



Training
Roboterprogrammierung 1
KUKA System Software 8.6
KUKA College



Schulungsunterlage

Stand: 16.02.2021
PROG P1 KSS 8 V3.2.2
KUKA Deutschland GmbH

© Copyright 2021

KUKA Deutschland GmbH
Zugspitzstraße 140
D-86165 Augsburg
Deutschland

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

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

KIM-PS5-DOC

Original-Dokumentation

Publikation: Pub COLLEGE PROG P1 KSS 8 (V3.2) (PDF-COL) de
PB17011

Buchstruktur: P1KSS8 - Roboterprogrammierung 1 (R3) V4.2
BS10350

Version: PROG P1 KSS 8 V3.2.2

Inhaltsverzeichnis

1	Aufbau und Funktion eines KUKA-Robotersystems.....	9
1.1	Lerneinheit: Aufbau und Funktion eines KUKA Robotersystems.....	9
1.2	Roboter Basics.....	9
1.3	Mechanik eines KUKA Roboters.....	10
1.4	Robotersteuerung (V)KR C4.....	12
1.5	Das KUKA smartPAD.....	15
1.5.1	Übersicht smartPAD 1.....	17
1.5.2	Übersicht smartPAD 2.....	20
1.5.3	smartPAD an- und abstecken.....	23
1.5.4	smartPAD Grundeinstellungen	27
1.6	Roboterprogrammierung.....	29
1.7	Robotersicherheit.....	32
2	Dokumentation des Robotersystems.....	35
2.1	Lerneinheit: Dokumentation des Robotersystems.....	35
2.2	KUKA Xpert.....	35
2.3	KUKA Xpert URL.....	36
2.4	Registrierung.....	37
2.5	Anmeldung.....	38
2.6	KUKA Xpert Basic nutzen.....	40
2.7	Übung: KUKA Xpert Basic nutzen.....	41
3	Roboter bewegen.....	43
3.1	Lerneinheit: Roboter bewegen.....	43
3.2	Meldungen der Robotersteuerung lesen und interpretieren.....	43
3.3	Betriebsarten der Robotersteuerung.....	47
3.3.1	Stopp-Reaktionen.....	48
3.3.2	Betriebsart wechseln.....	50
3.4	Roboterachsen einzeln bewegen	51
3.4.1	Inkrementelles Handverfahren.....	55
3.4.2	Roboter in Notfällen ohne Steuerung bewegen.....	56
3.4.3	Übung: Bedienung und achsspezifisches Handverfahren.....	59
3.5	Koordinatensysteme im Zusammenhang mit Robotern.....	60
3.5.1	Koordinatensysteme und "Rechte-Hand-Regel"	61
3.6	Roboter im Weltkoordinatensystem bewegen.....	63
3.6.1	Übung: Bedienung und Handverfahren im Weltkoordinatensystem mittels Verfahrtasten.....	70
3.6.2	Übung: Bedienung und Handverfahren im Weltkoordinatensystem mittels 6D-Maus.....	71
3.7	Roboter mittels Verfahrtart Spur bewegen.....	72
3.8	Anzeigemöglichkeiten von Roboterpositionen.....	76
3.8.1	Aktuelle Roboterposition auf dem smartPAD abfragen.....	78
4	Inbetriebnahme des Roboters.....	83
4.1	Lerneinheit: Inbetriebnahme des Roboters.....	83
4.2	Benutzergruppen auf der Steuerung.....	83
4.2.1	Benutzergruppe wechseln.....	84

4.2.2	Kennwort ändern.....	86
4.2.3	Rechte auf der Steuerung verwalten.....	87
4.3	Prinzip des Justierens.....	88
4.3.1	Justagemittel.....	91
4.3.2	Ermittlung der mechanischen Nullstellung.....	94
4.3.3	Roboter justieren.....	95
4.3.3.1	Erst- /Standardjustage.....	96
4.3.3.2	Justage mit Lastkorrektur.....	100
4.3.3.3	Übung: Justage, Lastjustage mit Offset.....	106
4.3.4	Justage am Kleinroboter AGILUS.....	107
4.3.5	Wichtige Justagedateien.....	109
4.4	Werkzeug- und Basisverwaltung.....	111
4.4.1	Werkzeug- und Basisverwaltung auf dem smartPAD.....	111
4.5	Lasten am Roboter.....	116
4.5.1	Werkzeuglastdaten.....	116
4.5.2	Zusatzzlasten am Roboter.....	118
4.6	Vermessung eines Werkzeuges.....	120
4.6.1	Neues Werkzeug/Werkstück hinzufügen.....	125
4.6.1.1	TCP-Vermessung 4-Punkt-Methode.....	127
4.6.1.2	XYZ-Referenz Methode.....	132
4.6.1.3	ABC-Welt Methode.....	136
4.6.1.4	ABC-2-Punkt Methode.....	139
4.7	Roboter im TOOL-Koordinatensystem bewegen.....	143
4.7.1	Übung: Handverfahren im Werkzeugkoordinatensystem.....	147
4.8	Vermessen einer Basis.....	148
4.8.1	Neue Basis/festes Werkzeug hinzufügen.....	151
4.8.1.1	3-Punkt-Methode.....	153
4.8.2	Roboter im Basiskoordinatensystem bewegen.....	158
4.8.2.1	Übung: Handverfahren im Basiskoordinatensystem.....	162
4.9	Roboter mittels Verfahrt Ausrichten bewegen.....	163
4.10	Speicherauslastung anzeigen.....	166
4.11	Steuerung herunterfahren.....	167
5	Roboterprogramme ausführen.....	171
5.1	Lerneinheit: Roboterprogramme ausführen.....	171
5.2	Roboterprogramme anwählen.....	171
5.3	Wie sieht ein Roboterprogramm aus?.....	173
5.4	Initialisierungsfahrt durchführen.....	174
5.5	Programmstart durchführen.....	176
5.5.1	Übung: Roboterprogramme ausführen.....	179
6	Umgang mit Programmdateien.....	181
6.1	Lerneinheit: Umgang mit Programmdateien.....	181
6.2	Programmmodul erstellen.....	181
6.3	Programmmodul bearbeiten.....	184
6.4	Roboterprogramme archivieren und wiederherstellen.....	187
6.5	Programm- und Zustandsänderungen nachvollziehen mittels Logbuch.....	191
6.5.1	Logbuch exportieren.....	193
7	Programmierte Bewegungen erstellen und ändern.....	195

7.1	Lerneinheit: Programmierte Bewegungen erstellen und ändern.....	195
7.2	Welche Informationen werden für die Erstellung neuer Bewegungsbefehle benötigt?.....	195
7.3	SPTP - Taktzeitoptimierte Bewegungen (Achsbewegung).....	197
7.3.1	Status und Turn.....	199
7.3.2	SPTP Überschleifen.....	200
7.3.3	Bewegungen mittels Inlineformular programmieren.....	201
7.3.3.1	Inlineformular: SPTP.....	204
7.3.4	Übung: Luftprogramm - Programmhandhabung und SPTP-Bewegungen.....	207
7.4	Bahnbewegungen erstellen.....	208
7.4.1	Singularitäten.....	209
7.4.2	SLIN Programmieren.....	212
7.4.2.1	Inlineformular: SLIN.....	214
7.4.2.2	Orientierungsführung bei SLIN.....	215
7.4.3	SCIRC Programmieren.....	217
7.4.3.1	Inlineformular: SCIRC.....	220
7.4.3.2	Überschleifen bei Bahnbewegungen.....	221
7.4.3.3	Orientierungsführung bei SCIRC.....	222
7.4.3.4	SCIRC: Orientierungsverhalten – Beispiel Hilfspunkt.....	224
7.4.3.5	SCIRC: Orientierungsverhalten – Beispiel Zielpunkt.....	228
7.4.3.6	Einschränkungen bei \$CIRC_MODE	229
7.4.4	Bahnfahren und Überschleifen.....	230
7.5	Mit globalen Punkten programmieren.....	231
7.5.1	Globale Punkte in einer Übersicht anzeigen und ändern.....	234
7.6	Ändern von Bewegungsbefehlen.....	236
8	Programmieren von Spline-Bewegungen.....	245
8.1	Lerneinheit: Programmieren von Spline-Bewegungen.....	245
8.2	Was ist ein SPLINE ?.....	245
8.3	Beschreibung des SPLINE-Blocks.....	249
8.4	SPLINE-Block mit Bewegung programmieren.....	252
8.4.1	SLIN im SPLINE-Block.....	255
8.4.2	SCIRC im SPLINE-Block.....	256
8.4.3	SPL im SPLINE-Block.....	258
8.4.4	Bewegungsparameter.....	258
8.5	Programmierhinweise.....	259
8.5.1	Punkte in einem SPLINE-Block verändern.....	260
8.5.2	Geschwindigkeitsprofil bei Spline-Bewegungen.....	262
8.5.3	Satzanwahl bei Spline-Bewegungen.....	266
8.6	Überschleifen von Spline-Bewegungen.....	267
8.7	Übung: Bahnkontur mit Spline-Block.....	268
9	Logische Funktionen im Roboterprogramm nutzen.....	269
9.1	Lerneinheit: Logische Funktionen im Roboterprogramm nutzen.....	269
9.2	Einstieg in die Logikprogrammierung.....	269
9.3	Anzeige von Variablen.....	270
9.3.1	Anzeige von Ein- und Ausgängen.....	272
9.3.2	Anzeige von Flags, Zähler und Timern.....	273
9.4	Einfache Logikprogrammierung.....	274
9.4.1	Programmieren von Wartefunktionen.....	275

9.4.2	Programmierung von einfachen Schaltfunktionen.....	280
9.4.3	Optionale Übung: Bahnbewegung mit Logik versehen.....	285
9.5	Programmierung von Logik mit SPLINE und SPLINE-Einzelsätzen.....	286
9.5.1	Spline-Trigger programmieren.....	286
9.5.1.1	SPLINE Trigger programmieren.....	287
9.5.1.2	SPLINE-Trigger mittels Inline-Formulars programmieren.....	291
9.5.1.3	Programmierhinweise.....	293
9.5.2	Bedingen Stop programmieren.....	294
9.5.2.1	Programmierhinweise.....	295
9.5.2.2	Bedingen Stop programmieren.....	296
9.5.2.3	Bedingen Stop mittels Inlineformular programmieren.....	299
9.5.3	Konstantfahrbereich.....	302
9.5.3.1	Konstantfahrbereich programmieren.....	303
9.5.3.2	Programmierhinweise.....	307
9.5.4	Übung: Splinebewegung mit Logik versehen.....	309
9.5.5	Übung: Konstantfahrbereich und bedingter Stop.....	310
10	Technologiepakete nutzen.....	311
10.1	Lerneinheit: Technologiepakete nutzen.....	311
10.2	GripperTech kennenlernen.....	311
10.3	Greiferbedienung mit KUKA.GripperTech.....	312
10.3.1	Greifersignale am smartPAD anzeigen.....	313
10.4	Greifer konfigurieren.....	315
10.4.1	Greifer parametrieren.....	321
10.5	Greifer mittels Inlineformularen programmieren.....	323
10.5.1	Greiferzustände mittels Inlineformular prüfen.....	327
10.5.2	Übung: Greiferpogrammierung Schild.....	330
10.5.3	Übung: Greiferpogrammierung Stift.....	331
11	Konfiguration und Programmierung von externen Werkzeugen.....	333
11.1	Lerneinheit: Konfiguration und Programmierung von externen Werkzeugen.....	333
11.2	Robotergeführtes Werkstück und feststehendes Werkzeug.....	333
11.2.1	Vermessen eines feststehenden Werkzeugs.....	334
11.2.2	Vermessen eines robotergeführten Werkstücks.....	339
11.2.3	Übung: Externes Werkzeug und robotergeführtes Werkstück vermessen.....	345
11.3	Handverfahren mit einem feststehenden Werkzeug.....	346
11.3.1	Übung: Handverfahren mit feststehendem Werkzeug.....	351
11.4	Bewegungsprogrammierung mit externem TCP.....	352
11.4.1	Übung: Bewegungsprogrammierung mit externen TCP.....	353
12	Einführung in die Expertenebene.....	355
12.1	Lerneinheit: Einführung in die Expertenebene.....	355
12.2	Programme erstellen mittels Templates.....	355
12.3	Ansichten im Explorer anpassen.....	357
12.4	Ansicht im Editor anpassen.....	358
12.5	Fehler im Programm finden und beheben.....	360
12.6	Roboterprogramme strukturieren.....	362
12.6.1	Programme kommentieren.....	362
12.6.2	Programme einrücken.....	365
12.6.3	Programmzeilen ausblenden.....	366

12.6.4	Roboterprogramme verknüpfen.....	366
12.6.5	Übung: Unterprogrammaufruf programmieren	370
13	Variablen und Vereinbarungen.....	371
13.1	Lerneinheit: Variablen und Vereinbarungen.....	371
13.2	Datenhaltung in KRL.....	371
13.2.1	Namenskonventionen.....	371
13.2.2	Datentypen.....	372
13.2.3	Anlegen von Variablen.....	373
13.2.4	Doppeldeklaration von Variablen.....	373
13.2.5	Lebensdauer und Gültigkeit von Variablen.....	373
13.2.6	Variablendeclaration in Abhängigkeit des Speicherortes.....	373
13.2.7	KUKA Systemdaten.....	374
13.3	Arbeiten mit einfachen Datentypen.....	374
13.3.1	Logikbefehle und Schleifen mittels Inline-Formulare programmieren.....	375
13.3.2	Deklarieren von Variablen.....	375
13.3.3	Initialisierung von Variablen mit einfachen Datentypen.....	378
13.3.4	Variable über ein Inlineformular deklarieren und initialisieren.....	381
13.3.5	Manipulation von Variablenwerten einfacher Datentypen mit KRL.....	382
13.3.6	Übung: Einfache Datentypen	387
14	Nutzen von Programmablaufkontrollen.....	389
14.1	Lerneinheit: Nutzen von Programmablaufkontrollen.....	389
14.2	Schleifen programmieren.....	389
14.2.1	Endlosschleife programmieren.....	389
14.2.2	Zählschleife programmieren.....	392
14.2.3	Abweisende Schleife programmieren.....	395
14.2.4	Nicht abweisende Schleife programmieren.....	398
14.3	Abfragen oder Verzweigungen programmieren.....	401
14.4	Verteiler programmieren (SWITCH- CASE).....	404
14.5	Sprungbefehl programmieren.....	407
14.6	Wartefunktionen in KRL programmieren.....	408
14.6.1	Zeitabhängige Wartefunktion.....	409
14.6.2	Signalabhängige Wartefunktion.....	410
14.7	Übung: Schleifentechniken	414
15	Arbeiten mit einer übergeordneten Steuerung.....	415
15.1	Lerneinheit: Arbeiten mit einer übergeordneten Steuerung.....	415
15.2	Vorbereitung zum Programmstart über die SPS.....	415
15.3	SPS-Anbindung anpassen (Cell.src).....	418
15.4	Fragen: Arbeiten an einer übergeordneten Steuerung	421
16	Anhang.....	423
16.1	Beschreibung der Automatik Extern Schnittstelle.....	423
16.1.1	Konfiguration der Automatik-Extern-Schnittstelle.....	430
16.1.2	Übung: Konfiguration der Automatik Extern Schnittstelle (manuell).....	434
16.2	Abkürzungen.....	435
Index		437

1 Aufbau und Funktion eines KUKA-Robotersystems

1.1 Lerneinheit: Aufbau und Funktion eines KUKA Robotersystems

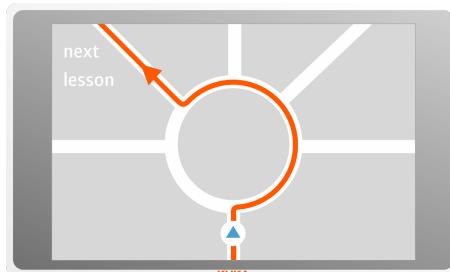


Abb. 1-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Robotiker Basics
- Mechanik eines KUKA Roboters
- Robotersteuerung KR C4
- KUKA smartPAD
- Roboterprogrammierung
- Sicherheit am Roboter

1.2 Roboter Basics

Was ist ein Roboter?

Der Begriff *Roboter* kommt vom slawischen Wort *robo*, was für *schwere Arbeit* steht. Die offizielle Definition für einen Industrieroboter lautet: "Ein Roboter ist ein freiprogrammierbares, programmgesteuertes Handhabungsgerät." Zum Robotersystem gehören auch die Steuerung, das Bediengerät sowie deren Verbindungsleitungen und die Software.

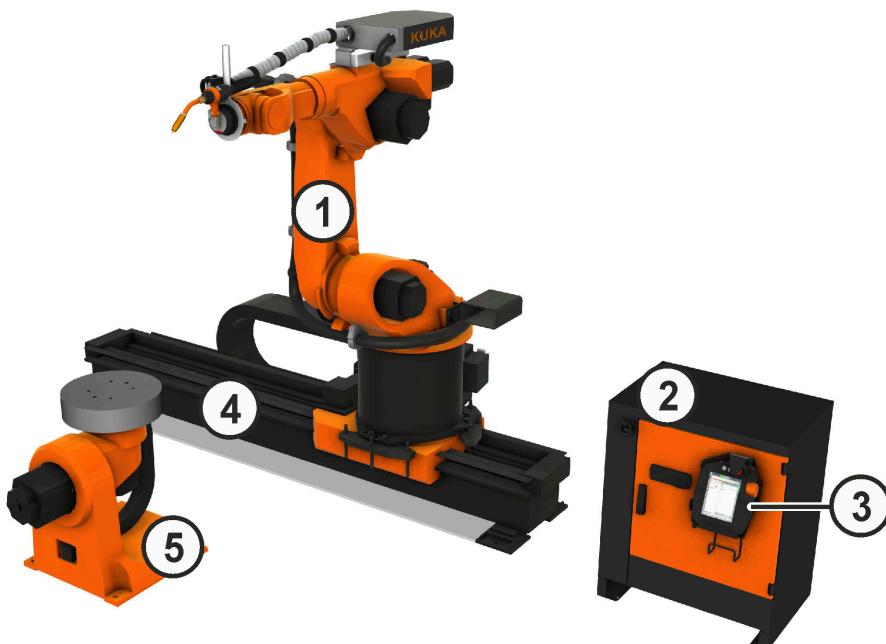


Abb. 1-2: Industrieroboter

- 1 Manipulator (Robotermechanik)
- 2 Steuerung (Steuerschrank (V)KR C4)
- 3 Bedien- und Programmierhandgerät (KUKA smartPAD)
- 4 Optional: KUKA Lineareinheit
- 5 Optional: KUKA Drehkipptisch

Alles außerhalb der Systemgrenzen des Industrieroboters wird als *Peripherie* bezeichnet:

- Werkzeuge (Effektor/Tool)
- Schutzeinrichtungen
- Transportbänder
- Sensoren
- Maschinen
- etc.

1.3 Mechanik eines KUKA Roboters

Was ist ein Manipulator?

Die Robotermechanik des Manipulators besteht aus einer Anzahl von beweglichen, aneinander geketteten starren Gliedern, die über Achsen miteinander verbunden sind. Die Achsen lassen sich präzise über Servomotoren mit vorgelagertem Getriebe bewegen.

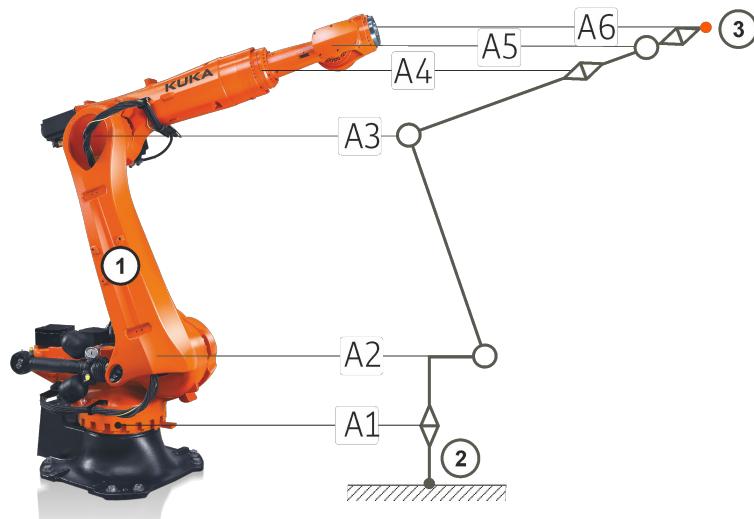


Abb. 1-3: Manipulator

- 1 Manipulator (Robotermechanik)
 - 2 Start der kinematischen Kette: Roboterfuß (ROBROOT)
 - 3 Freies Ende der kinematischen Kette: Flansch (FLANGE)
- A1 Roboterachsen 1 - 6
...
A6

Übersicht Komponenten Robotermechanik

Die Komponenten einer Robotermechanik bestehen vorwiegend aus Aluminium- und Stahlguss. In vereinzelten Fällen werden auch Kohlefaser-Komponenten verwendet.



Abb. 1-4: Übersicht Komponenten Robotermechanik

- | | |
|-----------------------------------|------------|
| 1 Grundgestell | 5 Schwinge |
| 2 Karussell | 6 Arm |
| 3 Schlauchpaket (Medien, Energie) | 7 Hand |
| 4 Gewichtsausgleich | |

Achsen des Roboters

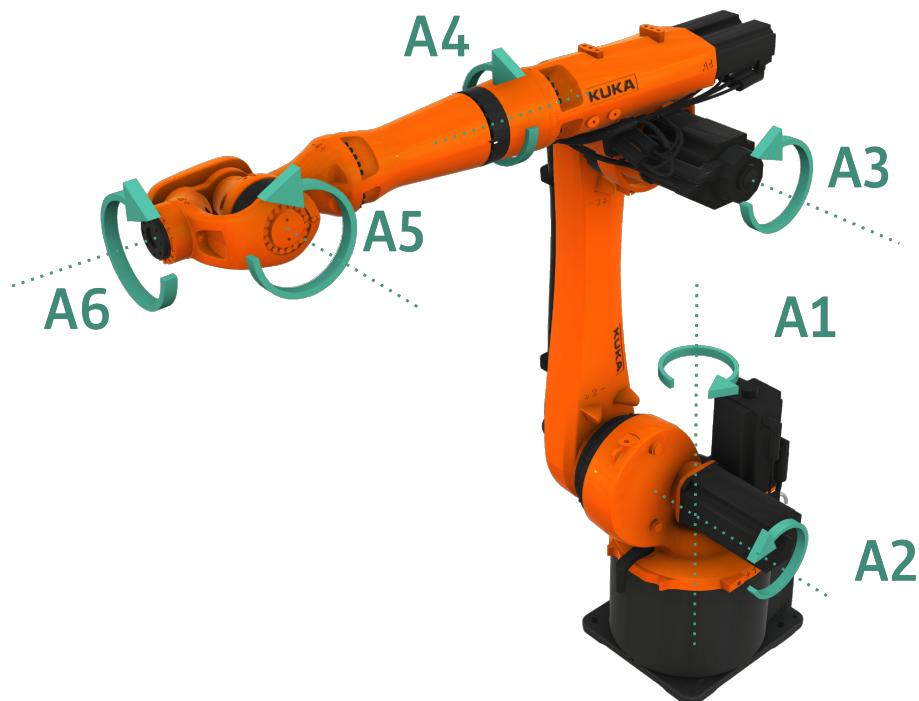


Abb. 1-5: Achsen eines KUKA Roboters

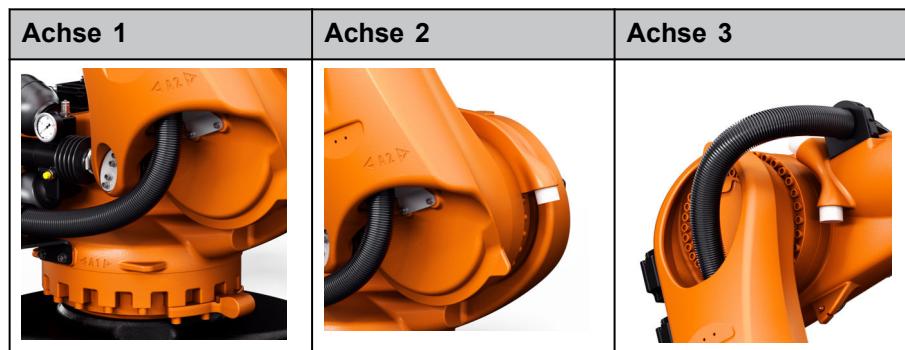
Beispiele aus den technischen Daten von Manipulatoren aus der KUKA Produktpalette:

- **Anzahl Achsen:**
 - 3 Achsen bei KR SCARA
 - 4 Achsen bei KR40 PA
 - 6 Achsen bei Standard Vertikal-Knickarmroboter
 - 7 Achsen bei Leichtbauroboter
- **Reichweite:** von 0,54 m (KR3 R540) bis 3,9 m (KR 120 R3900 ultra K)
- **Eigengewicht:** von 23 kg (LBR iiwa 7 R800) bis 4700 kg (KR1000 Titan).
- **Wiederholgenauigkeit:** 0,03 mm - 0,2 mm Wiederholgenauigkeit

Mechanische Endanschläge

Die Achsbereiche der Grundachsen A1 bis A3 und der Handachse A5 des Roboters sind durch mechanische Endanschläge mit Puffern begrenzt.

An den Zusatzachsen können weitere mechanische Endanschläge montiert sein.



HINWEIS

- Wenn der Roboter oder eine Zusatzachse gegen ein Hindernis oder einen Puffer am mechanischen Endanschlag oder der Achsbegrenzung fährt, können Sachschäden am Robotersystem entstehen.
- Vor der Wiederinbetriebnahme des Robotersystems ist Rücksprache mit der KUKA Roboter GmbH erforderlich.
Der betroffene Puffer ist gegen einen neuen Puffer zu tauschen, bevor der Roboter weiterbetrieben wird.
- Fährt der Roboter (die Zusatzachse) schneller als 250 mm/s gegen einen Puffer, muss der Roboter (die Zusatzachse) getauscht oder eine Wiederinbetriebnahme durch die KUKA Roboter GmbH durchgeführt werden.

1.4 Robotersteuerung (V)KR C4

Wer sorgt für Bewegung?

Die Robotermechanik wird durch Servomotoren bewegt, die von der (V)KR C4-Steuerung angesteuert werden.



Abb. 1-6: Steuerschrank (V)KR C4

(V)KR-C4-Steuerung

Robotersteuerung (Bahnplanung): Regelung von sechs Roboterachsen sowie zusätzlich bis zu drei externen Achsen.



Abb. 1-7: (V)KR C4 Achsregelung

(V)KR C4 extended Steuerung

Robotersteuerung (Bahnplanung): Regelung von sechs Roboterachsen sowie zusätzlich bis zu sechs externen Achsen



Abb. 1-8: (V)KR C4 extended Achsregelung

Softwarebasierte Steuerung

KR C4 basierende Steuerungen arbeiten nach einem modularen Konzept. Eine zentrale Rolle spielt der **KPC-KUKA PC**. Roboterprogramme greifen, je nach Applikationen, auf die notwendigen Hardwareschnittstellen zu. **Systemrelevanten Applikationen** können durch **kundenspezifisch Applikationen** ergänzt werden.

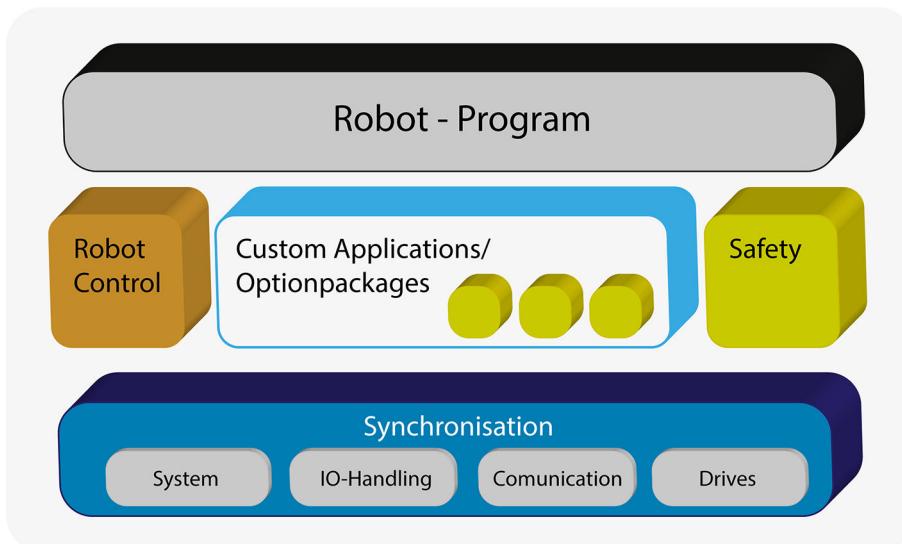


Abb. 1-9: Säulenmodell

Zu den **systemrelevanten Applikationen** zählen:

- **Robot Control**
KUKA-Kernsystem zur Robotersteuerung
- **Safety**
KUKA-eigene integrierte Sicherheitssteuerung

Beispiele **kundenspezifischer Optionen** :

- **ready2_spot**
dient zum robotergeführten Fügen/Schweißen von Bauteilen
- **ready2_fasten**
dient zum roboterbasierten Verschrauben von Schrauben in industrieller Umgebung
- **KUKA.VisionTech**
dient dazu, mithilfe von einer oder mehreren Kameras die Position eines Roboters bezogen auf die Lage eines Bauteils zu bestimmen und zu korrigieren.
- **KUKA.GlueTech**
dient dazu, mithilfe einer Klebeapplikation/-düse robotergeführt eine Kleberaube auf einem Bauteil aufzutragen.
- **KUKA.Gripper&SpotTech**
dient dazu, Werkzeuge und Greifer in einer Arbeitsumgebung zu steuern und zu überwachen

Kommunikationsmöglichkeiten



Abb. 1-10: Kommunikationsmöglichkeiten (V)KR C4

- Kommunikationsmöglichkeiten über Bussysteme (z. B. ProfiNet, Ethernet IP, EtherCAT)
 - speicherprogrammierbare Steuerungen (SPS)
 - weitere Steuerungen (z.B. Schweißsteuerung oder Robotersteuerung)
 - Sensoren und Aktoren
- Kommunikationsmöglichkeiten über Netzwerk
 - Leitrechner
 - Service-Notebook

1.5 Das KUKA smartPAD

Wie wird ein KUKA-Roboter bedient?

Die Bedienung eines KUKA-Roboters erfolgt über ein Bedienhandgerät, das KUKA smartPAD.



Abb. 1-11: KUKA smartPAD Varianten

Das KUKA smartPAD gibt es in zwei verschiedenen Varianten.

- smartPAD 1
- smartPAD 2

Das smartPAD 2 ist der Nachfolger des smartPAD 1 und ist voll abwärtskompatibel zu allen Bestandssteuerungen auf KR C4 Basis.

Merkmale des KUKA smartPAD:

- Touch-Screen (berührungsempfindliche Bedienoberfläche)
- großes, hochformatiges Display
- KUKA-Menütaste
- flexibel verwendbare +/- Tasten
 - Standard: Verwendung als Verfahrtasten, oder Einstellung der Verfahrgeschwindigkeiten
- Tasten für die Bedienung der Technologiepakete (z. B. Greiferfunktionen, Servozange, Docken)
- Tasten für den Programmablauf (Stopp/Rückwärts/Vorwärts)
- Taste zum Einblenden der Tastatur
- Schlüsselschalter für
 - den Wechsel der Betriebsart.
 - die Freigabe weiterer Funktionen.
- NOT-HALT-Taster
- 6D-Maus
- absteckbar
- USB-Anschluss

1.5.1 Übersicht smartPAD 1

Beschreibung

smartPAD Vorderseite



Abb. 1-12: KUKA smartPAD Vorderseite

Pos.	Beschreibung
1	Taster für die Anforderung "smartPAD abstecken"
2	Freigabeschalter für die Auswahl der Betriebsart. Der Schalter kann in folgenden Varianten ausgeführt sein: <ul style="list-style-type: none"> • mit Schlüssel • ohne Schlüssel Über diesen Schalter ruft man den Verbindungsmanager auf. Über den Verbindungsmanager kann die Betriebsart gewechselt werden.
3	NOT-HALT-Taster. Zum Stoppen des Roboters in Gefahrensituationen. Der NOT-HALT-Taster verriegelt sich, wenn dieser gedrückt wird.
4	6D-Maus: zum manuellen Verfahren des Roboters
5	Verfahrtasten: zum manuellen Verfahren des Roboters

Pos.	Beschreibung
6	Taste zum Einstellen des POV-Programm-Override
7	Taste zum Einstellen des HOV-Hand-Override
8	Hauptmenü-Taste: Diese blendet auf der smartHMI die Menüpunkte ein.
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 im Tippbetrieb. 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.

smartPAD Rückseite



Abb. 1-13: smartPAD Rückseite

Pos.	Element	Beschreibung
1	Stylus	Passiver Eingabestift zur Bedienung des resistiven Displays
2	Zustimmungs-schalter	<p>Die Zustimmungsschalter haben 3 Stellungen:</p> <ul style="list-style-type: none"> • Nicht gedrückt • Mittelstellung • Durchgedrückt <p>Einer der Zustimmungsschalter muss in den Betriebsarten T1 und T2 in der Mittelstellung gehalten werden, damit der Manipulator verfahren werden kann.</p> <p>In den Betriebsarten Automatik und Automatik Extern haben die Zustimmungsschalter keine Funktion.</p>
3	Start-Taste (grün)	Mit der Start-Taste startet man ein Programm.
4	Zustimmungs-schalter	Siehe 2
5	Typenschild	Aufdruck der Artikel- und Seriennummer
6	Zustimmungs-schalter	Siehe 2
7	USB-Anschluss	<p>Der USB-Anschluss wird z. B. verwendet für Archivierung/Wiederherstellung.</p> <p>Nur für FAT32 formatierte USB-Sticks.</p>

1.5.2 Übersicht smartPAD 2

Beschreibung

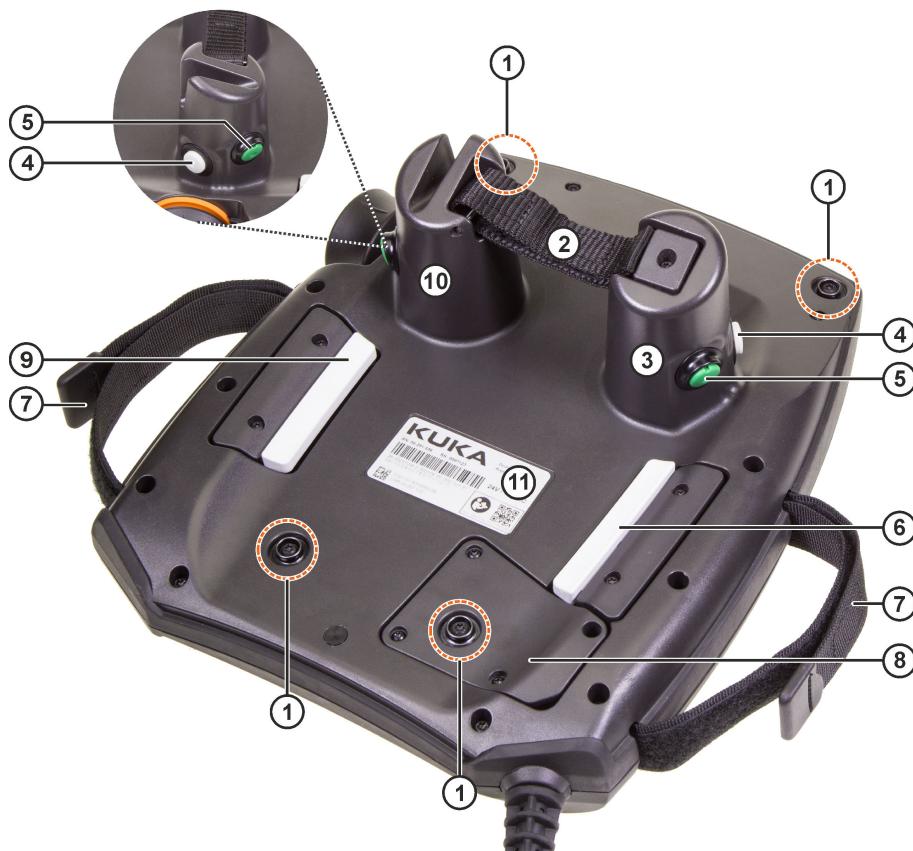
Vorderseite



Abb. 1-14: Vorderseite

Pos.	Beschreibung
1	2 USB 2.0 Schnittstellen mit Abdeckung Der USB-Anschluss wird z. B. für die Archivierung verwendet. Für NTFS und FAT32 formatierte USB-Sticks.
2	SmartPAD zum Abstecken vorbereiten
3	Schlüsselschalter zum Aufrufen des Verbindungs-Managers. Der Schalter kann nur umgelegt werden, wenn der Schlüssel steckt. Über den Verbindungs-Manager kann der Roboter und die Betriebsart vorselektiert werden.
4	NOT-HALT-Einrichtung NOT-HALT-Taster. Zum Stoppen des Roboters in Gefahrensituationen. Der NOT-HALT-Taster verriegelt sich, wenn er gedrückt wird.
5	Mit der 6D-Maus kann der Roboter manuell verfahren werden.
6	Verfahrtasten Mit den Verfahrtasten wird der Roboter manuell verfahren.

Pos.	Beschreibung
7	Handschlaufe mit Klettverschluss Werden die Handschlaufen nicht benutzt, können sie vollständig eingezogen werden.
8	Taste zum Einstellen des Programm-Overrides
9	Taste zum Einstellen des Hand-Overrides
10	Anschlusskabel
11	Benutzertasten Die Funktion der Benutzertasten ist frei programmierbar. Die Benutzertasten können z. B. dazu verwendet werden, Peripheriegeräte zu steuern oder applikationsspezifische Aktionen auszulösen.
12	Start-Taste Mit der Start-Taste wird ein Programm gestartet. Die Start-Taste wird auch verwendet, um Frames manuell anzufahren und den Roboter zurück auf die Bahn zu fahren.
13	Start-Rückwärts-Taste Mit der Taste kann man im Programm die Frames zurückfahren.
14	STOP-Taste Mit der STOP-Taste wird ein laufendes Programm angehalten.
15	Tastatur-Taste Blendet die Tastatur ein.
16	Hauptmenü-Taste Die Hauptmenü-Taste blendet das Hauptmenü auf der smartHMI ein und aus.

Rückseite smartPAD 2**Abb. 1-15: smartPAD 2 Rückseite**

Pos.	Beschreibung
1	Druckknöpfe zur Befestigung des (optionalen) Tragegurts
2	Halteschlaufe Dom
3	Linker Dom: Halten des smartPAD mit der rechten Hand
4	Zustimmungsschalter Die Zustimmungsschalter haben 3 Stellungen: <ul style="list-style-type: none"> • Nicht gedrückt • Mittelstellung • Durchgedrückt (Panikstellung) Einer der Zustimmungsschalter muss in den Betriebsarten T1, T2 in der Mittelstellung gehalten werden, damit der Manipulator verfahren kann. In der Betriebsart Automatik hat der Zustimmungsschalter standardmäßig keine Funktion.
5	Start-Taste (grün) Mit der Start-Taste wird ein Programm gestartet. Die Start-Taste wird auch verwendet, um Frames manuell anzufahren und den Roboter zurück auf die Bahn zu fahren.
6	Zustimmungsschalter
7	Handschlaufe mit Klettverschluss
8	Deckel (Abdeckung Anschlusskabel)
9	Zustimmungsschalter

Pos.	Beschreibung
10	Rechter Dom: Halten des smartPAD mit der linken Hand
11	Typenschild

1.5.3 smartPAD an- und abstecken

Beschreibung

- Das smartPAD kann über die Anforderungstaste bei laufender Robotersteuerung abgesteckt werden. Das Abstecken des smartPAD kann zu jedem Zeitpunkt und zu jeder Betriebsart erfolgen.

HINWEIS

Jedoch darf ein laufender Up- oder Downgrade Vorgang nicht unterbrochen werden.

- Durch das Anstecken übernimmt das smartPAD die aktuelle Betriebsart der Robotersteuerung. Beim Anstecken spielt die smartPAD-Variante (Firmwarestand) keine Rolle.

HINWEIS

Es kann ein automatischer Up- oder Downgrade initiiert werden, der nicht unterbrochen werden darf.

- Erst 30 s nach dem Anstecken sind der NOT-HALT und die Zustimmungsschalter wieder funktionsfähig. Die smartHMI (Bedienoberfläche) wird automatisch wieder angezeigt (dies dauert nicht länger als 15 s).

HINWEIS

Ein Abstecken des smartPADs ohne vorherige Anforderung führt zu einem NOT-HALT des Roboters.

Funktionsweise



WARNUNG

- Wird das smartPAD ohne vorheriges Betätigen des Tasters **Anforderung Abkoppeln** abgesteckt, wird ein lokaler NOT-HALT ausgelöst.
- Dies gilt auch, wenn mittels Taster die Anforderung zwar ausgelöst wurde, jedoch das Zeitfenster für das Abkoppeln von 25 Sekunden abläuft und danach das smartPAD abgesteckt wird.
- Wird der Roboter in einem Anlagenverbund betrieben, kann über die übergeordnete Sicherheitssteuerung einen Anlagen-NOT-HALT auslösen.



WARNUNG

- Wenn das smartPAD abgesteckt ist, kann die Anlage nicht mehr über die NOT-HALT-Einrichtung des smartPADs abgeschaltet werden.
- Aus diesem Grund ist ein zusätzlicher, nicht abkoppelbarer, externer NOT-HALT an der Anlage erforderlich.

**WARNUNG**

- Der Betreiber hat dafür Sorge zu tragen, dass abgesteckte smartPADs sofort aus der Anlage entfernt und außer Sicht- und Reichweite des am Industrieroboter arbeitenden Personals verwahrt werden oder sich in der dafür vorgesehen Halterung mit abgedeckten NOT-HALT befindet.
- Dies dient dazu, Verwechslungen zwischen wirksamen und nicht wirksamen NOT-HALT-Einrichtungen zu vermeiden.

**WARNUNG**

Wenn diese Maßnahmen nicht beachtet werden, können Tod von Personen, schwere Körperverletzungen oder erheblicher Sachschaden die Folge sein.

**WARNUNG**

- Der Bediener, der ein smartPAD an die Robotersteuerung ansteckt, muss danach mindestens 30 s am smartPAD verbleiben. Nach dieser Zeit ist die Verbindung zum smartPAD wiederhergestellt.
- Die Funktion des NOT-HALTs und der Zustimmungsschalter kann sichergestellt werden. So wird z. B. vermieden, dass ein anderer Benutzer in einer Notsituation auf einen momentan nicht wirksamen NOT-HALT zugreift.

smartPAD Aktualisierung

Während des Bootens kann es automatisiert zu einer smartPAD Aktualisierung kommen. Meist initiiert durch ein Update der Steuerungssoftware. Dieser Aktualisierungsvorgang darf nicht unterbrochen werden. Das smartPAD-eigene Betriebssystem wird über die Steuerung "down- oder upgradet".

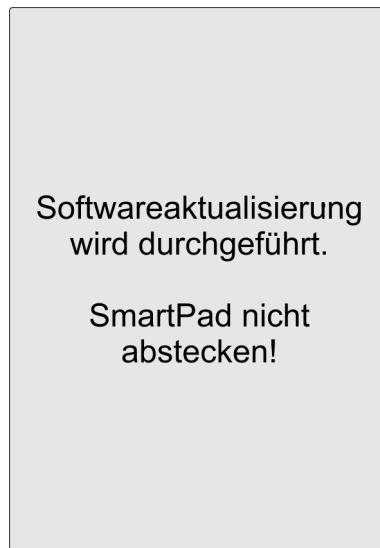


Abb. 1-16: smartPad Aktualisierung

HINWEIS

Das Abbrechen der Aktualisierung kann zu einer fehlerhaften Installation der Software auf dem smartPAD führen. Das smartPAD lässt sich dann möglicherweise nicht mehr starten.

Vorgehensweise

smartPAD abstecken

- Auf dem smartPAD den Anforderungstaster zum Abstecken drücken.
Auf der smartHMI werden eine Meldung und mit einem Timer angezeigt.

Der Zähler läuft 25 s. Während dieser Zeit kann das smartPAD von der Robotersteuerung abgesteckt werden.

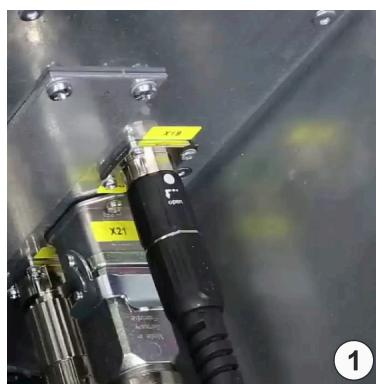


Abb. 1-17: Anforderungstaster: smartPAD abkoppeln

HINWEIS

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

- Das smartPAD von der Robotersteuerung abstecken.



1	Stecker im eingesteckten Zustand
2	den oberen schwarze Ring um ca. 25° in Pfeilrichtung drehen (siehe Aufdruck)
3	Stecker nach unten abziehen



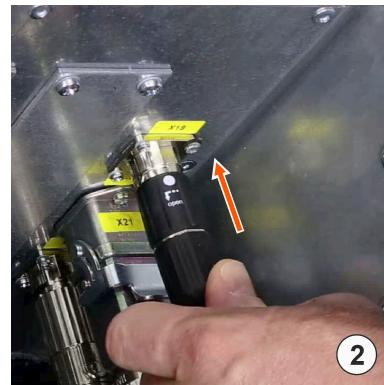
- Wenn der Zähler ausläuft, ohne dass das smartPAD abgesteckt wurde, hat dies keine Auswirkungen.
- Der Anforderungstaster kann beliebig oft gedrückt werden, um den Timer anzuzeigen.

smartPAD anstecken

1. Sicherstellen, dass der lokale NOT-HALT des smartPADs gelöst ist.
2. Den smartPAD-Stecker am X19 einrasten.

HINWEIS

Auf Markierung auf Buchse und smartPAD-Stecker achten!



1	Stecker im abgesteckten Zustand (Markierung beachten)
2	Den Stecker nach oben schieben. Der obere schwarze Ring dreht sich beim Hochschieben selbsttätig um ca. 25°.
3	Stecker rastet selbstständig ein



WARNUNG

- Der Benutzer, der ein smartPAD an die Robotersteuerung ansteckt, muss danach mindestens 30 s am smartPAD verbleiben, 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.

1.5.4 smartPAD Grundeinstellungen

Beschreibung

Beim Hochlaufen der Steuerung sowie bei geöffnetem Verbindungsmanager (>>> Abb. 1-18) werden drei grüne Symbole (1,2 und 3) am oberen Bildschirmrand des smartPADs eingeblendet. Diese signalisieren den aktuellen Zustand der Steuerung sowie dessen Verbindung zum smartPAD. Es werden die klassisch Ampelfarben rot, gelb und grün verwendet.

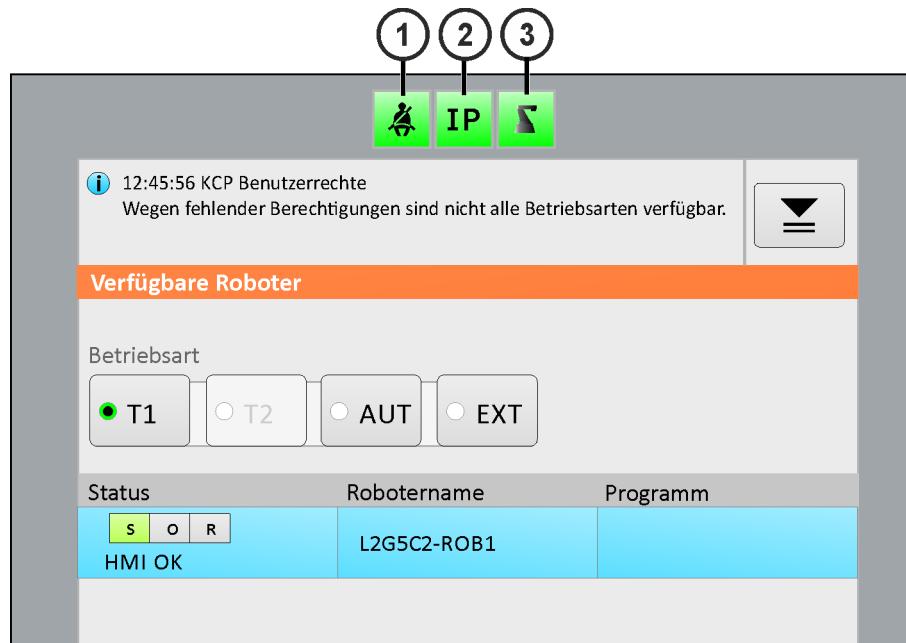


Abb. 1-18: Verbindungsmanager

- 1 Safety-Kommunikation hergestellt
- 2 IP-Adresse erfolgreich zugewiesen
- 3 KR C4 ist einsatzbereit

Einblenden von smartPAD-spezifischen Einstellungen/Konfigurationen

1. Den Verbindungsmanager mittels Schlüsselschalter öffnen.



Abb. 1-19: Schlüsselschalter für Verbindungsmanager und Betriebsart

2. Die Robotertaste drücken »
3. Das **smartPAD Session Manager Menü** öffnet sich.

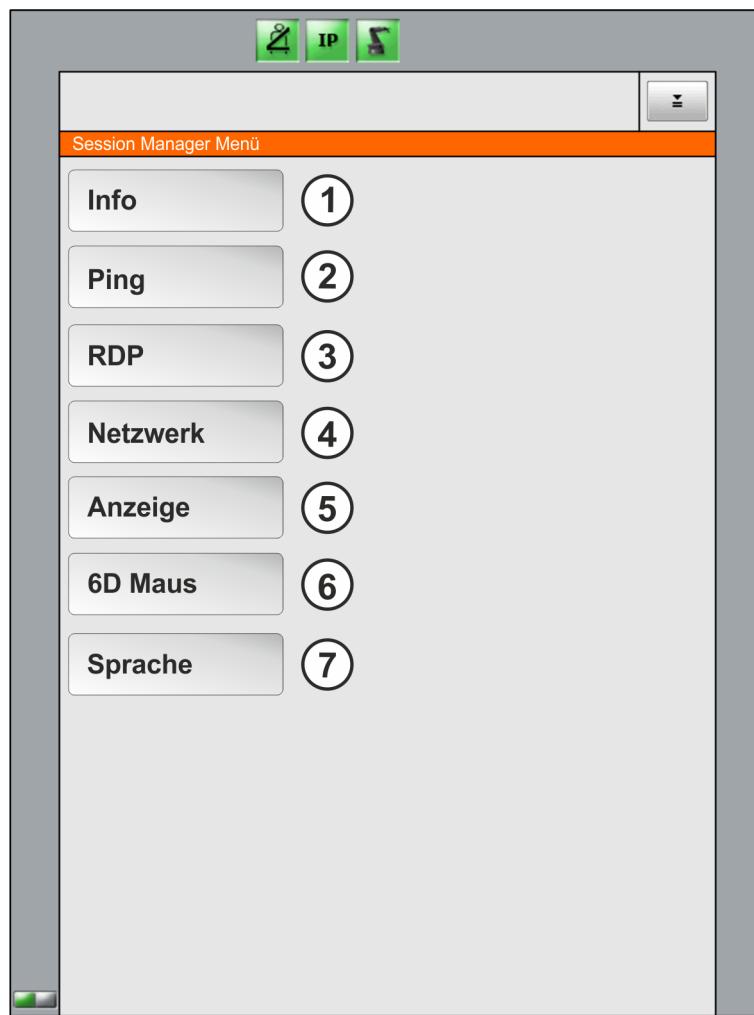


Abb. 1-20: Session Manager Menü

1	Info Übersicht der Versionsstände (smartPAD)
2	Ping Verbindungstest vom smartPAD zum KPC
3	RDP IP- und LOG-IN Daten vom smartPAD zum KPC
4	Netzwerk verwendete Netzwerkbereiche der Steuerung
5	Anzeige Kalibrierung und allgemeine Bildschirmstellungen
6	6D-Maus Kalibrierung und Test der 6D-Maus
7	Sprache Sprachumstellung Englisch/Deutsch

Anzeigeeinstellungen

1. Über den Menüpunkt Anzeige (5) öffnet sich die Anzeigeeinstellung. In diesem Fenster kann der Bildschirmhelligkeit angepasst sowie der Touch-Screen kalibriert werden.

2. Geänderte Einstellungen werden mit der Taste **Speichern** (4) übernommen.



Abb. 1-21: Anzeigeeinstellungen

- 1 Helligkeit
- 2 Hintergrundbeleuchtung
- 3 Touchscreenkalibrierung
- 4 Einstellungen speichern

1.6 Roboterprogrammierung

Beschreibung

Durch die Programmierung des Roboters wird erreicht, dass Bewegungsabläufe und Prozesse automatisch und immer wiederkehrend abgearbeitet werden können.

Dazu benötigt die Steuerung eine Vielzahl von Informationen:

- Aktuelle Roboterposition = Position des aktuellen Werkzeugs (Tool) im aktuellen Raum (Basis)
- Art der Bewegung
- Geschwindigkeit/Beschleunigung
- Signalinformationen für z. B. Wartebedingungen, Verzweigungen oder Schleifen

Welche Sprache spricht die Steuerung?

- Einfache Programme werden mit vordefinierten Formularen (Inlineformulare) erstellt.

Beispiel-Inlineformular

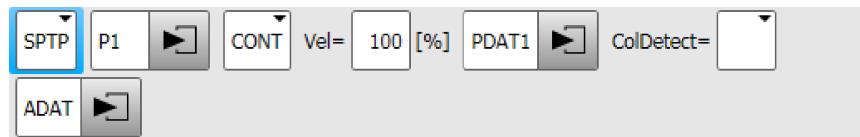


Abb. 1-22: Inline Formular

- Für Schleifen, Logik und Programmierung kann die Programmiersprache **KRL - KUKA Robot Language** verwendet.

Beispielprogramm:

```

LOOP
SPTP P1 Vel=100% PDAT1 Tool[2] Base[4]
SPTP P2 Vel=100% PDAT2 Tool[2] Base[4]
SPTP P3 Vel=100% PDAT3 Tool[2] Base[4]
SLIN P4 Vel=0.5m/s CPDATA4 Tool[2] Base[4]
...
ENDLOOP

```

Wie wird ein KUKA Roboter programmiert?

Zur Programmierung eines KUKA Roboters können verschiedene Programmiermethoden eingesetzt werden:

- Online-Programmierung** mit dem Teach-in-Verfahren.



Abb. 1-23: Roboterprogrammierung mit dem KUKA smartPAD

- Programmieren mit WorkVisual:** Mit der Softwareumgebung WorkVisual können Programme erstellt und online auf der Steuerung getestet

werden. Für die Fehlersuche stehen Debugging-Werkzeuge zu Verfügung.

Alternativ ist auch die Offline-Programmierung im Rahmen eines Roboterprojektes möglich.

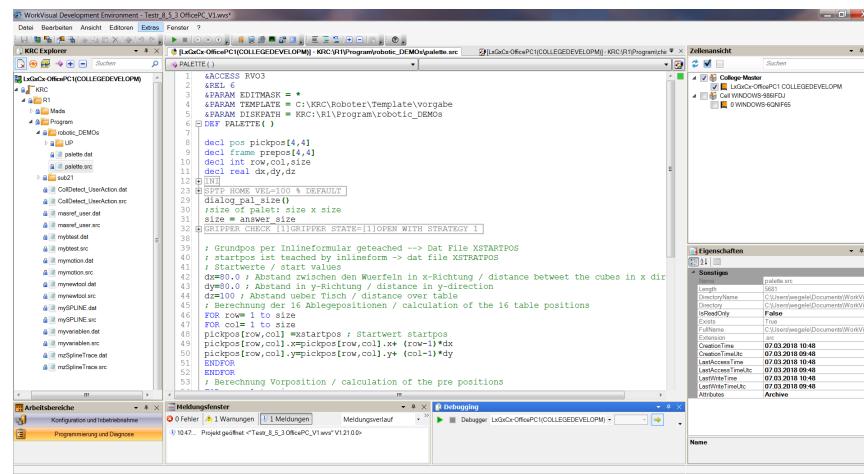


Abb. 1-24: Programmieren mit WoV

- **Offline-Programmierung**

- **Grafisch-interaktive Programmierung:** Simulation des Roboterprozesses



Abb. 1-25: Simulation mit KUKA Sim

1.7 Robotersicherheit

Beispiel: College-Schulungszelle

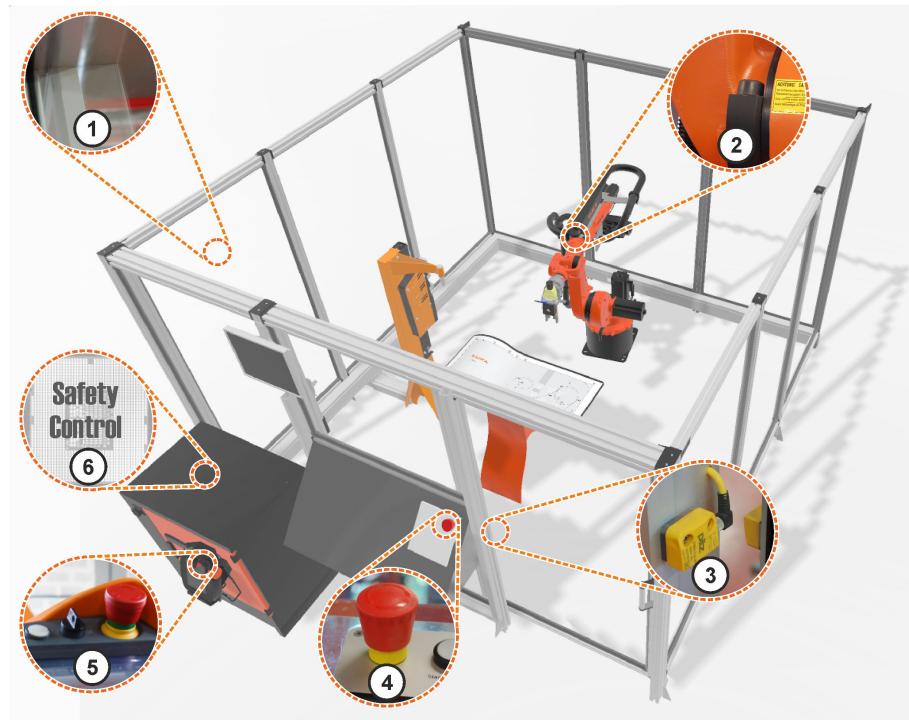


Abb. 1-26: Schulungszelle

- 1 Schutzzaun
- 2 Mechanische Endanschläge bzw. Achsbegrenzung für Achse 1, 2 und 3
- 3 Schutztür mit Türkontakt zur Überwachung der beweglichen Schutzeinrichtung
- 4 NOT-HALT-Einrichtungen (extern)
- 5 NOT-HALT-Einrichtung, Zustimmungsschalter, Schlüsselschalter zum Aufruf des Verbindungsmanagers
- 6 Integrierte (V)KR-C4-Sicherheitssteuerung



Je nach Zellenkonfiguration werden die sicheren Eingänge (Pos. 3 und 4) von einer übergeordneten SPS erfasst und an die (V)KR C4 in sicherer Technik übertragen.

Aufgaben der Sicherheitssteuerung



GEFAHR

- Das Robotersystem kann ohne funktionsfähige Sicherheits- und Schutzeinrichtungen Personen- oder Sachschaden verursachen.
- Wenn Sicherheits- oder Schutzeinrichtungen demontiert oder deaktiviert sind, darf das Robotersystem nicht betrieben werden.

NOT-HALT Einrichtung

Lokaler NOT-HALT

- Die NOT-HALT-Einrichtung des Industrieroboters ist der NOT-HALT-Taster am smartPAD.

