Anweisung mit **Befehl OK** speichern.

10.6.12 Inline-Formular WAITFOR

Die Anweisung setzt eine signalabhängige Wartefunktion.

Bei Bedarf können mehrere Signale (maximal 12) logisch verknüpft werden. Wenn eine Verknüpfung hinzugefügt wird, werden im Inline-Formular Felder für die zusätzlichen Signale und für weitere Verknüpfungen eingeblendet.

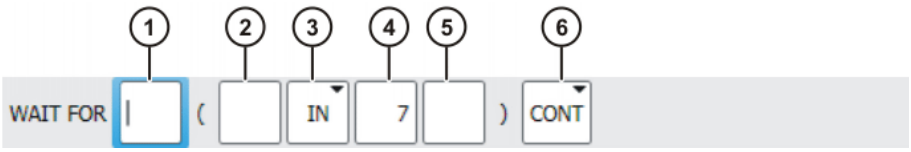


Abb. 10-47: Inline-Formular WAITFOR

Pos.	Beschreibung
1	Äußere Verknüpfung hinzufügen. Der Operator steht zwischen den geklammerten Ausdrücken. ■ **AND** ■ **OR** ■ **EXOR** NOT hinzufügen. ■ **NOT** ■ **[leer]** Den gewünschten Operator über die entsprechende Schaltfläche einfügen.
2	Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks. ■ **AND** ■ **OR** ■ **EXOR** NOT hinzufügen. ■ **NOT** ■ **[leer]** Den gewünschten Operator über die entsprechende Schaltfläche einfügen.
3	Signal, auf das gewartet wird ■ **IN** ■ **OUT** ■ **CYCFLAG** ■ **TIMER** ■ **FLAG**
4	Nummer des Signals
5	Wenn für das Signal ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.
6	■ **CONT**: Bearbeitung im Vorlauf ■ **[leer]**: Bearbeitung mit Vorlaufstopp

10.6.13 Schalten auf der Bahn - SYN OUT

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **OUT** > **SYN OUT** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.14 "Inline-Formular SYN OUT, Option START/END" Seite 364)
   (>>> 10.6.15 "Inline-Formular SYN OUT, Option PATH" Seite 366)
4. Anweisung mit **Befehl OK** speichern.

10.6.14 Inline-Formular SYN OUT, Option START/END

Die Schaltaktion kann bezogen auf den Start- oder den Zielpunkt der Bewegung ausgelöst werden. Die Schaltaktion kann zeitlich verschoben werden. Die Bewegung kann LIN, CIRC oder PTP sein.

Anwendungsfälle sind z. B.:

- Schließen oder Öffnen der Schweißzange beim Punktschweißen
- Ein- oder Ausschalten des Schweißstroms beim Bahnschweißen
- Zu- oder Abschalten des Volumenstroms beim Kleben oder Abdichten

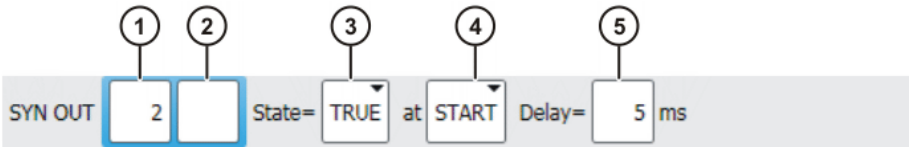


Abb. 10-48: Inline-Formular SYN OUT, Option START/END

Pos.	Beschreibung
1	Nummer des Ausgangs
2	Wenn für den Ausgang ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.
3	Status, auf den der Ausgang geschaltet wird ■ **TRUE** ■ **FALSE**
4	Punkt, auf den sich SYN OUT bezieht: ■ **START**: Startpunkt der Bewegung ■ **END**: Zielpunkt der Bewegung
5	Zeitliche Verschiebung der Schaltaktion ■ **-1 000 … +1 000 ms** **Hinweis:** Die Zeitangabe ist absolut. D. h., der Schaltpunkt ändert sich je nach Geschwindigkeit des Roboters.

Beispiel 1

Start- und Zielpunkt sind Genauhaltepunkte.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

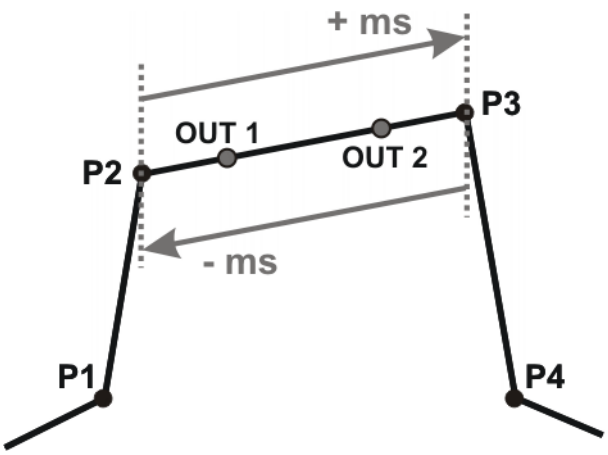


Abb. 10-49

OUT 1 und OUT 2 geben die ungefähren Positionen an, an denen geschaltet wird. Die gepunkteten Linien geben die Schaltgrenzen an.

Schaltgrenzen:

- START: Der Schaltpunkt kann maximal bis zum Genauhaltepunkt P3 verzögert werden (+ ms).
- END: Der Schaltpunkt kann maximal bis zum Genauhaltepunkt P2 vorverlegt werden (- ms).

Wenn für die zeitliche Verschiebung größere Werte angegeben werden, schaltet die Steuerung automatisch an der Schaltgrenze.

Beispiel 2

Startpunkt ist Genauhaltepunkt und Zielpunkt ist überschliffen.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

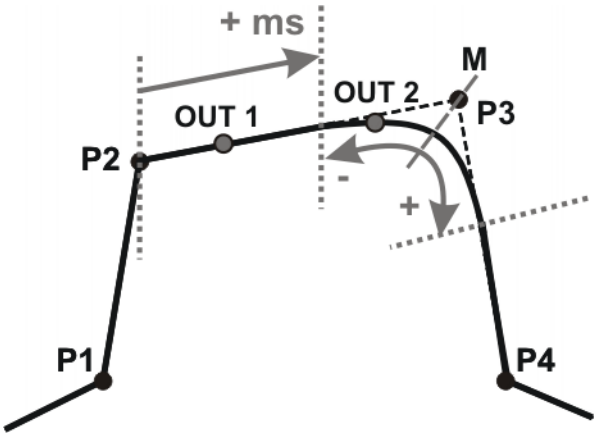


Abb. 10-50

OUT 1 und OUT 2 geben die ungefähren Positionen an, an denen geschaltet wird. Die gepunkteten Linien geben die Schaltgrenzen an. M = Mitte des Überschleifbereichs.

Schaltgrenzen:

- START: Der Schaltpunkt kann maximal bis zum Beginn des Überschleifbereichs von P3 verzögert werden (+ ms).

- END: Der Schaltpunkt kann maximal bis zum Beginn des Überschleifbereichs von P3 vorverlegt werden (-).

  Der Schaltpunkt kann maximal bis zum Ende des Überschleifbereichs von P3 verzögert werden (+).

Wenn für die zeitliche Verschiebung größere Werte angegeben werden, schaltet die Steuerung automatisch an der Schaltgrenze.

Beispiel 3

Start- und Zielpunkt sind überschliffen.

```
LIN P1 VEL=0.3m/s CPDAT1
LIN P2 CONT VEL=0.3m/s CPDAT2
SYN OUT 1 '' State= TRUE at START Delay=20ms
SYN OUT 2 '' State= TRUE at END Delay=-20ms
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

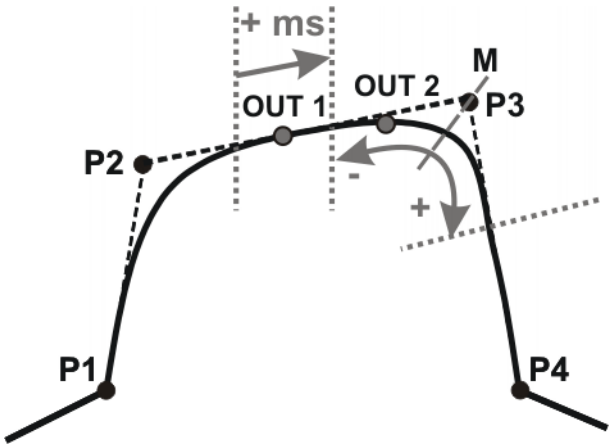


Abb. 10-51

OUT 1 und OUT 2 geben die ungefähren Positionen an, an denen geschaltet wird. Die gepunkteten Linien geben die Schaltgrenzen an. M = Mitte des Überschleifbereichs.

Schaltgrenzen:

- START: Der Schaltpunkt kann frühestens am Ende des Überschleifbereichs von P2 liegen.

  Der Schaltpunkt kann maximal bis zum Beginn des Überschleifbereichs von P3 verzögert werden (+ ms).
- END: Der Schaltpunkt kann maximal bis zum Beginn des Überschleifbereichs von P3 vorverlegt werden (-).

  Der Schaltpunkt kann maximal bis zum Ende des Überschleifbereichs von P3 verzögert werden (+).

Wenn für die zeitliche Verschiebung größere Werte angegeben werden, schaltet die Steuerung automatisch an der Schaltgrenze.

10.6.15 Inline-Formular SYN OUT, Option PATH

Die Schaltaktion bezieht sich auf den Zielpunkt der Bewegung. Die Schaltaktion kann örtlich und zeitlich verschoben werden. Die Bewegung kann LIN oder CIRC sein. Sie darf keine PTP-Bewegung sein.

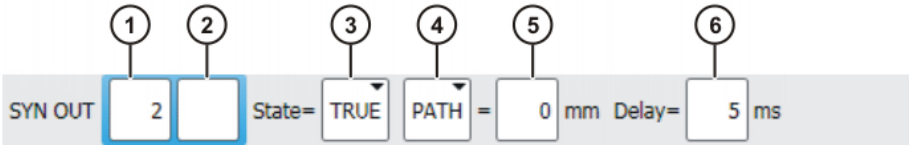


Abb. 10-52: Inline-Formular SYN OUT, Option PATH

Pos.	Beschreibung
1	Nummer des Ausgangs
2	Wenn für den Ausgang ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.
3	Status, auf den der Ausgang geschaltet wird ■ **TRUE** ■ **FALSE**
4	■ **PATH**: SYN OUT bezieht sich auf den Zielpunkt der Bewegung.
5	Dieses Feld wird nur angezeigt, wenn **PATH** ausgewählt wurde. Entfernung des Schaltpunkts vom Zielpunkt ■ **-2 000 … +2 000 mm**
6	Zeitliche Verschiebung der Schaltaktion ■ **-1 000 … +1 000 ms** **Hinweis:** Die Zeitangabe ist absolut. D. h., der Schaltpunkt ändert sich je nach Geschwindigkeit des Roboters.

Beispiel 1

Startpunkt ist Genauhaltepunkt und Zielpunkt ist überschliffen.

```
LIN P1 VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

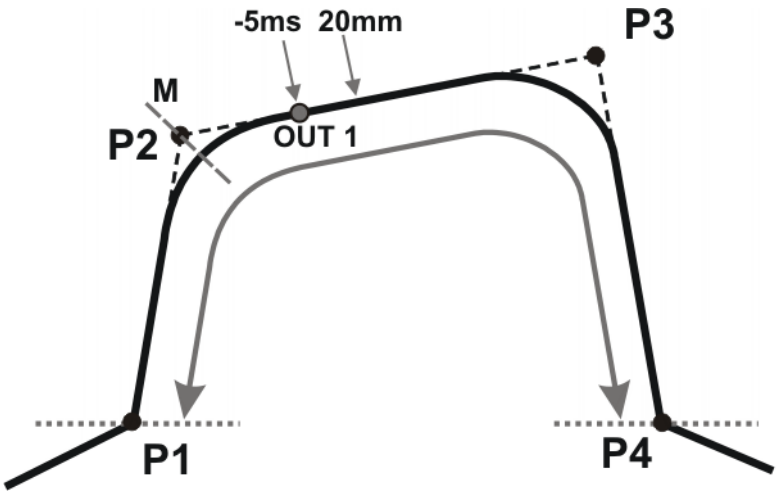


Abb. 10-53

OUT 1 gibt die ungefähre Position an, an der geschaltet wird. Die gepunkteten Linien geben die Schaltgrenzen an. M = Mitte des Überschleifbereichs.

Schaltgrenzen:

- Der Schaltpunkt kann frühestens bis zum Genauhaltepunkt P1 vorverlegt werden.
- Der Schaltpunkt kann maximal bis zum nächsten Genauhaltepunkt P4 verzögert werden. Wenn P3 ein Genauhaltepunkt wäre, könnte der Schaltpunkt maximal bis P3 verzögert werden.

Wenn für die örltiche oder zeitliche Verschiebung größere Werte angegeben werden, schaltet die Steuerung automatisch an der Schaltgrenze.

Beispiel 2

Start- und Zielpunkt sind überschliffen.

```
LIN P1 CONT VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

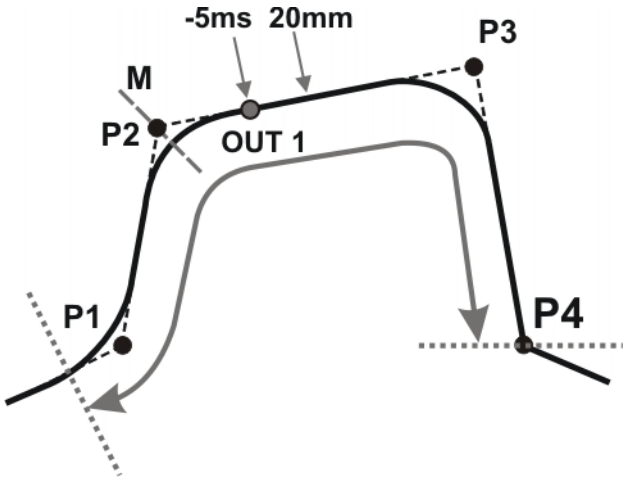


Abb. 10-54

OUT 1 gibt die ungefähre Position an, an der geschaltet wird. Die gepunkteten Linien geben die Schaltgrenzen an. M = Mitte des Überschleifbereichs.

Schaltgrenzen:

- Der Schaltpunkt kann frühestens bis zum Beginn des Überschleifbereichs von P1 vorverlegt werden.
- Der Schaltpunkt kann maximal bis zum nächsten Genauhaltepunkt P4 verzögert werden. Wenn P3 ein Genauhaltepunkt wäre, könnte der Schaltpunkt maximal bis P3 verzögert werden.

Wenn für die örltiche oder zeitliche Verschiebung größere Werte angegeben werden, schaltet die Steuerung automatisch an der Schaltgrenze.

10.6.16 Puls setzen auf der Bahn - SYN PULSE

Voraussetzung

- Programm ist angewählt.
- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll.
2. Menüfolge **Befehle** > **Logik** > **OUT** > **SYN PULSE** wählen.
3. Im Inline-Formular die Parameter einstellen.
   (>>> 10.6.17 "Inline-Formular SYN PULSE" Seite 369)
4. Anweisung mit **Befehl OK** speichern.

10.6.17 Inline-Formular SYN PULSE

Mit SYN PULSE kann am Start- oder am Zielpunkt der Bewegung ein Impuls ausgelöst werden. Der Impuls kann zeitlich und/oder örtlich verschoben werden: D. h., er muss nicht genau am Punkt ausgelöst werden, sondern er kann auch vorher oder nachher ausgelöst werden.

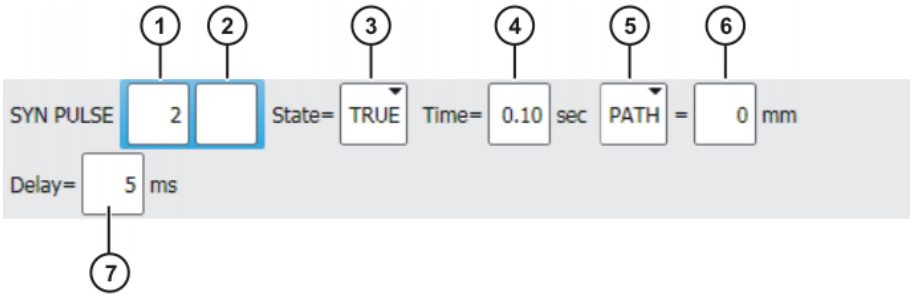


Abb. 10-55: Inline-Formular SYN PULSE

Pos.	Beschreibung
1	Nummer des Ausgangs
2	Wenn für den Ausgang ein Name existiert, wird er angezeigt. Nur für Benutzergruppe Experte: Durch Drücken auf **Langtext** kann ein Name eingegeben werden. Der Name ist frei wählbar.
3	Status, auf den der Ausgang geschaltet wird ■ **TRUE** ■ **FALSE**
4	Dauer des Impulses ■ **0,1 … 3 s**
5	Punkt, auf den sich SYN PULSE bezieht: ■ **START**: Startpunkt der Bewegung ■ **END**: Zielpunkt der Bewegung Beispiele und Schaltgrenzen siehe SYN OUT. (>>> 10.6.14 "Inline-Formular SYN OUT, Option START/END" Seite 364) ■ **PATH**: SYN PULSE bezieht sich auf den Zielpunkt. Zusätzlich ist eine örtliche Verschiebung möglich. Beispiele und Schaltgrenzen siehe SYN OUT. (>>> 10.6.15 "Inline-Formular SYN OUT, Option PATH" Seite 366)
6	Entfernung des Schaltpunkts vom Zielpunkt ■ **-2 000 … +2 000 mm** Diese Feld wird nur angezeigt, wenn **PATH** ausgewählt wurde.
7	Zeitliche Verschiebung der Schaltaktion ■ **-1 000 … +1 000 ms** **Hinweis:** Die Zeitangabe ist absolut. Der Schaltpunkt ändert sich je nach Geschwindigkeit des Roboters.

10.6.18 Logikanweisung ändern

Voraussetzung
- Programm ist angewählt.

- Betriebsart T1

Vorgehensweise

1. Cursor in die Zeile mit der Anweisung setzen, die geändert werden soll.
2. **Ändern** drücken. Das Inline-Formular zur Anweisung öffnet sich.
3. Die Parameter ändern.
4. Änderungen mit **Befehl OK** speichern.

11 Programmierung für Benutzergruppe Experte (KRL-Syntax)

HINWEIS Bei Programmen mit folgenden Achsbewegungen oder -positionen kann es an den Getrieben der Achsen zu einer Unterbrechung des Schmierfilms kommen:

- Bewegungen < 3°
- Oszillierende Bewegungen
- Dauerhaft oben liegende Getriebebereiche

Es muss sichergestellt werden, dass die Getriebe ausreichend mit Öl versorgt werden. Hierfür muss bei oszillierenden oder kurzen Bewegungen (< 3°) so programmiert werden, dass sich die betroffenen Achsen regelmäßig (z. Bsp. je Zyklus) um mehr als 40° bewegen.
Im Falle dauerhaft oben liegender Getriebebereiche muss eine ausreichende Ölversorgung erreicht werden, indem man Umorientierungen der Zentralhand programmiert. Auf diese Weise kann das Öl durch die Schwerkraft in alle Getriebebereiche gelangen. Erforderliche Häufigkeit der Umorientierungen:

- Bei geringer Belastung (Getriebetemperatur < +35 °C): 1-mal täglich
- Bei mittlerer Belastung (Getriebetemperatur +35 bis 55 °C): stündlich
- Bei starker Belastung (Getriebetemperatur > +55 °C): alle 10 min

Wenn dies nicht beachtet wird, können Schäden an den Getrieben entstehen.

Wenn ein angewähltes Programm in der Benutzergruppe Experte bearbeitet wird, muss danach der Cursor aus der bearbeiteten Zeile genommen werden und in eine beliebige andere Zeile gesetzt werden! Nur so ist gewährleistet, dass die Bearbeitung übernommen wird, wenn das Programm wieder abgewählt wird.

11.1 Übersicht KRL-Syntax

Variablen und Vereinbarungen	
DECL	(>>> 11.4.1 "DECL" Seite 378)
ENUM	(>>> 11.4.2 "ENUM" Seite 380)
STRUC	(>>> 11.4.3 "STRUC" Seite 381)

Bewegungsprogrammierung	
PTP	(>>> 11.5.1 "PTP" Seite 382)
LIN, CIRC	(>>> 11.5.3 "LIN, CIRC" Seite 384)
PTP_REL	(>>> 11.5.2 "PTP_REL" Seite 383)
LIN_REL, CIRC_REL	(>>> 11.5.4 "LIN_REL, CIRC_REL" Seite 385)
SPLINE … ENDSPLINE	(>>> 11.6.1 "SPLINE ... ENDSPLINE" Seite 390)
PTP_SPLINE … ENDSPLINE	(>>> 11.6.2 "PTP_SPLINE ... ENDSPLINE" Seite 391)
SLIN, SCIRC, SPL	(>>> 11.6.3 "SLIN, SCIRC, SPL" Seite 392)
SLIN_REL, SCIRC_REL, SPL_REL	(>>> 11.6.4 "SLIN_REL, SCIRC_REL, SPL_REL" Seite 393)
SPTP	(>>> 11.6.5 "SPTP" Seite 395)
SPTP_REL	(>>> 11.6.6 "SPTP_REL" Seite 396)
TIME_BLOCK	(>>> 11.6.8 "TIME_BLOCK" Seite 398)
CONST_VEL	(>>> 11.6.9 "CONST_VEL" Seite 400)
STOP WHEN PATH	(>>> 11.6.10 "STOP WHEN PATH" Seite 403)

Programmablaufkontrolle	
CONTINUE	(>>> 11.7.1 "CONTINUE" Seite 405)
EXIT	(>>> 11.7.2 "EXIT" Seite 406)
FOR … TO … ENDFOR	(>>> 11.7.3 "FOR ... TO ... ENDFOR" Seite 406)
GOTO	(>>> 11.7.4 "GOTO" Seite 407)
HALT	(>>> 11.7.5 "HALT" Seite 408)
IF … THEN … ENDIF	(>>> 11.7.6 "IF ... THEN ... ENDIF" Seite 408)
LOOP … ENDLOOP	(>>> 11.7.7 "LOOP ... ENDLOOP" Seite 409)
ON_ERROR_PROCEED	(>>> 11.7.8 "ON_ERROR_PROCEED" Seite 409)
REPEAT … UNTIL	(>>> 11.7.9 "REPEAT ... UNTIL" Seite 414)
SWITCH … CASE … ENDSWITCH	(>>> 11.7.10 "SWITCH ... CASE ... ENDSWITCH" Seite 415)
WAIT FOR …	(>>> 11.7.11 "WAIT FOR …" Seite 416)
WAIT SEC …	(>>> 11.7.12 "WAIT SEC …" Seite 417)
WHILE … ENDWHILE	(>>> 11.7.13 "WHILE ... ENDWHILE" Seite 417)

Ein-/Ausgänge	
ANIN	(>>> 11.8.1 "ANIN" Seite 418)
ANOUT	(>>> 11.8.2 "ANOUT" Seite 419)
PULSE	(>>> 11.8.3 "PULSE" Seite 420)
SIGNAL	(>>> 11.8.4 "SIGNAL" Seite 424)

Unterprogramme und Funktionen	
DEFFCT … ENDFCT	(>>> 11.9.3 "DEFFCT ... ENDFCT" Seite 426)
RETURN	(>>> 11.9.4 "RETURN" Seite 426)

Interrupt-Programmierung	
BRAKE	(>>> 11.10.1 "BRAKE" Seite 432)
INTERRUPT	(>>> 11.10.3 "INTERRUPT" Seite 434)
INTERRUPT … DECL … WHEN … DO	(>>> 11.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Seite 432)
RESUME	(>>> 11.10.4 "RESUME" Seite 435)

Bahnbezogene Schaltaktionen (=Trigger)	
TRIGGER WHEN DISTANCE	(>>> 11.11.1 "TRIGGER WHEN DISTANCE" Seite 437)
TRIGGER WHEN PATH	(>>> 11.11.2 "TRIGGER WHEN PATH" Seite 440)

Kommunikation
(>>> 11.12 "Kommunikation" Seite 448)

Operatoren	
Arithmetische Operatoren	(>>> 11.13.1 "Arithmetische Operatoren" Seite 448)
Geometrischer Operator	(>>> 11.13.2 "Geometrischer Operator" Seite 449)
Vergleichsoperatoren	(>>> 11.13.3 "Vergleichsoperatoren" Seite 453)
Logische Operatoren	(>>> 11.13.4 "Logische Operatoren" Seite 453)
Bit-Operatoren	(>>> 11.13.5 "Bit-Operatoren" Seite 454)
Prioritäten der Operatoren	(>>> 11.13.6 "Priorität der Operatoren" Seite 456)

Systemfunktionen	
DELETE_BACKWARD_BUFFER()	(>>> 11.14.1 "DELETE_BACKWARD_BUFFER()" Seite 457)
ROB_STOP()	(>>> 11.14.2 "ROB_STOP() und ROB_STOP_RELEASE()" Seite 458)
SET_BRAKE_DELAY()	(>>> 11.14.3 "SET_BRAKE_DELAY()" Seite 459)
VARSTATE()	(>>> 11.14.4 "VARSTATE()" Seite 462)

Stringvariablen manipulieren
(>>> 11.15 "Stringvariablen bearbeiten" Seite 464)

11.2 Zeichen und Schriftarten

In den Syntaxbeschreibungen werden folgende Zeichen und Schriftarten verwendet:

Syntax-Element	Darstellung
KRL-Code	■ Schriftart Courier ■ Großschreibung Beispiele: `GLOBAL`; `ANIN ON`; `OFFSET`
Elemente, die durch programmspezifische Angaben ersetzt werden müssen	■ Kursiv ■ Groß-/Kleinschreibung Beispiele: *Strecke*; *Zeit*; *Format*
Optionale Elemente	■ In spitzen Klammern Beispiel: <`STEP` *Schrittweite*>
Elemente, die sich gegenseitig ausschließen	■ Getrennt durch das Zeichen "|" Beispiel: `IN` |`OUT`

11.3 Wichtige KRL-Begriffe

11.3.1 SRC-Dateien und DAT-Dateien

Ein KRL-Programm besteht in der Regel aus einer **SRC-Datei** und einer **DAT-Datei** gleichen Namens.

- SRC-Datei: Enthält den Programmcode.
- DAT-Datei: Enthält permanente Daten und Punktkoordinaten. Die DAT-Datei wird auch **Datenliste** genannt.

Die SRC-Datei und die dazugehörige DAT-Datei heißen zusammen **Modul**.

Abhängig von der Benutzergruppe werden Programme im Navigator als Module oder als einzelne Dateien angezeigt:

- Benutzergruppe "Anwender"

  Ein Programm wird als Modul angezeigt. Die SRC-Datei und die DAT-Datei existieren im Hintergrund. Sie sind für den Benutzer nicht sichtbar und können nicht einzeln bearbeitet werden.
- Benutzergruppe "Experte"

  Defaultmäßig werden die SRC-Datei und die DAT-Datei einzeln angezeigt. Sie können einzeln bearbeitet werden.

11.3.2 Namenskonventionen und Schlüsselwörter

Namen

Beispiele für Namen in KRL: Variablennamen, Programmnamen, Punktnamen

- Namen in KRL dürfen maximal 24 Zeichen lang sein.

  Teilweise sind weniger als 24 Zeichen erlaubt, z. B. in Inline-Formularen maximal 23 Zeichen.
- Namen in KRL dürfen Buchstaben (A-Z), Ziffern (0-9) sowie die Sonderzeichen "_" und "$" enthalten.
- Namen in KRL dürfen nicht mit Ziffern beginnen.
- Namen in KRL dürfen keine Schlüsselwörter sein.

Für Inline-Formulare in Technologiepaketen können andere Einschränkungen gelten.

Die Namen aller Systemvariablen beginnen mit dem $-Zeichen. Um Verwechslungen zu vermeiden, bei benutzerdefinierten Variablen den Namen nicht mit diesem Zeichen beginnen.

Schlüsselwörter

Schlüsselwörter sind Buchstabenfolgen mit einer fest zugeordneten Bedeutung. Sie dürfen in Programmen nicht anders als in dieser Bedeutung verwendet werden. Die Groß- und Kleinschreibung ist nicht relevant. Ein Schlüsselwort gilt in jeder Schreibweise als Schlüsselwort.

Beispiel: Die Buchstabenfolge CASE ist Bestandteil der KRL-Syntax SWITCH … CASE … ENDSWITCH. Deshalb darf CASE nicht anderweitig verwendet werden, beispielsweise als Variablenname.

Im System werden Schlüsselwörter in reservierte und nicht reservierte Schlüsselwörter unterschieden:

- Reservierte Schlüsselwörter

  Diese dürfen nur in der festgelegten Bedeutung verwendet werden.
- Nicht reservierte Schlüsselwörter

  Bei nicht reservierten Schlüsselwörtern ist die Bedeutung auf einen bestimmten Kontext beschränkt. Außerhalb dieses Kontextes wird ein nicht reserviertes Schlüsselwort vom Compiler als Name interpretiert.

In der Praxis ist es nicht sinnvoll, zwischen reservierten und nicht reservierten Schlüsselwörtern zu unterscheiden. Um Fehlermeldungen oder Compiler-Probleme zu vermeiden, verwendet man deshalb Schlüsselwörter nie anders als in der festgelegten Bedeutung.

Übersicht wichtiger Schlüsselwörter:

Alle nicht programmspezifischen Elemente der KRL-Syntax, die in dieser Dokumentation beschrieben wird, sind Schlüsselwörter.

Darüber hinaus sind folgende wichtige Schlüsselwörter zu nennen:

AXIS	ENDFCT
BOOL	ENDFOR
CHAR	ENDIF
CAST_FROM	ENDLOOP
CAST_TO	ENDSWITCH
CCLOSE	ENDWHILE
CHANNEL	EXT
CIOCTL	EXTFCT
CONFIRM	FALSE

CONST	FRAME
COPEN	GLOBAL
CREAD	INT
CWRITE	MAXIMUM
DEF	MINIMUM
DEFAULT	POS
DEFDAT	PRIO
DEFFCT	PUBLIC
E6AXIS	SREAD
E6POS	SWRITE
END	REAL
ENDDAT	TRUE

11.3.3 Datentypen

Übersicht

Es gibt 2 Arten von Datentypen:

- Benutzerdefinierte Datentypen

  Benutzerdefinierte Datentypen werden immer aus den Datentypen ENUM oder STRUC hergeleitet.
- Vordefinierte Datentypen, z. B.:
  - Einfache Datentypen
  - Datentypen für die Bewegungsprogrammierung

Folgende einfache Datentypen sind vordefiniert:

Datentyp	Schlüsselwort	Beschreibung
Integer	`INT`	Ganze Zahl ■ **$-2^{31}-1$ … $2^{31}-1$** Beispiele: 1; 32; 345
Real	`REAL`	Gleitkommazahl ■ **+1.1E-38 … +3.4E+38** Beispiele: 1.43; 38.50; 300.25
Boolean	`BOOL`	Logischer Zustand ■ **TRUE** ■ **FALSE**
Character	`CHAR`	1 Zeichen ■ **ASCII-Zeichen** Beispiele: "A"; "1"; "q"

Folgende Datentypen für die Bewegungsprogrammierung sind vordefiniert:

Strukturtyp AXIS

A1 bis A6 sind Winkelwerte (rotatorische Achsen) oder Translationswerte (translatorische Achsen) zum achsspezifischen Verfahren der Roboterachsen 1 bis 6.

```
STRUC AXIS REAL A1, A2, A3, A4, A5, A6
```

Strukturtyp E6AXIS

E1 bis E6 sind Winkel- oder Translationswerte der Zusatzachsen 7 bis 12.

```
STRUC E6AXIS REAL A1, A2, A3, A4, A5, A6, E1, E2, E3, E4, E5, E6
```

Strukturtyp FRAME

X, Y, Z sind Raumkoordinaten und A, B, C die Orientierung des Koordinatensystems.

```
STRUC FRAME REAL X, Y, Z, A, B, C
```

Strukturtypen POS und E6POS

S (Status) und T (Turn) legen Achsstellungen eindeutig fest.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

11.3.4 Geltungsbereiche

Lokal

Datenobjekt	Geltungsbereich
Variable Signal	■ Wenn das Objekt in einer SRC-Datei definiert wurde: Es ist in der Programmroutine gültig, in der es definiert wurde, d. h. zwischen DEF und END (Hauptprogramm ODER lokales Unterprogramm). Variablen, die in einer SRC-Datei definiert sind, werden "Laufzeitvariablen" genannt. ■ Wenn das Objekt in einer DAT-Datei definiert wurde: Es ist in der SRC-Datei gültig, die zur DAT-Datei gehört.
Konstante	Gültig in dem Modul, zu dem die Datenliste gehört, in der sie deklariert wurde.
Benutzerdefinierter Datentyp	■ Wenn der Datentyp in einer SRC-Datei definiert wurde: Er ist ab der Programmebene gültig, in der er definiert wurde. ■ Wenn der Datentyp in einer DAT-Datei definiert wurde: Er ist in der SRC-Datei gültig, die zur DAT-Datei gehört.
Unterprogramm	Gültig im Hauptprogramm der gemeinsamen SRC-Datei.
Funktion	Gültig im Hauptprogramm der gemeinsamen SRC-Datei.
Interrupt	Gültig ab der Programmebene, in der er deklariert wurde.

Global

Immer global gültig sind:

- Das erste Programm in einer SRC-Datei. Es trägt defaultmäßig den Namen der SRC-Datei.
- Vordefinierte Datentypen
- KRL-Systemvariablen
- Variablen und Signale, die in der $CONFIG.DAT definiert wurden

Die unter "Lokal" genannten Datenobjekte können global verfügbar gemacht werden.

- (>>> 11.3.4.1 "Unterprogramme, Funktionen, Interrupts global verfügbar machen" Seite 376)
- (>>> 11.3.4.2 "Variablen, Konstanten, Signale, Benutzer-Datentypen global verfügbar machen" Seite 377)

Wenn ein lokales und ein globales Objekt mit dem gleichen Namen existieren, dann verwendet der Compiler das lokale Objekt in dessen Geltungsbereich.

11.3.4.1 Unterprogramme, Funktionen, Interrupts global verfügbar machen

Bei der Vereinbarung das Schlüsselwort GLOBAL verwenden.

Beispiel Unterprogramm:

```
...
END
------------------------------
GLOBAL DEF MY_SUBPROG
...
```

Beispiel Funktion:

```
...
END
------------------------------
GLOBAL DEFFCT INT MY_FCT(my_var:IN)
...
```

Beispiel Interrupt:

```
GLOBAL INTERRUPT DECL 23 WHEN $IN[12]==TRUE DO UP1(20,VALUE)
```

11.3.4.2 Variablen, Konstanten, Signale, Benutzer-Datentypen global verfügbar machen

Variablen, Signale und benutzerdefinierte Datentypen können über eine Datenliste oder über die $CONFIG.DAT global verfügbar gemacht werden.

Konstanten müssen immer in einer Datenliste deklariert und gleichzeitig initialisiert werden. Sie können deshalb nur über eine Datenliste global verfügbar gemacht werden.

Datenliste

Das Objekt über eine Datenliste global verfügbar machen:

1. In der Datenliste im Programmkopf das Schlüsselwort PUBLIC hinzufügen:

```
DEFDAT MY_PROG PUBLIC
```

2. Bei der Vereinbarung das Schlüsselwort GLOBAL verwenden.
   Beispiel (Vereinbarung einer Variablen):

```
DEFDAT MY_PROG PUBLIC
EXTERNAL DECLARATIONS
DECL GLOBAL INT counter
...
ENDDAT
```

GLOBAL kann für Variablen, Signale und benutzerdefinierte Datentypen nur verwendet werden, wenn sie in einer Datenliste vereinbart sind.

PUBLIC dient ausschließlich dem hier dargestellten Zweck, nämlich in Datenlisten zusammen mit GLOBAL bestimmte Datenobjekt global verfügbar zu machen. PUBLIC allein hat keine Auswirkung.

$CONFIG.DAT

- Das Objekt in der $CONFIG.DAT im Abschnitt USER GLOBALS deklarieren.

Das Schlüsselwort GLOBAL ist hier nicht notwendig und kann hier auch nicht verwendet werden.

Einschränkung

Datentypen, die in einer Datenliste mit dem Schlüsselwort GLOBAL definiert wurden, dürfen in der $CONFIG.DAT nicht verwendet werden.

Beispiel:

In DEFDAT PROG() wurde der Aufzählungstyp SWITCH_TYP mit dem Schlüsselwort GLOBAL definiert:

```
DEFDAT PROG()
```

```
GLOBAL ENUM SWITCH_TYP ON, OFF
...
```

Wenn dieser Datentyp in der $CONFIG.DAT verwendet wird, meldet der Compiler den Fehler "*Typ unbekannt: *** DECL SWITCH_TYP MY_VAR*".