- Der Taster muss bei einer gefahrbringenden Situation oder im Notfall gedrückt werden.

Externer NOT-HALT

- Es muss immer mindestens eine externe NOT-HALT-Einrichtung installiert werden. (>>>> "Externer NOT-HALT" Seite 33)
- Dies stellt sicher, dass auch bei abgestecktem smartPAD eine NOT-HALT-Einrichtung zur Verfügung steht.

Reaktionen des Industrieroboters, wenn der NOT-HALT-Taster gedrückt wird:

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

Betrieb fortsetzen

- Um den Betrieb fortsetzen zu können, muss der NOT-HALT-Taster durch Drehen entriegelt und die anschließend auftretende Meldung quittiert 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.

Externer NOT-HALT

- An jeder Bedienstation, die eine Roboterbewegung oder eine andere gefahrbringende Situation auslösen kann, müssen NOT-HALT-Einrichtungen zur Verfügung stehen. 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.

Bedienerschutz

- Das Signal **Bedienerschutz** dient zur Verriegelung beweglich, trennender Schutzeinrichtungen, z. B. Schutztüren.
- Ohne dieses Signal ist kein Automatikbetrieb möglich.
- Bei einem Signalverlust während des Automatikbetriebs (z. B. Schutztür wird geöffnet) stoppt der Manipulator mit einem Sicherheitshalt 1.
- In den Test-Betriebsarten T1 und 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.

Sicherer Betriebshalt

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

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, solange das sichere, externe Signal FAL-SE ist.
- Wenn das sichere, externe Signal TRUE ist, kann der Manipulator wieder verfahren werden.
- Es ist keine Quittierung notwendig.

2 Dokumentation des Robotersystems

2.1 Lerneinheit: Dokumentation des Robotersystems

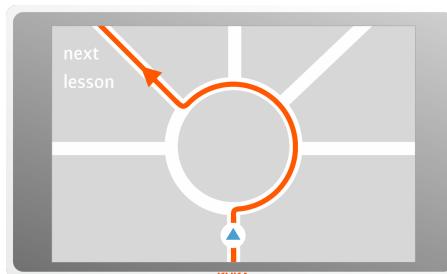


Abb. 2-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Vorstellung KUKA Xpert
- Registrierung
- Nutzung von KUKA Xpert Basic

2.2 KUKA Xpert

Beschreibung



KUKA

Xpert

Abb. 2-2

KUKA Xpert ist eine semantische Wissensdatenbank von KUKA, die technische Informationen und Dokumentationen zu allen KUKA Produkten enthält. Aufbauend auf semantisch verknüpften Daten unterstützt das System bei der Beantwortung von Fragen zu KUKA Produkten.

Alle Informationen zu einem Thema sind auf einen Blick sichtbar. Über verschiedene Filter lassen sich die Suchergebnisse einschränken. Eine Freitextsuche ist jederzeit möglich.



Die Verwendung der in KUKA Xpert befindlichen Informationen erfolgt unter alleiniger Verantwortung und auf eigene Gefahr des Nutzers. Schadensersatzansprüche jeglicher Art gegen KUKA sind ausgeschlossen.

Lizenzen

Für KUKA Xpert stehen verschiedene Lizenzpakete zur Verfügung, die untereinander kombinierbar sind. Die folgende Tabelle zeigt, welche Informationstypen in den jeweiligen Lizizenzen enthalten sind:

KUKA Xpert Basic	<p>Enthält technische Dokumentationen für KUKA Produkte wie:</p> <ul style="list-style-type: none"> • Betriebs- und Montageanleitungen • Bedien- und Programmieranleitungen • Spezifikationen • Sicherheit • Grundlegende Produktinformationen • Ersatzteillinformationen <p>KUKA Xpert Basic ist in allen anderen Lizenzen bereits enthalten.</p>
KUKA Xpert Basic Plus	Enthält zusätzlich zu den Inhalten aus KUKA Xpert Basic Zugriff auf kundenspezifische Dokumente.
KUKA Xpert	<p>Enthält uneingeschränkten Zugriff auf alle Inhalte von KUKA Xpert. Dies entspricht:</p> <ul style="list-style-type: none"> • Falldatenbank mit über 20.000 Meldungen samt Ursachen und Lösungen • Über 500 Arbeitsanweisungen z. B. zum Tausch von Ersatzteilen, Programmierung und Justage • Über 300 Systemvariablen für KSS 5 - 8.5 • Wartungsinfos wie Ölmengen, Drücke und Zahndrehmomente zu jedem Roboter • Guidelines wie Step-by-Step Anleitungen zur Realisierung von bestimmten Funktionen in Bussystemen • Funktionsbeschreibungen wie z. B. Erklärungen zu Parametern von Funktionen in Bussystemen • Downloads wie Codes oder Konfigurationsdateien • Prozesskräfte • Flanschlasten • Genauigkeitsdaten • Kompatibilitätsübersichten

2.3 KUKA Xpert URL



Es wird nur die Basic-Lizenz auszugsweise dargestellt. Andere Lizenzmodelle sind kein Bestandteil dieser Schulung.

Beschreibung

Mit den Begleitdokumenten eines KUKA Produkts wird ein Beiblatt KUKA Xpert URL mitgeliefert. Auf ihm befinden sich Links, die zur Dokumentation und weiteren Informationen der einzelnen Systemkomponenten (Roboter/Steuerung/Software) führen.

KUKA Xpert Basic enthält folgende Informationen:

- Technische Dokumentation
- Produktinformationen
- Ersatzteillinformationen

Zur Nutzung von KUKA Xpert Basic ist ein my.KUKA Login notwendig.

Das Beiblatt KUKA Xpert URL ist wie folgt aufgebaut:

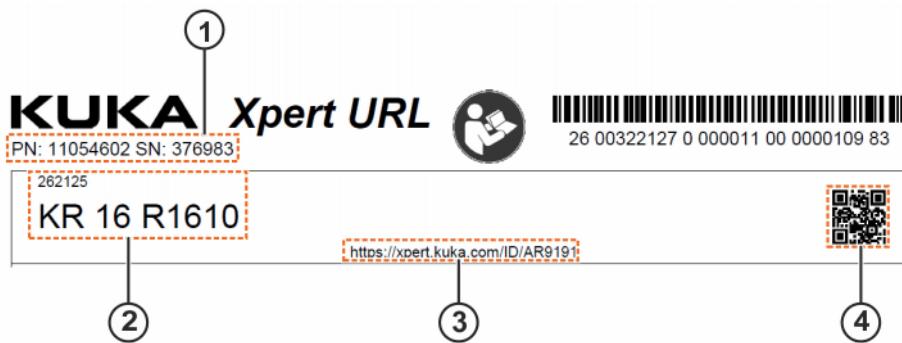


Abb. 2-3: Beiblatt KUKA Xpert URL

Pos.	Beschreibung
1	PN: Produktnummer SN: Seriennummer
2	Artikelnummer und Produktbezeichnung
3	KUKA Xpert URL
4	KUKA Xpert URL als QR-Code

2.4 Registrierung

Die Registrierung erfolgt im my.KUKA Portal. Das my.KUKA Portal ist ein globales KUKA-Kundenportal und der digitale Zugangspunkt für Unternehmen. Über das Portal können Kunden spezifische Zusatzinformationen, digitale Funktionen und Dienstleistungen finden. Falls noch kein my.KUKA Login vorhanden ist, muss ein neuer Account angelegt werden.



Es nur möglich, sich mit seiner Firmen E-Mail-Adresse zu registrieren. Eine Registrierung mit einer privaten E-Mail-Adresse ist ausgeschlossen.

Vorgehensweise

- Den Link auf dem Beiblatt eingeben oder den QR-Code scannen. Alternativ hier klicken: <https://xpert.kuka.com>.
- Auf **Sie haben noch keinen my.KUKA Account?** klicken, um einen neuen Account zu erstellen.

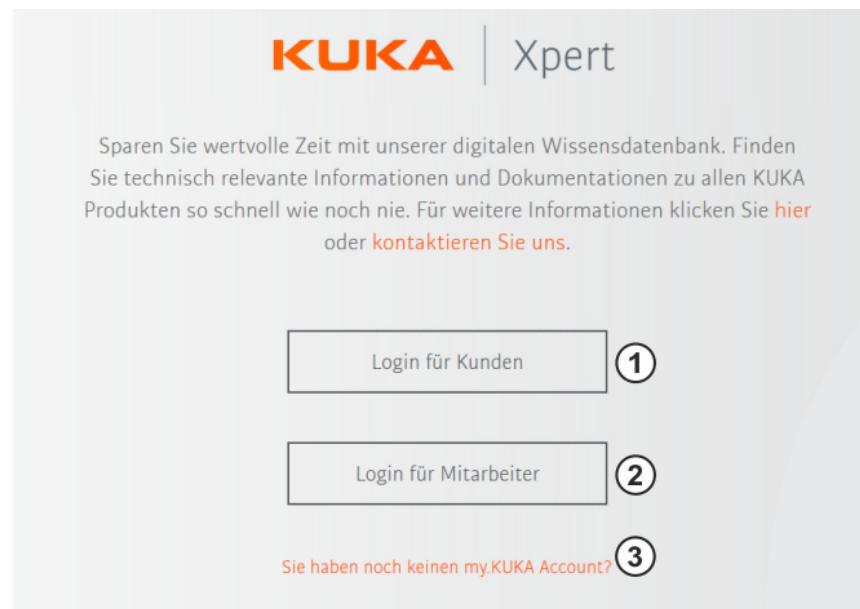


Abb. 2-4

Pos.	Erklärung
1	Login für Kunden
2	Login für KUKA Mitarbeiter
3	Weiterleitung zum Registrierungsformular

3. Registrierungsformular ausfüllen und Datenschutzbedingungen sowie Nutzungsbedingungen akzeptieren. Eine Registrierungsemail wird an die angegebene Adresse gesendet.

Abb. 2-5

4. In der zugesendeten E-Mail auf die Schaltfläche **Ihr Konto überprüfen** klicken. Eine Webseite öffnet sich.
 5. Passwort festlegen und weitere Informationen ausfüllen.

2.5 Anmeldung

Vorgehensweise

1. Den Link <https://xpert.kuka.com/Account/Login#/> eingeben.
2. Auf **Login für Kunden** klicken, um sich einzuloggen.



Abb. 2-6

Pos.	Erklärung
1	Login für Kunden
2	Login für KUKA Mitarbeiter
3	Weiterleitung zum Registrierungsformular

3. Auf **Login** klicken oder E-Mail-Adresse und Passwort eingeben und auf **Login** klicken.<https://xpert.kuka.com/Account/Login#/>

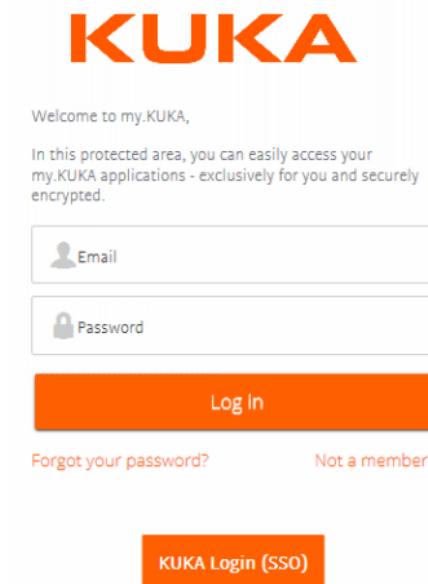


Abb. 2-7

2.6 KUKA Xpert Basic nutzen

Beschreibung

KUKA Xpert Basic enthält die gesamte technische Dokumentation sowie grundlegende Produktinformationen und Ersatzteilinformationen.

Übersicht

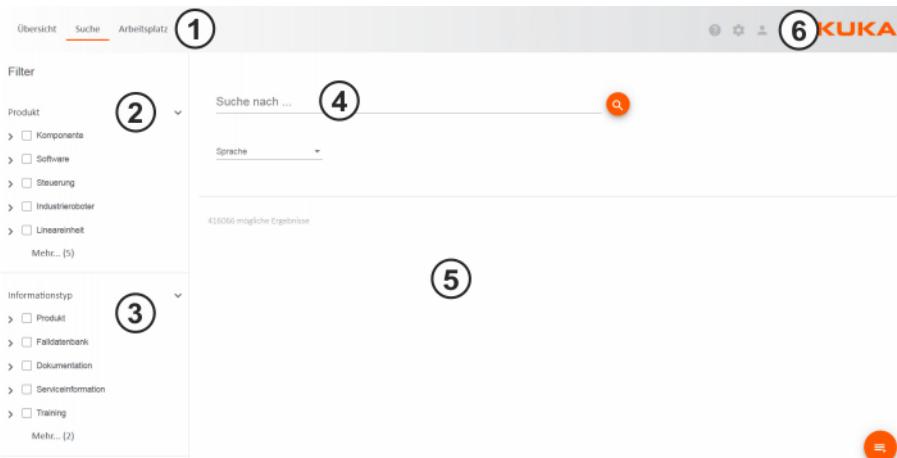


Abb. 2-8: KUKA Xpert Übersicht

Pos.	Erklärung
1	Übersicht: Festlegung der Sitzungen Suche: Suchbereich Arbeitsplatz: Historie der letzten Suchen
2	Produkt: Setzen der einzelnen Filter zu dem jeweiligen Produkt. Eine Mehrfachauswahl ist möglich.
3	Informationstyp: Setzen der einzelnen Filter zu dem jeweiligen Informationstyp. Das Suchergebnis kann nachträglich gefiltert werden. Abhängig von den bereits gewählten Filtern oder Suchergebnissen werden unter Umständen weitere, dynamische Filter angeboten.
4	Freitextsuche
5	Zeigt die Suchbegriffe in ihrem Kontext innerhalb des Suchergebnisses an.
6	Hilfe Einstellungen Abmelden

2.7 Übung: KUKA Xpert Basic nutzen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und setzen Sie die vom Trainer angegebene Aufgabenstellung um.

- Übung: KUKA Xpert Basic nutzen

Aufgabenstellung	
------------------	--

3 Roboter bewegen

3.1 Lerneinheit: Roboter bewegen

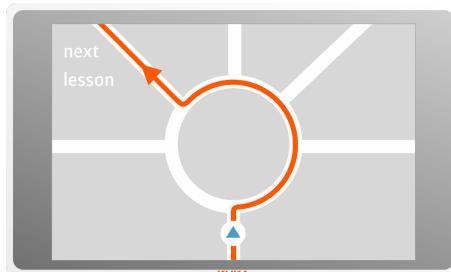


Abb. 3-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Meldungen lesen und interpretieren
- Betriebsarten auswählen und einstellen
- Roboterachsen einzeln bewegen
- Koordinatensysteme im Zusammenhang mit Robotern
- Roboter im Weltkoordinatensystem bewegen
- Roboter mittels Verfahrtart Spur rückwärts und wieder vorwärts bewegen
- Anzeige der Roboterposition

3.2 Meldungen der Robotersteuerung lesen und interpretieren

Übersicht Meldungen

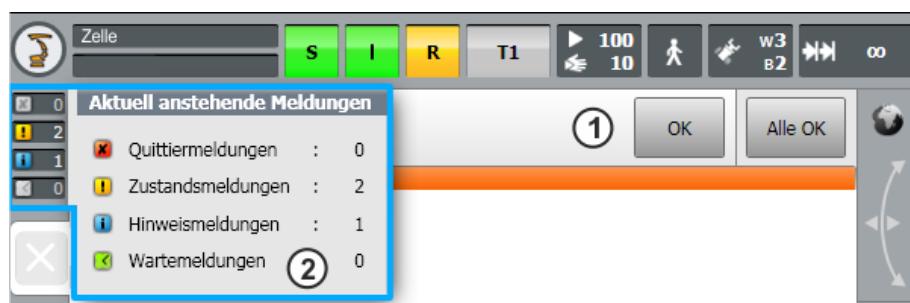


Abb. 3-2: Meldungsfenster und Meldungszähler

1	Meldungsfenster: Anzeige der aktuellen Meldung
2	Meldungszähler: Anzahl der Meldungen je Meldungstyp

Über das Meldungsfenster kommuniziert die Steuerung mit dem Bediener. Dabei verfügt sie über fünf verschiedene Meldungstypen:

Übersicht Meldungstypen:

Symbol	Typ
	Quittiermeldung <ul style="list-style-type: none"> Zur Darstellung von Zuständen, bei der zur weiteren Abarbeitung des Roboterprogramms eine Bestätigung des Bedieners erforderlich ist (z. B. "Quit NOT-HALT"). Eine vom System ausgegebene Quittiermeldung hat immer zur Folge, dass der Roboter anhält oder nicht startet.
	Zustandsmeldung <ul style="list-style-type: none"> Zustandsmeldungen melden aktuelle Zustände der Steuerung (z. B. "NOT-HALT"). Zustandsmeldungen sind nicht quittierbar, solange der Zustand ansteht.
	Hinweismeldung <ul style="list-style-type: none"> Hinweismeldungen geben Informationen zur korrekten Bedienung des Roboters (z. B. "Start-Taste erforderlich"). Hinweismeldungen sind quittierbar. Sie müssen aber nicht quittiert werden, da das angewählte Roboterprogramm nicht angehalten wird, oder das Handverfahren des Roboters weiterhin möglich ist.
	Wartemeldung <ul style="list-style-type: none"> Wartemeldungen geben an, auf welches Ereignis (Zustand, Signal oder Zeit) die Steuerung wartet. Wartemeldungen können mit Drücken der Schaltfläche "Simuliere" manuell abgebrochen werden.
	Dialogmeldung <ul style="list-style-type: none"> Dialogmeldungen dienen als Abfrage innerhalb Programmkontrollstrukturen oder für Systeminteraktionen. Ein Meldungsfenster mit Schaltflächen wird angezeigt. Diese bieten verschiedene Antwortmöglichkeiten.
	Mit OK kann eine quittierbare Meldung quittiert werden. Mit Alle OK können alle quittierbaren Meldungen auf einmal quittiert werden.
HINWEIS	
Die Schaltfläche "Simuliere" darf nur verwendet werden, wenn Kollision und sonstige Gefahren ausgeschlossen werden können!	

Einfluss von Meldungen

- Meldungen beeinflussen die Funktionalität des Roboters.
- Eine vom System ausgegebene Quittiermeldung hat immer zur Folge, dass der Roboter anhält oder nicht startet.
- Die Meldung muss dann erst quittiert werden, um den Roboter zu bewegen.
- Das Kommando "**OK**" (quittieren) drückt eine Aufforderung an den Bediener aus, sich mit der Meldung bewusst auseinander zu setzen.



Tipps zum Umgang mit Meldungen:

- Bewusst lesen!
- Ältere Meldungen zuerst lesen. Eine neuere Meldung könnte eine Folge einer älteren sein.
- Nicht einfach die Schaltfläche **Alle OK** drücken.
- Besonders nach dem Hochfahren: Meldungen durchsehen. Dabei alle Meldungen anzeigen lassen. Durch Drücken auf das Meldungsfenster expandiert die Meldungsliste.

Umgang mit Meldungen

Meldungen werden immer mit Datum und Uhrzeit ausgegeben, um den genauen Zeitpunkt des Ereignisses nachvollziehen zu können.



Abb. 3-3: Meldungen quittieren

Vorgehensweise zum Betrachten und Quittieren von Meldungen:

1. Das Meldungsfenster (1) berühren, um die Meldungsliste einzublenden.
2. Quittieren:
 - Mit **OK** (2) einzelne Meldungen quittieren,
 - alternativ: Mit **Alle OK** (3) alle Meldungen quittieren,
3. ein weiteres Berühren der obersten Meldung oder ein Berühren des "X" am linken Bildschirmrand reduziert die Meldungsliste wieder.

Hilfe

Ist eine Meldung für den Benutzer unschlüssig oder Bedarf diese einer näheren Erklärung, so kann, abhängig vom Meldungstyp, eine zusätzliche Hilfe abgerufen werden.

1. Neben der Meldung auf die Schaltfläche mit dem Fragezeichen (1) tippen.

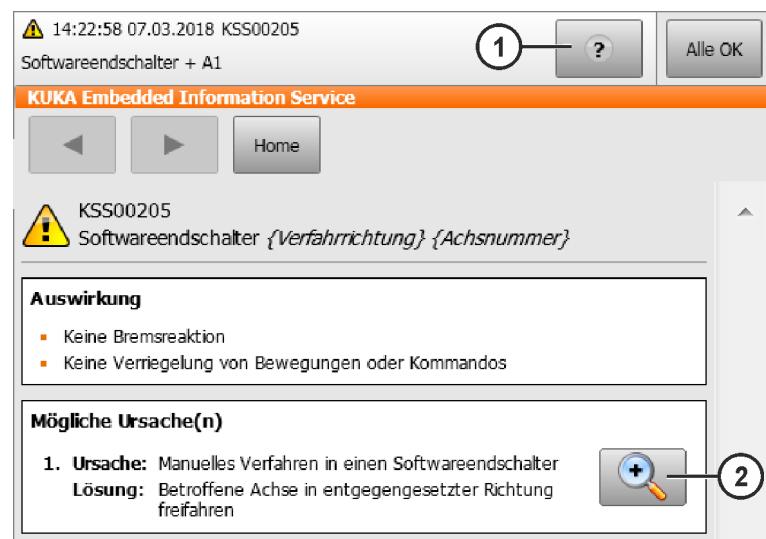


Abb. 3-4: Meldung, Hilfe

- Es wird im Hauptfenster die Auswirkung der Meldung auf das Programm oder das Robotersystem beschrieben.
- Im Fenster Mögliche Ursachen wird eine oder mehrere (mögliche) Lösung/-en vorgeschlagen.
- Weitere Informationen können über die Schaltfläche Lupe (2) aufgerufen werden.

2. Detailseite

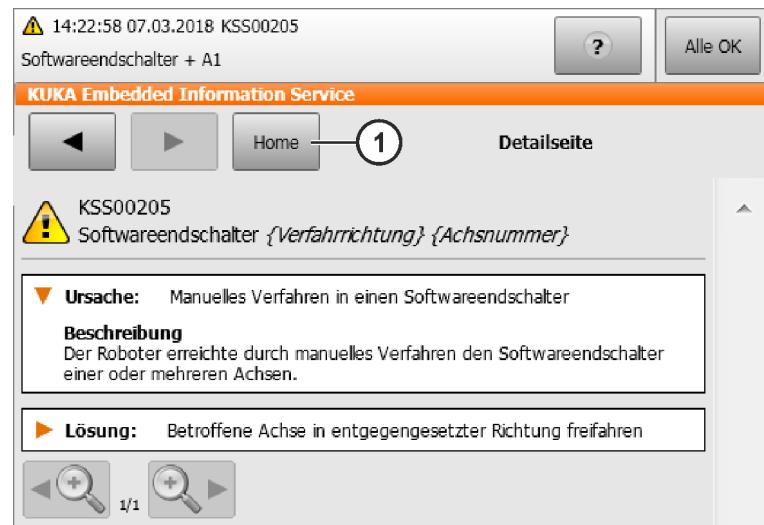


Abb. 3-5: Hilfe, Detailseite

- Über verschiedene Schaltflächen (Lupen und Pfeile) kann innerhalb des Hilfetexts navigiert werden.
- Über die Schaltfläche Home (1) gelangt man auf die erste Seite des Hilfetexts zurück.

3.3 Betriebsarten der Robotersteuerung

Betriebsarten eines KUKA Roboters



Abb. 3-6: Betriebsarten



Die Zuordnung, welche Betriebsart der jeweiligen Benutzergruppe zur Verfügung steht, kann über Benutzergruppe Administrator zugeordnet werden. Mitunter stehen nicht alle Betriebsarten zur Verfügung.

- **T1 (Manuell Reduzierte Geschwindigkeit)**
 - Für Testbetrieb, Programmierung und Teachen
 - Geschwindigkeit im Programmbetrieb max. 250 mm/s
 - Geschwindigkeit im Handbetrieb max. 250 mm/s
- **T2 (Manuell Hohe Geschwindigkeit)**
 - Für Testbetrieb
 - Geschwindigkeit im Programmbetrieb entsprechend der programmierten Geschwindigkeit!
 - Diese kann vom Benutzer durch den POV - *Program-Override* reduziert werden.
 - Bewegen des Roboters mittels 6D-Maus oder Verfahrtasten ist nicht möglich.
- **AUT** (Automatik)
 - Für Industrieroboter ohne übergeordnete Steuerung
 - Geschwindigkeit im Programmbetrieb entsprechend der programmierten Geschwindigkeit!
 - Diese kann vom Benutzer durch den POV - *Program-Override* reduziert werden.
 - Bewegen des Roboters mittels 6D-Maus oder Verfahrtasten ist nicht möglich.
- **AUT EXT** (Automatik Extern)
 - Für Industrieroboter mit übergeordneter Steuerung (SPS)
 - Geschwindigkeit im Programmbetrieb entsprechend der programmierten Geschwindigkeit!
 - Diese kann vom Benutzer durch den POV - *Program-Override* reduziert werden.

- Bewegen des Roboters mittels 6D-Maus oder Verfahrtasten ist nicht möglich.

Manuell Reduzierte Geschwindigkeit (T1)

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

- Bedienerschutz (Schutztür) wird nicht überwacht.
 - Die Anzahl der Personen welche sich im durch Schutzeinrichtung abgegrenzten Raum aufhalten, ist auf ein Minimum zu reduzieren.
- Wenn es notwendig ist, dass sich mehrere Personen im durch Schutzeinrichtungen abgegrenzten Raum aufhalten, muss Folgendes beachtet werden:
- Alle Personen müssen ungehinderte Sicht auf das Robotersystem haben.
 - Zwischen allen Personen muss immer die Möglichkeit zum Blickkontakt bestehen.
- Der Bediener muss eine Position einnehmen, aus der er den Gefahrenbereich einsehen kann und einer Gefahr ausweichen kann.

Manuell Hohe Geschwindigkeit (T2)

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

- Bedienerschutz (Schutztür) wird nicht überwacht.
-  In den KUKA College Schulungszellen wird jedoch, abweichend vom Standard, die Schutztür überwacht und muss geschlossen sein. Alle Personen befinden sich außerhalb des durch Schutzeinrichtungen abgegrenzten Raums.
- Diese Betriebsart darf nur verwendet werden, wenn die Anwendung einen Test mit höherer Geschwindigkeit erfordert als mit der manuell reduzierten Geschwindigkeit.
 - Handverfahren ist in dieser Betriebsart nicht möglich.
 - Durch das Abfahren von Programmen im T2 Betrieb wird die programmierte Geschwindigkeit erreicht.
Diese kann mittels des *POV - Program-Override* reduziert werden.
 - Der Roboterbediener, sowie weitere Personen, müssen eine Position außerhalb des Gefahrenbereichs einnehmen.

Betriebsarten Automatik und Automatik Extern

- Sicherheits- und Schutzeinrichtungen müssen vorhanden und voll funktionsfähig sein.
- Alle Personen befinden sich außerhalb des durch Schutzeinrichtungen abgegrenzten Raums.

3.3.1 Stopp-Reaktionen

Beschreibung

- Stopp-Reaktionen des Industrieroboters werden aufgrund von Bedienhandlungen oder als Reaktion auf Überwachungen und Fehlermeldungen ausgeführt.
- Die folgenden Tabellen zeigen die Stopp-Reaktionen in Abhängigkeit 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	
Robotersteuerung abschalten (Spannungsausfall)	STOP 0	
Interner Fehler im nicht-sicherheitsgerichteten Teil der Robotersteuerung	STOP 0 oder STOP 1 (abhängig von der Fehlerursache)	
Betriebsart wechseln während des Betriebs	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	

Begriffsbestimmung

Begriff	Beschreibung
Sicherer Betriebshalt	<p>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.</p> <p>Der sichere Betriebshalt kann auch extern ausgelöst werden.</p> <p>Wenn ein sicherer Betriebshalt ausgelöst wird, meldet die Robotersteuerung einen sicheren Ausgang (X13, SIB Extended) oder alternativ über ein sicheres Feldbusprotokoll zu einer übergeordneten Steuerung. Der Ausgang wird auch dann gesetzt, wenn zum Zeitpunkt des Auslösens nicht alle Achsen stillstanden und somit ein Sicherheitshalt STOP 0 ausgelöst wird.</p>
Sicherheitshalt STOP 0	<p>Ein Stopp, der von der Sicherheitssteuerung ausgelöst und durchgeführt wird. Die Sicherheitssteuerung schaltet sofort die Antriebe und die Spannungsversorgung der Bremsen ab.</p> <p>Hinweis: Dieser Stopp wird im Dokument als Sicherheitshalt 0 bezeichnet.</p>

Begriff	Beschreibung
Sicherheitshalt STOP 1	<p>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.</p> <p>Wenn ein Sicherheitshalt STOP 1 ausgelöst wird, setzt die Robotersteuerung einen Ausgang über den Feldbus.</p> <p>Der Sicherheitshalt STOP 1 kann auch extern ausgelöst werden.</p> <p>Hinweis: Dieser Stopp wird im Dokument als Sicherheitshalt 1 bezeichnet.</p>
Sicherheitshalt STOP 2	<p>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.</p> <p>Wenn ein Sicherheitshalt STOP 2 ausgelöst wird, setzt die Robotersteuerung einen Ausgang über den Feldbus.</p> <p>Der Sicherheitshalt STOP 2 kann auch extern ausgelöst werden.</p> <p>Hinweis: Dieser Stopp wird im Dokument als Sicherheitshalt 2 bezeichnet.</p>
Stopp-Kategorie 0	<p>Die Antriebe werden sofort abgeschaltet und die Bremsen fallen ein. Der Manipulator und die Zusatzachsen (optional) bremsen bahnnah.</p> <p>Hinweis: Diese Stopp-Kategorie wird im Dokument als STOP 0 bezeichnet.</p>
Stopp-Kategorie 1	<p>Nach 1 s werden die Antriebe abgeschaltet und die Bremsen fallen ein.</p> <p>Der Manipulator und die Zusatzachsen (optional) bremsen bahntreu.</p> <p>Hinweis: Diese Stopp-Kategorie wird im Dokument als STOP 1 bezeichnet.</p>
Stopp-Kategorie 2	<p>Die Antriebe werden nicht abgeschaltet und die Bremsen fallen nicht ein. Der Manipulator und die Zusatzachsen (optional) bremsen mit einer bahntreuen Bremsrampe.</p> <p>Hinweis: Diese Stopp-Kategorie wird im Dokument als STOP 2 bezeichnet.</p>

3.3.2 Betriebsart wechseln

Vorgehensweise



Wenn die Betriebsart während des Betriebs gewechselt wird, werden die Antriebe sofort abgeschaltet. Der Industrieroboter stoppt mit einem Sicherheitshalt 2.

- Am smartPAD den Schalter für den Verbindungsmanager umlegen. Der Verbindungsmanager wird angezeigt.



- Die Betriebsart wählen.

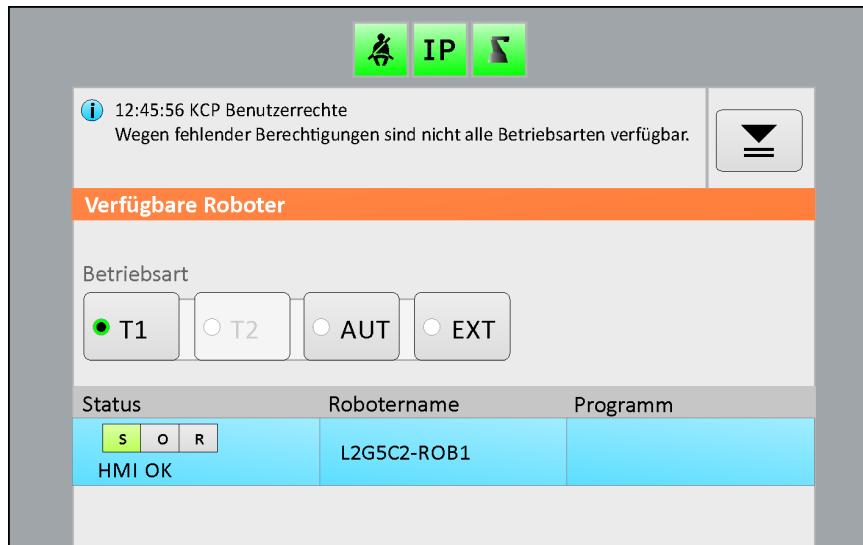


Abb. 3-7: Betriebsarten

- Die gewünschte Betriebsart (T1, T2, AUT, EXT) über die jeweilige Schaltfläche antippen.
 - Die gewählte Betriebsart wird mit einem grünen Punkt markiert.
- Den Schalter für den Verbindungsmanager wieder in die ursprüngliche Position bringen.

Die gewählte Betriebsart wird in der Statusleiste des smartPADs angezeigt.



i Die Betriebsart **AUT** ist **nicht** in der VSS - Volkswagen System Software verfügbar.

i Ab der KSS 8.5 können Betriebsarten durch das Rechtemanagement den jeweiligen Benutzergruppen gezielt zugewiesen als auch gesperrt werden.

3.4 Roboterachsen einzeln bewegen

Roboterachsen achsspezifisch verfahren

Jede Achse eines Roboters kann einzeln in Plus- und Minusrichtung verfahren werden. Dazu werden Verfahrtasten oder die 6D-Maus des KUKA smartPADs verwendet. Die Verfahrgeschwindigkeit kann mit dem HOV-Hand Override verringert werden. Das Handverfahren ist nur in der Be-

triebsart T1 möglich, dazu muss einer der Zustimmtaster in Mittelstellung gehalten werden.

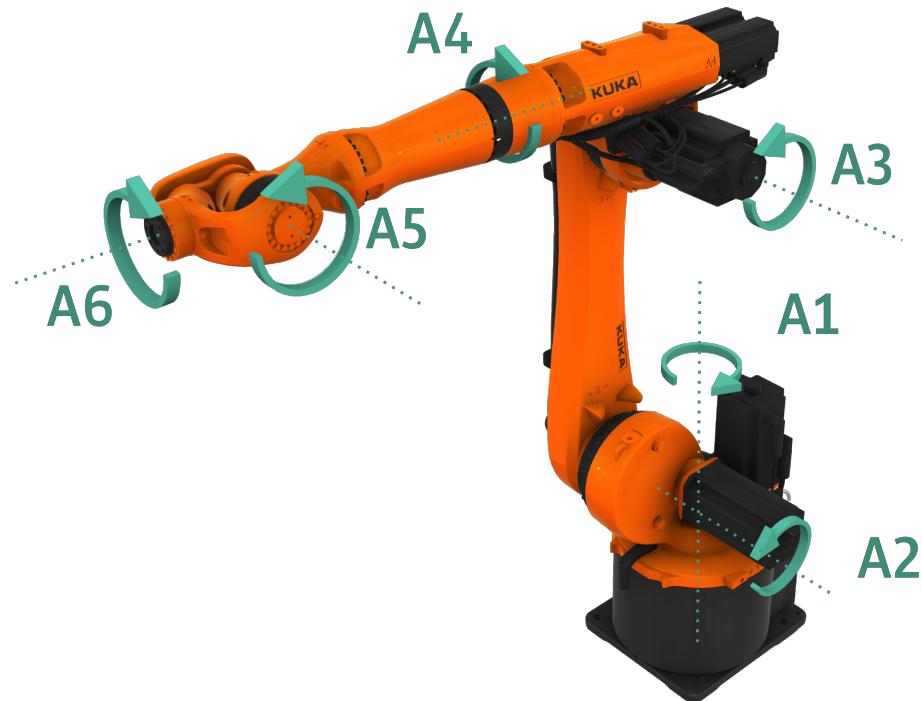


Abb. 3-8: Achsen eines KUKA Roboters



Pfeile in der Darstellung zeigen jeweils in die positive Verfahrrichtung (+).

Prinzip

- Über die Betätigung einer der Zustimmtasten werden die Antriebe aktiviert. Wenn die Antriebsfreigabe vorhanden ist, wird die Beschriftung der Verfahrtasten grün. Sobald eine Verfahrtaste oder die 6D-Maus betätigt wird, startet die Regelung der Roboterachsen und die gewünschte Bewegung wird durchgeführt.
- Es gibt die Möglichkeit der ständigen Bewegung sowie der inkrementellen Bewegung. Dazu ist in der Statusleiste das Schrittmaß (Inkrementgröße) auszuwählen. (>>> ["Vorgehensweise" Seite 55](#))

Folgende Meldungen beeinflussen den manuellen Betrieb:

Meldung	Ursache	Abhilfe
"Aktive Kommandos verriegelt"	Es steht eine (STOP-)Meldung oder ein Zustand an, der eine Verriegelung aktiver Kommandos bewirkt (z. B. NOT-HALT gedrückt oder Antriebe noch nicht bereit).	NOT-HALT entriegeln und Meldungen im Meldungsfenster quittieren. Nach dem Drücken eines Zustimmungsschalters stehen die Antriebe zur Verfügung.
"Software-Endschalter - A5"	Der Software-Endschalter der angezeigten Achse (z. B. A5) wurde in der angegebenen Richtung (+ oder -) angefahren.	Die angezeigte Achse in die Gegenrichtung verfahren.

Praxistipp

- Das gewünschte Programm anwählen.
- Einen der Zustimmungsschalter betätigen.

3. Warten, bis die Antriebe bereit sind (Verfahrtasten leuchten grün, und Anzeige Antriebe I grün).
4. Dann gewünschte Aktion starten » Start-Taste, Verfahrtasten oder 6D-Maus.



Wird diese Reihenfolge nicht eingehalten, wird die Meldung "Aktive Kommandos verriegelt" angezeigt.

Sicherheitshinweise zum achsspezifischen Handverfahren

Betriebsart

- Der Handbetrieb des Roboters ist nur in der Betriebsart T1 (Manuell Reduzierte Geschwindigkeit) möglich.
- Die Handverfahrgeschwindigkeit beträgt im T1-Betrieb maximal 250 mm/s.
- Die Betriebsart wird über den Verbindungsmanager eingestellt.

Zustimmungsschalter

- Um den Roboter verfahren zu können, muss ein Zustimmungsschalter betätigt und gehalten werden.
- Am smartPAD sind drei und am smartPAD2 vier Zustimmungsschalter angebracht.
- Die Zustimmungsschalter haben drei Stellungen:
 - nicht gedrückt
 - Mittelstellung
 - Durchgedrückt (Panikstellung)

Software-Endschalter

- Die Bewegung des Roboters wird durch die positiven und negativen Maximalwerte der Software-Endschalter begrenzt.

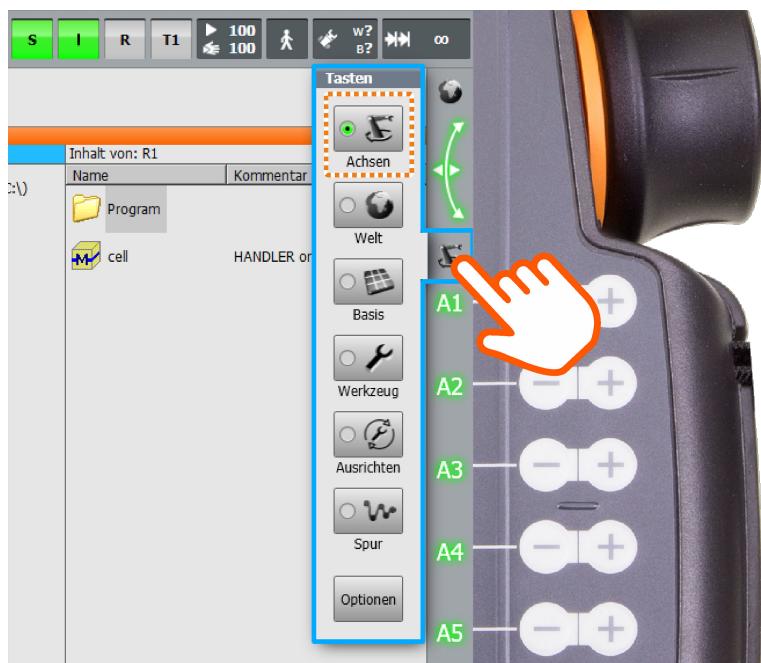
HINWEIS

- Falls im Meldungsfenster die Meldung "Justage durchführen" erscheint, kann auch über diese Grenzen gefahren werden.
- Dies kann zu Schäden am Robotersystem führen!

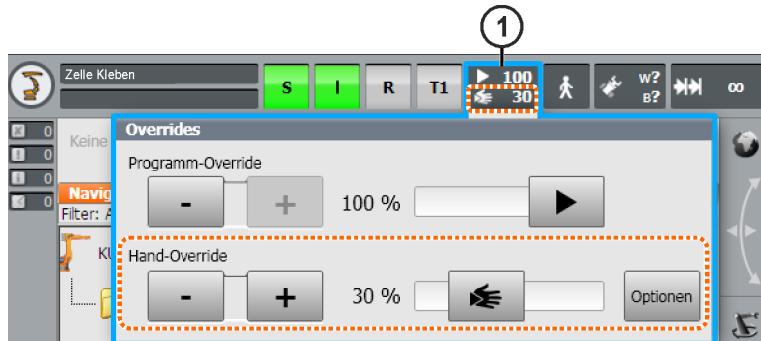
Vorgehensweise

Achsspezifische Bewegung durchführen

1. Als Option für die Verfahrtasten **Achsen** auswählen



2. Hand-Override einstellen



3. Zustimmschalter auf Mittelstellung drücken und halten

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



3.4.1 Inkrementelles Handverfahren

Beschreibung

Inkrementelles Handverfahren

- Das inkrementelle Handverfahren ermöglicht es, den Roboter um eine definierte Distanz zu bewegen, z. B. um 10 mm oder 3°.
- Das inkrementelle Handverfahren kann beim Verfahren mit den Verfahrtasten zugeschaltet werden.

Beim Verfahren mit der Space-Maus ist das inkrementelle Handverfahren nicht möglich.

- Die jeweilige Verfahrtaste muss solange betätigt bleiben, bis die voreingestellte Distanz erreicht wird. Bei Erreichen der eingestellten Distanz stoppt der Roboter selbständig.

Anwendungsbereiche:

- Positionieren von Punkten in gleichen Abständen
- exakte Punktkorrektur, wenn die Distanz in mm oder Grad bekannt ist, z. B. Daten aus einer Messvorrichtung
- Herausfahren aus einer Position um eine definierte Distanz, z. B. im Fehlerfall

Voraussetzung

- Betriebsart T1

Vorgehensweise

- In der Statusleiste die Inkrementgröße (1) auswählen:



Abb. 3-9: Inkrementelles Handverfahren

- Den Roboter kann mit den Verfahrtasten kartesisch oder achsspezifisch verfahren werden.



Wenn die Roboterbewegung unterbrochen wird, z. B. durch Loslassen des Zustimmungsschalters, wird bei der nächsten Bewegung die unterbrochene Fahrdistanz nicht fortgesetzt, sondern von neuem begonnen.

Einstellungen/Inkrementgrößen:

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°

Inkremeante in mm:

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

Inkremeante in Grad:

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

3.4.2 Roboter in Notfällen ohne Steuerung bewegen

Beschreibung

Mit der Freidreh-Vorrichtung kann der Roboter nach einem Unfall oder Störfall mechanisch bewegt werden.

Die Freidreh-Vorrichtung kann für die Grundachs-Antriebsmotoren und je nach Robotervariante auch für die Handachs-Antriebsmotoren verwendet werden.

Sie darf **nur** in Ausnahmesituationen und Notfällen, z. B. für die Befreiung von Personen, eingesetzt werden.



Abb. 3-10: Freidreh-Vorrichtung

Wenn die Freidreh-Vorrichtung verwendet wurde, muss anschließend die einwandfreie Funktion der Bremsen sichergestellt sein.

- Hierzu ist ein Bremsentest durchzuführen. Fällt dieser negativ aus, ist der betroffene Motor zu tauschen.
- Ist der Bremsentest auf der Steuerung nicht verfügbar oder kann nicht durchgeführt werden, sind die betroffenen Motoren zu tauschen.



Der Bremsentest ist eine feste Funktion der KSS8.x. Dieser muss jedoch durch den Inbetriebnehmer/ Sicherheitsinbetriebnehmer freigeschaltet und aktiviert werden.

Vorgehensweise



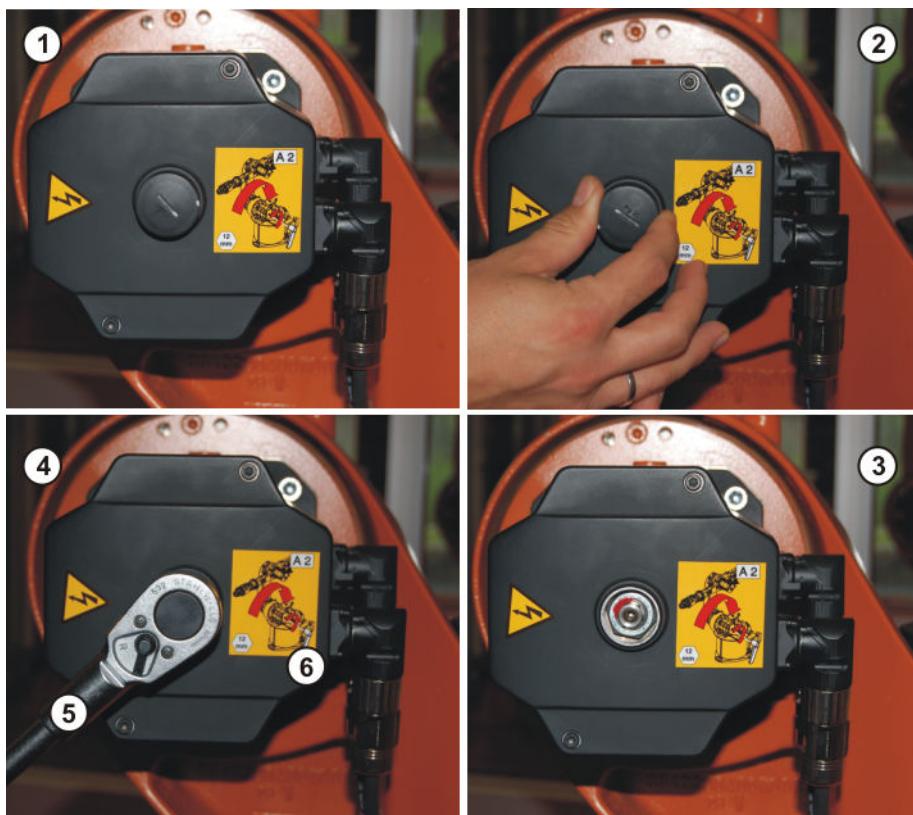
WARNUNG

- Die Motoren erreichen während des Betriebs Temperaturen, die zu Hautverbrennungen führen können.
- Berührungen sind zu vermeiden.
- Es sind geeignete Schutzmaßnahmen zu ergreifen, wie z. B. das Tragen von geeigneten Schutzhandschuhen.

1. Robotersteuerung ausschalten und gegen unbefugtes Wiedereinschalten (z. B. mit einem Vorhängeschloss) sichern.
2. Schutzkappe am Motor entfernen.
3. Freidreh-Vorrichtung auf den entsprechenden Motor aufsetzen und die Achse in die gewünschte Richtung bewegen.



- Optional bestellbar ist eine Kennzeichnung der Richtungen mit Pfeilen auf den Motoren.
- Der Widerstand der mechanischen Motorbremse und gegebenenfalls zusätzliche Achslasten sind zu überwinden.

Beispiel am Motor der Achse 2**Abb. 3-11: Vorgehensweise mit Freidreh-Einrichtung**

Pos.	Beschreibung
1	Motor A2 mit geschlossener Schutzkappe
2	Öffnen der Schutzkappe am Motor A2
3	Motor A2 mit entfernter Schutzkappe
4	Aufsetzen der Freidreh-Einrichtung auf Motor A2
5	Freidreh-Einrichtung
6	Schild (Option) mit Beschreibung der Drehrichtung

**VORSICHT**

- Beim Bewegen einer Achse mit der Freidreh-Vorrichtung kann die Motorbremse beschädigt werden.
- Es können Personen- und Sachschäden entstehen.
- Nach Benutzen der Freidreh-Vorrichtung muss ein Bremsentest durchgeführt oder der entsprechende Motor getauscht werden.



Weitere Informationen sind in der Betriebs- und Montageanleitung für den Roboter zu finden.

3.4.3 Übung: Bedienung und achsspezifisches Handverfahren

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bedienung und achsspezifisches Handverfahren**

3.5 Koordinatensysteme im Zusammenhang mit Robotern

Bei der Bedienung, Programmierung und Inbetriebnahme von Industrierobotern haben Koordinatensysteme eine große Bedeutung. In der Robotersteuerung sind folgende Koordinatensysteme definiert:

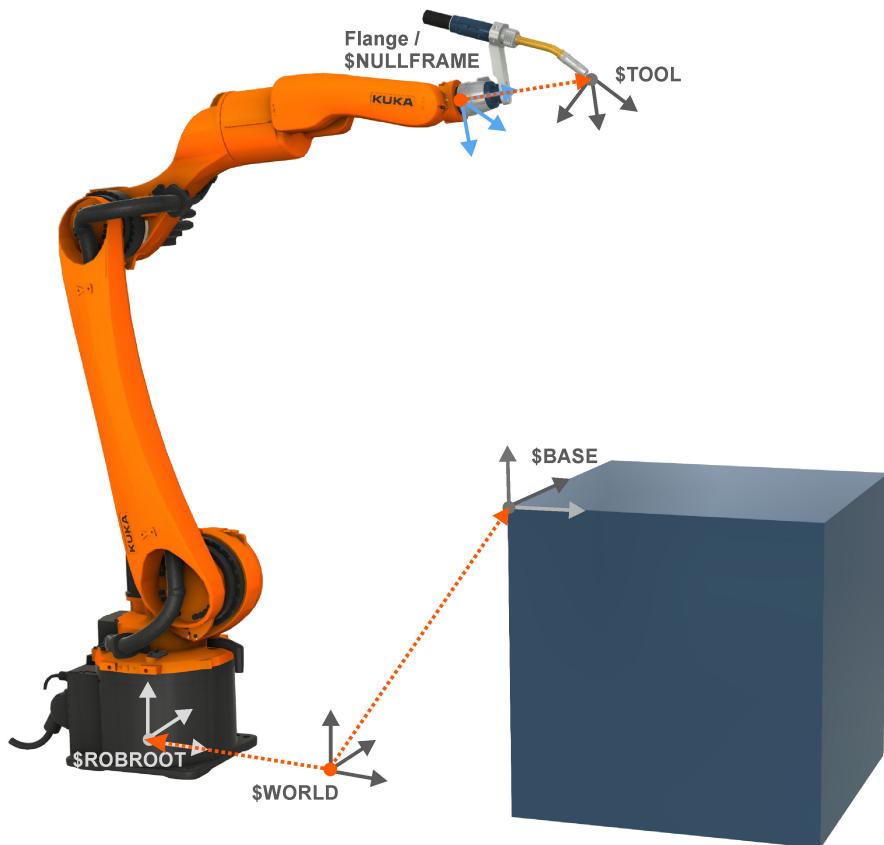


Abb. 3-12: Übersicht Koordinatensysteme

- **ROBROOT**

Roboterfuß-Koordinatensystem

- Es liegt fest im Roboterfuß.
- Ist Ursprung des Roboters.
- Das ROBROOT kann in Bezug auf das WORLD-Koordinatensystem verschoben werden.

- **WORLD**

Weltkoordinatensystem

- Liegt im Auslieferungszustand deckungsgleich im ROBROOT Koordinatensystem.
- Kann aus dem Roboterfuß "herausgeschoben" werden.
- Beschreibt die Position des WORLD Koordinatensystem in Bezug auf das ROBROOT Koordinatensystem.
- Wird unter anderem bei Robotersystemen der Wand- und Deckenmontage verwendet.

- **BASE**

Basiskoordinatensystem

- Ist ein frei definierbares, kundenspezifisches Koordinatensystem
- Beschreibt die Position der Basis in Bezug auf WORLD.

- **FLANGE / NULLFRAME**

Flanschkoordinatensystem

- Das FLANGE Koordinatensystem liegt fest im Roboterflansch.
- Der Ursprung ist die Mitte des Roboterflansches.
- Ist Bezugspunkt für das TOOL Koordinatensystem.
- Es wird auch als NULLFRAME bezeichnet.

- **TOOL**

Werkzeugkoordinatensystem

- Ist ein frei definierbares Koordinatensystem.
- Der Ursprung des TOOL Koordinatensystems wird als **TCP** (Tool Center Point) bezeichnet.
- Verwendung für die Vermessung von Werkzeugen.

3.5.1 Koordinatensysteme und "Rechte-Hand-Regel"

Wo und wie Koordinatensysteme am Robotersysteme wirken, wurde in den vorangegangenen Abschnitten erklärt. Jedoch stellt sich im Alltag oftmals die Frage: "Welche Taste muss ich auf dem smartPAD drücken, damit der Roboter in eine bestimmte Richtung fährt?". Eine Hilfsmittel zur Visualisierung hat man immer mit dabei, und zwar die eigene rechte Hand.

Orientierung

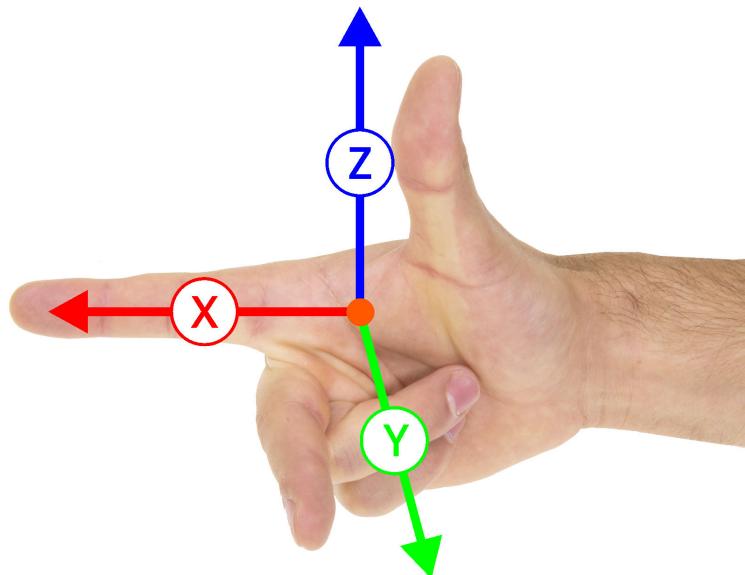
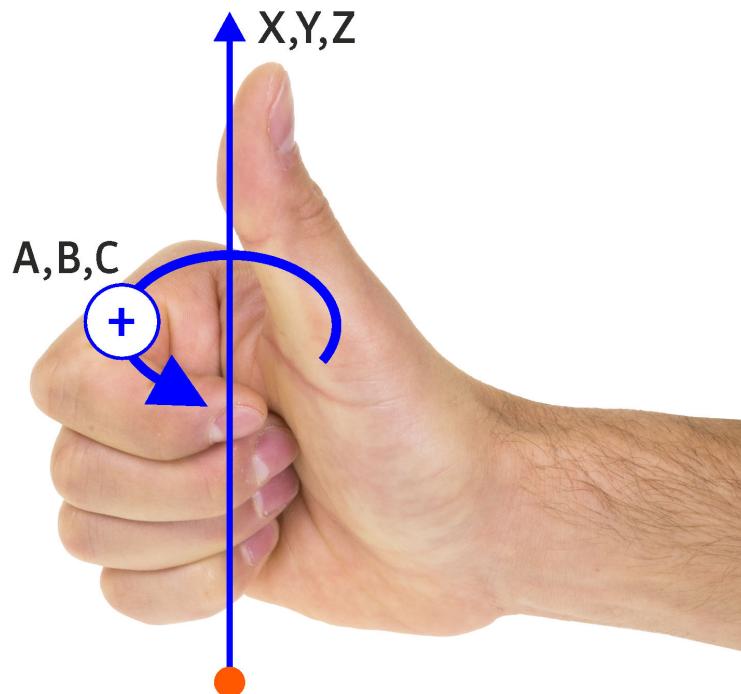


Abb. 3-13: Orientierung, rechte Hand

Daumen, Zeigefinger und den Mittelfinger zu einem rechtwinkligen Koordinatensystem formen und z. B. den Zeigefinger mit X-Achse und den Daumen mit der Z-Achse in Deckung bringen. Der Mittelfinger zeigt jetzt in die Richtung der Y-Achse. Die Fingerspitzen zeigen jeweil in die (+) positive Verfahrrichtung.

Drehung**Abb. 3-14: Drehung, rechte Hand**

Die rechte Hand zu einer Faust ballen und "virtuell" die gewünschte Achse umgreifen. Der ausgestreckte Daumen zeigt in die "positive" Richtung. Alle anderen Finger zeigen nun die positive Drehrichtung an.

3.6 Roboter im Weltkoordinatensystem bewegen

Bewegung im Weltkoordinatensystem

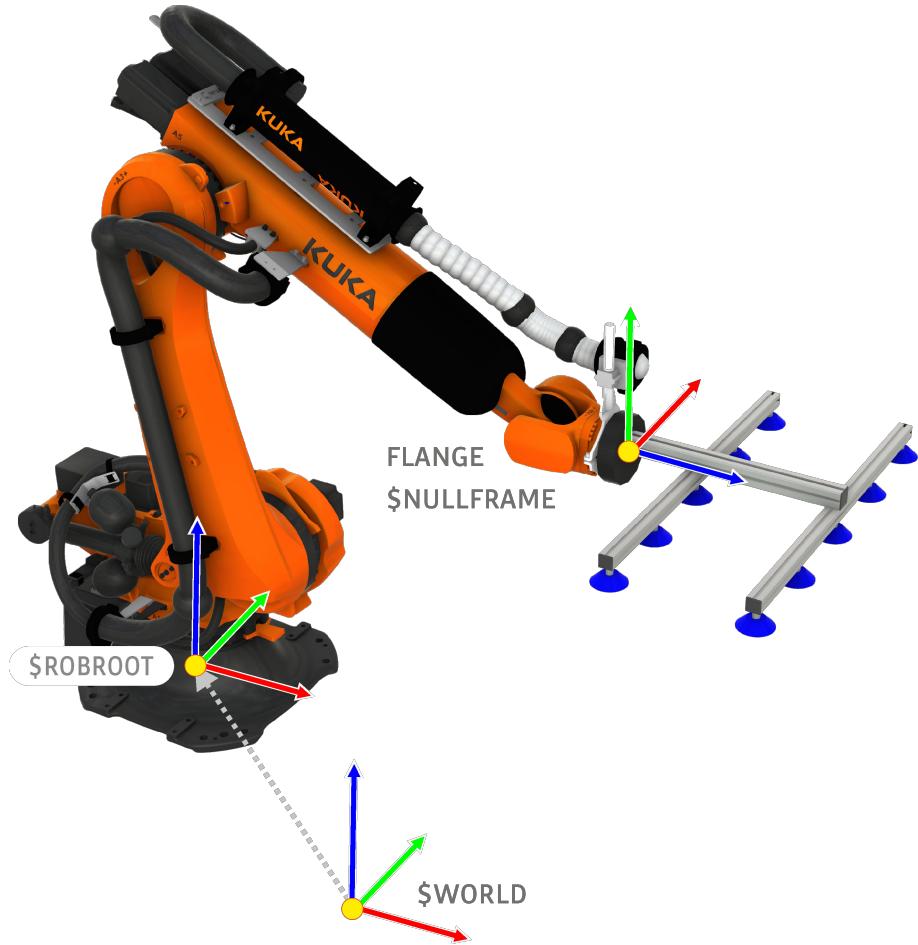


Abb. 3-15: Prinzip Handverfahren Weltkoordinatensystem

- Der Flanschmittelpunkt (NULLFRAME) kann entsprechend der Koordinatenrichtungen des Weltkoordinatensystems bewegt werden.
Dabei bewegen sich **alle** Roboterachsen.
- Dazu werden die Verfahrtasten oder die Space Mouse des KUKA smartPADs verwendet.
- In der Standardeinstellung liegt das Weltkoordinatensystem im Roboterfuß (\$ROBROOT).
- Die Geschwindigkeit kann verändert werden (Hand-Override: HOV)
- Handverfahren ist nur in der Betriebsart T1 möglich.
- Der Zustimmschalter muss in der Mittelstellung gehalten werden.

Space Mouse

- Die Space Mouse ermöglicht eine intuitive Bewegung des Roboters und ist die ideale Wahl beim Handverfahren im Weltkoordinatensystem.
- Mausposition und Freiheitsgrade sind veränderbar.

Handverfahrmöglichkeiten im Weltkoordinatensystem

Ein Roboter kann auf zwei verschiedene Arten in einem Koordinaten-
system bewegt werden:

- **Translatorisch** (geradlinig) entlang der Orientierungsrichtungen des Koordinatensystems: X, Y und Z.
- **Rotatorisch** (drehend/schwenkend) um die Orientierungsrichtungen des Koordinatensystems herum: Winkel A, B und C.

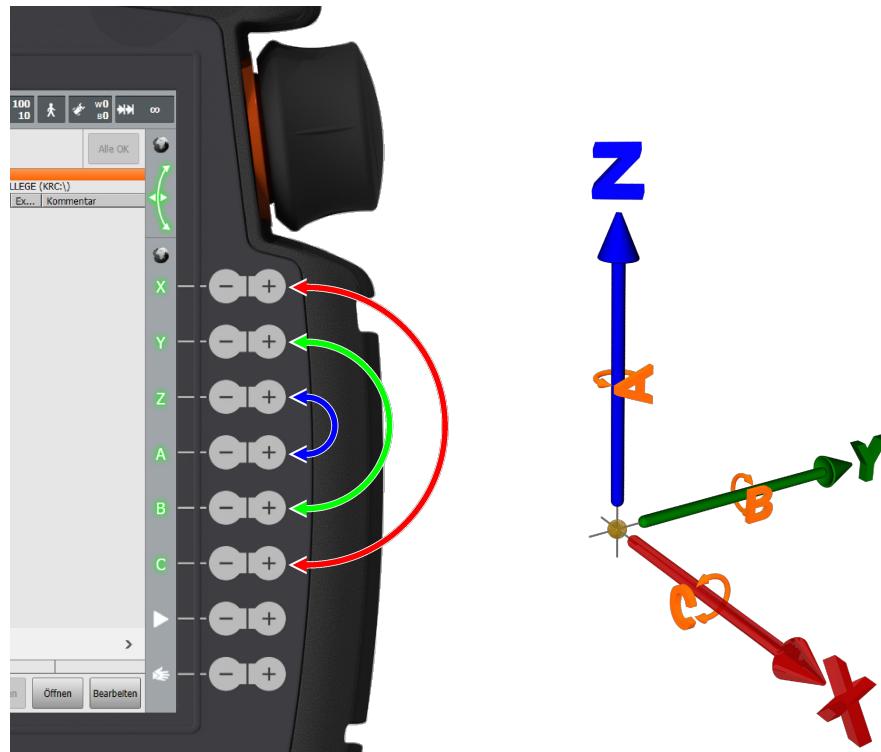


Abb. 3-16: Kartesisches Koordinatensystem

Bei einem Fahrbefehl (z. B. Tastendruck auf Verfahrtaste) berechnet die Steuerung zunächst eine Strecke. Anfangspunkt der Strecke ist der Werkzeugbezugspunkt (**TCP - Tool Center Point**). Die Richtung der Strecke gibt das Weltkoordinatensystem vor. Die Steuerung regelt dann alle Achsen so, dass das Werkzeug auf dieser Strecke geführt (Translation) und um diese gedreht (Rotation) wird.

Vorteile

- Die Bewegung des Roboters ist immer vorhersehbar.
- Die Bewegungen des Tool Center Points im Raum sind immer eindeutig, da der Ursprung und die Koordinatenrichtungen immer bekannt sind.
- Das Weltkoordinatensystem ist bei justiertem Roboter immer verwendbar.
- Mit der 6D-Maus ist eine intuitive Bedienung möglich.

Nachteile

- Der Roboter kann in eine singuläre Stellung gefahren werden.
- In diesem Fall muss mittels der achsspezifischen Verfahrtart der Roboter aus der Singularität verfahren werden.

Verwendung der 6D-Maus

Alle Bewegungsarten sind mit der 6D-Maus möglich:

- **Translatorisch:** durch Drücken und Ziehen der 6D-Maus

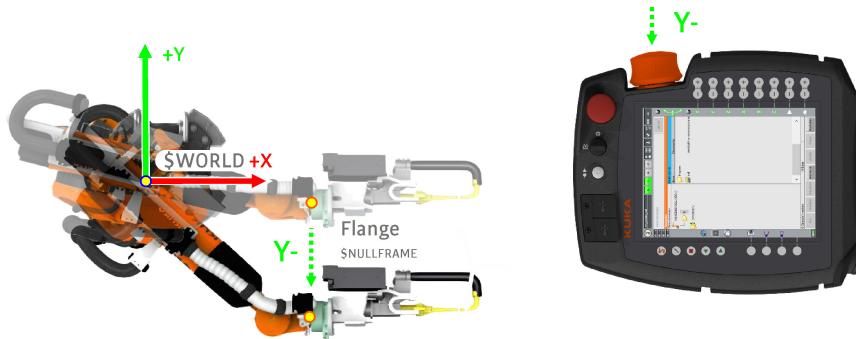


Abb. 3-17: Beispiel: Bewegung nach links

- **Rotatorisch:** durch Drehen und Schwenken der 6D-Maus

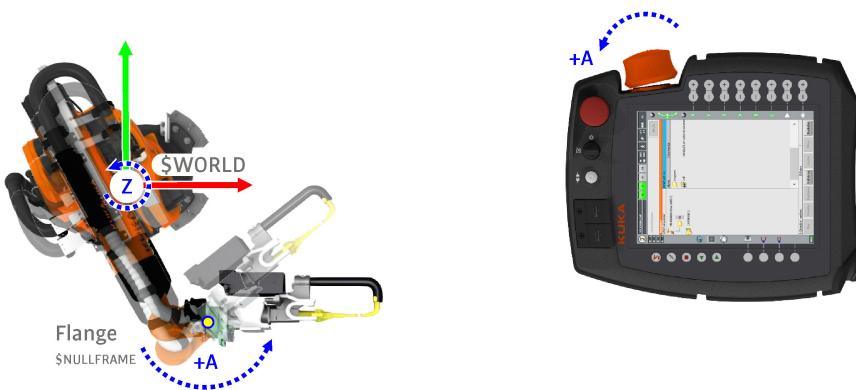


Abb. 3-18: Beispiel: Rotatorische Bewegung um Z: Winkel A

Position des Bedieners

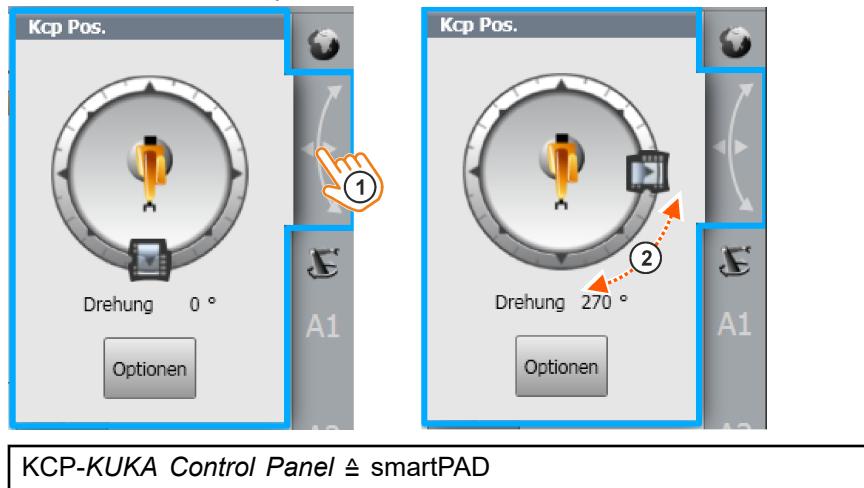
Entsprechend der Position des Bedieners zum Roboter kann die Position der 6D-Maus angepasst werden.



Abb. 3-19: 6D-Maus, Positionen

Translatorische Bewegung durchführen (Welt)

1. smartPAD Position anpassen.



- Optionsfenster durch Antippen auf die Pfeildarstellung (1) neben der 6D-Maus öffnen.

- smartPAD Position durch Verschieben des symbolisierten smart-PADs ändern.

2. Weitere Mauseinstellungen können über die Schaltfläche **Optionen** angepasst werden.

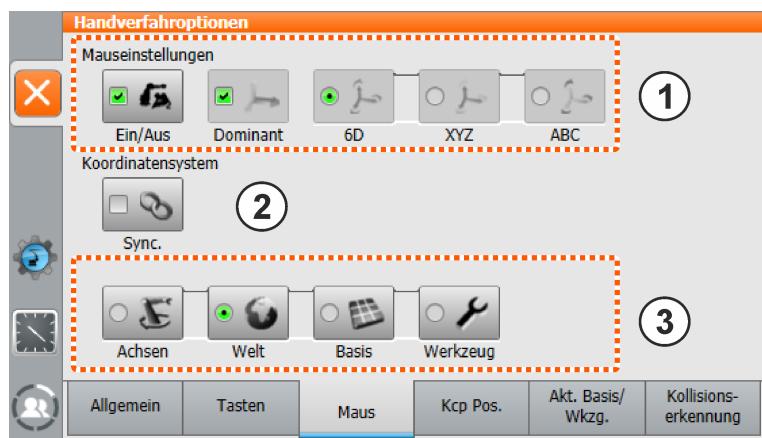


Abb. 3-20: Handverfahroptionen

- 1 Mauseinstellungen
Einschränkung/Erweiterung der Freiheitsgrade für die 6D-Maus
- 2 Sync. Koordinatensystem
Die Einstellung für das Koordinatensystem gelten für Maus und Verfahrtasten
- 3 Koordinatensystem
Koordinatensystem, in dem der Roboter von der 6D-Maus bewegt werden soll



Mit der Schaltfläche **SYNC** kann die Auswahl der Koordinatensysteme für die 6D-Maus und den Verfahrtasten synchronisiert werden.

3. Als Option für die 6D-Maus **Welt** (1) auswählen

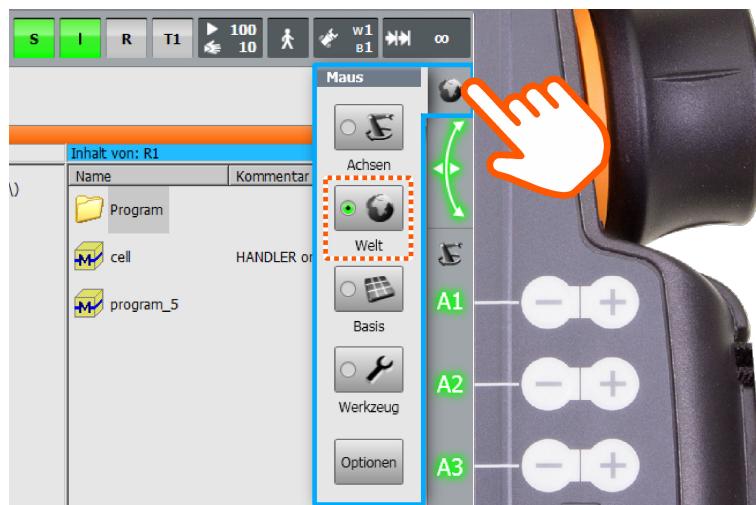
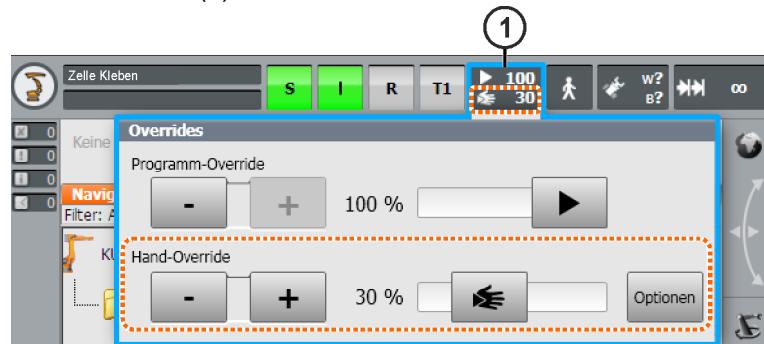


Abb. 3-21: Handverfahren im Weltkoordinatensystem, 6D-Maus

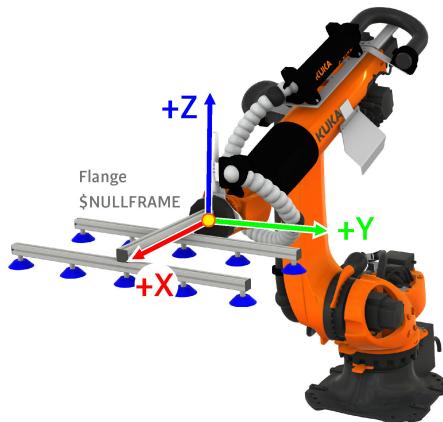
4. Hand-Override (1) einstellen



5. Einen der vier Zustimmungsschalter auf Mittelstellung drücken und halten.



6. Verfahren mit der 6D-Maus in die entsprechende Richtung.



7. Alternativ können auch die Verfahrtasten verwendet werden.

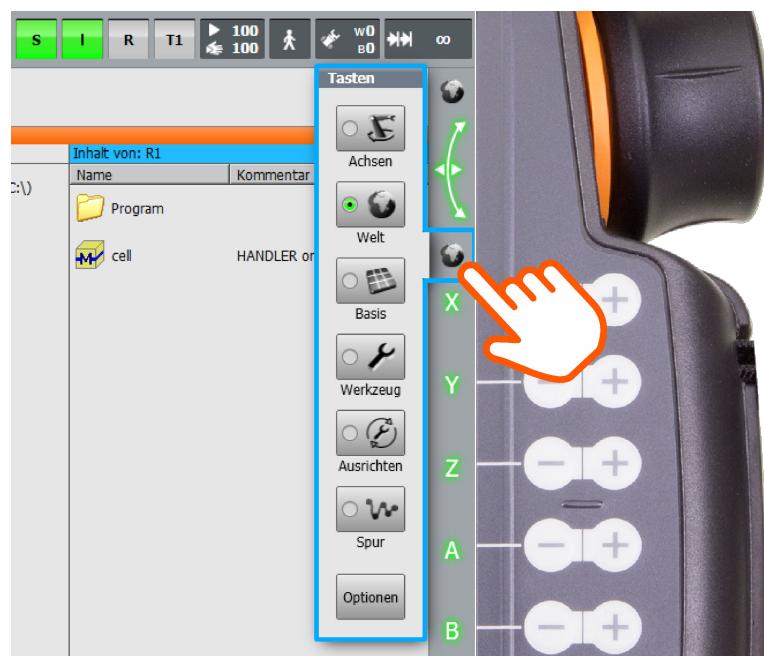


Abb. 3-22: Handverfahren im Weltkoordinatensystem, Verfahrtasten

3.6.1 Übung: Bedienung und Handverfahren im Weltkoordinatensystem mittels Verfahrtasten

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bedienung und Handverfahren im Weltkoordinatensystem mittels Verfahrtasten**

3.6.2 Übung: Bedienung und Handverfahren im Weltkoordinatensystem mittels 6D-Maus

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bedienung und Handverfahren im Weltkoordinatensystem mittels 6D-Maus**

3.7 Roboter mittels Verfahrtart Spur bewegen

Beschreibung

Wird der Roboter bewegt, zeichnet die Robotersteuerung zyklisch die aktuelle Roboterposition in einem Puffer auf. Das gilt unabhängig davon, wie der Roboter bewegt wird. Auch der Wechsel zwischen Programmaufruf und zwischenzeitliches Handverfahren hat keinen Einfluss auf die Positionsauzeichnung. Ein Rückwärtsfahren in die unmittelbaren Vergangenheit des Roboters ist möglich. Der Roboter lässt sich an beliebiger Stelle mittels der Verfahrtart **Spur** und den **±-Tasten** verfahren, also auch wieder zur Ausgangsposition zurück.

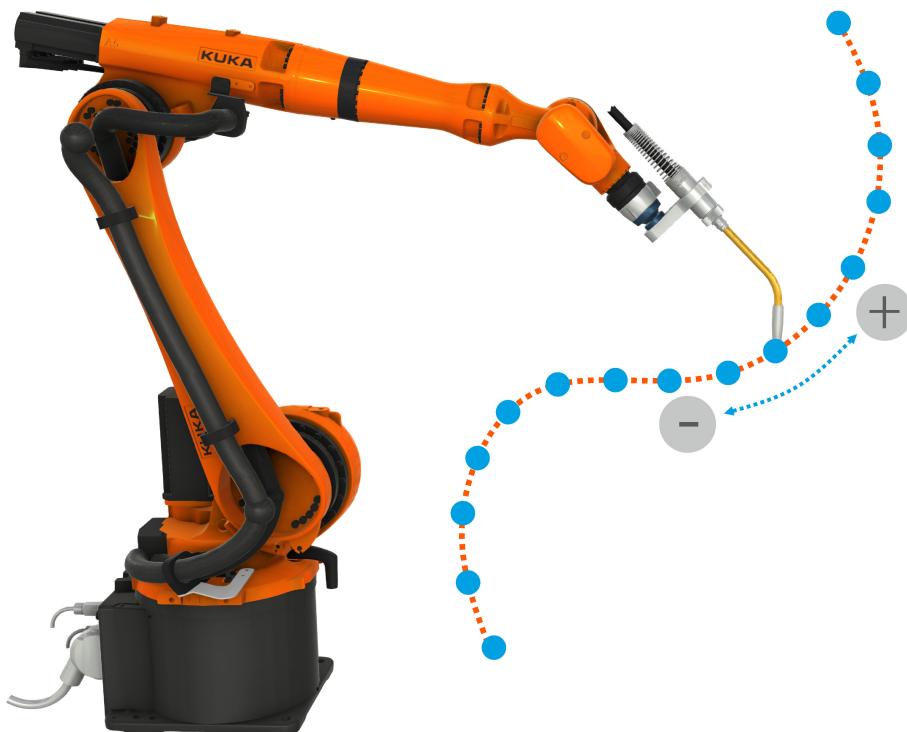


Abb. 3-23: Roboter verfahren mit Spur



Die abgefahrene Bahn entspricht näherungsweise der Originalbahn. Sie wird durch das Verbinden der gespeicherten Punkte mittels SPLINE-Kontur ermittelt.



Der Puffer für das Rückwärtsfahren über die Verfahrtasten ist nicht identisch mit dem Puffer für das Rückwärtsfahren über die Start-Rückwärts-Taste des smartPADs.



Wird der Roboter beim Verfahren durch einen Software-Endschalter begrenzt, wird die Aufzeichnung zurückgesetzt und eine neue Aufzeichnung begonnen.

Vorgehensweise



Die Verwendung der Verfahrtart Spur ist nur in der Betriebsart T1 möglich.

1. Im Fenster **Handverfahroptionen** in der Registerkarte **Tasten** die Option **Spur** (1) wählen.

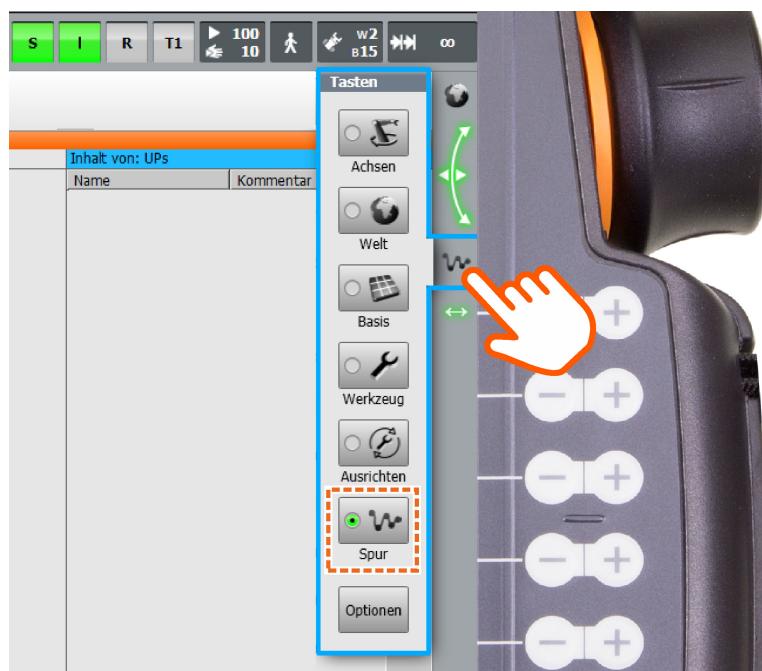


Abb. 3-24: Verfahrtart Spur

- Den Hand-Override einstellen.

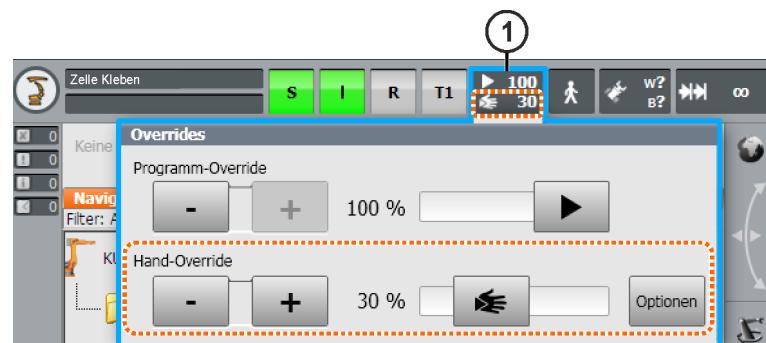


Abb. 3-25: Hand-Override über Statusleiste einstellen

- Einen der Zustimmungsschalter drücken und halten.



Abb. 3-26: Zustimmungsschalter

4. Neben der obersten Verfahrtaste wird folgendes Doppelpfeilsymbol angezeigt:



Abb. 3-27: Handverfahren Spur

5. Bei dieser Verfahrtaste auf Minus drücken, um den Roboter entlang der aufgezeichneten Bahn rückwärts zu verfahren.
6. Auf Plus drücken, um den Roboter entlang der aufgezeichneten Bahn vorwärts zu verfahren.

Der Roboter kann abwechselnd vor und zurück verfahren werden. Wenn in der einen oder anderen Richtung das Ende der aufgezeichneten Bahn erreicht wird, zeigt die Robotersteuerung folgende Meldung an: *Das Ende der aufgezeichneten Bahn ist erreicht.*

Vergleich "Start Rückwärts" und "Verfahrtart Spur"

Die Tabelle zeigt die wichtigsten Unterschiede zwischen dem Rückwärtsfahren über die Start-Rückwärts-Taste und dem über die Verfahrtasten.

Über Startrückwärts	Über die Verfahrtasten
Programmbewegungen können rückwärts gefahren werden.	Nahezu alle Arten von Bewegungen können rückwärts gefahren werden.
Der Bahnverlauf kann sich von der Vorwärtsbahn unterscheiden.	Der Bahnverlauf ist wie bei der Vorwärtsbahn.
Überschleifen oder Pendeln wird beim Rückwärtfahren nicht ausgeführt.	
Die ursprünglichen Bewegungen können rückwärts einzeln gefahren werden.	Die Bahn ist ein Kontinuum. Es kann an beliebigen Stellen gestoppt werden, jedoch können die ursprünglichen Bewegungen rückwärts nicht einzeln (Bewegung für Bewegung) gefahren werden.

Löschen bei Start aus Puffer

Wenn der Benutzer den Roboter über die Option **Spur** rückwärts verfahren hat und dann eine andere Bewegung startet, wird ein Teil des Puffers gelöscht.

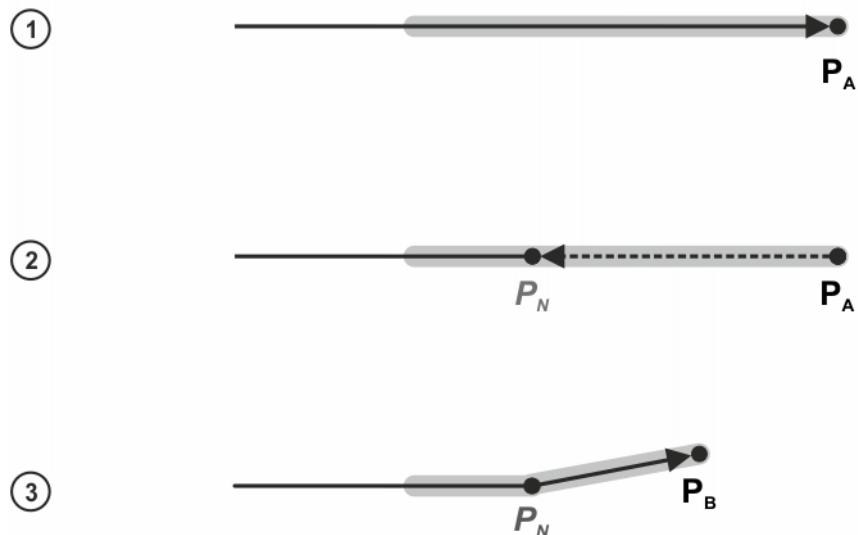


Abb. 3-28: Löschen bei Start aus Puffer

Schritt	Beschreibung
1	Ausgangssituation: Der Benutzer hat den Roboter vorwärts verfahren. Der Roboter befindet sich an einem Punkt P_A . Der letzte Teil der Bahn dorthin ist im Puffer gespeichert (dicker grauer Balken).
2	Der Benutzer verfährt den Roboter nun rückwärts (gestrichelter schwarzer Pfeil). Der Roboter befindet sich jetzt an einem Punkt P_N im Puffer. Zu diesem Zeitpunkt ist im Puffer noch die gleiche Bahn gespeichert wie bei Schritt 1.

Schritt	Beschreibung
3	<p>An Punkt P_N startet der Benutzer eine andere Bewegung. Gemeint ist jede Bewegung außer weiteres Fahren auf der aufgezeichneten Bahn (gleichgültig, ob rückwärts oder vorwärts).</p> <p>Der Roboter befindet sich nun an einem Punkt P_B.</p> <ul style="list-style-type: none"> Der Teil von P_N bis P_A wurde aus dem Puffer gelöscht. Die neue Bewegung von P_N nach P_B wurde aufgezeichnet.

3.8 Anzeigmöglichkeiten von Roboterpositionen

Anzeigmöglichkeiten von Roboterpositionen

Die gegenwärtige Roboterposition lässt sich in zwei verschiedenen Arten darstellen:

- **Achsspezifisch:**

```
$AXIS_ACT={A1..., A2..., A3..., A4..., A4..., A5..., A6...}
```

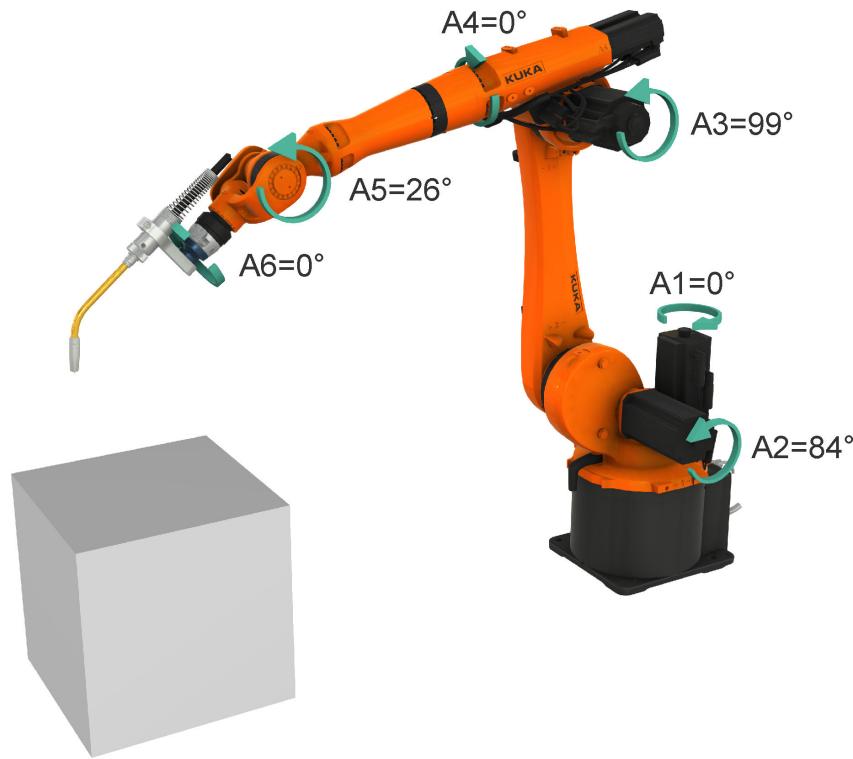


Abb. 3-29: Achsspezifische Roboterposition

- Es wird für jede Achse der aktuelle Achswinkel angezeigt:
- Dies entspricht dem absoluten Winkelwert ausgehend von der mechanischen Nullstellung der Achse.

- **Kartesisch:**

```
$POS_ACT={X..., Y..., Z..., A..., B..., C..., S..., T..., E1..., ...}
```

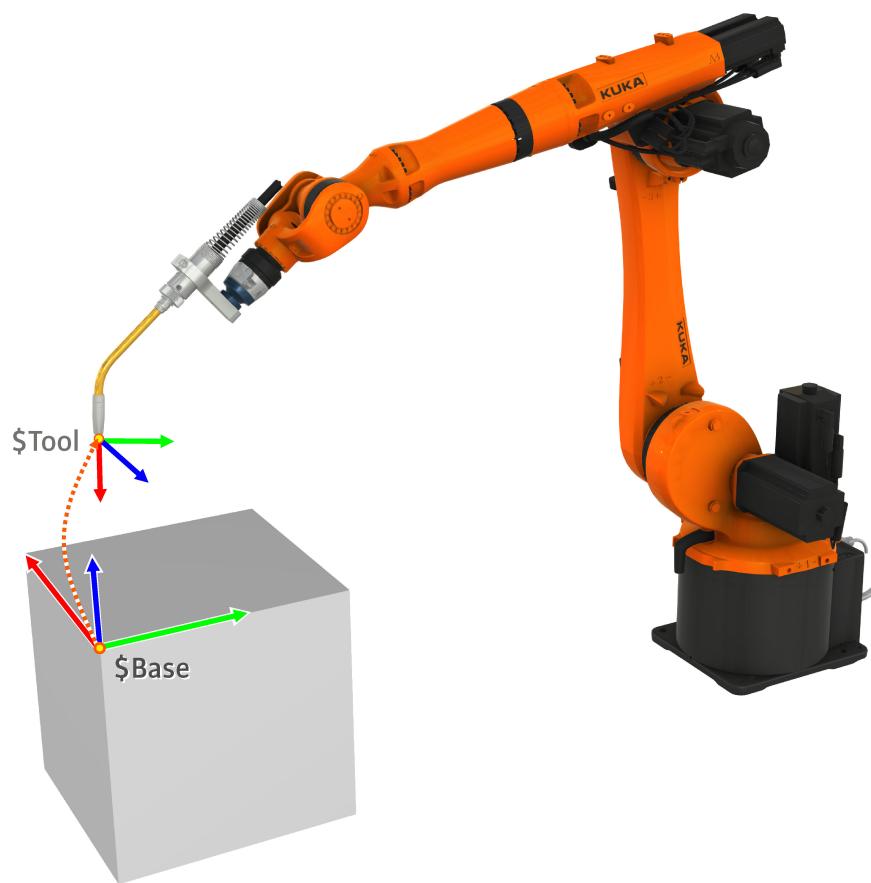


Abb. 3-30: Kartesische Position

- Es wird die aktuelle Position des aktuellen TCPs (Toolkoordinatensystems) in Bezug zum aktuell ausgewählten Basiskoordinatensystem angezeigt.
- Wenn kein Toolkoordinatensystem ausgewählt ist, gilt das Flanschkoordinatensystem!
- Wenn kein Basiskoordinatensystem ausgewählt ist, gilt das Weltkoordinatensystem!

Kartesische Position mit verschiedenen Basiskoordinatensystemen

Betrachtet man das unten stehende Bild, fällt sofort auf, dass der Roboter drei mal die gleiche Stellung einnimmt. Die Positionsanzeige liefert aber in jedem der drei Fälle unterschiedliche Werte:

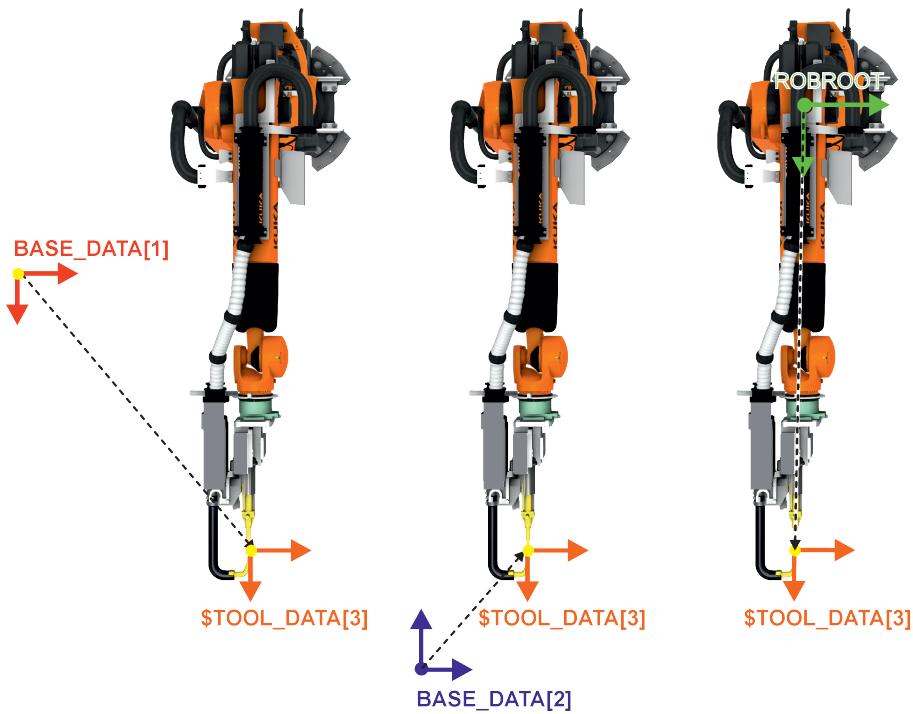


Abb. 3-31: Drei Werkzeugpositionen - eine Roboterstellung!

Es wird die Position des Toolkoordinatensystems/TCP im jeweiligen Basiskoordinatensystem angezeigt:

- für die Basis 1
- für die Basis 2
- für die Basis 0: dies entspricht dem Weltkoordinatensystem (in den meisten Fällen auch dem \$NULLFRAME)!



Nur wenn die richtige Basis und das richtige Werkzeug ausgewählt sind, liefert die Anzeige der kartesischen Istposition die zu erwartende Werte!



Abb. 3-32: Werkzeug und Basis wählen

3.8.1 Aktuelle Roboterposition auf dem smartPAD abfragen

Beschreibung

- Die aktuelle Roboterposition kann auf dem smartPAD in zwei verschiedenen Varianten abgefragt werden.
 - **achsspezifische Position**
(>>> "Anzeige der achsspezifische Roboterposition" Seite 79)
 - **kartesische Position**
(>>> "Anzeige der kartesischen Roboterposition" Seite 80)
- Die Roboterposition kann in jeder Benutzergruppe, als auch in jeder Betriebsart eingeblendet werden.



Die Roboterposition kann auch parallel zu einem Programm oder im Handverfahrbetrieb eingeblendet werden.

Vorgehensweise

Anzeige der achsspezifische Roboterposition

1. Menüpfad: Robotertaste > Anzeige > Istposition
2. Über die Schaltfläche **Kartesisch/Achsspezifisch** (7) kann zwischen den Ansichten gewechselt werden.
3. Über die Schaltfläche **Motor** (6) wird zusätzlich der Motorwinkel (3) in der Spalte (4) eingeblendet.



Abb. 3-33: Roboterposition, achsspezifisch

Pos.	Beschreibung
1	Achsenname und symbolische Darstellung des Achstyps <ul style="list-style-type: none"> • : rotatorisch • : endlos drehend • : linear
2	Aktuelle Achsposition
3	Symbol "Motor"
	Wird nur angezeigt, wenn die Checkbox Motor mit Häkchen ist.
4	Aktueller Motorwinkel
	Wird nur angezeigt, wenn die Checkbox Motor mit Häkchen ist.

Pos.	Beschreibung
5	<p>Bei rotatorischen und linearen Achsen: grafische Darstellung der aktuellen Position, bezogen auf den erlaubten Achsbereich</p> <ul style="list-style-type: none"> Der Balken stellt den erlaubten Bereich dar, d. h. den Bereich zwischen den Software-Endschaltern. Der linke Rand entspricht dem negativen Endschalter, der rechte dem positiven. <p>Die Mitte des Balkens entspricht der Mitte des erlaubten Bereichs. Beispiel: Der negative Endschalter liegt bei 60°, der positive bei 100°. Die Mitte des Balkens entspricht somit 20°.</p> <ul style="list-style-type: none"> Der senkrechte Strich stellt die Position der Achse in dem Bereich dar. Wenn eine Achse einen Endschalter erreicht hat, wird der Balken rot und eine Meldung im Meldungsfenster (A) ausgegeben. <p>Bei endlos drehenden Achsen weicht die Darstellung ab.</p>

Anzeige der kartesischen Roboterposition

1. Menüpfad: Robotertaste > Anzeige > Istposition
2. Über die Schaltfläche **Kartesisch/Achsspezifisch** (7) kann zwischen den Ansichten gewechselt werden.

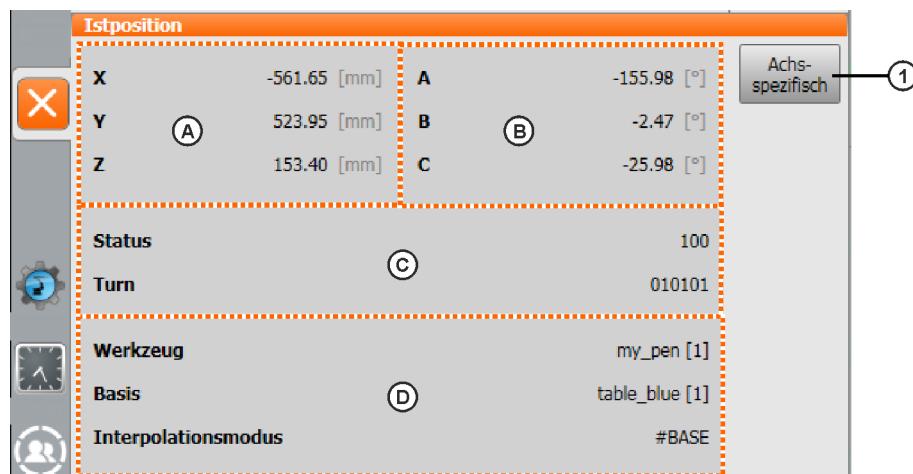


Abb. 3-34: Roboterposition, kartesisch

Po s.	Anzeigefelder	Beschreibung
A	X, Y, Z	Aktuelle Position
B	A, B, C	Aktuelle Orientierung
C	Status	Status in binärer Darstellung, 3-stellig
	Turn	Turn in binärer Darstellung, 6-stellig

Po s.	Anzeigefelder	Beschreibung
D	Werkzeug	<ul style="list-style-type: none"> • Name und Nummer des gültigen Werkzeugs bzw. der gültigen Basis oder \$NULLFRA-ME[0]
	Basis	<ul style="list-style-type: none"> • Wenn kein Werkzeug/keine Basis gültig ist, z. B. nach dem Hochfahren der Steuerung, wird "?[-1]" angezeigt.
	Interpolati- onsmodus	<ul style="list-style-type: none"> • #BASE: Das Werkzeug ist am Anbauflansch montiert. • #TCP: Das Werkzeug ist ein feststehendes Werkzeug. <p>(>>> <i>"Geänderter Bewegungsablauf bei feststehendem Werkzeug" Seite 346</i>)</p>

4 Inbetriebnahme des Roboters

4.1 Lerneinheit: Inbetriebnahme des Roboters

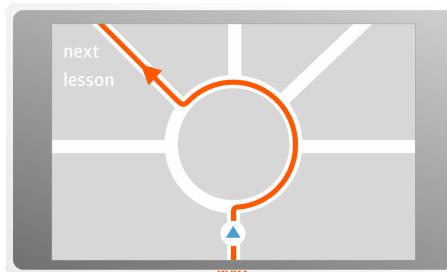


Abb. 4-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Benutzergruppen auf der Steuerung
- Prinzip des Justierens
- Werkzeug- und Basisverwaltung
- Lasten am Roboter
- Vermessung eines Werkzeugs
- Vermessung einer Basis
- Speicherauslastung anzeigen
- Steuerung herunterfahren

4.2 Benutzergruppen auf der Steuerung

Beschreibung

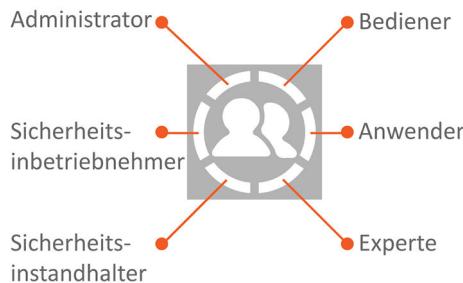


Abb. 4-2: Benutzergruppen

- Durch Benutzergruppen können auf der Robotersteuerung unterschiedlichen Personenkreisen gezielt Funktionen und Berechtigungen zugewiesen werden.
- Alle Benutzergruppen außer der Default-Benutzergruppe sind mit einem Passwort geschützt.
Das Default-Passwort für alle Gruppen lautet "kuka". Das Passwort kann pro Gruppe geändert werden.
- Um einem Mißbrauch vorzubeugen, sollte das Passwort für jede Gruppe separat geändert werden.



Äußerst sensibel sind in diesem Zusammenhang die Benutzergruppen Admin, Sicherheitsinbetriebnehmer und Sicherheitsinstandhalter zu betrachten.

- Jeder Benutzergruppe können über die Rechteverwaltung weitere Berechtigungen zugewiesen werden. Die Rechteverwaltung steht nur der Benutzergruppe Administrator zur Verfügung.

Benutzergruppen

- **Bediener**
Stark begrenzte Rechte: Der Bediener darf keine Funktionen ausführen, die das System permanent verändern.
- **Anwender**
Der Anwender darf Funktionen ausführen, die für den normalen Betrieb des Roboters erforderlich sind.
Beispiel: Programmieren von einfachen Bewegungsprogrammen mittels Inline-Formularen.
- **Experte**
Der Experte darf Funktionen ausführen, für die Expertenwissen erforderlich ist.
Beispiel: Programmieren von Expertenprogrammen.
- **Sicherheitsinstandhalter**
Diese Benutzergruppe kann eine vorbereitete Sicherheitsschnittstelle (durch den Sicherheitsinbetriebnehmer) auf der Steuerung oder personensichere Zellenkonfigurationen (z. B. SafeOperation) mittels Aktivierungscode aktivieren.
Der Sicherheitsinstandhalter verfügt über weitergehende Rechte.
- **Sicherheitsinbetriebnehmer**
Diese Benutzergruppe ist berechtigt die Sicherheitsschnittstelle auf der Steuerung abzuändern oder personensichere Zellenkonfigurationen (z. B. SafeOperation) zu konfigurieren.
- **Administrator**
Der Administrator darf alle Funktionen ausführen (inkl. Sicherheitstechnik).
Die Benutzergruppe hat Zugriff auf die Rechteverwaltung.

4.2.1 Benutzergruppe wechseln

Anzeige der aktuellen Benutzergruppe

- Auf der smartHMI zeigt das Symbol **Benutzergruppe** durch die Anzahl seiner weißen Segmente an, welche Benutzergruppe aktuell ausgewählt ist.
- Beispiel: 3 weiße Segmente = **Experte**



Abb. 4-3: Benutzergruppe: **Experte**

Vorgehensweise

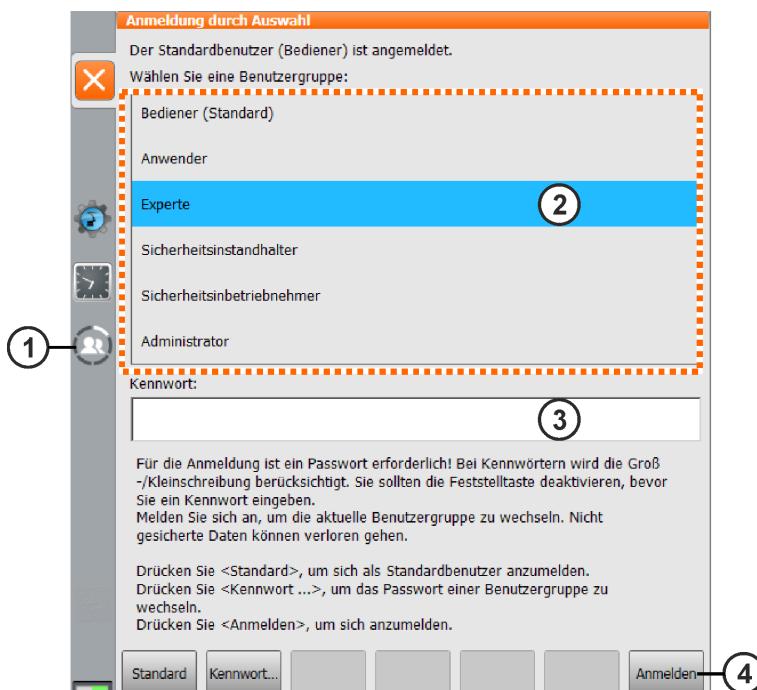


Abb. 4-4: Benutzergruppe wechseln

1. Auf der smartHMI das Symbol **Benutzergruppe** (1) berühren.
Alternativ: **Robotertaste > Konfiguration > Benutzergruppe** wählen.
Das Fenster für die Benutzergruppenauswahl öffnet sich.
2. Gruppe wechseln:
 - Um in die Default-Benutzergruppe zu wechseln, **Standard** drücken.
Standard wird nicht angezeigt, wenn man sich bereits in der Default-Benutzergruppe befindet.
 - Um in eine andere Benutzergruppe zu wechseln, die gewünschte Benutzergruppe (2) markieren.
Dann Passwort (3) eingeben und mit **Anmelden** (4) bestätigen.

4.2.2 Kennwort ändern

Vorgehensweise

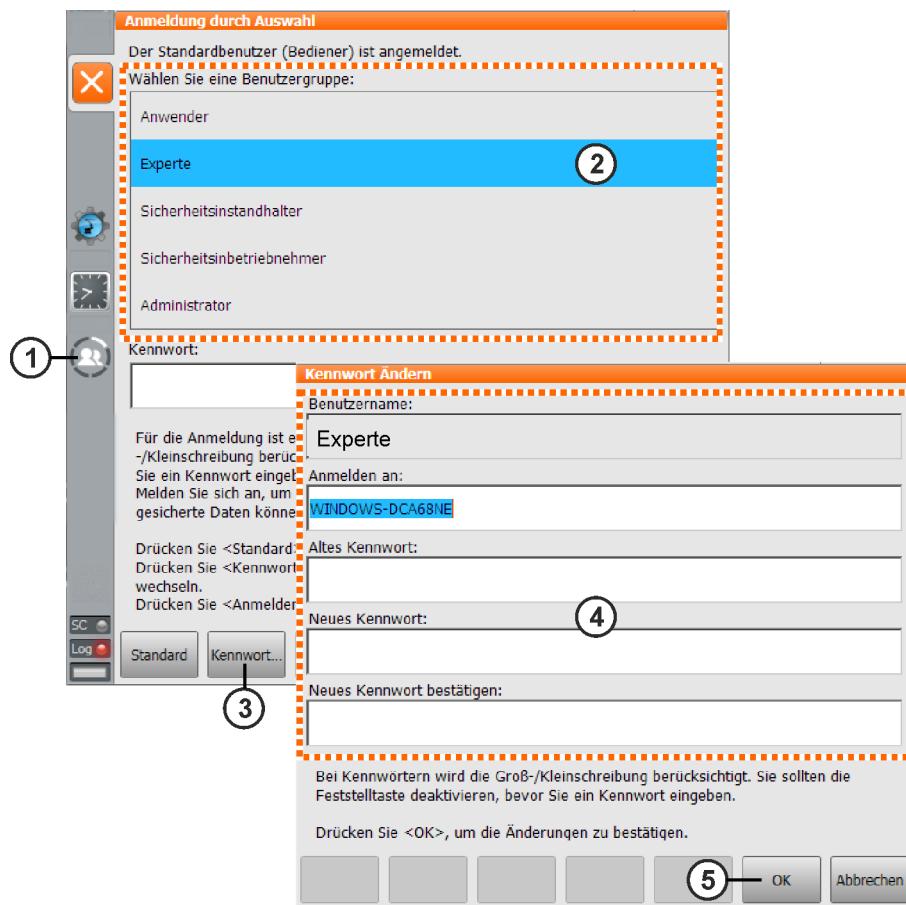


Abb. 4-5: Kennwort ändern

1. Auf der smartHMI das Symbol **Benutzergruppe** (1) berühren.
Oder: Im Hauptmenü **Konfiguration > Benutzergruppe** wählen.
Das Fenster für die Benutzergruppenauswahl öffnet sich.
2. Die Benutzergruppe (2), deren Passwort geändert werden soll, markieren.
3. Unten in der Schaltflächenleiste auf **Kennwort ...** (3) drücken.
4. Das alte Passwort eingeben. Das neue Passwort zweimal eingeben.
(4)
Die Eingaben werden aus Sicherheitsgründen verschlüsselt angezeigt.
Die Groß- und Kleinschreibung wird berücksichtigt.
5. **OK** (5) drücken. Das neue Passwort ist sofort gültig.

4.2.3 Rechte auf der Steuerung verwalten

Beschreibung



Abb. 4-6

- Auf der Steuerung steht ausschließlich für die Benutzergruppe **Administrator** der Menüpunkt Rechteverwaltung zur Verfügung.

Menüpfad: Robotertaste > Inbetriebnahme > Rechteverwaltung

Alle Benutzergruppen haben Leserechte.

- Über die **Rechteverwaltung** wird festgelegt,
 - welche **Betriebsarten** den einzelnen Benutzergruppen zur Verfügung stehen.

Rechteverwaltung

Zuordnung der erlaubten Betriebsarten zu Benutzerruppen

The diagram illustrates the mapping of functional groups to user groups and their assigned operating modes. It features a vertical double-headed arrow between the 'Funktionsgruppen' (Functional Groups) and 'Benutzerruppen' (User Groups) sections, and a horizontal arrow pointing left from the 'Betriebsarten' (Operating Modes) section towards the 'Funktionsgruppen' section.

Benutzer	T1	T2	Aut	Ext
Bediener	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anwender	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Experte	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sicherheitsinstandhalter	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sicherheitsinbetriebnehmer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Administrator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Funktionsgruppen

Betriebsarten

Abb. 4-7: Rechteverwaltung, Betriebsarten

- welche **Funktionsgruppen** den einzelnen Benutzergruppen zur Verfügung stehen.

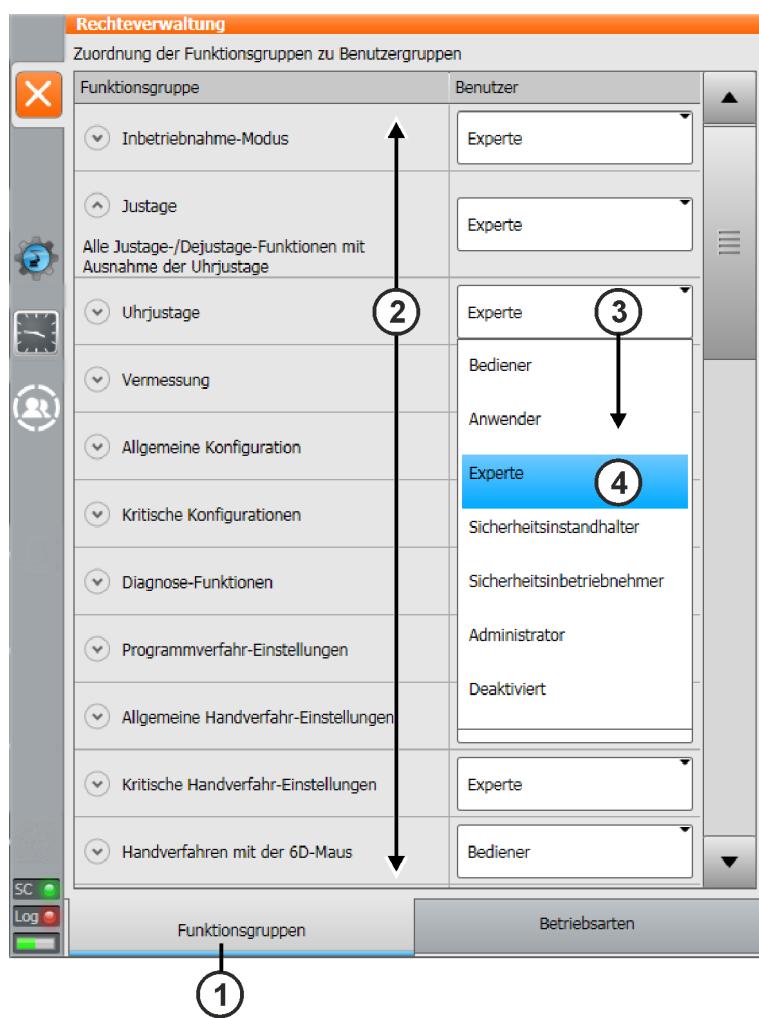


Abb. 4-8: Funktionsgruppen

- 1 Registerkarte Funktionsgruppen
- 2 Funktion welche für bestimmte Benutzergruppen zugeordnet werden können
- 3 Anzeige der aktueller Benutzergruppe
- 4 Setzen der neuen Benutzergruppe

4.3 Prinzip des Justierens

Warum justieren?



Abb. 4-9

Bei der Justage wird jeder Roboterachse ein Referenzwert zugewiesen. Somit weiß die Robotersteuerung, wo sich diese Achse befindet.

Nur wenn ein Industrieroboter vollständig und korrekt justiert ist, kann er optimal genutzt werden. Nur dann weist er seine volle Punkt- und Bahngenaugkeit auf. Programme können abgearbeitet werden. Ein nicht justierter Roboter kann lediglich achsspezifisch verfahren werden.

Ein kompletter Justagevorgang beinhaltet das Justieren jeder einzelnen Achse. Mittels eines technischen Hilfsmittels (**EMD - Electronic Mastering**)

Device) wird jeder Achse an ihrer **mechanischen Nullstellung** ein roboterspezifischer Referenzwert zugewiesen (z. B. 0° , -90° , 110° ...).

Da somit die mechanische und elektrische Position der Achse in Übereinstimmung gebracht werden, erhält jede Achse einen eindeutigen Winkelwert. Die Justagestellung ist bei allen Robotern ähnlich, jedoch nicht gleich. Die genauen Positionen können sich auch zwischen den einzelnen Robotern eines Robotertyps unterscheiden.

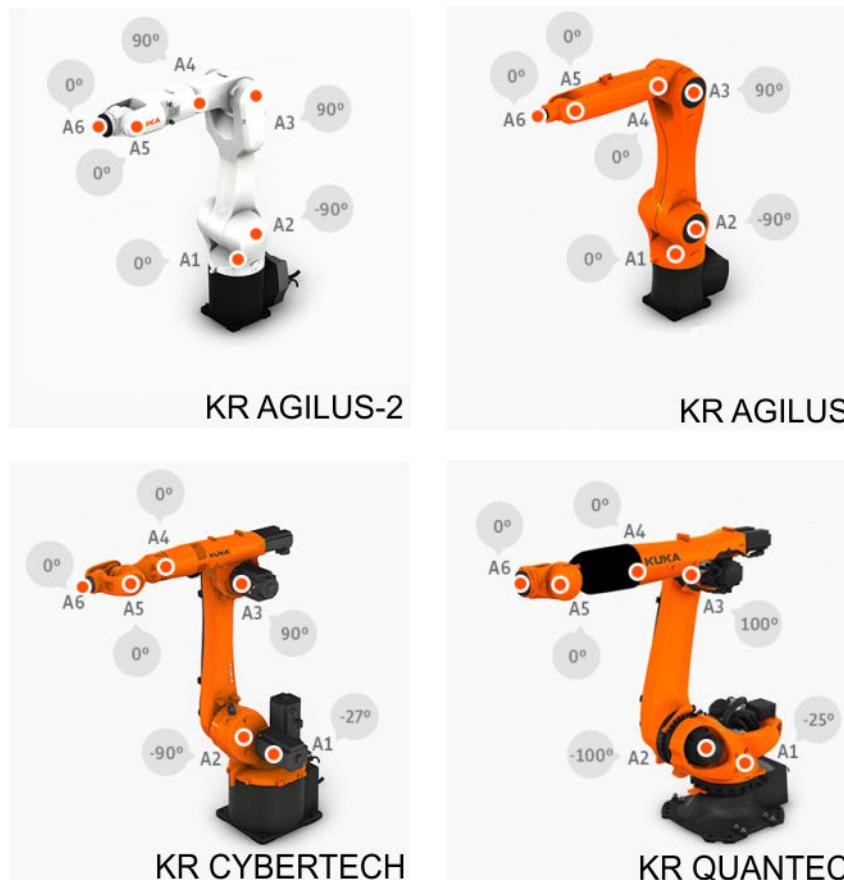


Abb. 4-10: Justagepositionen

Beispielhafte Winkelwerte für mechanische Referenzstellungen:

Achse	Cybertech KR16 R1610	QUANTEC KR120 R2700-2
A1	27°	-25°
A2	-90°	-100°
A3	90°	100°
A4	0°	0°
A5	0°	0°
A6	0°	0°

Wann wird justiert?

Grundsätzlich muss ein Roboter immer justiert sein. In folgenden Fällen muss die Justage durchgeführt werden:

- Bei der Inbetriebnahme des Roboters



Abb. 4-11: Roboter Inbetriebnahme

- Nach Instandhaltungsmaßnahmen an Komponenten, die an der Positionsverfassung beteiligt sind (z. B. Motor mit Resolver oder RDC).

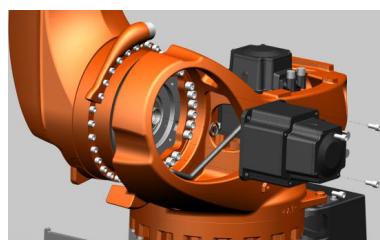


Abb. 4-12: Motortausch

- Wenn die Roboterachsen ohne Steuerung bewegt wurden, z. B. mittels Freidreh-Vorrichtung.



Abb. 4-13: Freidreheinrichtung

- Nach mechanischen Reparaturen/Problemen ist, je nach Vorgabe, eine Justage oder Lastjustage durchzuführen und die ermittelten Abweichungen sind zu übernehmen.
 - nach dem Wechsel eines Getriebes

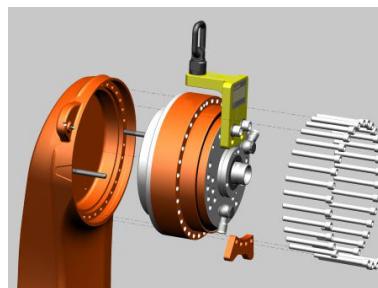


Abb. 4-14: Austausch, Getriebe

- nach dem Auffahren auf einen Endanschlag mit mehr als 250 mm/s



Abb. 4-15: Fahren gegen Endanschlag

- nach einer Kollision



Abb. 4-16: Kollision



Vor Instandhaltungsmaßnahmen ist es generell sinnvoll, die aktuelle Justage zu prüfen.