```
DEFDAT $CONFIG

DECL SWITCH_TYP MY_VAR
...
```

11.3.5 Konstanten

Der Wert einer Konstanten kann beim Programmlauf nach der Initialisierung nicht mehr verändert werden. Mit Konstanten kann man verhindern, dass ein Wert während des Programmlaufs versehentlich geändert wird.

Konstanten müssen in einer Datenliste deklariert und gleichzeitig initialisiert werden. Vor dem Datentyp muss das Schlüsselwort CONST stehen.

`DECL <GLOBAL> CONST` *Datentyp Variablenname = Wert*

11.4 Variablen und Vereinbarungen

11.4.1 DECL

Beschreibung Deklaration von Variablen, Feldern und Konstanten

Syntax

Deklaration von Variablen

Deklaration von Variablen in Programmen:

`<DECL>` *Datentyp Name1 <, ..., NameN>*

Deklaration von Variablen in Datenlisten:

`<DECL> <GLOBAL>` *Datentyp Name1 <, ..., NameN>*

Deklaration von Variablen in Datenlisten, mit gleichzeitiger Initialisierung:

`<DECL> <GLOBAL>` *Datentyp Name = Wert*

Bei der Deklaration mit gleichzeitiger Initialisierung ist für jede Variable eine eigene DECL-Vereinbarung notwendig. Es ist nicht möglich, mehrere Variablen mit einer einzigen DECL-Vereinbarung zu deklarieren und zu initialisieren.

Deklaration von Feldern

Deklaration von Feldern in Programmen:

`<DECL>` *Datentyp Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >*

Deklaration von Feldern in Datenlisten:

`<DECL> <GLOBAL>` *Datentyp Name1 [Dimension1 <, ..., Dimension3>] <, ..., NameN [DimensionN1 <,..., DimensionN3>] >*

Zur Deklaration von Feldern oder von Konstanten-Feldern in Datenlisten mit gleichzeitiger Initialisierung:

- Es darf nicht in einer Zeile deklariert und initialisiert werden. Die Initialisierung muss aber direkt auf die Zeile mit der Deklaration folgen. Zeilen dazwischen sind nicht erlaubt, auch keine Leerzeilen.
- Wenn mehrere Elemente eines Feldes initialisiert werden, müssen die Elemente in aufsteigender Reihenfolge der Feldindizes (angefangen beim rechten Feldindex) angegeben werden.

- Wenn alle Feldelemente eines Feldes vom Typ CHAR mit der gleichen Zeichenkette vorbesetzt werden sollen, muss nicht jedes Feldelement einzeln initialisiert werden. Man lässt den rechten Feldindex weg. (Bei einem eindimensionalen Feldindex schreibt man keinen Index.)

Deklaration von Feldern in Datenlisten, mit gleichzeitiger Initialisierung:

```
<DECL> <GLOBAL> Datentyp Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1> ] = Wert1
<Name [1 <, 1, 2> ] = Wert2>
. . .
Name [Dimension1 <, Dimension2, Dimension3> ] = WertN
```

Deklaration von Konstanten-Feldern in Datenlisten, mit gleichzeitiger Initialisierung:

```
DECL <GLOBAL> CONST Datentyp Name [Dimension1 <,..., Dimension3> ]
Name [1 <, 1, 1> ] = Wert1
<Name [1 <, 1, 2> ] = Wert2>
. . .
Name [Dimension1 <, Dimension2, Dimension3> ] = WertN
```

Erläuterung der Syntax

Element	Beschreibung
DECL	DECL kann weggelassen werden, wenn *Datentyp* ein vordefinierter Datentyp ist. Wenn *Datentyp* ein benutzerdefinierter Datentyp ist, ist DECL obligatorisch.
GLOBAL	(>>> 11.3.4 "Geltungsbereiche" Seite 376)
CONST	Das Schlüsselwort CONST darf nur in Datenlisten verwendet werden.
Datentyp	Angabe des gewünschten Datentyps
Name	Name des Objekts (Variable, Feld oder Konstante), das deklariert wird
Dimension	Typ: INT *Dimension* legt die Anzahl der Feldelemente für die jeweilige Dimension fest. Felder haben mindestens 1, maximal 3 Dimensionen.
Wert	Der Datentyp von *Wert* muss mit *Datentyp* verträglich, aber nicht identisch sein. Bei verträglichen Datentypen führt das System automatisch eine Typanpassung durch.

Beispiel 1

Deklarationen mit vordefinierten Datentypen. Das Schlüsselwort DECL kann auch weggelassen werden.

```
DECL INT X
DECL INT X1, X2
DECL REAL ARRAY_A[7], ARRAY_B[5], A
```

Beispiel 2

Deklarationen von Feldern mit gleichzeitiger Initialisierung (nur in Datenlisten möglich).

```
INT A[7]
A[1]=27
A[2]=313
A[6]=11
CHAR TEXT1[80]
TEXT1[]="message"
CHAR TEXT2[2,80]
TEXT2[1,]="first message"
TEXT2[2,]="second message"
```

11.4.2 ENUM

Beschreibung

Definition eines Aufzählungstyps (=ENUM-Datentyp)

Syntax

`<GLOBAL> ENUM` *NameEnumtyp Konstante1*`<, ..., KonstanteN>`

Erläuterung der Syntax

Element	Beschreibung
`GLOBAL`	(>>> 11.3.4 "Geltungsbereiche" Seite 376) **Hinweis:** Datentypen, die mit dem Schlüsselwort GLOBAL definiert wurden, dürfen in der $CONFIG.DAT nicht verwendet werden.
NameEnumtyp	Name des neuen Aufzählungstyps. Empfehlung: Für benutzerdefinierte Datentypen Namen vergeben, die auf _TYP enden, damit sie von Variablennamen unterscheidbar sind.
Konstante	Die Konstanten sind die Werte, die eine Variable des Aufzählungstyps annehmen kann. Jede Konstante darf bei der Definition des Aufzählungstyps nur einmal vorkommen.

Beispiel 1

Definiton eines Aufzählungstyps mit dem Namen COUNTRY_TYP.

```
ENUM COUNTRY_TYP SWITZERLAND, AUSTRIA, ITALY, FRANCE
```

Deklaration einer Variablen vom Typ COUNTRY_TYP:

```
DECL COUNTRY_TYP MYCOUNTRY
```

Initialisierung der Variablen vom Typ COUNTRY_TYP:

```
MYCOUNTRY = #AUSTRIA
```

Beispiel 2

Ein Aufzählungstyp mit dem Namen SWITCH_TYP und den Konstanten ON und OFF wird definiert.

```
DEF PROG()
ENUM SWITCH_TYP ON, OFF
DECL SWITCH_TYP GLUE
   IF A>10 THEN
      GLUE=#ON
   ELSE
      GLUE=#OFF
   ENDIF
END
```

Einschränkung

Datentypen, die in einer Datenliste mit dem Schlüsselwort GLOBAL definiert wurden, dürfen in der $CONFIG.DAT nicht verwendet werden.

Beispiel:

In DEFDAT PROG() wurde der Aufzählungstyp SWITCH_TYP mit dem Schlüsselwort GLOBAL definiert:

```
DEFDAT PROG()

GLOBAL ENUM SWITCH_TYP ON, OFF
...
```

Wenn dieser Datentyp in der $CONFIG.DAT verwendet wird, meldet der Compiler den Fehler "*Typ unbekannt: *** DECL SWITCH_TYP MY_VAR*".

```
DEFDAT $CONFIG
```

```
DECL SWITCH_TYP MY_VAR
...
```

11.4.3 STRUC

Beschreibung

Definition eines Strukturtyps (=STRUC-Datentyp). Mehrere Datentypen werden hierbei zu einem neuen Datentyp zusammengefasst.

Syntax

`<GLOBAL> STRUC` *NameStrukturtyp Datentyp1 Komponente1A*<, *Komponente1B*, ...> < , *Datentyp2 Komponente2A*<, *Komponente2B*, ...>>

Erläuterung der Syntax

Element	Beschreibung
GLOBAL	(>>> 11.3.4 "Geltungsbereiche" Seite 376) **Hinweis:** Datentypen, die mit dem Schlüsselwort GLOBAL definiert wurden, dürfen in der $CONFIG.DAT nicht verwendet werden.
NameStrukturtyp	Name des neuen Strukturtyps. Die Namen von benutzerdefinierten Datentypen sollten auf _TYP enden, damit sie von Variablennamen unterscheidbar sind.
Datentyp	TYP: Beliebiger Datentyp Auch Strukturtypen sind als Datentypen zugelassen.
Komponente	Name der Komponente. Darf innerhalb des Strukturtyps nur einmal vorkommen. Felder sind als Komponenten eines Strukturtyps nur zugelassen, wenn sie den Typ CHAR haben und eindimensional sind. In der Definition des Strukturtyps folgt dazu auf den Namen des Feldes die Feldgrenze in eckigen Klammern.

Wertzuweisung

Es gibt 2 Möglichkeiten, um Variablen, die auf einem STRUC-Datentyp basieren, Werte zuzuweisen:

- Mehreren Komponenten einer Variablen Werte zuweisen: mit einem **Aggregat**
- Einer einzelnen Komponente einer Variablen einen Wert zuweisen: mit dem **Punkt-Separator**

Hinweise zum Aggregat:

- Die Werte eines Aggregats können einfache Konstanten oder selbst Aggregate sein, sie können jedoch keine Variablen sein (siehe auch Beispiel 3).
- In einem Aggregat müssen nicht sämtliche Komponenten der Struktur angegeben werden.
- Die Komponenten brauchen nicht in der Reihenfolge angegeben zu werden, in der sie definiert wurden.
- Jede Komponente darf in einem Aggregat nur einmal enthalten sein.
- Am Anfang eines Aggregates kann, durch Doppelpunkt abgetrennt, der Name des Strukturtyps angegeben sein.

Beispiel 1

Definition eines Strukturtyps CAR_TYP mit den Komponenten AIR_COND, YEAR und PRICE.

```
STRUC CAR_TYP BOOL AIR_COND, INT YEAR, REAL PRICE
```

Deklaration einer Variablen vom Typ CAR_TYP:

```
DECL CAR_TYP MYCAR
```

Initialisierung der Variablen MYCAR vom Typ CAR_TYP mit einem **Aggregat**:

```
MYCAR = {CAR_TYP: PRICE 15000, AIR_COND TRUE, YEAR 2003}
```

Eine Variable, die auf einem Strukturtyp basiert, muss nicht mit einem Aggregat initialisiert werden. Es ist ebenso möglich, die Komponenten einzeln mit dem Punkt-Separator zu initialisieren.

Änderung einer einzelnen Komponente mittels **Punkt-Separator**:

```
MYCAR.AIR_COND = FALSE
```

Beispiel 2

Definition eines Strukturtyps S_TYP mit der Komponente NUMBER vom Datentyp REAL und der Feldkomponente TEXT[80] vom Datentyp CHAR.

```
STRUC S_TYP REAL NUMBER, CHAR TEXT[80]
```

Beispiel 3

Beispiel für Aggregate als Werte eines Aggregats:

```
STRUC INNER_TYP INT A, B, C
STRUC OUTER_TYP INNER_TYP Q, R
DECL OUTER_TYP MYVAR
...
MYVAR = {Q {A 1, B 4}, R {A 3, C 2}}
```

11.5 Bewegungsprogrammierung: PTP, LIN, CIRC

11.5.1 PTP

Beschreibung

Führt eine Punkt-zu-Punkt-Bewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind absolut.

Syntax

`PTP` *Zielpunkt* <*Überschleifen*>

Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Typ: POS, E6POS, AXIS, E6AXIS, FRAME Der Zielpunkt kann kartesisch oder achsspezifisch angegeben werden. Kartesische Koordinaten beziehen sich auf das BASE-Koordinatensystem. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann übernimmt die Steuerung für die fehlenden Komponenten die Werte der vorhergehenden Position.
Überschleifen	*Überschleifen* bewirkt, dass der Zielpunkt überschliffen wird. Gleichzeitig legt dieser Parameter fest, wann das Überschleifen frühestens beginnt. (>>> 11.5.5 "Überschleif-Parameter für PTP, LIN, CIRC und …_REL" Seite 387)

Beispiel 1

Zielpunkt angegeben in kartesischen Koordinaten.

```
PTP {X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0,S 'B010',T 'B1010'}
```

Beispiel 2

Zielpunkt angegeben in achsspezifischen Koordinaten. Der Zielpunkt wird überschliffen.

```
PTP {A1 10,A2 -80.6,A3 -50,A4 0,A5 14.2, A6 0} C_DIS
```

Beispiel 3

Zielpunkt angegeben mit nur 2 Komponenten. Für die restlichen Komponenten übernimmt die Steuerung die Werte der vorhergehenden Position.

```
PTP {Z 500,X 123.6}
```

11.5.2 PTP_REL

Beschreibung

Führt eine Punkt-zu-Punkt-Bewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind relativ zur aktuellen Position.

> Eine REL-Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn eine REL-Bewegung abgebrochen wurde, fährt der Roboter deswegen von der Abbruch-Position aus noch einmal die komplette REL-Bewegung.

Zum Verhalten der Robotersteuerung bei endlos drehenden rotatorischen Achsen: (>>> 11.5.6 "REL-Bewegungen bei endlos drehenden rotatorischen Achsen" Seite 388)

Syntax

`PTP_REL` *Zielpunkt* <*Überschleifen*> <#BASE|#TOOL>

Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Typ: POS, E6POS, AXIS, E6AXIS Der Zielpunkt kann kartesisch oder achsspezifisch angegeben werden. Die Steuerung interpretiert die Koordinaten als relativ zur aktuellen Position. Kartesische Koordinaten beziehen sich auf das BASE-Koordinatensystem. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann setzt die Steuerung die fehlenden Komponenten auf den Wert 0. Das heißt, die absoluten Werte dieser Komponenten bleiben unverändert.
Überschleifen	*Überschleifen* bewirkt, dass der Zielpunkt überschliffen wird. Gleichzeitig legt dieser Parameter fest, wann das Überschleifen frühestens beginnt. (>>> 11.5.5 "Überschleif-Parameter für PTP, LIN, CIRC und …_REL" Seite 387)
#BASE, #TOOL	Nur zulässig, wenn der Zielpunkt kartesisch angegeben wurde. ■ #BASE (Default): Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zur physikalischen Basis gehört. ■ #TOOL: Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zum physikalischen Werkzeug gehört. $IPO_MODE hat keinen Einfluss auf die Bedeutung von #BASE und #TOOL.

Beispiel 1

Die Achse 2 wird um 30 Grad in negative Richtung verfahren. Alle anderen Achsen bewegen sich nicht.

```
PTP_REL {A2 -30}
```

Beispiel 2

Der Roboter bewegt sich von der aktuellen Position aus um 100 mm in X-Richtung und um 200 mm in negative Z-Richtung. Y, A, B, C, und S bleiben konstant. T wird nach dem kürzesten Weg berechnet.

```
PTP_REL {X 100,Z -200}
```

11.5.3 LIN, CIRC

Beschreibung

LIN:

LIN führt eine Linearbewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind absolut.

CIRC:

CIRC führt eine Kreisbewegung aus. Damit die Steuerung die Kreisbewegung berechnen kann, müssen ein Hilfspunkt und ein Zielpunkt angegeben werden.

Die Koordinaten von Hilfs- und Zielpunkt sind absolut.

Syntax

LIN:

`LIN` *Zielpunkt* <*Überschleifen*>

CIRC:

`CIRC` *Hilfspunkt, Zielpunkt*<`,` `CA` *Kreiswinkel*> <*Überschleifen*>

Erläuterung der Syntax

Element	Beschreibung
Hilfspunkt	Typ: POS, E6POS, FRAME Wenn nicht alle Komponenten des Hilfspunkts angegeben werden, dann übernimmt die Steuerung für die fehlenden Komponenten die Werte der vorhergehenden Position. Die Orientierungswinkel sowie die Angaben Status und Turn innerhalb eines Hilfspunktes werden grundsätzlich ignoriert. Der Hilfspunkt kann nicht überschliffen werden. Er wird immer genau angefahren. Die Koordinaten beziehen sich auf das BASE-Koordinatensystem.
Zielpunkt	Typ: POS, E6POS, FRAME Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann übernimmt die Steuerung für die fehlenden Komponenten die Werte der vorhergehenden Position. Die Angaben Status und Turn innerhalb eines Zielpunkts vom Typ POS oder E6POS werden ignoriert. Die Koordinaten beziehen sich auf das BASE-Koordinatensystem.
Kreiswinkel	Einheit: Grad; ohne Begrenzung (>>> 9.9 "Kreiswinkel" Seite 311)
Überschleifen	*Überschleifen* bewirkt, dass der Zielpunkt überschliffen wird. Gleichzeitig legt dieser Parameter fest, wann das Überschleifen frühestens beginnt. (>>> 11.5.5 "Überschleif-Parameter für PTP, LIN, CIRC und …_REL" Seite 387)

Beispiele

Zielpunkt mit zwei Komponenten. Für die restlichen Komponenten übernimmt die Steuerung die Werte der vorhergehenden Position.

```
LIN {Z 500,X 123.6}
```

Der Zielpunkt der Kreisbewegung wird durch einen Kreiswinkel von 260° festgelegt. Der Zielpunkt wird überschliffen.

```
CIRC {X 5,Y 0, Z 9.2},{X 12.3,Y 0,Z -5.3,A 9.2,B -5,C 20}, CA 260
C_ORI
```

11.5.4 LIN_REL, CIRC_REL

Beschreibung

LIN_REL:

LIN_REL führt eine Linearbewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind relativ zur aktuellen Position.

CIRC_REL:

CIRC_REL führt eine Kreisbewegung aus. Damit die Steuerung die Kreisbewegung berechnen kann, müssen ein Hilfspunkt und ein Zielpunkt angegeben werden. Die Koordinaten von Hilfs- und Zielpunkt sind relativ zur aktuellen Position.

Eine REL-Anweisung bezieht sich immer auf die aktuelle Roboterposition. Wenn eine REL-Bewegung abgebrochen wurde, fährt der Roboter deswegen von der Abbruch-Position aus noch einmal die komplette REL-Bewegung.

Zum Verhalten der Robotersteuerung bei endlos drehenden rotatorischen Achsen: (>>> 11.5.6 "REL-Bewegungen bei endlos drehenden rotatorischen Achsen" Seite 388)

Syntax

LIN_REL:

`LIN_REL` *Zielpunkt* <*Überschleifen*> <#BASE|#TOOL>

CIRC_REL:

`CIRC_REL` *Hilfspunkt, Zielpunkt*<`,` `CA` *Kreiswinkel*> <*Überschleifen*> <#BASE|#TOOL>

Erläuterung der Syntax

Element	Beschreibung
Hilfspunkt	Typ: POS, E6POS, FRAME Der Hilfspunkt muss in kartesischen Koordinaten angegeben werden. Die Steuerung interpretiert die Koordinaten als relativ zur aktuellen Position. Die Koordinaten beziehen sich auf das BASE-Koordinatensystem. Wenn $ORI_TYPE, Status und/oder Turn angeben werden, dann werden diese Angaben ignoriert. Wenn nicht alle Komponenten des Hilfspunkts angegeben werden, dann setzt die Steuerung die fehlenden Komponenten auf den Wert 0. Das heißt, die absoluten Werte dieser Komponenten bleiben unverändert. Die Orientierungswinkel sowie die Angaben Status und Turn innerhalb eines Hilfspunktes werden ignoriert. Der Hilfspunkt kann nicht überschliffen werden. Er wird immer genau angefahren.
Zielpunkt	Typ: POS, E6POS, FRAME Der Zielpunkt muss in kartesischen Koordinaten angegeben werden. Die Steuerung interpretiert die Koordinaten als relativ zur aktuellen Position. CIRC_REL: Die Koordinaten beziehen sich auf das BASE-Koordinatensystem. LIN_REL: Die Koordinaten können sich auf das BASE- oder auf das TOOL-Koordinatensystem beziehen. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann setzt die Steuerung die fehlenden Komponenten auf den Wert 0. Das heißt, die absoluten Werte dieser Komponenten bleiben unverändert. Die Angaben Status und Turn innerhalb eines Zielpunkts vom Typ POS oder E6POS werden ignoriert.
Kreiswinkel	Einheit: Grad; ohne Begrenzung (>>> 9.9 "Kreiswinkel" Seite 311)
Überschleifen	*Überschleifen* bewirkt, dass der Zielpunkt überschliffen wird. Gleichzeitig legt dieser Parameter fest, wann das Überschleifen frühestens beginnt. (>>> 11.5.5 "Überschleif-Parameter für PTP, LIN, CIRC und …_REL" Seite 387)
`#BASE`, `#TOOL`	■ #BASE (Default): Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zur physikalischen Basis gehört. ■ #TOOL: Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zum physikalischen Werkzeug gehört. $IPO_MODE hat keinen Einfluss auf die Bedeutung von #BASE und #TOOL.

Informationen zu $APO sind in der Dokumentation **Systemvariablen** zu finden.

Beispiele

Beispiel 1:

Der TCP bewegt sich von der aktuellen Position aus um 100 mm in X-Richtung und um 200 mm in negative Z-Richtung im BASE-Koordinatensystem. Y, A, B, C und S bleiben konstant. T ergibt sich durch die Bewegung.

```
LIN_REL {X 100,Z -200}
```

Beispiel 2:

Der TCP bewegt sich von der aktuellen Position aus um 100 mm in negative X-Richtung im TOOL-Koordinatensystem. Y, Z, A, B, C und S bleiben konstant. T ergibt sich durch die Bewegung.

Dieses Beispiel ist geeignet, um das Werkzeug entlang der Stoßrichtung rückwärts zu bewegen. Voraussetzung ist, dass die Werkzeugstoßrichtung in X-Richtung vermessen wurde.

```
LIN_REL {X -100} #TOOL
```

Beispiel 3:

Der Zielpunkt der Kreisbewegung wird durch einen Kreiswinkel von 500° festgelegt. Der Zielpunkt wird überschliffen.

```
CIRC_REL {X 100,Y 3.2,Z -20},{Y 50},CA 500 C_VEL
```

11.5.5 Überschleif-Parameter für PTP, LIN, CIRC und …_REL

Parameter

Nicht jeder Parameter kann in jeder Anweisung verwendet werden.

Parameter	Beschreibung
C_PTP	Das Überschleifen beginnt frühestens, wenn die halbe Weglänge zwischen Startpunkt und Zielpunkt zurückgelegt wurde, bezogen auf die Kontur der Bewegung ohne Überschleifen.
C_DIS	Das Überschleifen beginnt frühestens, wenn die Entfernung zum Zielpunkt den Wert von $APO.CDIS unterschreitet.
C_ORI	Das Überschleifen beginnt frühestens, wenn der dominierende Orientierungswinkel den Wert von $APO.CORI unterschreitet.
C_VEL	Das Überschleifen beginnt frühestens, wenn die Geschwindigkeit in der Abbremsphase zum Zielpunkt hin den Wert von $APO.CVEL unterschreitet.

Informationen zu $APO sind in der Dokumentation **Systemvariablen** zu finden.

PTP, PTP_REL

Beim **PTP-PTP**-Überschleifen muss der Parameter C_PTP oder C_DIS sein. Falls ein zweiter Parameter angegeben ist, ignoriert die Robotersteuerung diesen.

Beim **PTP-CP**-Überschleifen können zwei Parameter angegeben werden. Von den beiden Parametern wirkt sich derjenige aus, der in der jeweiligen Situation den kleineren Überschleif-Radius ergibt.

Mögliche Kombinationen für das PTP-CP-Überschleifen:

1. Parameter → / 2. Parameter ↓	C_PTP	C_DIS
(ohne)	Möglich	Möglich
C_DIS	Möglich	Nicht möglich!

1. Parameter → / 2. Parameter ↓	C_PTP	C_DIS
C_VEL	Möglich	Möglich
C_ORI	Möglich	Möglich

Beispiel: PTP-CP-Überschleifen

```
PTP XP1 C_PTP C_DIS
LIN XP2
```

Die Robotersteuerung errechnet den Überschleif-Radius, der sich unter den aktuellen Gegebenheiten (Geschwindigkeit usw.) durch jeden der beiden Parameter C_PTP und C_DIS ergeben würde. Nur der kleinere der beiden Radien wirkt sich dann tatsächlich aus. Er ist die Grenze, bei der das Überschleifen frühestens beginnt.

LIN, CIRC, LIN_REL, CIRC_REL

Beim **CP-CP**-Überschleifen muss der Parameter C_DIS, C_VEL oder C_ORI sein. Falls ein zweiter Parameter angegeben ist, ignoriert die Robotersteuerung diesen.

Beim **CP-PTP**-Überschleifen können zwei Parameter angegeben werden. Von den beiden Parametern wirkt sich derjenige aus, der in der jeweiligen Situation den kleineren Überschleif-Radius ergibt.

Mögliche Kombinationen für das CP-PTP-Überschleifen:

1. Parameter → / 2. Parameter ↓	C_DIS	C_VEL	C_ORI
(ohne)	Möglich	Möglich	Möglich
C_PTP	Möglich	Möglich	Möglich
C_DIS	Möglich	Möglich	Möglich

11.5.6 REL-Bewegungen bei endlos drehenden rotatorischen Achsen

Beschreibung

Bewegung	Beschreibung
SPTP_REL Nur für Zusatzachsen: SLIN_REL, SCIRC_REL, SPL_REL	Die Zielposition ergibt sich direkt durch die Addition der Startposition und der Angabe in der REL-Anweisung. Die Gesamtlänge des Wegs, der sich ergibt, spielt keine Rolle.
PTP_REL Nur für Zusatzachsen: LIN_REL, CIRC_REL	Die Achse fährt nur Positionen in folgendem Intervall an: ■ Von "Startposition minus 180° " ■ Bis "Startposition plus 180° " Wenn die Addition der Startposition und der Angabe in der REL-Anweisung einen Wert außerhalb dieses Intervalls ergibt, dann ergibt sich die tatsächliche Zielposition aus der Differenz des Werts zu 360° oder zu einem Vielfachen von 360°.

Beispiele

A6 und E1 seien endlos drehende rotatorische Achsen mit Startposition 120°.

Die Position sei bei X = 1 500 mm.

Beispiel 1:

Anweisung	Zielposition
SPTP_REL {A6 330}	A6 = 450°
PTP_REL {A6 330}	A6 = 90°

Erläuterung zur Zielposition von PTP_REL:

Das erlaubte Intervall geht von -60° bis 300°. Die Position 450° liegt außerhalb dieses Intervalls und wird deshalb nicht angefahren.

Die Zielposition muss innerhalb des Intervalls liegen UND sich folgendermaßen ergeben:

- 450° ± (x * 360°)

Die Zielposition, die diese Kriterien erfüllt, ist 90°.

450° - (1 * 360°) = 90°

Beispiel 2:

Anweisung	Zielposition
SPTP_REL {A6 550}	A6 = 670°
PTP_REL {A6 550}	A6 = -50°

Beispiel 3:

Anweisung	Zielposition
SPTP_REL {E1 950}	E1 = 1070°
PTP_REL {E1 950}	E1 = -10°

Die Anweisungen enthalten keine Angaben zur Roboterposition. Dies entspricht implizit: {X 0, Y 0, Z 0, A 0, B 0, C 0}

In beiden Fällen bleibt also die kartesische Roboterposition unverändert.

Beispiel 4:

Anweisung	Zielposition
SPTP_REL {A6 0, E1 950}	A6 = 120°, E1 = 1 070°
PTP_REL {A6 0, E1 950}	A6 = 120°, E1 = -10°

Die Roboter-Zielposition ist, wenn nicht spezifiziert, implizit kartesisch, wie bei Beispiel 3 erläutert.

Wenn man jedoch erreichen will, dass statt der kartesischen die achsspezifische Roboterposition unverändert bleibt, muss für mindestens eine Roboterachse explizit eine Nullbewegung angegeben werden, so wie hier in Beispiel 4.

Beispiel 5:

Anweisung	Zielposition
SLIN_REL {X 300, E1 880}	X = 1 800 mm, E1 = 1 000°
LIN_REL {X 300, E1 880}	X = 1 800 mm, E1 = 280°

Zusatzachsen-Bewegungen sind immer achsspezifisch. Sie werden deshalb auch in diesen Anweisungen, die für Roboterpositionen nur kartesische Koordinaten zulassen, in Grad spezifiziert.

11.6 Bewegungsprogrammierung: Spline

11.6.1 SPLINE ... ENDSPLINE

Beschreibung

SPLINE … ENDSPLINE definiert einen CP-Spline-Block. Ein CP-Spline-Block darf enthalten:

- SLIN-, SCIRC- und SPL-Segmente (Anzahl nur durch Speicherkapazität begrenzt.)
- PATH-Trigger
- 1 Zeitblock (TIME_BLOCK …)
  oder 1 Konstantfahrbereich (CONST_VEL …)
- STOP WHEN PATH
- Kommentare
- Leerzeilen

Der Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.

Der Startpunkt eines Spline-Blocks ist der letzte Punkt vor dem Spline-Block.
Der Zielpunkt eines Spline-Blocks ist der letzte Punkt im Spline-Block.
Ein Spline-Block löst keinen Vorlaufstopp aus.

Syntax

```
SPLINE < WITH SysVar1 = Wert1 <, SysVar2 = Wert2, … > >
Segment1
…
<SegmentN>
ENDSPLINE <C_SPL>
```

Erläuterung der Syntax

Element	Beschreibung
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)
Wert	Wertzuweisung an die Systemvariable. Der Wert gilt nicht für Segmente, denen ein eigener Wert zugewiesen wird. Von diesem Fall abgesehen, gilt der Wert wie üblich so lange, bis der Systemvariablen ein neuer Wert zugewiesen wird. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.

Bis zur System Software 8.2 lautete die Kennung für das Überschleifen beim Spline "C_DIS". Wenn in höheren Versionen von 8.x Programme verwendet werden, die auf 8.2 oder älteren Ständen basieren und C_DIS enthalten, braucht dieses nicht in C_SPL geändert werden, sondern kann belassen werden.

Beispiel

```
SPLINE
  SPL P1
  TRIGGER WHEN PATH=GET_PATH() ONSTART DELAY=0 DO <subprog> PRIO=-1
  SPL P2
  SLIN P3
  SPL P4
  SCIRC P5, P6 WITH $VEL.CP=0.2
  SPL P7 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
  SCIRC P8, P9
  SPL P10
ENDSPLINE
```

11.6.2 PTP_SPLINE ... ENDSPLINE

Beschreibung

PTP_SPLINE … ENDSPLINE definiert einen PTP-Spline-Block. Ein PTP-Spline-Block darf enthalten:

- SPTP-Segmente (Anzahl nur durch Speicherkapazität begrenzt.)
- PATH-Trigger
- 1 Zeitblock (TIME_BLOCK …)
- STOP WHEN PATH
- Kommentare
- Leerzeilen

Der Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.

Der Startpunkt eines Spline-Blocks ist der letzte Punkt vor dem Spline-Block.
Der Zielpunkt eines Spline-Blocks ist der letzte Punkt im Spline-Block.
Ein Spline-Block löst keinen Vorlaufstopp aus.

Syntax

`PTP_SPLINE` < `WITH` *SysVar1* = *Wert1* <, *SysVar2* = *Wert2*, … > >

Segment1

…

<*SegmentN*>

`ENDSPLINE` <`C_SPL`>

Erläuterung der Syntax

Element	Beschreibung
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)
Wert	Wertzuweisung an die Systemvariable. Der Wert gilt nicht für Segmente, denen ein eigener Wert zugewiesen wird. Von diesem Fall abgesehen, gilt der Wert wie üblich so lange, bis der Systemvariablen ein neuer Wert zugewiesen wird. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.

Beispiel

```
PTP_SPLINE WITH $ACC_AXIS[1]={CP 20, ORI1 80, ORI2 80}
  SPTP P1
  TRIGGER WHEN PATH=GET_PATH() ONSTART DELAY=0 DO <subprog> PRIO=-1
  SPTP P2
  SPTP P3
  SPTP P4 WITH $ACC_AXIS[1]={CP 10}
ENDSPLINE C_SPL
```

11.6.3 SLIN, SCIRC, SPL

Beschreibung

SLIN, SCIRC:

SLIN und SCIRC können als Segment in einem CP-Spline-Block oder als Einzelbewegung programmiert werden.

Es ist möglich, eine SLIN- oder SCIRC-Einzelbewegung in einen CP-Spline-Block zu kopieren, aber nur, wenn sie keine Zuweisung an Systemvariablen enthält, die dort verboten sind.

SPL:

SPL kann als Segment in einem CP-Spline-Block programmiert werden.

Syntax

SLIN:

`SLIN` *Zielpunkt* <`WITH` *SysVar1* = *Wert1* <, *SysVar2* = *Wert2*, …, >> <`C_SPL`>

SCIRC:

`SCIRC` *Hilfspunkt*, *Zielpunkt* <, `CA` *Kreiswinkel*> <`WITH` *SysVar1* = *Wert1* <,*SysVar2* = *Wert2* , … >> <`C_SPL`>

SPL:

`SPL` *Zielpunkt* < `WITH` *SysVar1* = *Wert1* <,*SysVar2* = *Wert2* , …>>

Erläuterung der Syntax

Element	Beschreibung
Hilfspunkt	Typ: POS, E6POS, FRAME
Zielpunkt	Die Koordinaten beziehen sich auf das BASE-Koordinatensystem. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann übernimmt die Steuerung für die fehlenden Komponenten die Werte der vorhergehenden Position. Wenn diese vorhergehende Position der Zielpunkt eines Kreises mit Kreiswinkel ist, gibt es folgende Unterscheidung: ■ Wenn die vorhergehende Position außerhalb eines Spline-Blocks liegt, werden die Werte des tatsächlich erreichten Zielpunkts übernommen, nicht des programmierten Zielpunkts. ■ Wenn die vorhergehende Position in einem Spline-Block liegt, werden die Werte des programmierten Zielpunkts übernommen, nicht die des tatsächlich erreichten Zielpunkts. Wenn der Robotersteuerung keine vorhergehende Position bekannt ist, werden die fehlenden Komponenten aus der aktuellen Roboterposition übernommen.
Kreiswinkel	Einheit: Grad; ohne Begrenzung (>>> 9.9 "Kreiswinkel" Seite 311)
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)

Element	Beschreibung
Wert	Wertzuweisung an die Systemvariable Bei Segmenten: Die Zuweisung gilt nur für dieses Segment. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. Nur möglich für Einzelbewegungen, nicht für Segmente. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.

Bis zur System Software 8.2 lautete die Kennung für das Überschleifen beim Spline "C_DIS". Wenn in höheren Versionen von 8.x Programme verwendet werden, die auf 8.2 oder älteren Ständen basieren und C_DIS enthalten, braucht dieses nicht in C_SPL geändert werden, sondern kann belassen werden.

Beispiele

```
SCIRC P2, P3 WITH $CIRC_TYPE=#PATH
```

```
SPL P4 WITH $ACC={CP 2.0, ORI1 200, ORI2 200}
```

11.6.4 SLIN_REL, SCIRC_REL, SPL_REL

Beschreibung

SLIN_REL, SCIRC_REL:

SLIN_REL und SCIRC_REL können als Segment in einem CP-Spline-Block oder als Einzelbewegung programmiert werden.

Es ist möglich, eine SLIN_REL- oder SCIRC_REL-Einzelbewegung in einen CP-Spline-Block zu kopieren, aber nur, wenn sie keine Zuweisung an Systemvariablen enthält, die dort verboten sind.

SPL_REL:

SPL_REL kann als Segment in einem CP-Spline-Block programmiert werden.

Zum Verhalten der Robotersteuerung bei endlos drehenden rotatorischen Achsen: (>>> 11.5.6 "REL-Bewegungen bei endlos drehenden rotatorischen Achsen" Seite 388)

Syntax

SLIN_REL:

SLIN_REL *Zielpunkt* <WITH *SysVar1* = *Wert1* <, *SysVar2* = *Wert2*, …, >> <C_SPL><#BASE|#TOOL>

SCIRC_REL:

SCIRC_REL *Hilfspunkt*, *Zielpunkt* <, CA *Kreiswinkel*> <WITH *SysVar1* = *Wert1* <,*SysVar2* = *Wert2* , … >> <C_SPL><#BASE|#TOOL>

SPL_REL:

SPL_REL *Zielpunkt* < WITH *SysVar1* = *Wert1* <,*SysVar2* = *Wert2* , …>><#BASE|#TOOL>

Erläuterung der Syntax

Element	Beschreibung
Hilfspunkt *Zielpunkt*	Typ: POS, E6POS, FRAME Der Punkt muss in kartesischen Koordinaten angegeben werden. Die Steuerung interpretiert die Koordinaten als relativ zum Zielpunkt der Vorgängerbewegung. Wenn nicht alle Komponenten des Punkts angegeben werden, dann setzt die Steuerung die fehlenden Komponenten auf den Wert 0. Das heißt, die absoluten Werte dieser Komponenten bleiben unverändert. Angaben zu Status und Turn, falls vorhanden, werden von der Steuerung ignoriert. (Dies steht im Gegensatz zu SPTP_REL, wo sie berücksichtigt werden!) Beim Hilfspunkt werden außerdem die Orientierungswinkel ignoriert. Der Hilfspunkt kann nicht überschliffen werden. Er wird immer genau angefahren.
Kreiswinkel	Einheit: Grad; ohne Begrenzung (>>> 9.9 "Kreiswinkel" Seite 311)
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)
Wert	Wertzuweisung an die Systemvariable Bei Segmenten: Die Zuweisung gilt nur für dieses Segment. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. Nur möglich für Einzelbewegungen, nicht für Segmente. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.
`#BASE`, `#TOOL`	■ #BASE (Default): Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zur physikalischen Basis gehört. ■ #TOOL: Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zum physikalischen Werkzeug gehört. $IPO_MODE hat keinen Einfluss auf die Bedeutung von #BASE und #TOOL.

Beispiel

```
DECL E6POS P1 = {X 1500, Y -200, Z 2000, A 0, B 0, C 0, S 6, T27}

SPTP HOME
SLIN P1

SLIN_REL{X 0, Y 500, Z 0, A 0, B 0, C 0} WITH $BASE=$NULLFRAME #BASE
SLIN_REL{X 400} WITH $TOOL=$NULLFRAME C_SPL #TOOL
SLIN_REL{A 20}
SPTP_REL{A3 90} C_SPL
SPTP_REL Z 50, B -30} WITH $VEL.AXIS[4]=90 C_SPL #TOOL
SPTP_REL{A1 100}

SPLINE
```

```
   SPL P1
   SPL_REL{Z -300, B50} #TOOL
ENDSPLINE
PTPSPLINE
   SPTP P1
   SPTP_REL{A1 -100, A5 -70}
ENDSPLINE
```

11.6.5 SPTP

Beschreibung

SPTP kann als Segment in einem PTP-Spline-Block oder als Einzelbewegung programmiert werden.

Es ist möglich, eine SPTP-Einzelbewegung in einen PTP-Spline-Block zu kopieren, aber nur, wenn sie keine Zuweisung an Systemvariablen enthält, die dort verboten sind.

Syntax

`SPTP` *Zielpunkt* <`WITH` *SysVar1* = *Wert1* <, *SysVar2* = *Wert2* , …>> <`C_SPL`>

Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Typ: AXIS, E6AXIS, POS, E6POS, FRAME Die kartesischen Koordinaten beziehen sich auf das BASE-Koordinatensystem. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann übernimmt die Steuerung für die fehlenden Komponenten die Werte der vorhergehenden Position. Wenn diese vorhergehende Position der Zielpunkt eines Kreises mit Kreiswinkel ist, werden die Werte des tatsächlich erreichten Zielpunkts übernommen, nicht des programmierten Zielpunkts. Wenn der Robotersteuerung keine vorhergehende Position bekannt ist, werden die fehlenden Komponenten aus der aktuellen Roboterposition übernommen.
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)
Wert	Wertzuweisung an die Systemvariable Bei SPTP-Segmenten: Die Zuweisung gilt nur für dieses Segment. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. Nur möglich für Einzelbewegungen, nicht für Segmente. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.

Bis zur System Software 8.2 lautete die Kennung für das Überschleifen beim Spline "C_DIS". Wenn in höheren Versionen von 8.x Programme verwendet werden, die auf 8.2 oder älteren Ständen basieren und C_DIS enthalten, braucht dieses nicht in C_SPL geändert werden, sondern kann belassen werden.

11.6.6 SPTP_REL

Beschreibung

SPTP_REL kann als Segment in einem PTP-Spline-Block oder als Einzelbewegung programmiert werden.

Es ist möglich, eine SPTP_REL-Einzelbewegung in einen PTP-Spline-Block zu kopieren, aber nur, wenn sie keine Zuweisung an Systemvariablen enthält, die dort verboten sind.

Zum Verhalten der Robotersteuerung bei endlos drehenden rotatorischen Achsen: (>>> 11.5.6 "REL-Bewegungen bei endlos drehenden rotatorischen Achsen" Seite 388)

Syntax

`SPTP_REL` *Zielpunkt* <`WITH` *SysVar1* = *Wert1* <, *SysVar2* = *Wert2* , …>> <`C_SPL`><`#BASE`|`#TOOL`>

Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Typ: AXIS, E6AXIS, POS, E6POS, FRAME Der Zielpunkt kann kartesisch oder achsspezifisch angegeben werden. Die Steuerung interpretiert die Koordinaten als relativ zum Zielpunkt des Vorgängersatzes. Wenn nicht alle Komponenten des Zielpunkts angegeben werden, dann setzt die Steuerung die fehlenden Komponenten auf den Wert 0. Das heißt, die absoluten Werte dieser Komponenten bleiben unverändert. Angaben zu Status und Turn, falls vorhanden, werden von der Steuerung berücksichtigt. (Dies steht im Gegensatz zu SLIN_REL, SCIRC_REL und SPL_REL, wo sie ignoriert werden!)
SysVar	(>>> 11.6.7 "Systemvariablen für WITH" Seite 397)
Wert	Wertzuweisung an die Systemvariable Bei SPTP-Segmenten: Die Zuweisung gilt nur für dieses Segment. Die Systemvariablen können auch per Funktionsaufruf belegt werden. Für diese Funktionen gelten die gleichen Einschränkungen wie für die Funktionen im Trigger. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)
C_SPL	■ Mit C_SPL: Der Zielpunkt wird überschliffen. $APO definiert, wann das Überschleifen frühestens beginnt. Nur möglich für Einzelbewegungen, nicht für Segmente. ■ Ohne C_SPL: Der Zielpunkt wird genau angefahren.
`#BASE`, `#TOOL`	Nur zulässig, wenn der Zielpunkt kartesisch angegeben wurde. ■ #BASE (Default): Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zur physikalischen Basis gehört. ■ #TOOL: Die Koordinaten dieses Zielpunkts beziehen sich auf das Koordinatensystem, das zum physikalischen Werkzeug gehört. $IPO_MODE hat keinen Einfluss auf die Bedeutung von #BASE und #TOOL.

Beispiel

(>>> "Beispiel" Seite 394)

11.6.7 Systemvariablen für WITH

Spline-Block, Spline-Einzelbewegung

Für Spline-Blöcke und Spline-Einzelbewegungen sind folgende Systemvariablen mit der WITH-Zeile beschreibbar:

$ACC
$ACC_AXIS
$ACC_EXTAX
$APO
$BASE
$CIRC_TYPE
$ECO_LEVEL
$GEAR_JERK
$IPO_MODE
$JERK
$LOAD
$ORI_TYPE
$ROTSYS
$SPL_ORI_JOINT_AUTO
$SYNC_ID
$SYNC_LIST
$TOOL
$VEL
$VEL_AXIS
$VEL_EXTAX

Zusätzlich für SCIRC und SLIN: $CIRC_MODE

Spline-Segment

Für Spline-Segmente sind folgende Systemvariablen mit der WITH-Zeile beschreibbar:

$ACC
$ACC_AXIS
$ACC_EXTAX
$CIRC_TYPE
$EX_AX_IGNORE
$GEAR_JERK
$JERK
$ORI_TYPE
$ROTSYS
$SYNC_ID
$VEL
$VEL_AXIS
$VEL_EXTAX

Zusätzlich für SCIRC und SLIN: $CIRC_MODE

11.6.8 TIME_BLOCK

Beschreibung

TIME_BLOCK kann in CP- und in PTP-Spline-Blöcken verwendet werden.

TIME_BLOCK ermöglicht es, den Spline-Block oder einen Teil davon in einer definierten Zeit abzufahren. Zusätzlich ist es möglich, Bereichen innerhalb von TIME_BLOCK einen Zeitanteil zuzuweisen.

Im Spline-Block können Punkte geändert, hinzugefügt oder entfernt werden, ohne die Zeitvorgaben zu ändern. Dies ermöglicht es dem Benutzer, die kartesische Bahn zu korrigieren und dabei die bestehenden Zeitvorgaben beizubehalten.

Ein Spline-Block darf 1 Zeitblock enthalten, d. h. 1 Anweisung TIME_BLOCK START … TIME_BLOCK END. Dazwischen können beliebig viele TIME_BLOCK PART verwendet werden. Der Zeitblock darf nur in Spline-Blöcken verwendet werden.

Ein CP-Spline-Block kann entweder 1 Zeitblock oder 1 Konstantfahrbereich enthalten, aber nicht beides.

Syntax

```
SPLINE
<Spline-Segmente …>
…
TIME_BLOCK START
Spline-Segment
<Spline-Segmente …>
...
< <TIME_BLOCK PART = Anteil_1>
…
Spline-Segment
<Spline-Segmente …>
...
TIME_BLOCK PART = Anteil_N >
TIME_BLOCK END = Gesamtzeit
<Spline-Segmente …>
…
ENDSPLINE
```

Erläuterung der Syntax

Vor TIME_BLOCK START und nach TIME_BLOCK END müssen nicht zwingend Spline-Segmente stehen. Es wird jedoch empfohlen, folgendermaßen zu programmieren:

- Zwischen SPLINE und TIME_BLOCK START steht mindestens 1 Spline-Segment.
- Zwischen TIME_BLOCK END und ENDSPLINE steht mindestens 1 Spline-Segment.

Vorteile:

- Die programmierte Gesamtzeit wird auch beim Überschleifen exakt eingehalten.
- Segmente vor TIME_BLOCK START ermöglichen es, auf die erforderliche Geschwindigkeit zu beschleunigen.

Element	Beschreibung
Anteil	Typ: INT oder REAL; Konstante, Variable oder Funktion Wunsch-Anteil von *Gesamtzeit* für folgende Strecke: ■ Vom Punkt vor `TIME_BLOCK PART=`*vorheriger_Anteil* bis zum Punkt vor `TIME_BLOCK PART=`*Anteil* ■ Wenn *vorheriger_Anteil* nicht existiert: Vom Punkt vor `TIME_BLOCK START` bis zum Punkt vor `TIME_BLOCK PART=`*Anteil* Mit "Wunsch-Anteil" ist gemeint: Die Anteile werden von der Robotersteuerung so genau wie möglich eingehalten. In der Regel werden sie aber nicht exakt eingehalten. Der Benutzer kann die Anteile so vergeben, dass sie zusammen 100 ergeben. Dann kann er die Anteile als Prozent von *Gesamtzeit* betrachten. Die Anteile müssen jedoch nicht 100 ergeben, sondern können eine beliebige Summe ergeben! Die Robotersteuerung setzt die Summe der Anteile immer gleich mit *Gesamtzeit*. Hierdurch können die Anteile sehr flexibel verwendet und auch geändert werden. Wenn Anteile vergeben werden, muss immer ein `TIME_BLOCK PART` direkt vor `TIME_BLOCK END` stehen. Dazwischen dürfen keine Segmente stehen.
Gesamtzeit	Typ: INT oder REAL; Konstante, Variable oder Funktion; Einheit: s Zeit, in der folgende Strecke abgefahren wird: ■ Vom Punkt vor `TIME_BLOCK START` bis zum Punkt vor `TIME_BLOCK END` Der Wert muss größer 0 sein. Die Gesamtzeit wird exakt eingehalten. Wenn sie nicht einhaltbar ist, z. B. weil eine zu kurze Zeit programmiert wurde, fährt der Roboter die schnellstmögliche Zeit. In T1 und T2 wird außerdem eine Meldung angezeigt.

Wenn der Wert für *Anteil* oder *Gesamtzeit* über eine Funktion zugewiesen wird, gelten die gleichen Einschränkungen wie für die Funktionen im Trigger.
(>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)

Beispiel

```
SPLINE
  SLIN P1
  SPL P2
  TIME_BLOCK START
    SLIN P3
  TIME_BLOCK PART = 12.7
    SPL P4
    SPL P5
    SPL P6
  TIME_BLOCK PART = 56.4
    SCIRC P7, P8
    SPL P9
  TIME_BLOCK PART = 27.8
  TIME_BLOCK END = 3.9
  SLIN P10
ENDSPLINE
```

Die Punkte P2 bis P9 werden exakt in der programmierten Zeit von 3,9 s durchlaufen. Die Robotersteuerung setzt die Gesamtzeit von 3,9 s gleich mit der Summe aller Anteile, also 96,9 Anteile.

Strecke	Zeit, die die Robotersteuerung der Strecke zuweist
P2 … P3	12,7 Anteile von 3,9 s = 0,51 s
P3 … P6	56,4 Anteile von 3,9 s = 2,27 s
P6 … P9	27,8 Anteile von 3,9 s = 1,12 s

Satzanwahl

Je nachdem, auf welche Zeile eine Satzanwahl ausgeführt wird, plant die Robotersteuerung den Zeitblock oder nicht.

Satzanwahl auf Zeile …	Zeitblock wird geplant?
im Spline-Block, vor TIME_BLOCK START	Ja
TIME_BLOCK START	Nein Der Spline-Block wird ausgeführt, wie wenn keine TIME_BLOCK-Anweisungen vorhanden wären.
im Zeitblock	
TIME_BLOCK END	
im Spline-Block, nach TIME_BLOCK END	

Wenn die Robotersteuerung den Zeitblock nicht plant, gibt sie folgende Meldung aus: *Zeitblock aufgrund der SAK-Fahrt ignoriert.*

$PATHTIME

Die Daten des zeitbasierten Splines können über die Systemvariable $PATHTIME gelesen werden. $PATHTIME wird mit den Daten gefüllt, sobald die Robotersteuerung die Planung des Spline-Blocks abgeschlossen hat. Die Daten bleiben erhalten, bis der nächste Spline-Block geplant worden ist.

$PATHTIME ist eine Struktur und besteht aus folgenden Komponenten:

Komponente	Beschreibung
REAL $PATHTIME.TOTAL	Tatsächlich benötigte Zeit für den gesamten Spline-Block (s)
REAL $PATHTIME.SCHEDULED	Geplante Gesamtzeit für den Zeitblock (s)
REAL $PATHTIME.PROGRAMMED	Programmierte Gesamtzeit für den Zeitblock (s)
INT $PATHTIME.N_SECTIONS	Anzahl N der Zeilen TIME_BLOCK_PART
REAL $PATHTIME.MAX_DEV	Größte Abweichung unter allen TIME_BLOCK_PART zwischen programmierter Zeit und geplanter Zeit (%)
INT $PATHTIME.MAX_DEV_SECTION	Nummer des TIME_BLOCK_PART mit der größten Abweichung zwischen programmierter Zeit und geplanter Zeit

11.6.9 CONST_VEL

Beschreibung

Mit CONST_VEL definiert man Konstantfahrbereiche. CONST_VEL kann nur in CP-Spline-Blöcken verwendet werden.

Zwischen CONST_VEL END und ENDSPLINE muss mindestens 1 Spline-Segment stehen.

Ein CP-Spline-Block kann entweder 1 CONST_VEL oder 1 TIME_BLOCK enthalten, aber nicht beides.

In dieser Dokumentation sind weiterführende Informationen zu Konstantfahrbereichen zu finden.
(>>> 10.3.6 "Konstantfahrbereich im CP-Spline-Block" Seite 347)

Syntax

```
CONST_VEL START = Offset <ONSTART>
<Spline-Segmente ...>
...
CONST_VEL END = Offset <ONSTART>
```

Erläuterung der Syntax

Element	Beschreibung
ONSTART	Bezugspunkt der Anweisung ■ Mit ONSTART: Startpunkt ■ Ohne ONSTART: Zielpunkt Wenn der Start- oder Zielpunkt überschliffen ist, ergibt sich der Bezugspunkt auf die gleiche Weise wie beim homogenen Überschleifen beim PATH-Trigger. (>>> 11.11.2.2 "Bezugspunkt beim homogenen Überschleifen" Seite 444)
Offset	Typ: INT oder REAL; Konstante, Variable oder Funktion; Einheit: mm Über CONST_VEL START = *Offset* kann man den Bereichsanfang örtlich verschieben. Über CONST_VEL END = *Offset* kann man das Bereichsende örtlich verschieben. ■ Positiver Wert: Verschiebung in Richtung Bewegungsende ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang Auf welchen Punkt sich die Verschiebung bezieht, ist abhängig davon, ob ONSTART gesetzt ist oder nicht. Wenn keine Verschiebung gewünscht ist, muss *Offset*=0 programmiert werden. (>>> 10.3.6.2 "Maximale Grenzen" Seite 349)

Der Wert für *Offset* kann über eine Funktion zugewiesen werden. Es gelten die gleichen Einschränkungen wie für die Funktionen im Trigger.
(>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)

Beispiel

Hier erstreckt sich der Konstantfahrbereich über mehrere Segmente mit verschiedenen programmierten Geschwindigkeiten. Die niedrigste der Geschwindigkeiten, also 0,2 m/s, gilt in diesem Fall für den gesamten Bereich.

```
1  PTP P0
2  SPLINE WITH $VEL.CP = 2.5
3   SLIN P1
4   CONST_VEL START = +100
5   SPL P2 WITH $VEL.CP = 0.5
6   SLIN P3 WITH $VEL.CP = 0.2
7   SPL P4 WITH $VEL.CP = 0.4
8   CONST_VEL END = -50
```

```
9   SCIRC P5, P6
10  SLIN P7
11 ENDSPLINE
```

11.6.9.1 Systemvariablen zu CONST_VEL

$STOP_CONST_VEL_RED

Wenn in einem Konstantfahrbereich die maximal mögliche konstante Geschwindigkeit unter der programmierten Geschwindigkeit liegt, gibt die Robotersteuerung eine der folgenden Meldungen aus:

- *Im CONST_VEL-Bereich statt $VEL.CP={Soll $VEL.CP}m/s nur {erreichte Geschwindigkeit}m/s wg. Zeile {Zeile des limitierenden Segments} erreicht.*
- *In CONST_VEL START statt $VEL.CP={Soll $VEL.CP} m/s nur {erreichte Geschwindigkeit} m/s wg. Zeile {Zeile des limitierenden Segments} erreicht.*
- *In CONST_VEL END statt $VEL.CP={Soll $VEL.CP} m/s nur {erreichte Geschwindigkeit} m/s wg. Zeile {Zeile des limitierenden Segments} erreicht.*

Für diese Meldungen ist für T1/T2 konfigurierbar, ob sie Hinweismeldungen oder Quittiermeldungen sein sollen. Dies geschieht über die Systemvariable $STOP_CONST_VEL_RED.

Wert	Beschreibung
FALSE (Default)	Hinweismeldung
TRUE	■ Betriebsart T1 oder T2: Quittiermeldung Der Roboter stoppt. Im Programm zeigt der Satzzeiger das auslösende Spline-Segment an. Das Programm kann erst fortgesetzt werden, wenn der Benutzer die Meldung quittiert hat. ■ Betriebsart AUT oder AUT EXT: Hinweismeldung

$STOP_CONST_VEL_RED wird beim Kaltstart und bei Programmanwahl mit FALSE initialisiert, jedoch nicht beim Zurücksetzen eines Programms.

$STOP_CONST_VEL_RED kann über die Variablenkorrektur oder über KRL geändert werden.

$CONST_VEL

$CONST_VEL gibt für den gerade in Planung befindlichen CP-Spline-Block an, welche Geschwindigkeit (mm/s) im Konstantfahrbereich gefahren werden wird. Der Wert bleibt erhalten, bis ein weiterer Spline-Block mit Konstantfahrbereich geplant wird.

$CONST_VEL ist schreibgeschützt. $CONST_VEL ist ungültig, wenn sich eine der folgenden Bewegungen in Planung befindet:

- CP-Spline-Block ohne Konstantfahrbereich
- PTP-Spline-Block
- Spline-Einzelbewegung
- PTP, LIN, CIRC

$CONST_VEL_C

$CONST_VEL_C entspricht $CONST_VEL; mit dem Unterschied, dass sich $CONST_VEL_C auf den CP-Spline-Block bezieht, der gerade abgearbeitet wird.

Mögliche praktische Nutzung:

Der Wert von $CONST_VEL kann pro Konstantfahrbereich in eine benutzerspezifische, lokale Variable geschrieben werden. Dies liefert einen Überblick über die erreichbaren konstanten Geschwindigkeiten. Dementsprechend kann dann die Applikation programmiert werden, z. B. der Zeitpunkt, wann eine Düse geöffnet werden muss o. ä.

11.6.10 STOP WHEN PATH

Beschreibung

Mit STOP WHEN PATH kann der Benutzer einen Bedingten Stopp programmieren.

In dieser Dokumentation sind weiterführende Informationen zum Bedingten Stopp zu finden.
(>>> 10.3.5 "Bedingter Stopp" Seite 344)

Positionen

Der Bedingte Stopp kann an folgenden Positionen verwendet werden:

- Im Spline-Einzelsatz
- Im Spline-Block (CP und PTP)

  Zwischen STOP WHEN PATH und ENDSPLINE muss mindestens 1 Segment stehen.
- Vor einem Spline-Block (CP und PTP)

  STOP WHEN PATH bezieht sich in diesem Fall auf den Spline-Block. Zwischen STOP WHEN PATH und dem Spline-Block dürfen Anweisungen stehen, jedoch keine Bewegungsanweisungen.

Syntax

`STOP WHEN PATH =` *Offset* `<ONSTART> IF` *Bedingung*

Erläuterung der Syntax

Element	Beschreibung
`ONSTART`	Bezugspunkt der Anweisung ■ Ohne ONSTART: Zielpunkt ■ Mit ONSTART: Startpunkt Wenn der Bezugspunkt überschliffen ist, gelten die gleichen Regeln wie beim PATH-Trigger. (>>> 11.11.2.1 "Bezugspunkt beim Überschleifen – Übersicht" Seite 444)

Element	Beschreibung
Offset	Typ: INT oder REAL; Konstante, Variable oder Funktion; Einheit: mm Über *Offset* kann man den Stopp-Punkt örtlich verschieben. ■ Positiver Wert: Verschiebung in Richtung Bewegungsende ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang Auf welchen Punkt sich die Verschiebung bezieht, ist abhängig davon, ob ONSTART gesetzt ist oder nicht. Wenn keine Verschiebung gewünscht ist, muss *Offset*=0 programmiert werden. Der Stopp-Punkt kann nicht beliebig weit verschoben werden. Es gelten die gleichen Grenzen wie beim PATH-Trigger. (>>> "Max. Verschiebung" Seite 441)
Bedingung	Typ: BOOL Stopp-Bedingung. Zulässig sind: ■ eine globale boolesche Variable ■ ein Signalname ■ ein Vergleich ■ eine einfache logische Verknüpfung: NOT, OR, AND oder EXOR

Der Wert für *Offset* kann über eine Funktion zugewiesen werden. Es gelten die gleichen Einschränkungen wie für die Funktionen im Trigger.
(>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)

11.6.11 $EX_AX_IGNORE

Beschreibung

$EX_AX_IGNORE kann nur in der WITH-Zeile von Spline-Segmenten verwendet werden.

Jedes Bit von $EX_AX_IGNORE entspricht einer Zusatzachs-Nummer. Wenn ein bestimmtes Bit auf 1 gesetzt ist, ignoriert die Robotersteuerung am Zielpunkt des Segments die geteachte bzw. programmierte Position dieser Zusatzachse. Stattdessen errechnet die Robotersteuerung auf Grundlage der umgebenden Zusatzachs-Positionen die optimale Position für diesen Punkt.

Empfehlung: Immer wenn an einem Punkt keine bestimmte Position der Zusatzachse erforderlich ist, $EX_AX_IGNORE verwenden und das Bit dieser Zusatzachse auf 1 setzen. Dies verringert die Taktzeit.

In den Programmablaufarten MSTEP und ISTEP stoppt der Roboter an den von der Robotersteuerung errechneten Positionen.

Bei einer Satzanwahl auf einen Punkt mit "$EX_AX_IGNORE = Bit n = 1" nimmt der Roboter die von der Robotersteuerung errechnete Position an.

Für folgende Segmente ist "$EX_AX_IGNORE = Bit n = 1" nicht erlaubt:

- Für das erste Segment in einem Spline-Block (nur bis einschließlich KUKA System Software 8.2)
- Für das letzte Segment in einem Spline-Block

- Wenn mehrere Segmente mit kartesisch gleichen Zielpunkten aufeinanderfolgen, ist "$EX_AX_IGNORE = Bit n = 1" für das erste und das letzte Segment nicht erlaubt. (nur bis einschließlich KUKA System Software 8.2)

Gültig ab KUKA System Software 8.3: Wenn $EX_AX_IGNORE zu einem SPTP-Segment programmiert wird und die betroffene Zusatzachse mathematisch gekoppelt ist, verwirft die Robotersteuerung $EX_AX_IGNORE. D. h., die geteachte bzw. programmierte Position dieser Zusatzachse wird berücksichtigt. In T1/T2 gibt die Robotersteuerung dazu folgende Meldung aus: *Verwerfe $EX_AX_IGNORE in Zeile {Satznummer}, weil {Nummer der Zusatzachse} mathematisch gekoppelt ist.*

Syntax

`$EX_AX_IGNORE=`*Bitfeld*

Erläuterung der Syntax

Element	Beschreibung
Bitfeld	■ **Bit n = 1**: Geteachte/programmierte Position der Zusatzachse wird ignoriert. ■ **Bit n = 0**: Geteachte/programmierte Position der Zusatzachse wird berücksichtigt.

Bit n	5	4	3	2	1	0
Achse	E6	E5	E4	E3	E2	E1

Beispiel

```
SPLINE
  SPL P1
  SPL P2
  SLIN P3 WITH $EX_AX_IGNORE = 'B000001'
  SPL P4
ENDSPLINE
```

Für P3 ignoriert die Robotersteuerung die geteachte Position der Zusatzachse E1.

11.7 Programmablaufkontrolle

11.7.1 CONTINUE

Beschreibung

Mit CONTINUE kann man einen Vorlaufstopp verhindern, der in der folgenden Programmzeile auftreten würde.

CONTINUE bezieht sich immer auf die folgende Zeile, auch wenn es sich dabei um eine Leerzeile handelt! Ausnahme: Wenn in der folgenden Zeile ON_ERROR_PROCEED steht, bezieht sich CONTINUE erst auf die Zeile danach.

Syntax

`CONTINUE`

Beispiele

Verhindern der beiden Vorlaufstopps:

```
CONTINUE
$OUT[1]=TRUE
CONTINUE
$OUT[2]=FALSE
```

Die Ausgänge werden in diesem Fall im Vorlauf gesetzt. Wann genau sie gesetzt werden, ist nicht vorhersehbar.

ON_ERROR_PROCEED mit CONTINUE:

```
ON_ERROR_PROCEED
```

```
CONTINUE
$OUT[1]=TRUE
```

```
CONTINUE
ON_ERROR_PROCEED
$OUT[1]=TRUE
```

Diese Anweisungsfolgen sind von der Wirkung her identisch. In beiden Beispielen wirken ON_ERROR_PROCEED und CONTINUE auf $OUT[1]=TRUE.

11.7.2 EXIT

Beschreibung Aussprung aus einer Schleife. Das Programm wird dann nach der Schleife fortgesetzt. EXIT darf in jeder Schleife verwendet werden.

Syntax `EXIT`

Beispiel Die Schleife wird verlassen, wenn `$IN[1]` TRUE wird. Das Programm wird dann nach ENDLOOP fortgesetzt.

```
DEF EXIT_PROG()
PTP HOME
LOOP
   PTP POS_1
   PTP POS_2
   IF $IN[1] == TRUE THEN
      EXIT
   ENDIF
   CIRC HELP_1, POS_3
   PTP POS_4
ENDLOOP
PTP HOME
END
```

11.7.3 FOR ... TO ... ENDFOR

Beschreibung Ein Anweisungsblock wird so oft ausgeführt, bis ein Zähler einen definierten Wert über- oder unterschreitet.

Nach dem letzten Durchlauf des Anweisungsblocks wird das Programm mit der ersten Anweisung nach ENDFOR fortgesetzt. Die Schleife kann mit EXIT vorzeitig verlassen werden.

Schleifen können verschachtelt werden. Bei verschachtelten Schleifen wird erst die äußere Schleife komplett durchlaufen. Danach wird die innere Schleife komplett durchlaufen.

Syntax

`FOR` *Zaehler* `=` *Startwert* `TO` *Endwert* `<STEP` *Schrittweite*`>`

<Anweisungen>

`ENDFOR`

Erläuterung der Syntax

Element	Beschreibung
Zaehler	Typ: INT Variable, die die Durchläufe zählt. Wird mit *Startwert* vorbesetzt. Die Variable muss vorher deklariert werden. Der Wert von *Zaehler* kann in Anweisungen innerhalb und außerhalb der Schleife benutzt werden. Nach dem Verlassen der Schleife hat *Zaehler* den zuletzt angenommenen Wert.
Startwert; Endwert	Typ: INT *Zaehler* muss mit *Startwert* vorbesetzt werden. Nach jedem Schleifendurchlauf verändert sich *Zaehler* automatisch um die Schrittweite. Wenn *Endwert* über- oder unterschritten ist, ist die Schleife beendet.
Schrittweite	Typ: INT Wert, um den *Zaehler* bei jedem Schleifendurchlauf verändert wird. Der Wert darf negativ sein. Defaultwert: 1. ■ Wert positiv: Die Schleife ist beendet, wenn *Zaehler* größer als *Endwert* ist. ■ Wert negativ: Die Schleife ist beendet, wenn *Zaehler* kleiner als *Endwert* ist. Der Wert darf weder Null noch eine Variable sein.

Beispiel

Die Variable `B` wird in 5 Durchläufen um je 1 erhöht.

```
INT A
...
FOR A=1 TO 10 STEP 2
   B=B+1
ENDFOR
```

11.7.4 GOTO

Beschreibung

Unbedingter Sprung an spezifizierte Stelle im Programm. Das Programm wird an dieser Stelle fortgesetzt.

Das Sprungziel muss sich im selben Unterprogramm oder in derselben Funktion wie die GOTO-Anweisung befinden.

Nicht möglich sind folgende Sprünge:

- Von außen in eine IF-Anweisung springen.
- Von außen in eine Schleife springen.
- Von einer CASE-Anweisung in eine andere CASE-Anweisung springen.

GOTO macht Programme unübersichtlich. Besser: Stattdessen mit IF, mit SWITCH oder mit einer Schleife arbeiten.

Syntax

`GOTO` *Marke*

`...`

Marke`:`

Erläuterung der Syntax

Element	Beschreibung
Marke	Stelle, an die gesprungen wird. An der Zielstelle muss *Marke* am Ende einen Doppelpunkt haben.

Beispiel 1

Unbedingter Sprung zur Programmstelle GLUESTOP.

```
GOTO GLUESTOP
...
GLUESTOP:
```

Beispiel 2

Unbedingter Sprung zur Programmstelle ENDE aus einer IF-Anweisung heraus.

```
IF X>100 THEN
   GOTO ENDE
ELSE
   X=X+1
ENDIF
A=A*X
...
ENDE:
END
```

11.7.5 HALT

Beschreibung

Hält das Programm an. Die zuletzt durchlaufene Bewegungsanweisung wird jedoch noch vollständig ausgeführt.

Das Programm kann nur mit der Starttaste fortgesetzt werden. Dann wird die nächste Anweisung nach HALT ausgeführt.

In einem Interrupt-Programm wird das Programm erst nach vollständiger Abarbeitung des Vorlaufs angehalten.

Syntax

HALT

11.7.6 IF ... THEN ... ENDIF

Beschreibung

Bedingte Verzweigung. Abhängig von einer Bedingung wird entweder der erste Anweisungsblock (THEN-Block) oder der zweite Anweisungsblock (ELSE-Block) ausgeführt. Danach wird das Programm nach ENDIF fortgesetzt.

Der ELSE-Block darf fehlen. Bei nicht erfüllter Bedingung wird das Programm dann sofort nach ENDIF fortgesetzt.

Die Anzahl der Anweisungen in den Anweisungsblöcken ist nicht begrenzt. Mehrere IF-Anweisungen können ineinander verschachtelt werden.

Syntax

IF *Bedingung* THEN

Anweisungen

<ELSE

Anweisungen>

ENDIF

Erläuterung der Syntax

Element	Beschreibung
Bedingung	Typ: BOOL Möglich: ■ Variable vom Typ BOOL ■ Funktion vom Typ BOOL ■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ BOOL

Beispiel 1 IF-Anweisung ohne ELSE

```
IF A==17 THEN
   B=1
ENDIF
```

Beispiel 2 IF-Anweisung mit ELSE

```
IF $IN[1]==TRUE THEN
   $OUT[17]=TRUE
ELSE
   $OUT[17]=FALSE
ENDIF
```

11.7.7 LOOP ... ENDLOOP

Beschreibung Schleife, die einen Anweisungsblock endlos wiederholt. Die Schleife kann mit EXIT verlassen werden.

Schleifen können verschachtelt werden. Bei verschachtelten Schleifen wird erst die äußere Schleife komplett durchlaufen. Danach wird die innere Schleife komplett durchlaufen.

Syntax

```
LOOP
Anweisungen
ENDLOOP
```

Beispiel Die Schleife wird solange durchlaufen, bis der Eingang $IN[30] TRUE wird.

```
LOOP
   LIN P_1
   LIN P_2
   IF $IN[30]==TRUE THEN
      EXIT
   ENDIF
ENDLOOP
```

11.7.8 ON_ERROR_PROCEED

Beschreibung Mit ON_ERROR_PROCEED kann man eine Laufzeitfehler-Meldung unterdrücken, die durch die folgende Programmzeile ausgelöst würde. Die Robotersteuerung überspringt die fehlerauslösende Anweisung und füllt die Systemvariable $ERR mit Infos zum Fehler.

(>>> 11.7.8.1 "$ERR" Seite 410)

Meldungen zu internen Fehlern oder Systemfehlern können nicht unterdrückt werden.

ON_ERROR_PROCEED bezieht sich immer auf die folgende Zeile, auch wenn es sich dabei um eine Leerzeile handelt! Ausnahme: Wenn in der folgenden Zeile CONTINUE steht, bezieht sich ON_ERROR_PROCEED erst auf die Zeile danach.

Wenn die Zeile nach ON_ERROR_PROCEED ein Unterprogramm-Aufruf ist, dann bezieht sich die Anweisung auf den Aufruf selber, nicht auf die erste Zeile des Unterprogramms.

$ERR, ERR_RAISE() Wichtige Hilfsmittel beim Arbeiten mit ON_ERROR_PROCEED sind $ERR und ERR_RAISE().

Die Funktion ERR_RAISE() kann eine unterdrückte Laufzeitfehler-Meldung nachträglich ausgeben. Sie kann nur die Systemvariable $ERR verarbeiten, oder eine als OUT-Parameter von $ERR abgeleitete Variable.

Einschränkungen

ON_ERROR_PROCEED wirkt nicht bei Bewegungsanweisungen:

SPLINE/ENDPLINE; PTP_SPLINE/ENDSPLINE; PTP; LIN; CIRC; PTP_REL; LIN_REL; CIRC_REL; ASYPTP; ASYSTOP; ASYCONT; ASYCANCEL; MOVE_EMI

ON_ERROR_PROCEED wirkt nicht bei folgenden Kontrollstrukturen:

FOR/ENDFOR; GOTO; IF/ELSE/ENDIF; LOOP/ENDLOOP; REPEAT/UNTIL; SKIP/ENDSKIP; SWITCH/CASE/DEFAULT/ENDSWITCH; WHILE/END-WHILE

Syntax

```
ON_ERROR_PROCEED
```

Beispiele

(>>> 11.7.8.2 "Beispiele zu $ERR, ON_ERROR_PROCEED und ERR_RAISE()" Seite 412)

ON_ERROR_PROCEED mit CONTINUE:

```
ON_ERROR_PROCEED
CONTINUE
$OUT[1]=TRUE
```

```
CONTINUE
ON_ERROR_PROCEED
$OUT[1]=TRUE
```

Diese Anweisungsfolgen sind von der Wirkung her identisch. In beiden Beispielen wirken ON_ERROR_PROCEED und CONTINUE auf $OUT[1]=TRUE.

11.7.8.1 $ERR

Beschreibung

Struktur mit Informationen zum aktuellen Programm

Über die Variable kann das aktuell ausgeführte Programm vorlaufbezogen ausgewertet werden. Beispielsweise kann man mit Hilfe der Variablen Fehler im Programm auswerten, um mit einer geeigneten Fehlerstrategie auf diese Fehler zu reagieren.

Die Variable ist schreibgeschützt und kann nur gelesen werden.

$ERR existiert getrennt für den Roboter- und Submit-Interpreter. Jeder Interpreter kann nur auf seine eigene Variable zugreifen. Für den Kommando-Interpreter existiert $ERR nicht.

Jede Unterprogramm-Ebene hat eine eigene Repräsentation von $ERR. Dadurch überschreiben sich die Informationen aus den verschiedenen Ebenen nicht, und zu einem Zeitpunkt können Informationen aus verschiedenen Ebenen ausgelesen werden.

ON_ERROR_PROCEED löscht implizit die Informationen von $ERR im aktuellen Interpreter und auf der aktuellen Ebene.

(>>> 11.7.8.2 "Beispiele zu $ERR, ON_ERROR_PROCEED und ERR_RAISE()" Seite 412)

Syntax

```
$ERR=Information
```

Erläuterung der Syntax

Element	Beschreibung
Information	Typ: Error_T Liste mit den Informationen zum Programm, das aktuell ausgeführt wird

Error_T

```
STRUC Error_T INT number, PROG_INT_E interpreter,
INT_TYP_E int_type, INT int_prio, line_nr, CHAR mo-
dule[24], up_name[24], TRIGGER_UP_TYPE trigger_type
```

Element	Beschreibung
number	Nur im Falle eines Laufzeitfehlers: Meldungsnummer Wenn kein Fehler aufgetreten ist, wird der Wert Null angezeigt.
interpreter	Aktueller Interpreter ■ #R_INT: Roboter-Interpreter ■ #S_INT: Submit-Interpreter
int_type	Aktueller Programmtyp und Interrupt-Status ■ #I_NORMAL: Das Programm ist kein Interrupt-Programm. ■ #I_INTERRUPT: Das Programm ist ein Interrupt-Programm. ■ #I_STOP_INTERRUPT: Interrupt durch $STOPMESS (Fehlerstopp)
int_prio	Priorität des Interrupts
line_nr	Nur im Falle eines Laufzeitfehlers: Nummer der Zeile, die den Fehler ausgelöst hat **Hinweis**: Die Nummer entspricht in der Regel nicht der Zeilennummer im smartHMI-Programmeditor! Um die Zählung nachzuvollziehen, das Programm mit einem einfachen Editor öffnen und Zeilen, die mit "&" beginnen, nicht mitzählen. Wenn kein Fehler aufgetreten ist, wird der Wert Null angezeigt.
module[]	Name des aktuellen Programms
up_name[]	Name des aktuellen Unterprogramms
trigger_type	Kontext, in dem der zu einem Unterprogramm gehörige Trigger ausgelöst wurde ■ #TRG_NONE: Das Unterprogramm ist kein Trigger-Unterprogramm. ■ #TRG_REGULAR: Das Trigger-Unterprogramm wurde beim Vorwärtsfahren geschaltet. ■ #TRG_BACKWARD: Das Trigger-Unterprogramm wurde beim Rückwärtsfahren geschaltet. ■ #TRG_RESTART: Das Trigger-Unterprogramm wurde beim Umschalten auf das wieder Vorwärtsfahren geschaltet. ■ #TRG_REPLAY: Das Trigger-Unterprogramm wurde nach dem Rückwärtsfahren wiederholt geschaltet. **Hinweis**: Diese Komponente steht ab KUKA System Software 8.3 zur Verfügung.

11.7.8.2 Beispiele zu $ERR, ON_ERROR_PROCEED und ERR_RAISE()

Beispiel 1

Wenn man nicht jede mögliche Laufzeitfehler-Meldung unterdrücken möchte, sondern nur bestimmte, kann man diese Unterscheidung über SWITCH … ENDSWITCH vornehmen. In diesem Beispiel wird nur die Meldung 1422 unterdrückt. Eventuelle andere Laufzeitfehler-Meldungen würden angezeigt.

```
 1  DEF myProg ()
 2  DECL E6POS myPos
 3  INI
 4  ON_ERROR_PROCEED
 5  myPos = $POS_INT
 6  SWITCH ($ERR.NUMBER)
 7    CASE 0
 8    CASE 1422
 9       ;bei Bedarf Fehlerstrategie programmieren
          ...
10    DEFAULT
11       ERR_RAISE ($ERR)
12  ENDSWITCH
    ...
13  END
```

Zeile	Beschreibung
4, 5	Zeile 5 löst die Meldung 1422 *{$Variable} Wert ungültig* aus. (Außer das Programm wird von einem Interrupt aufgerufen.) ON_ERROR_PROCEED in der Zeile vorher unterdrückt die Fehlermeldung.
6 … 12	Fallunterscheidung in Abhängigkeit von $ERR.NUMBER
7	Falls in Zeile 5 kein Fehler aufgetreten ist, ist $ERR.NUMBER==0. In diesem Fall ist keine Aktion notwendig.
8, 9	Falls die Meldung 1422 ausgelöst wurde, ist $ERR.NUMBER==1422. Bei Bedarf kann eine Fehlerstrategie programmiert werden.
10, 11	Falls eine andere Meldung als 1422 ausgelöst wurde, wird diese Meldung nun (nachträglich) über ERR_RAISE ausgegeben.

Beispiel 2

Dieses Beispiel verdeutlicht, dass jede Programmebene ihre eigene Repräsentation von $ERR besitzt.