Sicherheitshinweise zur Justage

Bei nicht justierten Roboterachsen ist die Funktion des Roboters erheblich eingeschränkt:

- **Kein Programmbetrieb möglich**
Programmierte Punkte können nicht angefahren werden.
- **Kein kartesisches Handverfahren möglich**
Bewegungen in den Koordinatensystemen sind nicht möglich.
- **Software-Endschalter sind deaktiviert**
Damit ist es möglich, gegen die mechanischen Endanschläge zu fahren.

4.3.1 Justagemittel

Beschreibung

Für die Justage des Roboters stehen zwei Justagemittel zur Verfügung:

- Messuhr
(>>> "Messuhr" Seite 92)
- Justage-Set
(>>> "SEMD-/MEMD-Justage-Set" Seite 92)

Messuhr



Abb. 4-17: Messuhr

- Mit Hilfe der Messuhr kann der Roboter "Standard" justiert werden.
- Die Genauigkeit ist stark vom Justierenden abhängig (z. B. Parallaxenfehler).
- Es ist auf das SEMD/MEMD - Justage Set zurückzugreifen.
(>>> ["SEMD-/MEMD-Justage-Set" Seite 92](#))

HINWEIS

Eine Lastjustage mit der Messuhr ist nicht möglich.

SEMD-/MEMD-Justage-Set



Abb. 4-18: SEMD-/MEMD-Justage-Set

- 1 Leitungen
 - 2 Universal Justagebox
 - 3 MEMD-Sensor für kleine Justagepatronen
 - 4 SEMD-Sensor für große Justagepatronen
- Das SEMD-/MEMD-Justage-Set löst folgende Justage-Sets ab:

- **EMT** - *Elektronischer Messtaster* für KR C1- und KR C2-Technik
hierzu ist die **Option für KR C2-Technik** (Adapter) notwendig.
- **EMD** - *Electronic Mastering Device* für KR C4-Technik
- Mit dem SEMD-/EMD-Justage-Set können alle Robotervarianten mit **großer** und **kleiner** Justagepatrone justiert werden.
 - **MEMD** - *Micro Electronic Mastering Device*
für kleine Justagepatronen
 - **SEMD** - *Standard Electronic Mastering Device*
für große Justagepatronen
- Das Justageset ist auch getrennt als SEMD- oder MEMD- Justageset verfügbar.

Verfügbare Justage-Sets

Justag- Set	Artikelnummer	Beschreibung
SEMD/ MEMD	<u>00-228-936</u>	Justagegerät für alle Messpatronen (M8-/M20-Feingewinde) anwendbar für alle Robotertypen
SEMD	<u>00-228-934</u>	Justagegerät für Messpatronen mit M20-Feingewinde anwendbar für z.B. Quantec-Serie
MEMD	<u>00-208-642</u>	Justagegerät für Messpatronen mit M8 Feingewinde anwendbar für z.B. AGILUS
Option KR C2	<u>00-228-327</u>	Adapterkabel für die Verwendung des SEMD als Justagegerät für KR C2-Systeme (KTL).

MEMD/SEMD im Einsatz



Abb. 4-19: SEMD Anschlussbeispiel

4.3.2 Ermittlung der mechanischen Nullstellung

Ablauf einer Justage

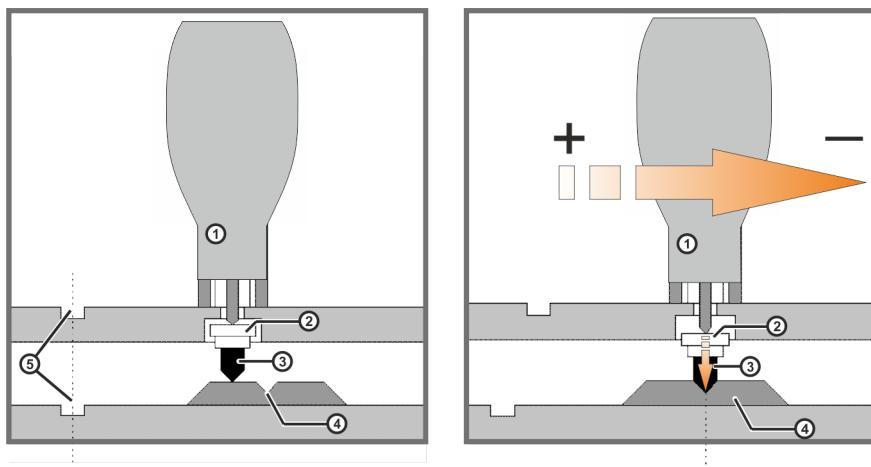


Abb. 4-20: Ablauf der EMD-Justage

- | | |
|-------------------------------------|------------------------|
| 1 EMD (Electronic Mastering Device) | 4 Messkerbe |
| 2 Messpatrone | 5 Vorjustagemarkierung |
| 3 Messstift | |

Berechnung des mechanischen Nullpunkts

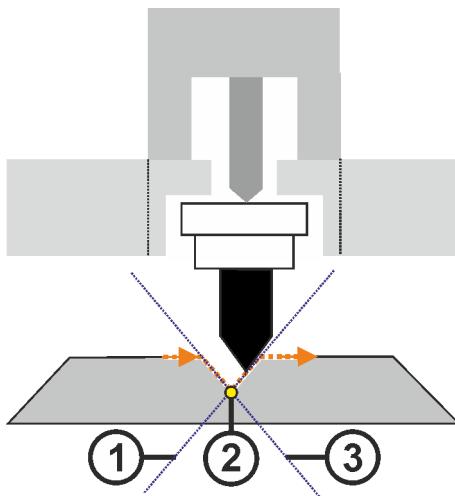


Abb. 4-21

Justiert wird, um den mechanische Nullpunkt (2) der Achse zu ermitteln. Dabei bewegt sich die Achse so lange, bis der Messstift die Justagekerbe einmal durchfahren hat. Anschließend wird die exakte Position der Kerbe ermittelt. Der Schnittpunkt der fallenden (1) und steigenden (3) Flanke wird berechnet und entspricht dem tiefsten Punkt (2). Der tiefste Punkt (2) entspricht der mechanischen Nullstellung der Achse. Die Achse wird auf den in den Maschinendaten hinterlegten Justagewert (Gradangabe) abgeglichen. Jede Achse ist daher mit einer Messpatrone, einer Messkerbe und einer Vorjustagemarkierung ausgestattet.



Bei verschiedenen Robotertypen wie z. B. Agilus 1 und KR CYBER-TECH nano besitzt die Achse 6 keine Messpatrone. Bei diesen Roboter-typen wird die mechanische Nullstellung mittels einer Markierung an der betreffenden Achse durchgeführt.

4.3.3 Roboter justieren

Möglichkeiten der Roboterjustage

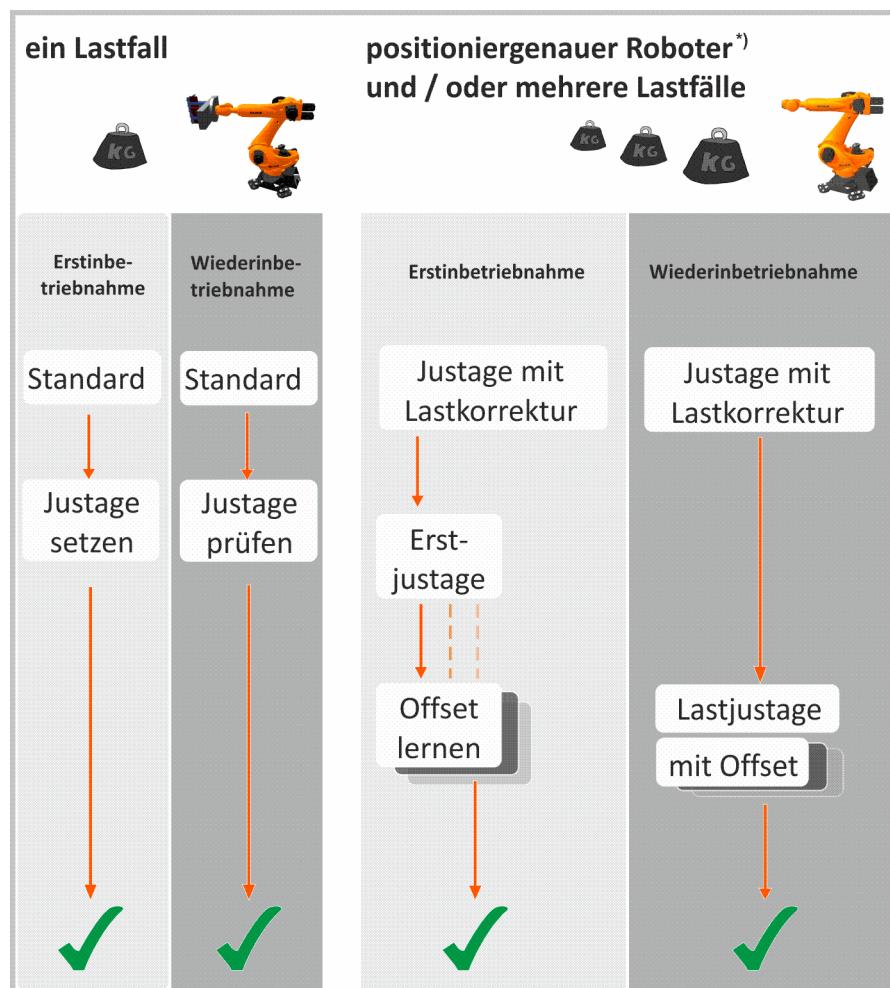


Abb. 4-22: Justagemöglichkeiten

- **Standard Justage**

Bildspalte: *ein Lastfall*

Diese Justageart wird verwendet, wenn der Roboter ein fest installiertes Werkzeug mit einem konstanten Gewicht führt, z. B. Schweißbrenner, Punktschweißzange, leerer Greifer

Auswirkungen auf mögliche Wiederinbetriebnahme:

- Wird **nach** der Programmerstellung der Roboter mit einem veränderten Werkzeug oder Last justiert, so hat dies direkten Einfluss auf die Genauigkeit der geteachten Punkte.
- Die veränderte Last führt zu einer größeren/ kleineren "Verwindung" des Roboters und dadurch auch zu veränderten Resolverwerten, die gespeichert werden.
- Die gespeicherten Resolverwerte selbst sind Berechnungsgrundlagen ("Startpositionen") für die Positionsbestimmung des Roboters.

- **Justage mit Lastkorrektur**

Bildspalte: *Positionsgenauer Roboter und/oder mehrere Lastfälle*

Diese Justageart wird verwendet, wenn:

- Ohne Demontage von Werkzeug und Zusatzlast eine Erstjustage (ohne Last) wiederhergestellt werden soll. *)

- Der Roboter in seinem Applikationsumfeld mit einem Werkzeugwechselsystem, z. B. Greifer und einer Punktschweißzange arbeitet.



*) Das positioniergenaue Robotermodell basiert auf einer Erstjustage ohne Last. Daher ist es von größter Wichtigkeit, dass bei jedem Justagevorgang dieser Zustand wiederhergestellt wird. Um dies zu vereinfachen und eine Demontage von Werkzeugen und Zusatzlasten zu vermeiden, steht der Weg über die Justage mit Lastkorrektur zur Verfügung.

Auswirkungen auf mögliche Wiederinbetriebnahme

- Der Roboter wird während der Inbetriebnahme stets Erstjustiert (ohne Werkzeug/Last) und anschließend für jedes Werkzeug die Offsets gelernt.
- Der dabei ermittelte Offset zur Erstjustage wird werkzeugspezifisch gespeichert.
- Der Roboter kann nach einem Störfall mit dem aktuell montierten Werkzeug/Last justiert werden.
- Durch den werkzeugspezifischen Offset ist der Roboter in der Lage, auf die Erstjustage zurück zu rechnen.

4.3.3.1 Erst- /Standardjustage

Vorgehensweise

Erstjustage



Abb. 4-23: Erstjustage

HINWEIS

Eine Erstjustage darf nur durchgeführt werden, wenn der Roboter ohne Last ist. Es darf kein Werkzeug und keine Zusatzlast montiert sein.

1. Roboter in Vorjustagestellung bringen.



Abb. 4-24: Beispiele Vorjustagestellung

- Einen langsamen Hand-Override einstellen.
- Die zu justierende Achse langsam mittels der \pm Tasten in die Kimm-Korn-Markierung verfahren.

2. Menüpfad: *Robotertaste > Inbetriebnahme > Justieren > EMD > Mit Lastkorrektur > Erstjustage*
3. Fenster **Erstjustage**

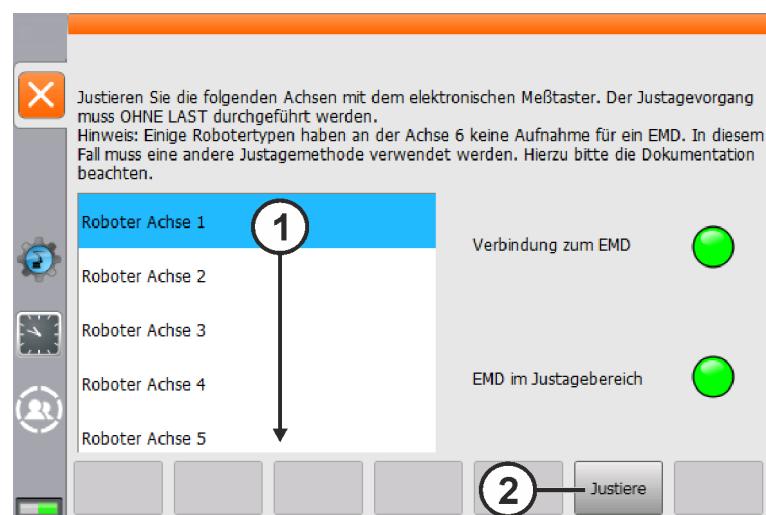


Abb. 4-25: Erst-/ Standardjustage

- Die zu justierende Achse im Menü (1) antippen.
Die aktive Achse ist blau hinterlegt.

4. **MEMD/SEMD auf die Messpatrone schrauben und anschließen.**

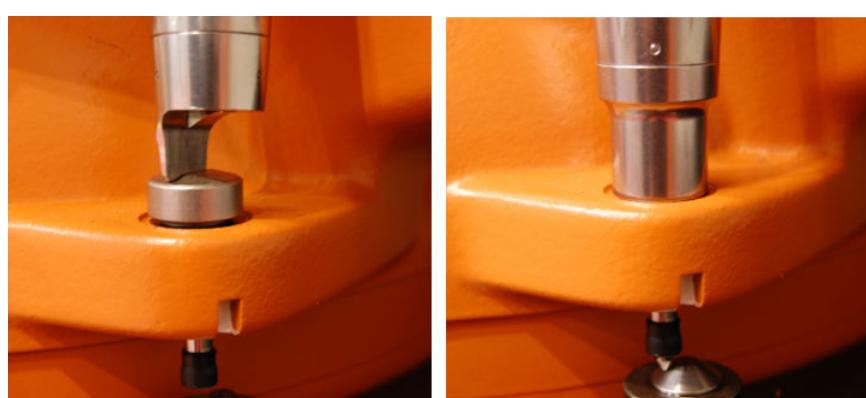


Abb. 4-26: EMD, auf Messpatrone geschraubt

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

Hierzu kann der im Justageset beiliegende Schraubendreher verwendet werden.

5. MEMD/SEMD Leitung anschließen



Abb. 4-27: EMD-Leitung, verbunden

- Das MEMD/SEMD auf die Messpatrone schrauben.
- Die Messleitung am EMD und am Anschluss X32 des Anschlusskastens des Grundgestells anschließen.



VORSICHT

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

6. Justagefahrt durchführen, Offsett lernen.

- Mit der Schaltfläche Justiere (2) die Justagefahrt starten.
- Die Meldung *Start-Taste erforderlich* wird im Meldungsfenster angezeigt.

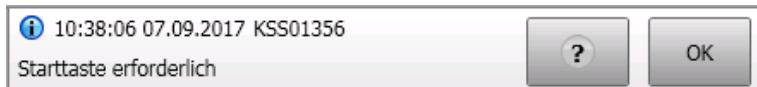


Abb. 4-28: Meldungsfenster

7. Zustimmungsschalter und Start-Taste drücken.



Abb. 4-29: Position Start-Tasten am smartPAD

- Der Roboter fährt die Justagekerbe ab und ermittelt die Justageposition.
 - Die Tasten sind während der gesamten Messfahrt gedrückt zu halten.
 - Der Roboter stoppt automatisch.
8. Die soeben justierte Achse wird im Auswahlfenster ausgeblendet.
 9. Messleitung vom EMD entfernen. Dann EMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
 10. Schritte für alle zu justierenden Achsen wiederholen.
 11. Das Fenster schließen.
 12. Messleitung vom Anschluss X32 entfernen.

Standardjustage



Abb. 4-30: Roboter bei der Standardjustage

HINWEIS

Die Standardjustage wird, im Gegensatz zur Erstjustage, mit dem amontierten Werkzeug vermessen.

- **Menüpfad:** Robotertaste > Inbetriebnahme > Justieren > EMD > Standard > Justage setzen
- Alle weiteren Schritte sind identisch mit der Erstjustage.
(>>> "Erstjustage" Seite 96)

HINWEIS

- Wurde ein Roboter Erstjustiert und Offsets gelernt, darf mit angebautem Werkzeug keine Standardjustage durchgeführt werden.
- Erst- und Standardjustage werden innerhalb der Steuerung an der gleichen Stelle gespeichert.
- Ein Rückrechnen auf die verschiedenen Werkzeuge ist dann nicht mehr möglich!
- Die programmierten Punkte können nicht mehr genau angefahren werden.

4.3.3.2 Justage mit Lastkorrektur

Situation aus dem Alltag eines Anglers

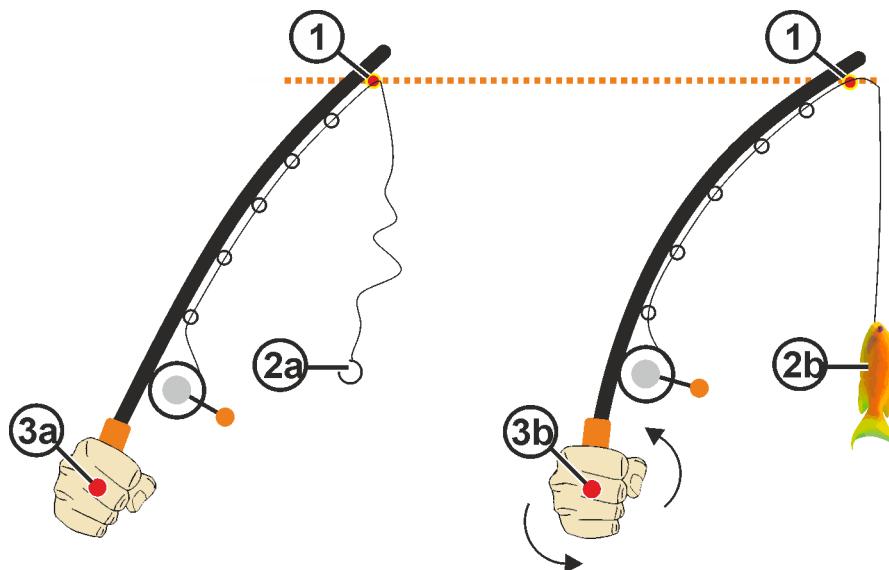


Abb. 4-31: Einfluss von Last auf einen Drehpunkt

Ein Angler möchte einen Fisch angeln. Dazu hält er seine Angel über das Wasser und versucht einen festen Punkt (1) mit einem definierten Abstand über der Wasseroberfläche zu erreichen. Dazu "dreht" er solange seine Hand, bis diese Position erreicht ist. Die Angel ist zu diesem Zeitpunkt unbelastet (2a) und "es hat noch kein Fisch angebissen". Beißt ein Fisch an (2b), wird die Angel belastet und führt zu einer Verwindung/Verbiegung, da diese elastisch ist. Um den Ausgangspunkt (1) zu halten, muss mit der Hand (3b) "nachgedreht" werden.

Wozu Offset lernen?



Abb. 4-32: Getriebeispiel und Elastizität

Durch das Gewicht des am Flansch befestigten Werkzeugs ist, aufgrund großer Hebeleinflüsse, der Roboter einer statischen Belastung ausgesetzt. Infolge materialbedingter Elastizitäten der Komponenten und Getriebe kann es zu Unterschieden in den Roboterstellungen eines belasteten und eines unbelasteten Roboters kommen. Diese Unterschiede von einigen wenigen Inkrementen wirken sich auf die Genauigkeit des Roboters aus.



Inkremeante sind die kleinsten digitalen Zählschritte, um die Drehung an einem Robotermotor zu erfassen. Inkremeante sind vergleichbar mit einer Gradeinteilung.



Inbetriebnahme des Roboters

Abb. 4-33: Offset lernen

"Offset lernen" wird mit Last durchgeführt. Die Differenz zur Erstjustage (ohne Last) wird gespeichert.

- Wenn der Roboter mit verschiedenen Lasten arbeitet, sollte die Funktion "Offset lernen" für jede Last durchgeführt werden, um die Justage einfacher wieder herstellen zu können.
- Bei Greifern, die schwere Teile aufnehmen und ein vereinfachtes Wiederherstellen der Erstjustage gewünscht ist, sollte "Offset lernen" jeweils für den Greifer und für den Greifer mit Bauteil durchgeführt werden.



Die zu verwendende Justageart (Standard oder mit Lastkorrektur) muss bei bestehenden Anlagen, der bei der Inbetriebnahme verwendeten Justageart entsprechen. Wird eine falsche Justageart verwendet, kann dies zu einem falsch justierten Roboter führen.

Vorgehensweise Offset lernen



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

1. Die Erstjustage wurde am Roboter ohne Werkzeug und Last durchgeführt.

Menüpfad: Robotertaste > Inbetriebnahme > Justieren > EMD > Mit Lastkorrektur > Erstjustage

**Abb. 4-34: Erstjustage**

2. Das für die Justage notwendige Werkzeug am Roberflansch anschrauben.



Abb. 4-35: Justage mit Werkzeug

3. Roboter in Vorjustagestellung bringen.



Abb. 4-36: Beispiele Vorjustagestellung

- Einen langsamen Hand-Override einstellen.
 - Alle zu justierenden Achsen langsam auf die Kimme-Korn-Markierung mittels der \pm -Tasten verfahren.
4. Menüpfad: Robotertaste > Inbetriebnahme > Justieren > EMD > Mit Lastkorrektur > Offset lernen
 5. Werkzeugnummer eingeben.

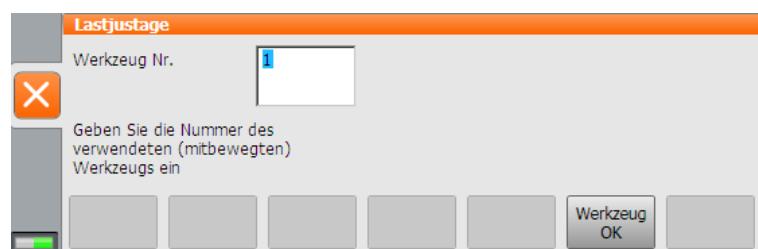


Abb. 4-37: Werkzeug Auswahl für die Lastjustage

- Werkzeugnummer im Eingabefeld numerisch eingeben.
 - Mit der Schaltfläche **Werkz. OK** bestätigen.
6. Justagefenster **Offset lernen**

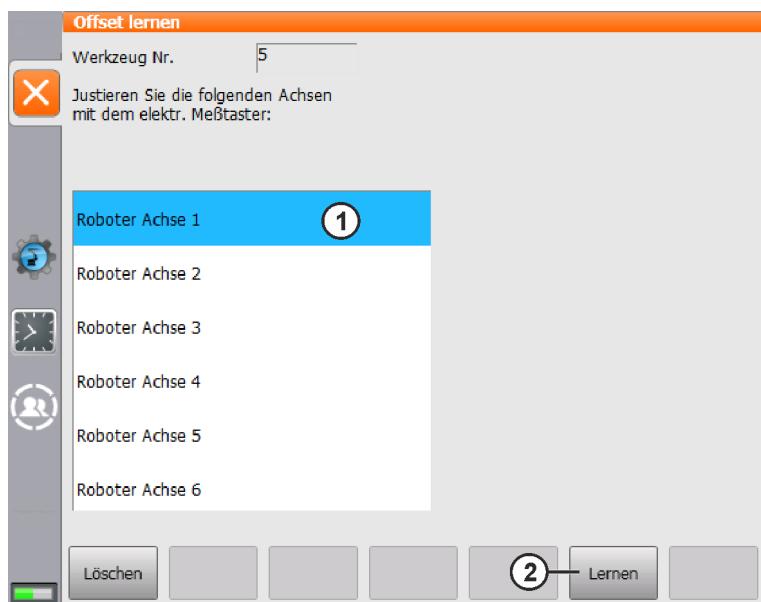


Abb. 4-38: Offset lernen

- Die zu justierende Achse im Menü (1) antippen.
Die aktive Achse blau hinterlegt.

7. **MEMD/ SEMD auf die Messpatrone schrauben und anschließen.**

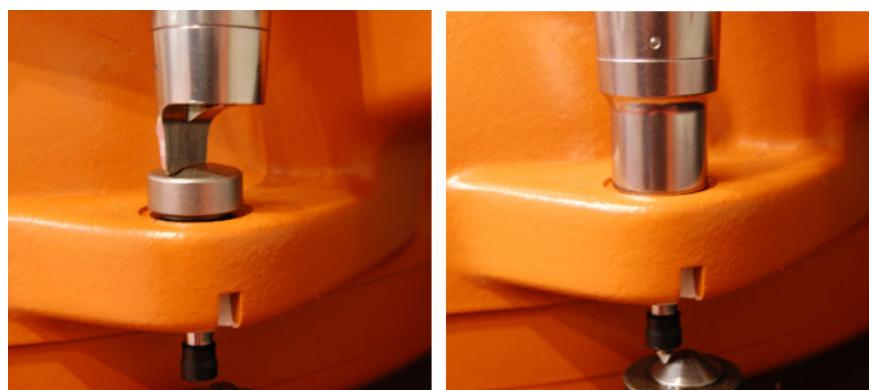


Abb. 4-39: EMD, auf Messpatrone geschraubt

- An der physikalischen Achse, die im Fenster markiert ist, die Schutzkappe der Messpatrone entfernen.
Hierzu kann der im Justageset beiliegende Schraubendreher verwendet werden.

8. **MEMD/ SMED Leitung anschließen**



Abb. 4-40: EMD-Leitung, verbunden

- Das MEMD/ SEMD auf die Messpatrone schrauben.

- Die Messleitung am EMD bzw. am Anschluss X32 des Anschlusskastens des Grundgestells anschließen.

**VORSICHT**

Das EMD immer ohne Messleitung an der Messpatrone anschrauben. Danach erst die Messleitung am EMD anbringen. Andernfalls kann die Messleitung beschädigt werden.

Ebenso beim Entfernen des EMDs immer zuerst die Messleitung vom EMD entfernen. Danach erst das EMD von der Messpatrone entfernen.

Nach der Justage die Messleitung vom Anschluss X32 entfernen. Andernfalls können Störsignale auftreten oder Schäden verursacht werden.

9. Justagefahrt durchführen.

- Mit der Schaltfläche Lernen (2) die Justagefahrt starten.
- Die Meldung *Start-Taste erforderlich* wird im Meldungsfenster angezeigt.

**Abb. 4-41: Meldungsfenster****10. Zustimmungsschalter und Start-Taste drücken.****Abb. 4-42: Position Start-Tasten am smartPAD**

- Der Roboter fährt selbstständig die Justagekerbe ab und ermittelt die Justageposition.
Die Tasten sind während der gesamten Messfahrt gedrückt zu halten.
- Der Roboter stoppt automatisch.

11. Zustimmungsschalter und Start-Taste drücken.

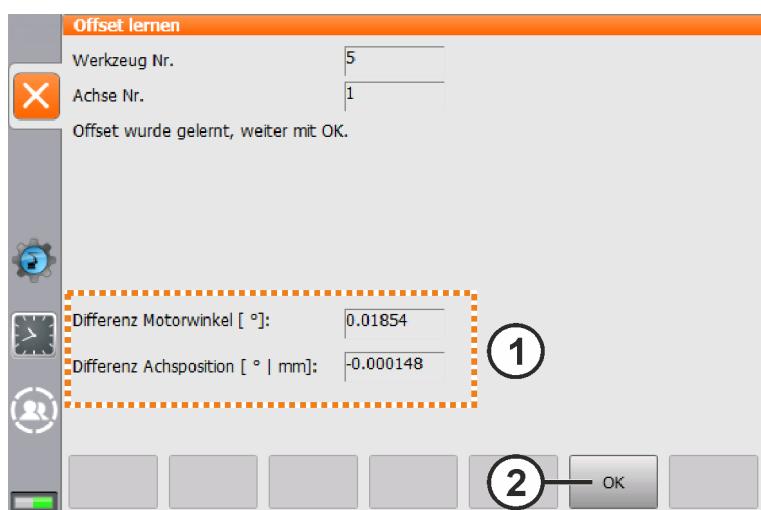


Abb. 4-43: Differenz Motorwinkel/ Achsposition

- Ein Fenster öffnet sich.
Die Abweichung bei dieser Achse gegenüber der Erstjustage wird in Grad Motorwinkel und Achswinkel angezeigt (1).
- Mit der Schaltfläche OK werden die Werte gespeichert und die nächste Achse kann justiert werden.
- Die soeben justierte Achse wird im Auswahlfenster ausgeblendet.
(>>> Abb. 4-38)

12. Messleitung vom EMD entfernen. Dann EMD von der Messpatrone entfernen und Schutzkappe wieder anbringen.
13. Schritte für alle zu justierenden Achsen wiederholen.
14. Messleitung vom Anschluss X32 entfernen.
15. Das Fenster mit **Schließen** verlassen.

Lastjustage nach einem Störfall

- Wird nach einer Störung/Kollision ein Motor oder ein Getriebe getauscht, so muss der Roboter neu justiert werden.
- Grundsätzlich ist zu prüfen, in welchem Zustand sich der Roboter befindet.

– Es befindet sich **kein** Werkzeug am Roboter.

Menüpfad: Robotertaste > Justieren > EMD > Mit Lastkorrektur > Lastjustage > Erstjustage

Der Vermessungsvorgang entspricht der Erstinbetriebnahme "Erstjustage"

(>>> 4.3.3.1 "Erst-/Standardjustage" Seite 96)

- Es befindet sich ein Werkzeug am Roboter

Menüpfad: Robotertaste > Justieren > EMD > Mit Lastkorrektur > Lastjustage > Mit Offset

Der Vermessungsvorgang entspricht der Erstinbetriebnahme "Justage mit Lastkorrektur"

(>>> 4.3.3.2 "Justage mit Lastkorrektur" Seite 100)

HINWEIS

Die **Lastjustage ohne Offset** ist nur zu verwenden, wenn der Roboter mechanisch **nicht** verändert wurde. Sie ist **nicht** zulässig bei Crash, Motortausch, Getriebetausch, etc.

4.3.3.3 Übung: Justage, Lastjustage mit Offset

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Justage, Lastjustage mit Offset**

4.3.4 Justage am Kleinroboter AGILUS

Beschreibung

Justagepositionen am KR AGILUS 1

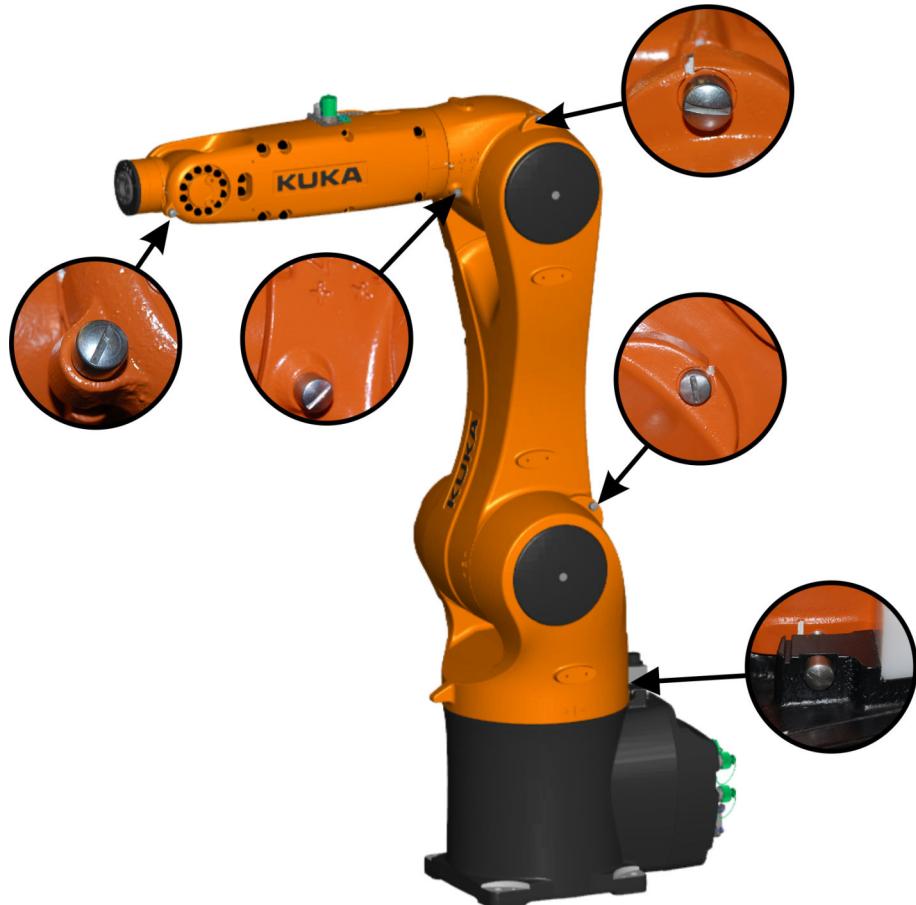


Abb. 4-44: Justageposition KR AGILUS

Winkelwerte der mechanischen Nullstellung (=Referenzwerte)

Achse	KR AGILUS 1
A1	0°
A2	-90°
A3	+90°
A4	0°
A5	0°
A6	0°

Vorgehensweise

- Die Achsen 1 - 5 werden mit dem MEMD-Sensor für kleine Justagepatronen justiert.
- 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, diese 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.

- Menüpfad: **Robotertaste > Inbetriebnahme > Justieren > Referenz** wählen.
Das Optionsfenster **Referenz-Justage** öffnet sich. Die A6 wird angezeigt und ist markiert.
- Justiere** drücken. Die A6 wird justiert und aus dem Optionsfenster ausgeblendet.
- Das Fenster schließen.

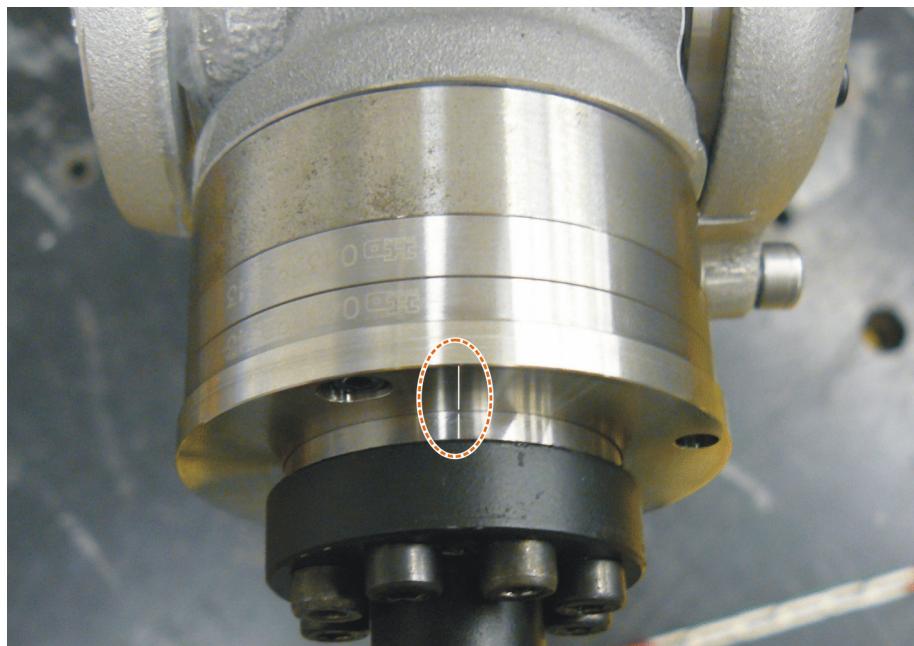


Abb. 4-45: Justagestellung A6 – Ansicht von oben vorn

Justagepositionen am KR AGILUS 2



Abb. 4-46: KR Agilus 2, Justagepatronen



Im Nachfolger KR Agilus 2 ist für die Justage der Achse 6 eine Justagepatrone vorhanden.

4.3.5 Wichtige Justagedateien

Justage LOG-Dateien

- Wird der Roboter justiert, werden spezifische Justagedaten in einer LOG-Datei protokolliert.
- Die ermittelten Offsets werden unter C:\KRC\ROBOTER\LOG\Mastery.log in Grad gespeichert.
- Folgende spezifische Justagedaten werden in der **Mastery.log** gespeichert:
 - Zeitstempel (Datum, Uhrzeit)
 - Achse
 - Seriennummer des Roboters
 - Justagewert (*FirstEncoderValue*)
 - Werkzeugnummer
 - Offset Wert (*Encoder Difference*) in Grad am Motor
- **Beispiel Mastery.log:**

```
Date: 01.09.11 Time: 13:41:07 Axis 1 Serialno.:
864585 First Mastering (FirstEncoderValue: 1.138909)
Date: 01.09.11 Time: 13:42:07 Axis 2 Serialno.: 864585
First Mastering (FirstEncoderValue: 0.644334)
Date: 01.09.11 Time: 13:42:56 Axis 3 Serialno.: 864585
First Mastering (FirstEncoderValue: 0.745757)
Date: 01.09.11 Time: 13:43:29 Axis 4 Serialno.: 864585
First Mastering (FirstEncoderValue: 1.450234)
Date: 01.09.11 Time: 13:44:03 Axis 5 Serialno.: 864585
First Mastering (FirstEncoderValue: 0.686983)
Date: 01.09.11 Time: 13:44:30 Axis 6 Serialno.: 864585
First Mastering (FirstEncoderValue: 0.901439)
...
Date: 01.09.11 Time: 14:07:10
Axis 1 Serialno.: 864585
Tool Teaching for Tool No 1 (Encoder Difference:
-0.001209)
Date: 01.09.11 Time: 14:08:44
Axis 2 Serialno.: 864585
Tool Teaching for Tool No 1 (Encoder Difference:
0.005954)
...
```

Justage-Datei (*.cal)

- Die **[Seriennummer].cal** wird auf dem EDS der RDC gespeichert und beinhaltet die Justagewerte für den Roboter.
- Diese ist in der Regel nicht sichtbar, kann aber kopiert oder ersetzt werden.
Hierzu sind gesonderte Kenntnisse und ein höheres Benutzerprofil notwendig.

Unter **C:\KRC\Roboter\RDC\[Ser.Nr.].cal** werden zusätzlich die Justagepositionen abgespeichert, wenn

- auf der HMI im Menü *Inbetriebnahme > Roboterdaten* der Button "**RDC Daten speichern**" gedrückt wird.
- Nach einem Neustart der Steuerung wird der Ordner RDC wieder gelöscht.

```
[AbsolutMotorValues]
Axis1=0.463675
Axis2=16.257612
Axis3=36.209384
Axis4=66.48317
Axis5=20.9606263
Axis6=68.8641154

[MotorDifference for Tool 1]

[MotorDifference for Tool 2]

...
[MotorDifference for Tool 14]
Axis1=-0.157869
Axis2=0.553673
Axis3=-1.059499
Axis4=-0.169838
Axis5=-0.877667571
Axis6=-0.32594979

[MotorDifference for Tool 15]

...
[MotorDifference for Tool 128]

[CalibrationDifference]
Axis1=0.00116852
Axis2=-0.00313334352
Axis3=-0.0100344565
Axis4=0.0115672826
Axis5=-0.0113223559
Axis6=0.02928708
```

4.4 Werkzeug- und Basisverwaltung

Beschreibung

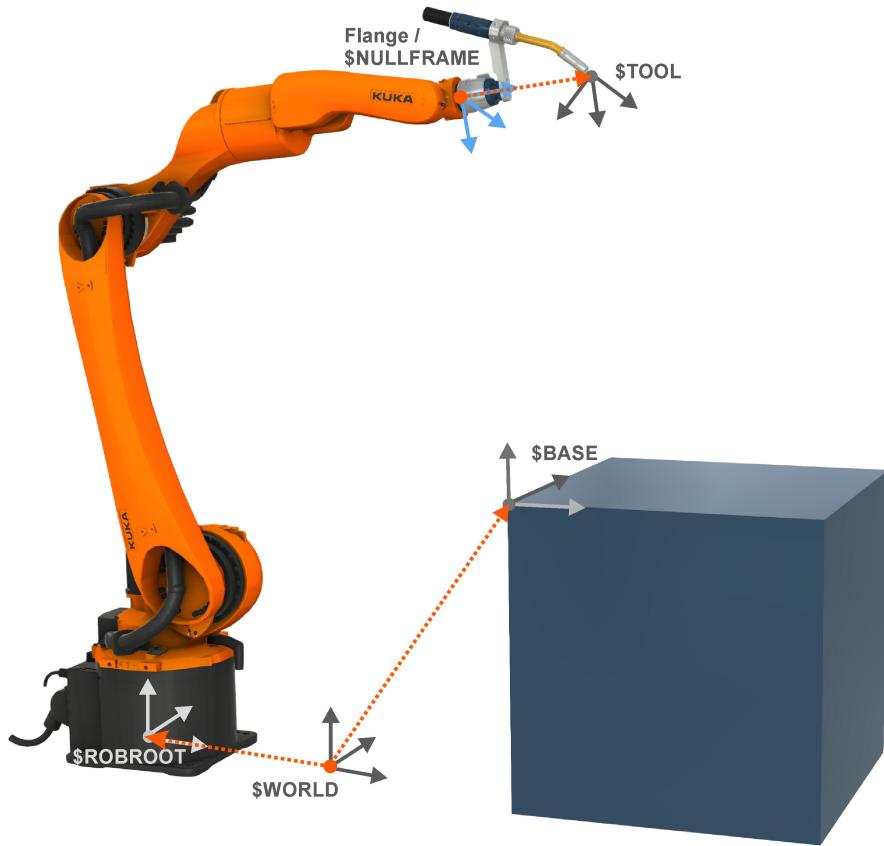


Abb. 4-47: Übersicht Koordinatensysteme

- Mit dem Plugin für die **Tool/Base-Verwaltung** steht ein Werkzeug zur Verfügung, mit dem Werkzeug- und Basissysteme angelegt, bearbeitet und gelöscht werden können.
- Die Tool/Base-Verwaltung ist wie folgt verfügbar:
 - auf dem smartPAD der Steuerung
(>>> [4.4.1 "Werkzeug- und Basisverwaltung auf dem smartPAD"](#)
[Seite 111](#))
 - in WorkVisual



- Das nachträgliche Ändern von Werkzeug- und Basiskoordinatensystemen kann Auswirkung auf bereits erstellte Programme oder geätzte Positionen haben.
- Änderungen der Werkzeug- und Basiskoordinatensysteme sollten mit einem bestehenden Archiv oder Speicherabzug zusammengeführt/ abgeglichen werden.

4.4.1 Werkzeug- und Basisverwaltung auf dem smartPAD

Beschreibung

Auf dem smartPAD können Werkzeug- und Basis-Koordinatensysteme vermessen und korrigiert werden. Auch das Neuanlegen oder Löschen von Koordinatensystem ist hier möglich.

Vorgehensweise

1. Menüpfad: Robotertaste > Inbetriebnahme > Tool-/Base-Verwaltung



Abb. 4-48: Menüpfad

2. Das Fenster "Tool-/Base-Verwaltung" öffnet sich.

Die Registerkarte **Werkzeug/Werkstück** ist vorselektiert.

Im folgenden Screenshot handelt es sich um Demo-Werkzeuge des KUKA-Colleges.

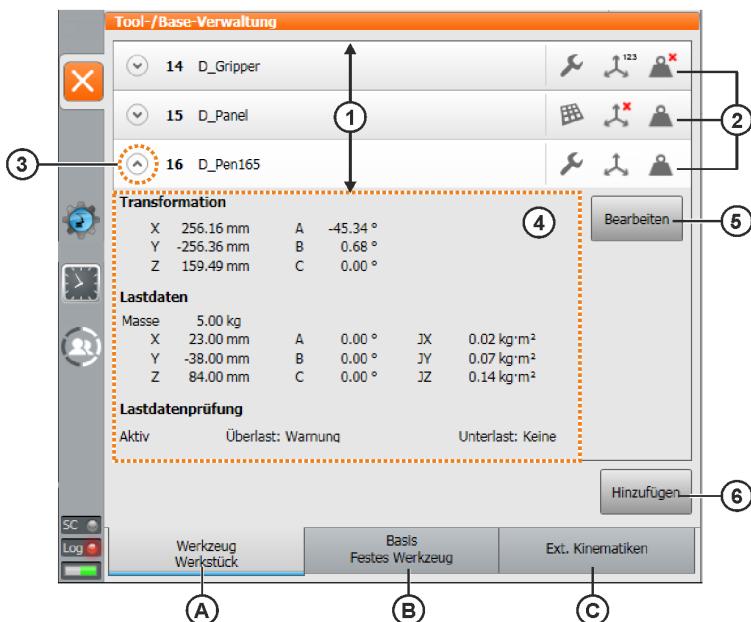


Abb. 4-49: Tool-/Base-Verwaltung

A	Registerkarte	Werkzeug oder Werkstück
B	Registerkarte	Basis oder feststehende Werkzeuge
C	Registerkarte	externe Kinematiken
1	Übersicht der auf der Steuerung vorhandenen Werkzeuge oder Werkstücke	<ul style="list-style-type: none"> – Werkzeug-/Werkstücknummer – Werkzeug-/Werkstückname – Schnellübersicht, siehe Punkt 2

2	Schnellübersicht	<ul style="list-style-type: none"> - Art der Vermessung = Werkzeug = Werkstück - Vermessung Koordinaten-system = Vermessung auf der Steuerung (z. B. 4-Punktmethode) = numerische Vermessung = keine Vermessung, es ist kein Offset bekannt - Lastdaten = Lastdaten sind hinterlegt = Lastdaten fehlen
3	Symbol	Klappt für das selektierte Werkzeug oder -stück die gespeicherten Vermessungs- und Lastdaten auf.
4	Übersicht Werkzeug oder -stück	Vermessungs- und Lastdaten <ul style="list-style-type: none"> - Transformation - Lastdaten - Lastdatenprüfung
5	Schaltfläche Bearbeiten	Öffnet das Konfigurationsfenster für die Werkzeug oder -stück Vermessung
6	Schaltfläche Hinzufügen	Legt ein neues Werkzeug oder -stück an.

3. Für die Ansicht **Basis/ Feststehendes Werkzeug** in die gleichlautende Registerkarte (B) wechseln.

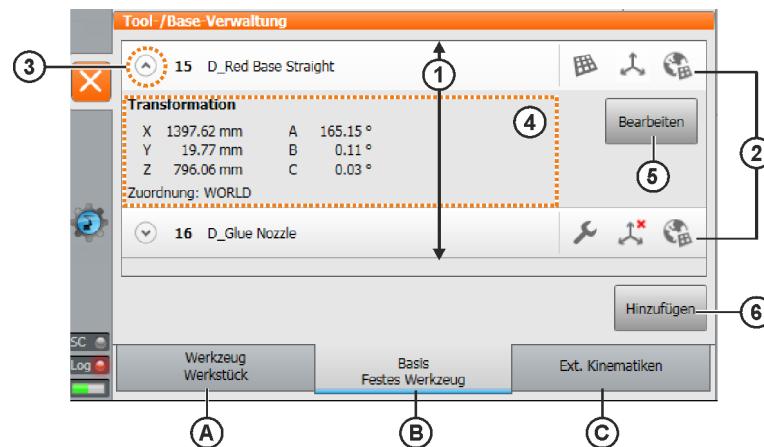


Abb. 4-50: Tool-/Base-Verwaltung

A	Registerkarte	Werkzeug oder Werkstück
B	Registerkarte	Basis oder feststehende Werkzeuge
C	Registerkarte	externe Kinematiken
1	Übersicht der auf der Steuerung vorhandenen Basen oder feststehenden Werkzeuge	<ul style="list-style-type: none"> - Nummer der Basis oder des feststehenden Werkzeugs - Name der Basis oder des feststehenden Werkzeugs - Schnellübersicht, siehe Punkt 2
2	Schnellübersicht	<ul style="list-style-type: none"> - Art der Vermessung <ul style="list-style-type: none"> = feststehendes Werkzeug = Basis-Koordinatensystem - Vermessung Koordinatensystem <ul style="list-style-type: none"> = Vermessung auf der Steuerung (z. B. 3-Punkt) = numerische Vermessung = keine Vermessung, es ist kein Offset bekannt - = Bezug des vermessenen Koordinatensystems auf das Weltkoordinatensystem
3	Symbol	Klappt für die selektierte Basis oder des feststehenden Werkzeug die gespeicherten Offset-Daten auf.
4	Übersicht Werkzeug oder -stück	Basis-Offset-Daten <ul style="list-style-type: none"> - Transformation - Zuordnung
5	Schaltfläche Bearbeiten	Öffnet das Konfigurationsfenster für die Vermessung der Basis oder feststehende Werkzeug.
6	Schaltfläche Hinzufügen	Legt eine neue Basis oder ein feststehendes Werkzeug an.

4. Soll eine **externe Kinematik** vermessen werden, in die gleichlautende Registerkarte (C) wechseln.

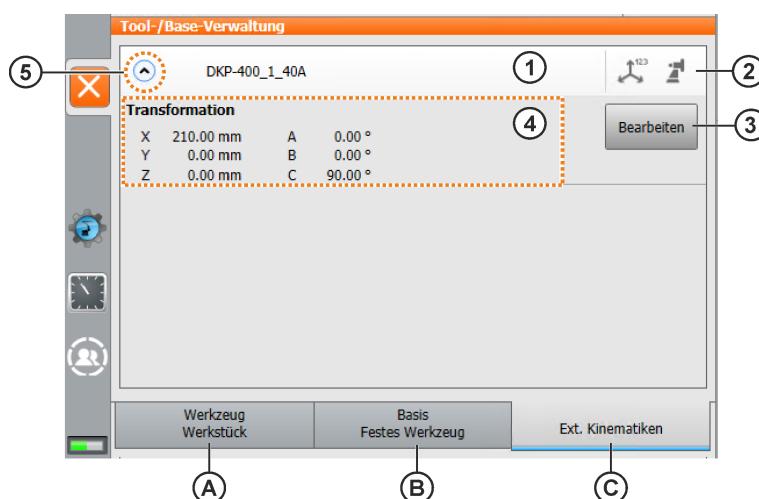


Abb. 4-51: Reiter Ext. Kinematiken

A	Registerkarte	Werkzeug oder Werkstück
B	Registerkarte	Basis oder feststehende Werkzeuge
C	Registerkarte	externe Kinematiken
1	Übersicht der auf der Steuerung vorhandenen Externen Kinematiken	<ul style="list-style-type: none"> Name des feststehenden Werkzeugs
2	Schnellübersicht	<ul style="list-style-type: none"> Typ der Kinematik  = externe Kinematik  = Roboter als RoboTeam-Teilnehmer. Eine Nummer rechts daneben gibt den RoboTeam-Index an.  = Conveyor Vermessung Koordinatensystem  = Vermessung auf der Steuerung (z. B. 3-Punkt)  = numerische Vermessung  = keine Vermessung, es ist kein Offset bekannt
3	Schaltfläche Bearbeiten	Öffnet das Konfigurationsfenster für die Vermessung externen Kinematik
4	Transformation	Transformation des Fußpunktes/ Koordinatenursprungs

5	Symbol 	Klappt für die selektierte Basis oder des feststehenden Werkzeug die gespeicherten Offset-Daten auf.
	Eine externe Kinematik kann nicht manuell über das smartPAD hinzugefügt werden. Die Kinematik muss über WorkVisual in das aktive Projekt eingebunden werden und kann auf dem smartPAD geöffnet und vermessen werden.	

4.5 Lasten am Roboter

Neben dem Flansch können am Roboter auch Lasten am Arm, Schwinge und Karussell an vorgesehenen Aufnahmepunkten befestigt werden. Für Steuerung müssen die Last und die dazugehörigen Schwerpunktabstände bekannt sein.

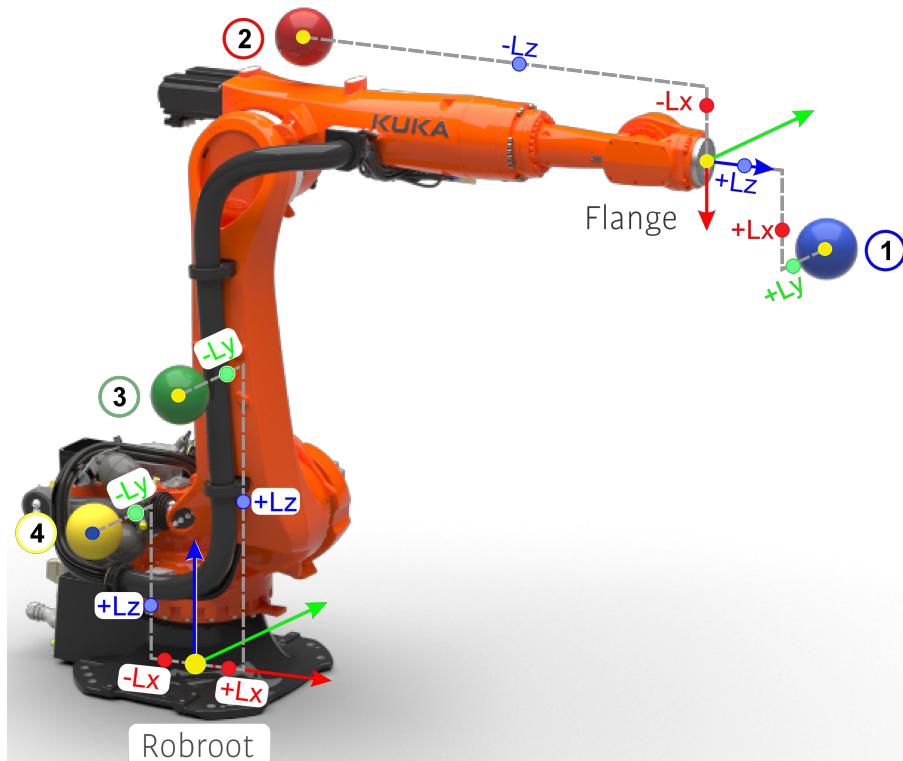


Abb. 4-52: Lasten am Roboter

- | | |
|----------------------|----------------------|
| 1 Traglast | 3 Zusatzlast Achse 2 |
| 2 Zusatzlast Achse 3 | 4 Zusatzlast Achse 1 |

4.5.1 Werkzeuglastdaten

Was sind Werkzeuglastdaten?

- Unter Werkzeuglastdaten versteht man alle am Roboterflansch montierten Lasten.
- Sie bilden eine zusätzlich am Roboter montierte Masse, die vom Roboter mitbewegt werden muss.
- Die einzutragenden Werte sind die Masse, die Lage des Schwerpunkts (Punkt, in dem die Masse wirkt) und die Massenträg-

heitsmomente mit den dazugehörenden Hauptträgheitsachsen (Orientierung der Hauptträgheitsachsen A, B und C).

- Die Traglastdaten **müssen** in die Robotersteuerung eingegeben und dem richtigen Werkzeug zugewiesen werden.
- **Ausnahme:** Ist das Optionspaket KUKA.LoadDataDetermination auf der Steuerung installiert, können während der Vermessung die Lastdaten automatisiert ermittelt werden.

Werkzeuglastdaten können folgenden Quellen entnommen werden:

- Software-Option KUKA.LoadDataDetermination (nur für Traglasten)
- Herstellerangaben
- manuelle Berechnung
- CAD-Programme

Auswirkungen der Lastdaten

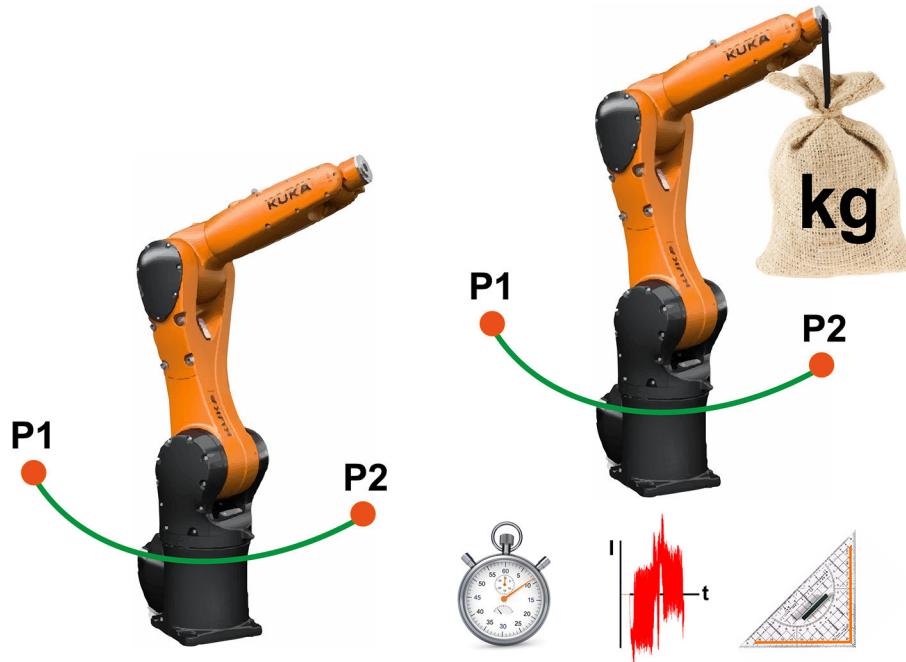


Abb. 4-53: Auswirkungen bei falschen Werkzeuglastdaten

- Die eingetragenen Werkzeuglastdaten wirken sich auf eine Vielzahl von Steuerungsabläufen aus.

Dazu gehören:

- Steuerungsalgorithmen (Berechnung der Beschleunigung)
- Geschwindigkeits- und Beschleunigungsüberwachung
- Momentenüberwachung
- Kollisionsüberwachung
- Energieüberwachung
- u.v.a.
- Deswegen ist es von größter Wichtigkeit, dass die Werkzeuglastdaten korrekt eingetragen werden.
- Führt der Roboter seine Bewegungen mit korrekt eingetragenen Werkzeuglastdaten aus,
 - kann von seiner hohen Genauigkeit profitiert werden,
 - sind Bewegungsabläufe mit optimalen Taktzeiten möglich,

- erreicht der Roboter eine lange Standzeit (durch geringen Verschleiß).

Vorgehensweise

Die Eingabe der Werkzeuglastdaten wird bei der Vermessung eines Werkzeugs beschrieben.

Siehe: (>>> "Werkzeug- /Werkstücklastdaten hinterlegen" Seite 131)

4.5.2 Zusatzlasten am Roboter

Zusatzlasten am Roboter

Zusatzlasten sind am Karussell, Schwinge oder Arm zusätzlich angebrachte Komponenten, z. B.:

- Energiezuführung
- Ventile
- Materialzuführung
- Materialvorrat

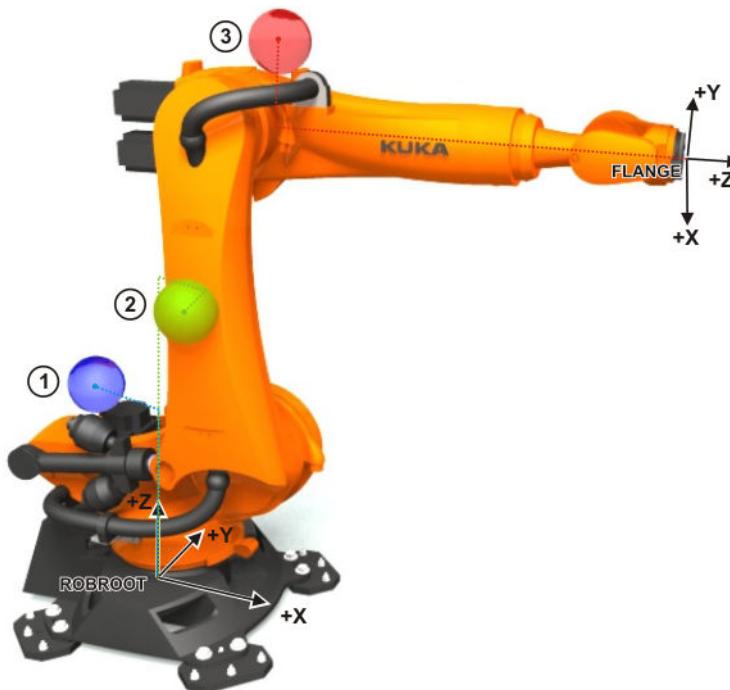


Abb. 4-54: Zusatzlasten am Roboter

HINWEIS

Nicht an jedem KUKA-Roboter ist eine Montage von Zusatzlasten an allen drei Positionen möglich.

Die Zusatzlastdaten müssen in die Robotersteuerung eingegeben werden.

Zu den notwendigen Angaben gehören:

- Masse (m) in kg
- Abstand des Schwerpunkts der Masse zum Bezugssystem (X, Y und Z) in mm
- Orientierung der Hauptträgheitsachsen zum Bezugssystem (A, B und C) in Grad (°)

- Trägheitsmomente der Masse um die Trägheitsachsen (J_x , J_y und J_z) in kgm^2

Bezugssysteme der X-, Y-, Z-Werte je Zusatzlast:

Last	Bezugssystem
Zusatzlast A1	ROBROOT-Koordinatensystem $A_1 = 0^\circ$
Zusatzlast A2	ROBROOT-Koordinatensystem $A_2 = -90^\circ$
Zusatzlast A3	FLANGE-Koordinatensystem $A_4 = 0^\circ$, $A_5 = 0^\circ$, $A_6 = 0^\circ$

Zusatzlastdaten können folgenden Quellen entnommen werden:

- Herstellerangaben
- Manuelle Berechnung
- CAD-Programme

Einflüsse der Zusatzlasten auf die Roboterbewegung

Die Angabe der Zusatzlastdaten beeinflusst die Roboterbewegung in unterschiedlicher Weise:

- Bahnplanung
- Beschleunigungen
- Taktzeit
- Verschleiß



WARNUNG

Wird ein Roboter mit falschen Zusatzlastdaten betrieben, sind Gefahr für Leib und Leben und/oder erheblicher Sachschaden die Folge.

Vorgehensweise

1. **Menüpfad:** Robotertaste > Inbetriebnahme > Zusatzlastdaten
2. Die Nummer der Achse eingeben, an der die Zusatzlast (1) befestigt wird. Mit **Weiter** (3) bestätigen.

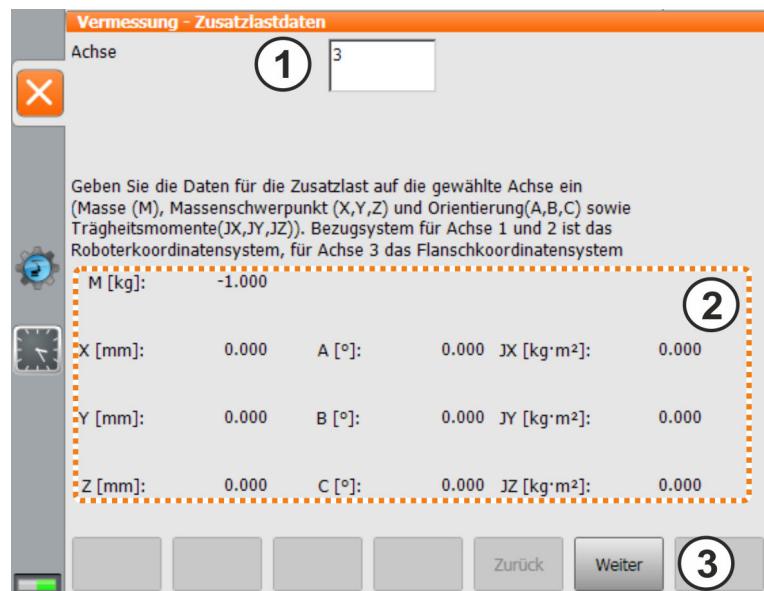


Abb. 4-55: Auswahl der belasteten Achse

3. Die Zusatzlastdaten (1) eingeben. Mit **Weiter** (2) bestätigen.

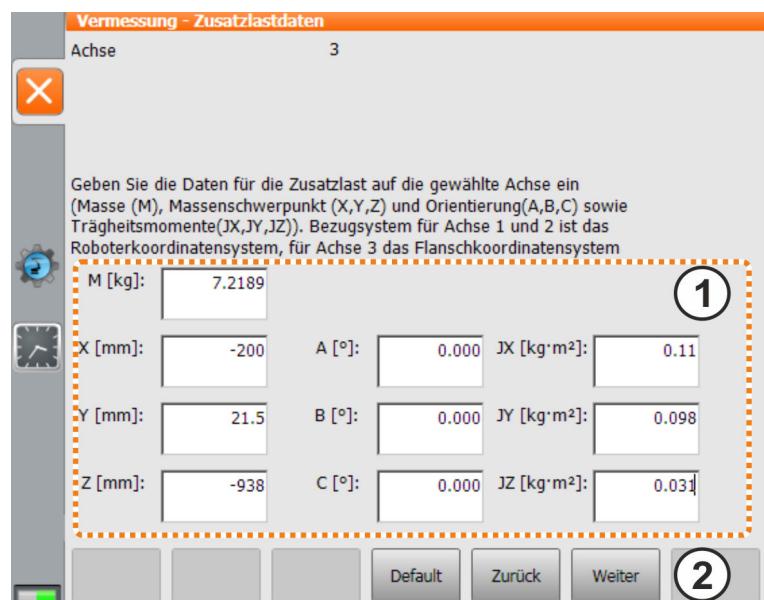


Abb. 4-56: Daten für die Zusatzlast

4. Speichern drücken.

4.6 Vermessung eines Werkzeuges

Beschreibung

Ein Werkzeug *vermessen* bedeutet, dass ein Koordinatensystem generiert wird, welches in einem Referenzpunkt des Werkzeuges seinen Ursprung hat. Diesen Referenzpunkt nennt man **TCP** (Tool Center Point), das Koordinatensystem ist das **Werkzeug-Koordinatensystem**.



Abb. 4-57: Beispiele für vermessene Werkzeugkoordinatensysteme

- Ein Werkzeug *vermessen* bedeutet, dass ein Koordinatensystem generiert wird, welches in einem Referenzpunkt des Werkzeuges seinen Ursprung hat.
- Diesen Referenzpunkt nennt man **TCP** (Tool Center Point), das Koordinatensystem ist das **Werkzeug-Koordinatensystem**.

Vermessungsschritte

Bei der Vermessung wird der Abstand des Werkzeugkoordinatensystems (in X, Y und Z) zum Flanschkoordinatensystem sowie die Verdrehung zueinander (Winkel A, B und C) gespeichert.

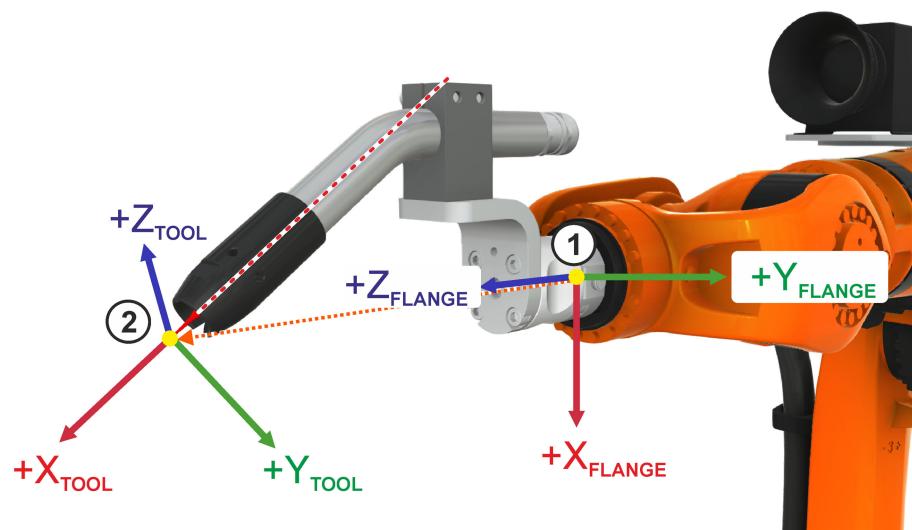


Abb. 4-58: Vermessen eines Werkzeugs

- 1 Flanschkoordinatensystem
- 2 Werkzeugkoordinatensystem

HINWEIS

In der Standardkonfiguration können 16 Werkzeug-Koordinatensysteme gespeichert werden. (Variable: TOOL_DATA[1...16]).

Vorteile

Wenn ein Werkzeug exakt vermessen wurde, ergeben sich für das Bedien- und Programmierpersonal in der Praxis folgende Vorteile:

Umorientieren um den TCP

Das Werkzeug kann um den TCP (z. B. Werkzeugspitze) umorientiert/ausgerichtet werden.

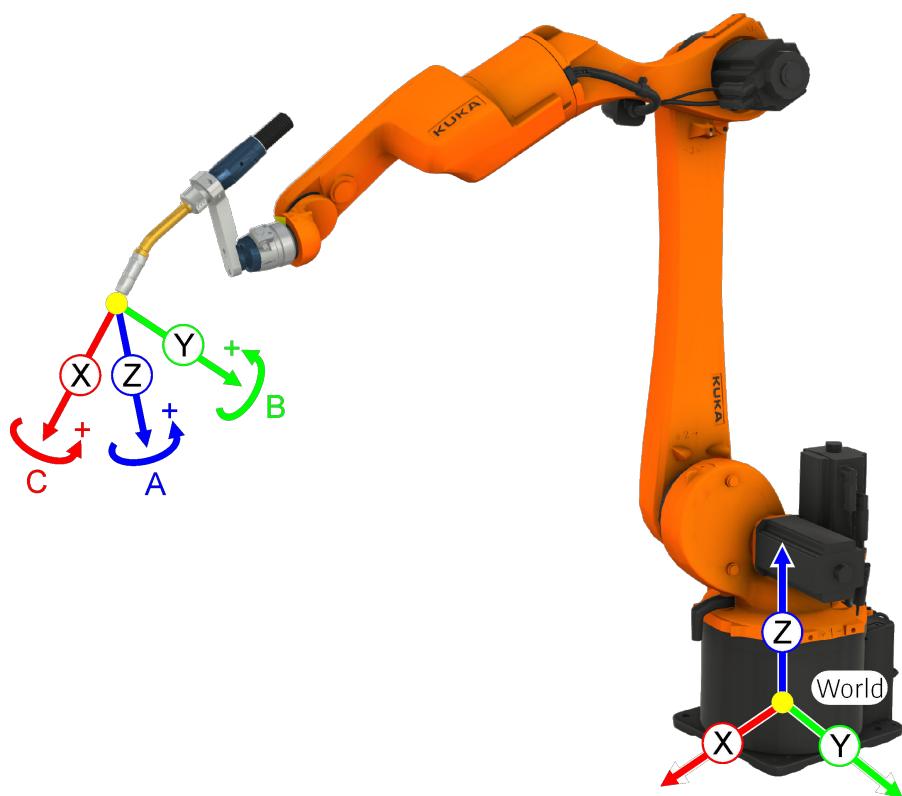


Abb. 4-59: Umorientieren um den TCP

Verfahren in Werkzeugstoßrichtung

Der Roboter kann entlang der Werkzeugstoßrichtung verfahren werden.

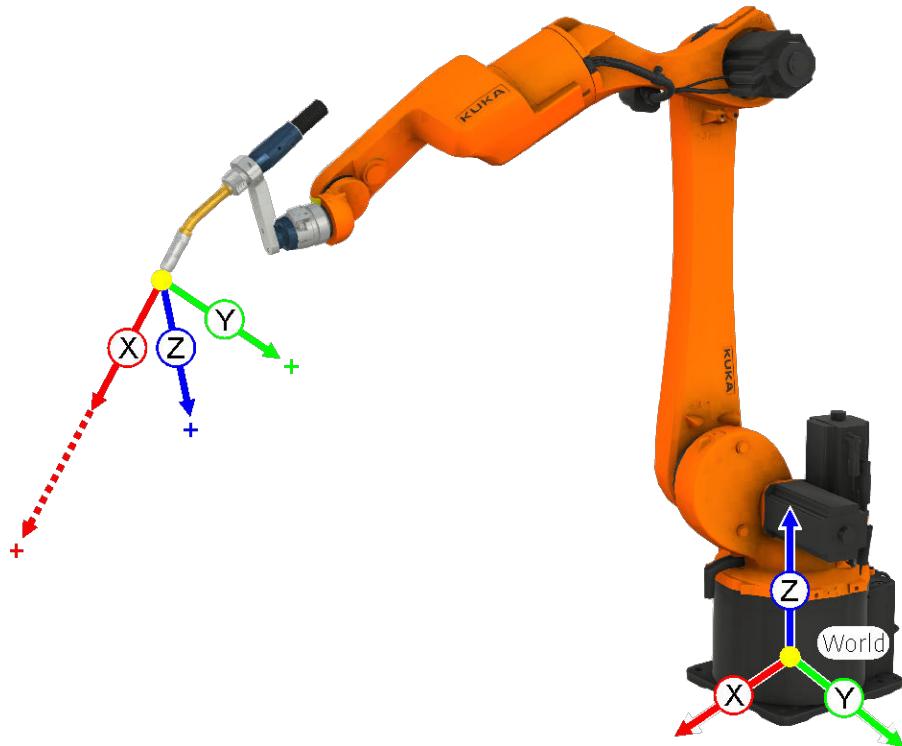


Abb. 4-60: Stoßrichtung TCP

Bahnbewegungsprogrammierung

Die programmierte Verfahrgeschwindigkeit wird entlang der Bahn am TCP gehalten.

Eine definierte Orientierungsführung entlang der Bahn ist mit einem vermessenen Werkzeug-TCP möglich.

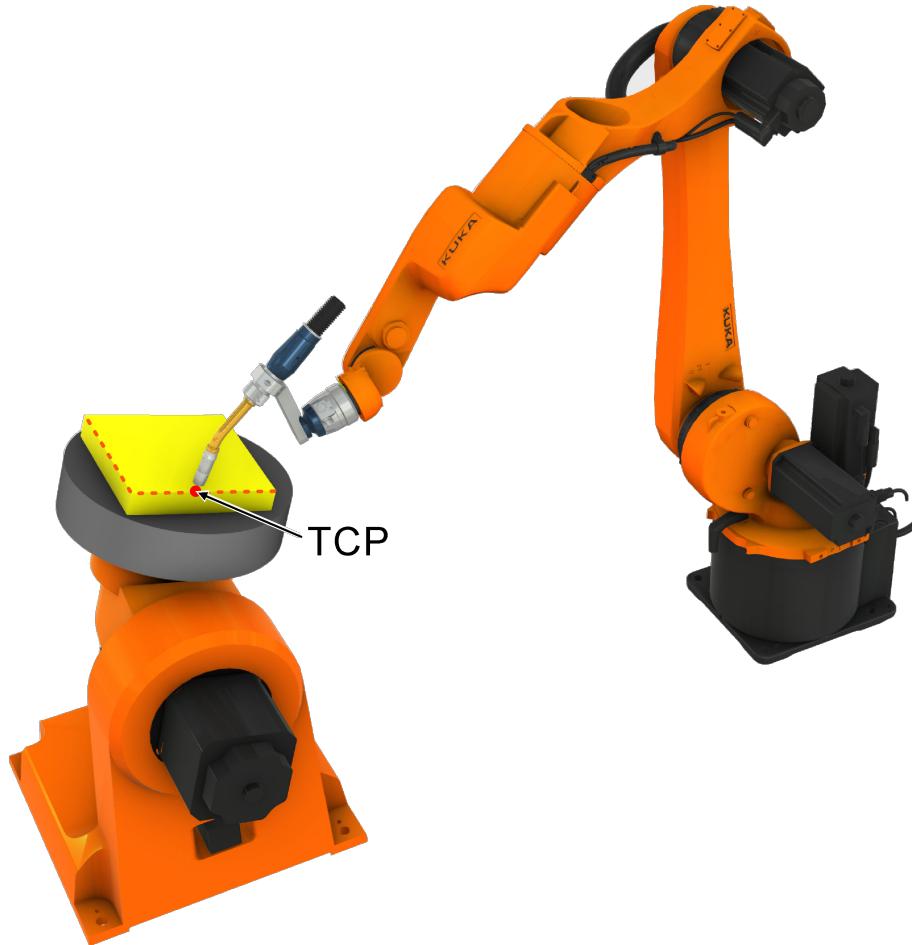


Abb. 4-61: Programmbetrieb mit TCP

Möglichkeiten der Werkzeugvermessung

Die Werkzeugvermessung besteht aus drei Schritten:

Schritt	Beschreibung
1	Ursprung des TOOL-Koordinatensystems festlegen Folgende Methoden stehen zur Auswahl: <ul style="list-style-type: none">• XYZ-4-Punkt• XYZ-Referenz
2	Orientierung des TOOL-Koordinatensystems festlegen Folgende Methoden stehen zur Auswahl: <ul style="list-style-type: none">• ABC-World• ABC-2-Punkt
Alternativ	Direkte Eingabe der Werte für den Abstand zum Flanschmittelpunkt (X, Y, Z) und die Verdrehung (A, B, C). <ul style="list-style-type: none">• <i>numerische Eingabe</i>

Schritt	Beschreibung
3	Eingabe der Werkzeuglastdaten <ul style="list-style-type: none"> • Last des Werkzeugs + Werkstück in [kg] • Schwerpunktabstand zum Roboterflansch • Trägheitsmomente

4.6.1 Neues Werkzeug/Werkstück hinzufügen

Vorgehensweise

Ein neues Werkzeug/Werkstück hinzufügen

1. Die Tool-/Baseverwaltung auf der Steuerung öffnen.
2. In der Registerkarte **Werkzeug/Werkstück** (1) auf die Schaltfläche **Hinzufügen** (2) tippen.

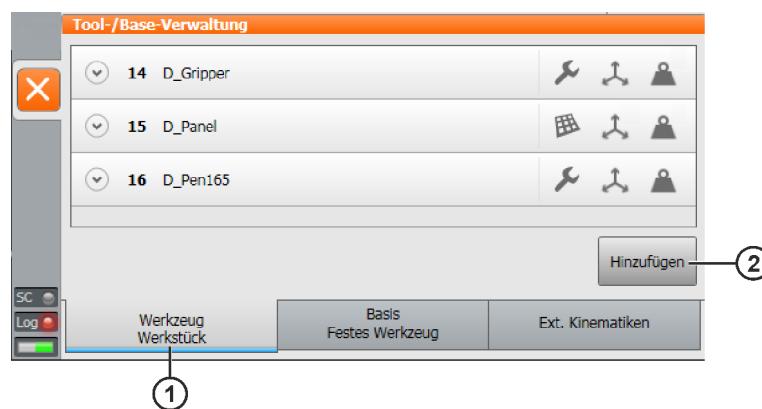


Abb. 4-62: Werkzeug/Werkstück hinzufügen

3. Werkzeug neu anlegen

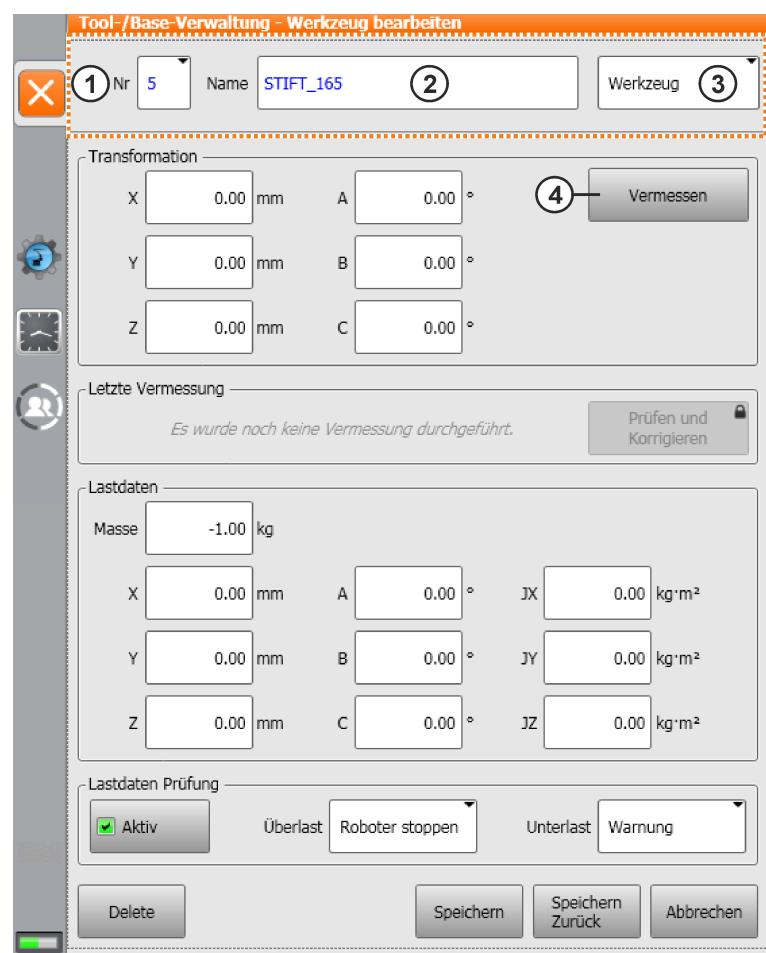


Abb. 4-63: Werkzeug neu anlegen

- Eine "freie" Werkzeugnummer im Pulldownfenster (1) auswählen.
- Name für das neue Werkzeug im Eingabefeld (2) festlegen.
- Die Art des zu vermessenden Objektes (3) festlegen.
Hier Werkzeug oder Werkstück auswählen

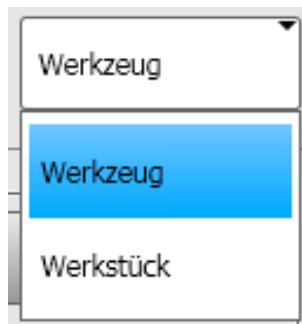


Abb. 4-64: Auswahl

- **Vermessungsmethode**

Durch das Tippen auf die Schaltfläche **Vermessen** (4) klappt ein weiteres Kontextmenü auf:



Abb. 4-65: Vermessungsmethode

- Die gewünschte Vermessungsmethode antippen.
Ermitteln des TCPs
 - **XYZ-4Punkt**
(>>> [4.6.1.1 "TCP-Vermessung 4-Punkt-Methode" Seite 127](#))
 - **XYZ-Referenz**
(>>> [4.6.1.2 "XYZ-Referenz Methode" Seite 132](#))
- Ermittlung der Orientierung
 - **ABC-Welt**
(>>> [4.6.1.3 "ABC-Welt Methode" Seite 136](#))
 - **ABC-2-Punkt**
(>>> [4.6.1.4 "ABC-2-Punkt Methode" Seite 139](#))

4.6.1.1 TCP-Vermessung 4-Punkt-Methode

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

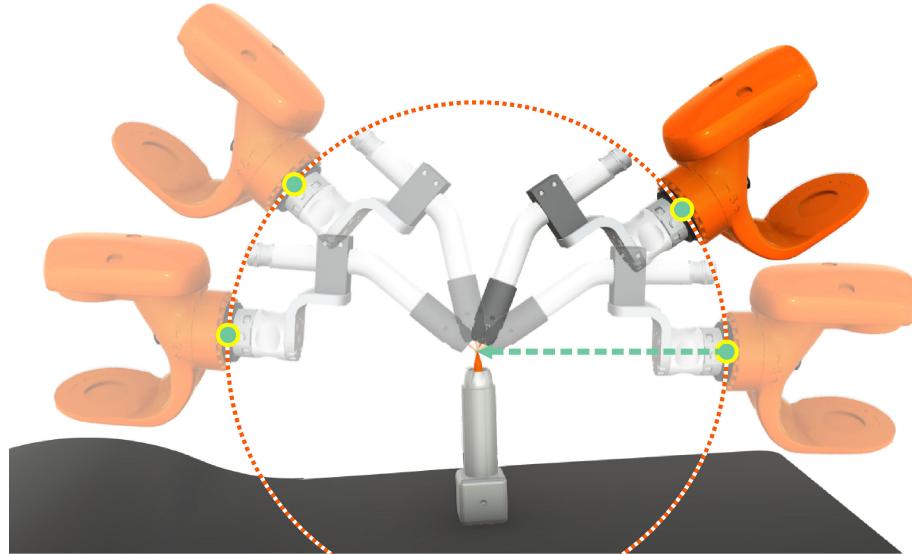


Abb. 4-66: 4-Punkt-Methode

- Mit dem TCP des zu vermessenden Werkzeugs wird ein Referenzpunkt aus 4 verschiedenen Richtungen angefahren.
- Der Referenzpunkt kann beliebig gewählt werden.
- Aus den vier abgespeicherten und vermessenen Flanschpositionen berechnet die Robotersteuerung den TCP.



Die vier Flanschpositionen, mit denen der Referenzpunkt angefahren wird, müssen ausreichend weit auseinander und dürfen nicht auf einer Ebene liegen.

Vorgehensweise

XYZ-4-Punkt-Methode

1. Ein neues Werkzeug hinzufügen.
(>>> [4.6.1 "Neues Werkzeug/Werkstück hinzufügen" Seite 125](#))
2. Den Referenzpunkt in räumlicher Nähe des Roboters positionieren.
3. Für die Vermessung die XYZ-4-Punkt-Methode wählen.



Abb. 4-67: Vermessungsmethode

4. Im Vermessungsmenü die Arbeitsanweisung beachten:
Richten Sie das Werkzeug aus verschiedenen Richtungen auf einen Referenzpunkt aus.



Abb. 4-68: XYZ-4-Punkt Vermessung

- Den Kalibrirungspunkt/Messpunkt 1 markieren. Die entsprechende Zeile wird blau markiert (1).
- Mittels 6D-Maus oder Verfahrtasten den TCP des neuen Werkzeugs an die Referenzspitze fahren.



Umso genauer die Punkte angefahren werden, je genauer wird am Ende der ermittelte TCP.
Den HOV-Handoverride an entsprechender Stelle reduzieren oder auf das inkrementelle Handverfahren zurückgreifen.

- Mittels der Schaltfläche **Touch-Up** (2) die Punktkoordinaten speichern.
5. Den Vorgang für die verbleibenden Kalibrierungspunkte/Messpunkte wiederholen.

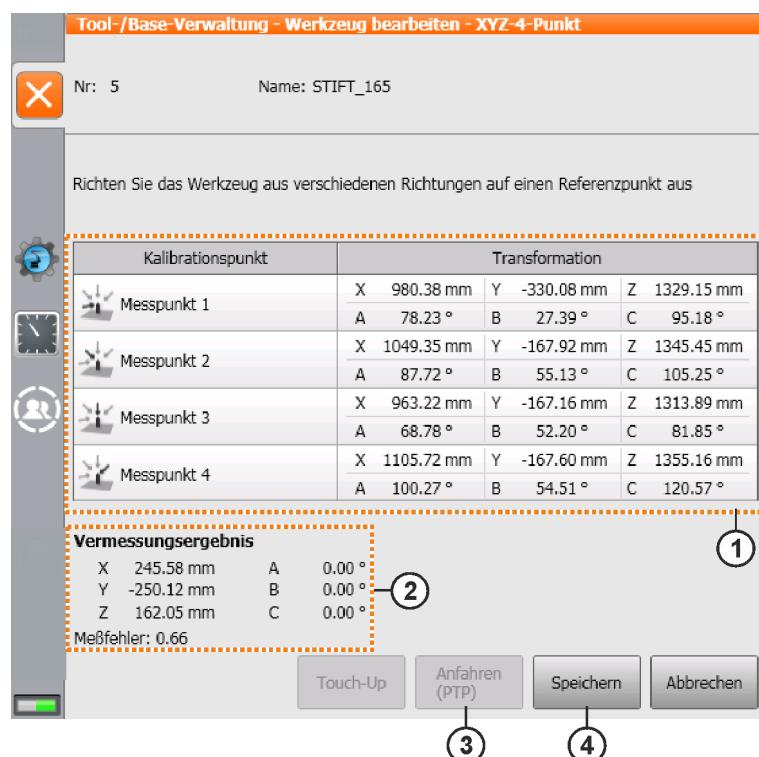


Abb. 4-69: XYZ-4-Punkt Vermessung

- Die ermittelten Koordinaten (1) werden tabellarisch dargestellt.
- Der ermittelte Werkzeugoffset sowie der **Messfehler** wird im Feld Vermessungsergebnis eingeblendet.



Ist der Messfehler zu groß, können gezielt Messpunkte wiederholt markiert werden und mittels der Schaltfläche **Anfahren (PTP)** (3) angefahren werden. In diesem Fall den Punkt präziser anfahren und das Ergebnis erneut mittels der Schaltfläche Touch-Up aufnehmen. Den Messfehler erneut prüfen.

- Den ermittelten Werkzeugoffset **Speichern** (4).
6. Werkzeugdaten prüfen und speichern.

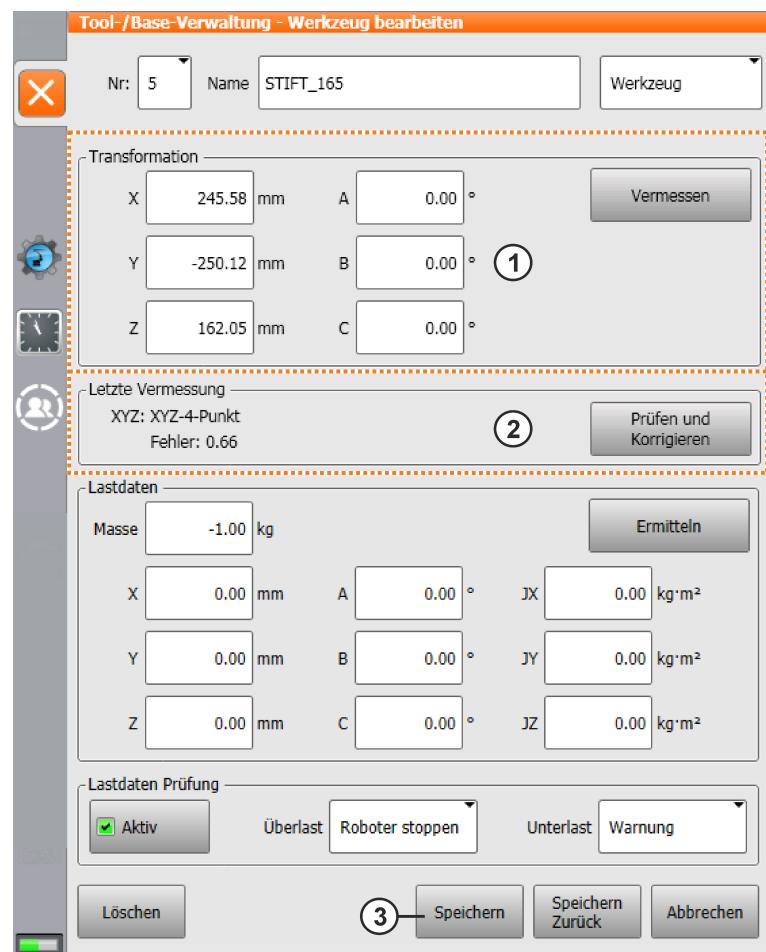


Abb. 4-70: XYZ-4-Punkt Vermessung

- Den Werkzeugoffset im Feld **Transformation** (1) prüfen.
- Die Art der Vermessung sowie der Fehler wird im Feld (2) angezeigt.
Wurde das Werkzeug vermessen, so steht Funktion **Prüfen und Korrigieren** über die gleichnamige Schaltfläche zur Verfügung.
- Alle eingegebenen Werte und Offsets **speichern** (3)

Werkzeug- /Werkstücklastdaten hinterlegen

Vorgehensweise

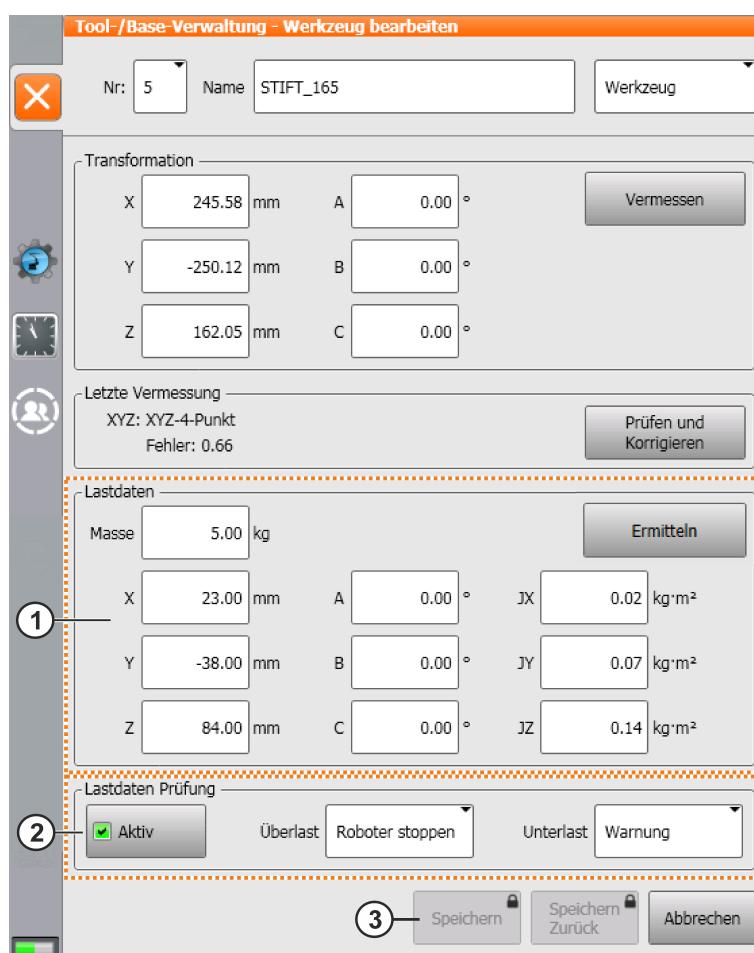


Abb. 4-71: Lastdaten

- Ein Werkzeug/Werkstück wurde/ist definiert und die Tool-/Base Verwaltung geöffnet.
(>>> [4.6.1 "Neues Werkzeug/Werkstück hinzufügen" Seite 125](#))
- Die Lastdaten des Werkzeugs und das vorgesehene Feld (1) einzutragen.



KUKA.LoadDataDetermination ermöglicht die automatische Ermittlung von Lastdaten. Ist das Optionspaket auf der Steuerung installiert, steht eine separate Schaltfläche für die Ausführung der automatischen Ermittlung zur Verfügung.

Parameter/Einheit	Beschreibung	
M	kg	Masse
X, Y, Z	mm	Lage des Schwerpunkts, relativ zum FLANGE-Koordinatensystem
A, B, C	Grad	Orientierung der Hauptträgheitsachsen, relativ zum FLANGE-Koordinatensystem
Massenträgheitsmomente:		
JX	kgm^2	Trägheit um die X-Achse des Hauptachsen-

Parameter/Einheit	Beschreibung	
JY	kgm ²	Trägheit um die Y-Achse des Hauptachsen-systems
JZ	kgm ²	Trägheit um die Z-Achse des Hauptachsen-systems

3. Lastdaten Prüfung aktivieren/deaktivieren und konfigurieren (3).

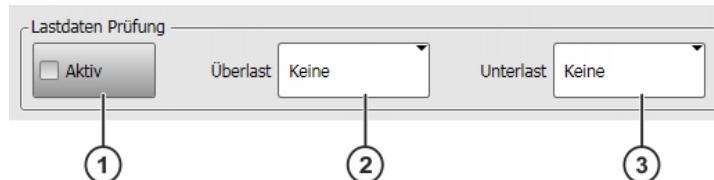


Abb. 4-72: Lastdatenprüfung

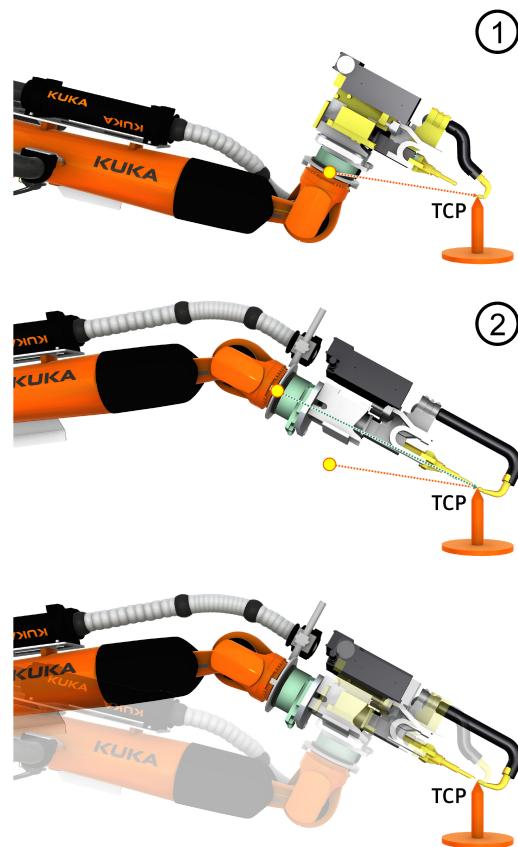
Pos.	Beschreibung
1	<p>Häkchen: Die Prüfung für das im gleichen Fenster angezeigte Werkzeug ist aktiv. Bei Über- oder Unterlast erfolgen die definierten Reaktionen.</p> <p>Kein Häkchen: Die Prüfung für das im gleichen Fenster angezeigte Werkzeug ist inaktiv. Bei Über- oder Unterlast erfolgt keine Reaktion.</p>
2	<p>Hier kann definiert werden, welche Reaktion bei Überlast erfolgen soll.</p> <ul style="list-style-type: none"> – Keine: Keine Reaktion – Warnung: Die Robotersteuerung gibt folgende Zustandsmeldung aus: <i>Beim Überprüfen der Roboterlast (Tool {Nr.}) ist Überlast ermittelt worden.</i> – Roboter stoppen: Die Robotersteuerung gibt eine Quittierungsmeldung 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.

4.6.1.2 XYZ-Referenz Methode

Beschreibung

Die **XYZ-Referenz-Methode** wird verwendet, um mehrere typgleiche Werkzeuge mit ähnlicher Geometrie in der Steuerung anzulernen.

Hierzu wird ein bereits bekanntes Werkzeug (1) als Referenz verwendet. Die Robotersteuerung vergleicht die Flanschpositionen und errechnet den TCP des neuen Werkzeugs (2). Die Anzahl der notwendigen Vermessungsfahrten im Vergleich mit der **XYZ-4-Punkt-Methode** kann hier auf zwei reduziert werden.

Referenzwerkzeug**Vorgehensweise**

1. Das für die Vermessung notwendige Referenzwerkzeug am Roboterflansch befestigen/wechseln
2. Ein neues Werkzeug wurde über die Werkzeug- und Basisverwaltung hinzugefügt.
(>> **4.6.1 "Neues Werkzeug/Werkstück hinzufügen"** Seite 125)
3. Über die Schaltfläche **Vermessen** die Methode **XYZ-Referenz** auswählen.

**Abb. 4-73: Vermessungsmethode**

4. Die **Werkzeug- und Basisverwaltung** öffnet sich.

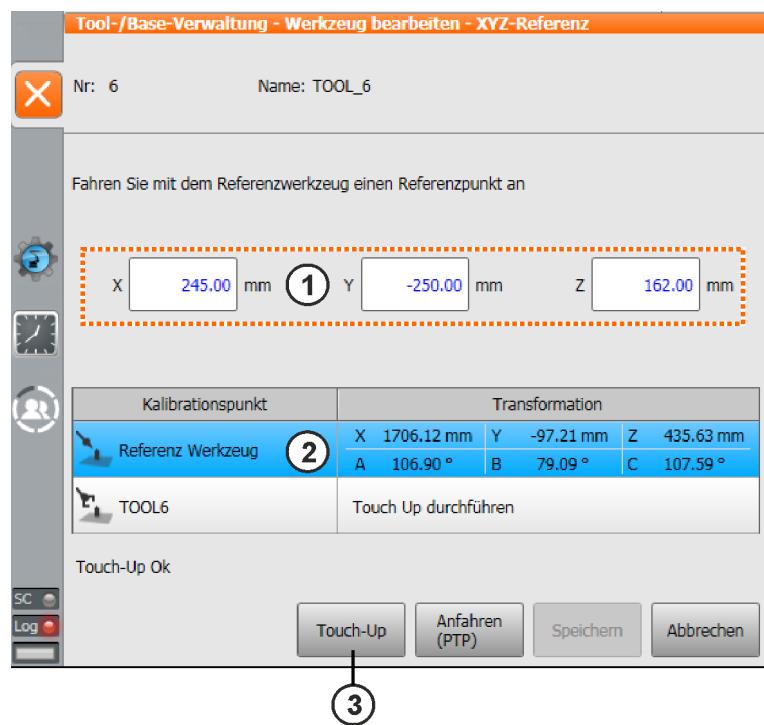


Abb. 4-74: XYZ-Referenz

- Die Offsetwerte (TCP) des Referenzwerkzeugs (1) eintragen.
Diese werden benötigt, um richtig auf den Roboterflansch zurückrechnen zu können.
Ist das Referenzwerkzeug selbst auf der Steuerung vorhanden, so können die Angaben aus der Werkzeug- und Basisverwaltung entnommen werden.
- Auf die Zeile **Referenz Werkzeug** (2) tippen.
Die Zeile wird blau markiert.
- Den TCP des Referenzwerkzeug auf den frei im Raum gewählten Referenzpunkt fahren.
- Die Position mittels der Schaltfläche **Touch-Up** (3) aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.



Mittels der Schaltfläche **Anfahren (PTP)** können bereits geteachte Punkte wiederholt angefahren werden.

5. Das Referenzwerkzeug am Roboterflansch entfernen und gegen das zu vermessende Werkzeug wechseln.

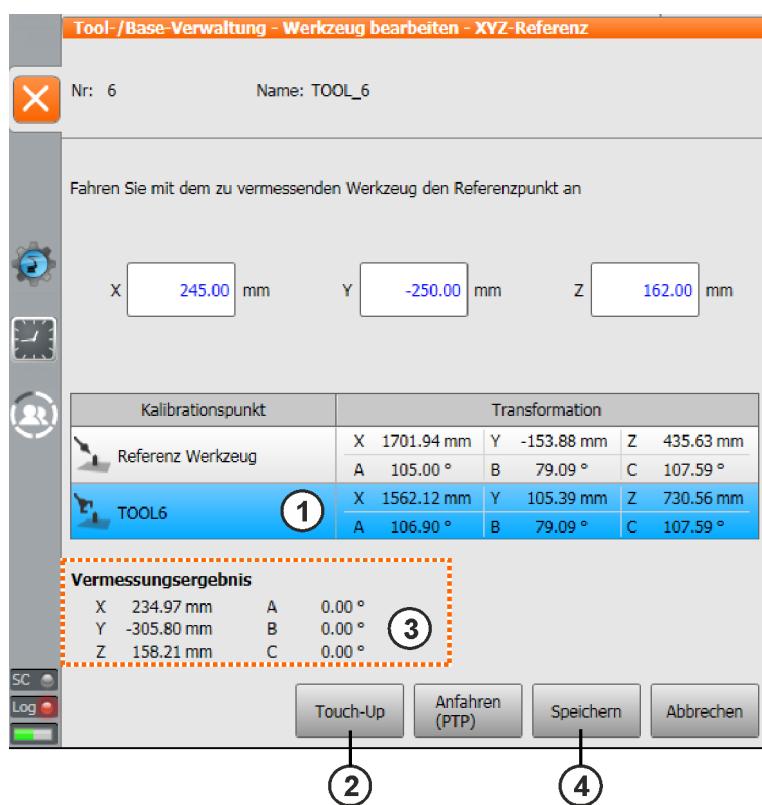


Abb. 4-75: XYZ-Referenz

- Den TCP des Werkzeugs auf den Referenzpunkt fahren.
 - Auf die Zeile mit dem **Werkzeugnamen** (1) tippen.
Die Zeile wird blau markiert.
 - Mit der Schaltfläche **Touch-Up** (2) die Position übernehmen
Die ermittelten Werte werden im Feld Transformation angezeigt.
 - Der ermittelte Werkzeugoffset wird im Fenster **Vermessungsergebnis** (3) angezeigt.
6. Das Vermessungsergebnis mit der Schaltfläche **OK** (4) sichern.

i Aus der Werkzeugübersicht der Tool-/Base-Verwaltung können die Offsetwerte für das Referenzwerkzeug entnommen werden.

The screenshot shows the 'Tool-/Base-Verwaltung' window. It displays transformation values for 'TOOL_6': X 245.0 mm, Y -250.0 mm, Z 162.0 mm, and angles A -45.00°, B 0.00°, C 0.00°. A red dashed box highlights the 'Transformation' section. Below it, the 'Lastdaten' section shows mass properties: Masse 5.69 kg, X 32.00 mm, Y -52.00 mm, Z 91.00 mm, and moments of inertia: JX 0.02 kg·m², JY 0.03 kg·m², JZ 0.06 kg·m². At the bottom, it says 'Lastdatenprüfung' with status 'Aktiv' and 'Überlast: Roboter stoppen'.

4.6.1.3 ABC-Welt Methode

Beschreibung

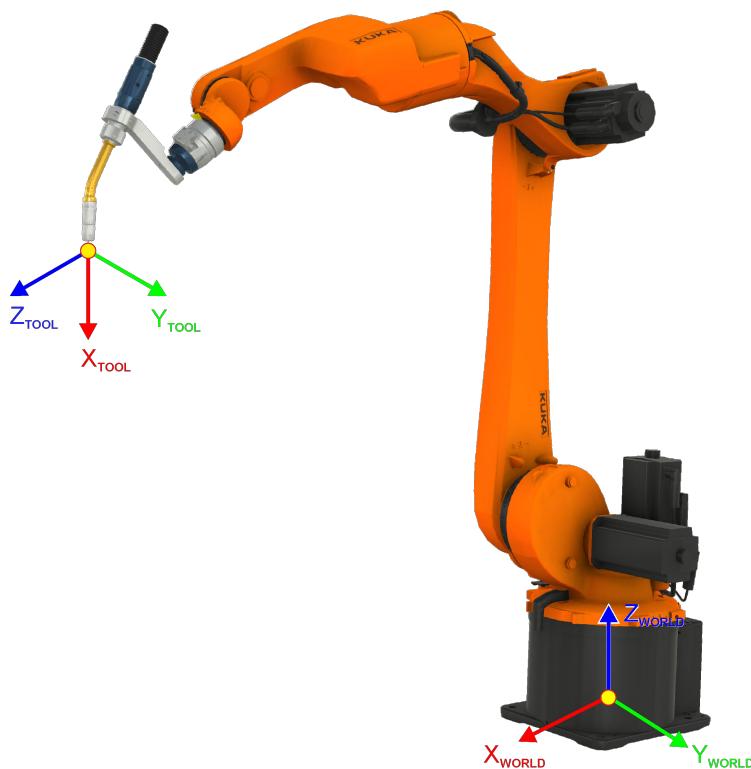


Abb. 4-76: ABC-World-Methode

- Die Achsen des TOOL-Koordinatensystems werden parallel zu den Achsen des WORLD-Koordinatensystems ausgerichtet.
- Dadurch wird der Robotersteuerung die Orientierung des TOOL-Koordinatensystems bekanntgegeben.

Varianten

Die Methode hat zwei Varianten:

- **5D**
 - Mit der Variante 5D wird die Stoßrichtung des Werkzeugs festgelegt.
 - Die Stoßrichtung ist in der Standardeinstellung die X-Achse.
 - Die Richtung der anderen Achsen wird vom System festgelegt und ist für den Benutzer nicht ohne Weiteres zu erkennen.
 - Anwendungsbereich: z. B. MIG/MAG-Schweißen, Laser- oder Wasserstrahlschneiden
- **6D:**
 - Mit der Variante 6D ist es möglich, alle Achsrichtungen festzulegen, nicht nur die Stoßrichtung
 - Anwendungsbereich: z. B. für Schweißzangen, Greifer oder Klebedüsen

Vorgehensweise

1. Ein neues Werkzeug wurde vermessen und in der Werkzeug- und Basisverwaltung gespeichert.

(>>> 4.6.1.1 "TCP-Vermessung 4-Punkt-Methode" Seite 127) oder

(>>> 4.6.1.2 "XYZ-Referenz Methode" Seite 132)

2. Das Werkzeug in der Werkzeug- und Basisverwaltung öffnen.
3. Über die Schaltfläche **Vermessen** die Methode *ABC-Welt* auswählen.



Abb. 4-77: Vermessungsmethode

4. Das Fenster **Werkzeug bearbeiten** öffnet sich.

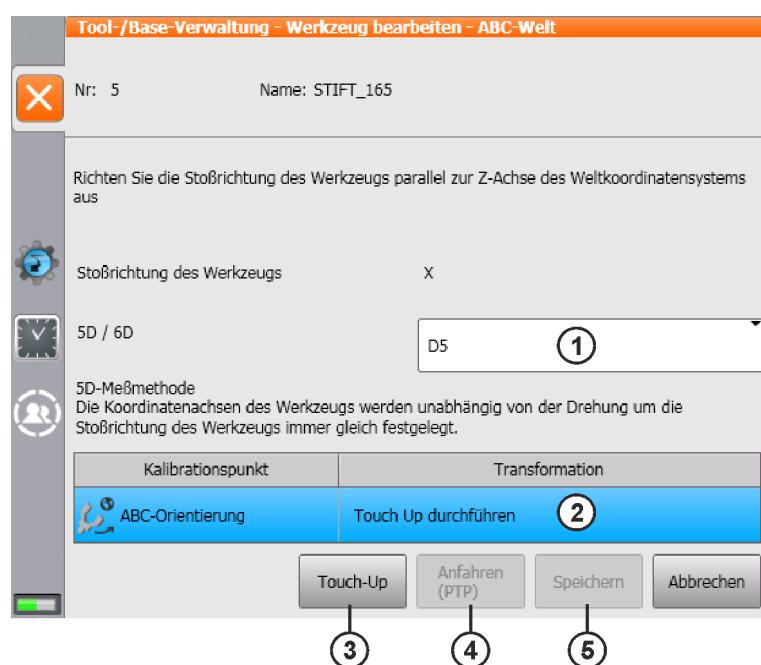


Abb. 4-78: ABC-Welt

- Im Auswahlfenster **5D/6D** (1) die Art der Orientierungsermittlung festlegen.
- Auf die Zeile **ABC-Orientierung** (2) tippen.
Die Zeile wird blau hinterlegt.
- Das Werkzeug des Roboters in Bezug auf das Weltkoordinatensystem ausrichten.
- Die Orientierung mittels der Schaltfläche **Touch-Up** (3) übernehmen.
- Die Werte mittels der gleichnamigen Schaltfläche **Speichern** (5).



Mittels der Schaltfläche **Anfahren (PTP)** können bereits geteachte Punkte wiederholt angefahren werden.

Übung: Werkzeugvermessung Stift

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Werkzeugvermessung Stift**

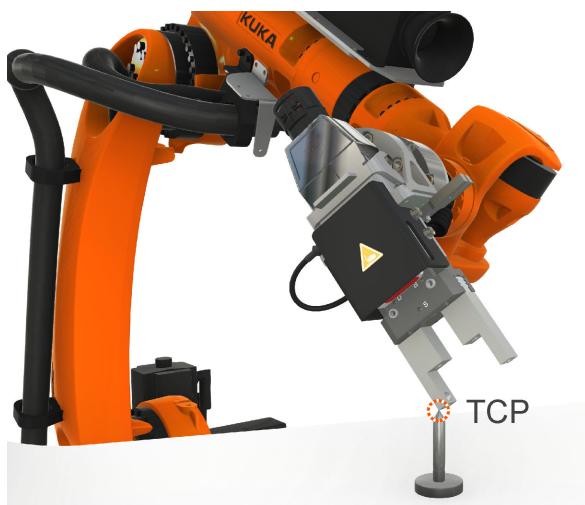
4.6.1.4 ABC-2-Punkt Methode

Vorgehensweise

Der Robotersteuerung werden die Achsen des Werkzeug-Koordinatensystems bekanntgegeben, indem ein Punkt auf der negativen X-Achse und ein Punkt in der XY-Ebene angefahren werden. Diese Methode wird verwendet, wenn nicht nur die Stoßrichtung sondern auch alle Achsrichtungen festgelegt werden sollen.

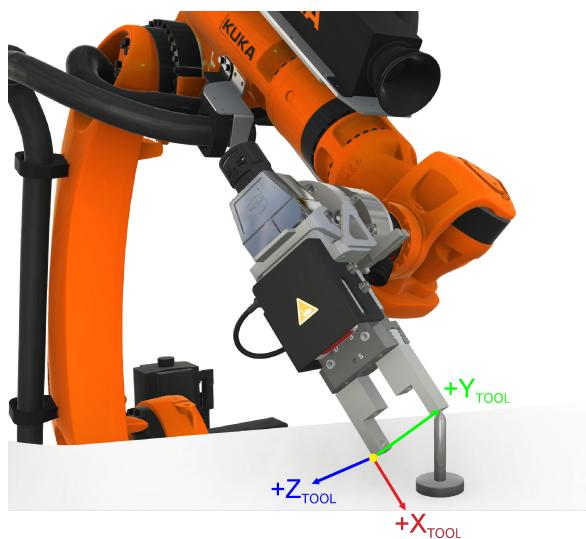
Kalibrierungspunkte

Schritt 1: TCP



Schritt 2: X-Achse

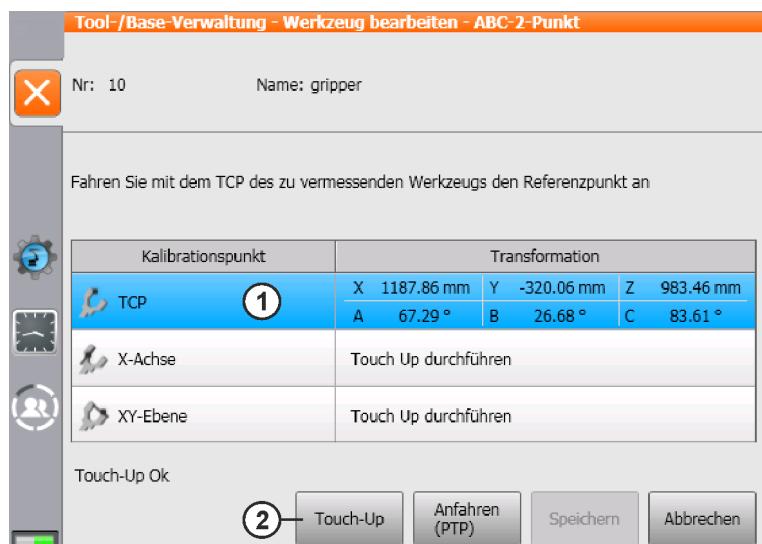


**Schritt 3:
XY-Ebene****Vorgehensweise**

1. Ein neues Werkzeug wurde vermessen und in der Werkzeug- und Basisverwaltung gespeichert.
(>>> [4.6.1.1 "TCP-Vermessung 4-Punkt-Methode" Seite 127](#)) oder
(>>> [4.6.1.2 "XYZ-Referenz Methode" Seite 132](#))
2. Das Werkzeug in der Werkzeug- und Basisverwaltung öffnen.
3. Über die Schaltfläche **Vermessen** die Methode **ABC-2-Punkt** auswählen.

**Abb. 4-79: Vermessungsmethode**

4. Das Fenster **Werkzeug bearbeiten - ABC-2-Punkt** öffnet sich.
Kalibrierungspunkt: TCP

**Abb. 4-80: Kalibrierungspunkt, TCP**

- Auf die Zeile Kalibrierungspunkt TCP (1) tippen.
Die Zeile wird blau hinterlegt.
- Den TCP des Werkzeugs auf den festgelegten Referenzpunkt fahren.
- Die Position mittels der Schaltfläche **Touch-Up** (2) übernehmen.

5. Kalibrierungspunkte: X-Achse und XY-Ebene

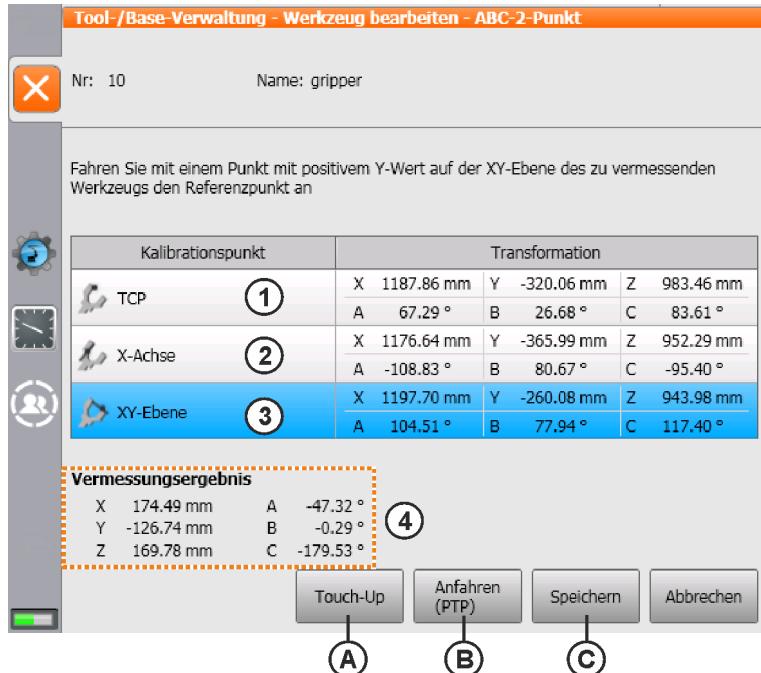


Abb. 4-81: ABC-2-Punkt, Vermessung

X-Achse

- Mit dem Referenzpunkt einen Punkt auf der negativen X-Achse (Stoßrichtung) des zu vermessenden Werkzeugs anfahren.
- Auf die Zeile Kalibrierungspunkt X-Achse (2) tippen.
Die Zeile wird blau hinterlegt.
- Die Position mittels der Schaltfläche **Touch-Up** (A) übernehmen.

Y-Achse

- Mit dem Referenzpunkt einen Punkt mit positiven Y-Wert auf der XY-Ebene des zu vermessenden Werkzeugs anfahren.
- Auf die Zeile Kalibrierungspunkt XY-Ebene (3) tippen.
Die Zeile wird blau hinterlegt.
- Die Position mittels der Schaltfläche **Touch-Up** (A) übernehmen.



Mittels der Schaltfläche **Anfahren (PTP)** können bereits geteachte Punkte wiederholt angefahren werden.

- Die ermittelte Orientierung wird im Fenster Vermessungsergebnis (4) eingeblendet und zeigen den Abstand und Verdrehung zum Flanschkoordinatensystem an.
- Diese mittels der gleichnamigen Schaltfläche **Speichern**.