```
 1  DEF myMainProg ()
 2  INT myVar, myVar2
 3  INI
 4  ON_ERROR_PROCEED
 5  mySubProg (myVar)
 6  HALT
 7  myVar2 = 7
 8  mySubProg (myVar2)
 9  END
----------------------------------------
10  DEF mySubProg (myTest:IN)
11  INT myTest
12  HALT
13  END
```

Zeile	Beschreibung
4, 5	Zeile 5 löst die Meldung 1422 *{$Variable} Wert ungültig* aus, weil myVar nicht initialisiert ist und deshalb nicht an ein Unterprogramm übergeben werden kann. ON_ERROR_PROCEED in der Zeile vorher unterdrückt die Fehlermeldung.
6	Wenn man hier $ERR über die Variablenkorrektur ausliest, haben folgende Komponten folgende Werte: $ERR.number == 1422 $ERR.line_nr == 15 $ERR.module[] == "MYMAINPROG" $ERR.up_name[] == "MYMAINPROG"
12	Wenn man hier im Unterprogramm $ERR über die Variablenkorrektur ausliest, haben folgende Komponten folgende Werte: $ERR.number == 0 $ERR.line_nr == 0 $ERR.module[] == "MYMAINPROG" $ERR.up_name[] == "MYSUBPROG" Dies macht deutlich: $ERR hat immer die Informationen aus der aktuellen Ebene, d. h. in diesem Fall aus dem Unterprogramm MySubProg. Die Infos aus MyMainProg dagegen sind nicht bekannt.

Beispiel 3

Dieses Beispiel zeigt ebenfalls, dass jede Programmebene ihre eigene Repräsentation von $ERR besitzt. Außerdem zeigt es, wie die $ERR-Informationen in eine andere Ebene übertragen werden können.

```
 1  DEF myMainProg2 ()
 2  INI
 3  ON_ERROR_PROCEED
 4  $OUT[-10] = TRUE
 5  myHandleErr ($ERR, $ERR)
 6  END
---------------------------------------
 7  DEF myHandleErr (inErr:IN, outErr:OUT)
 8  DECL Error_T inErr, outErr
 9  ON_ERROR_PROCEED
10  $OV_PRO=100/0
11  ERR_RAISE($ERR)
12  ERR_RAISE(outErr)
13  ERR_RAISE(inErr)
    ...
14  END
```

Zeile	Beschreibung
3, 4	Zeile 4 löst die Meldung 1444 *Unzulässiger Feldindex* aus. ON_ERROR_PROCEED in der Zeile vorher unterdrückt die Fehlermeldung.
5, 7	Der Inhalt von $ERR wird 2-mal in ein Unterprogramm übergeben, einmal als IN-Parameter und einmal OUT-Parameter.

Zeile	Beschreibung
9, 10	Zeile 10 löst die Meldung 1451 *Division durch 0* aus. ON_ERROR_PROCEED in der Zeile vorher unterdrückt die Fehlermeldung.
11	ERR_RAISE($ERR) gibt die Meldung aus Zeile 10 aus, nicht die aus Zeile 4. $ERR hat immer die Informationen aus der aktuellen Ebene, d. h. in diesem Fall aus dem Unterprogramm myHandleErr.
12	ERR_RAISE(outErr) gibt die Meldung aus Zeile 4 im Hauptprogramm aus, denn outErr ist eine Referenz von $ERR aus dem Hauptprogramm.
13	ERR_RAISE(inErr) ist nicht zulässig und löst deshalb die Meldung 1451 *{(Variablenname)} unzulässiges Argument* aus. ERR_RAISE kann nur $ERR verarbeiten oder eine von $ERR abgeleitete OUT-Variable.

Beispiel 4

$ERR kann nicht nur für Fehlerbehandlungen verwendet werden, sondern auch, um die aktuelle Umgebung zu ermitteln.

In diesem Beispiel wird sowohl aus einem Roboterprogramm als auch aus einem Submit-Programm ein Parameter an ein Unterprogramm übergeben. Im Unterprogramm wird ermittelt, aus welchem Interpreter der Parameter stammt. Je nach Ergebnis wird eine andere Aktion durchgeführt.

Roboterprogramm:

```
DEF Main ()
...
mySUB (55)
...
END
```

Submit-Programm:

```
DEF SPS ()
...
LOOP
   mySUB (33)
   ...
ENDLOOP
...
END
```

Unterprogramm:

```
GLOBAL DEF mySUB (par:IN)
INT par
INI
IF ($ERR.INTERPRETER==#R_INT) THEN
   $OUT_C[par] = TRUE
   ELSE
   $OUT[par] = TRUE
ENDIF
...
END
```

11.7.9 REPEAT ... UNTIL

Beschreibung

Nicht abweisende Schleife. Schleife, die einen Anweisungsblock so lange wiederholt, bis eine bestimmte Bedingung erfüllt ist.

Der Anweisungsblock wird mindestens einmal ausgeführt. Die Bedingung wird nach jedem Schleifendurchlauf geprüft. Wenn die Bedingung erfüllt ist, wird das Programm mit der nächsten Anweisung nach der UNTIL-Zeile fortgesetzt.

Schleifen können verschachtelt werden. Bei verschachtelten Schleifen wird erst die äußere Schleife komplett durchlaufen. Danach wird die innere Schleife komplett durchlaufen.

Syntax

```
REPEAT
Anweisungen
UNTIL Abbruchbedingung
```

Erläuterung der Syntax

Element	Beschreibung
Abbruchbedingung	Typ: BOOL Möglich: ■ Variable vom Typ BOOL ■ Funktion vom Typ BOOL ■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ BOOL

Beispiel 1

Die Schleife wird solange durchlaufen, bis $IN[1] wahr ist.

```
R=1
REPEAT
   R=R+1
UNTIL $IN[1]==TRUE
```

Beispiel 2

Die Schleife wird einmal durchlaufen, obwohl die Abbruchbedingung schon vor dem Schleifendurchlauf erfüllt ist, da die Abbruchbedingung erst am Ende der Schleife abgefragt wird. Nach dem Durchlauf hat `R` den Wert 102.

```
R=101
REPEAT
   R=R+1
UNTIL R>100
```

11.7.10 SWITCH ... CASE ... ENDSWITCH

Beschreibung

Wählt einen von mehreren möglichen Anweisungsblöcken aus, abhängig von einem Auswahlkriterium. Jeder Anweisungsblock hat mindestens eine Kennung. Der Block, dessen Kennung mit dem Auswahlkriterium übereinstimmt, wird ausgewählt.

Wenn der Block abgearbeitet ist, wird das Programm nach ENDSWITCH fortgesetzt.

Wenn keine Kennung mit dem Auswahlkriterium übereinstimmt, dann wird der DEFAULT-Block abgearbeitet. Wenn kein DEFAULT-Block vorhanden ist, wird kein Block abgearbeitet und das Programm nach ENDSWITCH fortgesetzt.

Die SWITCH-Anweisung kann nicht mit EXIT verlassen werden.

Syntax

```
SWITCH Auswahlkriterium
CASE Kennung1 <,Kennung2,...>
Anweisungsblock
<CASE KennungM <,KennungN,...>
Anweisungsblock >
```

```
<DEFAULT
```

Default-Anweisungsblock>

```
 ENDSWITCH
```

Zwischen der SWITCH-Zeile und der ersten CASE-Zeile darf weder eine Leerzeile noch ein Kommentar stehen. Innerhalb einer SWITCH-Anweisung darf DEFAULT nur einmal vorkommen.

Erläuterung der Syntax

Element	Beschreibung
Auswahlkriterium	Typ: INT, CHAR, ENUM Kann eine Variable, ein Funktionsaufruf oder ein Ausdruck vom genannten Datentyp sein.
Kennung	Typ: INT, CHAR, ENUM Der Datentyp der Kennung muss mit dem Datentyp des Auswahlkriteriums übereinstimmen. Ein Anweisungsblock kann beliebig viele Kennungen haben. Mehrere Kennungen müssen durch ein Komma voneinander getrennt werden.

Beispiel 1

Auswahlkriterium und Kennung sind vom Typ INT.

```
INT VERSION
...
SWITCH VERSION
   CASE 1
      UP_1()
   CASE 2,3
      UP_2()
      UP_3()
      UP_3A()
   DEFAULT
      ERROR_UP()
ENDSWITCH
```

Beispiel 2

Auswahlkriterium und Kennung sind vom Typ CHAR. Hier wird die Anweisung `UP_5()` nie ausgeführt, da die Kennung `C` schon vorher verwendet wurde.

```
SWITCH NAME
   CASE "A"
      UP_1()
   CASE "B","C"
      UP_2()
      UP_3()
   CASE "C"
      UP_5()
ENDSWITCH
```

11.7.11 WAIT FOR ...

Beschreibung

WAIT FOR hält das Programm an, bis eine bestimmte Bedingung erfüllt ist. Danach wird das Programm fortgesetzt.

WAIT FOR löst einen Vorlaufstopp aus.

Der Compiler erkennt nicht, wenn der Ausdruck durch eine fehlerhafte Formulierung nie den Wert TRUE annehmen kann. In diesem Fall wird der Programmablauf endlos angehalten, weil das Programm auf eine unerfüllbare Bedingung wartet.

Syntax

`WAIT FOR` *Bedingung*

Erläuterung der Syntax

Element	Beschreibung
Bedingung	Typ: BOOL Bedingung, bei der der Programmablauf fortgesetzt werden soll. ■ Wenn die Bedingung FALSE ist, wird der Programmablauf angehalten, bis die Bedingung TRUE wird. ■ Wenn die Bedingung beim WAIT-Aufruf bereits TRUE ist, wird der Programmablauf nicht angehalten.

Beispiele

Unterbrechen des Programmablaufs bis `$IN[17]` TRUE ist:

```
WAIT FOR $IN[17]
```

Unterbrechen des Programmablaufs bis `BIT1` FALSE ist:

```
WAIT FOR BIT1==FALSE
```

11.7.12 WAIT SEC …

Beschreibung

Hält den Programmablauf an und setzt ihn nach einer Wartezeit fort. Die Wartezeit wird in Sekunden angegeben.

WAIT SEC löst einen Vorlaufstopp aus.

Syntax

`WAIT SEC` *Wartezeit*

Erläuterung der Syntax

Element	Beschreibung
Wartezeit	Typ: INT, REAL Sekunden, für die der Programmablauf unterbrochen werden soll. Wenn der Wert negativ ist, dann wird nicht gewartet. Bei kleinen Wartezeiten ist die Genauigkeit durch ein Vielfaches von 12 ms bestimmt.

Beispiel

Unterbrechen des Programmablaufs für 17,156 Sekunden:

```
WAIT SEC 17.156
```

Unterbrechen des Programmablaufs entsprechend dem Variablenwert von V_ZEIT in Sekunden:

```
WAIT SEC V_ZEIT
```

11.7.13 WHILE ... ENDWHILE

Beschreibung

Abweisende Schleife. Schleife, die einen Anweisungsblock so lange wiederholt, solange eine bestimmte Bedingung erfüllt ist.

Wenn die Bedingung nicht erfüllt ist, wird das Programm mit der nächsten Anweisung nach ENDWHILE fortgesetzt. Die Bedingung wird vor jedem Schleifendurchlauf geprüft. Wenn die Bedingung von vorneherein nicht erfüllt ist, wird der Anweisungsblock nicht ausgeführt.

Schleifen können verschachtelt werden. Bei verschachtelten Schleifen wird erst die äußere Schleife komplett durchlaufen. Danach wird die innere Schleife komplett durchlaufen.

Syntax

`WHILE` *Wiederholbedingung*

Anweisungsblock

```
ENDWHILE
```

Erläuterung der Syntax

Element	Beschreibung
Wieder-holbedingung	Typ: BOOL Möglich: ■ Variable vom Typ BOOL ■ Funktion vom Typ BOOL ■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ BOOL

Beispiel 1

Die Schleife wird 99 mal durchlaufen. Nach dem letzten Durchlauf hat `W` den Wert 100.

```
W=1
WHILE W<100
   W=W+1
ENDWHILE
```

Beispiel 2

Die Schleife wird solange durchlaufen, solange `$IN[1]` TRUE ist.

```
WHILE $IN[1]==TRUE
   W=W+1
ENDWHILE
```

11.8 Ein-/Ausgänge

11.8.1 ANIN

Beschreibung

Zyklisches Lesen (alle 12 ms) eines analogen Eingangs. ANIN löst einen Vorlaufstopp aus.

Die Robotersteuerung verfügt über 32 analoge Eingänge ($ANIN[1] … $ANIN[32]).

- Es sind maximal drei ANIN ON-Anweisungen gleichzeitig zulässig.
- Maximal zwei ANIN ON-Anweisungen können dieselbe Variable *Wert* verwenden oder auf denselben analogen Eingang zugreifen.
- Alle in einer ANIN-Anweisung verwendeten Variablen müssen in Datenlisten deklariert sein (lokal oder in der $CONFIG.DAT).

$ANIN[…] zeigt die Eingangsspannung an, angepasst auf den Bereich zwischen -1.0 und +1.0. Die tatsächliche Spannung hängt von den Einstellungen des Analogmoduls ab.

Syntax

Zyklisches Lesen starten:

`ANIN ON` *Wert* `=` *Faktor* `*` *Signalname* `*` <*±Offset*>

Zyklisches Lesen beenden:

`ANIN OFF` *Signalname*

Erläuterung der Syntax

Element	Beschreibung
Wert	Typ: REAL In *Wert* wird das Ergebnis des zyklischen Lesens abgelegt. *Wert* kann eine Variable oder ein Signalname für einen Ausgang sein.
Faktor	Typ: REAL Beliebiger Faktor. Kann eine Konstante, eine Variable oder ein Signalname sein.
Signalname	Typ: REAL Gibt den analogen Eingang an. *Signalname* muss vorher mit `SIGNAL` deklariert worden sein . Es ist nicht möglich, statt dem Signalnamen direkt den analogen Eingang $ANIN[x] anzugeben.
Offset	Typ: REAL Kann eine Konstante, eine Variable oder ein Signalname sein.

Beispiel

In diesem Beispiel wird der Programm-Override (= Systemvariable $OV_PRO) über den analogen Eingang $ANIN[1] festgelegt.

$ANIN[1] muss vorher im Deklarationsteil mit einem beliebigen Signalnamen, in diesem Fall SIGNAL_1, verknüpft werden.

```
SIGNAL SIGNAL_1 $ANIN[1]
...
ANIN ON $OV_PRO = 1.0 * SIGNAL_1
```

Mit `ANIN OFF` wird das zyklische Abfragen von `SIGNAL_1` beendet:

```
ANIN OFF SIGNAL_1
```

11.8.2 ANOUT

Beschreibung

Zyklisches Schreiben (alle 12 ms) auf einen analogen Ausgang. ANOUT löst einen Vorlaufstopp aus.

Die Robotersteuerung verfügt über 32 analoge Ausgänge ($ANOUT[1] … $ANOUT[32]).

- Es sind maximal vier ANOUT ON-Anweisungen gleichzeitig zulässig.
- Alle in einer ANOUT-Anweisung verwendeten Variablen müssen in Datenlisten deklariert sein (lokal oder in der $CONFIG.DAT).

$ANOUT[…] kann mit Werten von -1.0 bis +1.0 beschrieben werden. Die tatsächlich erzeugte Spannung hängt von den Einstellungen des Analogmoduls ab. Wenn man versucht, Spannungen außerhalb des Wertebereichs zu setzen, zeigt die Robotersteuerung folgende Meldung an: *Begrenzung {Signalname}*

Syntax

Zyklisches Schreiben starten:

`ANOUT ON` *Signalname* `=` *Faktor* `*` *Regelglied* `<`*±Offset*`> <DELAY =` *±Zeit*`> <MINIMUM =` *Minimalwert*`> <MAXIMUM =` *Maximalwert*`>`

Zyklisches Schreiben beenden:

`ANOUT OFF` *Signalname*

Erläuterung der Syntax

Element	Beschreibung
Signalname	Typ: REAL Gibt den analogen Ausgang an. *Signalname* muss vorher mit SIGNAL deklariert worden sein . Es ist nicht möglich, statt dem Signalnamen direkt den analogen Ausgang $ANOUT[x] anzugeben.
Faktor	Typ: REAL Beliebiger Faktor. Kann eine Konstante, eine Variable oder ein Signalname sein.
Regelglied	Typ: REAL Kann eine Konstante, eine Variable oder ein Signalname sein.
Offset	Typ: REAL Kann eine Konstante, eine Variable oder ein Signalname sein.
Zeit	Typ: REAL Einheit: Sekunden. Mit dem Schlüsselwort DELAY und einer positiven oder negativen Zeitangabe kann das Ausgangssignal verzögert (+) oder vorzeitig (-) ausgegeben werden.
Minimalwert, Maximalwert	Typ: REAL Minimal- und/oder Maximalwert, der am Ausgang anliegen soll. Wird nicht unter- oder überschritten, auch wenn die berechneten Werte darunter oder darüber liegen. Zulässige Werte: -1.0 bis +1.0 Kann eine Konstante, eine Variable, eine Strukturkomponente oder ein Feld-Element sein. Der Minimalwert muss in jedem Fall kleiner sein als der Maximalwert. Die Reihenfolge der Schlüsselwörter MINIMUM und MAXIMUM muss eingehalten werden.

Beispiel

In diesem Beispiel steuert der Ausgang $ANOUT[5] die Klebstoff-Ausgabe.

Dem analogen Ausgang wird im Deklarationsteil ein beliebiger Name, in diesem Fall GLUE, zugewiesen. Die Klebstoff-Menge soll abhängig sein von der aktuellen Bahngeschwindigkeit (= Systemvariable $VEL_ACT). Darüber hinaus soll das Ausgangssignal um 0,5 Sekunden vorzeitig ausgegeben werden. Die Minimal-Spannung soll 3 V betragen. (Es wird ein Modul verwendet, dessen Spannung von +10 V bis -10 V geht.)

```
SIGNAL GLUE $ANOUT[5]
...
ANOUT ON GLUE = 0.5 * $VEL_ACT DELAY=-0.5 MINIMUM=0.30
```

Mit ANOUT OFF wird die zyklische Analogausgabe beendet:

```
ANOUT OFF GLUE
```

11.8.3 PULSE

Beschreibung

Setzt einen Impuls. Der Ausgang wird dabei für eine bestimmte Dauer auf einen definierten Pegel gesetzt. Danach wird der Ausgang automatisch vom System zurückgesetzt. Das Setzen und Rücksetzen des Ausgangs geschieht unabhängig vom vorherigen Pegel des Ausgangs.

An maximal 16 Ausgängen gleichzeitig dürfen Impulse gesetzt werden.

Wenn PULSE vor dem ersten Bewegungssatz programmiert wird, läuft die Impulsdauer auch dann ab, wenn die Starttaste wieder losgelassen wird und der Roboter die SAK-Position noch nicht erreicht hat.

Die PULSE-Anweisung löst einen Vorlaufstopp aus. Nur wenn sie in einer TRIGGER-Anweisung verwendet wird, wird sie bewegungsbegleitend ausgeführt.

Bei NOT-HALT, Bedienstopp oder Fehlerstopp wird der Impuls nicht abgebrochen!

Syntax

`PULSE (`*Signal, Pegel, Impulsdauer*`)`

Erläuterung der Syntax

Element	Beschreibung
Signal	Typ: BOOL Ausgang, an den der Impuls angelegt wird. Zulässig sind: ■ OUT[Nr] ■ Signalvariable
Pegel	Typ: BOOL Logischer Ausdruck: ■ TRUE steht für einen positiven Impuls (high). ■ FALSE steht für einen negativen Impuls (low).
Impulsdauer	Typ: REAL Wertebereich: 0,1 bis 3,0 Sekunden. Impulszeiten außerhalb dieses Bereichs lösen einen Programmstopp aus. Pulsraster: 0,1 Sekunden, d.h. die Impulsdauer wird auf- oder abgerundet. Die PULSE-Anweisung wird in der Steuerung im niederprioren Zyklustakt bearbeitet. Dadurch ergibt sich eine Toleranz in der Größenordnung des Pulsrasters (0,1 Sekunden). Im Durchschnitt liegt die Zeitabweichung bei ca. 1% - 2%. Bei sehr kurzen Impulsen liegt die Abweichung bei ca. 13%.

$OUT+PULSE

Wenn ein Ausgang bereits vor dem Impuls gesetzt ist, dann wird er durch die fallende Flanke des Impulses zurückgesetzt:

```
$OUT[50] = TRUE
PULSE($OUT[50],TRUE,0.5)
```

Tatsächlicher Impulsverlauf an Ausgang 50:

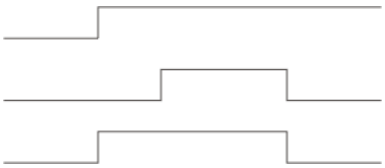


Abb. 11-1: $OUT+PULSE Beispiel 1

Wenn ein Ausgang, der auf Low liegt, mit einem negativen Impuls belegt wird, dann bleibt der Ausgang bis zum Ende des Pulses auf Low und geht dann auf High:

```
$OUT[50] = FALSE
PULSE($OUT[50],FALSE,0.5)
```

Tatsächlicher Impulsverlauf an Ausgang 50:

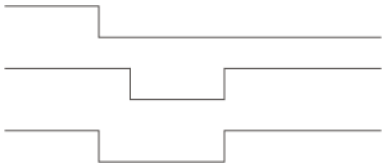


Abb. 11-2: $OUT+PULSE Beispiel 2

PULSE+$OUT

Wenn während der Impulsdauer der gleiche Ausgang gesetzt wird, dann wird dieser durch die fallende Flanke des Impulses zurückgesetzt:

```
PULSE($OUT[50],TRUE,0.5)
$OUT[50] = TRUE
```

Tatsächlicher Impulsverlauf an Ausgang 50:

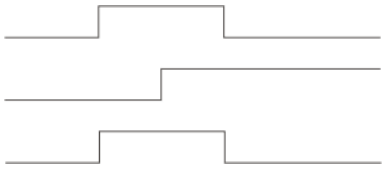


Abb. 11-3: PULSE+$OUT Beispiel 1

Wenn während der Impulsdauer der Ausgang zurückgesetzt wird, dann verkürzt sich die Impulsdauer dementsprechend:

```
PULSE($OUT[50],TRUE,0.5)
$OUT[50] = FALSE
```

Tatsächlicher Impulsverlauf an Ausgang 50:

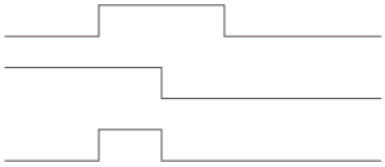


Abb. 11-4: PULSE+$OUT Beispiel 2

Wenn ein Ausgang während eines Impulses auf FALSE gesetzt und dann wieder auf TRUE gesetzt wird, wird der Impuls unterbrochen und beim TRUE-Setzen des Ausgangs fortgesetzt. Die Gesamtdauer von der ersten steigenden Flanke bis zur letzten fallenden Flanke (d. h. inklusive der Dauer der Unterbrechung) entspricht der Dauer, die in der PULSE-Anweisung angegeben wurde.

```
PULSE($OUT[50],TRUE,0.8)
$OUT[50]=FALSE
$OUT[50]=TRUE
```

Tatsächlicher Impulsverlauf an Ausgang 50:

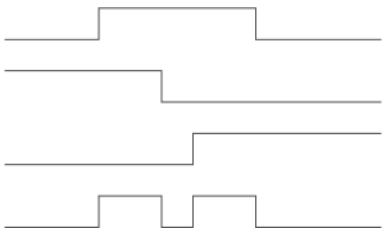


Abb. 11-5: PULSE+$OUT Beispiel 3

Der tatsächliche Impulsverlauf ist nur wie oben angegeben, wenn $OUT[x]= TRUE noch während des Impulses einsetzt. Wenn $OUT[x]= TRUE dagegen nach dem Impuls einsetzt (siehe Linie 3), dann ist der tatsächliche Impulsverlauf folgendermaßen (Linie 4):

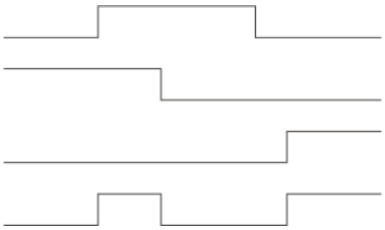


Abb. 11-6: PULSE+$OUT Beispiel 4

PULSE+PULSE

Wenn sich mehrere PULSE-Anweisungen überschneiden, bestimmt immer die letzte PULSE-Anweisung das Ende des gesamten Impulsverlaufs.

Wenn ein Impuls vor der fallenden Flanke nochmals aktiviert wird, dann beginnt in diesem Moment die Dauer des zweiten Impulses. Die gesamte Impulsdauer ist also kürzer als die addierten Werte des ersten und zweiten Impulses:

```
PULSE($OUT[50],TRUE,0.5)
PULSE($OUT[50],TRUE,0.5)
```

Tatsächlicher Impulsverlauf an Ausgang 50:

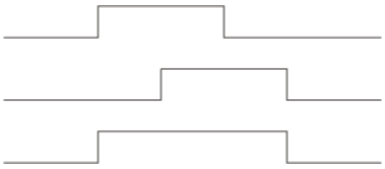


Abb. 11-7: PULSE+PULSE Beispiel 1

Wenn während der Impulsdauer eines positiven Impulses ein zweiter, negativer Impuls auf den gleichen Ausgang gelegt wird, dann wird ab diesem Moment nur noch der zweite Impuls berücksichtigt:

```
PULSE($OUT[50],TRUE,0.5)
PULSE($OUT[50],FALSE,0.5)
```

Tatsächlicher Impulsverlauf an Ausgang 50:

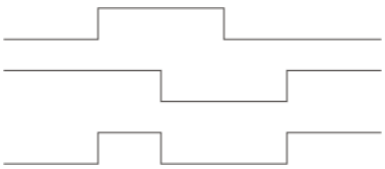


Abb. 11-8: PULSE+PULSE Beispiel 2

```
PULSE($OUT[50],TRUE,3.0)
PULSE($OUT[50],FALSE,1.0)
```

Tatsächlicher Impulsverlauf an Ausgang 50:

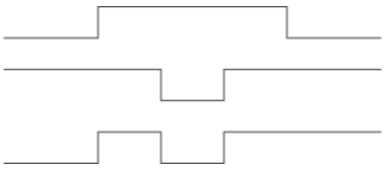


Abb. 11-9: PULSE+PULSE Beispiel 3

PULSE+END

Wenn vor der END-Anweisung ein Impuls programmiert wird, dann verlängert sich die Programmbearbeitungsdauer entsprechend:

```
PULSE($OUT[50],TRUE,0.8)
END
```

Programm aktiv

Tatsächlicher Impulsverlauf an Ausgang 50:

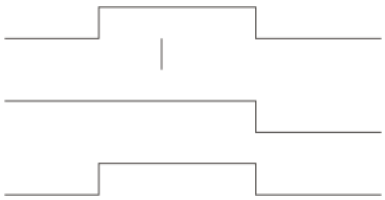


Abb. 11-10: PULSE+END Beispiel

PULSE+RESET/ CANCEL

Wenn die Programmbearbeitung zurückgesetzt (RESET) oder abgebrochen (CANCEL) wird, während ein Impuls aktiv ist, dann wird der Impuls sofort zurückgesetzt:

```
PULSE($OUT[50],TRUE,0.8)
RESET oder CANCEL
```

Tatsächlicher Impulsverlauf an Ausgang 50:

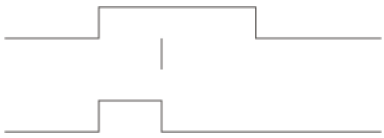


Abb. 11-11: PULSE+RESET Beispiel

11.8.4 SIGNAL

Beschreibung

SIGNAL verknüpft vordefinierte Signalvariablen für Ein- oder Ausgänge mit einem Namen.

Eine solche Verknüpfung, d. h. eine SIGNAL-Vereinbarung, ist notwendig, um einen analogen Ein- oder Ausgang ansprechen zu können. Ein Eingang oder Ausgang darf in mehreren SIGNAL-Vereinbarungen vorkommen.

Der Benutzer kann in folgenden Dateien Signale vereinbaren:

- In DAT-Dateien im Abschnitt EXTERNAL DECLARATIONS
- In SRC-Dateien im Deklarationsteil
- In der $CONFIG.DAT im Abschnitt USER GLOBALS

Es gibt auch SIGNAL-Vereinbarungen, die im System bereits vordefiniert sind. Sie sind im Verzeichnis KRC:\STEU\MADA in der Datei $machine.DAT zu finden. Diese Vereinbarungen können in der $machine.DAT über das Schlüsselwort FALSE stillgelegt werden.

Syntax

Vereinbarung von Signalnamen für Ein- oder Ausgänge:

```
<GLOBAL> SIGNAL Signalname Signalvariable <TO Signalvariable>
```

Stilllegen einer vordefinierten SIGNAL-Vereinbarung:

```
SIGNAL Systemsignalname FALSE
```

Erläuterung der Syntax

Element	Beschreibung
GLOBAL	Nur möglich für Signale, die in einer DAT-Datei definiert sind. (>>> 11.3.4 "Geltungsbereiche" Seite 376)
Signalname	Beliebiger Name

Element	Beschreibung
Signalvariable	Vordefinierte Signalvariable. Folgende Typen stehen zur Auswahl: ■ $IN[x] ■ $OUT[x] ■ $ANIN[x] ■ $ANOUT[x]
`TO`	Fasst mehrere aufeinanderfolgende binäre Ein- oder Ausgänge (maximal 32) zu einem digitalen Ein- oder Ausgang zusammen. Die zusammengefassten Signale lassen sich dezimal, hexadezimal (Präfix H) oder als Bitmuster (Präfix B) ansprechen. Eine Bearbeitung mit boolschen Operatoren ist ebenfalls möglich.
Systemsignalname	Im System vordefinierter Signalname, z. B. `$T1`.
`FALSE`	Legt eine SIGNAL-Vereinbarung, die im System vordefiniert ist, still. Die Ein- oder Ausgänge, auf die sich die SIGNAL-Vereinbarung bezieht, stehen damit wieder für andere Zwecke zur Verfügung. FALSE ist hier kein boolscher Wert, sondern ein Schlüsselwort. Die Option TRUE gibt es nicht. Wenn die mit FALSE stillgelegte SIGNAL-Vereinbarung wieder gültig sein soll, muss die Programmzeile, die das FALSE enthält, gelöscht werden.

Beispiel 1

Dem Ausgang $OUT[7] wird der Name `START_PROCESS` zugewiesen. Der Ausgang $OUT[7] wird gesetzt.

```
SIGNAL START_PROCESS $OUT[7]
START_PROCESS = TRUE
```

Beispiel 2

Die Ausgänge $OUT[1] bis $OUT[8] werden unter dem Namen `OUTWORT` zu einem digitalen Ausgang zusammengefasst. Die Ausgänge $OUT[3], $OUT[4], $OUT[5] und $OUT[7] werden gesetzt.

```
SIGNAL OUTWORT $OUT[1] TO $OUT[8]
OUTWORT = 'B01011100'
```

11.9 Unterprogramme und Funktionen

11.9.1 Unterprogramm aufrufen

Beschreibung

Unterprogramme sind Programme, zu denen aus dem Hauptprogramm verzweigt wird. Wenn das Unterprogramm abgearbeitet ist, wird das Hauptprogramm in der Zeile direkt nach dem Unterprogramm-Aufruf fortgesetzt.

- **Lokale Unterprogramme** sind in der gleichen SRC-Datei enthalten wie das Hauptprogramm. Sie können mit dem Schlüsselwort GLOBAL global bekannt gemacht werden.
- **Globale Unterprogramme** sind Programme mit einer eigenen SRC-Datei, zu denen aus einem anderen Programm verzweigt wird.

Ein Unterprogramm ruft man auf, indem man im Hauptprogramm den Namen des Unterprogramms angibt, gefolgt von runden Klammern.

Beispiel

Im folgenden Beispiel wird das Unterprogramm `my_subprogram` aufgerufen:

```
my_subprogram()
```

11.9.2 Funktion aufrufen

Beschreibung

Eine Funktion ist ein Unterprogramm, das einen bestimmten Wert an das Hauptprogramm zurückliefert. Funktionen besitzen einen Datentyp.

Ein Funktion wird ähnlich aufgerufen wie ein Unterprogramm: Im Hauptprogramm den Namen der Funktion angeben, gefolgt von runden Klammern. Ein Funktionsaufruf kann jedoch nie alleine stehen, sondern der Wert muss stets einer Variablen vom gleichen Datentyp zugewiesen werden.

Beispiel

Beispiele für den Aufruf aus dem Hauptprogramm:

```
REALVAR = REALFUNCTION()
```

```
INTVAR = 5 * INTFUNCTION() + 1
```

11.9.3 DEFFCT ... ENDFCT

Beschreibung

Diese Syntax beschreibt den Aufbau einer Funktion.

Syntax

DEFFCT *Datentyp Name* (<*Variable*: IN |OUT>)

<*Anweisungen*>

RETURN *Funktionswert*

ENDFCT

Erläuterung der Syntax

Element	Beschreibung
Datentyp	Datentyp der Funktion
Name	Name der Funktion
Variable	Wenn in die Funktion ein Wert übergeben wird: Name der Variablen, in die der Wert übergeben wird (>>> 11.9.5 "Parameter in Unterprogramm oder Funktion übergeben" Seite 427)
IN | OUT	Wenn in die Funktion ein Wert übergeben wird: Art der Übergabe
Funktionswert	(>>> 11.9.4 "RETURN" Seite 426)

Beispiel

(>>> 11.9.4 "RETURN" Seite 426)

11.9.4 RETURN

Beschreibung

Sprung aus einem Unterprogramm oder aus einer Funktion zurück in das aufrufende Programm.

Unterprogramme

RETURN kann verwendet werden, um ins Hauptprogramm zurückzukehren, wenn im Unterprogramm eine bestimmte Bedingung erfüllt ist. Es können keine Werte aus dem Unterprogramm an das Hauptprogramm übergeben werden.

Funktionen

Funktionen müssen mit einer RETURN-Anweisung beendet werden, die den ermittelten Wert enthält. Der ermittelte Wert wird dadurch an das aufrufende Programm übergeben.

Syntax

In Unterprogrammen:

RETURN

In Funktionen:

`RETURN` *Funktionswert*

Erläuterung der Syntax

Element	Beschreibung
Funktionswert	Typ: Der Datentyp von *Funktionswert* muss mit dem Datentyp der Funktion übereinstimmen. *Funktionswert* ist der Wert, der mit der Funktion ermittelt wurde. Der Wert kann als Konstante, als Variable oder als Ausdruck angegeben sein.

Beispiel 1

Rücksprung aus einem Unterprogramm in das aufrufende Programm, abhängig von einer Bedingung.

```
DEF PROG_2()
   ...
   IF $IN[5]==TRUE THEN
   RETURN
   ...
END
```

Beispiel 2

Rücksprung aus einer Funktion in das aufrufende Programm. Der Wert `X` wird übergeben.

```
DEFFCT INT CALCULATE(X:IN)
   INT X
   X=X*X
   RETURN X
ENDFCT
```

11.9.5 Parameter in Unterprogramm oder Funktion übergeben

Beschreibung

Aus einem Hauptprogramm können Parameter in lokale und globale Unterprogramme und Funktionen übergeben werden.

Es gibt 2 Arten, wie Parameter übergeben werden können:

- Als IN-Parameter
  Der Wert der Variablen bleibt im Hauptprogramm unverändert.
  Diese Übergabeart wird auch "Call by Value" genannt.
- Als OUT-Parameter
  Das Unterprogramm liest den Wert, verändert ihn und schreibt den neuen Wert zurück ins Hauptprogramm.
  Diese Übergabeart wird auch "Call by Reference" genannt.

Empfehlung: Einen Parameter immer in eine Variable vom gleichen Datentyp übergeben.
Es ist möglich, Parameter in einen anderen Datentyp zu übergeben, allerdings mit bestimmten Einschränkungen.
(>>> 11.9.6 "Parameter an anderen Datentyp übergeben" Seite 431)

Beispiel 1

Parameter an lokales Unterprogramm übergeben:

```
1  DEF MY_PROG( )
2  DECL REAL r,s
3  ...
4  CALC_1(r)
5  ...
6  CALC_2(s)
7  ...
```

```
 8  END
_____________________________________
 9  DEF CALC_1(num1:IN)
10  DECL REAL num1
11  ...
12  END
_____________________________________
13  DEF CALC_2(num2:OUT)
14  DECL REAL num2
15  ...
16  END
```

Zeile	Beschreibung
4	Das Unterprogramm CALC_1 wird aufgerufen und der Parameter "r" wird übergeben.
6	Das Unterprogramm CALC_2 wird aufgerufen und der Parameter "s" wird übergeben.
9	`num1`: Der Name der Variablen, an die der Wert von "r" übergeben wird. `IN` bedeutet: "r" wird nur zum Lesen übergeben.
10, 14	Die Variablen, an die Werte übergeben werden, müssen deklariert werden.
13	`num2`: Der Name der Variablen, an die der Wert von "s" übergeben wird. `OUT` bedeutet: "s" wird übergeben, verändert und zurück ins Hauptprogramm geschrieben.

Beispiel 2

Parameter an globale Funktion übergeben:

Hauptprogramm MY_PROG( ):

```
1  DEF MY_PROG( )
2  DECL REAL result, value
3  value = 2.0
4  result = CALC(value)
5  ...
   ...
   END
```

Zeile	Beschreibung
3	"value" wird der Wert "2.0" zugewiesen.
4	Die Funktion CALC wird aufgerufen, und der Wert von "value" wird übergeben. Der Rückgabewert der Funktion wird der Variablen "result" zugewiesen.

Was passiert, wenn der Wert als IN-Parameter übergeben wird?

Funktion CALC() mit IN:

```
1  DEFFCT REAL CALC(num:IN)
2  DECL REAL return_value, num
3  num = num + 8.0
4  return_value = num * 100.0
5  RETURN(return_value)
6  ENDFCT
```

Zeile	Beschreibung
1	Der Wert von "value" wird als IN-Parameter an "num" übergeben. Der Wert ist noch 2.0.
3	Der Wert von "num" wird verändert. Der Wert ist jetzt 10.0.
4, 5	Der Wert von "return_value" wird berechnet und an das Hauptprogramm in die Variable "result" zurückgegeben. Der Wert ist 1 000.0.
6	Die Funktion ist beendet, das Hauptprogramm wird ab Zeile 5 weiter abgearbeitet. **Hinweis:** Der Wert von "value" im Hauptprogramm ist unverändert 2.0.

Was passiert, wenn der Wert als OUT-Parameter übergeben wird?

Funktion CALC() mit OUT:

```
1  DEFFCT REAL CALC(num:OUT)
2  DECL REAL return_value, num
3  num = num + 8.0
4  return_value = num * 100.0
5  RETURN(return_value)
6  ENDFCT
```

Zeile	Beschreibung
1	Der Wert von "value" wird als OUT-Parameter an "num" übergeben. Der Wert ist noch 2.0.
3	Der Wert von "num" wird verändert. Der Wert ist jetzt 10.0.
4, 5	Der Wert von "return_value" wird berechnet und an das Hauptprogramm in die Variable "result" zurückgegeben. Der Wert ist 1 000.0.
6	Die Funktion ist beendet, das Hauptprogramm wird ab Zeile 5 weiter abgearbeitet. **Hinweis:** Der Wert von "value" im Hauptprogramm ist jetzt 10.0.

Mehrere Parameter übergeben

Mehrere Parameter übergeben:

Welcher Parameter an welchen übergeben wird, wird automatisch durch die Reihenfolge festgelegt: Der erste Parameter wird an den ersten Parameter im Unterprogramm übergeben, der zweite an den zweiten Parameter im Unterprogramm usw.

```
1  DEF MY_PROG( )
2  DECL REAL w
3  DECL INT a, b
4  ...
5  CALC(w, b, a)
6  ...
7  CALC(w, 30, a)
8  ...
9  END

10 DEF CALC(ww:OUT, bb:IN, oo:OUT)
11 DECL REAL ww
12 DECL INT oo, bb
13 ...
14 END
```

Zeile	Beschreibung
5	"w" wird als OUT-Parameter an "ww" übergeben. "b" wird als IN-Parameter an "bb" übergeben. "a" wird als OUT-Parameter an "oo" übergeben.
7	"w" wird als OUT-Parameter an "ww" übergeben. "30" wird als IN-Parameter an "bb" übergeben. "a" wird als OUT-Parameter an "oo" übergeben.

Es ist auch möglich, an eine "Empfänger"-Variable im Unterprogramm keinen Wert zu übergeben, vorausgesetzt, dass dieser Wert im Unterprogamm nicht benötigt wird. Dies erleichtert es, das Programm an sich verändernde Abläufe anzupassen.
Beispiel: CALC (w, ,a)

Felder übergeben

Felder übergeben:

- Felder dürfen nur als OUT-Parameter übergeben werden.
- Nur komplette Felder können in ein anderes Feld übergeben werden.
- Das Feld im Unterprogramm immer ohne Feldgröße deklarieren. Die Feldgröße passt sich dem Ausgangsfeld an.

```
 1  DEF MY_PROG( )
 2  DECL CHAR name[10]
 3  ...
 4  name="OKAY"
 5  CALC(name[])
 6  ...
 7  END

_____________________________________

 8  DEF CALC(my_name[]:OUT)
 9  DECL CHAR my_name[]
10  ...
11  END
```

Zeile	Beschreibung
5, 8	Nur komplette Felder können in ein anderes Feld übergeben werden.
8	Felder dürfen nur als OUT-Parameter übergeben werden.
9	Das Feld im Unterprogramm immer ohne Feldgröße deklarieren. Die Feldgröße passt sich dem ursprünglichen Feld an.

Bei der Übergabe von mehrdimensionalen Felder gibt man ebenfalls keine Feldgrößen an. Allerdings muss die Dimension des Feldes durch Kommas spezifiziert werden.

Beispiele:

`FELD_1D[]` (1-dimensional)

`FELD_2D[,]` (2-dimensional)

`FELD_3D[,,]`(3-dimensional)

Einzelne Feldelemente übergeben:

Ein einzelnes Feldelement darf nur in eine Variable übergeben werden, nicht in ein Feld.

```
 1  DEF MY_PROG( )
 2  DECL CHAR name[10]
 3  ...
 4  name="OKAY"
 5  CALC(name[1])
 6  ...
 7  END

 8  DEF CALC(symbol:IN)
 9  DECL CHAR symbol
10  ...
11  END
```

Zeile	Beschreibung
2	Ein CHAR-Feld mit 10 Elementen wird deklariert.
4	Den ersten 4 Elementen des Felds werden Werte zugewiesen. Dies entspricht: name[1] = "O" name[2] = "K" name[3] = "A" name[4] = "Y" (Eine CHAR-Variable kann immer nur 1 ASCII-Zeichen enthalten.)
5	Das Unterprogramm CALC wird aufgerufen und der Wert des ersten Elements wird übergeben, d. h. der Wert "O".
8	Einzelne Feldelemente können auch als IN-Parameter übergeben werden.
9	Die Variable, an die der Wert des Feldelements übergeben wird, muss deklariert werden (eine Variable, kein Feld).

11.9.6 Parameter an anderen Datentyp übergeben

Einen Wert an den gleichen Datentyp zu übergeben ist immer möglich. Für die Übergabe an einen anderen Datentyp gilt:

Typ im Hauptprogramm	Typ im Unterprogramm	Auswirkung
BOOL	INT, REAL, CHAR	Übergabe nicht möglich; Fehlermeldung
INT, REAL, CHAR	BOOL	
INT	REAL	INT-Wert wird als REAL-Wert verwendet
INT	CHAR	Zeichen aus der ASCII-Tabelle wird verwendet
CHAR	INT	INT-Wert aus der ASCII-Tabelle wird verwendet
CHAR	REAL	REAL-Wert aus der ASCII-Tabelle wird verwendet
REAL	INT	REAL-Werte werden gerundet
REAL	CHAR	REAL-Werte werden gerundet, Zeichen aus der ASCII-Tabelle wird verwendet

11.10 Interrupt-Programmierung

11.10.1 BRAKE

Beschreibung

BRAKE stoppt den Roboter.

BRAKE darf nur in einem Interrupt-Programm verwendet werden. Das Interrupt-Programm wird erst fortgesetzt, wenn der Roboter zum Stillstand gekommen ist. Sobald das Interrupt-Programm abgeschlossen ist, wird die Roboterbewegung fortgesetzt.

Syntax

```
BRAKE <F>
```

Erläuterung der Syntax

Element	Beschreibung
F	F löst einen STOP 1 aus. Bei einer BRAKE-Anweisung ohne F bremst der Roboter mit einem STOP 2.

Beispiel

(>>> 11.10.3 "INTERRUPT" Seite 434)

11.10.2 INTERRUPT ... DECL ... WHEN ... DO

Beschreibung

Bei einem definierten Ereignis, z. B. einem Eingang, unterbricht die Steuerung das aktuelle Programm und arbeitet ein definiertes Unterprogramm ab. Das Ereignis und das Unterprogramm werden mit INTERRUPT ... DECL ... WHEN ... DO definiert.

Wenn das Unterprogramm abgearbeitet ist, wird das unterbrochene Programm an der Unterbrechungsstelle fortgesetzt. Ausnahme: RESUME.

Ein Unterprogramm, das von einem Interrupt aufgerufen wird, wird Interrupt-Programm genannt.

Maximal 32 Interrupts dürfen gleichzeitig deklariert sein. Eine Interrupt-Deklaration kann jederzeit durch eine neue überschrieben werden.

Die Interrupt-Deklaration ist eine Anweisung. Sie muss im Anweisungsteil des Programms stehen und darf nicht im Deklarationsteil stehen!

Nach der Deklaration ist ein Interrupt zunächst deaktiviert. Der Interrupt muss aktiviert werden, bevor auf das definierte Ereignis reagiert werden kann! (>>> 11.10.3 "INTERRUPT" Seite 434)

Interrupt-Programme dürfen keine Spline-Bewegungen enthalten.

Syntax

```
<GLOBAL> INTERRUPT DECL Prio WHEN Ereignis DO Unterprogramm
```

Erläuterung der Syntax

Element	Beschreibung
GLOBAL	Ein Interrupt wird erst ab der Ebene erkannt, in der er deklariert ist. Das bedeutet: Ein Interrupt, der in einem Unterprogramm deklariert wurde, ist im Hauptprogramm nicht bekannt (und kann dort auch nicht aktiviert werden). Wenn ein Interrupt auch in übergeordneten Ebenen bekannt sein soll, dann muss der Deklaration das Schlüsselwort GLOBAL vorangestellt werden.
Prio	Typ: INT Wenn mehrere Interrupts gleichzeitig auftreten, wird zuerst der Interrupt mit der höchsten Priorität bearbeitet, dann die Interrupts niedrigerer Priorität. 1 = höchste Priorität. Zur Verfügung stehen die Prioriäten 1, 2, 4 - 39 und 81 - 128. **Hinweis:** Die Prioriäten 3 und 40 - 80 sind für die Verwendung durch das System reserviert. Sie dürfen vom Benutzer **nicht** verwendet werden, weil dadurch systeminterne Interrupts überschrieben würden und Fehler die Folge wären.
Ereignis	Typ: BOOL Ereignis, bei dem der Interrupt auftreten soll. Unzulässig sind Strukturkomponenten. Zulässig sind: ■ eine globale boolesche Variable ■ ein Signalname ■ ein Vergleich ■ eine einfache logische Verknüpfung: NOT, OR, AND oder EXOR
Unterprogramm	Name des Interrupt-Programms, das abgearbeitet werden soll. Laufzeitvariablen dürfen nicht als Parameter an das Interrupt-Programm übergeben werden, außer Variablen, die in einer Datenliste deklariert sind.

Beispiel 1

Deklaration eines Interrupts mit der Priorität 23, der das Unterprogramm UP1 aufruft, wenn $IN[12] wahr ist. An das Unterprogramm werden die Parameter 20 und VALUE übergeben.

```
INTERRUPT DECL 23 WHEN $IN[12]==TRUE DO UP1(20,VALUE)
```

Beispiel 2

Auf einer programmierten Bahn befinden sich zwei Gegenstände, deren Positionen durch zwei Sensoren, angeschlossen an die Eingänge 6 und 7, erkannt werden. Danach sollen die beiden erkannten Positionen angefahren werden.

Dazu werden die beiden erkannten Positionen als Punkte P_1 und P_2 gespeichert. Im zweiten Teil des Hauptprogramms werden diese Punkte angefahren.

Wenn die Robotersteuerung ein mit INTERRUPT ... DECL ... WHEN ... DO definiertes Ereignis erkennt, speichert sie die aktuelle Roboterposition immer in den Systemvariablen $AXIS_INT (achsspezifisch) und $POS_INT (kartesisch).

Hauptprogramm:

```
DEF PROG()
...
INTERRUPT DECL 10 WHEN $IN[6]==TRUE DO UP1()
INTERRUPT DECL 20 WHEN $IN[7]==TRUE DO UP2()
...
```

```
INTERRUPT ON
LIN START
LIN END
INTERRUPT OFF
LIN P_1
LIN P_2
...
END
```

Lokales Interrupt-Programm 1:

```
DEF UP1()
P_1=$POS_INT
END
```

Lokales Interrupt-Programm 2:

```
DEF UP2()
P_2=$POS_INT
END
```

11.10.3 INTERRUPT

Beschreibung

Führt eine der folgenden Aktionen durch:

- Aktiviert einen Interrupt.
- Deaktiviert einen Interrupt.
- Sperrt einen Interrupt.
- Gibt einen Interrupt frei.

Der Interrupt muss vorher deklariert worden sein. (>>> 11.10.2 "INTERRUPT ... DECL ... WHEN ... DO" Seite 432)

Syntax

`INTERRUPT` *Aktion* <*Nummer*>

Erläuterung der Syntax

Element	Beschreibung
Aktion	■ `ON`: Aktiviert einen Interrupt. ■ `OFF`: Deaktiviert einen Interrupt. ■ `DISABLE`: Sperrt einen aktivierten Interrupt. ■ `ENABLE`: Gibt einen gesperrten Interrupt frei.
Nummer	Typ: INT Nummer (=Priorität) des Interrupts, auf den sich *Aktion* beziehen soll. *Nummer* kann weggelassen werden. In diesem Fall beziehen sich `ON` oder `OFF` auf alle deklarierten Interrupts, `DISABLE` oder `ENABLE` auf alle aktiven Interrupts.

Zur gleichen Zeit dürfen maximal 16 Interrupts aktiv sein. Hierbei ist besonders zu beachten:

- Wenn bei `INTERRUPT ON` die *Nummer* weggelassen wird, werden dadurch alle deklarierten Interrupts aktiv. Die erlaubte Anzahl von 16 darf jedoch nicht überschritten werden.
- Wenn ein Trigger ein Unterprogramm aufruft, zählt er solange als aktiver Interrupt, solange das Unterprogramm noch nicht abgearbeitet ist.

Wenn bei der Interrupt-Deklaration als *Ereignis* eine boolsche Variable, z. B. ein Eingang, definiert wurde:

- In diesem Fall wird der Interrupt durch den Wechsel des Zustandes ausgelöst, z. B. bei $IN[x]==TRUE durch den Wechsel von FALSE auf TRUE. Der Zustand darf also bei INTERRUPT ON nicht bereits bestehen, da dann der Interrupt nicht ausgelöst wird!
- Zusätzlich muss in diesem Fall beachtet werden: Der Zustandswechsel darf frühestens einen Interpolationstakt nach INTERRUPT ON erfolgen.

  (Dies kann erreicht werden, indem man nach INTERRUPT ON ein WAIT SEC 0.012 programmiert. Wenn kein Vorlaufstopp gewünscht ist, kann zusätzlich vor dem WAIT SEC ein CONTINUE programmiert werden.)

  Der Grund ist, dass INTERRUPT ON einen Interpolationstakt (= 12 ms) benötigt, bis der Interrupt tatsächlich aktiviert ist. Wenn der Zustand zuvor wechselt, kann der Interrupt den Wechsel nicht erkennen.

Beispiel 1

Der Interrupt mit der Priorität 2 wird aktiviert. (Der Interrupt muss bereits deklariert sein.)

```
INTERRUPT ON 2
```

Beispiel 2

Hardwaremäßig wird während eines Klebeauftrags ein nicht bahntreuer NOT-HALT durchgeführt. Der Klebeauftrag wird per Programm gestoppt und nach Freigabe (durch Eingang 10) die Klebepistole auf die Bahn zurückpositioniert.

```
DEF PROG()
...
INTERRUPT DECL 1 WHEN $STOPMESS DO STOP_PROG()
LIN P_1
INTERRUPT ON
LIN P_2
INTERRUPT OFF
...
END
```

```
DEF STOP_PROG()
BRAKE F
GLUE=FALSE
WAIT FOR $IN[10]
LIN $POS_RET
GLUE=TRUE
END
```

11.10.4 RESUME

Beschreibung

RESUME bricht alle laufenden Interrupt-Programme und alle laufenden Unterprogramme ab bis zu der Ebene, in der der aktuelle Interrupt deklariert wurde.

RESUME darf nur in Interrupt-Programmen vorkommen. (Jedoch nicht in Interrupt-Programmen, die von einem als GLOBAL deklarierten Interrupt aufgerufen werden). Zum Zeitpunkt der RESUME-Anweisung darf der Vorlaufzeiger nicht in der Ebene sein, in der der Interrupt deklariert wurde, sondern er muss sich mindestens eine Ebene tiefer befinden.

Änderungen der Variablen $BASE im Interrupt-Programm sind nur dort wirksam. Der Rechnervorlauf, d. h. die Variable $ADVANCE, darf im Interrupt-Programm nicht geändert werden.

Zum Verhalten der Robotersteuerung nach einem RESUME ist Folgendes zu beachten:

- Wenn die erste Bewegungsanweisung nach RESUME ein CIRC ist, dann wird dieser als LIN gefahren.
- Wenn die erste Bewegungsanweisung nach RESUME ein SCIRC ist, dann wird dieser als SLIN gefahren.

Grund: Nach einem RESUME befindet sich der Roboter nicht am ursprünglichen Startpunkt der Bewegung. Sie würde dadurch anders ausfallen als ursprünglich geplant, was besonders bei CIRC-/SCIRC-Bewegungen ein erhebliches Gefahrenpotential birgt.

Alle anderen Bewegungen werden nach einem RESUME als die Bewegungsart gefahren, als die sie programmiert sind.

⚠ WARNUNG Wenn die erste Bewegungsanweisung nach RESUME ein CIRC oder SCIRC ist, dann muss der Roboter von jeder möglichen Position, an der er sich bei RESUME befinden könnte, den Zielpunkt der Bewegung gefahrlos als LIN bzw. SLIN anfahren können. Dies muss beim Programmieren von RESUME-Anweisungen berücksichtigt werden.
Wenn dies nicht berücksichtigt wird, können Tod, Verletzungen oder Sachschäden die Folge sein.

Syntax

```
RESUME
```

Beispiel

Der Roboter soll auf einer Bahn ein Teil suchen. Das Teil wird durch einen Sensor am Eingang 15 erkannt. Nachdem das Teil gefunden wurde, soll der Roboter nicht bis zum Endpunkt der Bahn weiterfahren, sondern an die Unterbrechungsposition zurückfahren und das Teil aufnehmen. Dann soll das Hauptprogramm fortgesetzt werden.

Hauptprogramm PROG():

```
DEF PROG()
INI
...
INTERRUPT DECL 21 WHEN $IN[15] DO FOUND()
PTP HOME
...
SEARCH()
...
END
```

Bewegungen, die mit BRAKE und RESUME abgebrochen werden sollen, müssen sich grundsätzlich in einem Unterprogramm befinden. Deshalb ist die Suchstrecke nicht direkt im Hauptprogramm programmiert, sondern im Unterprogramm SEARCH().

Unterprogramm SEARCH() mit Suchstrecke:

```
DEF SEARCH()
INTERRUPT ON 21
SPLINE
  SPL START_SEARCH
  SPL IN_BETWEEN
  SPL END_SEARCH
ENDSPLINE
WAIT FOR TRUE
...
END
```

Der Vorlaufzeiger darf sich zum Zeitpunkt der RESUME-Anweisung nicht in der Ebene befinden, in der der aktuelle Interrupt deklariert wurde. Um dies zu verhindern, wird hier über WAIT FOR TRUE ein Vorlaufstopp ausgelöst.

Interrupt-Programm FOUND():

```
DEF FOUND()
INTERRUPT OFF 21
BRAKE
```

```
LIN $POS_INT
... ;Der Roboter greift das gefundene Teil.
RESUME
END
```

Durch den Bremsvorgang entfernt sich der Roboter etwas von der Stelle, an der der Interrupt ausgelöst wurde. LIN $POS_INT bewirkt, dass der Roboter an die Stelle zurückfährt, an der der Interrupt ausgelöst wurde.

Die Bewegungsart LIN wurde hier verwendet, weil Interrupt-Programme keine Spline-Bewegungen enthalten dürfen.

Nach LIN $POS_INT greift der Roboter das Teil. (Hier im Beispiel nicht ausprogrammiert.)

RESUME bewirkt, dass nach dem Greifen des Teils das Hauptprogramm fortgesetzt wird. Ohne RESUME würde nach END das Unterprogramm SEARCH() fortgesetzt werden.

11.11 Bahnbezogene Schaltaktionen (=Trigger)

11.11.1 TRIGGER WHEN DISTANCE

Beschreibung

Der Trigger löst eine vom Benutzer definierte Anweisung aus. Die Robotersteuerung führt die Anweisung parallel zur Roboterbewegung aus.

Der Trigger kann sich wahlweise auf den Start- oder den Zielpunkt der Bewegung beziehen. Die Anweisung kann entweder direkt am Bezugspunkt ausgelöst werden, oder sie kann noch zeitlich verschoben werden.

Syntax

`TRIGGER WHEN DISTANCE=`*Position* `DELAY=`*Zeit* `DO` *Anweisung* `<PRIO=`*Prioritaet*`>`

Erläuterung der Syntax

Element	Beschreibung
Position	Typ: INT; Variable oder Konstante Bezugspunkt des Triggers ■ **0:** Startpunkt ■ **1:** Zielpunkt (>>> "Bezugspunkt beim Überschleifen" Seite 438)
Zeit	Typ: REAL; Variable, Konstante oder Funktion; Einheit: ms Zeitliche Verschiebung in Bezug auf *Position*. Wenn keine Verschiebung gewünscht ist, dann *Zeit* = 0 setzen. ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang ■ Positiver Wert: Verschiebung in Richtung Bewegungsende (>>> "Max. Verschiebung" Seite 438) Wenn *Zeit* eine Funktion aufruft, gelten für die Funktion Einschränkungen. (>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)

Element	Beschreibung
Anweisung	Möglich: ■ Wertzuweisung an eine Variable **Hinweis:** Auf der linken Seite der Zuweisung darf keine Laufzeitvariable stehen. ■ OUT-Anweisung; PULSE-Anweisung; CYCFLAG-Anweisung ■ Aufruf eines Unterprogramms. In diesem Fall muss *Prioritaet* angegeben werden.
Prioritaet	Typ: INT; Variable oder Konstante Priorität des Triggers. Nur relevant, wenn *Anweisung* ein Unterprogramm aufruft, dann aber obligatorisch. Zur Verfügung stehen die Prioriäten 1, 2, 4 - 39 sowie 81 - 128. Die Prioriäten 40 - 80 sind reserviert für Fälle, in denen die Priorität automatisch durch das System vergeben wird. Wenn die Priorität automatisch durch das System vergeben werden soll, programmiert man: `PRIO = -1`. Wenn mehrere Trigger gleichzeitig Unterprogramme aufrufen, wird zuerst der Trigger mit der höchsten Priorität bearbeitet, dann die Trigger mit niedrigerer Priorität. 1 = höchste Priorität.

Wenn ein Trigger ein Unterprogramm aufruft, zählt er solange als aktiver Interrupt, solange das Unterprogramm noch nicht abgearbeitet ist. Zur gleichen Zeit dürfen maximal 16 Interrupts aktiv sein.

Bezugspunkt beim Überschleifen

Wo liegt der Bezugspunkt, wenn der Start- bzw. Zielpunkt überschliffen ist?

- DISTANCE = 0:
  Wenn der Startpunkt überschliffen ist, liegt der Bezugspunkt am Ende des Überschleifbogens.
- DISTANCE = 1:
  Wenn der Zielpunkt überschliffen ist, liegt der Bezugspunkt in der Mitte des Überschleifbogens.

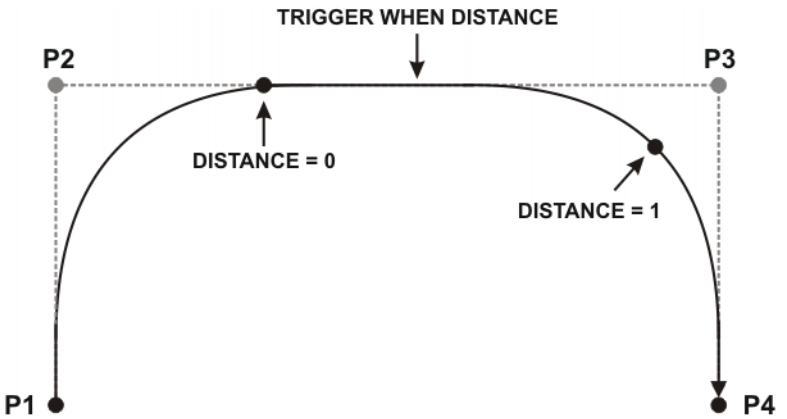


Abb. 11-12: TRIGGER WHEN DISTANCE Bezugspunkt beim Überschleifen

Max. Verschiebung

Der Schaltpunkt kann nicht beliebig weit verschoben werden. Die folgende Tabelle gibt an, welche Verschiebungen maximal möglich sind. Wenn größere und somit ungültige Verschiebungen programmiert werden, schaltet die Robotersteuerung den Trigger spätestens an der erlaubten Grenze. In T1/T2 gibt sie dazu eine Meldung aus.

DISTANCE = …	Negative Verschiebung maximal bis …	Positive Verschiebung maximal bis …
DISTANCE = 0	- - - (Keine negative Verschiebung möglich.)	■ Bis zum Zielpunkt ■ Wenn der Zielpunkt überschliffen ist: Bis zum Anfang des Überschleifbogens
DISTANCE = 1 und Zielpunkt = Genauhalt	■ Bis zum Startpunkt ■ Wenn der Startpunkt überschliffen ist: Bis zum Ende des Überschleifbogens	- - - (Keine positive Verschiebung möglich.)
DISTANCE = 1 und Zielpunkt = überschliffen	Bis zum Anfang des Überschleifbogens des Zielpunkts	Bis zum Ende des Überschleifbogens des Zielpunkts

Beispiel 1

130 Millisekunden nach P_2 wird $OUT[8] auf TRUE gesetzt.

```
LIN P_2
TRIGGER WHEN DISTANCE=0 DELAY=130 DO $OUT[8]=TRUE
LIN P_3
```

Beispiel 2

In der Mitte des Überschleifbogens von P_5 wird das Unterprogramm MY_SUBPROG mit der Priorität 5 aufgerufen.

```
PTP P_4
TRIGGER WHEN DISTANCE=1 DELAY=0 DO MY_SUBPROG() PRIO=5
PTP P_5 C_DIS
PTP P_6
```

Beispiel 3

Erläuterung zur Grafik (>>> Abb. 11-13)**:**

In der Grafik sind mit Pfeilen die ungefähren Positionen eingezeichnet, an denen die Trigger ausgelöst würden. Anfang, Mitte und Ende jedes Überschleifbogens sind markiert (mit **start*, **middle* und **end*).

```
 1  DEF PROG()
 2  ...
 3  PTP P_0
 4  TRIGGER WHEN DISTANCE=0 DELAY=40 DO A=12
 5  TRIGGER WHEN DISTANCE=1 DELAY=-20 DO UP1() PRIO=10
 6  LIN P_1
 7  ...
 8  TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(A) PRIO=5
 9  TRIGGER WHEN DISTANCE=1 DELAY=15 DO B=1
10  LIN P_2 C_DIS
11  ...
12  TRIGGER WHEN DISTANCE=0 DELAY=10 DO UP2(B) PRIO=12
13  TRIGGER WHEN DISTANCE=1 DELAY=0 DO UP(A,B,C) PRIO=6
14  LIN P_3 C_DIS
15  ...
16  TRIGGER WHEN DISTANCE=0 DELAY=50 DO UP2(A) PRIO=4
17  TRIGGER WHEN DISTANCE=1 DELAY=-80 DO A=0
18  LIN P_4
19  ...
20  END
```

Zeile	Schaltbereiche der Trigger
4	Schaltbereich: P_0 bis P_1
5	Schaltbereich: P_0 bis P_1
8	Schaltbereich: P_1 bis P_2*start
9	Schaltbereich: P_2*start bis P_2*end
12	Schaltbereich: P_2*end bis P_3*start
13	Schaltbereich: P_3*start bis P_3*end
16	Schaltbereich: P_3*end bis P_4
17	Schaltbereich: P_3*end bis P_4

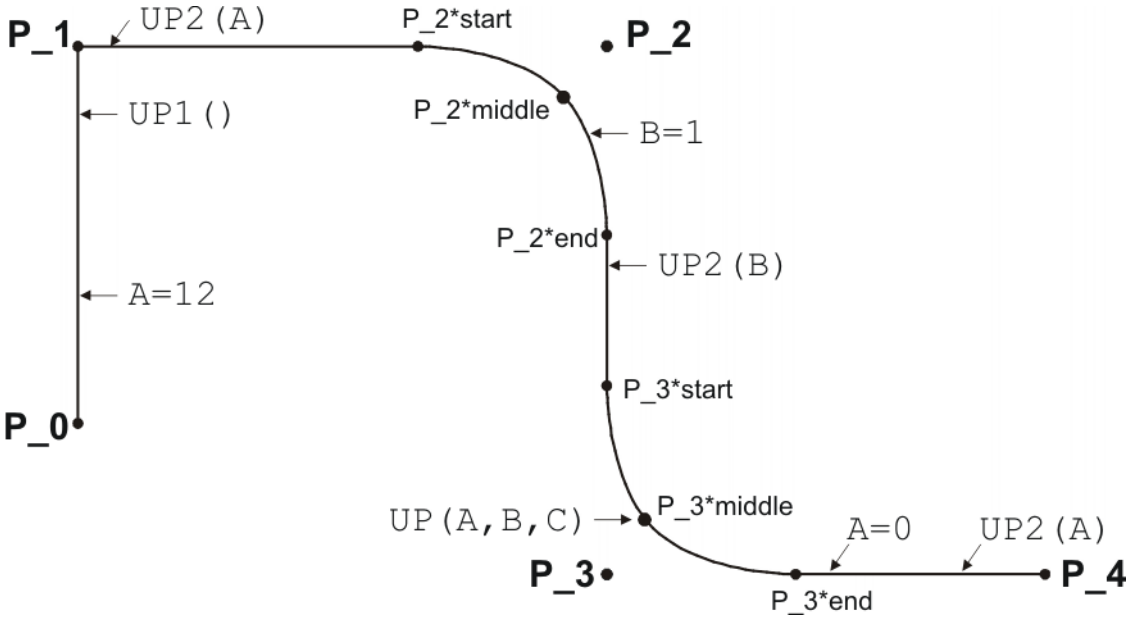


Abb. 11-13: Beispiel TRIGGER WHEN DISTANCE

11.11.2 TRIGGER WHEN PATH

Beschreibung

Der Trigger löst eine vom Benutzer definierte Anweisung aus. Die Robotersteuerung führt die Anweisung parallel zur Roboterbewegung aus.

Der Trigger kann sich wahlweise auf den Start- oder den Zielpunkt der Bewegung beziehen. Die Anweisung kann entweder direkt am Bezugspunkt ausgelöst werden, oder sie kann noch örtlich und/oder zeitlich verschoben werden.

- Der Trigger kann nicht für PTP-Bewegungen verwendet werden.
- Wenn der Trigger in einem Spline-Block verwendet wird, darf er nicht zwischen dem letzten Segment und ENDSPLINE stehen.

Syntax

`TRIGGER WHEN PATH =` *Strecke* `<ONSTART> DELAY =` *Zeit* `DO` *Anweisung* `<PRIO =` *Prioritaet*`>`

Funktionen

`PATH` und `DELAY` können Funktionen aufrufen. Für die Funktionen gelten Einschränkungen.

(>>> 11.11.3 "Einschränkungen für Funktionen im Trigger" Seite 447)

Erläuterung der Syntax

Element	Beschreibung
`ONSTART`	Bezugspunkt des Triggers ■ Mit ONSTART: Startpunkt ■ Ohne ONSTART: Zielpunkt (>>> 11.11.2.1 "Bezugspunkt beim Überschleifen – Übersicht" Seite 444)
Strecke	Typ: REAL; Variable, Konstante oder Funktion; Einheit: mm (Ausnahme: bei PTP-Splines ohne Einheit) Örtliche Verschiebung in Bezug auf den Bezugspunkt. Wenn keine örtliche Verschiebung gewünscht ist, dann *Strecke* = 0 setzen. ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang ■ Positiver Wert: Verschiebung in Richtung Bewegungsende (>>> "Max. Verschiebung" Seite 441)
Zeit	Typ: REAL; Variable, Konstante oder Funktion; Einheit: ms Zeitliche Verschiebung in Bezug auf *Strecke*. Wenn keine zeitliche Verschiebung gewünscht ist, dann *Zeit* = 0 setzen. ■ Negativer Wert: Verschiebung in Richtung Bewegungsanfang ■ Positiver Wert: Trigger wird nach Ablauf von *Zeit* geschaltet. (>>> "Max. Verschiebung" Seite 441)
Anweisung	Möglich: ■ Wertzuweisung an eine Variable **Hinweis:** Auf der linken Seite der Zuweisung darf keine Laufzeitvariable stehen. ■ OUT-Anweisung; PULSE-Anweisung; CYCFLAG-Anweisung ■ Aufruf eines Unterprogramms. In diesem Fall muss *Prioritaet* angegeben werden.
Prioritaet	Typ: INT; Variable oder Konstante Priorität des Triggers. Nur relevant, wenn *Anweisung* ein Unterprogramm aufruft, dann aber obligatorisch. Zur Verfügung stehen die Prioriäten 1, 2, 4 - 39 sowie 81 - 128. Die Prioriäten 40 - 80 sind reserviert für Fälle, in denen die Priorität automatisch durch das System vergeben wird. Wenn die Priorität automatisch durch das System vergeben werden soll, programmiert man: `PRIO = -1`. Wenn mehrere Trigger gleichzeitig Unterprogramme aufrufen, wird zuerst der Trigger mit der höchsten Priorität bearbeitet, dann die Trigger mit niedrigerer Priorität. 1 = höchste Priorität.

Wenn ein Trigger ein Unterprogramm aufruft, zählt er solange als aktiver Interrupt, solange das Unterprogramm noch nicht abgearbeitet ist. Zur gleichen Zeit dürfen maximal 16 Interrupts aktiv sein.

Max. Verschiebung

Der Schaltpunkt kann nur bis zu bestimmten Grenzen verschoben werden. Wenn größere und somit ungültige Verschiebungen programmiert werden,

schaltet die Robotersteuerung den Trigger spätestens an der erlaubten Grenze. In T1/T2 gibt sie dazu eine Meldung aus.

Maximale Verschiebung für *Strecke* + negativen *Zeit*-Wert:

Die Grenzen gelten für die Gesamtverschiebung, die sich aus örtlicher und negativer zeitlicher Verschiebung ergibt.

Negative Verschiebung maximal bis …	Positive Verschiebung maximal bis …
Bis zum Startpunkt (wenn dieser nicht überschliffen ist)	Bis zum Zielpunkt (wenn dieser nicht überschliffen ist)
Wenn der Startpunkt überschliffen ist: ■ Wenn der Startpunkt ein überschliffener PTP-Punkt ist: Bis zum Ende des Überschleifbogens ■ Wenn der Startpunkt ein anderer überschliffener Punkt ist: Bis zum Anfang des Überschleifbogens	Wenn der Zielpunkt überschliffen ist: ■ Beim homogenen Überschleifen: Bis zum nächsten Genauhalt nach der TRIGGER-Anweisung ■ Beim gemischten Überschleifen (Spline): Bis zum Schaltpunkt, den ein ONSTART-Trigger mit PATH = 0 hätte, wenn er in der Bewegung stände, in die hinein überschliffen wird. (>>> 11.11.2.3 "Bezugspunkt beim gemischten Überschleifen (Spline)" Seite 446) ■ Beim gemischten Überschleifen (LIN/CIRC/PTP): Bis zum Anfang des Überschleifbogens

Maximale Verschiebung für positiven *Zeit*-Wert:

Die positive zeitliche Verschiebung kann maximal 1 000 ms betragen. Jede zeitliche Verschiebung zwischen 0 und 1 000 ms wird geschaltet, auch dann, wenn das Programm inzwischen bereits abgewählt wurde!

Beispiel

```
LIN P_2 C_DIS
TRIGGER WHEN PATH = -20.0 DELAY= -10 DO $OUT[2]=TRUE
LIN P_3 C_DIS
LIN P_4 C_DIS
LIN P_5
```

In der Grafik ist mit einem Pfeil die ungefähre Position eingezeichnet, an der die Anweisung $OUT[2]=TRUE ausgelöst würde.

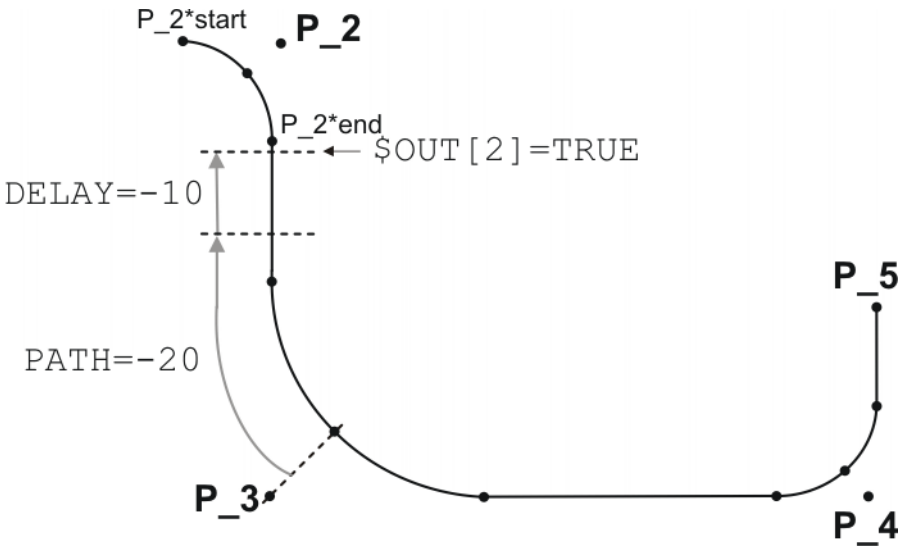


Abb. 11-14: Beispiel TRIGGER WHEN PATH

Schaltbereich: P_2*start bis P_5.

Wenn P_2 nicht überschliffen wäre, wäre der Schaltbereich P_2 bis P_5.

Der Schaltbereich geht bis P_5, weil P_5 der nächste Genauhaltepunkt nach dem TRIGGER ist. Wenn P3 nicht überschliffen wäre, wäre der Schaltbereich P_2 bis P_3, da P_3 im Programm der nächste Genauhaltepunkt nach dem Trigger ist.

Beispiel