Übung: Werkzeugvermessung Greifer, 2 Punkt Methode

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Werkzeugvermessung Greifer, 2 Punkt Methode**

4.7 Roboter im TOOL-Koordinatensystem bewegen

Handverfahren im TOOL-Koordinatensystem

Beim Handverfahren im TOOL-Koordinatensystem ist es möglich, den Roboter linear entlang der Achsen eines vermessenen Werkzeugs zu bewegen. Das Koordinatensystem ist nicht ortsfest (World-/Basekoordinatensystem), sondern wird vom Roboter geführt. Dabei bewegen sich **alle** Roboterachsen. Der Ursprung des TOOL-Koordinatensystems wird **TCP - Tool Center Point** genannt und entspricht dem Arbeitspunkt des Werkzeugs, z. B. die Spitze einer Klebedüse.

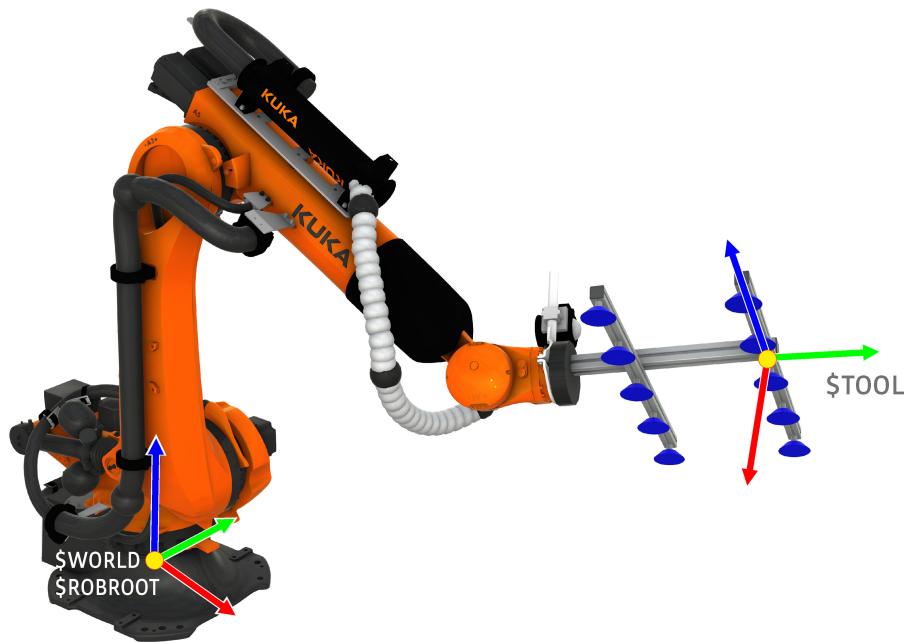


Abb. 4-82: Roboter-Werkzeugkoordinatensystem

Für das Handverfahren in der Betriebsart T1 muss der Zustimmungsschalter gedrückt sein. Dazu werden die Verfahrtasten oder die Space-Maus des KUKA smartPAD verwendet. Die Geschwindigkeit kann über den **HOV-Hand-Override** verändert werden. 16 verschiedene TOOL-Koordinatensysteme sind in der Standardkonfiguration auswählbar.



Unvermessene TOOL-Koordinatensysteme entsprechen beim Handverfahren immer dem FLANSCH-Koordinatensystem.

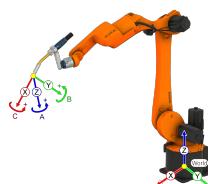


Prinzip Handverfahren Werkzeug

Ein Roboter kann auf zwei verschiedene Arten in einem Koordinatensystem bewegt werden:

**Translatorisch**

Geradlinig entlang der Orientierungsrichtungen des Koordinatensystems: X, Y, Z

**Rotatorisch**

Drehend/ schwenkend um die Orientierungsrichtungen des Koordinatensystems herum:

Winkel: A, B und C

Vorteile

- Die Bewegung des Roboters ist immer vorhersehbar, sobald das TOOL-Koordinatensystem bekannt ist.
- Es besteht die Möglichkeit, in Werkzeugstoßrichtung zu verfahren und um den Tool Center Point zu orientieren.
Dabei sind singuläre Stellungen zu vermeiden.



Unter *Werkzeugstoßrichtung* versteht man die Arbeits- oder Prozessrichtung des Werkzeugs. Im TOOL-Koordinatensystem bewegt sich das Werkzeug (in der Regel) in Richtung Werkstück, wenn in X+ verfahren wird.

Beispiele für die Werkzeugstoßrichtung sind u. a.:

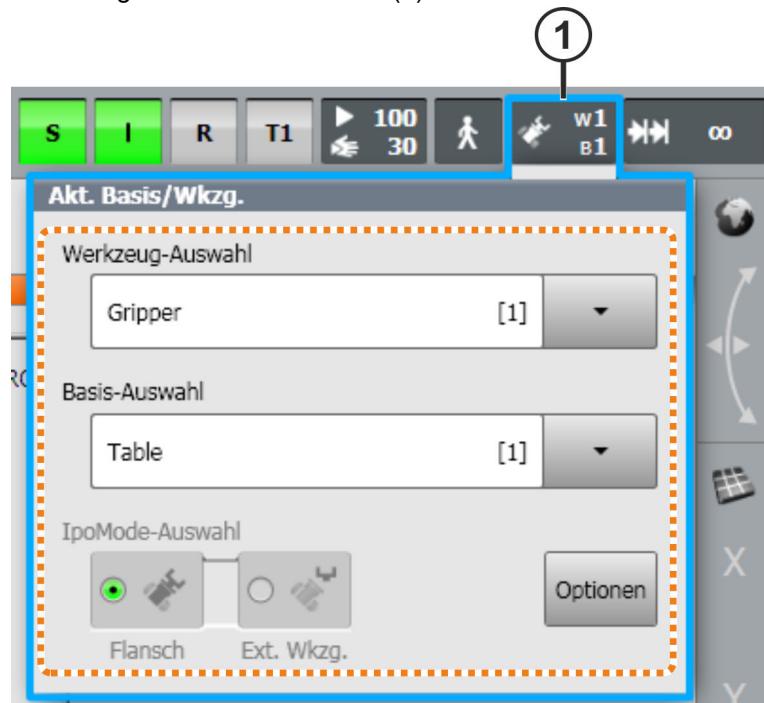
- Austrittsrichtung des Klebstoffs bei einer Klebedüse
- Greifrichtung (Führungsdrorn) beim Handling
- Richtung der festen Elektrode beim Widerstandspunktschweißen

Vorgehensweise

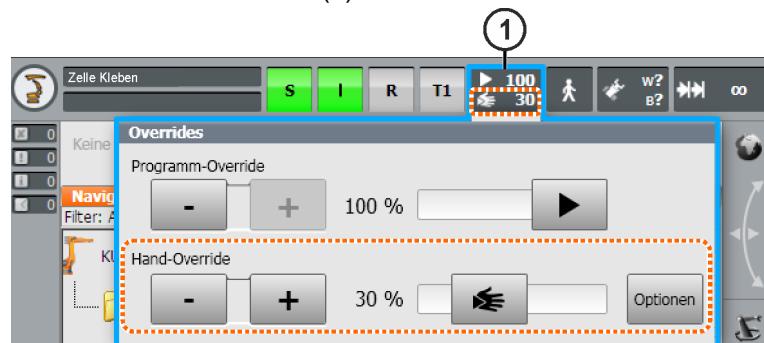
- Als zu verwendetes Koordinatensystem **Werkzeug** (1) auswählen.



2. Werkzeug-Nummer auswählen (1)



3. Hand-Override einstellen (1)



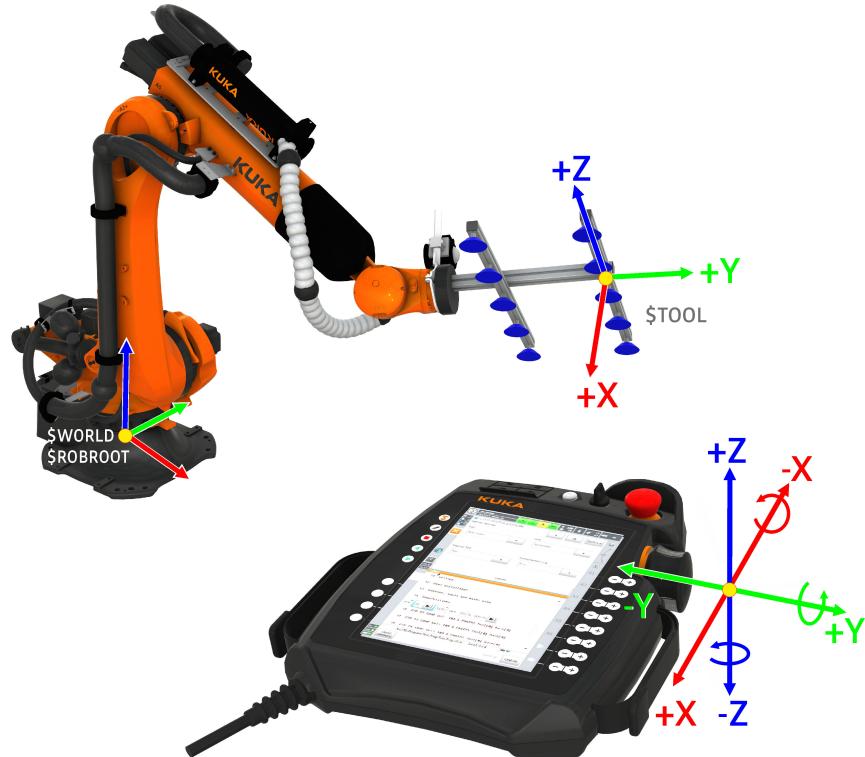
4. Einen der vier Zustimmtaster auf Mittelstellung drücken und halten.



5. Bewegen des Roboters mit den Verfahrtasten.



6. Alternativ: Verfahren mit der 6D-Maus in die entsprechende Richtung.



4.7.1 Übung: Handverfahren im Werkzeugkoordinatensystem

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Handverfahren im Werkzeugkoordinatensystem**

4.8 Vermessen einer Basis

Beschreibung

Eine Basis *vermessen* bedeutet die Erstellung eines Koordinatensystems an einer bestimmten Stelle in der Umgebung des Roboters, ausgehend vom Weltkoordinatensystem. Ziel ist es, dass sich die Handverfahrbewegungen sowie die programmierten Positionen des Roboters dann auf dieses Koordinatensystem beziehen. Deswegen bieten sich z. B. definierte Kanten von Werkstückaufnahmen, Fächern, Paletten oder Maschinen als sinnvolle Bezugspunkte für ein Basis-Koordinatensystem an.



Abb. 4-83: Basis, Verwendung

Ausgangssituation

Aus einem Ablageraster sollen USB-Sticks vom Roboter für die Weiterverarbeitung aufgenommen werden. Bei der Programmumsetzung ist ein definiertes Werkzeug und Bezugsbasis festzulegen. Daraus ergeben sich einige Voreile.

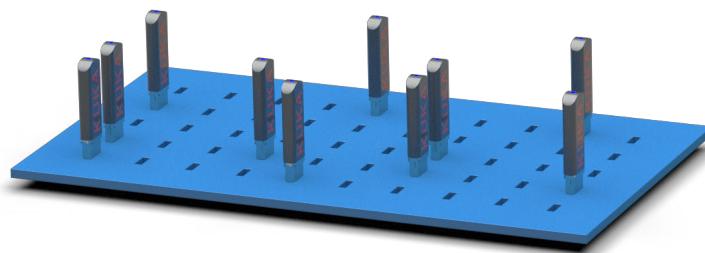


Abb. 4-84: Ausgangsbeispiel

Vorteile

Nach erfolgter Vermessung einer Basis ergeben sich folgende Vorteile:

- Verfahren entlang der Werkstückkanten

Der TCP kann entlang der Kanten der Arbeitsfläche des Werkstücks manuell verfahren werden.

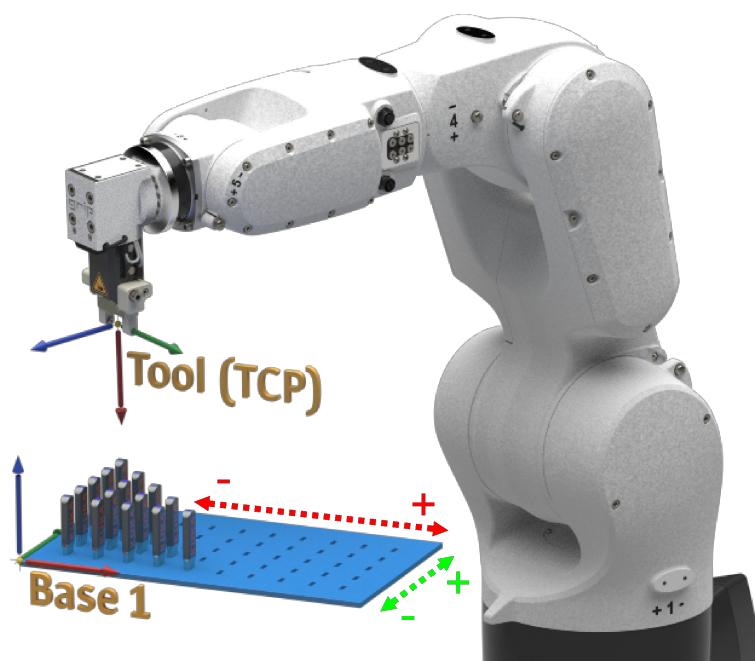


Abb. 4-85: Vorteile der Basisvermessung: Verfahrrichtung

- Bezugskoordinatensystem:
Geteachte Punkte beziehen sich auf das gewählte Koordinatensystem.

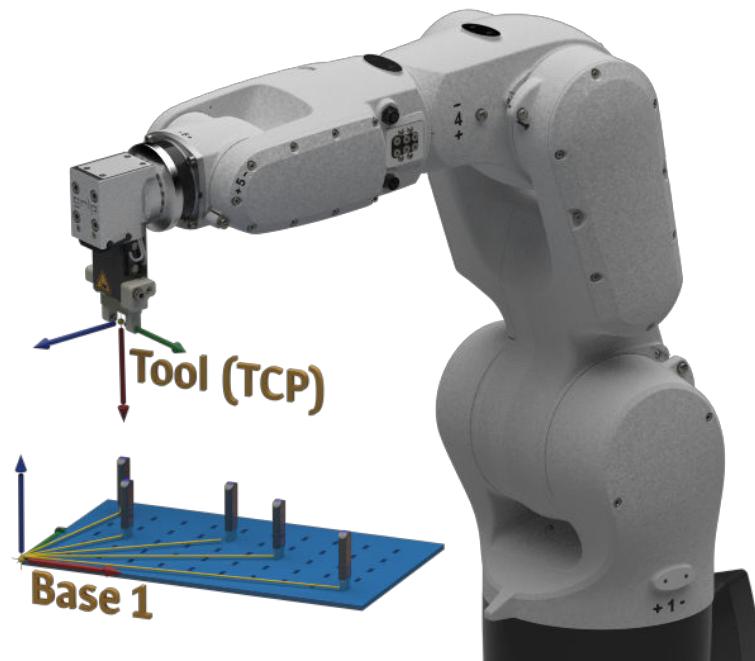


Abb. 4-86: Vorteile der Basisvermessung: Bezug auf das gewünschte Koordinatensystem

- Korrektur/Verschiebung des Koordinatensystems:
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.

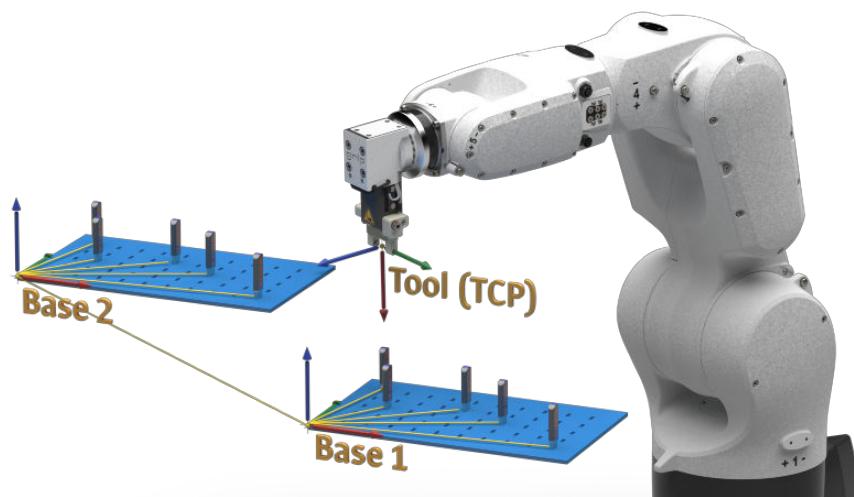


Abb. 4-87: Vorteile der Basisvermessung: Verschiebung des Basiskoordinatensystems

- Nutzung mehrerer Basiskoordinatensysteme:
Es können bis zu 32 verschiedene Koordinatensysteme angelegt und je nach Programmschritt verwendet werden.

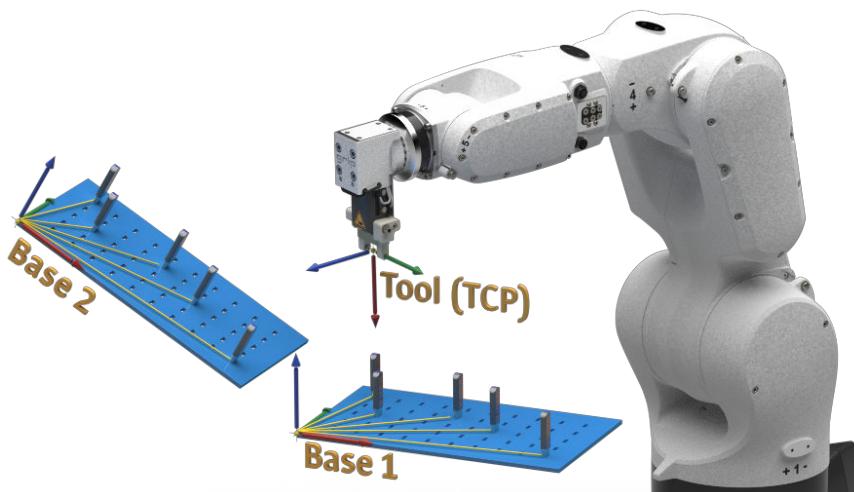


Abb. 4-88: Vorteile der Basisvermessung: Verwendung mehrerer Basiskoordinatensysteme

Möglichkeiten der Basisvermessung

Folgende Methoden zur Basisvermessung stehen zur Verfügung:

Methoden	Beschreibung
3-Punkt-Methode	1. Definition des Ursprungs 2. Definition der positiven X-Richtung 3. Definition der positiven Y-Richtung (XY-Ebene) (=> 4.8.1.1 "3-Punkt-Methode" Seite 153)
Indirekte Methode	<ul style="list-style-type: none"> Die indirekte Methode wird verwendet, wenn der Ursprung der Basis nicht angefahren werden kann, z. B. weil er im Inneren eines Werkstücks oder außerhalb des Arbeitsraums des Roboters liegt. 4 Punkte mit bekannten Bezug zur zu vermessenden Basis, deren Koordinaten bekannt sein müssen (CAD-Daten), werden angefahren. Die Robotersteuerung berechnet die Basis auf Grundlage dieser Punkte.
Numerische Eingabe	<ul style="list-style-type: none"> Direkte Eingabe der Werte für den Abstand zum Weltkoordinatensystem (X, Y, Z) und der Verdrehung (A, B, C).

4.8.1 Neue Basis/festes Werkzeug hinzufügen

Vorgehensweise

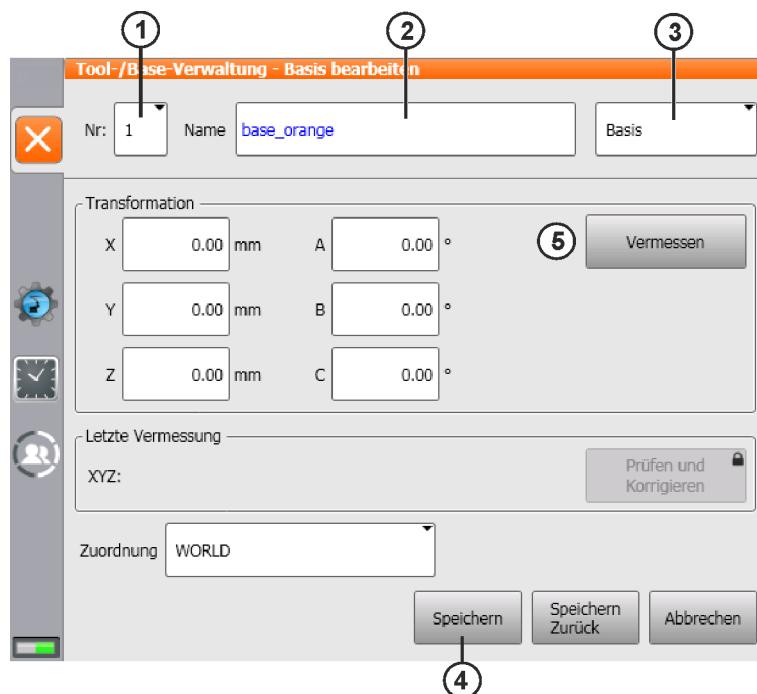
Eine neue Basis/festes Werkzeug hinzufügen

- Die Tool-/Baseverwaltung auf der Steuerung öffnen.
- In der Registerkarte **Basis/ festes Werkzeug** (1) auf die Schaltfläche **Hinzufügen** (2) tippen.



Abb. 4-89: Basis hinzufügen

Eine neu angelegte Basis/ festes Werkzeug bearbeiten

**Abb. 4-90: Basis bearbeiten**

1. Die gewünschte Speicherplatznummer (1) und den Namen (2) für die Basis/ festes Werkzeug auswählen/eingeben
Die Nummern der verwendeten Basis-Systeme werden automatisch ausgeblendet. Somit ist ein überschreiben einer vorhanden Basis nicht möglich.
2. Im Pulldown-Fenster (3) festlegen, ob es sich bei dem zu vermessenden Koordinatensystem um eine **Basis** oder ein **feststehendes Werkzeug** handelt.
3. Die bis jetzt eingebenden Daten mittels gleichnamiger Schaltfläche **Speichern** (4).
Info:
Es kann jederzeit der aktuelle Stand gespeichert werden.
4. Sind die Offset-Daten der Basis/festehenden Werkzeugs in Bezug auf das Weltkoordinatensystem bekannt, so können diese direkt numerisch im Feld Transformation eingetragen werden.
5. Sind die Offset-Daten nicht bekannt, so können diese auch messtechnisch ermittelt werden.
Hierzu auf die Schaltfläche **Vermessen** (5) tippen.

Vermessungsmethode

Durch das Tippen auf die Schaltfläche **Vermessen** klappt ein weiteres Kontextmenü auf:

**Abb. 4-91: Basis vermessen, Auswahl**

Die gewünschte Vermessungsmethode antippen.

- **3-Punkt**
(>>> 4.8.1.1 "3-Punkt-Methode" Seite 153)
- **Indirekt**

4.8.1.1 3-Punkt-Methode

Vorgehensweise



Die Basisvermessung kann nur mit einem vorher bereits vermessenen Werkzeug erfolgen (TCP muss bekannt sein).

1. Eine neue Basis wurde über die Werkzeug- und Basisverwaltung hinzugefügt.
(>>> 4.8.1 "Neue Basis/festes Werkzeug hinzufügen" Seite 151)
2. Über die Schaltfläche **Vermessen** die Methode **3-Punkt** auswählen.



Abb. 4-92: Basis vermessen, Auswahl

3. Mit dem TCP den Ursprung der Basis anfahren.

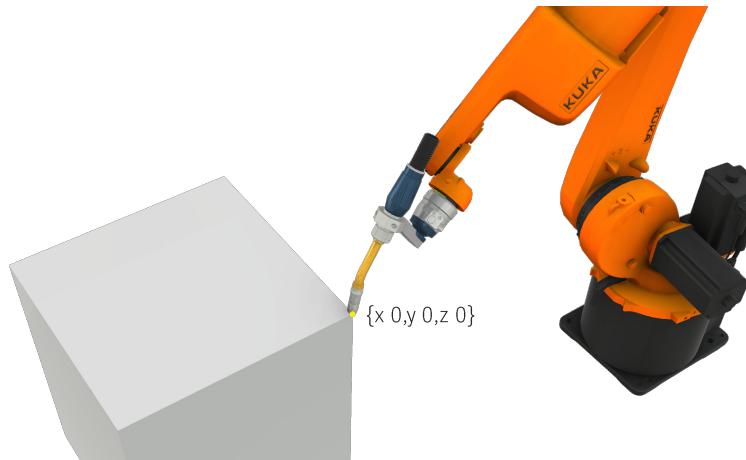


Abb. 4-93: Erster Punkt im Ursprung

4. Touch-Up Ursprung

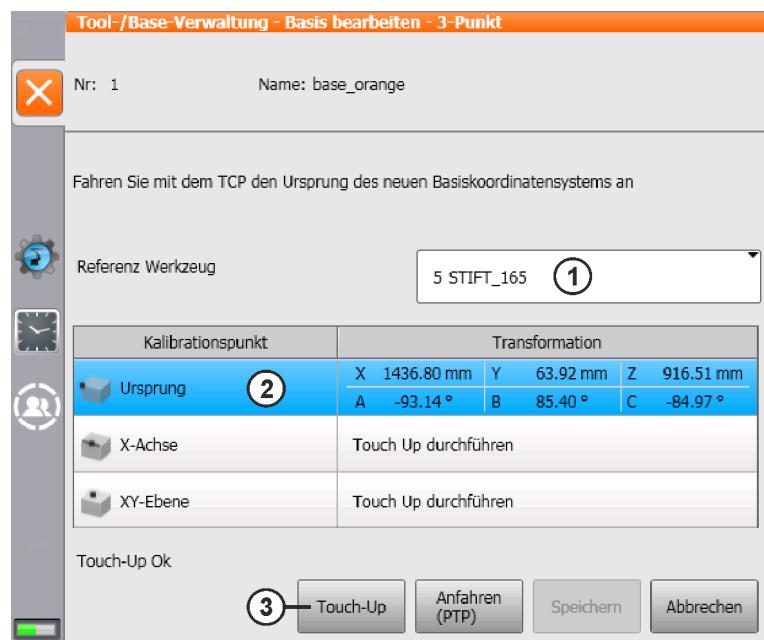


Abb. 4-94: Ursprung

- Das Referenz Werkzeug im Pull-Down-Menü (1) auswählen, das zur Vermessung verwendet wird.
 - Auf die Zeile **Ursprung** (2) tippen.
Die Zeile wird blau markiert.
 - Die Position mittels der Schaltfläche **Touch-Up** (3) aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.
5. Mit dem TCP einen Punkt auf der positiven X-Achse der Basis anfahren.

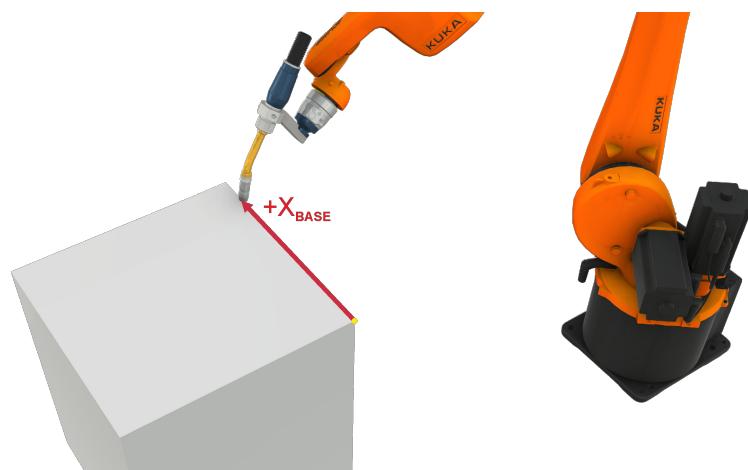


Abb. 4-95: Zweiter Punkt in X-Richtung

6. Touch-Up X-Achse

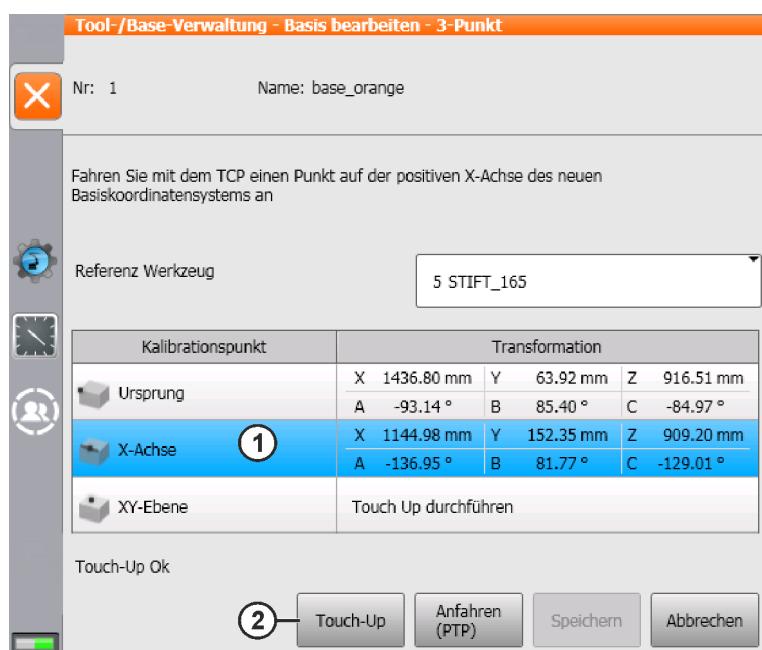


Abb. 4-96: X-Achse

- Auf die Zeile **X-Achse** (2) tippen.
Die Zeile wird blau markiert.
 - Die Position mittels der Schaltfläche **Touch-Up** (3) aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.
7. Mit dem TCP auf der XY-Ebene einen Punkt mit positivem Y-Wert anfahren.

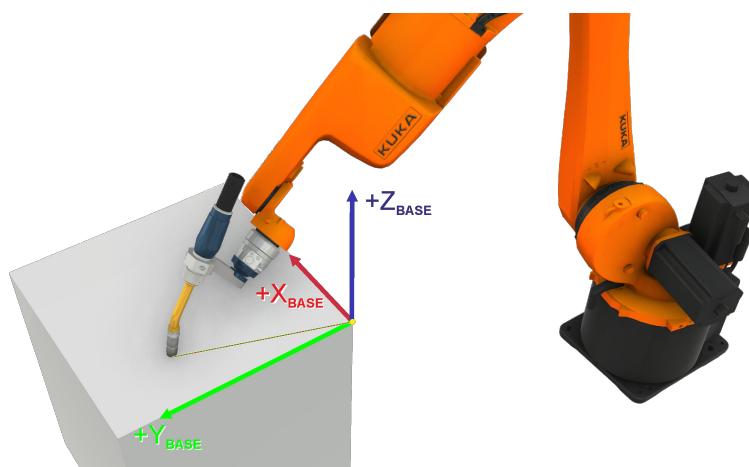


Abb. 4-97: Dritter Punkt auf XY-Ebene

8. Touch-Up XY-Ebene

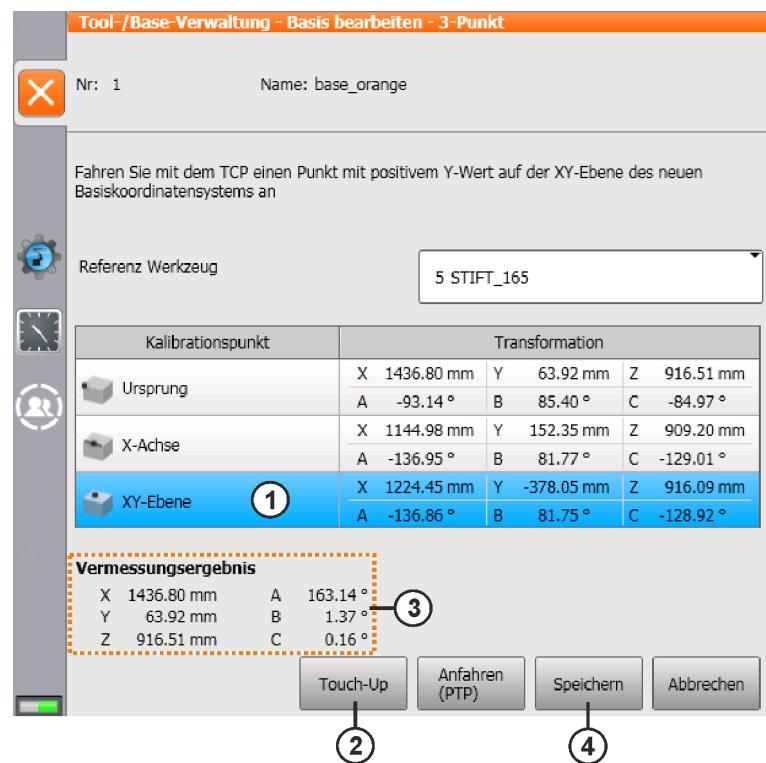


Abb. 4-98: XY-Ebene

- Auf die Zeile **XY-Ebene** (2) tippen.
Die Zeile wird blau markiert.
- Die Position mittels der Schaltfläche **Touch-Up** (2) aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.
- Die Vermessung der Basis ist durch die Aufnahme des letzten Punkts abgeschlossen. Die Offset-Werte werden im Feld **Vermessungsergebnis** (3) angezeigt und zeigen den Abstand und Verdrehung zum Weltkoordinatensystem an.
- Offset-Werte mit der gleichnamigen Schaltfläche **Speichern** (4).

Übung: Basisvermessung Tisch, 3-Punkt-Methode

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Basisvermessung Tisch, 3-Punkt-Methode**

4.8.2 Roboter im Basiskoordinatensystem bewegen

Bewegung im Basiskoordinatensystem

Das Werkzeug des Roboters kann entsprechend der Koordinatenrichtungen des Basiskoordinatensystems bewegt werden. Basiskoordinatensysteme können individuell vermessen sein und orientieren sich oftmals entlang der Werkstückkanten, Werkstückaufnahmen oder Paletten. Dadurch ist ein komfortables Handverfahren möglich. Hierbei bewegen sich **alle** benötigten Roboterachsen. Welche Achsen das sind, wird vom System entschieden und hängt von der Bewegung ab. 32 Basiskoordinatensysteme sind in der Standardkonfiguration auswählbar.

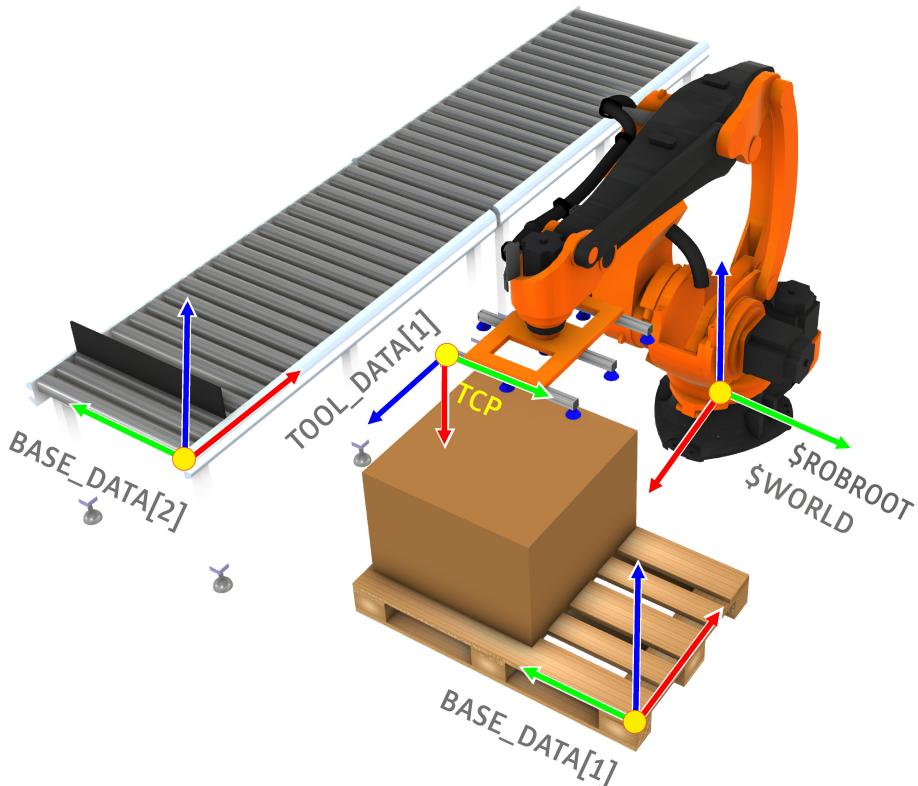


Abb. 4-99: Handverfahren im Basiskoordinatensystem

Vorteile

- Die Bewegung des Roboters ist immer vorhersehbar, sobald das Basiskoordinatensystem bekannt ist.
- Auch hier ist mit der 6D-Maus eine intuitive Bedienung möglich. Voraussetzung ist, dass der Bediener korrekt zum Basiskoordinatensystem steht. Hierzu evtl. die X-Achse als Bezugsachse wählen.
- Ermöglicht das parallele Verfahren, z. B. zu einem Werkstück oder einer Palettenkante.

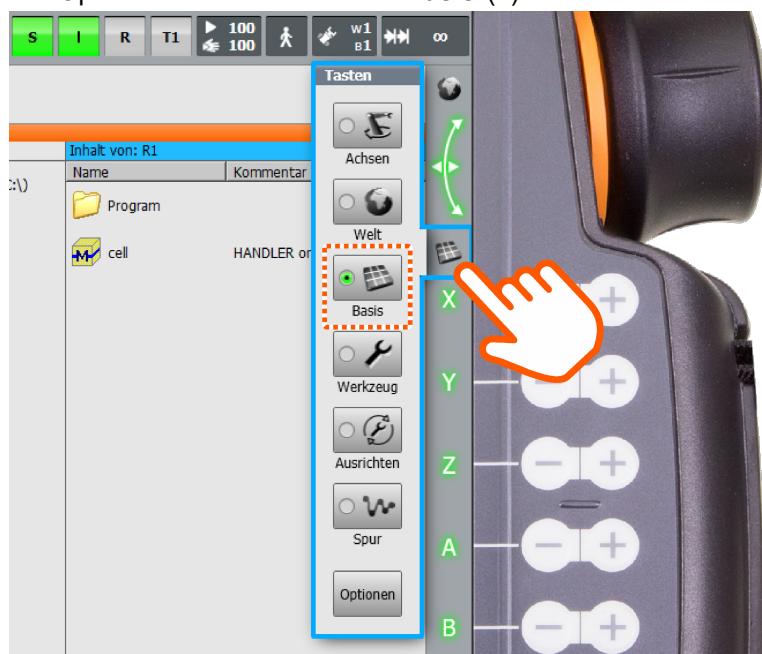
Dabei sind singuläre Stellungen zu vermeiden.



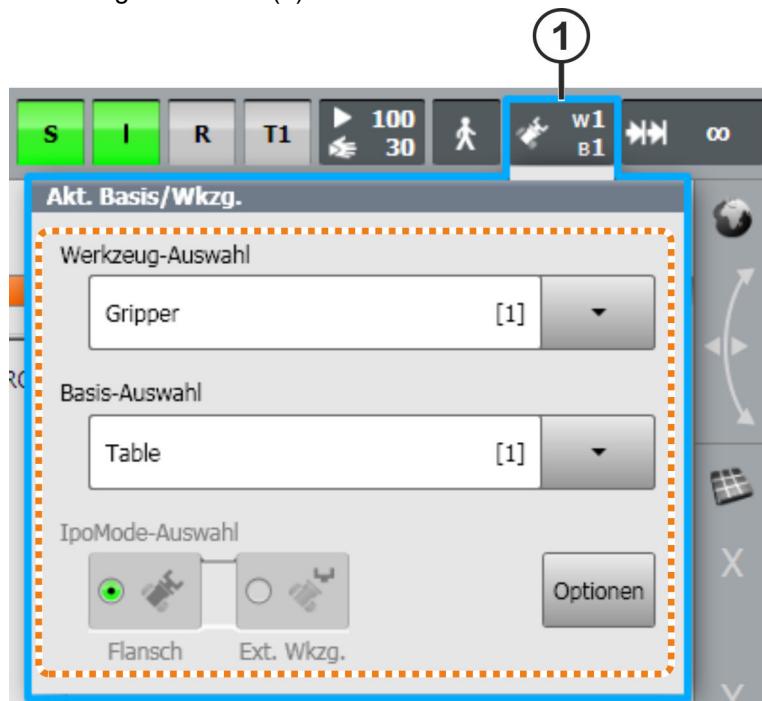
Wird zusätzlich noch das richtige Werkzeugkoordinatensystem eingestellt, kann im Basiskoordinatensystem um den Tool Center Point umorientiert werden.

Vorgehensweise

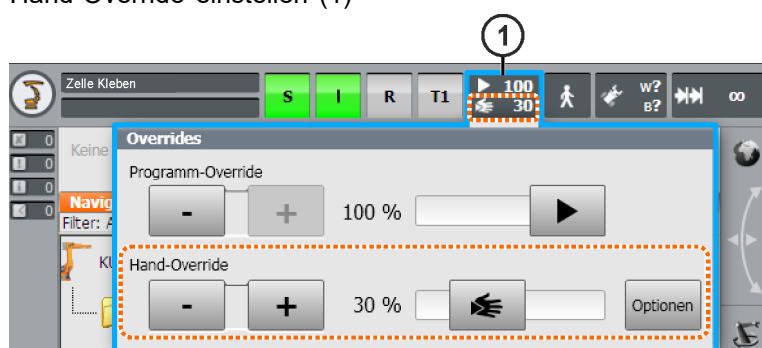
- Als Option für die Verfahrtasten **Basis** (1) auswählen.



- Werkzeug und Basis (1) auswählen



- Hand-Override einstellen (1)



4. Einen der vier Zustimmmtaster auf Mittelstellung drücken und halten.



5. Verfahren mit den Verfahrtasten in die gewünschte Richtung



6. Alternativ kann auch mit der Space Mouse verfahren werden.

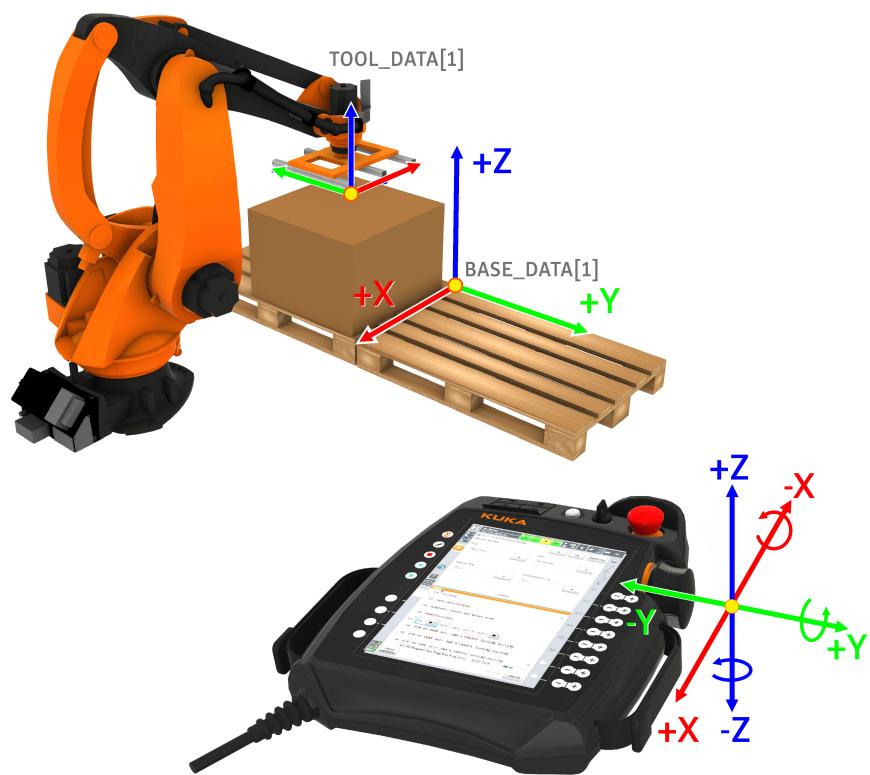


Abb. 4-100: Mausfahren, Basis

4.8.2.1 Übung: Handverfahren im Basiskoordinatensystem

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Handverfahren im Basiskoordinatensystem**

4.9 Roboter mittels Verfahrtart Ausrichten bewegen

Beschreibung

Das Ausrichten erlaubt ein einfacheres und schnelleres Positionieren des Roboters.

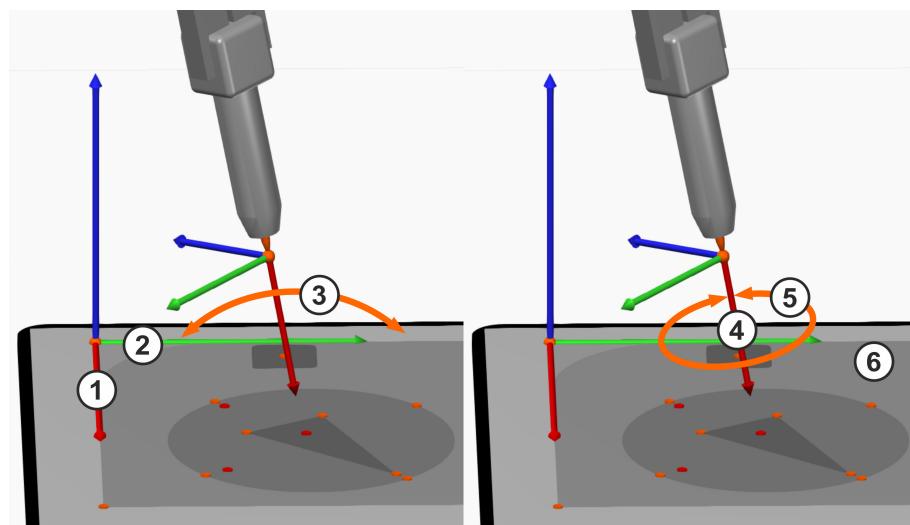


Abb. 4-101: Roboter verfahren: Ausrichten

rot X-Achse

gr Y-Achse

ün

bla Z-Achse

u

1,2 Verfahren in der aktiven Base-Ebene, die senkrecht zur Ausrichtung steht. Im Beispiel X (1) oder Y.

3 Ausrichten senkrecht zur aktiven Basis.

4 Verfahren in Stoßrichtung (+X) des aktiven Werkzeugs.

5 Drehen um die Stoßrichtung des aktiven Werkzeugs.

6 Drehen (Kegelbewegung) um die Ausrichtung zur Base.



Das Verfahren bezieht sich auf das aktive Tool und Base.

Vorgehensweise



Die Verwendung der Verfahrtart Ausrichten ist nur in der Betriebsart T1 möglich.

1. Im Fenster **Handverfahroptionen** in der Registerkarte **Tasten** die Option **Ausrichten** (1) wählen.

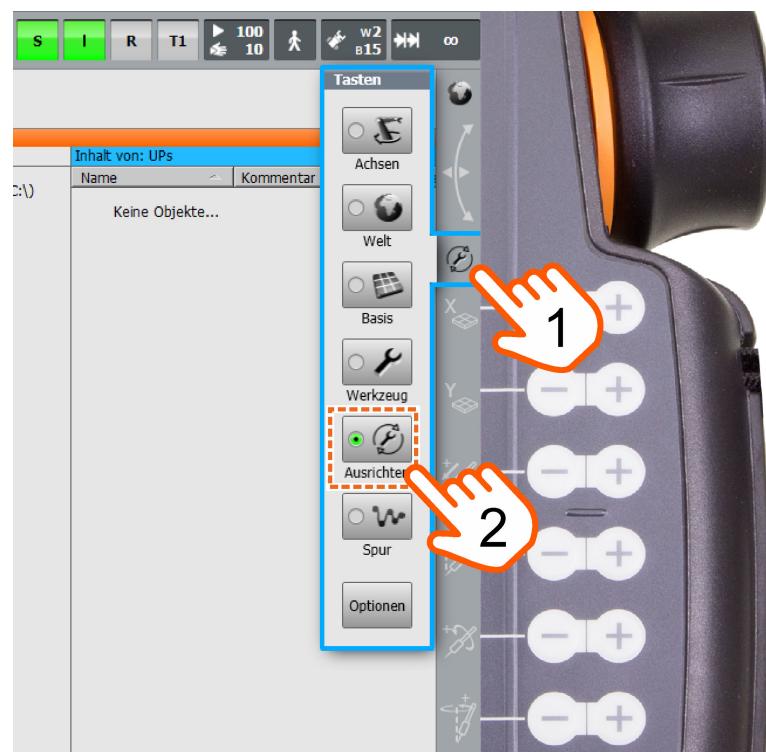


Abb. 4-102: Verfahrart Ausrichten

2. Den Hand-Override einstellen.

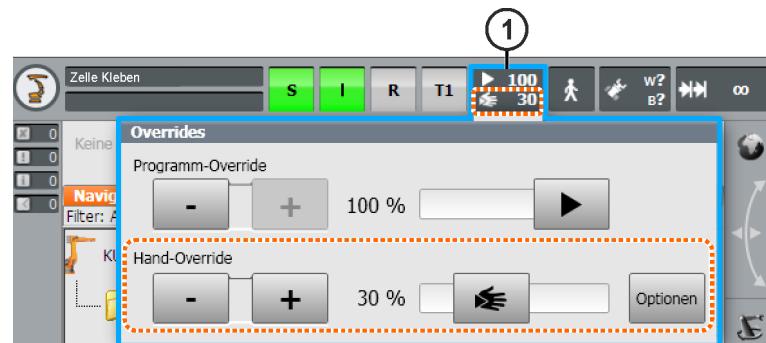


Abb. 4-103: Hand-Override über Statusleiste einstellen

3. Einen der Zustimmungsschalter drücken und halten.



Abb. 4-104: Zustimmungsschalter

4. Mittels der Verfahrtasten ist jetzt folgendes möglich:

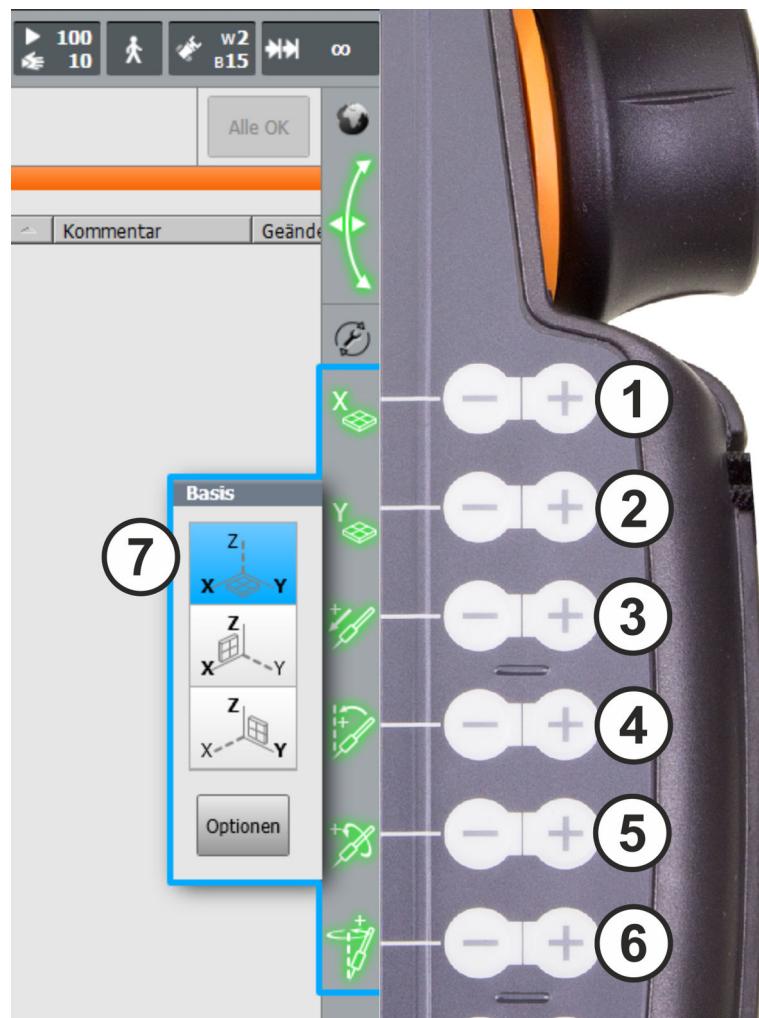


Abb. 4-105: Handverfahren Ausrichten

- Mittels der Verfahrtasten (1) kann in der angezeigten Richtung (z.B. X) im aktuellen Base verfahren werden
 - Mittels der Verfahrtasten (2) kann in der angezeigten Richtung (z. B. Y) im aktuellen Base verfahren werden
 - Mittels der Verfahrtasten (3) kann in Stoßrichtung des aktuellen Tools verfahren werden.
 - Mittels der Verfahrtasten (4) kann das aktuelle Tool senkrecht zur gewählte Baseebene ausgerichtet werden.
 - Mittels der Verfahrtasten (5) kann das aktuelle Tool um den Winkel C gedreht werden .
 - Mittels der Verfahrtasten (6) kann eine Kegelbewegung des aktuellen Tools in Bezug auf die gewählte Baseebene ausgeführt werden.
 - Mittels Basiskonfiguration (7) können die Richtungen für das Handverfahren im Base (1), (2) und die Ebene zum Ausrichten des Tools (4) festgelegt werden .
5. Konfiguration Ausrichten

4.10 Speicherauslastung anzeigen

Beschreibung

Durch Parallelprozesse und Zusatzoptionen steigt der Bedarf an Arbeitsspeicher und Festplattenspeicher auf der Steuerung. Das Plugin bietet einen Überblick über die aktuellen Arbeits- und Festplattenspeicherauslastung.

Vorgehensweise

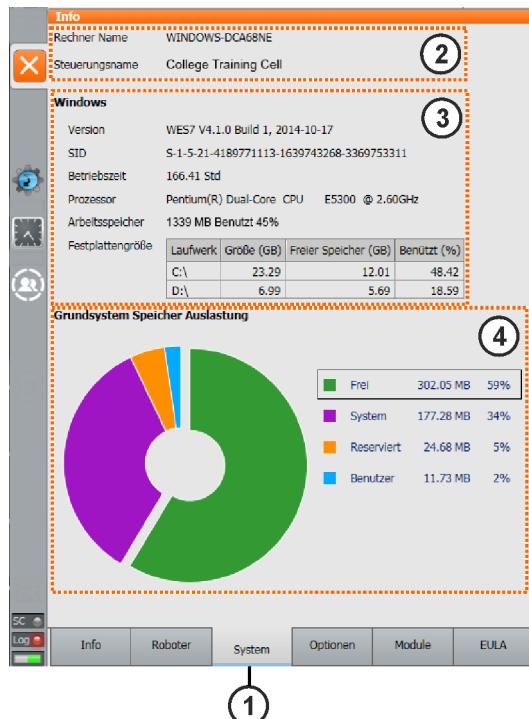


Abb. 4-106: Grundsystem, Speicherauslastung

Menüpfad: Robotertaste > Hilfe > Info > Registerkarte System (1)

- **Rechnername (2)**
Dieser Name wird während der Inbetriebnahme festgelegt und entspricht dem Windowsnamen.
Dateipfad: C:\KUKA\WES7RenameComputer
- **Steuerungsname (2)**
Dieser Name wird während der Inbetriebnahme gesetzt.
Menüpfad: Robotertaste > Inbetriebnahme > Roboterdaten
Der Steuerungsname wird auf der dem EDS der RDC abgelegt.
- **Windows Version (3)**
Versionsinformationen über das installierte Windows-Grundsystem
- **SID - Security Identifier (3)**
Eindeutige Identifikation des Windows-Grundsystems
Die SID wird während der Inbetriebnahme durch den Finalisierungsschritt **Sysprep** generiert.
- **Betriebszeit (3)**
Anzeige der Laufzeit [h], wie lange die Steuerung eingeschaltet eingeschaltet ist.
- **Prozessor (3)**
Genaue Modellbezeichnung des verbauten Prozessors.
- **Arbeitsspeicher (3)**
Anzeige des zur Verfügung stehenden Arbeitsspeicher [MB] und dessen Auslastung in [%], abzüglich des reservierten Bereichs für das Grundsystem.
- **Festplattengröße (3)**
Anzeige der Kapazitätsgröße und -belegung der Partitionen C:\ und D:\.
- **Grundsystem Speicherauslastung (4)**
Übersicht über die Auslastung der reservierten Speicherbereiche [MB] [%] im Grundsystem.

4.11 Steuerung herunterfahren

Beschreibung

Neben dem Hauptschalter kann die Steuerung auch über einen Menüpunkt heruntergefahren oder neu gestartet werden. Diese ermöglicht ein situativ gesteuertes Herunterfahren, das insbesondere bei Konfiguration, Inbetriebnahme oder Fehlersuche notwendig ist.



Abb. 4-107: Power



Der Menüpunkt Herunterfahren ist durch die Rechteverwaltung meist nur bestimmten Benutzergruppe zugänglich.

HINWEIS

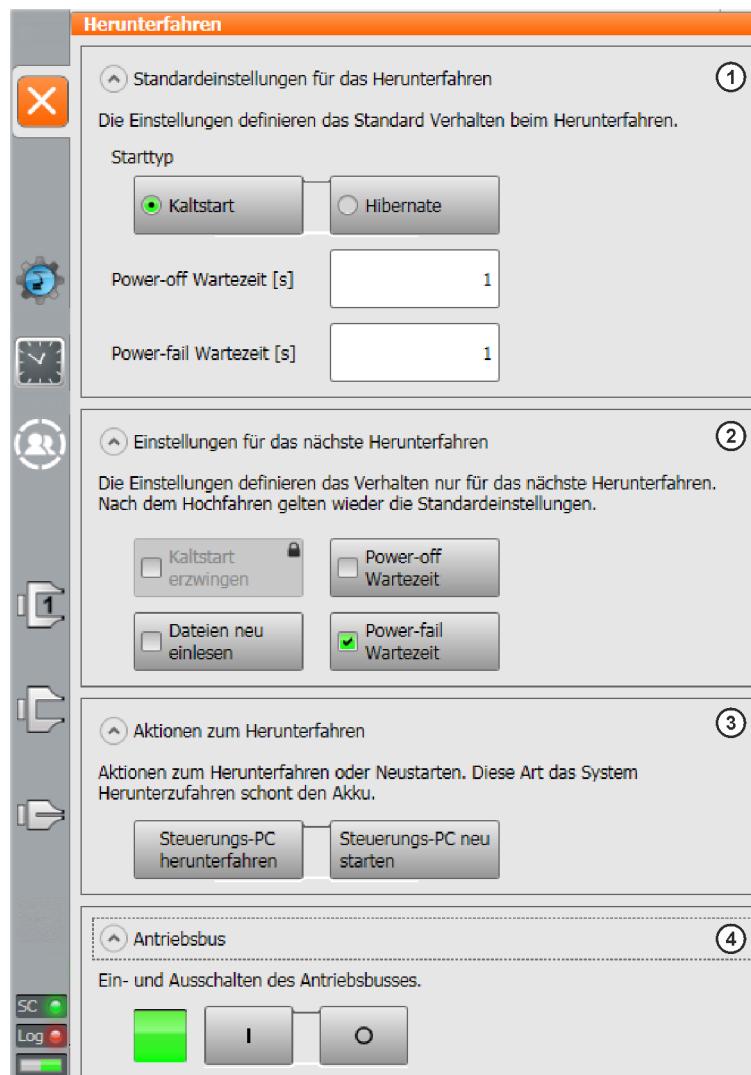
Beschädigung von Systemdateien durch zu frühes Betätigen des Hauptschalters

Wenn die Option **Steuerungs-PC neu starten** gewählt wird und der Hauptschalter betätigt wird, während der Neustart noch nicht abgeschlossen ist, können Systemdateien beschädigt werden.