```
1  PTP P0
2  SPLINE
3   SPL P1
4   SPL P2
5   SPL P3
6   SPL P4
7   TRIGGER WHEN PATH=0 ONSTART DELAY=10 DO $OUT[5]=TRUE
8   SCIRC P5, P6
9   SPL P7
10  TRIGGER WHEN PATH=-20.0 DELAY=0 DO SUBPR_2() PRIO=-1
11  SLIN P8
12 ENDSPLINE
```

Der Trigger in Zeile 10 würde zum gleichen Ergebnis führen, wenn er direkt vor dem Spline-Block (also zwischen Zeile 1 und Zeile 2) stehen würde. In beiden Fällen bezieht er sich auf den letzten Punkt der Spline-Bewegung, auf P8.

Empfohlen wird jedoch, den Trigger wie im Beispiel zu platzieren und nicht direkt vor dem Spline-Block.

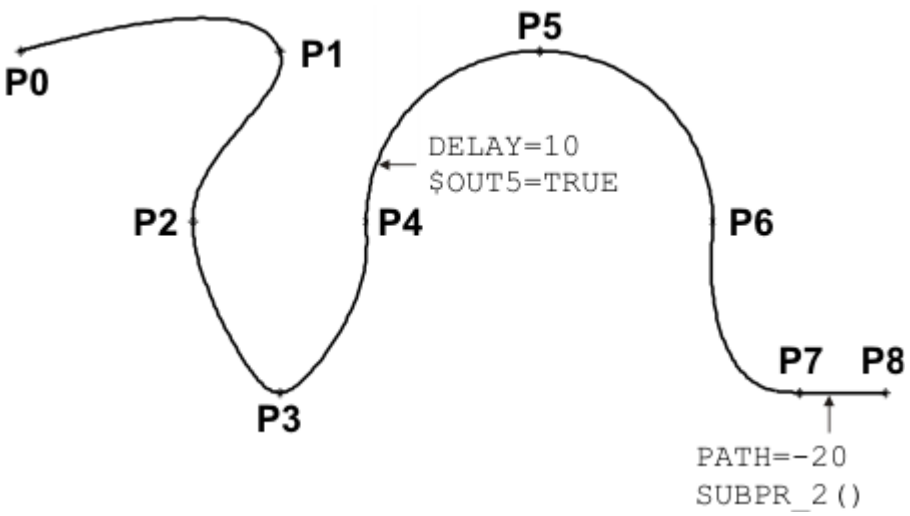


Abb. 11-15: Beispiel TRIGGER WHEN PATH (für Spline)

11.11.2.1 Bezugspunkt beim Überschleifen – Übersicht

Wo liegt der Bezugspunkt eines PATH-Triggers, wenn der Start- bzw. Zielpunkt überschliffen ist?

Dies ist in erster Linie abhängig davon, ob es sich um ein homogenes oder ein gemischtes Überschleifen handelt.

Homogen

Homogenes Überschleifen

- Von einer CP-Spline-Bewegung in eine CP-Spline-Bewegung
- Von einer PTP-Spline-Bewegung in eine PTP-Spline-Bewegung
- Von einer LIN- oder CIRC-Bewegung in eine LIN- oder CIRC-Bewegung

Jede Spline-Bewegung kann Spline-Block oder Einzelsatz sein.

(>>> 11.11.2.2 "Bezugspunkt beim homogenen Überschleifen" Seite 444)

Gemischt

Gemischtes Überschleifen

Hier ist die Position des Bezugspunkts zusätzlich davon abhängig, ob es sich um Spline-Bewegungen handelt oder um herkömmliche Bewegungen.

- Von einer CP-Spline-Bewegung in eine PTP-Spline-Bewegung, oder umgekehrt

  Jede Spline-Bewegung kann Spline-Block oder Einzelsatz sein.

  (>>> 11.11.2.3 "Bezugspunkt beim gemischten Überschleifen (Spline)" Seite 446)
- Von einer PTP-Bewegung in eine LIN- oder CIRC-Bewegung, oder umgekehrt

  (>>> 11.11.2.4 "Bezugspunkt beim gemischten Überschleifen (LIN/CIRC/PTP)" Seite 447)

11.11.2.2 Bezugspunkt beim homogenen Überschleifen

Das Prinzip wird hier anhand eines Beispiels mit CP-Spline-Blöcken erläutert. Es gilt ebenso für die anderen Arten des homogenen Überschleifens.

Beispiel

```
SPLINE
  ...
  SLIN P2
  TRIGGER WHEN PATH=0 DELAY=0 DO ...  ;Trigger 1
```

```
  SLIN P3
ENDSPLINE C_SPL
SPLINE
  TRIGGER WHEN PATH=0 ONSTART DELAY=0 DO ...  ;Trigger 2
  SLIN P4
  ...
ENDSPLINE
```

Trigger 1 und 2 beziehen sich beide auf P3. P3 ist überschliffen. Die Robotersteuerung überträgt den Punkt entsprechend der Überschleifdistanz auf den Überschleifbogen (= P3').

Zielpunkt überschliffen

Bezugspunkt von Trigger 1:

Die Robotersteuerung berechnet, wie weit es vom Beginn des Überschleifbogens bis zum Zielpunkt mit Genauhalt wäre. Diese Distanz wird auf den Überschleifbogen übertragen.

Die Strecke $P_{StartApprox}$ → **P3** ist gleich lang wie $P_{StartApprox}$ → $P3'_{Trigger\ 1}$.

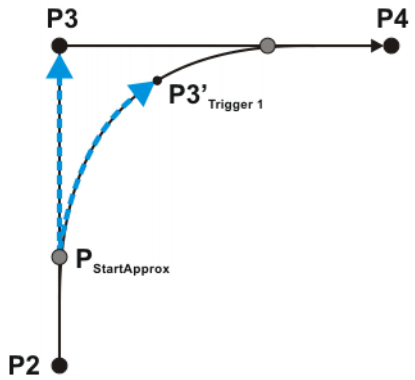


Abb. 11-16: Trigger 1: Bezugspunkt beim homogenen Überschleifen

$P_{StartApprox}$	Beginn des Überschleifbogens
P3	Bezugspunkt bei Genauhalt
$P3'_{Trigger\ 1}$	Bezugspunkt bei Überschleifen

Startpunkt überschliffen

Bezugspunkt von Trigger 2:

Die Robotersteuerung berechnet, wie weit es vom Ende des Überschleifbogens zurück bis zum Startpunkt mit Genauhalt wäre. Diese Distanz wird auf den Überschleifbogen übertragen.

Die Strecke $P_{EndApprox}$ → **P3** ist gleich lang wie $P_{EndApprox}$ → $P3'_{Trigger\ 2}$.

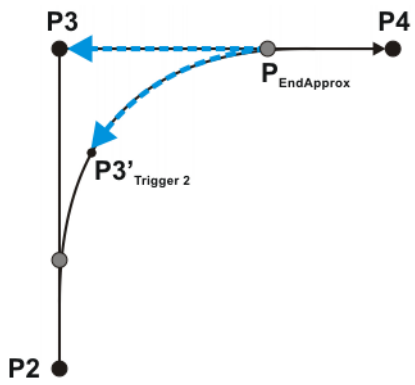


Abb. 11-17: Trigger 2: Bezugspunkt beim homogenen Überschleifen

$P_{EndApprox}$	Ende des Überschleifbogens
P3	Bezugspunkt bei Genauhalt
$P3'_{Trigger\ 2}$	Bezugspunkt bei Überschleifen

11.11.2.3 Bezugspunkt beim gemischten Überschleifen (Spline)

Beispiel

```
PTP_SPLINE
  ...
  SPTP P2
  TRIGGER WHEN PATH=0 DELAY=0 DO ...  ;Trigger 1
  SPTP P3
ENDSPLINE C_SPL

SPLINE
  TRIGGER WHEN PATH=0 ONSTART DELAY=0 DO ...  ;Trigger 2
  SLIN P4
  ...
ENDSPLINE
```

Trigger 1 und 2 beziehen sich beide auf P3. P3 ist überschliffen.

Startpunkt überschliffen

Bezugspunkt von Trigger 2:

> Dieser Bezugspunkt muss zuerst betrachtet werden, da sich der Bezugspunkt von Trigger 1 auf ihn bezieht!

Der Bezugspunkt ergibt sich nach dem gleichen Prinzip wie beim homogenen Überschleifen.

(>>> "Startpunkt überschliffen" Seite 445)

Zielpunkt überschliffen

Bezugspunkt von Trigger 1:

Der Bezugspunkt für Trigger 1 liegt an der gleichen Position wie der von Trigger 2.

Die Strecke $P_{StartApprox}$ → $P3'_{Trigger\ 1}$ ist in der Regel kürzer als $P_{StartApprox}$ → **P3**.

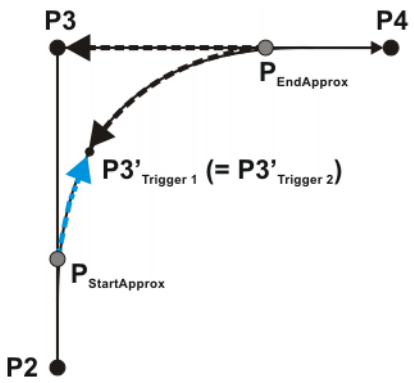


Abb. 11-18: Trigger 1: Bezugspunkt beim gemischten Überschleifen

$P_{StartApprox}$	Beginn des Überschleifbogens
$P_{EndApprox}$	Ende des Überschleifbogens
P3	Bezugspunkt bei Genauhalt
P3'	Bezugspunkt bei Überschleifen

Wenn der Trigger 1 verschoben würde und er durch die Verschiebung zwischen $\mathbf{P_{StartApprox}}$ und **P3'** zu liegen käme, ergibt sich die genaue Position folgendermaßen:

Die Robotersteuerung berechnet, bei wieviel Prozent der Strecke $\mathbf{P_{StartApprox}}$ → **P3** der Schaltpunkt liegen würde, wenn der Zielpunkt ein Genauhalt wäre. Dieser Anteil wird auf den Überschleifbogen übertragen. Der Schaltpunkt liegt also auf x % der Strecke $\mathbf{P_{StartApprox}}$ → $\mathbf{P3'_{Trigger\ 1}}$

11.11.2.4 Bezugspunkt beim gemischten Überschleifen (LIN/CIRC/PTP)

Startpunkt überschliffen

PTP-CP-Überschleifen:

Der Bezugspunkt liegt am Ende des Überschleifbogens.

Zielpunkt überschliffen

CP-PTP-Überschleifen:

Der Bezugspunkt liegt am Anfang des Überschleifbogens.

11.11.3 Einschränkungen für Funktionen im Trigger

Die Werte für `DELAY` und `PATH` können über Funktionen zugewiesen werden. Für diese Funktionen gelten folgende Einschränkungen:

- Das KRL-Programm, das die Funktion enthält, muss die Eigenschaft **Versteckt** haben. (>>> 7.4.2 "Eigenschaften von Dateien und Ordnern anzeigen oder ändern" Seite 241)
- Die Funktion muss global gültig sein.
- Die Funktionen dürfen nur folgende Anweisungen oder Elemente enthalten:
  - Wertzuweisungen
  - IF-Anweisungen
  - Kommentare
  - Leerzeilen
  - RETURN
  - Systemvariable lesen
  - Vordefinierte KRL-Funktion aufrufen

11.11.4 Nützliche Systemvariablen für das Arbeiten mit PATH-Triggern

11.11.4.1 $DIST_NEXT

Beschreibung

$DIST_NEXT gibt die Länge der Bahn von der aktuellen TCP-Position bis zum nächsten geteachten Punkt an.

Typ: REAL. Einheit:

- Bei CP-Bewegungen (Spline und herkömmlich): mm
- Bei SPTP-Bewegungen: Ohne Einheit

$DIST_NEXT kann nicht für PTP-Bewegungen verwendet werden. Der Wert ist dann immer Null.

$DIST_NEXT ist schreibgeschützt.

Vorgehensweise

$DIST_NEXT kann als Hilfe beim Programmieren von PATH-Triggern ohne ONSTART verwendet werden: Man kann damit ermitteln, welchen Wert man dem PATH-Parameter zuweisen muss.

1. Die Position auf der Bahn anfahren, wo der Schaltpunkt liegen soll.

2. Die Systemvariable auslesen.
3. Den Trigger vor dem nächsten Punkt programmieren.
   - Den Trigger ohne ONSTART programmieren.
   - Dem PATH-Parameter den Wert der Systemvariable zuweisen.

11.11.4.2 $DIST_LAST

Beschreibung

$DIST_LAST gibt die Länge der Bahn von der aktuellen TCP-Position bis zum vorhergehenden geteachten Punkt an. Der Wert ist in der Regel positiv.

Typ: REAL. Einheit:

- Bei CP-Bewegungen (Spline und herkömmlich): mm
- Bei SPTP-Bewegungen: Ohne Einheit

$DIST_LAST kann nicht für PTP-Bewegungen verwendet werden. Der Wert ist dann immer Null.

$DIST_LAST ist schreibgeschützt.

Vorgehensweise

$DIST_LAST kann als Hilfe beim Programmieren von PATH-Triggern mit ONSTART verwendet werden: Man kann damit ermitteln, welchen Wert man dem PATH-Parameter zuweisen muss.

1. Die Position auf der Bahn anfahren, wo der Schaltpunkt liegen soll.
2. Die Systemvariable auslesen.
3. Den Trigger nach dem vorhergehenden Punkt programmieren.
   - Den Trigger mit ONSTART programmieren.
   - Dem PATH-Parameter den Wert der Systemvariable zuweisen.

11.12 Kommunikation

Informationen zu folgenden Anweisungen sind in der Experten-Dokumentation CREAD/CWRITE zu finden:

- CAST_FROM
- CAST_TO
- CCLOSE
- CHANNEL
- CIOCTL
- COPEN
- CREAD
- CWRITE
- SREAD
- SWRITE

11.13 Operatoren

Bei jeder Operation prüft der Compiler die Zulässigkeit der Operanden.

11.13.1 Arithmetische Operatoren

Beschreibung

In KRL sind alle 4 Grundrechenarten zulässig.

Operator	Beschreibung
+	Addition oder positives Vorzeichen
-	Subtraktion oder negatives Vorzeichen

Operator	Beschreibung
*	Multiplikation
/	Division

Die arithmetischen Operatoren können auf die Datentypen INT und REAL angewendet werden.

Operand	Operand	Ergebnis
INT	INT	INT
INT	REAL	REAL
REAL	REAL	REAL

Wenn das Ergebnis einer INT-Division nicht ganzzahlig ist, werden die Nachkomma-Stellen abgeschnitten.

Beispiele

```
DEF ARITH()
DECL INT A,B,C,D,E
DECL REAL K,L,M
INI
A = 2            ;A=2
B = 9.8          ;B=10
C = 9.50         ;C=10
D = 9.48         ;D=9
E = 7/4          ;E=1
K = 3.5          ;K=3.5
L = 1.0          ;L=1.0
M = 3            ;M=3.0
...
A = A * E        ;A=2
B = B - 'HB'     ;B=-1
E = E + K        ;E=5
K = K * 10       ;K=35.0
L = 10/4         ;L=2.0
L = 10/4.0       ;L=2.5
L = 10/4.        ;L=2.5
L = 10./4        ;L=2.5
E = 10./4.       ;E=3
M = (10/3) * M   ;M=9.0
END
```

11.13.2 Geometrischer Operator

Beschreibung

Mit dem geometrischen Operator kann man Positionen geometrisch addieren. Die geometrische Addition wird auch "Frame-Verknüpfung" genannt.

Der geometrische Operator wird in KRL durch einen Doppelpunkt ":" dargestellt.

Der geometrische Operator ist z. B. für folgende Zwecke geeignet:

- Positionen verschieben, um sie an eine veränderte Werkstück-Größe anzupassen
- Rückzugs-Strategien

Beispiel

Mit dieser Anweisung zieht sich das Werkzeug 100 mm entgegen der Stoßrichtung zurück, unabhängig davon, an welcher Position sich der Roboter gerade befindet.

```
LIN  $POS_ACT : {x -100, y 0, z 0, a 0, b 0, c 0}
```

Voraussetzung ist, dass die Stoßrichtung in X-Richtung liegt.

$POS_ACT ist eine Systemvariable vom Strukturtyp E6POS und enthält die aktuelle kartesische Roboterposition.

Verknüpfte Typen

Der geometrische Operator kann die Datentypen FRAME und POS/E6POS verknüpfen.

Die Komponenten X, Y, Z, A, B und C müssen mit einem Wert belegt sein. Die Komponenten S und T bleiben von der Verknüpfung unberührt und müssen daher nicht mit einem Wert belegt sein.

Das Ergebnis hat immer den Datentyp des am weitesten rechts stehenden Operanden.

Verknüpfung von 2 Operanden:

Links	:	Rechts	Ergebnis
POS	:	POS	POS
POS	:	FRAME	FRAME
FRAME	:	FRAME	FRAME
FRAME	:	POS	POS

Beispiele für die Verknüpfung von 3 Operanden:

Links	:	Mitte	:	Rechts	Ergebnis
POS	:	POS	:	POS	POS
POS	:	POS	:	FRAME	FRAME
POS	:	FRAME	:	FRAME	FRAME
FRAME	:	FRAME	:	POS	POS

Bedeutung der Operanden

Wie kann man sich vorstellen, was die Operanden bedeuten?

Dies wird hier am vorher genannten Beispiel für eine Rückzugsbewegung dargestellt:

Linker Operand	:	Rechter Operand
`$POS_ACT`	:	`{x -100, y0, z0, a0, b0, c0}`
		Gehe zu diesem Ziel, …
… und zwar bezogen auf Koordinaten und Orientierung dieser Position.		

11.13.2.1 Reihenfolge der Operanden

Das Ergebnis einer geometrischen Addition unterscheidet sich je nach Reihenfolge der Operanden. Dies zeigt das folgende Beispiel in einer grafischen Darstellung.

- A = {x 1, y 1, z 0, a 0, b 0, c 0}
- B = {x 3, y 2, z 0, a -45, b 0, c 0}
- CS = Ursprungs-Koordinatensystem

Man kann das Ergebnis einer Operation rechnerisch mit KRL ermitteln. Es gibt die Position des rechten Operanden bezogen auf das Koordinatensystem des linken Operanden an.

Reihenfolge A:B

R = A : B bedeutet:

- A bezieht sich auf CS.
- B bezieht sich auf A.

Das Ergebnis gibt die Position von B bezogen auf CS an:

R = {x 4, y 3, a -45}

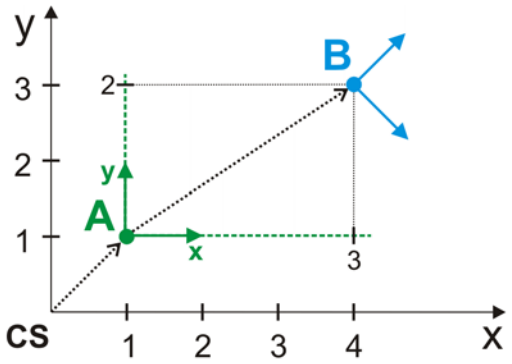


Abb. 11-19: R = A : B

Reihenfolge B:A

R = B : A bedeutet:

- B bezieht sich auf CS.
- A bezieht sich auf B.

Das Ergebnis gibt die Position von A bezogen auf CS an:

R = {x 4,414, y 2, a -45}

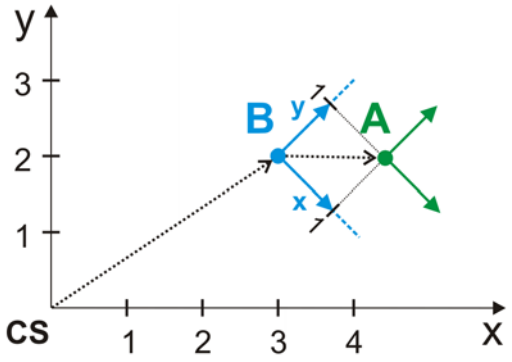


Abb. 11-20: R = B : A

11.13.2.2 Beispiel für doppelte Verknüpfung

Beschreibung

Dieses Beispiel zeigt, wie man mehrere Koordinatensysteme verknüpfen kann.

Um die Auswirkung der Verknüpfungen darzustellen, wird der Ursprung jedes Koordinatensystems bzw. der Verknüpfung angefahren. Dort wird 2 Sekunden gewartet, um die Position zu verdeutlichen. Um die Orientierungsänderung zu verdeutlichen, fährt die Werkzeugspitze danach zuerst 100 mm in X-Richtung, dann 100 mm in Y-Richtung und dann 100 mm in Z-Richtung.

Programm

```
 1  DEF geo_operator( )
 2  DECL AXIS home
 3  DECL FRAME ref_pos_x, ref_pos_y, ref_pos_z
 4  DECL FRAME My_BASE[2]
...
 5  INI
 6  home={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 30,A6 0}
 7  $BASE={X 1000,Y 0,Z 1000,A 0,B 0,C 0}
 8  ref_pos_X={X 100,Y 0,Z 0,A 0,B 0,C 0}
 9  ref_pos_Y={X 100,Y 100,Z 0,A 0,B 0,C 0}
10  ref_pos_Z={X 100,Y 100,Z 100,A 0,B 0,C 0}
11  My_BASE[1]={X 200,Y 100,Z 0,A 0,B 0,C 180}
```

```
12  My_BASE[2]={X 0,Y 200,Z 250,A 0,B 90,C 0}
...
13  PTP home
14  PTP $BASE
15  WAIT SEC 2
16  PTP ref_pos_X
17  PTP ref_pos_Y
18  PTP ref_pos_Z
19  PTP My_BASE[1]
20  WAIT SEC 2
21  PTP My_BASE[1]:ref_pos_X
22  PTP My_BASE[1]:ref_pos_Y
23  PTP My_BASE[1]:ref_pos_Z
24  PTP My_BASE[1]:My_BASE[2]
25  WAIT SEC 2
26  PTP My_BASE[1]:My_BASE[2]:ref_pos_X
27  PTP My_BASE[1]:My_BASE[2]:ref_pos_Y
28  PTP My_BASE[1]:My_BASE[2]:ref_pos_Z
29  PTP My_BASE[2]:My_BASE[1]
30  WAIT SEC 2
31  PTP My_BASE[2]:My_BASE[1]:ref_pos_X
32  PTP My_BASE[2]:My_BASE[1]:ref_pos_Y
33  PTP My_BASE[2]:My_BASE[1]:ref_pos_Z
34  PTP home
35  END
```

Zeile	Beschreibung
8 … 10	3 Frames initialisieren, für die Bewegung in X-, Y- und X-Richtung.
11, 12	2 benutzerspezifische Koordinatensysteme initialisieren. Diese dienen als Beispiele für die Verknüpfungen.
14	Den Ursprung des $BASE-Koordinatensystems anfahren.
16 … 18	In $BASE zunächst 100 mm in X-Richtung verfahren, dann 100 mm in Y-Richtung und dann 100 mm in Z-Richtung.
19	In $BASE den Ursprung des Koordinatensystems My_BASE[1] anfahren.
21 … 23	Die gleichen Koordinaten wie in Zeile 16 … 18 anfahren, jedoch diesmal nicht in $BASE, sondern im Koordinatensystem My_BASE[1]. D. h., diese Punkte liegen im Raum woanders als die Punkte von Zeile 16 … 18.
24	In My_BASE[1] den Ursprung des Koordinatensystems My_BASE[2] anfahren. My_BASE[1] selber liegt in $BASE.
26 … 28	Die gleichen Koordinaten wie in Zeile 16 … 18 werden angefahren, diesmal im Koordinatensystem My_BASE[1]:My_BASE[2].
29	In My_BASE[2] den Ursprung des Koordinatensystems My_BASE[1] anfahren. My_BASE[2] selber liegt in $BASE.
31 … 33	Die gleichen Koordinaten wie in Zeile 16 … 18 werden angefahren, diesmal im Koordinatensystem My_BASE[2]:My_BASE[1].

11.13.3 Vergleichsoperatoren

Beschreibung

Mit den Vergleichsoperatoren können logische Ausdrücke gebildet werden. Das Ergebnis eines Vergleichs ist immer vom Typ BOOL.

Operator	Beschreibung	Zulässige Datentypen
==	gleich	INT, REAL, CHAR, ENUM, BOOL
<>	ungleich	
>	größer	INT, REAL, CHAR, ENUM
<	kleiner	
>=	größer gleich	
<=	kleiner gleich	

- Kombinationen von Operanden aus INT, REAL und CHAR sind zulässig.
  Der Vergleich von Zahlenwerten (INT, REAL) und Zeichenwerten (CHAR) ist zulässig, weil jedem ASCII-Zeichen ein ASCII-Code zugeordnet ist. Der Code ist eine Zahl.
- Ein BOOL-Typ darf nur mit einem BOOL-Typen verglichen werden.
- Ein ENUM-Typ darf nur mit demselben ENUM-Typen verglichen werden.

Bei REAL-Werten ist die Prüfung auf Gleichheit oder Ungleichheit nur eingeschränkt sinnvoll: Aufgrund der begrenzten Anzahl an Gleitkomma-Stellen sind Rundungsfehler möglich. Diese können dazu führen, dass identische Formeln ungleiche Werte liefern.

Beispiele

Auch mehrfache Vergleiche sind zulässig:

```
...
DECL BOOL A, B
...
B= 10 < 3                                          ;B=FALSE
A = 10/3 == 3                                      ;A=TRUE
B = ((B == A) <> (10.00001 >= 10)) == TRUE         ;B=TRUE
A = "F" < "Z"                                      ;A=TRUE
...
```

Beispiel für einen Vergleich bei einem ENUM-Typ:

```
DEF TEST()
ENUM color_typ orange, blue
DECL BOOL A
DECL color_typ KUKA_color, my_color
INI
KUKA_color = #orange
my_color = #orange
...
A = my_color == KUKA_color                 ;A=TRUE
END
```

11.13.4 Logische Operatoren

Beschreibung

Logische Operatoren dienen zur logischen Verknüpfung von boolschen Variablen, Konstanten und einfachen logischen Ausdrücken, wie sie mit Hilfe der Vergleichsoperatoren gebildet werden.

Operator	Anzahl der Operanden	Beschreibung
NOT	1	Invertierung
AND	2	logisches UND
OR	2	logisches ODER
EXOR	2	exklusives ODER

Die Operanden einer logischen Verknüpfung müssen vom Typ BOOL sein. Das Ergebnis ist ebenfalls immer vom Typ BOOL.

Die folgende Tabelle zeigt die Ergebnisse der möglichen Verknüpfungen:

Operation		NOT A	A AND B	A OR B	A EXOR B
A = TRUE	B = TRUE	FALSE	TRUE	TRUE	FALSE
A = TRUE	B = FALSE	FALSE	FALSE	TRUE	TRUE
A = FALSE	B = TRUE	TRUE	FALSE	TRUE	TRUE
A = FALSE	B = FALSE	TRUE	FALSE	FALSE	FALSE

Die Tabelle gilt auch für Operationen mit Bit-Operatoren.

Beispiele

Auch mehrfache Verknüpfungen sind zulässig.

```
...
DECL BOOL A,B,C
...
A = TRUE                                ;A=TRUE
B = NOT A                               ;B=FALSE
C = (A AND B)OR NOT (B EXOR NOT A)      ;C=TRUE
A = NOT NOT C                           ;A=TRUE
...
```

11.13.5 Bit-Operatoren

Beschreibung

Die Bit-Operatoren verknüpfen ganze Zahlen miteinander, indem sie deren einzelne Bits logisch miteinander verknüpfen.

Die Ergebnisse der Verknüpfungen entsprechen denen der logischen Operatoren.

- Der Bit-Wert 1 entspricht TRUE.
- Der Bit-Wert 0 entspricht FALSE.

Operator	Anzahl der Operanden	Beschreibung
B_NOT	1	bitweise Invertierung
B_AND	2	bitweise UND-Verknüpfung
B_OR	2	bitweise ODER-Verknüpfung
B_EXOR	2	bitweise exklusive ODER-Verknüpfung

Die Bit-Operatoren können auf die Datentypen INT und CHAR angewendet werden.

INT hat in KRL 32 Bit und ist vorzeichenbehaftet. CHAR hat 8 Bit und ist nicht vorzeichenbehaftet.

Um die Ergebnisse von Bit-Operationen nachvollziehen zu können, muss man berücksichtigen, dass die Robotersteuerung vorzeichenbehaftete Binärzahlen als Zweierkomplement interpretiert. Hierbei bestimmt das höchstwertige Bit, ob die Zahl positiv oder negativ ist. Man muss deshalb alle Bits berücksichtigen.

Bei den folgenden Beispielen zu B_AND, B_OR und B_EXOR mit Integer-Werten ergeben sich positive Zahlen (höchstwertiges Bit = 0). Die Ergebnisse können direkt ins Dezimalsystem umgerechnet werden, so wie bei vorzeichenlosen Werten.

Die 28 führenden Nullen der Operanden werden durch "0 0 […]" angedeutet.

B_AND

	0	0	[…]	0	1	0	1	= 5
	0	0	[…]	1	1	0	0	= 12
B_AND	**0**	**0**	**[…]**	**0**	**1**	**0**	**0**	**= 4**

Abb. 11-21: Beispiel: Verknüpfung der Integer-Werte 5 und 12

B_OR

	0	0	[…]	0	1	0	1	= 5
	0	0	[…]	1	1	0	0	= 12
B_OR	**0**	**0**	**[…]**	**1**	**1**	**0**	**1**	**= 13**

Abb. 11-22: Beispiel: Verknüpfung der Integer-Werte 5 und 12

B_EXOR

	0	0	[…]	0	1	0	1	= 5
	0	0	[…]	1	1	0	0	= 12
B_EXOR	**0**	**0**	**[…]**	**1**	**0**	**0**	**1**	**= 9**

Abb. 11-23: Beispiel: Verknüpfung der Integer-Werte 5 und 12

B_NOT

Bei diesem Integer-Beispiel ergibt sich durch die Operation eine negative Zahl (höchstwertiges Bit = 1). Das Ergebnis kann deshalb nicht auf die gleiche Weise wie eine vorzeichenlose Zahl ins Dezimalsystem umgerechnet werden.

Damit der Benutzer das dezimale Ergebnis, zu dem die Robotersteuerung kommt, nachvollziehen kann, muss er die Regeln für die Interpretation von Zweierkomplement-Zahlen kennen. Die Regeln sind nicht Gegenstand dieser Dokumentation.

	0	0	[…]	1	0	1	0	= 10
B_NOT	**1**	**1**	**[…]**	**0**	**1**	**0**	**1**	**= -11**

Abb. 11-24: Beispiel: B_NOT mit Integer-Wert 10

Das dezimale Ergebnis einer B_NOT-Operation auf einen vorzeichenbehafteten Operanden kann man auch folgendermaßen ermitteln:

1. Dezimalwert des Operanden plus 1
2. Vorzeichen umdrehen

Weitere Beispiele

```
...
DECL INT A
...
A = 10 B_AND 9                          ;A=8
A = 10 B_OR 9                           ;A=11
A = 10 B_EXOR 9                         ;A=3
A = B_NOT 197                           ;A=-198
A = B_NOT 'HC5'                         ;A=-198
A = B_NOT 'B11000101'                   ;A=-198
A = B_NOT "E"                           ;A=154
...
```

Bits setzen und Bits prüfen:

Mit B_AND und B_OR kann man gezielt einzelne Bits einer Bitfolge auf 1 oder 0 setzen. Die restlichen Bits bleiben unverändert.

- Mit B_AND kann man einzelne Bits auf 0 setzen.
- Mit B_OR kann man einzelne Bits auf 1 setzen.

Außerdem kann man einzelne Bits prüfen, ob sie 1 oder 0 sind.

Beispiel:

Es existiert ein digitaler Ausgang mit einer Breite von 8 Bit. Der Ausgang kann über die INT-Variable DIG angesprochen werden.

Bits 1, 2 und 6 auf 0 setzen:

```
DIG = DIG B_AND 'B10111001'
```

Bits 0, 2, 3 und 7 auf 1 setzen:

```
DIG = DIG B_OR 'B10001101'
```

Prüfen, ob die Bits 0 und 7 auf 1 gesetzt sind. Wenn ja, wird `my_result` TRUE:

```
DECL BOOL my_result
...
my_result = DIG B_AND ('B10000001') == 'B10000001'
```

Prüfen, ob eines der beiden Bits 0 oder 7 auf 1 gesetzt ist. Wenn ja, wird `my_result` TRUE:

```
DECL BOOL my_result
...
my_result = DIG B_AND ('B10000001') > 0
```

11.13.6 Priorität der Operatoren

Die Priorität gibt die Reihenfolge an, in der die Operatoren innerhalb einer Anweisung ausgewertet werden.

Priorität	Operator
1	NOT; B_NOT
2	*; /
3	+; -
4	AND; B_AND
5	EXOR; B_EXOR

Priorität	Operator
6	OR; B_OR
7	==, <>; <, >, <=, >=

Grundsätzlich gilt:

- Geklammerte Ausdrücke werden zuerst bearbeitet.
- Ungeklammerte Ausdrücke werden gemäß ihrer Priorität ausgewertet.
- Verknüpfungen mit Operatoren gleicher Priorität werden von links nach rechts ausgewertet.

11.14 Systemfunktionen

11.14.1 DELETE_BACKWARD_BUFFER()

Beschreibung

Über `DELETE_BACKWARD_BUFFER()` kann man das Rückwärtsfahren für bestimmte Bewegungen gezielt verhindern. Die Funktion löscht die aufgezeichneten Vorwärtsbewegungen. Wenn der Benutzer versucht, rückwärts zu fahren, gibt die Robotersteuerung folgende Meldung aus: *Rückwärtsfahren nicht möglich: keine Aufzeichnung vorhanden*.

Die Funktion kann mit allen Interpretern verwendet werden. Sie kann auch im Trigger verwendet werden.

Die Funktion löst einen Vorlaufstopp aus.

`DELETE_BACKWARD_BUFFER()` bezieht sich auf das Rückwärtsfahren über die Start-Rückwärts-Taste. Auf andere Rückwärts-Funktionalitäten, z. B. auf Rückwärtsbewegungen im Rahmen von Fehlerstrategien in Technologiepaketen, wirkt es sich nicht aus.

Beispiel 1

Im folgenden Beispiel ist es während der Bewegung von P4 nach P5 noch möglich, das Rückwärtsfahren zu starten. An P5 ist es nicht mehr möglich:

```
PTP P4
PTP P5
DELETE_BACKWARD_BUFFER()
```

Wenn gewünscht ist, dass es bereits während der Bewegung von P4 nach P5 nicht mehr möglich sein soll, rückwärts zu fahren, kann man dies über einen Trigger erreichen:

```
PTP P4
TRIGGER WHEN DISTANCE=0 DELAY=0 do DELETE_BACKWARD_BUFFER() PRIO=-1
PTP P5
```

Beispiel 2

Im folgenden Beispiel sind folgende Aktionen möglich bzw. nicht möglich:

Aktion	Möglich?
Rückwärts nach P6 fahren	Ja
Rückwärts nach P5 fahren	Nein
Von P5 aus rückwärts fahren	Ja

```
PTP P3
PTP P4
PTP P5
TRIGGER WHEN DISTANCE=0 DELAY=0 do DELETE_BACKWARD_BUFFER() PRIO=-1
PTP P6
PTP P7
```

Wenn beispielsweise gewünscht ist, dass ab dem Verlassen von P3 bis einschließlich P6 kein Rückwärtsfahren möglich sein soll, müssen mehrere Trigger programmiert werden:

```
PTP P3
TRIGGER WHEN DISTANCE=0 DELAY=0 do DELETE_BACKWARD_BUFFER() PRIO=-1
PTP P4
TRIGGER WHEN DISTANCE=0 DELAY=0 do DELETE_BACKWARD_BUFFER() PRIO=-1
PTP P5
TRIGGER WHEN DISTANCE=0 DELAY=0 do DELETE_BACKWARD_BUFFER() PRIO=-1
PTP P6
PTP P7
```

11.14.2 ROB_STOP() und ROB_STOP_RELEASE()

Beschreibung

ROB_STOP() und ROB_STOP_RELEASE() können nur in Submit-Programmen verwendet werden.

- ROB_STOP() stoppt den Roboter und verhindert weitere Bewegungen. Dies betrifft sämtliche möglichen Bewegungen, seien es Bewegungen aufgrund von Programmablauf, manuellem Verfahren oder Kommandobetrieb.
- ROB_STOP_RELEASE() hebt eine durch ROB_STOP() verursachte Blockade wieder auf.

Wenn ROB_STOP() mehrfach hintereinander aufgerufen wird, ohne dass zwischendurch ROB_STOP_RELEASE() aufgerufen wurde, so wirkt sich nur der erste Aufruf aus. Die weiteren Aufrufe haben keine Auswirkung, d. h. sie lösen weder einen Stopp noch eine Meldung aus.

Wenn ROB_STOP_RELEASE() aufgerufen wird, ohne dass zuvor ROB_STOP() aufgerufen wurde, so hat dies keine Auswirkung.

Meldungen

ROB_STOP() löst folgende Zustandsmeldung aus: *Roboter-Halt durch Submit*

ROB_STOP_RELEASE() in einer Test-Betriebsart löst folgende Quittiermeldung aus: *Quitt. Roboter-Halt durch Submit*

ROB_STOP_RELEASE() in einer Automatik-Betriebsart löst keine Meldung aus.

Syntax

result = ROB_STOP (*stop_type*: IN)

Erläuterung der Syntax

Element	Beschreibung
result	Typ: BOOL Variable für den Rückgabewert. Rückgabewert: ■ TRUE: Der Stopp wurde ausgeführt. ■ FALSE: Für *stop_type* wurde ein ungültiger Parameter übergeben.
stop_type	Typ: ROB_STOP_T Stopptyp, mit dem der Roboter angehalten werden soll: ■ #RAMP_DOWN: Rampenstopp ■ #PATH_MAINTAINING: Bahntreuer NOT-HALT Andere Stopptypen sind nicht möglich.

ProConOS

Die Funktionalität "Roboterstopp" kann auch aus ProConOS heraus verwendet werden. Hierfür stehen folgende Funktionen zur Verfügung:

- PLC_ROB_STOP()

  Der gewünschte Stopptyp wird festgelegt über PLC_ROB_STOP_RAMP_DOWN oder PLC_ROB_STOP_PATH_MAINT.
- PLC_ROB_STOP_RELEASE()

Die Auswirkung ist gleich, egal ob ein Stopp über Submit oder über ProConOS ausgelöst wird. Von ProConOS kommen eigene Meldungstexte; diese lauten:

- *Roboter-Halt durch SoftSPS ({Name der aufrufenden Task})*
- *Quitt. Roboter-Halt durch SoftSPS*

Wenn sowohl über Submit als auch durch ProConOS ein Stopp angefordert wird, so wird dies jeweils mit einer Zustandsmeldung angezeigt. Es können also (maximal) 2 Zustandsmeldungen für einen ausgeführten Stopp angezeigt werden. In diesem Fall kann der Roboter erst wieder bewegt werden, wenn die Blockade sowohl vom Submit aus als auch von ProConOS aus aufgehoben wurde.

Wenn vom Submit und von ProConOS unterschiedliche Stopptypen angefordert wurden, richtet sich der tatsächlich ausgeführte Typ in der Regel nach der ersten Anforderung.

Ein durch ProConOS ausgelöster Stopp kann nicht über ein Submit-Programm zurückgenommen werden und umgekehrt.

11.14.3 SET_BRAKE_DELAY()

Beschreibung

Über die Funktion SET_BRAKE_DELAY kann man die Bremsverzögerung reduzieren, und zwar bezogen auf einen einzelnen Punkt.

SET_BRAKE_DELAY ist gedacht für die Verwendung am Ende eines Zyklus: Wenn der Roboter dort stehenbleibt, bevor der nächste Zyklus beginnt, kann durch SET_BRAKE_DELAY erreicht werden, dass die Bremsen früher schließen und dass somit auch die Antriebe früher abschalten. Hierdurch kann Energie gespart werden.

Bremsverzögerung:

Die Bremsverzögerung ist die Zeit, nach der die Achsbremsen schließen, wenn der Roboter (oder die Zusatzachse) einen Genauhalt erreicht hat. Es spielt keine Rolle, ob der Genauhalt als solcher programmiert wurde oder ob er sich ergibt, weil nicht überschliffen werden kann.

Wenn der Roboter am Punkt stehenbleibt, bis die Zeit abgelaufen ist, z. B. am Programmende, schließen die Bremsen. Wenn der Roboter weiter verfährt, bevor die Zeit abgelaufen ist, schließen die Bremsen nicht.

Die allgemein gültige Bremsverzögerung ist in Systemvariablen definiert. Mit SET_BRAKE_DELAY kann man für einen einzelnen Punkt einen geringeren Wert definieren, d. h. man erreicht, dass die Bremsen früher schließen.

Systemvariablen für die allgemein gültige Bremsverzögerung:

- $BRK_DEL_COM:

  Bremsverzögerung für Roboterachsen im Kommando-Modus (= manuelles Verfahren) (Default: 10 000 ms)
- $BRK_DEL_PRO:

  Bremsverzögerung für Roboterachsen im Programm-Modus (Default: 20 000 ms)
- $BRK_DEL_EX:

  Bremsverzögerung für Zusatzachsen (Default: 200 ms)

  $BRK_DEL_EX ist nur wirksam, wenn der Zusatzachsen-Modus gesetzt ist ($BRK_MODE, Bit 3 =1) und wenn die Zusatzachse nicht mathema-

tisch gekoppelt ist. Anderenfalls verhalten sich die Bremsen der Zusatzachse wie die der Roboterachsen, und es gelten auch deren Verzögerungszeiten.

Weitere Informationen zu $BRK_MODE sind in der Dokumentation **Konfiguration von Kinematiken** zu finden.

Weitere Eigenschaften

SET_BRAKE_DELAY löst einen Vorlaufstopp aus. Der Vorlaufstopp gilt getrennt für synchrone und asynchrone Achsen: Wenn z. B. über *axes_nr* eine synchrone Achse angegeben wird, gilt der Vorlaufstopp für alle synchronen Achsen, jedoch nicht für eine eventuell vorhandene asynchrone Achse.

SET_BRAKE_DELAY kann von allen Interpretern bearbeitet werden.

SET_BRAKE_DELAY wirkt sich nur aus, wenn sich der Roboter an einem Genauhalt befindet:

- SET_BRAKE_DELAY muss im Programm auf den Punkt folgen, für den es gelten soll. Da es einen Vorlaufstopp auslöst, ist dieser Punkt automatisch ein Genauhalt.
- Wenn es über einen Trigger ausgelöst wird, kann es sich nur auswirken, wenn sich der Trigger auf den Zielpunkt bezieht und dieser ein Genauhalt ist.

Syntax

result = SET_BRAKE_DELAY (*axes_nr*, *delay*)

Erläuterung der Syntax

Element	Beschreibung
result	Typ: INT Variable für den Rückgabewert. Die Bits zeigen an, für welche Achsen *delay* gesetzt worden ist. ■ Bit n = 0: Wert wurde für diese Achse nicht gesetzt. ■ Bit n = 1: Wert wurde für diese Achse gesetzt. Der Rückgabewert sagt nichts darüber aus, ob die Bremsen tatsächlich geschlossen wurden.

Element	Beschreibung
axes_nr	Typ: INT Bit-Feld für die Achsen, für die *delay* gesetzt werden soll. ■ Bit n = 0: Wert wird für diese Achse nicht gesetzt. ■ Bit n = 1: Wert wird für diese Achse gesetzt. Der Wert kann im Programm als Integer oder in Bit-Schreibweise angegeben werden, z. B. "63" oder " 'B111111' " für "alle Roboterachsen". Die Bremsen von Roboterachsen werden defaultmäßig gemeinsam geschlossen. Der Wert gilt in diesem Fall für alle Roboterachsen, auch die, die hier nicht angegeben sind. Wenn die Bremsen von Zusatzachsen gemeinsam geschlossen werden (abhängig von $BRK_MODE), dann gilt der Wert für alle Zusatzachsen, auch die, die hier nicht angegeben sind. Die Bremsen von Master-Slave-Achsen werden immer gemeinsam geschlossen.
delay	Typ: INT, Einheit: ms Gewünschte Verzögerung. Wertebereich: ■ **0** … definierte allgemeine Bremsverzögerung Wenn ein höherer Wert definiert ist, wird er intern gekappt auf den Wert der relevanten Systemvariable: $BRK_DEL_COM, $BRK_DEL_PRO oder $BRK_DEL_EX Der Wert 0 ist zulässig. Die tatsächliche Schließzeit ist jedoch immer mindestens so lange wie die mechanisch bedingte Schließzeit der Bremse. Diese beträgt wenige Sekundenbruchteile. Der genaue Wert ist abhängig von der einzelnen Achse.

Bit n	11 …	5	4	3	2	1	0
Achse	E6 …	A6	A5	A4	A3	A2	A1

$BRAKE_SIG

Der Zustand der Bremsen (offen oder geschlossen) kann über die Systemvariable $BRAKE_SIG angezeigt werden.

(>>> "$BRAKE_SIG" Seite 220)

Beispiel

Beim folgenden Programm sollen sich die Bremsen der Roboterachsen am Ende so früh wie möglich schließen. Nach dem letzten Punkt wurde deshalb SET_BRAKE_DELAY(63, 0) programmiert.

```
1  DEF my_test()
2  DECL INT my_result
3  DECL INT brake_state
   ...
4  PTP HOME Vel= 100 % DEFAULT
5  PTP P1 ...
   ...
6  PTP HOME Vel= 100 % DEFAULT
7  my_result = SET_BRAKE_DELAY(63, 0)
8  brake_state = $BRAKE_SIG
9  END
```

Zeile	Beschreibung
6	Letzter Punkt im Programm
7	Hier wird die Bremsverzögerung für den Punkt aus Zeile 6 für alle Roboterachsen auf 0 ms gesetzt.
8	Die Abfrage ergibt, dass die Bremsen an dieser Stelle (noch) offen sind. Der Grund ist, dass die Bremsen sich nicht in 0 ms schließen können, sondern sie benötigen die mechanisch bedingte Zeit. Nach Ablauf dieser Zeit sind die Bremsen geschlossen.

Negativ-Beispiel

In der Regel ist es nicht sinnvoll, SET_BRAKE_DELAY während eines Zyklus zu verwenden. Es gibt oft keine Punkte, an denen die Bremsen einfallen und wo dieser Vorgang beschleunigt werden müsste. Der Vorlaufstopp, den SET_BRAKE_DELAY auslöst, wirkt sich im Gegenteil sogar negativ auf die Taktzeit aus.

```
1  DEF my_test()
2  DECL INT my_result
   ...
3  PTP HOME Vel= 100 % DEFAULT
4  PTP P1 C_DIS ...
5  my_result = SET_BRAKE_DELAY(63, 0)
6  ;WAIT SEC 0.5
7  PTP P2 ...
   ...
```

Zeile	Beschreibung
5	Hier wird für P1 die Bremsverzögerung für alle Roboterachsen auf 0 ms gesetzt. P1 ist mit Überschleif programmiert. Da SET_BRAKE_DELAY einen Vorlaufstopp auslöst, wird P1 genau angefahren. Die Bremsen schließen sich an P1 jedoch nicht. Der Grund ist, dass die Bremsen zum Schließen die mechanisch bedingte Zeit benötigen würden. Die Robotersteuerung beginnt jedoch sofort beim Erreichen von P1 mit der nächsten Bewegung. Es kommt also nicht zum Schließen der Bremsen, obwohl die Verzögerung auf 0 ms gesetzt ist.
6	Zum Vergleich: Wenn diese Zeile einkommentiert würde, würden sich die Bremsen schließen. Die Robotersteuerung würde nicht sofort beim Erreichen von P1 mit der nächsten Bewegung beginnen, sondern 0.5 s an P1 stehenbleiben. Dies würde Zeit zum Schließen der Bremsen lassen.

11.14.4 VARSTATE()

Beschreibung

Mit VARSTATE() kann der Status einer Variablen abgefragt werden.

VARSTATE() ist eine Funktion mit einem Rückgabewert vom Typ VAR_STATE. VAR_STATE ist ein Aufzählungstyp, der im System folgendermaßen definiert ist:

```
ENUM VAR_STATE DECLARED, INITIALIZED, UNKNOWN
```

VARSTATE ist im System folgendermaßen definiert:

```
VAR_STATE VARSTATE(CHAR VAR_STR[80]:IN)
```

Beispiel 1

```
DEF PROG1()
INT MYVAR
...
IF VARSTATE("MYVAR")==#UNKNOWN THEN
   $OUT[11]=TRUE
ENDIF
...
IF VARSTATE("MYVAR")==#DECLARED THEN
   $OUT[12]=TRUE
ENDIF
...
IF VARSTATE("ANYVAR")==#UNKNOWN THEN
   $OUT[13]=TRUE
ENDIF
...
MYVAR=9
...
IF VARSTATE("MYVAR")==#DECLARED THEN
   $OUT[14]=TRUE
ENDIF
...
IF VARSTATE("MYVAR")==#INITIALIZED THEN
   $OUT[15]=TRUE
ENDIF
...
END
```

Erläuterung der Status-Abfragen:

- Die erste IF-Bedingung ist falsch, da MYVAR bereits deklariert ist. Der Ausgang 11 wird nicht gesetzt.
- Die zweite IF-Bedingung ist wahr, da MYVAR deklariert ist. Der Ausgang 12 wird gesetzt.
- Die dritte IF-Bedingung ist wahr, unter der Voraussetzung, dass auch in der $CONFIG.DAT keine Variable mit dem Namen ANYVAR existiert. Der Ausgang 13 wird gesetzt.
- Die vierte IF-Bedingung ist falsch, da MYVAR nicht nur deklariert, sondern an dieser Stelle auch bereits initialisiert ist. Der Ausgang 14 wird nicht gesetzt.
- Die fünfte IF-Bedingung ist wahr, da MYVAR initialisiert ist. Der Ausgang 15 wird gesetzt.

Beispiel 2

```
DEF PROG2()
INT MYVAR
INT YOURVAR
DECL VAR_STATE STATUS
...
STATUS=VARSTATE("MYVAR")
UP()
...
STATUS=VARSTATE("YOURVAR")
UP()
...
END
```

```
DEF UP()
...
IF VARSTATE("STATUS")==#DECLARED THEN
   $OUT[100]=TRUE
ENDIF
```

```
...
END
```

Erläuterung der Status-Abfrage:

In diesem Beispiel wird der Status indirekt abgefragt, d. h. über eine zusätzliche Variable. Die zusätzliche Variable muss vom Typ VAR_STATE sein. Bei der Deklaration darf das Schlüsselwort DECL nicht weggelassen werden. Der Name der zusätzlichen Variable ist beliebig. In diesem Beispiel lautet er STATUS.

11.15 Stringvariablen bearbeiten

Zur Bearbeitung von Stringvariablen stehen verschiedene Funktionen zur Verfügung. Die Funktionen können in SRC-Dateien, in SUB-Dateien und in der Variablenkorrektur verwendet werden.

Innerhalb von IF-Verzweigungen können die Funktionen verwendet werden, ohne dass der Rückgabewert explizit einer Variablen zugewiesen wird.

11.15.1 Länge einer Stringvariablen bei der Deklaration

Beschreibung

Die Funktion `StrDeclLen()` ermittelt die Länge einer Stringvariablen, entsprechend deren Deklaration im Deklarationsteil eines Programms.

Syntax

Length = `StrDeclLen(`*StrVar[]*`)`

Erläuterung der Syntax

Element	Beschreibung
Length	Typ: INT Variable für den Rückgabewert. Rückgabewert: Länge der Stringvariablen, wie im Deklarationsteil vereinbart
StrVar[]	Typ: CHAR-Feld Stringvariable, deren Länge ermittelt werden soll Da die Stringvariable StrVar[ ] ein Feld vom Typ CHAR ist, sind zur Längenermittlung einzelne Zeichen sowie Konstanten nicht zulässig.

Beispiel

```
1  CHAR ProName[24]
2  INT StrLength
   …
3  StrLength = StrDeclLen(ProName)
4  StrLength = StrDeclLen($Trace.Name[ ])
```

Zeile	Beschreibung
3	StrLength = 24
4	StrLength = 64

11.15.2 Länge einer Stringvariablen nach der Initialisierung

Beschreibung

Die Funktion `StrLen()` ermittelt die Länge der Zeichenfolge einer Stringvariablen, wie sie im Initialisierungsteil des Programms festgelegt wurde.

Syntax

Length = `StrLen(`*StrVar*`)`

Erläuterung der Syntax

Element	Beschreibung
Length	Typ: INT Variable für den Rückgabewert. Rückgabewert: Anzahl der Zeichen, welche momentan der Stringvariablen zugewiesen sind
StrVar	Typ: CHAR Zeichenfolge oder Variable, deren Länge ermittelt werden soll

Beispiel

```
1  CHAR PartA[50]
2  INT AB
   …
3  PartA[] = "This is an example"
4  AB = StrLen(PartA[])
```

Zeile	Beschreibung
4	AB = 18

11.15.3 Inhalt einer Stringvariablen löschen

Beschreibung Die Funktion `StrClear()` löscht den Inhalt einer Stringvariablen.

Syntax *Result* = `StrClear`*(StrVar[])*

Erläuterung der Syntax

Element	Beschreibung
Result	Typ: BOOL Variable für den Rückgabewert. Rückgabewert: ■ Der Inhalt der Stringvariablen wurde gelöscht: TRUE ■ Der Inhalt der Stringvariablen wurde nicht gelöscht: FALSE
StrVar[]	Typ: CHAR-Feld Variable, deren Zeichenfolge gelöscht werden soll

Beispiel

```
IF (NOT StrClear($Loop_Msg[])) THEN
HALT
ENDIF
```

Innerhalb von IF-Verzweigungen kann die Funktion verwendet werden, ohne dass der Rückgabewert explizit einer Variablen zugewiesen wird. Dies gilt für alle Funktionen zur Bearbeitung von Stringvariablen.

11.15.4 Stringvariable erweitern

Beschreibung Mit der Funktion `StrAdd()` kann eine Stringvariable mit dem Inhalt einer weiteren Stringvariablen erweitert werden.

Syntax *Sum* = `StrAdd(`*StrDest[], StrToAdd[]*`)`

Erläuterung der Syntax

Element	Beschreibung
Sum	Typ: INT Variable für den Rückgabewert. Rückgabewert: Summe von StrDest[ ] und StrToAdd[ ] Wenn die Summe länger ist als die vorher definierte Länge von *StrDest[ ]*, ist der Rückgabewert 0. Dies ist auch der Fall, wenn die Summe größer ist als 470 Zeichen.
StrDest[]	Typ: CHAR-Feld Die zu erweiternde Stringvariable Da die Stringvariable StrDest[ ] ein Feld vom Typ CHAR ist, sind einzelne Zeichen sowie Konstanten nicht zulässig.
StrToAdd[]	Typ: CHAR-Feld Die Zeichenfolge, mit welcher erweitert wird

Beispiel

```
1  DECL CHAR A[50], B[50]
2  INT AB, AC
   …
3  A[] = "This is an "
4  B[] = "example"
5  AB = StrAdd(A[],B[])
```

Zeile	Beschreibung
5	A[ ] = "This is an example" AB = 18

11.15.5 Stringvariable durchsuchen

Beschreibung

Mit der Funktion `StrFind()` kann eine Stringvariable nach einer Zeichenfolge durchsucht werden.

Syntax

Result = `StrFind(`*StartAt, StrVar[], StrFind[], CaseSens*`)`

Erläuterung der Syntax

Element	Beschreibung
Result	Typ: INT Variable für den Rückgabewert. Rückgabewert: Position des ersten gefundenen Zeichens. Wenn kein Zeichen gefunden wird, ist der Rückgabewert 0.
StartAt	Typ: INT Die Suche startet bei dieser Position.
StrVar[]	Typ: CHAR-Feld Die zu durchsuchende Stringvariable
StrFind[]	Typ: CHAR-Feld Nach dieser Zeichenfolge wird gesucht.
CaseSens	■ #CASE_SENS: Groß-/Kleinschreibung wird berücksichtigt. ■ #NOT_CASE_SENS: Groß-/Kleinschreibung wird nicht berücksichtigt.

Beispiel

```
1  DECL CHAR A[5]
2  INT B
3  A[]="ABCDE"
```

```
4  B = StrFind(1, A[], "AC", #CASE_SENS)
5  B = StrFind(1, A[], "a", #NOT_CASE_SENS)
6  B = StrFind(1, A[], "BC", #Case_Sens)
7  B = StrFind(1, A[], "bc", #NOT_CASE_SENS)
```

Zeile	Beschreibung
4	B = 0
5	B = 1
6	B = 2
7	B = 2

11.15.6 Inhalt von Stringvariablen vergleichen

Beschreibung Mit der Funktion `StrComp()` können zwei Stringvariablen verglichen werden.

Syntax *Comp* = `StrComp(`*StrComp1[], StrComp2[], CaseSens*`)`

Erläuterung der Syntax

Element	Beschreibung
Comp	Typ: BOOL Variable für den Rückgabewert. Rückgabewert: ■ Die Zeichenfolgen stimmen überein: TRUE ■ Die Zeichenfolgen stimmen nicht überein: FALSE
StrComp1[]	Typ: CHAR-Feld Stringvariable, die mit StrComp2[] verglichen wird
StrComp2[]	Typ: CHAR-Feld Stringvariable, die mit StrComp1[] verglichen wird
CaseSens	■ #CASE_SENS: Groß-/Kleinschreibung wird berücksichtigt. ■ #NOT_CASE_SENS: Groß-/Kleinschreibung wird nicht berücksichtigt.

Beispiel

```
1  DECL CHAR A[5]
2  BOOL B
3  A[]="ABCDE"
4  B = StrComp(A[], "ABCDE", #CASE_SENS)
5  B = StrComp(A[], "abcde", #NOT_CASE_SENS)
6  B = StrComp(A[], "abcd", #NOT_CASE_SENS)
7  B = StrComp(A[], "acbde", #NOT_CASE_SENS)
```

Zeile	Beschreibung
4	B = TRUE
5	B = TRUE
6	B = FALSE
7	B = FALSE

11.15.7 Stringvariable kopieren

Beschreibung Mit der Funktion `StrCopy()` kann der Inhalt einer Stringvariablen in eine andere Stringvariable kopiert werden.

Syntax *Copy* = `StrCopy(`*StrDest[], StrSource[]*`)`

Erläuterung der Syntax

Element	Beschreibung
Copy	Typ: BOOL Variable für den Rückgabewert. Rückgabewert: ■ Die Stringvariable wurde erfolgreich kopiert: TRUE ■ Die Stringvariable wurde nicht kopiert: FALSE
StrDest[]	Typ: CHAR-Feld Die Zeichenfolge wird in diese Stringvariable kopiert. Da StrDest[ ] ein Feld vom Typ CHAR ist, sind einzelne Zeichen sowie Konstanten nicht zulässig.
StrSource[]	Typ: CHAR-Feld Der Inhalt dieser Stringvariablen wird kopiert.

Beispiel

```
1  DECL CHAR A[25], B[25]
2  DECL BOOL C
3  A[] = ""
4  B[] = "Example"
5  C = StrCopy(A[], B[])
```

Zeile	Beschreibung
5	A[ ] = "Example" C = TRUE

12 Submit-Interpreter

12.1 Funktion des Submit-Interpreters

Funktion

Auf der Robotersteuerung laufen parallel 2 Tasks:

- Roboter-Interpreter
  Über den Roboter-Interpreter läuft das Bewegungsprogramm.
- Submit-Interpreter
  Über den Submit-Interpreter läuft ein SUB-Programm.
  Ein SUB-Programm kann Bedienungs- oder Überwachungsaufgaben erfüllen. Beispiele: Überwachung von Schutzeinrichtungen; Überwachung eines Kühlkreislaufs.
  Für kleinere Anwendungen ist somit keine SPS notwendig, da die Robotersteuerung solche Aufgaben mitübernehmen kann.

Der Submit-Interpreter startet beim Einschalten der Robotersteuerung automatisch. Dabei wird das Programm SPS.SUB gestartet.

Der Submit-Interpreter kann manuell gestoppt oder abgewählt werden und auch wieder gestartet werden.

SUB-Programme sind immer Dateien mit der Endung *.SUB. Das Programm SPS.SUB kann bearbeitet werden, und weitere SUB-Programme können erstellt werden.

⚠ WARNUNG Der Submit-Interpreter darf nicht für zeitkritische Anwendungen eingesetzt werden! Für solche Fälle muss eine SPS verwendet werden. Gründe:

- Der Submit-Interpreter teilt sich die Systemleistung mit dem Roboter-Interpreter, der die höhere Priorität hat. Der Submit-Interpreter wird deshalb nicht im Interpolationstakt der Robotersteuerung von 12 ms durchlaufen. Die Laufzeit des Submit-Interpreters ist darüber hinaus unregelmäßig.
- Die Laufzeit des Submit-Interpreters wird von der Anzahl der Zeilen im SUB-Programm beeinflusst. Auch Kommentarzeilen und Leerzeilen wirken sich aus.

Wenn eine Systemdatei, z. B. $config.dat oder $custom.dat, geändert wird und dadurch fehlerhaft wird, wird der Submit-Interpreter automatisch abgewählt. Wenn der Fehler in der Systemdatei behoben ist, muss der Submit-Interpreter manuell wieder angewählt werden.

Anzeige

Das Programm SPS.SUB befindet sich im Verzeichnis R1\System. Dieses Verzeichnis ist ab der Benutzergruppe Experte sichtbar.

Im Navigator sind SUB-Programme durch folgendes Symbol gekennzeichnet:

Ab der Benutzergruppe Experte wird außerdem die Dateierweiterung **sub** angezeigt.

Der Ablauf eines angewählten SUB-Programms wird defaultmäßig nicht angezeigt. Dies kann über die Systemvariable $INTERPRETER geändert werden. Das SUB-Programm kann jedoch nur angezeigt werden, wenn gleichzeitig ein Bewegungsprogramm angewählt ist.

$INTERPRETER	Beschreibung
1	Im Editor wird das angewählte Bewegungsprogramm angezeigt. (Default)
0	Im Editor wird das angewählte SUB-Programm angezeigt.

12.2 Submit-Interpreter manuell stoppen oder abwählen

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2

Vorgehensweise

- Im Hauptmenü **Konfiguration** > **SUBMIT Interpreter** > **Stoppen** oder **Abwählen** wählen.

Alternative Vorgehensweise

- In der Statusleiste die Statusanzeige **Submit-Interpreter** berühren. Ein Fenster öffnet sich.
  Stoppen oder **Abwählen** wählen.

Beschreibung

Befehl	Beschreibung
Stoppen	Der Submit-Interpreter wird gestoppt. Wenn er wieder gestartet wird, wird das SUB-Programm an der Stelle fortgesetzt, an der es gestoppt wurde.
Abwählen	Der Submit-Interpreter wird abgewählt.

Nach dem Stoppen oder Abwählen ist das Symbol für den Submit-Interpreter in der Statusleiste rot oder grau.

Symbol	Farbe	Beschreibung
	rot	Der Submit-Interpreter wurde gestoppt.
	grau	Der Submit-Interpreter ist abgewählt.

12.3 Submit-Interpreter manuell starten

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Submit-Interpreter wurde gestoppt oder abgewählt.

Vorgehensweise

- Im Hauptmenü **Konfiguration** > **SUBMIT Interpreter** > **Anwählen/Starten** wählen.

Alternative Vorgehensweise

- In der Statusleiste die Statusanzeige **Submit-Interpreter** berühren. Ein Fenster öffnet sich.
  Anwählen/Starten wählen.

Beschreibung

Wenn der Submit-Interpreter abgewählt ist, wählt der Befehl **Starten/Anwählen** das Programm SPS.SUB an.

Wenn der Submit-Interpreter gestoppt wurde, setzt der Befehl **Starten/Anwählen** das angewählte Programm an der Stelle fort, an der es gestoppt wurde.

Nach dem Start ist das Symbol für den Submit-Interpreter in der Statusleiste grün.

Symbol	Farbe	Beschreibung
	gelb	Der Submit-Interpreter ist angewählt. Der Satzzeiger steht auf der ersten Zeile des angewählten SUB-Programms.
	grün	Ein SUB-Programm ist angewählt und läuft.

12.4 Programm SPS.SUB bearbeiten

Beschreibung

Für benutzerspezifische Anpassungen stehen im Programm SPS.SUB folgende Folds zur Verfügung:

- USER INIT
- USER PLC

Andere Bereiche des Programms SPS.SUB dürfen vom Benutzer nicht verändert werden.

Wenn andere Bereiche in SPS.SUB geändert werden, kann dies die Funktionalität von Technologiepaketen beeinträchtigen.

Voraussetzung

- Das Programm SPS.SUB ist nicht angewählt oder wurde gestoppt.
- Benutzergruppe Experte

Vorgehensweise

1. Das Programm SPS.SUB im Navigator markieren und **Öffnen** drücken.
2. Die Änderungen eintragen:
   - Initialisierungen in den Fold USER INIT eintragen. Dieser Fold befindet sich im Fold INI.

```
USER INIT
  ; Please insert user defined initialization commands
```

   - Alle anderen Änderungen in den Fold USER PLC eintragen.

```
USER PLC
    ; Make your modifications here
```

3. Das Programm schließen. Die Sicherheitsabfrage, ob die Änderungen gespeichert werden sollen, mit **Ja** beantworten.
4. Das Programm SPS.SUB kann jetzt gestartet werden über das Hauptmenü mit **Konfiguration** > **SUBMIT Interpreter** > **Anwählen/Starten**.

SPS.SUB

Struktur des Programms SPS.SUB:

```
 1  DEF  SPS ( )
 2    DECLARATIONS
 3    INI
 4
 5    LOOP
 6       WAIT FOR NOT($POWER_FAIL)
 7       TORQUE_MONITORING()
 8
 9      ATB PLC LOOP
10      USER PLC
11    ENDLOOP
```

Zeile	Beschreibung
3	INI-Fold Dieser Fold enthält den Fold USER INIT: Hier kann der Benutzer Anweisungen eintragen, die nach dem Hochlauf nur einmal ausgeführt werden sollen.
5 … 10	LOOP-Anweisung. Für Programme, die dauerhaft im Hintergrund laufen sollen.
9	Manche Software-Optionen fügen in das Programm SPS.SUB Folds ein. Beispiel: KUKA.ArcTech Basic fügt den Fold `ATB PLC LOOP` ein. Welche Folds tatsächlich vorhanden sind, hängt davon ab, welche Optionen auf der Robotersteuerungen installiert sind.
10	USER PLC: Hier kann der Benutzer Anweisungen eintragen, die im LOOP ausgeführt werden sollen.

12.5 Neues SUB-Programm anlegen

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. In der Dateiliste den Ordner markieren, in dem das Programm angelegt werden soll. (Nicht in allen Ordnern können Programme angelegt werden.)
2. Auf die Schaltfläche **Neu** drücken.
   Das Fenster **Template Auswahl** öffnet sich.
3. Das Template **Submit** oder **Expert Submit** markieren und mit **OK** bestätigen.
4. Einen Namen für das Programm eingeben und mit **OK** bestätigen.

Beschreibung

Template "Submit":

Das Template **Submit** erzeugt eine SUB-Datei mit folgendem Aufbau:

```
1  DECLARATIONS
2  INI
3
4  LOOP
5  USER PLC
6  ENDLOOP
7  USER SUBROUTINE
```

Zeile	Beschreibung
1	Deklarationsteil
2	Initialisierungsteil. Für Anweisungen, die nach dem Hochlauf nur einmal ausgeführt werden sollen.
4, 5, 6	LOOP-Anweisung, die den Fold USER PLC enthält. USER PLC ist für Programme bestimmt, die dauerhaft im Hintergrund laufen sollen.
7	Für benutzerspezifische Subroutinen

Template "Expert Submit":

Das Template **Expert Submit** erzeugt eine leere SUB-Datei. Bei diesem Template muss alles vom Benutzer selbst programmiert werden.

Beim Programmieren eine LOOP-Anweisung verwenden. SUB-Programme ohne LOOP-Anweisung durchläuft der Submit-Interpreter nur einmal. Danach wird er automatisch abgewählt.

12.6 Programmierung

KRL-Code

In einem SUB-Programm können fast alle KRL-Anweisungen verwendet werden. Nicht möglich sind jedoch folgende Anweisungen:

- Anweisungen für Roboterbewegungen

  Roboterbewegungen können nur vom Roboter-Interpreter interpretiert werden. Aus diesem Grund können von einem SUB-Programm auch keine SRC-Programme, die Bewegungsanweisungen enthalten, als Unterprogramme aufgerufen werden.
- Anweisungen, die sich auf Roboterbewegungen beziehen

  Hierzu gehören BRAKE und alle TRIGGER.

Bewegungsanweisungen für Zusatzachsen können in einem SUB-Programm verwendet werden. Beispiel:

```
IF (($IN[12] == TRUE) AND ( NOT $IN[13] == TRUE)) THEN
ASYPTP {E2 45}
ASYPTP {E3 200}
...
IF ((NOT $IN[12] == TRUE) AND  ($IN[13] == TRUE)) THEN
ASYPTP {E2 0}
ASYPTP {E3 90}
```

Die Zusatzachsen E2 und E3 werden in Abhängigkeit von bestimmten Eingängen verfahren.

WAIT-Anweisungen oder Warteschleifen wurden hier nicht verwendet, da sie den Zyklus aufhalten.

Systemvariablen

Der Submit-Interpreter kann lesend auf alle Systemvariablen zugreifen, auf viele auch schreibend. Der Zugriff funktioniert auch dann, wenn die Systemvariablen parallel von einem Bewegungsprogramm benutzt werden.

Wenn in einem SUB-Programm eine Systemvariable geändert wird, auf die der Submit-Interpreter keinen Schreibzugriff hat, erscheint beim Start des Programms eine Fehlermeldung und der Submit-Interpreter stoppt.

In SUB-Programmen häufig benötigte Systemvariablen:

$MODE_OP = *Wert*	
Wert	**Beschreibung**
#T1	Robotersteuerung ist in Betriebsart T1.
#T2	Robotersteuerung ist in Betriebsart T2.
#AUT	Robotersteuerung ist in Betriebsart Automatik.
#EX	Robotersteuerung ist in Betriebsart Automatik Extern.
#INVALID	Robotersteuerung hat keinen definierten Zustand.

$OV_PRO = *Wert*		
Element	**Datentyp**	**Beschreibung**
Wert (%)	INT	Größe des Programmoverrides

Beispiel:

Wenn die programmierte Geschwindigkeit nicht erreicht wird, wird der Ausgang 2 auf FALSE gesetzt.

```
…
IF (($MODE_OP == #T1) OR ($OV_PRO < 100)) THEN
$OUT[2] = FALSE
```

```
ENDIF
…
```

⚠ WARNUNG In den Test-Betriebsarten darf $OV_PRO vom Submit-Interpreter aus nicht beschrieben werden, weil die Änderung für Benutzer, die am Industrieroboter arbeiten, unerwartet sein kann. Tod, Verletzungen oder Sachschäden können die Folge sein.

⚠ WARNUNG Sicherheitsrelevante Signale und Variablen (z. B. Betriebsart, NOT-HALT, Schutztürkontakt) möglichst über den Submit-Interpreter nicht ändern.
Wenn dennoch Änderungen notwendig sind, müssen alle sicherheitsrelevanten Signale und Variablen so verknüpft werden, dass sie vom Submit-Interpreter oder der SPS nicht in einen sicherheitsgefährdenden Zustand gesetzt werden können.

Ein-/Ausgänge

Der Submit-Interpreter kann auf die Ein- und Ausgänge der Robotersteuerung zugreifen.

⚠ WARNUNG Es wird nicht geprüft, ob der Roboter- und der Submit-Interpreter gleichzeitig auf denselben Ausgang zugreifen, da dies in bestimmten Fällen auch erwünscht sein kann.
Die Zuweisung der Ausgänge muss deshalb vom Benutzer sorgfältig geprüft werden. Anderenfalls kann es zu unerwarteten Ausgangssignalen kommen, z. B. an Sicherheitseinrichtungen. Tod, schwere Verletzungen oder erheblicher Sachschaden können die Folge sein.

Unterprogramme

In einem SUB-Programm können andere Programme als Unterprogramme aufgerufen werden. Möglich sind:

- Andere SUB-Programme
- SRC-Programme ohne Anweisungen für Roboterbewegungen

Beispiel:

Vom Programm SPS.SUB aus kann mit einer CWRITE-Anweisung und RUN das Programm CELL.SRC aufgerufen werden. Der Aufruf wirkt sich nur bei einem Kaltstart aus.

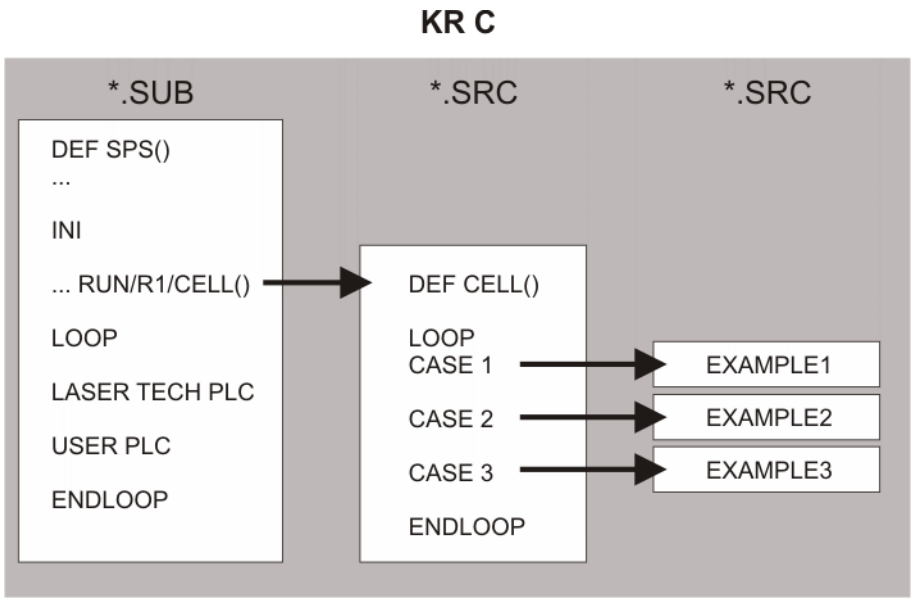


Abb. 12-1: SPS.SUB wählt CELL.SRC im Roboter-Interpreter an

Weitere Informationen zum Programm CELL.SRC sind in dieser Dokumentation zu finden.
(>>> 6.17.1 "CELL.SRC konfigurieren" Seite 196)
Weitere Informationen zu CWRITE-Anweisungen sind in der Experten-Dokumentation CREAD/CWRITE zu finden.

Kommunikation

Um einen binären Informationsaustausch zwischen einem laufenden Bewegungsprogramm und einem SUB-Programm zu ermöglichen, kann man die Flags der Robotersteuerung verwenden. Ein Flag wird mit dem Submit-Interpreter gesetzt und vom Roboter-Interpreter ausgelesen.

13 Diagnose

13.1 Logbuch

13.1.1 Logbuch anzeigen

Die Bedienhandlungen des Benutzers am smartPAD werden automatisch protokolliert.

Vorgehensweise

- Im Hauptmenü **Diagnose** > **Logbuch** > **Anzeigen** wählen.

Folgende Registerkarten stehen zur Verfügung:

- Log (>>> 13.1.2 "Registerkarte Log" Seite 477)
- Filter (>>> 13.1.3 "Registerkarte Filter" Seite 478)

13.1.2 Registerkarte Log

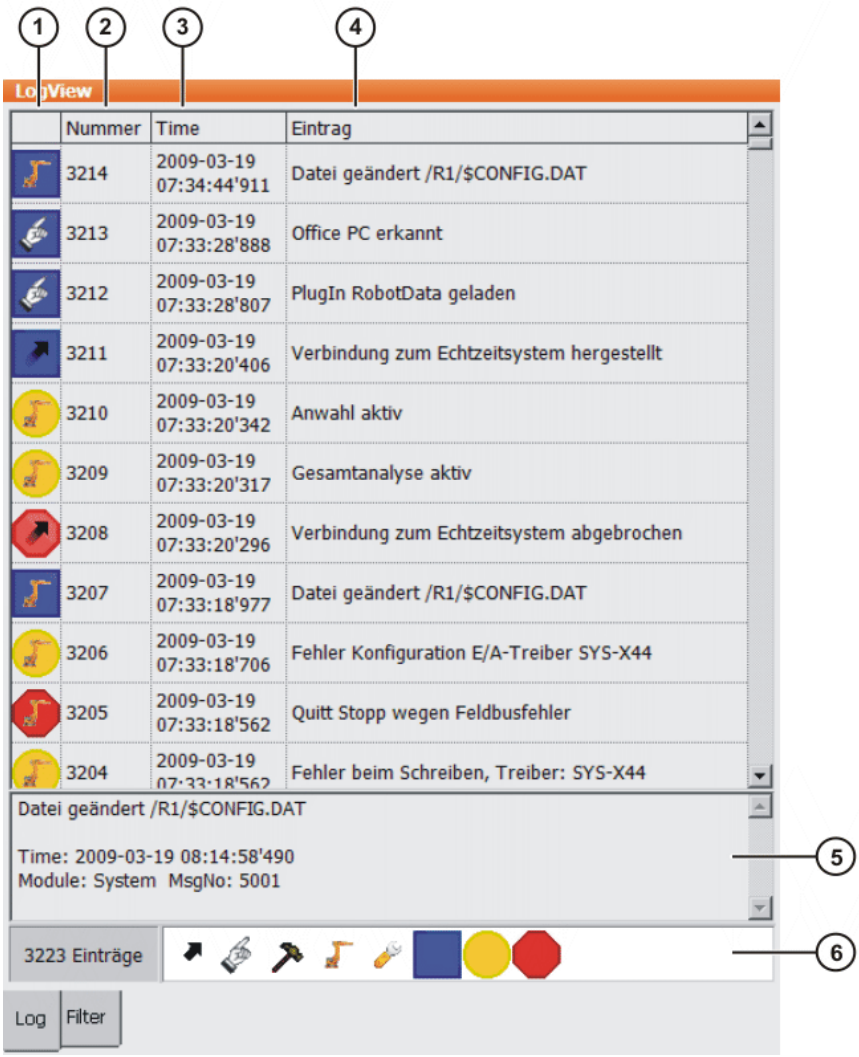


Abb. 13-1: Logbuch, Registerkarte Log

Pos.	Beschreibung
1	Art des Log-Ereignisses Beispiel  : Filtertyp "Hinweis" + Filterklasse "System" = Hinweis, der vom Grundsystem des Roboters kommt Die einzelnen Filtertypen und Filterklassen sind auf der Registerkarte **Filter** aufgelistet.
2	Nummer des Log-Ereignisses
3	Datum und Uhrzeit des Log-Ereignisses
4	Kurzbeschreibung des Log-Ereignisses
5	Detaillierte Beschreibung des markierten Log-Ereignisses
6	Anzeige der aktiven Filter

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Export	Exportiert die Log-Daten in eine Textdatei. (>>> 13.1.4 "Logbuch konfigurieren" Seite 479)
Aktual.	Aktualisiert die Log-Anzeige.

13.1.3 Registerkarte Filter

Die Registerkarte **Filter** wird nur angezeigt, wenn das Protokoll **Benutzerdefiniert** gewählt wurde.

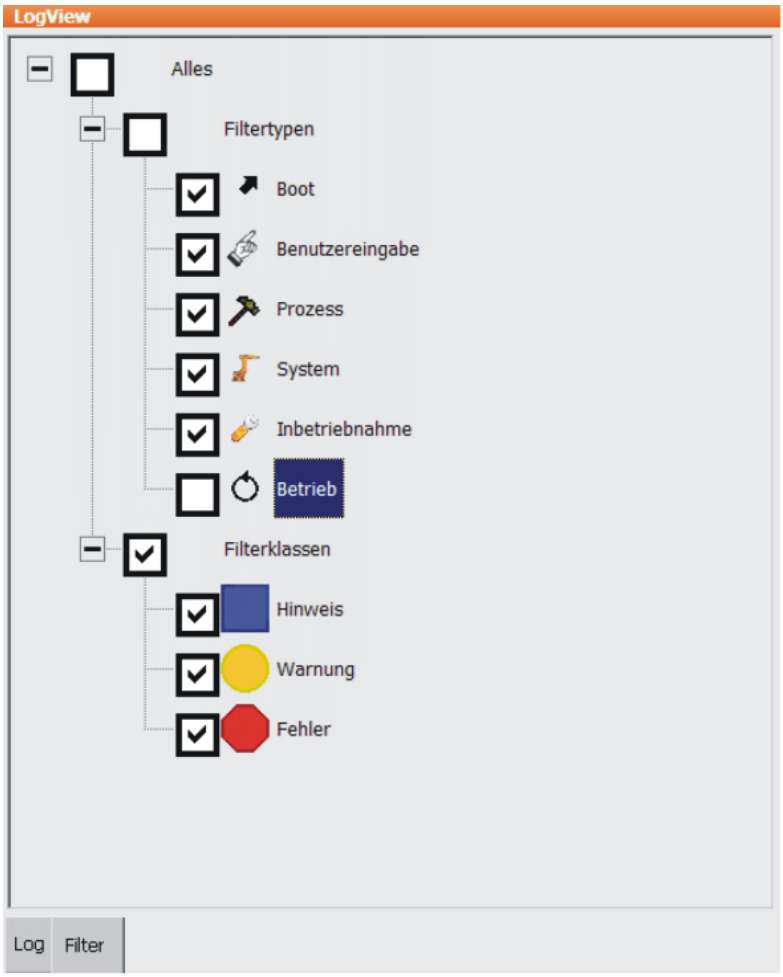


Abb. 13-2: Logbuch, Registerkarte Filter

13.1.4 Logbuch konfigurieren

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

1. Im Hauptmenü **Diagnose** > **Logbuch** > **Konfiguration** wählen. Ein Fenster öffnet sich.
2. Die gewünschten Einstellungen vornehmen.
3. **OK** drücken, um die Konfiguration zu speichern und das Fenster zu schließen.

Beschreibung

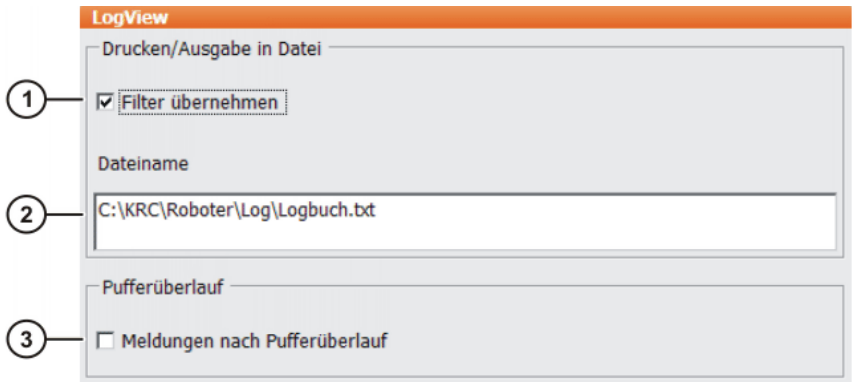


Abb. 13-3: Fenster Konfiguration Logbuch

Pos.	Beschreibung
1	■ **Checkbox aktiv**: Die mit dem Filter ausgewählten Log-Ereignisse werden in die Textdatei übernommen. ■ **Checkbox inaktiv**: Alle Log-Ereignisse werden in die Textdatei übernommen.
2	Pfad und Name der Textdatei eingeben. Default-Pfad: C:\KRC\ROBOTER\LOG\LOGBUCH.TXT
3	■ **Checkbox aktiv**: Log-Daten, die wegen Pufferüberlauf gelöscht wurden, werden in der Textdatei grau hinterlegt angezeigt. ■ **Checkbox inaktiv**: Log-Daten, die wegen Pufferüberlauf gelöscht wurden, werden in der Textdatei nicht angezeigt.

13.2 Aufrufliste (Caller Stack) anzeigen

Diese Funktion zeigt die Daten des Prozesszeigers ($PRO_IP) an.

Voraussetzung

- Benutzergruppe Experte
- Programm ist angewählt.

Vorgehensweise

- Im Hauptmenü **Diagnose** > **Aufrufstapel** wählen.

Beschreibung

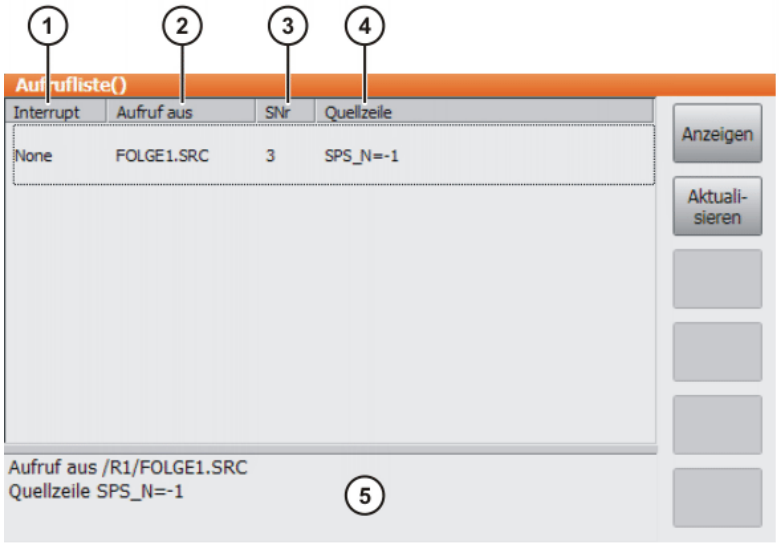


Abb. 13-4: Fenster Aufrufliste

Pos.	Beschreibung
1	■ **None**: Aufruf erfolgte nicht durch Interrupt ■ **[Nr.]**: Aufruf erfolgte durch Interrupt mit der Nummer [Nr.]
2	Diese Datei enthält den Aufruf.
3	Die Programmzeile mit dieser Nummer enthält den Aufruf. Voraussetzungen im Programm, damit anhand der Nummer die korrekte Zeile ermittelt werden kann: ■ Die Detailansicht ist eingeschaltet. ■ Alle Punkt-SPS sind geöffnet.
4	Quellzeile
5	Detaillierte Infomationen zu dem in der Liste markierten Eintrag

13.3 Interrupts anzeigen

Voraussetzung

- Benutzergruppe Experte

Vorgehensweise

- Im Hauptmenü **Diagnose** > **Interrupts** wählen.

Beschreibung

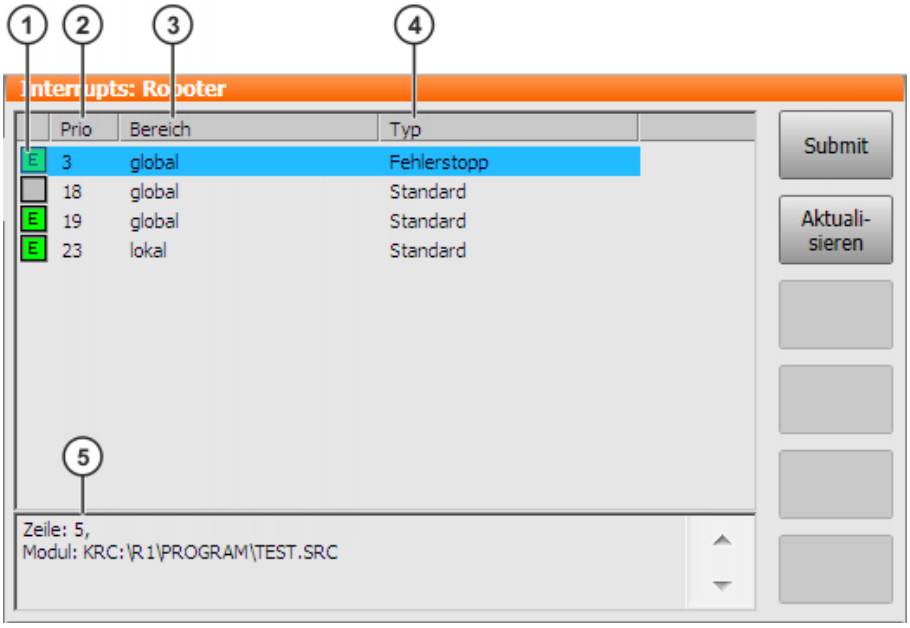


Abb. 13-5: Interrupts

Pos.	Beschreibung
1	Zustand des Interrupts ■ E Interrupt ON oder ENABLE ■ D Interrupt DISABLE ■ Interrupt OFF oder nicht eingeschaltet
2	Nummer/Priorität des Interrupts
3	Gültigkeitsbereich des Interrupts: Global oder lokal
4	Typ des Interrupts, abhängig vom definierten Ereignis in der Interrupt-Deklaration ■ **Standard**: z. B. $IN[…] ■ **Fehlerstop**: $STOPMESS ■ **NOT-HALT**: $ALARM_STOP ■ **Messen (schnelles Messen)**: $MEAS_PULSE[1…5] ■ **Trigger**: Trigger-Unterprogramm
5	Modul und Programmzeile der Interrupt-Deklaration

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Submit/ Roboter	Schaltet zwischen den Anzeigen für Roboter-Interrupts und Submit-Interrupts um.
Aktualisieren	Aktualisiert die Anzeige.

13.4 Diagnosedaten zum Grundsystem anzeigen

Beschreibung

Der Menüpunkt **Diagnosemonitor** ermöglicht es, zu zahlreichen Software-Teilbereichen des Grundsystems verschiedenste Diagnosedaten anzuzeigen.

Beispiele:

- Bereich **Kcp3 Treiber** (= Treiber für das smartPAD)
- Netzwerktreiber

Die angezeigten Daten sind abhängig vom ausgewählten Bereich. Angezeigt werden z. B. Zustände, Fehlerzähler, Meldungszähler usw.

Vorgehensweise

1. Im Hauptmenü **Diagnose** > **Diagnosemonitor** wählen.
2. Im Feld **Modul** einen Bereich auswählen.
   Zum ausgewählten Bereich werden Diagnosedaten angezeigt.

13.5 Daten automatisch verpacken für Fehleranalyse (KrcDiag)

Beschreibung

Wenn ein Fehler von der KUKA Roboter GmbH analysiert werden muss, kann man mit dieser Vorgehensweise die benötigten Daten verpacken, um sie KUKA zukommen zu lassen. Die Vorgehensweise erzeugt auf C:\KUKA\KRC-Diag eine ZIP-Datei. Diese enthält die Daten, die die KUKA Roboter GmbH benötigt, um einen Fehler zu analysieren. Hierzu gehören Informationen über Systemressourcen, Screenshots und vieles mehr.

Vorbereitung

Für das Datenpaket wird automatisch ein Screenshot von der aktuellen Ansicht der smartHMI erzeugt.

- Deshalb vor dem Starten des Vorgangs auf der smartHMI, wenn möglich, fehlerrelevante Informationen einblenden: Zum Bsp. das Meldungsfenster expandieren oder das Logbuch anzeigen.
  Welche Informationen sinnvoll sind, ist vom abhängig vom Einzelfall.

Vorgehensweise über "Diagnose"

- Im Hauptmenü **Diagnose** > **KrcDiag** wählen.
  Die Daten werden verpackt. Der Fortschritt wird in einem Fenster angezeigt. Wenn der Vorgang abgeschlossen ist, wird dies ebenfalls in dem Fenster angezeigt. Danach blendet sich das Fenster selbständig wieder aus.

Vorgehensweise über smartPAD

Diese Vorgehensweise verwendet keine Menüpunkte, sondern Tasten auf dem smartPAD. Sie kann deshalb auch dann eingesetzt werden, wenn die smartHMI nicht zur Verfügung steht, z. Bsp. aufgrund von Windows-Problemen.

Voraussetzung:

- Das smartPAD ist an der Robotersteuerung angesteckt.
- Die Robotersteuerung ist eingeschaltet.

Der Tasten müssen innerhalb von 2 Sekunden gedrückt werden. Ob auf der smartHMI dabei das Hauptmenü und die Tastatur angezeigt werden, ist unrelevant.

1. Die Hauptmenü-Taste drücken und halten.
2. Die Tastatur-Taste 2-mal drücken.
3. Die Hauptmenü-Taste loslassen.
   Die Daten werden verpackt. Der Fortschritt wird in einem Fenster angezeigt. Wenn der Vorgang abgeschlossen ist, wird dies ebenfalls in dem Fenster angezeigt. Danach blendet sich das Fenster selbständig wieder aus.

Vorgehensweise über "Archivieren"

Außerdem können die Daten auch über **Datei** > **Archivieren** > **[...]** verpackt werden. Hierbei besteht die Möglichkeit, sie auf einem USB-Stick oder einem Netzwerk-Pfad abzulegen.

(>>> 7.10 "Daten archivieren und wiederherstellen" Seite 255)

14 Installation

Die Robotersteuerung wird mit einem Windows-Betriebssystem und einer betriebsbereiten KUKA System Software (KSS) ausgeliefert. Bei der Erstinbetriebnahme ist daher keine Installation notwendig.

Eine Installation kann beispielsweise notwendig werden, wenn die Festplatte zerstört wurde und getauscht werden muss.

Die Robotersteuerung darf ausschließlich mit der von KUKA zusammen mit dieser Steuerung ausgelieferten Software betrieben werden. Wenn eine andere Software eingesetzt werden soll, ist Rücksprache mit der KUKA Roboter GmbH erforderlich. (>>> 15 "KUKA Service" Seite 493)

14.1 Systemvoraussetzungen

Die KSS 8.3 kann auf folgender Robotersteuerung eingesetzt werden:

- KR C4
- mit Windows Embedded Standard 7 V4.x
- und mit 2 GB Arbeitsspeicher

14.2 Windows und KUKA System Software (KSS) installieren (von Image)

Beschreibung

Es gibt mehrere Varianten, wie das Image eingespielt und finalisiert werden kann. Hier ist die am häufigsten benötigte Vorgehensweise beschrieben. Die Vorgehensweise zeigt auch, bei welchem Schritt bei Bedarf ein Master-Abbild erstellt werden kann.

Informationen zu den weiteren Varianten sowie zu verwandten Themen sind in folgenden Dokumentationen zu finden:

- Dokumentation **KUKA.RecoveryUSB 2.0**: Informationen zum Erstellen und Wiederherstellen von Images, zum Konfigurieren des Sticks und zu den möglichen Modi
- Experten-Dokumentation **WES7 System Preparation**: Informationen zur Finalisierung sowie zum Erstellen von Master-Abbildern

Voraussetzung

- Bootfähiger KUKA-USB-Stick mit Software **KUKA.RecoveryUSB 2.0** und Image
- Der Stick wurde so konfiguriert, dass der "Silent"-Modus aktiv ist.
- 2 GB Arbeitsspeicher
- Die Robotersteuerung ist ausgeschaltet.

Es wird dringend empfohlen, die gewünschten Software-Optionen (z. B. Technologiepakete) zu installieren, bevor **Finalize Installation** mit **Execute** gestartet wird. So ist es auch hier in der Vorgehensweise beschrieben.
Bei der Finalisierung werden die Projekte der Robotersteuerung auf Basis des aktiven Projekts neu erzeugt. Nur Optionen, die zu diesem Zeitpunkt bereits installiert sind, sind danach auch in allen Projekten enthalten (aktives Projekt, Initialprojekt und Basisprojekt).

Die LEDs auf dem CSP geben Auskunft über den Status der Installation. Informationen zu den LEDs sind in der Dokumentation **KUKA.RecoveryUSB** zu finden.

Vorgehensweise

1. Den USB-Stick an der Robotersteuerung anstecken.
2. Die Robotersteuerung einschalten. Die Installation startet automatisch.
   Die LEDs auf dem CSP beobachten! Es wird vorerst kein Bild auf dem smartPAD angezeigt.
3. Wenn die LEDs anzeigen, dass die Windows-Installation abgeschlossen ist, den Stick entfernen.
   - Nach der Windows-Installation startet die Robotersteuerung automatisch neu.
   - Danach startet die Robotersteuerung automatisch noch ein zweites Mal neu. (Spätestens zu diesem Zeitpunkt muss der Stick entfernt sein.)
4. Der Dialog **Finalize Installation** wird angezeigt.

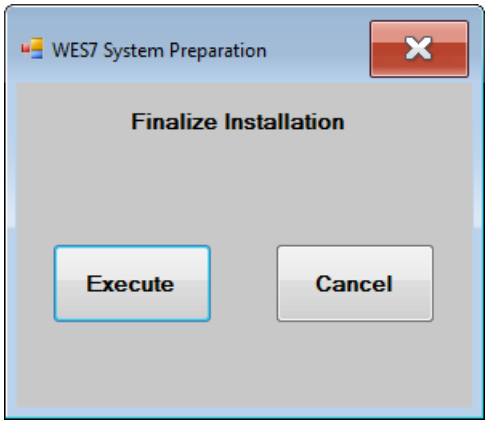


Abb. 14-1: Dialog Finalize Installation

5. Im Dialog **Finalize Installation** auf **Cancel** klicken.
   Eine Meldung informiert darüber, dass der Dialog beim nächsten Systemstart wieder angezeigt wird.

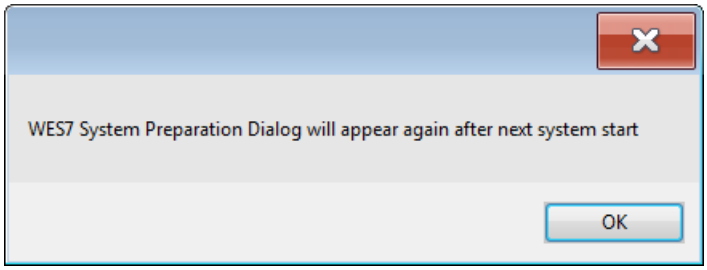


Abb. 14-2: Meldung nach Cancel

6. Die Meldung mit **OK** bestätigen.
7. Die gewünschten Software-Optionen installieren.
   Bei Bedarf kann die Robotersteuerung zwischen den einzelnen Installationen neu gestartet werden. Der Dialog **Finalize Installation** muss dann nach jedem Start wieder mit **Cancel** beantwortet werden.
8. Wenn die letzte Software-Option installiert wurde, die Robotersteuerung neu starten.
9. Den Dialog **Finalize Installation** jetzt mit **Execute** beantworten.
   Eine Meldung informiert darüber, dass die Systempreparation nun gestartet wird.

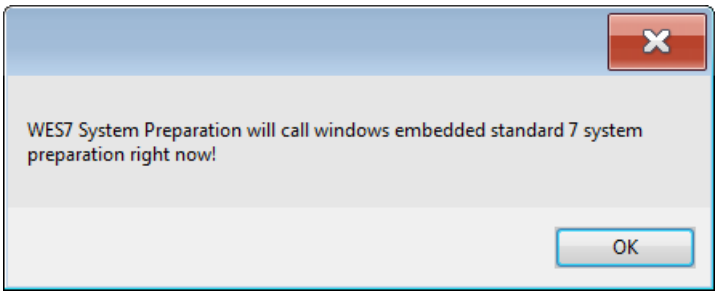


Abb. 14-3: Meldung nach Execute

10. Die Meldung mit **OK** bestätigen.

> Falls ein Master-Abbild erstellt werden soll: Hier noch keinen USB-Stick anstecken! Den Stick erst nach dem zweiten Herunterfahren anstecken – wie im Folgenden beschrieben.

Die Generalize Phase läuft ab. Danach startet die Robotersteuerung automatisch einmal neu und fährt dann automatisch herunter.

Nach dem Herunterfahren:

- Wenn ein Master-Abbild erstellt werden soll: Weiter mit Schritt 11.
- Wenn nicht: Weiter mit Schritt 12.

11. Diesen Schritt (Schritt 11) nur ausführen, wenn ein Master-Abbild erstellt werden soll.
    a. Den Recovery-USB-Stick anstecken.

> Der Stick muss jetzt so konfiguriert sein, dass ein Abbild erstellt werden kann!

    b. Die Robotersteuerung neu starten.
    c. Das Abbild erstellen.

       Im GUI-Modus muss die Erstellung über die Oberfläche gestartet werden, im Silent-Modus startet sie automatisch.

       Wenn die Erstellung abgeschlossen ist, fährt die Robotersteuerung automatisch herunter.
    d. Wenn die Robotersteuerung heruntergefahren ist: Den USB-Stick entfernen.
12. Die Robotersteuerung jetzt neu starten.

    Die Specialize Phase läuft ab. Dabei startet die Robotersteuerung automatisch 2-mal neu.
13. Nun beginnt das Mini-Setup:

    Die gewünschte Sprache wählen. Mit **Next** bestätigen.
14. Informationen zur Installation und zum Urheberrecht werden angezeigt. Mit **Next** bestätigen.
15. Angeben, ob die Robotersteuerung ein **OPS** (Offline Programming System) ist, auch "Office-PC" genannt. Dies ist in der Regel nicht der Fall, d. h. kein Häkchen setzen. Mit **Next** bestätigen.
16. Das System schlägt einen Robotertyp vor. Mit **Next** bestätigen.

    Oder: Wenn der vorgeschlagene Typ nicht dem verwendeten entspricht, einen anderen Typ auswählen. Dann mit **Next** bestätigen.
17. Eine Zusammenfassung der Setup-Einstellungen wird angezeigt. Mit **Next** bestätigen.

    Die Phase **Initial Project Setup** läuft ab. (Die Phase ist kurz.)

    Dann fährt die Robotersteuerung automatisch herunter.

Nun kann die Robotersteuerung neu gestartet werden, um das aktive Projekt in WorkVisual zu laden und zu konfigurieren.

Das Projekt enthält defaultmäßig den Roboter KR 210. Den Roboter in WorkVisual auf jeden Fall aus der Baumstruktur entfernen und den tatsächlich verwendeten Roboter einfügen.
Dieser Austausch muss auch dann gemacht werden, wenn der tatsächlich verwendete Roboter bereits der KR 210 ist.

Wenn das Projekt in WorkVisual konfiguriert wurde, kann es wieder auf die Robotersteuerung übertragen werden.

Bei Bedarf kann nun der Computer-Name geändert werden.

(>>> 14.3 "Computer-Name ändern" Seite 488)

14.3 Computer-Name ändern

Beschreibung

KUKA.RecoveryUSB vergibt bei jeder Wiederherstellung eines Abbilds einen neuen Computer-Namen. Der Name kann nach der Finalisation zu jedem beliebigen Zeitpunkt geändert werden.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware** wählen.
   Das Fenster **Installierte Zusatzsoftware** öffnet sich.
2. Auf die Schaltfläche **Neue Software** drücken.
   Das Fenster **Auswahl** öffnet sich.
3. Auf die Schaltfläche **Konfigurieren** drücken.
   Das Fenster **Installationspfade konfigurieren** öffnet sich.
4. Im Bereich **Installationspfade für Optionen** eine Zeile markieren.
   Hinweis: Wenn die Zeile bereits einen Pfad enthält, wird dieser überschrieben.
5. Auf **Pfadauswahl** drücken. Die vorhandenen Laufwerke werden angezeigt.
6. Zu C:\KUKA navigieren und dort den Ordner **WES7RenameComputer** markieren.
7. Auf **Speichern** drücken. Das Fenster **Installationspfade konfigurieren** wird wieder angezeigt. Es enthält jetzt den neuen Pfad.
8. Die Zeile mit dem neuen Pfad markieren und nochmal auf **Speichern** drücken.
   Das Fenster **Auswahl** wird wieder angezeigt.
9. Den Eintrag **WES7RenameComputer** markieren.
10. Auf **Installieren** drücken. Die Sicherheitsabfrage mit **Ja** beantworten.
11. Das Fenster **W7RenCom** und eine Popup-Tastatur werden angezeigt.
12. Den gewünschten Namen eingeben und zur Bestätigung auf die Schaltfläche **Set Computername** drücken.
13. Folgende Meldung wird angezeigt: *To finish the name setting you need to reboot the PC*.
    Die Meldung mit **OK** bestätigen.
14. Die Robotersteuerung neu starten.

14.4 Zusatzsoftware installieren

Mit dieser Funktion kann Zusatzsoftware installiert werden, z. B. Technologiepakete. Es können sowohl neue Programme als auch Updates installiert werden. Die Software wird von einem USB-Stick installiert. Alternativ kann sie auch von einem Netzwerk-Pfad installiert werden.

Das System prüft, ob die Zusatzsoftware für die KSS relevant ist. Wenn dies nicht der Fall ist, lehnt das System die Installation ab. Wenn eine Software, die vom System abgelehnt wird, dennoch installiert werden soll, muss Kontakt zur KUKA Roboter GmbH aufgenommen werden.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.
- USB-Stick mit der zu installierenden Software

HINWEIS Es wird empfohlen, einen KUKA-USB-Stick zu verwenden. Wenn ein Stick eines anderen Herstellers verwendet wird, können Daten verloren gehen.

Vorgehensweise

1. Den USB-Stick an der Robotersteuerung oder am smartPAD anstecken.
2. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware** wählen.
3. Auf **Neue Software** drücken: In der Spalte **Name** muss die neue Software angezeigt werden und in der Spalte **Pfad** das Laufwerk **E:** oder **K:**.
   Wenn nicht, auf **Aktualisieren** drücken.
4. Wenn die genannten Einträge jetzt angezeigt werden, weiter mit Schritt 5.
   Wenn nicht, muss erst der Pfad, von dem installiert werden soll, konfiguriert werden:
   a. Auf die Schaltfläche **Konfigurieren** drücken.
   b. Im Bereich **Installationspfade für Optionen** eine Zeile markieren.
      Hinweis: Wenn die Zeile bereits einen Pfad enthält, wird dieser überschrieben.
   c. Auf **Pfadauswahl** drücken. Die vorhandenen Laufwerke werden angezeigt.
   d. Wenn der Stick an der Robotersteuerung angesteckt ist: Auf **E:** die Ebene, auf der die Software liegt, markieren. Das kann direkt **E:** sein oder eine Unterebene.
      Wenn der Stick am smartPAD angesteckt ist: **K:** statt **E:**
   e. Auf **Speichern** drücken. Der Bereich **Installationspfade für Optionen** wird wieder angezeigt. Er enthält jetzt den neuen Pfad.
   f. Die Zeile mit dem neuen Pfad markieren und nochmal auf **Speichern** drücken.
5. Die neue Software markieren und auf **Installieren** drücken. Sicherheitsabfrage mit **Ja** beantworten.
6. Die Aufforderung zum Neustart mit **OK** bestätigen.
7. Den Stick abziehen.
8. Die Robotersteuerung neu starten.

Beschreibung

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
Neue Software	Alle Programme, die für eine Installation zur Verfügung stehen, werden angezeigt.
Zurück	Bereits installierte Zusatzsoftware wird angezeigt.

Schaltfläche	Beschreibung
Aktualisieren	Aktualisiert die Anzeige, z. B. nach Anschließen eines USB-Sticks.
Installieren	Zeigt weitere Schaltflächen an: ■ **Ja**: Die markierte Software wird installiert. Wenn danach ein Neustart erforderlich ist, wird dies durch eine Meldung angezeigt. ■ **Nein**: Die Software wird nicht installiert.
Konfigurieren	Diese Schaltfläche wird nur angezeigt, wenn vorher **Neue Software** gedrückt wurde. Hier können Pfade für die Installation von Zusatzsoftware oder KSS-Updates ausgewählt und gespeichert werden. Zeigt weitere Schaltflächen an: ■ **Pfadauswahl**: Ein neuer Pfad kann ausgewählt werden. ■ **Speichern**: Speichert die angezeigten Pfade.
Deinstallieren	Zeigt weitere Schaltflächen an: ■ **Ja**: Die markierte Software wird deinstalliert. ■ **Nein**: Die Software wird nicht deinstalliert.

14.5 KSS-Update

Beschreibung

Mit dieser Funktion können KSS-Updates installiert werden, z. B. von KSS 8.3.0 auf KSS 8.3.1.

Nach Installation oder Update der KUKA System Software führt die Robotersteuerung immer einen initialen Kaltstart aus.

Es wird empfohlen, vor dem Update einer Software alle zugehörigen Daten zu archivieren. Bei Bedarf kann so der alte Stand wiederhergestellt werden. Außerdem wird empfohlen, nach dem Update den aktuellen Stand zu archivieren.

Diese Funktion nicht verwenden, um eine neue Version zu installieren, z. B. von KSS 8.2 auf KSS 8.3. Die Funktion ebenfalls nicht verwenden, um eine Variante zu installieren, z. B. von KSS 8.3 auf KSS 8.3 SR. Für die Installation einer neuen Version oder Variante ist Rücksprache mit der KUKA Roboter GmbH erforderlich.
Diese Funktion kann nicht verwendet werden, um Updates von Zusatzsoftware, z.B. von Technologiepaketen, zu installieren.

Übersicht

Es gibt 2 Möglichkeiten, ein KSS-Update zu installieren:

- Von USB-Stick
  (>>> 14.5.1 "Update von USB-Stick" Seite 491)
- Vom Netz
  (>>> 14.5.2 "Update vom Netz" Seite 491)

14.5.1 Update von USB-Stick

HINWEIS Es muss ein nicht-bootfähiger USB-Stick verwendet werden.
Es wird empfohlen, einen nicht-bootfähigen KUKA-Stick zu verwenden. Wenn ein Stick eines anderen Herstellers verwendet wird, können Daten verloren gehen.

Voraussetzung

- Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.
- USB-Stick mit der zu installierenden Software

Vorgehensweise

1. USB-Stick anstecken.
2. Im Hauptmenü **Inbetriebnahme** > **Software Update** > **Automatisch** wählen.
3. Eine Sicherheitsabfrage wird angezeigt, ob wirklich ein Update durchgeführt werden soll. Mit **Ja** bestätigen.
   Im Meldungsfenster wird nun folgende Meldung angezeigt: *Software Update ist vorbereitet, bitte Rechner NEUSTARTEN!*
4. Im Hauptmenü **Herunterfahren** wählen, dann die Option **Steuerungs-PC neu starten** wählen. (**Dateien neu einlesen** ist nicht notwendig.)
5. Die Sicherheitsabfrage mit **Ja** bestätigen. Die Robotersteuerung startet neu und führt das Update durch.
   Danach führt die Robotersteuerung nochmal einen Neustart durch.
6. Wenn die Robotersteuerung den zweiten Neustart durchgeführt hat, kann der USB-Stick abgezogen werden.

Die aktualisierte System Software steht jetzt zur Verfügung.

14.5.2 Update vom Netz

Beschreibung

Bei einem Update vom Netz werden die Installationsdaten auf das lokale Laufwerk D:\ kopiert. Wenn sich auf D:\ bereits die Kopie einer System Software befindet, wird diese Kopie überschrieben.

Wenn der Kopiervorgang abgeschlossen ist, startet die Installation.

Voraussetzung

Für die Vorbereitung:

- Es ist kein Programm angewählt.
- Betriebsart T1 oder T2
- Benutzergruppe Experte

Für die Vorgehensweise:

- Es ist kein Programm angewählt.
- Betriebsart T1 oder T2

Vorbereitung

Den Netzwerkpfad, von dem aus das Update installiert werden soll, konfigurieren:

1. Im Hauptmenü **Inbetriebnahme** > **Zusatzsoftware** wählen.
2. Auf **Neue Software** drücken.
3. Auf **Konfigurieren** drücken.
4. Das Feld **Installationspfad für das KRC Update über das Netz** markieren. Auf **Pfadauswahl** drücken.

5. Den gewünschten Netzwerkpfad markieren (= das Verzeichnis, in dem die Datei Setup.exe liegt). Auf **Speichern** drücken.
6. Im Feld **Installationspfad für das KRC Update über das Netz** wird nun der gewählte Pfad angezeigt.

   Nochmal auf **Speichern** drücken.
7. Das Fenster schließen.

Der Netzwerkpfad muss nur einmal konfiguriert werden. Er bleibt für weitere Updates gespeichert.

Vorgehensweise

1. Im Hauptmenü **Inbetriebnahme** > **Software Update** > **Netz** wählen.
2. Eine Sicherheitsabfrage wird angezeigt, ob wirklich ein Update durchgeführt werden soll. Mit **Ja** bestätigen.

   Abhängig von der Netzauslastung kann der Vorgang bis zu 15 min dauern.
3. Eine Meldung wird angezeigt, dass beim nächsten Booten ein Kaltstart erzwungen wird. Die Steuerung ausschalten.
4. Warten, bis der Rechner komplett heruntergefahren ist. Dann die Steuerung wieder einschalten.
5. Wenn das Update abgeschlossen ist, wird der Rechner automatisch heruntergefahren und neu gestartet.

15 KUKA Service

15.1 Support-Anfrage

Einleitung

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

Informationen

Zur Abwicklung einer Anfrage werden folgende Informationen benötigt:

- Problembeschreibung inkl. Angaben zu Dauer und Häufigkeit der Störung
- Möglichst umfassende Informationen zu den Hardware- und Software-Komponenten des Gesamtsystems

  Die folgende Liste gibt Anhaltspunkte, welche Informationen häufig relevant sind:
  - Typ und Seriennummer der Kinematik, z. B. des Manipulators
  - Typ und Seriennummer der Steuerung
  - Typ und Seriennummer der Energiezuführung
  - Bezeichnung und Version der System Software
  - Bezeichnungen und Versionen weiterer/anderer Software-Komponenten oder Modifikationen
  - Diagnosepaket **KrcDiag**

    Für KUKA Sunrise zusätzlich: Vorhandene Projekte inklusive Applikationen

    Für Versionen der KUKA System Software älter als V8: Archiv der Software (**KrcDiag** steht hier noch nicht zur Verfügung.)
  - Vorhandene Applikation
  - Vorhandene Zusatzachsen

15.2 KUKA Customer Support

Verfügbarkeit

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

Argentinien

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

Australien

KUKA Robotics Australia Pty Ltd
45 Fennell Street
Port Melbourne VIC 3207
Australien
Tel. +61 3 9939 9656
info@kuka-robotics.com.au
www.kuka-robotics.com.au

Belgien

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

Brasilien

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

Chile

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

China

KUKA Robotics China Co., Ltd.
No. 889 Kungang Road
Xiaokunshan Town
Songjiang District
201614 Shanghai
P. R. China
Tel. +86 21 5707 2688
Fax +86 21 5707 2603
info@kuka-robotics.cn
www.kuka-robotics.com

Deutschland

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

Frankreich

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

Indien

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

Italien

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

Japan

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

Kanada

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

Korea

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

Malaysia

KUKA Robot Automation (M) Sdn Bhd
South East Asia Regional Office
No. 7, Jalan TPP 6/6
Taman Perindustrian Puchong
47100 Puchong
Selangor
Malaysia
Tel. +60 (03) 8063-1792
Fax +60 (03) 8060-7386
info@kuka.com.my

Mexiko

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

Norwegen

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

Österreich

KUKA Roboter CEE GmbH
Gruberstraße 2-4
4020 Linz
Österreich
Tel. +43 7 32 78 47 52
Fax +43 7 32 79 38 80
office@kuka-roboter.at
www.kuka.at

Polen

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

Portugal

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

Russland

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

Schweden

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

Schweiz

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

Spanien

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

Südafrika

Jendamark Automation LTD (Agentur)
76a York Road
North End
6000 Port Elizabeth
Südafrika
Tel. +27 41 391 4700
Fax +27 41 373 3869
www.jendamark.co.za

Taiwan

KUKA Robot Automation Taiwan Co., Ltd.
No. 249 Pujong Road
Jungli City, Taoyuan County 320
Taiwan, R. O. C.
Tel. +886 3 4331988
Fax +886 3 4331948
info@kuka.com.tw
www.kuka.com.tw

Thailand

KUKA Robot Automation (M)SdnBhd
Thailand Office
c/o Maccall System Co. Ltd.
49/9-10 Soi Kingkaew 30 Kingkaew Road
Tt. Rachatheva, A. Bangpli
Samutprakarn
10540 Thailand
Tel. +66 2 7502737
Fax +66 2 6612355
atika@ji-net.com
www.kuka-roboter.de

Tschechien

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

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

Index

W

X

Z