- Warten, bis der Neustart abgeschlossen ist. Dann erst den Hauptschalter betätigen.

Vorgehensweise

1. **Menüpfad:** *Robotertaste > Herunterfahren*
2. Die gewünschten Optionen wählen.
Es sind gesonderte Benutzerrechte erforderlich.
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.

**Abb. 4-108: Herunterfahren**

Pos.	Beschreibung
1	<p>Standardeinstellung für das Herunterfahren <i>Die Einstellungen definieren das Standard Verhalten beim Herunterfahren.</i></p> <ul style="list-style-type: none"> • 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. Geänderte XML-Dateien werden mit berücksichtigt. • 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 wiederhergestellt. • Power-off Wartezeit [s] Wenn die Steuerung mittels Hauptschalter ausgeschaltet wird, fährt sie erst nach der hier festgelegten Wartezeit herunter. • Power-fail Wartezeit [s] Für Anlagen mit keiner zuverlässigen Netzversorgung Angabe in [s] ohne Spannungsversorgung, in der das Herunterfahren der Steuerung unterdrückt wird.
2	<p>Einstellungen für das nächste Herunterfahren <i>Die Einstellungen definieren das Verhalten nur für das nächste Herunterfahren.</i> <i>Nach dem Hochfahren gelten wieder die Standardeinstellungen.</i></p> <ul style="list-style-type: none"> • Kaltstart erzwingen Die Robotersteuerung fährt herunter und startet dann mit einem Kaltstart neu. • Dateien neu einlesen <ul style="list-style-type: none"> – Wurden XML-Dateien auf der Steuerung, durch z. B. Projektaktivierung, Änderung der Netzwerkkonfiguration etc. verändert so ist diese Einstellung zu aktivieren. In den Konfigurationsmenüs erscheint meist ein entsprechender Hinweis. – Steht nur zur Verfügung, wenn Kaltstart oder Kaltstart erzwingen ausgewählt wurde. • Power-off Wartezeit Aktivierung der unter (1) angegebenen Power-off Wartezeit • Power-fail Wartezeit Aktivierung der unter (1) angegeben Power-fail Wartezeit

Pos.	Beschreibung
3	<p>Aktionen zum Herunterfahren</p> <p><i>Aktionen zum Herunterfahren oder Neustarten. Diese Art das System Herunterzufahren schont den Akku.</i></p> <ul style="list-style-type: none"> • Steuerungs-PC herunterfahren Die Robotersteuerung fährt herunter. Der Hauptschalter befindet sich in der Stellung EIN und die Sleep-LED blinkt am CSP! • Steuerungs-PC neu starten Die Robotersteuerung fährt, ohne Betätigung des Hauptschalters, herunter und startet dann mit einem Kaltstart neu.
4	<p>Antriebsbus</p> <p><i>Ein- Ausschalten des Antriebsbusses</i></p> <p>Ein » Aus</p> <ul style="list-style-type: none"> • grün: Antriebsbus ist an. • rot: Antriebsbus ist aus. • grau: Status des Antriebsbusses ist unbekannt.

5 Roboterprogramme ausführen

5.1 Lerneinheit: Roboterprogramme ausführen

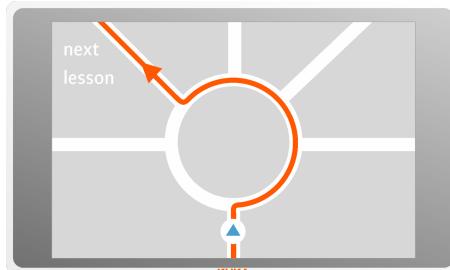


Abb. 5-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Roboterprogramme anwählen
- Wie sieht ein Roboterprogramm aus?
- Initialisierungsfahrt durchführen
- Programmstart durchführen

5.2 Roboterprogramme anwählen

Auswahl und Start von Roboterprogrammen

Wenn ein Roboterprogramm abgearbeitet werden soll, muss es angewählt sein. Die Roboterprogramme stehen auf der Bedienoberfläche im Navigator zur Verfügung. Üblicherweise sind die Verfahrprogramme in Ordnern angelegt. Das Cell-Programm (Verwaltungsprogramm zur Ansteuerung des Roboters von einer SPS) befindet sich immer im Ordner "R1".

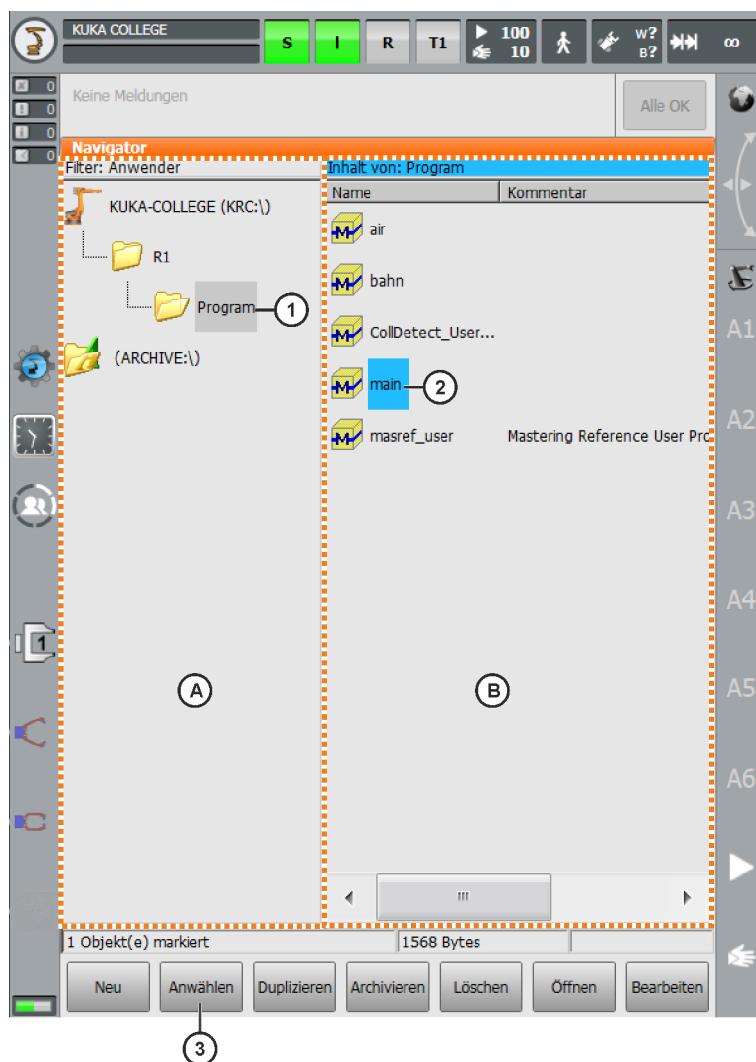


Abb. 5-2: Navigator

- A Navigator: Verzeichnis-/Laufwerksstruktur
- B Navigator: Verzeichnis-/Datenliste
- 1 selektierter Programmordner
- 2 selektiertes Programm
- 3 Schaltfläche zum Anwählen eines Programms

Zum Starten eines Programms stehen sowohl die Start Vorwärts als auch die Start Rückwärts -Tasten zur Verfügung.



- Ein Programm rückwärts mit der -Taste abfahren zu lassen ist nur dann möglich, wenn die INI Zeile eines Programms durchlaufen ist und der Roboter unmittelbar zuvor die zurückzufahrende Bahn vorwärts abgefahren ist.
- Wird der Roboter jedoch nach, z. B. einem Programmstopp manuell verfahren, ist eine Rückwärtsfahrt nicht mehr möglich.
- Ab den Softwareständen KSS 8.5 und höher steht die gesonderte "Handverfahrt" Spur zur Verfügung. (>>> **3.7 "Roboter mittels Verfahrt Spur bewegen" Seite 72**)
Hier bewegt man sich zwar optisch/physikalisch auf der Bahn, hat aber die programmierte Bahn verlassen.



Abb. 5-3: Programmablaufrichtungen: Vorwärts/Rückwärts

Pos.	Beschreibung
1	Start-Taste Vorwärts
2	Start-Taste Rückwärts

Wird ein Programm abgearbeitet, stehen für die programmgesteuerte Bewegung des Roboters mehrere **Programmablaufarten** zur Verfügung:

	GO <ul style="list-style-type: none"> • Programm läuft kontinuierlich bis zum Programmende ab. • Im Testbetrieb muss die Start-Taste gedrückt gehalten werden.
	Bewegung <ul style="list-style-type: none"> • Jeder Bewegungsbefehl einzeln abgearbeitet • Nach Beendigung einer Bewegung muss jeweils erneut "Start" gedrückt werden.
	Einzelschritt Nur in der Benutzergruppe "Experte" verfügbar! <ul style="list-style-type: none"> • Zeile für Zeile wird abgearbeitet (unabhängig vom Inhalt der Zeile). • Nach jeder Zeile muss erneut die Start-Taste betätigt werden.

5.3 Wie sieht ein Roboterprogramm aus?

Beschreibung

```

1 DEF KUKA_Prog()
2
3INI
4 SPTP HOME VEL= 100 % DEFAULT
5 SPTP P1 VEL= 100 % PDAT1 Tool[5] Base[10]
6 SPTP P2 VEL= 100 % PDAT1 Tool[5] Base[10]
7 SLIN P3 VEL= 1 m/s CPDAT1 Tool[5] Base[10]
8 SLIN P4 VEL= 1 m/s CPDAT2 Tool[5] Base[10]
9 SPTP P5 VEL= 100% PDAT1 Tool[5] Base[10]
10 SPTP HOME VEL= 100 % DEFAULT
11
12 END

```

Zeile	Beschreibung
1 und 12	<p>Nur in der Benutzergruppe Experte sichtbar:</p> <ul style="list-style-type: none"> "DEF Programmname()" steht immer am Anfang eines Programmes "END" beschreibt das Ende eines Programms
3	<ul style="list-style-type: none"> Die "INI"-Zeile beinhaltet Aufrufe von Standardparametern, die zur korrekten Abarbeitung des Programms notwendig sind. Die "INI"-Zeile muss immer zuerst abgearbeitet werden!
4 bis 10	<ul style="list-style-type: none"> Eigentlicher Programmtext mit Bewegungsbefehlen, Warte-/Logikbefehlen etc. Der Fahrbefehl "SPTP Home" wird häufig am Anfang und am Ende eines Programms verwendet, da dies eine eindeutige und bekannte Position ist.

5.4 Initialisierungsfahrt durchführen

SAK-Fahrt



Abb. 5-4: Sprinter

Die Initialisierungsfahrt eines KUKA Roboters nennt sich SAK-Fahrt.



SAK steht für **Satzkoinzidenz**. Koinzidenz bedeutet "Übereinstimmung" sowie "Zusammentreffen von zeitlichen/räumlichen Ereignissen".

Eine SAK-Fahrt wird in folgenden Fällen durchgeführt:

- Programm-Anwahl
- Programm-Reset (zurücksetzen)
- Handverfahren während Programmablauf
- Änderung im Programm
- Satzanwahl

Beispiele

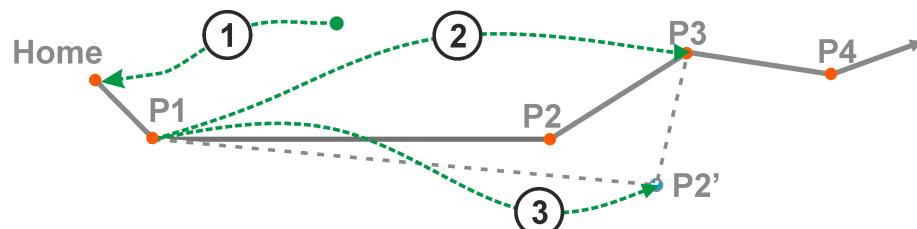


Abb. 5-5: Beispielhafte Gründe für eine SAK-Fahrt

- 1 SAK-Fahrt in die Home-Position nach Programm-Anwahl oder -Reset
Bildbeispiel: Roboter steht außerhalb der Home-Position » Programmstart
- 2 SAK-Fahrt nach Satzanwahl.
Bildbeispiel: Roboter steht im P1 » Satzanwahl auf P3 » Programmstart
- 3 SAK-Fahrt nach Änderung eines Bewegungsbefehls
Bildbeispiel: Roboter steht im P1 » Änderung der Postion P2 auf P2' » Programmstart

Gründe für eine SAK-Fahrt

- Eine SAK-Fahrt ist notwendig, um eine Übereinstimmung der aktuellen Roboterstellung mit den Koordinaten des aktuellen Punkts im Roboterprogramm herzustellen.
- Erst wenn die aktuelle Roboterposition gleich einer programmierten Position ist, kann die Bahnplanung erfolgen. Es muss zunächst also immer der TCP auf die Bahn gebracht werden.

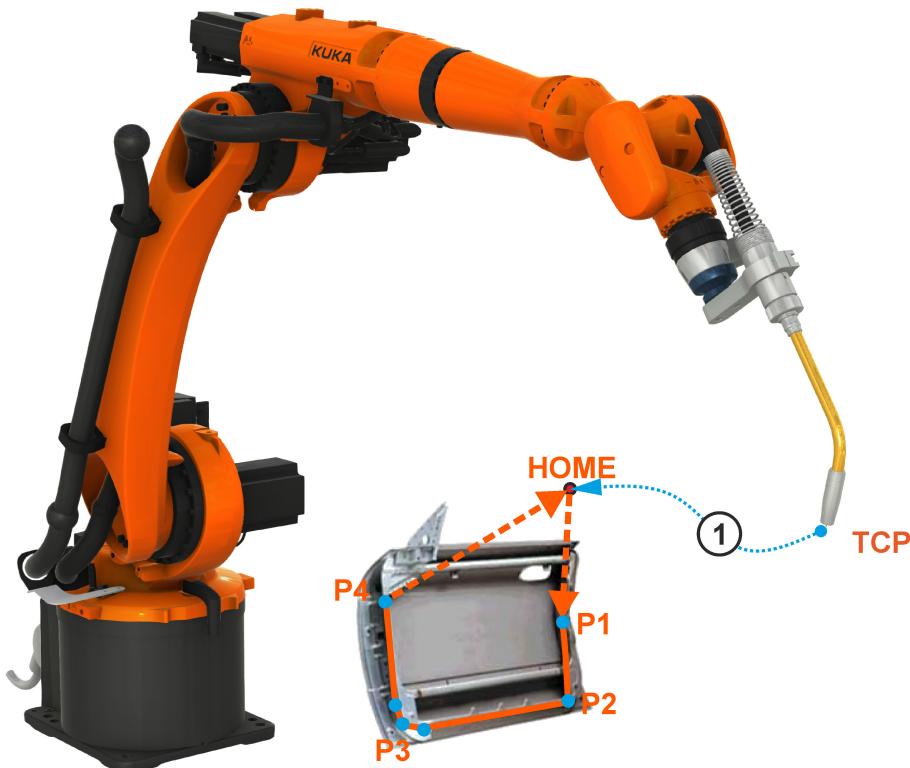


Abb. 5-6: Beispiel für eine SAK-Fahrt bei Programmanwahl

1	SAK-Fahrt in die Grundstellung / HOME-Position nach Programm-Anwahl oder -Reset
KUKA System Software	HOME
Volkswagen System Software	Grundstellung
HINWEIS	
Der Roboter fährt direkt (schnellsten Bahn) von der aktuellen Position in die Grundstellung / HOME-Position. Kollisionsgefahr!	

5.5 Programmstart durchführen

1. Programm anwählen

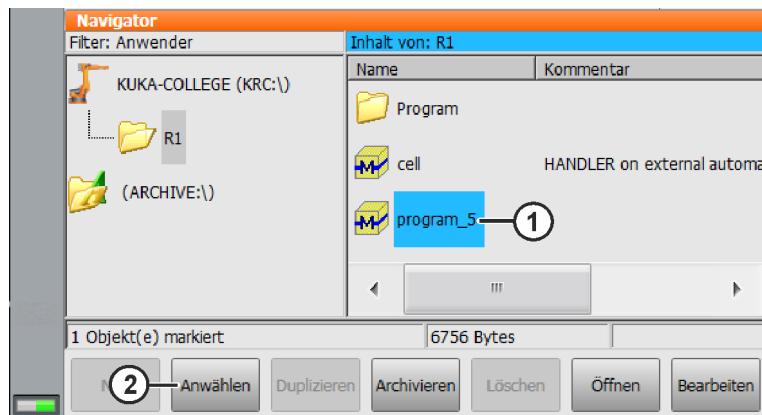


Abb. 5-7: Programmanwahl

2. Programmgeschwindigkeit einstellen (Programm-Override, POV)



Abb. 5-8: POV Einstellung

3. Zustimmtaste betätigen



Abb. 5-9: Zustimmungsschalter

4. Starttaste (+) drücken und gedrückt halten:

- Die "INI"-Zeile wird durchlaufen und das für das Modul notwendigen Koordinatensysteme und Technologiepakete werden initialisiert.
- Der Roboter führt die SAK-Fahrt durch.



Abb. 5-10: Programmablaufrichtungen: Vorwärts/Rückwärts

Pos.	Beschreibung
1	Start Tasten Vorwärts
2	Start Taste Rückwärts



WARNUNG

Eine SAK-Fahrt erfolgt als PTP-Bewegung von der Ist-Position zur Zielposition, wenn der angewählte Bewegungssatz den Fahrbefehl PTP enthält. Enthält der angewählte Bewegungssatz LIN oder CIRC, wird die SAK-Fahrt als LIN-Bewegung ausgeführt. Es ist notwendig die Bewegung zu beobachten, um Kollisionen zu vermeiden. Bei der SAK-Fahrt ist die Geschwindigkeit automatisch reduziert.

5. Nach Erreichen der Zielposition wird die Bewegung angehalten



Abb. 5-11

Die Hinweismeldung "SAK erreicht" wird angezeigt.

6. Weiterer Ablauf (je nach eingestellter Betriebsart):



Abb. 5-12: Antriebe aktivieren

- **T1 und T2:** Programm durch Drücken der Starttaste weiterführen.
- **AUT:** Antriebe aktivieren.
Über die Schaltflächen Antriebe im Fenster Fahrbedingungen die Antriebe aktivieren.

Anschließend Programm mit Impuls auf *Start* starten.

- Im Cell-Programm Betriebsart auf **EXT** umstellen und Fahrbefehl von SPS übertragen.

Programm zurücksetzen und abwählen

Wird auf den Programmstatus getippt öffnet sich ein Kontextmenü, über das ein angewähltes Programm abgewählt oder zurückgesetzt werden kann.



Abb. 5-13: Prgramm zurücksetzen, abwählen

5.5.1 Übung: Roboterprogramme ausführen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- Roboterprogramme ausführen

6 Umgang mit Programmdateien

6.1 Lerneinheit: Umgang mit Programmdateien

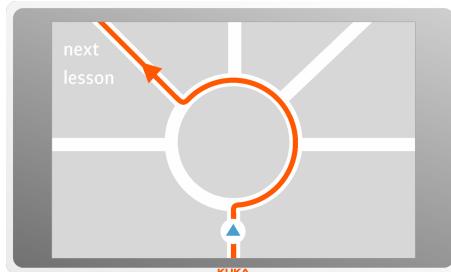


Abb. 6-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Programmmodul erstellen und bearbeiten
- Roboterprogramme archivieren und wiederherstellen
- Programm- und Zustandsänderungen nachvollziehen mittels Logbuch

6.2 Programmmodul erstellen

Programmmodul im Navigator

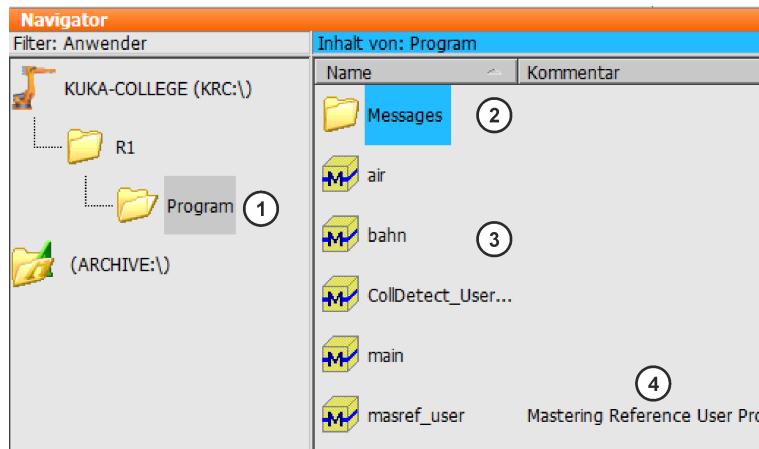


Abb. 6-2: Module im Navigator

- 1 Hauptordner für Programme: "Program"
 - 2 Unterordner
 - 3 Programmmodul/Modul
 - 4 Kommentar eines Programmmoduls
- Programmmodul sollte im Ordner "Program" abgelegt werden.
 - Es besteht auch die Möglichkeit, neue Ordner zu erstellen und Programmmodul dort abzuspeichern.
 - Module sind durch das Symbol mit dem "M" gekennzeichnet.
 - Ein Modul kann mit einem Kommentar versehen werden (z. B. eine kurze Funktionsbeschreibung)

Eigenschaften von Programmmodulen

Ein Modul besteht immer aus zwei Teilen:

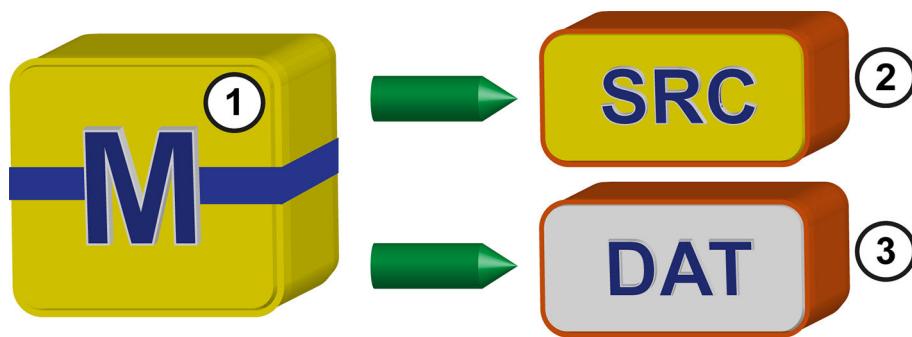


Abb. 6-3: Modul/Folge bzw. Unterprogramm in Anwender- und Expertenansicht

- 1 Modul-Ansicht in Anwenderebene
- 2 Source-Datei: enthält die Programmbefehle
wird auf Anwenderebene beim Anwählen des Moduls angezeigt
- 3 DAT-Datei: enthält die gespeicherten Positionen



Über die Rechteverwaltung sind *.SRC und *.DAT Files standardmäßig nur in der Benutzergruppe Experte bearbeitbar. Der Anwender kann sich in der Regel nur die Detailansicht lesend anzeigen lassen.

- **Quellcode:**

Die *SRC-Datei* enthält den Programmcode.

```
DEF MAINPROGRAM()
INI
SPTP HOME Vel= 100% DEFAULT
SPTP P1 Vel=100% PDAT1 TOOL[1] BASE[2]
SPTP P2 Vel=100% PDAT2 TOOL[1] BASE[2]
...
END
```

- **Datenliste:**

Die *DAT-Datei* enthält permanente Daten und Punktkoordinaten.

```
DEFDAT MAINPROGRAM()
DECL E6POS XP1={X 900, Y 0, Z 800, A 0, B 0, C 0, S 6,
T 27, E1 0, E2 0, E3 0, E4 0, E5 0, E6 0}
DECL FDAT FPOINT1 ...
...
ENDDAT
```

Programmmodule erstellen

1. In der Verzeichnisstruktur den Ordner (1) markieren, in dem das Programm angelegt werden soll, z. B. den Ordner **Programm**.

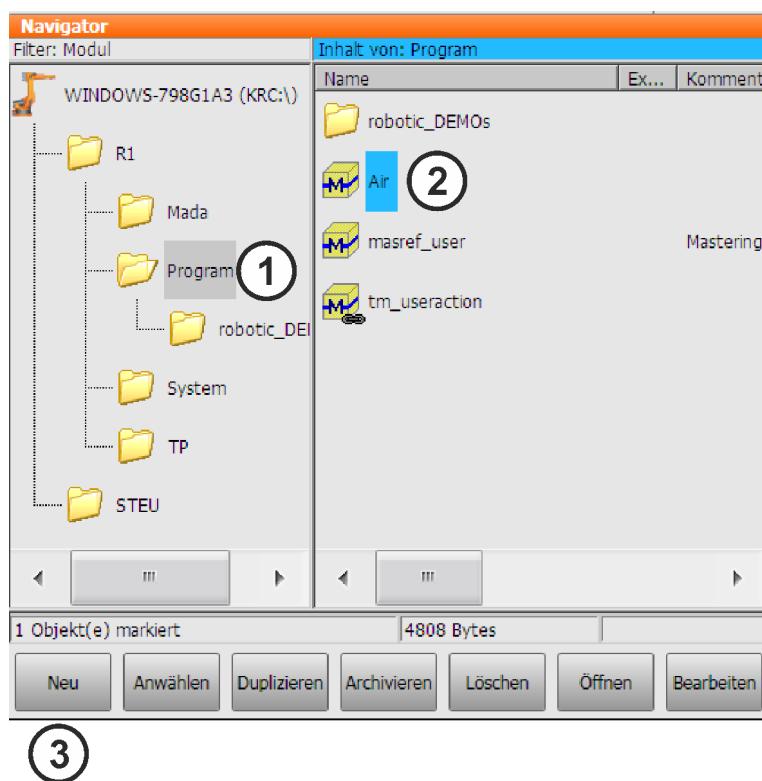


Abb. 6-4: Modul erstellen

Soll innerhalb des Ordners ein Modul erstellt werden, so ist der Fokus in den Ordner (2) zu richten.

2. Mittels Softkey **Neu** (3) wird ein neues Modul erstellt
3. Programmname vergeben

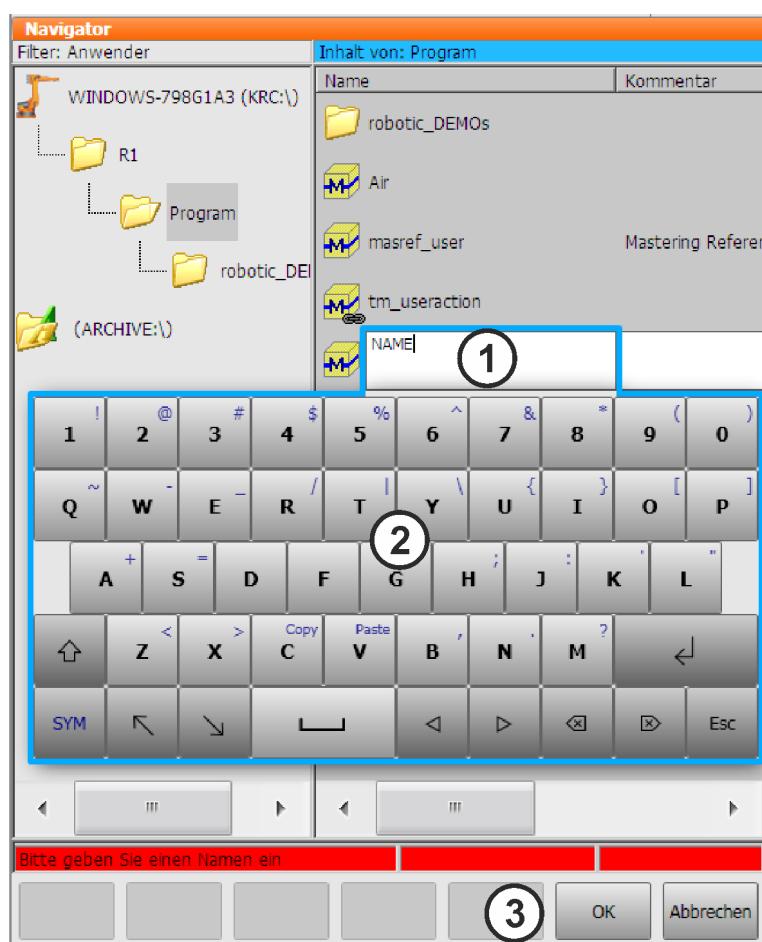


Abb. 6-5: Programmname

Das Eingabefeld für den Namen ist automatisch aktiv (1). Die Eingabe erfolgt mittels eingeblender Tastatur (2). Mit **OK** (3) die Eingabe bestätigen.

6.3 Programmmodul bearbeiten

Bearbeitungsmöglichkeiten

Wie bei gängigen File-Systemen können Programmmoduln auch im Navigator des KUKA smartPAD bearbeitet werden.

Zur Bearbeitung gehören:

- Duplizieren/Kopieren
- Löschen
- Umbenennen

Vorgehensweise

Programm duplizieren

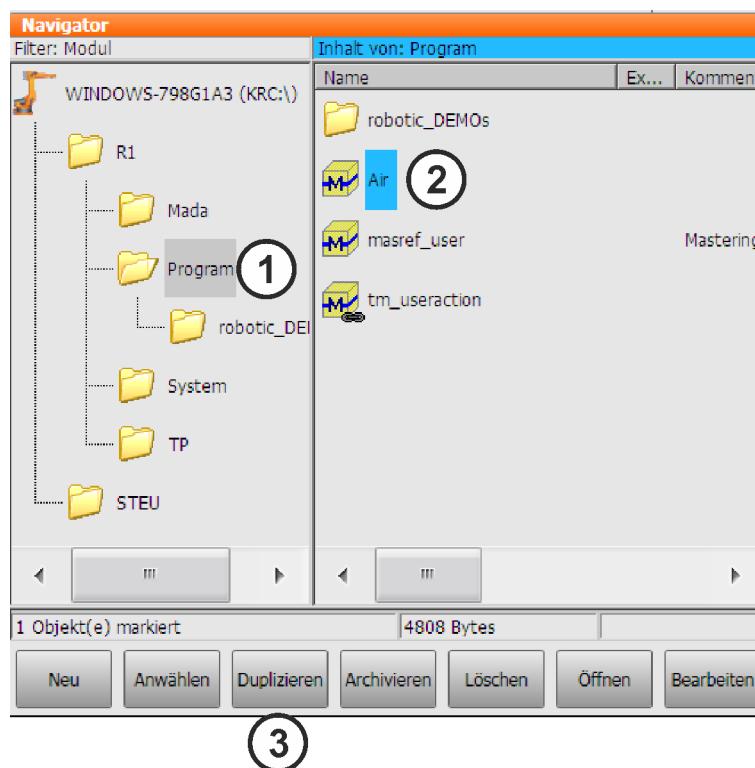


Abb. 6-6: Modul duplizieren

1. In der Verzeichnisstruktur den Ordner (1) markieren, in dem sich die Datei befindet.
2. In der Dateiliste die Datei (2) markieren.
3. Softkey **Duplizieren** (3) wählen.
4. Dem neuen Modul einen neuen Programmnamen zuweisen.
5. Das Modul mit der Schaltfläche **OK** übernehmen.



- In der Benutzergruppe "Experte" und Filtereinstellung "Detail" sind pro Modul zwei Dateien im Navigator abgebildet (SRC und DAT-File).
- Ist das der Fall, müssen beide Dateien dupliziert werden!

Programm löschen

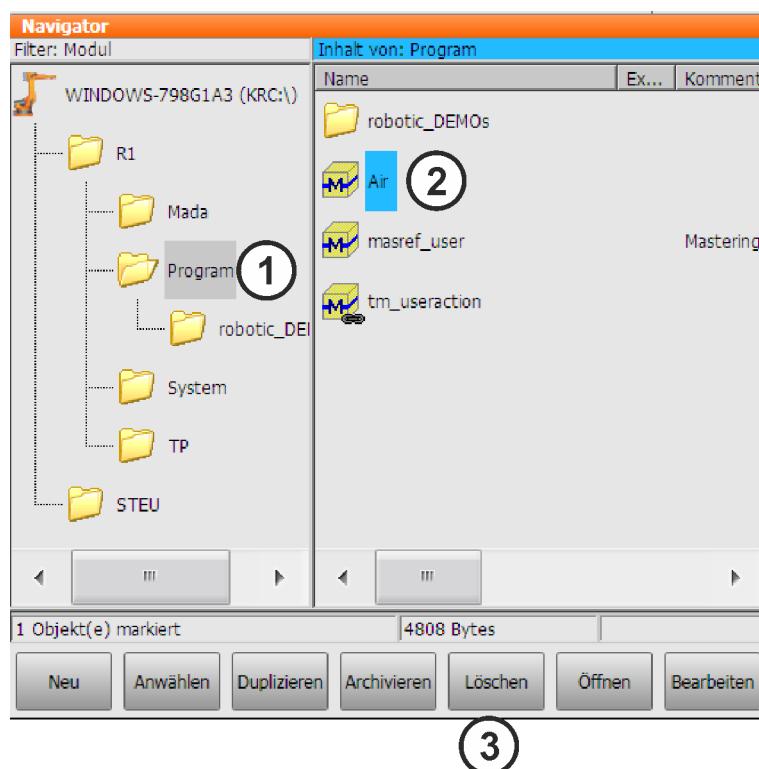


Abb. 6-7: Modul löschen

1. In der Verzeichnisstruktur den Ordner (1) markieren, in dem sich die Datei befindet.
2. In der Dateiliste die Datei (2) markieren.
3. Softkey **Löschen** (3) > wählen.
4. Sicherheitsabfrage mit **Ja** bestätigen. Das Modul wird gelöscht.



- In der Standardeinstellung der Rechteverwaltung sind für die Benutzergruppe "Experte" die Filtereinstellung "Detail" sind pro Modul zwei Dateien im Navigator abgebildet (SRC und DAT-File).
- Ist das der Fall, müssen beide Dateien gelöscht werden!
- **Gelöschte Dateien sind nicht wiederherstellbar!**

Programm umbenennen

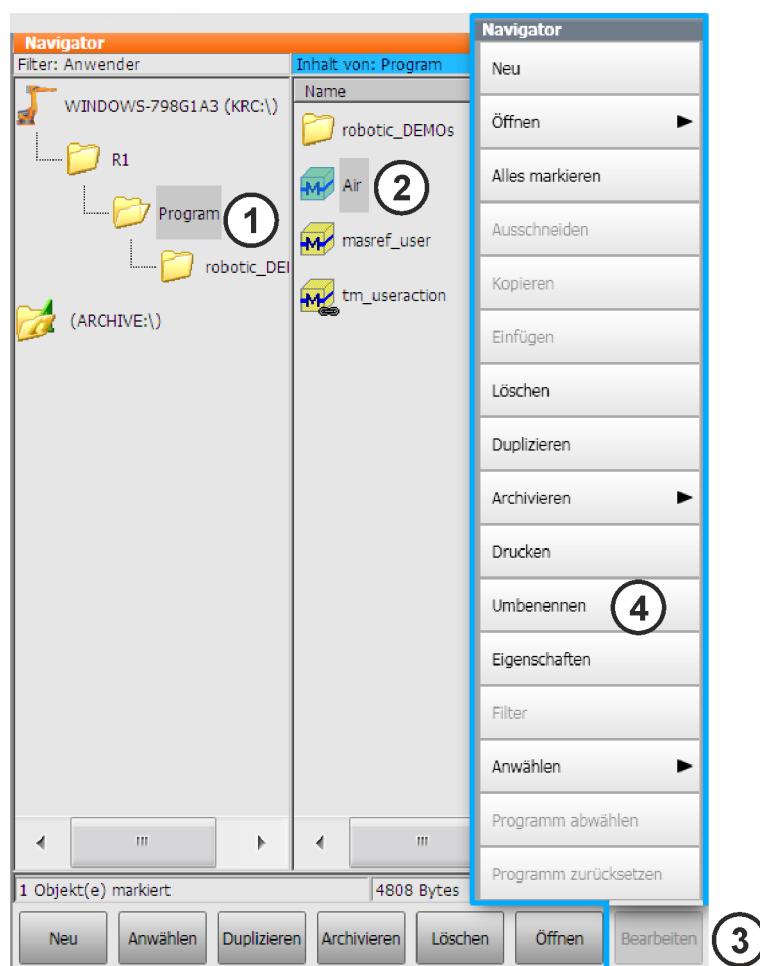


Abb. 6-8: Modul umbenennen

1. In der Verzeichnisstruktur den Ordner markieren, in dem sich die Datei (1) befindet.
2. In der Dateiliste die Datei (2) markieren.
3. Softkey **Bearbeiten** (3) > **Umbenennen** (4) wählen.
4. Den Dateinamen mit dem neuen Namen überschreiben und mit **OK** bestätigen.



- In der Benutzergruppe "Experte" und Filtereinstellung "Detail" sind pro Modul zwei Dateien im Navigator abgebildet (SRC und DAT-File).
- Ist das der Fall, müssen beide Dateien umbenannt werden!

(>>> [12.3 "Ansichten im Explorer anpassen" Seite 357](#))

6.4 Roboterprogramme archivieren und wiederherstellen

Beschreibung

Gelöschte Dateien sind nicht wiederherstellbar!

Der Archivierungsvorgang erzeugt eine ZIP-Datei auf dem entsprechenden Zielmedium, die den gleichen Namen hat wie der Roboter. Der Robotername kann im Menüpunkt **Roboterdaten** individuell angepasst werden.

Speicherorte



Abb. 6-9: KUKA USB-Stick

Es stehen drei unterschiedliche USB-Speicherorte zur Verfügung:

- **USB (KCP)**
 - USB-Stick am KCP(smartPAD)
- **USB (Schrank)**
 - USB-Stick am Roboter-Steuerschrank
- **Netzwerk**
 - Archivierung auf einem Netzwerkpfad
 - Der gewünschte Netzwerkpfad muss unter **Roboterdaten** konfiguriert werden.



Bei dem Archivierungsvorgang wird zur erzeugten ZIP-Datei auf dem gewählten Speichermedium parallel auf dem Laufwerk D:\ eine weitere Archivierungsdatei (BackupAll.ZIP) abgelegt.

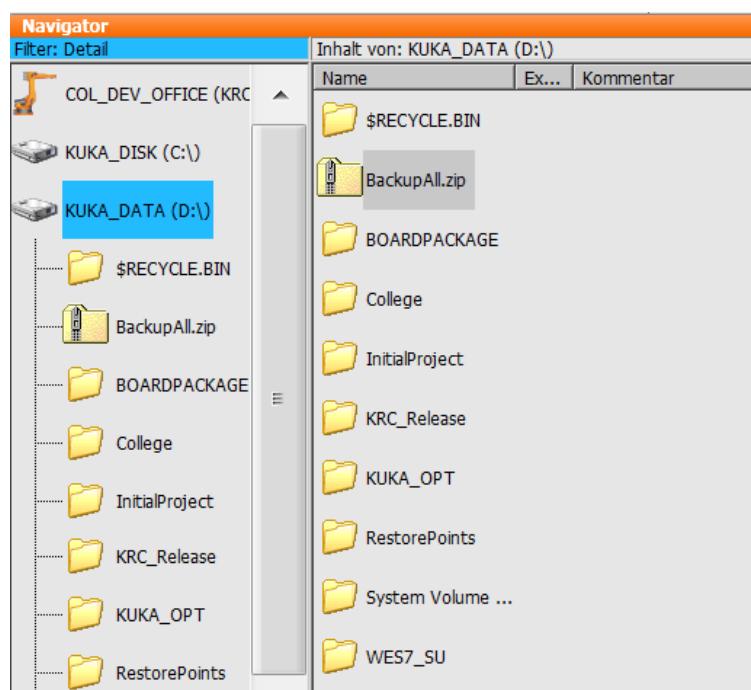


Abb. 6-10: Archiv

HINWEIS

Datenverlust durch USB-Sticks von Fremdherstellern

Wenn bei Tätigkeiten an der Robotersteuerung USB-Sticks von anderen Herstellern als KUKA verwendet werden, kann dies zu Datenverlust führen.

- Für Tätigkeiten an der Robotersteuerung, die einen USB-Stick erfordern, einen KUKA-Stick verwenden.
- Die KUKA-Sticks sind für den Einsatz an der Robotersteuerung validiert.

Folgende Datenauswahl zur Archivierung kann getroffen werden:

- **Alles:**
Die Daten, die notwendig sind, um ein bestehendes System wiederherzustellen zu können, werden archiviert.
- **Anwendungen:**
Alle benutzerdefinierten KRL-Module (Programme) und die zugehörigen Systemdateien werden archiviert.
- **Systemdaten:**
Die Maschinendaten werden archiviert.
- **Log Daten:**
Die LOG-Dateien werden archiviert.
- **KRCdiag:**
Archivierung von Daten, um sie zur Fehleranalyse der KUKA-Hotline zukommen zu lassen. Es wird eine ZIP-Archiv erzeugt, welches bequem per E-Mail verschickt werden kann. Dieses findet sich auch unter C:\KUKA\KRCdiag wieder.

Vorgehensweise

Daten Archivieren

1. Den USB-Stick in die ausgewählte USB-Schnittstelle stecken.



Abb. 6-11: KUKA USB-Stick

2. **Menüpfad:** Roboter-Taste > Datei > Archivieren > USB (Schrank) > gewünschter Unterpunkt

Hauptmenü	Datei	Archivieren
Datei ►	Archivieren ►	USB (KCP) ►
Konfiguration ►	Wiederherstellen ►	USB (Schrank) ►
Anzeige ►	Backup-Manager ►	Netzwerk ►
Diagnose ►		Logbuch

Abb. 6-12: Archivieren, Menüpfad

3. Auswählen, welche Inhalte in das Archiv sollen.



Abb. 6-13: Auswahl

4. Sicherheitsabfrage mit **Ja** bestätigen.
Das Meldungsfenster zeigt eine Meldung an, wenn die Archivierung abgeschlossen ist.
5. Wenn die LED am Stick nicht mehr leuchtet, kann dieser abgezogen werden.



Die Archivierung über die Variante USB (Schrank) ist zu bevorzugen. Hier kann der KCP-KUKA PC direkt über den Mainboard-Port auf den USB-Stick schreiben. Wird hingegen die Schnittstelle am smartPAD verwendet, müssen die Daten vom KPC über das Feldbusssystem EtherCAT an das smartPAD übertragen werden. Dieser Umweg dauert wesentlich länger.

Daten wiederherstellen



WARNUNG

Generell dürfen nur Archive mit der passenden Software-Version geladen werden. Wenn andere Archive geladen werden, können als Folgen auftreten:

- Fehlermeldungen
- Robotersteuerung ist nicht lauffähig
- Personen- und Sachschäden

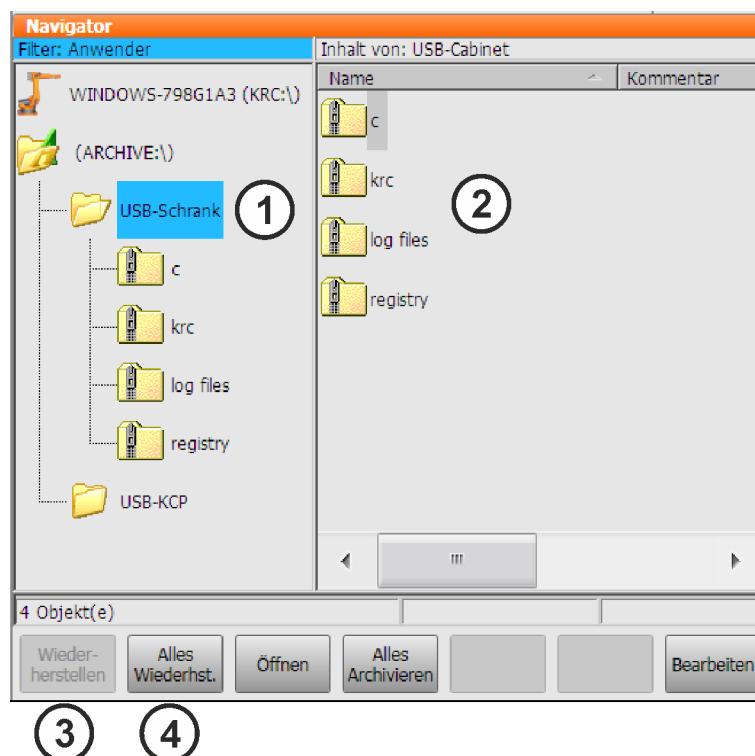


Abb. 6-14: Archiv wiederherstellen

1. Den USB-Stick mit der Sicherung in die Steuerung stecken.
Alternativ wird auch ein Netzwerkarchivpfad verwendet werden.
2. Auf dem smartPAD im Navigator das zu wiederherstellende Archiv (1) öffnen.
3. Es können optional auch Einzelemente (2) innerhalb des Archives markiert werden.

4. Einzelselektionen über die Schaltfläche mit der gleichnamigen Schaltfläche **Wiederherstellen** (3).
folgende Menüpunkte zur Auswahl:
 - **Alles**
 - **Anwendungen**
 - **Systemdaten**
5. Sicherheitsabfrage mit **Ja** bestätigen.
Die archivierten Dateien werden auf der Robotersteuerung wiederhergestellt.
6. Soll der komplette Inhalt wiederhergestellt werden die Schaltfläche **Alles Wiederherstellen** (4) verwenden.
7. Erst wenn die LED am USB-Medium nicht mehr leuchtet, kann das Medium entfernt werden.
Andernfalls kann das Medium beschädigt werden.



In folgenden Fällen gibt das System eine Fehlermeldung aus:

- Wenn die archivierten Daten eine andere Version haben als die auf dem System befindlichen.
- Wenn die Version der Technologiepakete nicht mit der installierten Version übereinstimmt.

6.5 Programm- und Zustandsänderungen nachvollziehen mittels Logbuch

Beschreibung

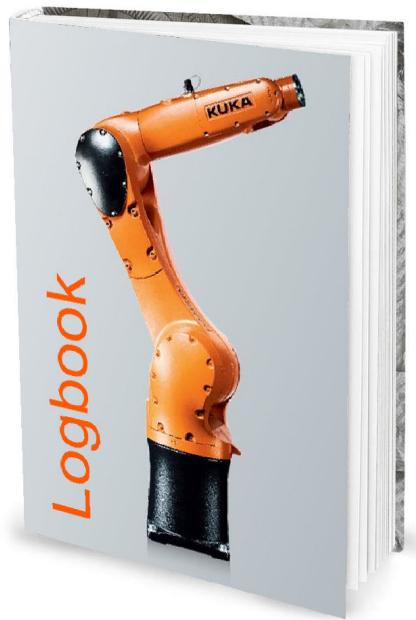


Abb. 6-15: Logbuch

- Alle Handlungen an der Steuerung, Fehlermeldungen und Zustände werden in einem Logbuch protokolliert.
- Jedes Ereignis erhält einen Zeitstempel und eine laufende Nummer.
- Das Logbuch kann nicht gelöscht werden.

Vorgehensweise

1. **Menüpfad:** KUKA Taste > Diagnose > Logbuch > Anzeigen



Abb. 6-16: Logbuch anzeigen und konfigurieren

2. Das Logbuch wird auf dem smartPAD angezeigt.



Abb. 6-17: Logbuch, Registerkarte Log

Pos.	Beschreibung
1	Art des Log-Ereignisses 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

Filterung von Log-Ereignissen

- Welche Einträge im Logbuch zur Anzeige gebracht werden, kann im Reiter Filter eingestellt werden.
- Hierzu die Einträge, die von Relevanz sind, mittels Maus markieren.

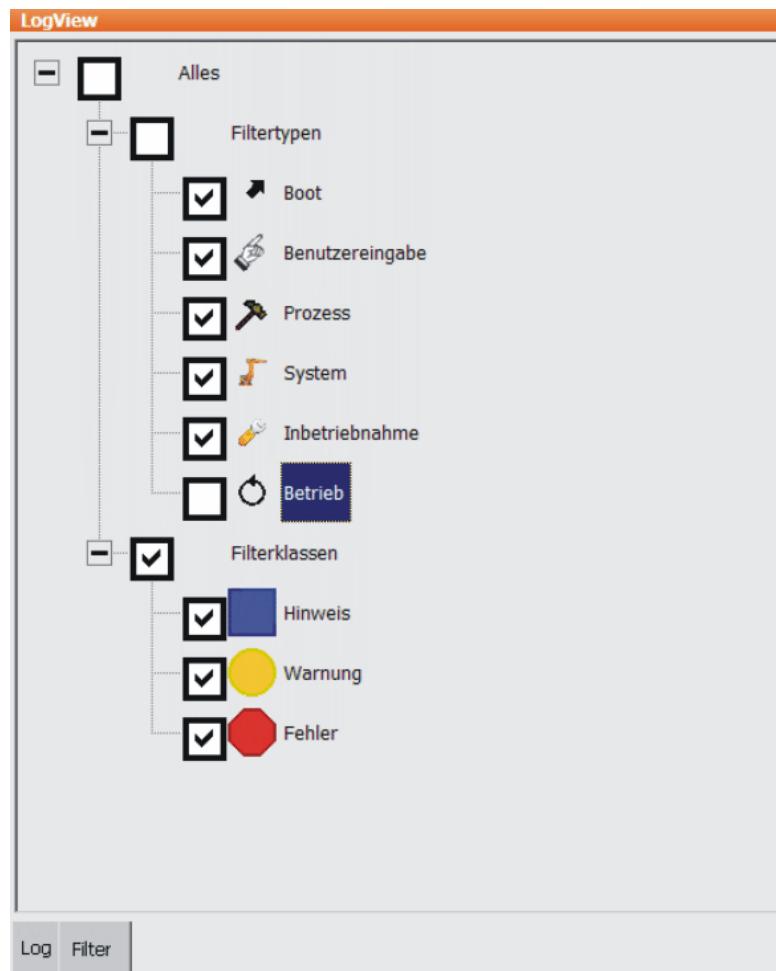


Abb. 6-18: Logbuch, Registerkarte Filter

6.5.1 Logbuch exportieren

Beschreibung

- Der aktuelle Stand des Logbuches kann in eine Textdatei extrahiert werden.
- Die Exportfunktion hat keinen Einfluss auf den originären Logbuchinhalt der Steuerung.

Vorgehensweise

- Es ist mindestens die Benutzergruppe **Anwender** erforderlich.
- In der Standardeinstellung ist für die Exportfunktion mindestens die Benutzergruppe **Experte** erforderlich.
- Menüpfad:** *Diagnose > Logbuch > Konfiguration*
- Die Exportparameter konfigurieren.

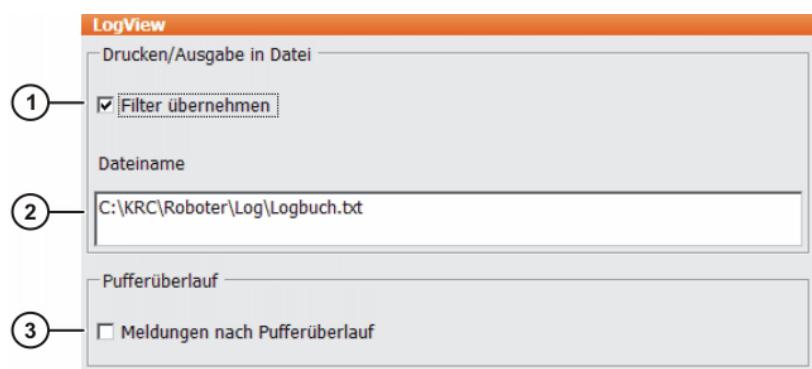


Abb. 6-19: Fenster Konfiguration Logbuch

1	Filtereinstellungen für Ausgabe übernehmen. Wenn kein Haken gesetzt ist, wird bei der Ausgabe nicht gefiltert.
2	Pfad der Textdatei.
3	Log-Daten, die wegen Pufferüberlauf gelöscht wurden, werden in der Textdatei grau hinterlegt angezeigt.

5. Mittels der Schaltfläche **OK** die Konfiguration speichern und das Fenster schließen.
6. In die Loganzeige wechseln
Menüpfad: *KUKA Taste > Diagnose > Logbuch > Anzeigen*
7. Mittels der gleichnamigen Schaltfläche (1) das Logbuch **Exportieren**



Abb. 6-20: Logbuch exportieren

7 Programmierte Bewegungen erstellen und ändern

7.1 Lerneinheit: Programmierte Bewegungen erstellen und ändern

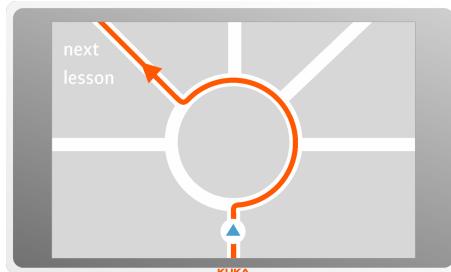


Abb. 7-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Erstellung neuer Bewegungsbefehle
- Taktzeitoptimierte Bewegung erstellen
- Bahnbewegung erstellen
- Mit globalen Punkten programmieren
- Ändern von Bewegungsbefehlen

7.2 Welche Informationen werden für die Erstellung neuer Bewegungsbefehle benötigt?

Programmierung einer Roboterbewegung

Wenn Roboterbewegungen programmiert werden müssen, stellen sich viele Fragen:

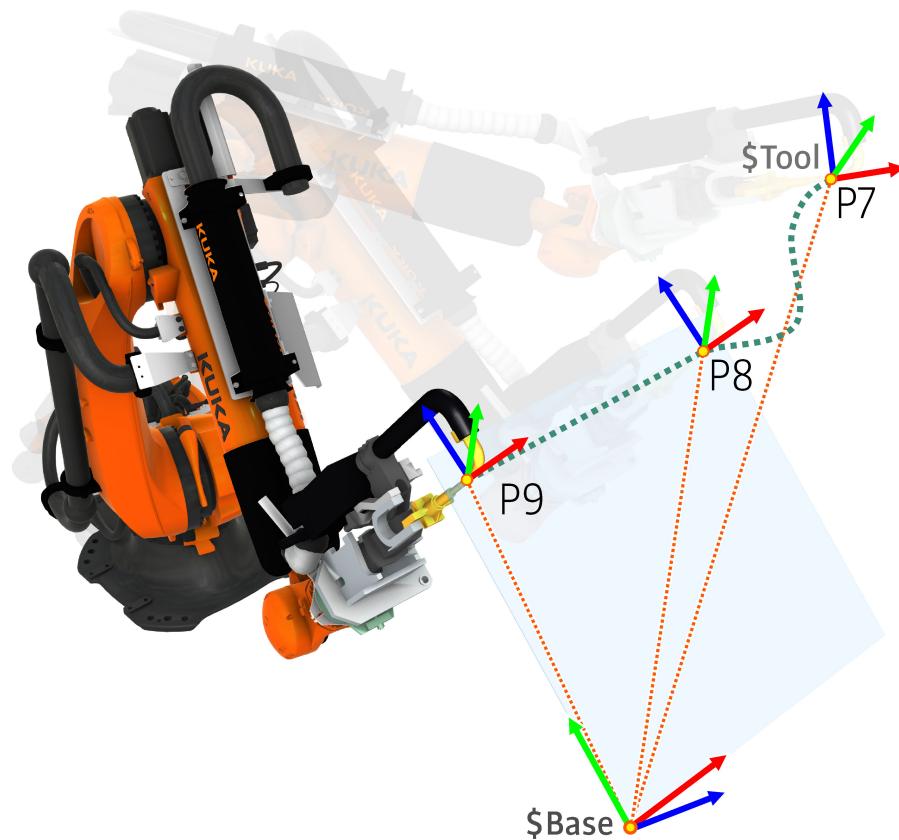


Abb. 7-2: Roboterbewegung

Frage	Lösung	Schlüsselwort
Wie merkt sich der Roboter seine Stellungen?	Die jeweilige Position des Werkzeugs im Raum wird abgespeichert. (Roboterposition entsprechend eingestelltem Tool und Base)	POS E6POS
Woher weiß der Roboter wie er sich bewegen muss?	Über die Angabe der Bewegungsart: Punkt zu Punkt, linear oder kreisförmig.	SPTP SLIN SCIRC
Wie schnell ist der Roboter bei seinen Bewegungen?	Die Angabe der Geschwindigkeit zwischen zwei Punkten und die Beschleunigung erfolgt beim Programmieren.	Vel. Acc.
Muss der Roboter in jedem Punkt stehen bleiben?	Um Taktzeit zu sparen können Punkte auch überschritten werden, es erfolgt dann kein Genauhalt.	CONT
Welche Orientierung nimmt das Werkzeug ein, wenn ein Punkt erreicht wird?	Für jede Bewegung kann die Orientierungsführung individuell eingestellt werden. Diese Einstellung gilt nur für Bahnbewegungen. (>>> "Bewegungsarten" Seite 197)	ORI_TYPE
Erkennt der Roboter ein Hindernis?	Nein, der Roboter folgt "stur" seiner programmierten Bahn. Kollisionsfreiheit liegt in der Verantwortung des Programmierers. Es gibt jedoch die Möglichkeit der "Kollisionsüberwachung" zum Maschinenschutz.	Kollisions-überwachung

Bewegungsarten

- Für die Programmierung von Bewegungsbefehlen stehen verschiedene Bewegungsarten zur Verfügung.
- Je nach Anforderung an den Arbeitsprozess des Roboters können die Bewegungen programmiert werden.
 - Achsspezifische Bewegungen**
SPTP » Point to Point
 - Bahnbewegungen:**
SLIN » Linear
SCIRC » Zirkular



PTP, LIN und CIRC-Bewegungen sind nicht Gegenstand dieser Schulungsunterlage. Nähere Informationen können sie der *Bedien- und Programmieranleitung KUKA System Software 8.x im Xpert* entnehmen.

Programmierung mittels Inlineformularen



Abb. 7-3: Inlineformular

- Beim Programmieren von Roboterbewegungen im Teach-In-Verfahren müssen die genannten punkt- und bewegungsspezifischen Informationen übermittelt werden.
- Dazu werden Inline-Formulare verwendet, in denen diese Informationen benutzerfreundlich eingegeben werden können.

7.3 SPTP - Taktzeitoptimierte Bewegungen (Achsbewegung)

Beschreibung

SPTP - *Spline Point To Point* ist eine achsspezifische Bewegung. Der Roboter führt den TCP des Werkzeugs entlang der **schnellsten** Bewegung 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. Die erste Bewegung im Programm muss eine SPTP-Bewegung sein, da nur hier Status und Turn ([7.3.1 "Status und Turn" Seite 199](#)) ausgewertet werden. Der exakte Verlauf der Bewegung ist nach dem Teachen der Punkte nicht vorhersehbar. Einmal abgefahren, wird der Roboter die Bewegung jedoch immer wieder identisch wiederholen.

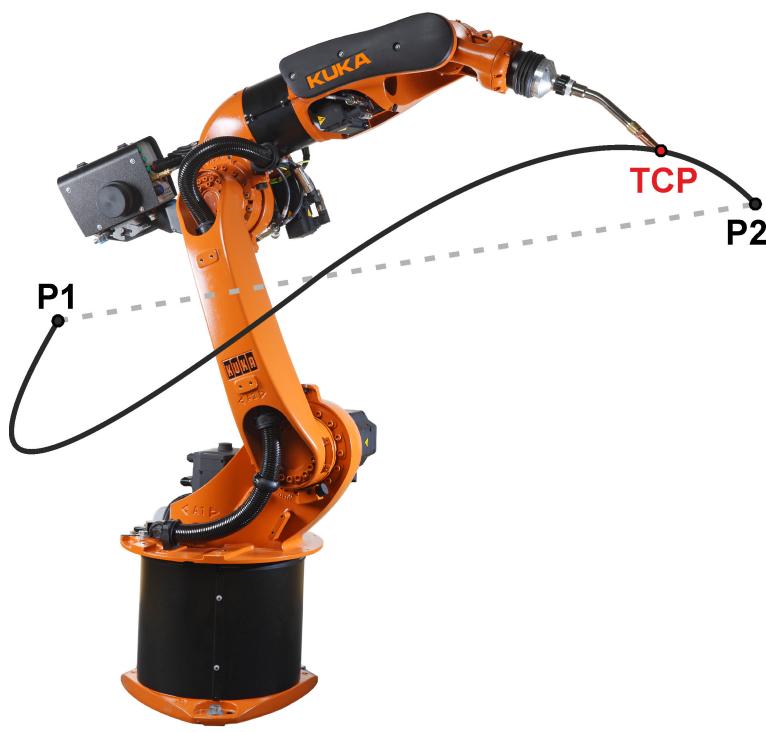


Abb. 7-4: SPTP-Bewegung

Synchro-PTP

Alle Achsen starten bei Synchro-PTP gemeinsam und kommen auch gleichzeitig zum Stehen. Dabei wird die Geschwindigkeitsangabe im Inline-formular mit berücksichtigt. Als führende Achse bezeichnet man die Achse, die am **längsten** braucht, um den Zielpunkt zu erreichen.

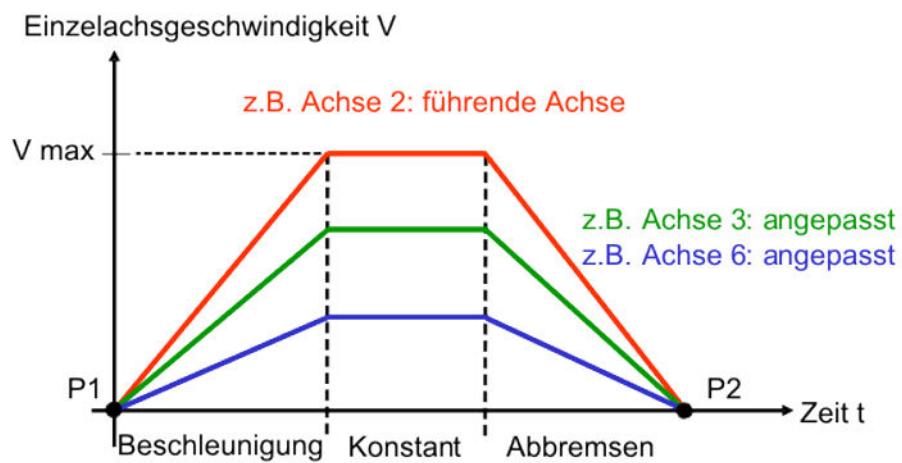


Abb. 7-5: Synchro-PTP

Verwendung

Verwendung findet die SPTP-Bewegung, wenn der Roboter Punkte möglichst schnell und gezielt anfahren soll. Die Bahn spielt dabei eine untergeordnete Rolle. Um eine kollisionsfreie Bewegung zu garantieren, kann

der Roboter durch überschliffene Zwischenpunkte zum Ziel "geführt" werden. Auch die Fahrt des Roboters zu oder zwischen einer Bahnbewegung kann mittels überschliffenen SPTP-Bewegungen Taktzeit optimiert werden.

Beispiele

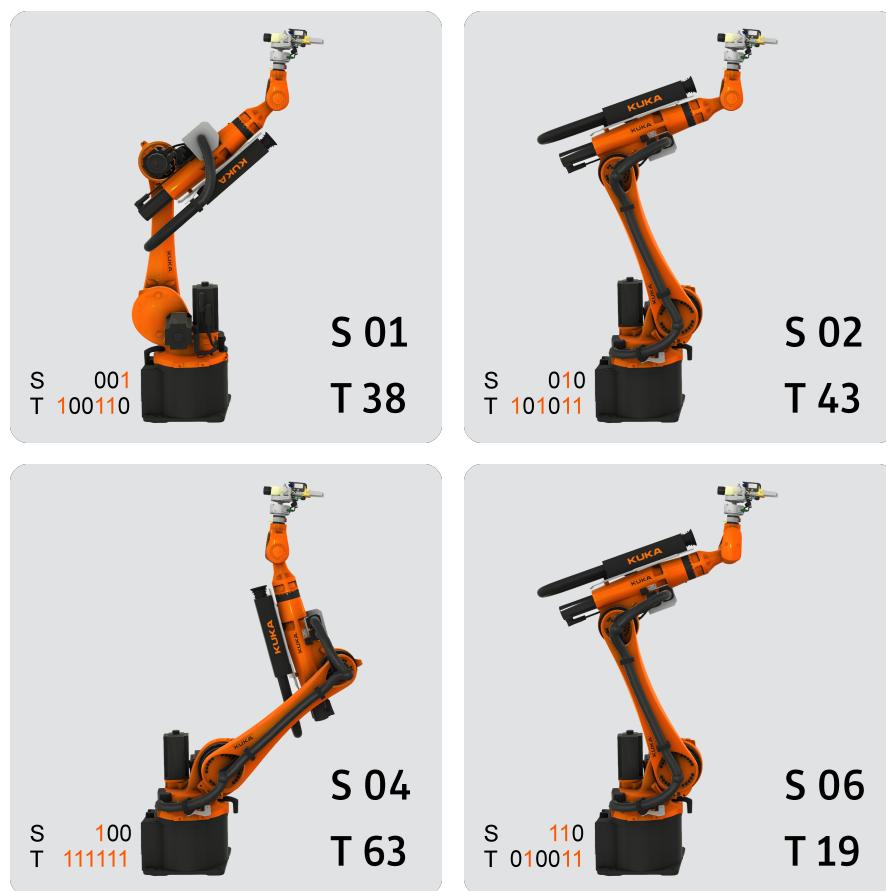
- Punktschweißen und Nieten
- Transportieren
- Messen, Prüfen

7.3.1 Status und Turn

Beschreibung

Status und Turn Auswirkung

Status und Turn dienen dazu, aus mehreren möglichen Achsstellungen für dieselbe Position des TCPs eine eindeutige Achsstellung festzulegen. Unterschiedliche Achsstellungen bedingt durch unterschiedliche Status- und Turn-Werte:



Status und Turn Auswertung

Die programmierten Status- und Turn-Werte berücksichtigt die Robotersteuerung nur bei SPTP-Bewegungen. Bei Bahnbewegungen (SLIN, SCIRC) und in SPLINE-Blöcken (SPL) werden sie ignoriert. Die erste Bewegungsanweisung in einem Programm ist SPTP (HOME).

- Die HOME-Positionen sind vom Typ E6AXIS (config.dat) und definieren eine eindeutige Ausgangslage.
- Weitere mögliche Typen für SPTP sind AXIS, E6POS und POS

```

DEFDAT MAINPROGRAM ()
DECL POS XPOINT1={X 900, Y 0, Z 800, A 0, B 0, C 0, S 6,
T 27}
DECL FDAT FPOINT1 ...
...
ENDDDAT

```

7.3.2 SPTP Überschleifen

Überschleifen

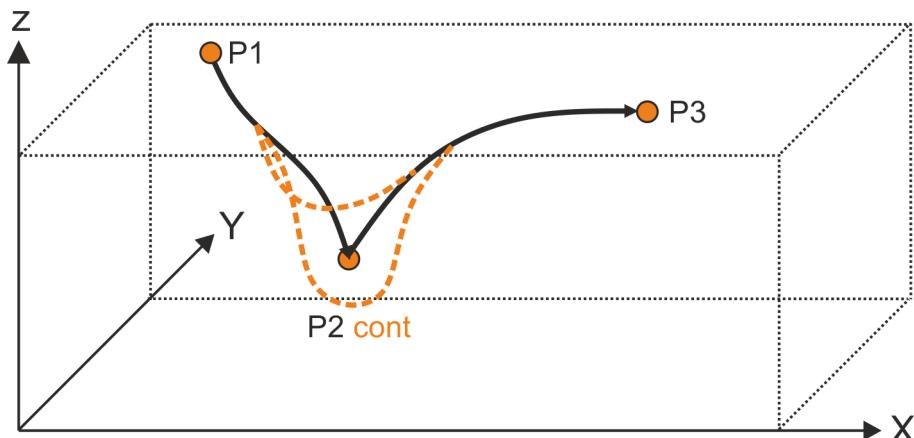


Abb. 7-6: Punkt-Überschleifen eines (S)PTP- Punktes

- Die Steuerung ist in der Lage, zur Beschleunigung des Bewegungsablaufes mit CONT gekennzeichnete Bewegungsbefehle zu überschleifen.
- Überschleifen bedeutet, dass die Punktkoordinate nicht genau angefahren wird.
- Die Bahn der Genauhaltekontur wird vorher verlassen.
- Der TCP wird entlang einer Überschleifikontur geführt, die in der Genauhaltekontur des nächsten Bewegungsbefehls mündet.

Vorteile des Überschleifen

Der Roboter muss zwischen den Punkten nicht mehr abbremsen und beschleunigen (1). Das Programm wird dadurch schneller abgearbeitet (2) was zu einer optimierten Taktzeit und einer hohen Energieeffizienz führt.

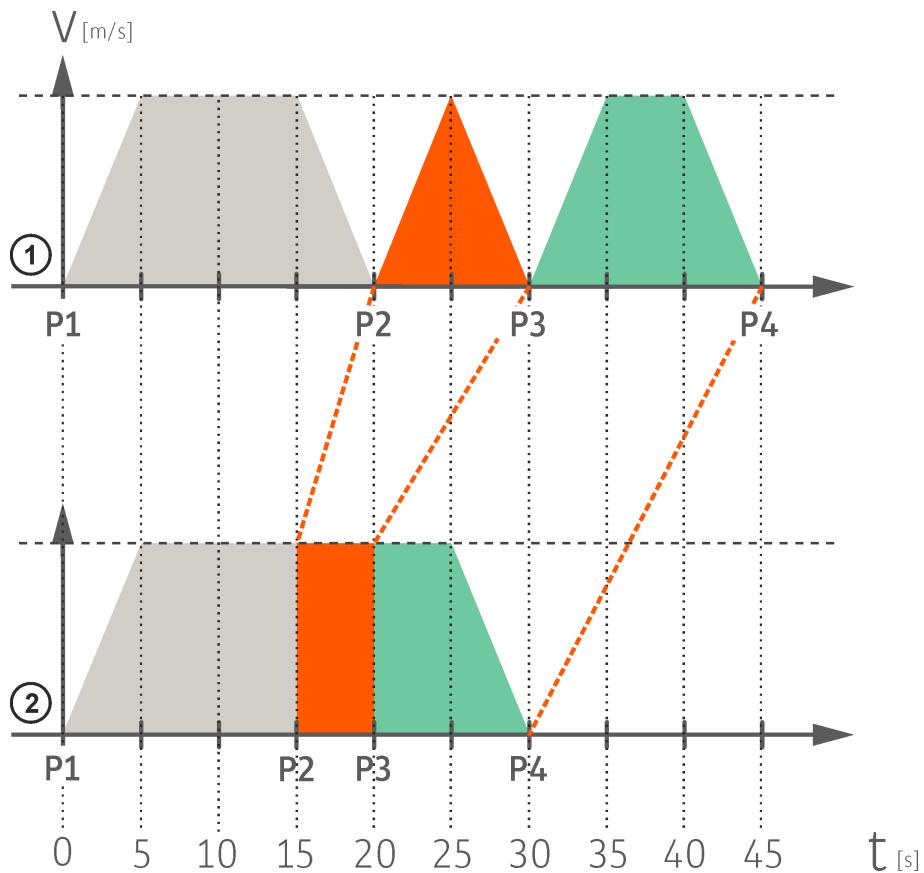


Abb. 7-7: Genauhalt - Überschleifen im Vergleich

SPTP Bahnverhalten beim Überschleifen

Um eine Überschleifbewegung durchführen zu können, muss die Steuerung die nachfolgenden Bewegungssätze einlesen können. Dies erfolgt durch den Rechnervorlauf. Die Angabe der Überschleifdistanz erfolgt in mm.

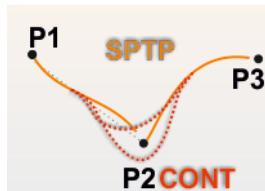


Abb. 7-8



Die Überschleifkontur ist nicht vorhersehbar!

7.3.3 Bewegungen mittels Inlineformular programmieren

Voraussetzung

- Betriebsart T1 ist eingestellt
- Roboterprogramm ist angewählt

Vorgehensweise

1. Den TCP an die Position verfahren, die als Zielpunkt geteacht werden soll.

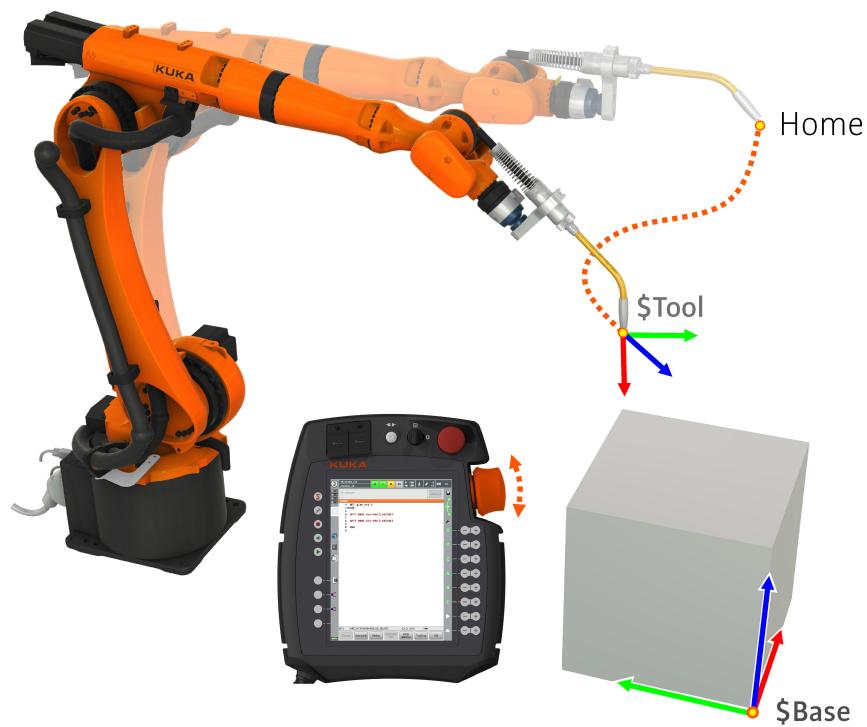


Abb. 7-9: Bewegungsbefehl

2. Inlineformular einfügen

7
8 SLIN P1 Vel=0.5 m/s CPD@3 Tool[1]:Greifer
↳ Base[1]:Basis_Tisch —①
9
10 →
11 ENDLOOP
12
13
14 END
15

Befehle

Letzter Befehl	SPTP
Bewegung	SLIN ③
Bewegungsparameter	SCIRC
Logik	SPLINE Block
Analogausgabe	SPL
Kommentar	PTP SPLINE Block
Trace	PTP
GripperTech	LIN
SpotTech	CIRC

Ändern
Befehle
Bewegung
Fold öffn/schl
Satzanwahl
Touch-Up
Bearbeiten

②

Abb. 7-10: Inlineformular einfügen

- Cursor in die Zeile setzen (1), nach der die Bewegungsanweisung eingefügt werden soll.
- Menüfolge **Befehle (2) > Bewegung > SPTP**

Alternative Vorgehensweise:

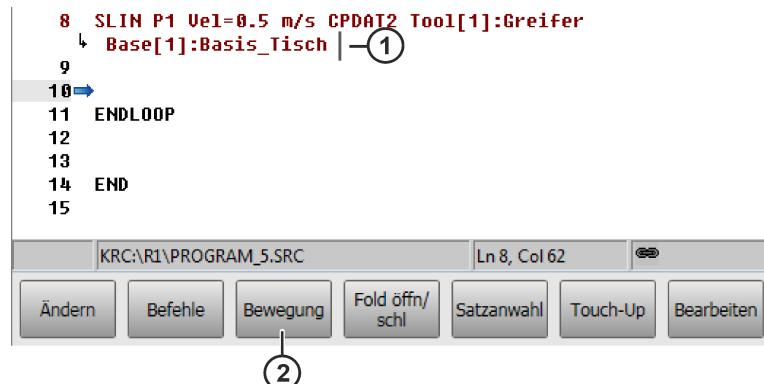


Abb. 7-11: Bewegung einfügen

- Cursor in die Zeile (1) setzen, nach der die Bewegungsanweisung eingefügt werden soll.
 - Über die gleichnamige Schaltfläche eine **Bewegung** (2) einfügen.
3. Ein vorausgefülltes Inlineformular wird nach der markierten Programmzeile eingefügt.

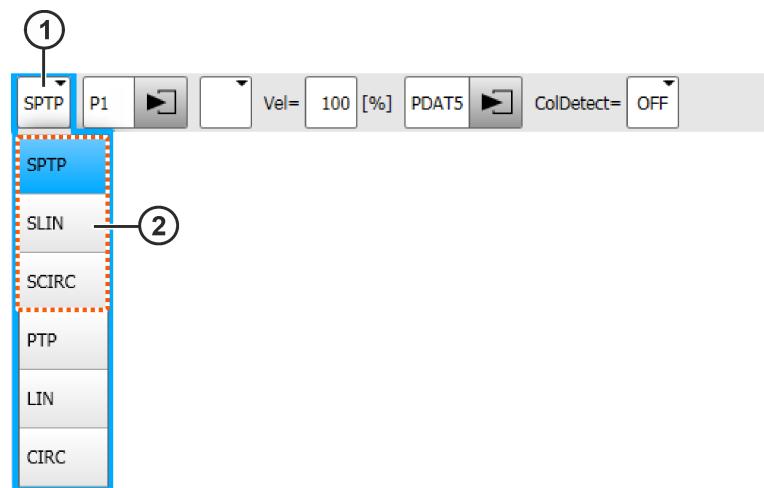


Abb. 7-12: Auswahl der Bewegungsart

Wurde über die Schaltfläche Bewegung das Inlineformular eingefügt, öffnet sich automatisch die Bewegungsauswahl (1). SPTP (2) ist bereits vorausgewählt.

4. Die einzelnen Bewegungsparameter konfigurieren.
 - **SPTP**
(>>> [7.3.3.1 "Inlineformular: SPTP" Seite 204](#))
5. Anweisung mit **Befehl OK** speichern. Die Position des TCP wird als Zielpunkt geteacht.

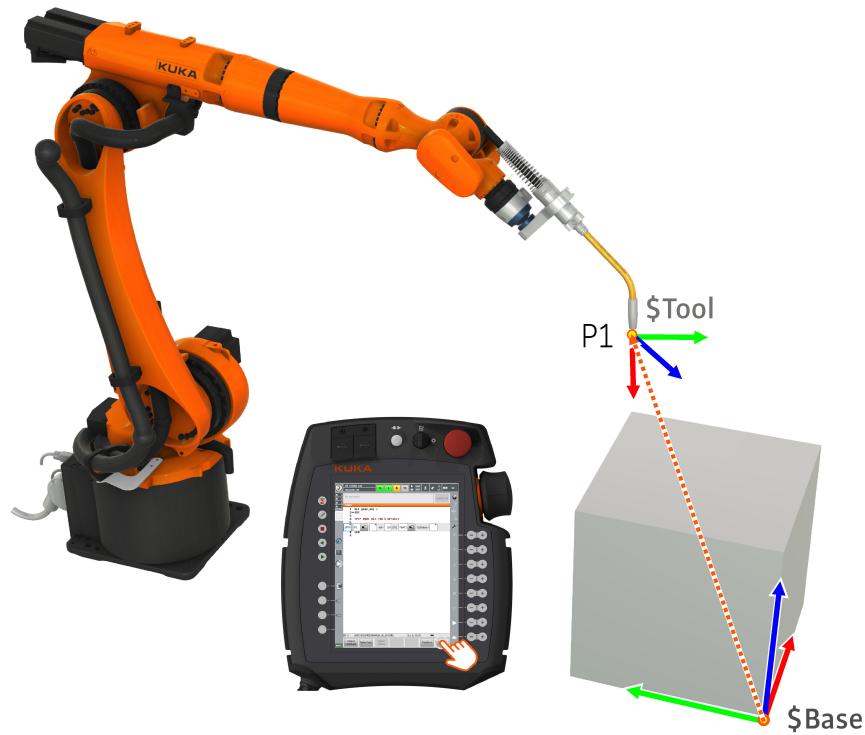


Abb. 7-13: Speicherung der Punktkoordinaten bei "Befehl OK" und "Touchup"

7.3.3.1 Inlineformular: SPTP

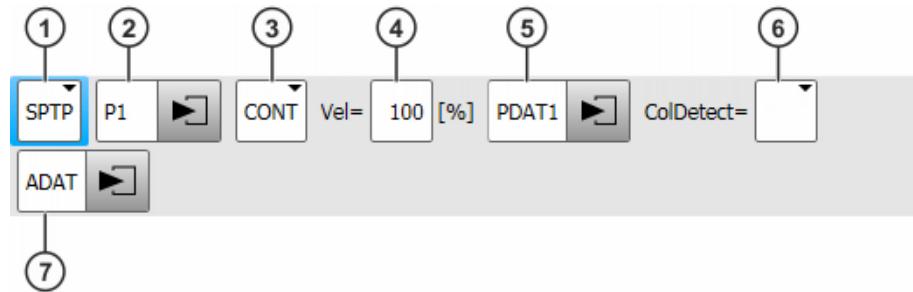


Abb. 7-14: Inline-Formular SPTP (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart SPTP
2	Punktname für Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Pfeilsymbol: (>>> <i>"Werkzeug und Basis setzen" Seite 206</i>)
3	<ul style="list-style-type: none"> CONT: Zielpunkt wird überschliffen. [leer]: Zielpunkt wird genau angefahren.
4	<ul style="list-style-type: none"> Geschwindigkeit 1 ... 100 %
5	Name für den Bewegungsdatensatz. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Pfeilsymbol: (>>> <i>"Bewegungsparameter ändern" Seite 205</i>)

Pos.	Beschreibung
6	Kollisionserkennung für diese Bewegung <ul style="list-style-type: none"> OFF: Die Kollisionserkennung ist ausgeschaltet CDSet_Set[Nr.]: Die Kollisionserkennung ist eingeschaltet. Für die Erkennung werden die Werte aus dem Datensatz Nr. verwendet.
7	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.

Bewegungsparameter ändern

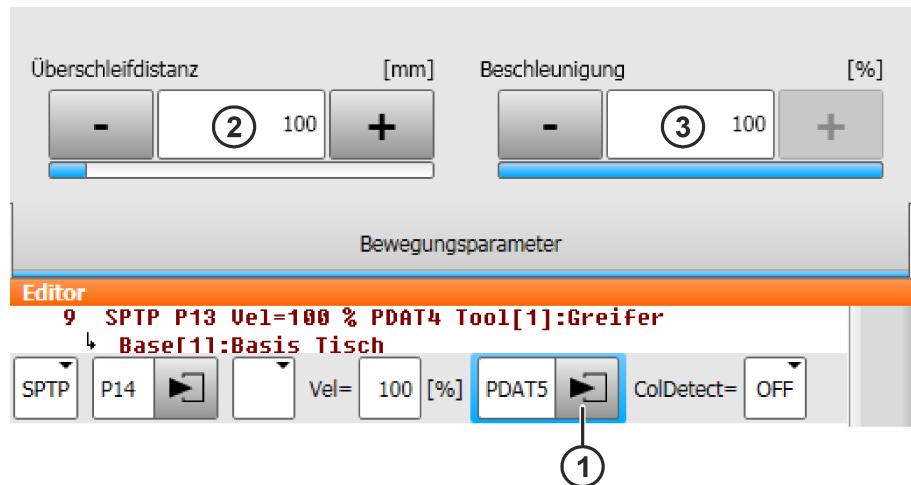


Abb. 7-15: Bewegungsparameter

1. Im geöffneten Inlineformular auf das Pfeilsymbol neben dem PDATx (1) tippen.
2. Im geöffneten Kontextmenü Bewegungsparameter folgende Werte einstellen.
 - Überschleifdistanz (2)
(>>> [7.3.2 "SPTP Überschleifen" Seite 200](#))
 - Beschleunigung (3)

Werkzeug und Basis setzen

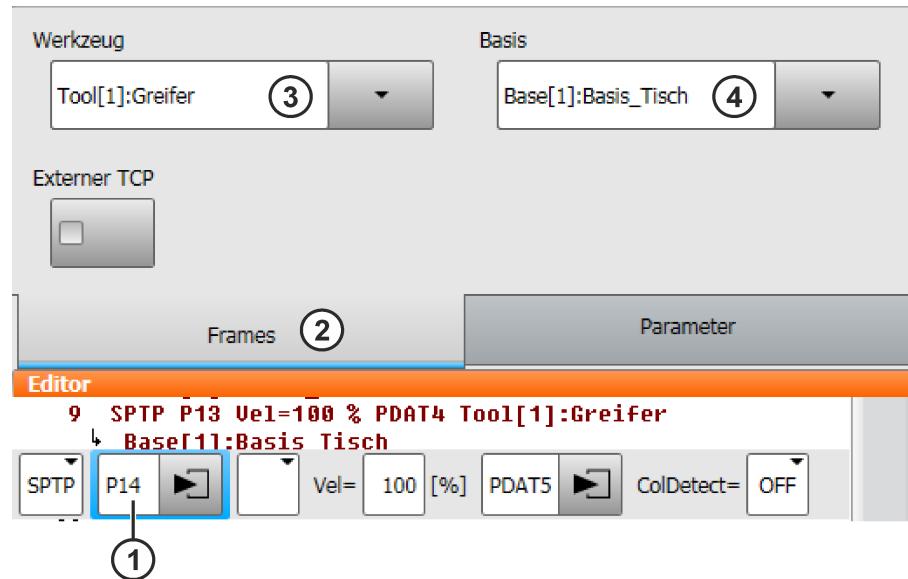


Abb. 7-16: Basis und Werkzeug setzen

1. Im Inline-Formular auf das Pfeilsymbol neben dem Punktnamen tippen.
2. Im geöffneten Kontextmenü über die Registerkarte Frames (2) das Werkzeug (3) und Basis (4) setzen

7.3.4 Übung: Luftprogramm - Programmhandhabung und SPTP-Bewegungen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Luftprogramm - Programmhandhabung und SPTP-Bewegungen**

7.4 Bahnbewegungen erstellen

Bewegungsarten SLIN und SCIRC

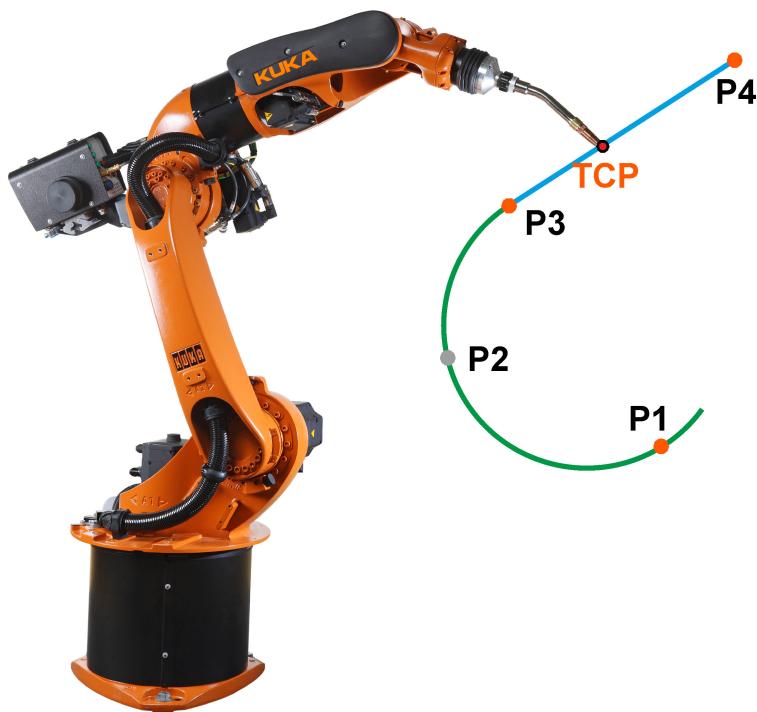
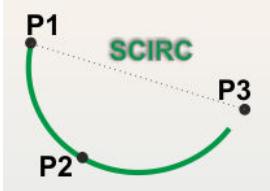


Abb. 7-17: Bahnbewegungen

Bewegungsart	Bedeutung	Verwendungsbeispiel
	<p><i>Linear:</i> Linear</p> <ul style="list-style-type: none"> • Geraadlinige Bahnbewegung: • Der TCP des Werkzeuges wird mit konstanter Geschwindigkeit und definierter Orientierung vom Start zum Zielpunkt geführt. • Geschwindigkeit und Orientierung beziehen sich auf den TCP. 	Bahnapplikationen, z. B.: <ul style="list-style-type: none"> • Bahnschweißen • Kleben • Laser-Schweißen/-Schneiden
	<p><i>Circular:</i> Zirkular</p> <ul style="list-style-type: none"> • Kreisförmige Bahnbewegung ist definiert über Startpunkt, Hilfspunkt und Zielpunkt • Der TCP des Werkzeuges wird mit konstanter Geschwindigkeit und definierter Orientierung vom Start zum Zielpunkt geführt. • Geschwindigkeit und Orientierung beziehen sich auf den TCP des Werkzeugs. 	Bahnapplikationen wie bei SLIN: <ul style="list-style-type: none"> • Kreise, Radien, Rundungen

7.4.1 Singularitäten

Beschreibung

Die KUKA Roboter mit 6 Freiheitsgraden haben 3 verschiedene singuläre Stellungen.

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 Achswinkeländerungen führen, spricht man von singulären Stellungen.

Dies ist keine mechanische, sondern eine mathematische Eigenschaft und existiert aus diesem Grund nur im Bereich der Bahn- nicht aber bei den Achsbewegungen.

Überkopf Singularität

- 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.

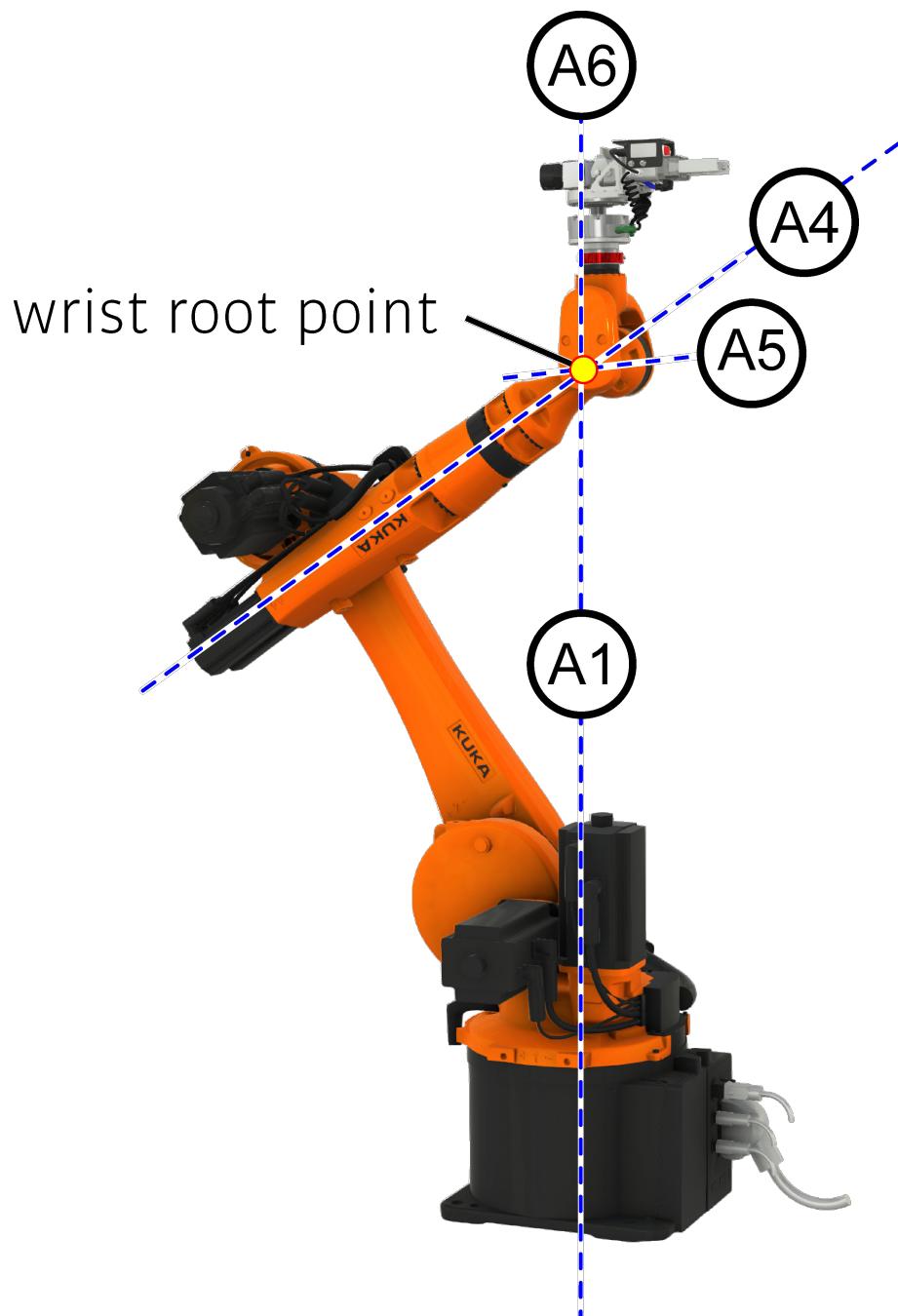


Abb. 7-18: Überkopf-Singularität

Strecklagen Singularität

- 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.

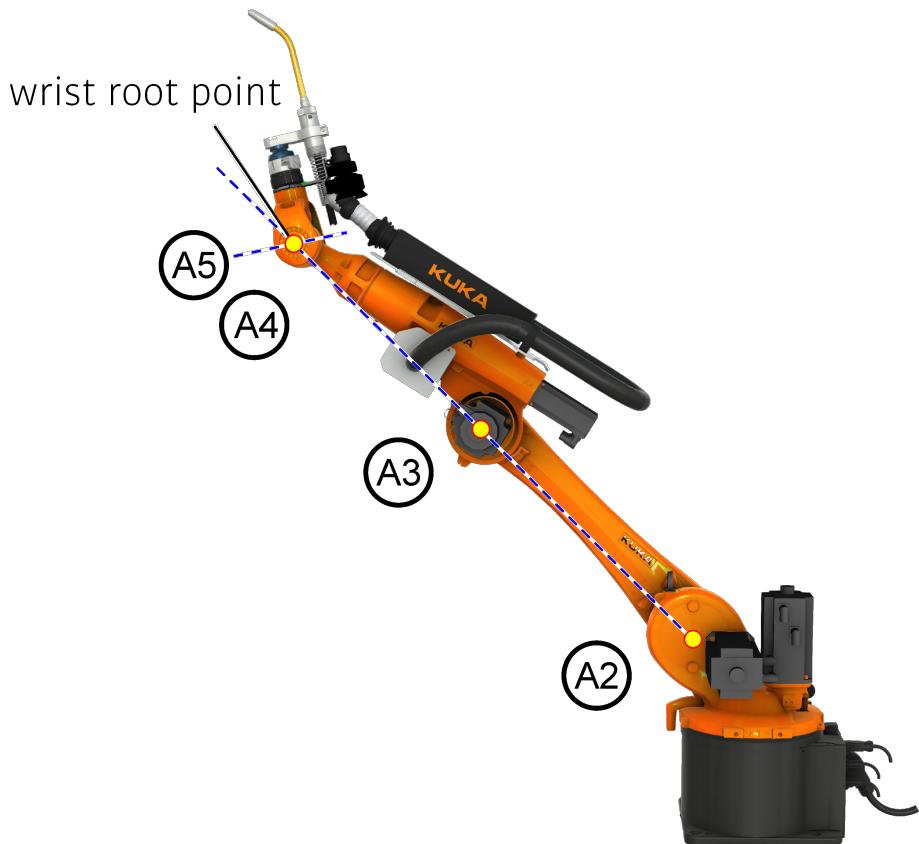


Abb. 7-19: Strecklage

Handachsen Singularität

- Bei der Handachsen-Singularität stehen die Achsen A4 und A6 parallel zueinander und Achse A5 innerhalb des Bereichs von $\pm 0,01812^\circ$.
- 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.

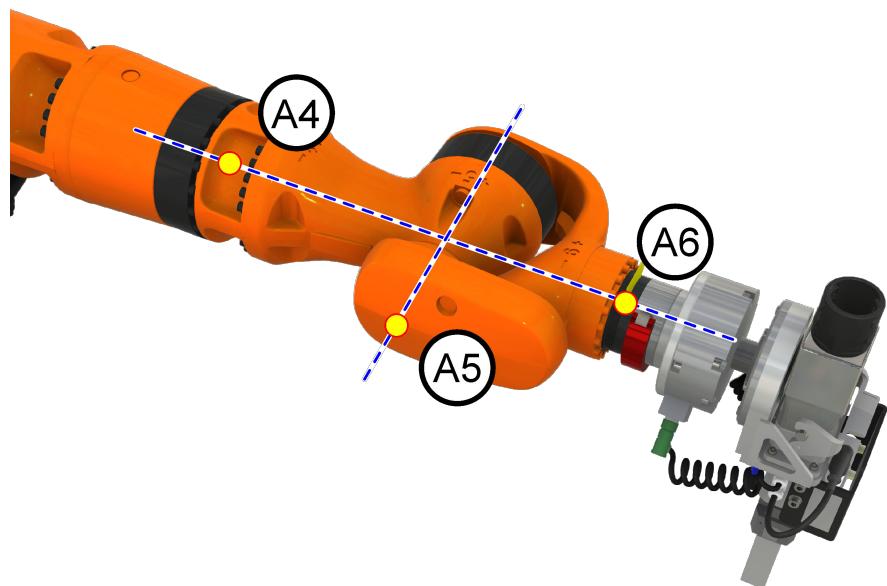


Abb. 7-20: Handachsen-Singularität

7.4.2 SLIN Programmieren

- Den TCP an die Position verfahren, die als Zielpunkt geteacht werden soll.

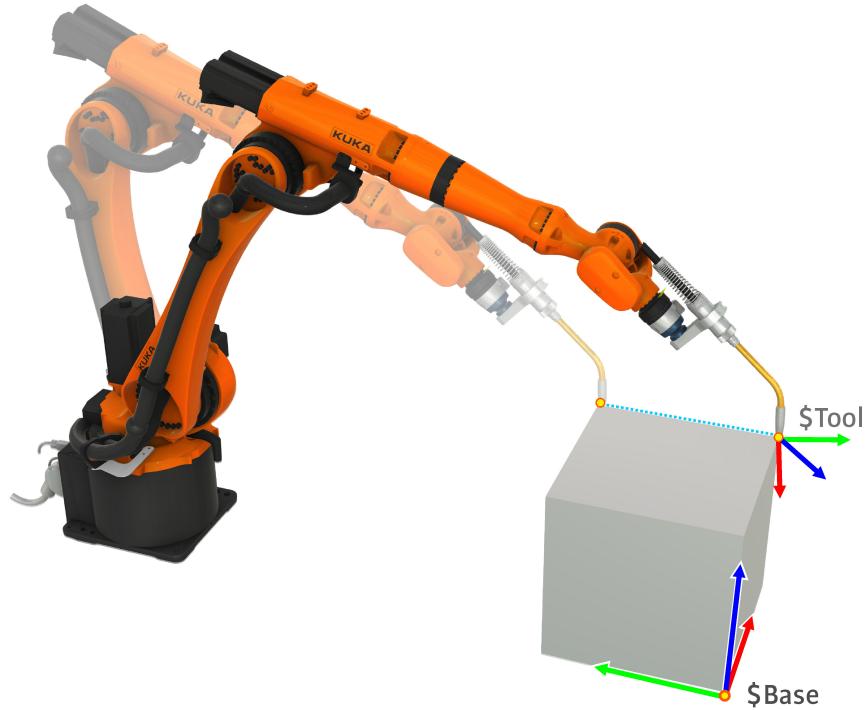


Abb. 7-21: Bewegungsbefehl mit SLIN

- Inline-Formulars SLIN einfügen

```

8   SLIN P1 Vel=0.5 m/s CPDATA Tool[1]:Greifer
    ↓ Base[1]:Basis_Tisch | -①
9
10 →
11 ENDLOOP
12
13
14 END
15

```

KRC:\R1\PROGRAM_5.SRC Ln 8, Col 62

Ändern Befehle **Bewegung** Fold öffn/ schl Satzanwahl Touch-Up Bearbeiten

(2)

Abb. 7-22: Bewegung einfügen

- Cursor in die Zeile (1) setzen, nach der die Bewegungsanweisung eingefügt werden soll.
 - Über die gleichnamige Schaltfläche eine **Bewegung** (2) einfügen.
- Ein vorausgefülltes Inline-Formular wird nach der markierten Programmzeile eingefügt.



Abb. 7-23: Auswahl der Bewegungsart

- Im Inline-Formular auf das Feld mit der Bewegungsart tippen (1).
 - Eine Auswahl öffnet sich, hier die Bewegungsart SLIN (2) wählen.
4. Die einzelnen Bewegungsparameter konfigurieren.
 - **SLIN**
(>> [7.4.2.1 "Inlineformular: SLIN" Seite 214](#))
 5. Anweisung mit **Befehl OK** speichern. Die gegenwärtige Position des TCP wird als Zielpunkt geteacht.

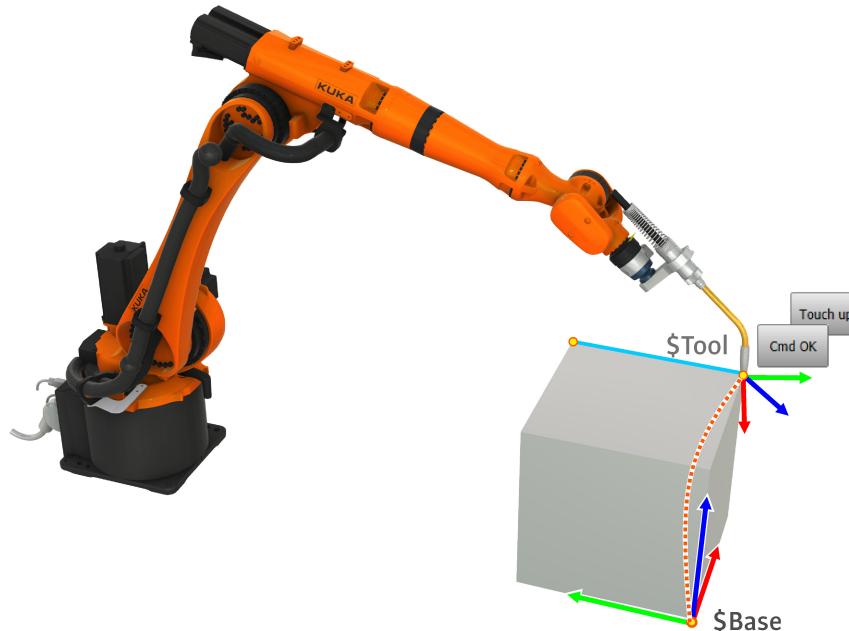


Abb. 7-24: Speicherung der Punktkoordinaten bei "Befehl OK" und "Touchup"

7.4.2.1 Inlineformular: SLIN

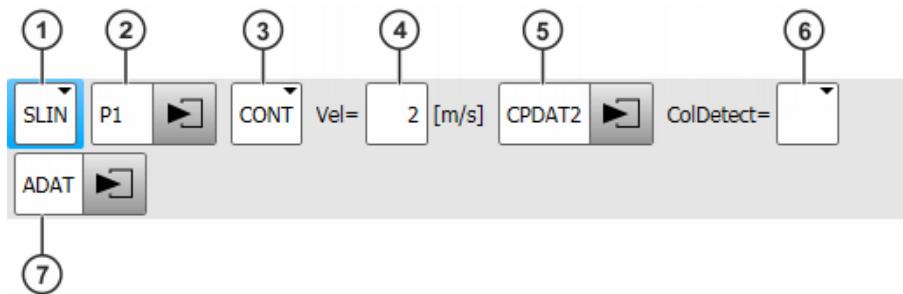


Abb. 7-25: Inline-Formular SLIN (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart SLIN
2	Punktname für Zielpunkt. Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Pfeilsymbol: (>>> "Werkzeug und Basis setzen" Seite 206)
3	<ul style="list-style-type: none"> CONT: Zielpunkt wird überschliffen. [leer]: Zielpunkt wird genau angefahren.
4	Geschwindigkeit <ul style="list-style-type: none"> 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.
6	Kollisionserkennung für diese Bewegung <ul style="list-style-type: none"> OFF: Die Kollisionserkennung ist ausgeschaltet CDSet_Set[Nr.]: Die Kollisionserkennung ist eingeschaltet. Für die Erkennung werden die Werte aus dem Datensatz Nr. verwendet.
7	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.

Orientierungsführung

- Bei Bahnbewegungen besteht die Möglichkeit die Orientierungsführung genau zu definieren.
- Das Werkzeug kann am Start- und am Zielpunkt einer Bewegung unterschiedliche Orientierungen haben.
 - Orientierungsführung des Werkzeugs**
(>>> 7.4.2.2 "Orientierungsführung bei SLIN" Seite 215)

7.4.2.2 Orientierungsführung bei SLIN

Bewegungsart SLIN

Orientierungsführungen bei der Bewegungsart **SLIN**

- **Standard oder Hand PTP**

Die Orientierung des Werkzeugs ändert sich während der Bewegung kontinuierlich.

Hand PTP dann verwenden, bevor der Roboter mit Standard in eine Handachsensingularität gerät, da die Orientierung durch lineare Überführung (achsspezifisches Verfahren) der Handachswinkel geschieht.

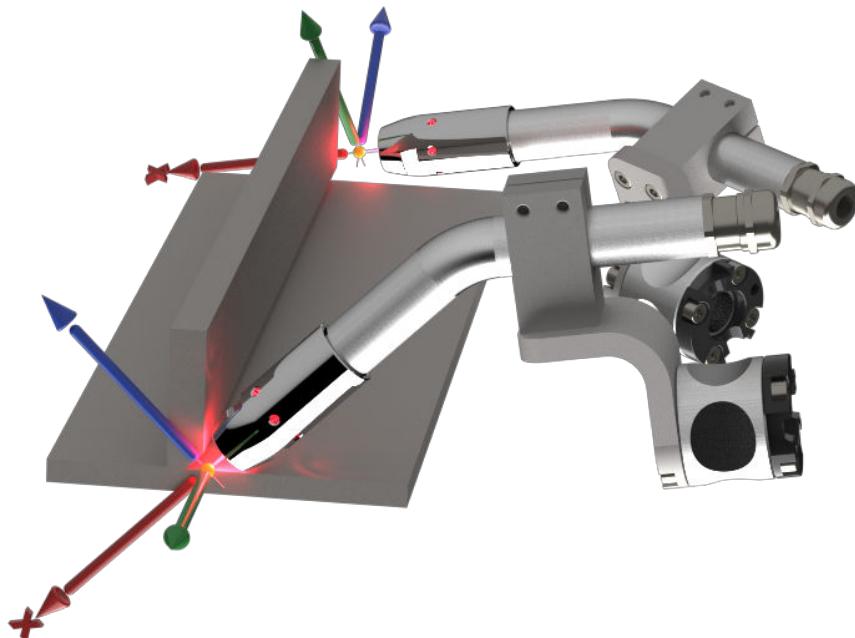


Abb. 7-26: Orientierungsführnng: Standard



VORSICHT

Singularitäten können zu einer schnellen und ausgeprägten Umorientierung führen. Insbesondere bei großen Werkzeugen können große Auslenkungen zu Sach- oder gar Personenschäden führen.

- **Konstant**

Die am Startpunkt geteachte Orientierung des Werkzeugs bleibt während der gesamten Bewegung konstant (oder erhalten). Die am Endpunkt geteichte Orientierung wird ignoriert (Orientierung des orangen Werkzeugs im Bild)

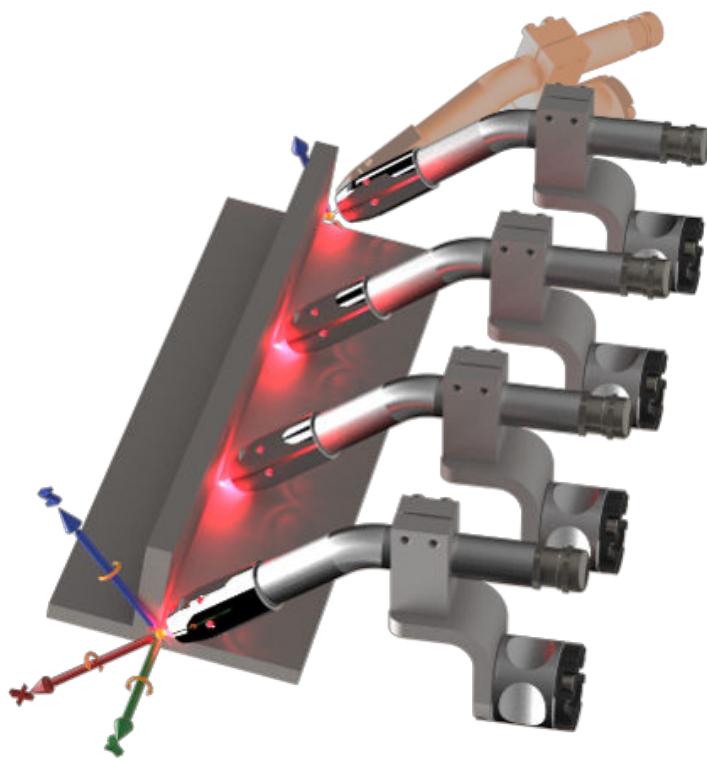


Abb. 7-27: Orientierungsführung: Konstant

Vorgehensweise

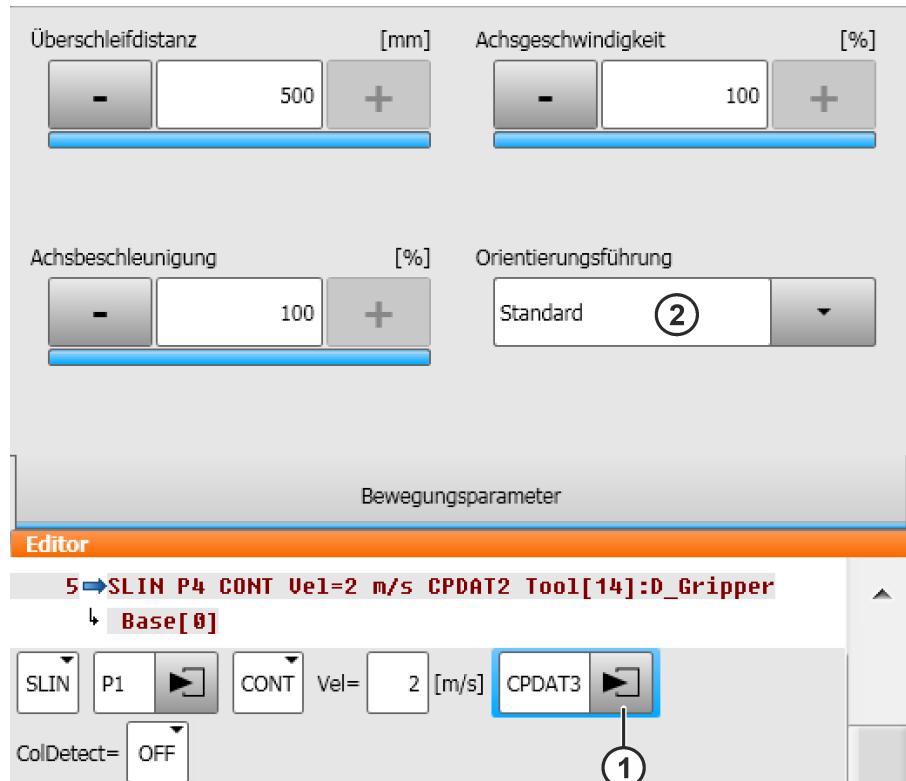


Abb. 7-28: Bewegungsparameter SLIN

1. Im geöffneten Inlineformular auf das Pfeilsymbol im Feld CPDATn (1) tippen.
2. Im eingeblendeten Fenster Bewertungsparameter folgende Anpassungen vornehmen

- Orientierungsführung (2)
(>>> "Bewegungsart SLIN" Seite 215)
- Werte
 - Standard
 - Hand PTP
 - Konstante Orientierungsführung

7.4.3 SCIRC Programmieren

Bewegungsablauf bei SCIRC

Hier bewegt sich der Bezugspunkt des Werkzeugs auf einem Kreisbogen zum Zielpunkt. Die Bahn wird durch Start-, Hilfs- und Endpunkt beschrieben. Als Startpunkt gilt dabei der Zielpunkt des vorangegangenen Bewegungsbefehles.

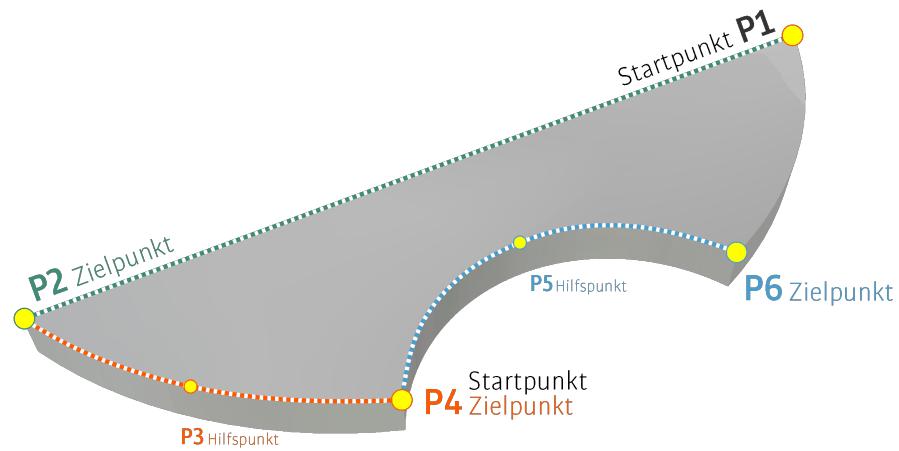


Abb. 7-29: Zwei Kreissegmente mit SCIRC

1. Den TCP an die Position verfahren, die als **HP**-Hilfspunkt und **ZP**-Zielpunkt aufgenommen werden sollen.

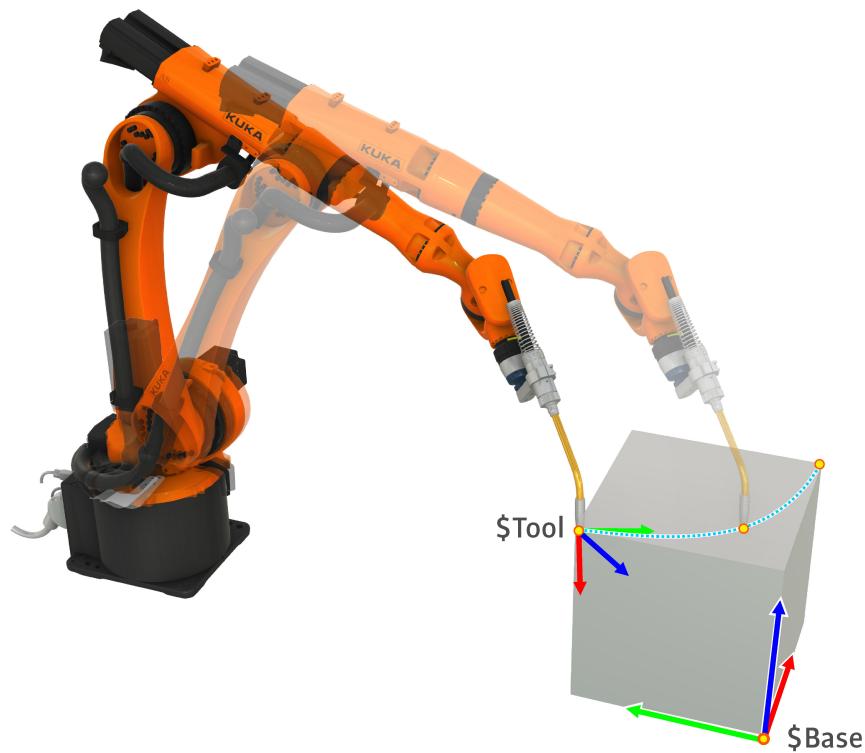


Abb. 7-30: Bewegungsbefehl mit SCIRC

2. Inline-Formulars **SCIRC** einfügen

```

8 SLIN P1 Vel=0.5 m/s CPDATA2 Tool[1]:Greifer
   ↓ Base[1]:Basis_Tisch | -①
9
10 ➔
11 ENDLOOP
12
13
14 END
15

```



Abb. 7-31: Bewegung einfügen

- Cursor in die Zeile (1) setzen, nach der die Bewegungsanweisung eingefügt werden soll.
 - Über die gleichnamige Schaltfläche eine **Bewegung** (2) einfügen.
3. Ein vorausgefülltes Inline-Formular wird nach der markierten Programmzeile eingefügt.



Abb. 7-32: Auswahl der Bewegungsart

- Im Inline-Formular auf das Feld Bewegungsart tippen (1).
 - Eine Auswahl öffnet sich, hier die Bewegungsart SCIRC (2) wählen.
4. Die einzelnen Bewegungsparameter konfigurieren.
- **SCIRC**
(>>> [7.4.3.1 "Inlineformular: SCIRC" Seite 220](#))

5. Hilfs- und Zielpunkt teachen

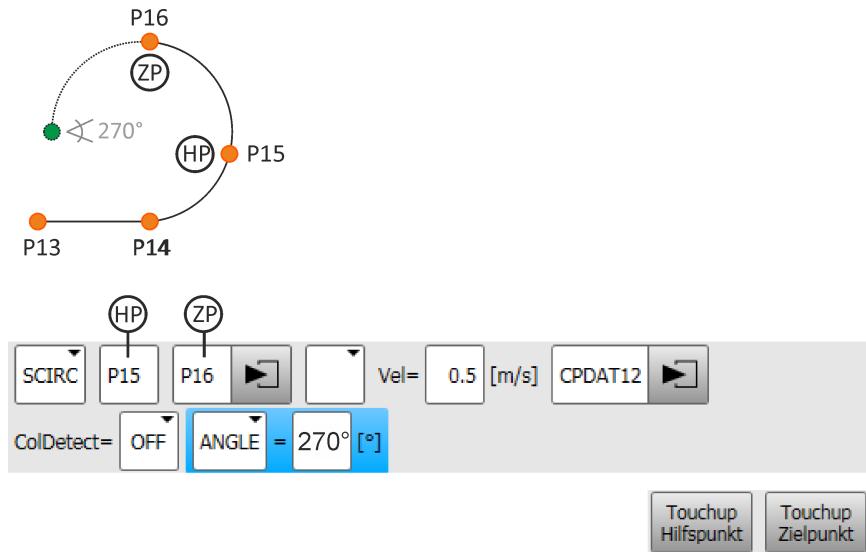


Abb. 7-33: SCIRC Punkte teachen

- Den Hilfspunkt auf der Kreisbahn (HP) anfahren und diesen mit der Schaltfläche **Touch-Up Hilfspunkt** übernehmen.
- Den Zeitpunkt auf der Kreisbahn (ZP) anfahren und diesen mit der Schaltfläche **Touch-Up Zielpunkt** übernehmen.



Optional kann über den Zusatzparameter ANGLE (blaues Feld) die Kreisbahn über den Zielpunkt hinaus erweitert oder verkürzt werden. Hierzu ist eine Winkleingabe notwendig.

HINWEIS

Bei der Änderung des Kreiswinkels ist der weitere Bahnverlauf zu prüfen. Kollisionsgefahr

6. Anweisung mit **Befehl OK** speichern.

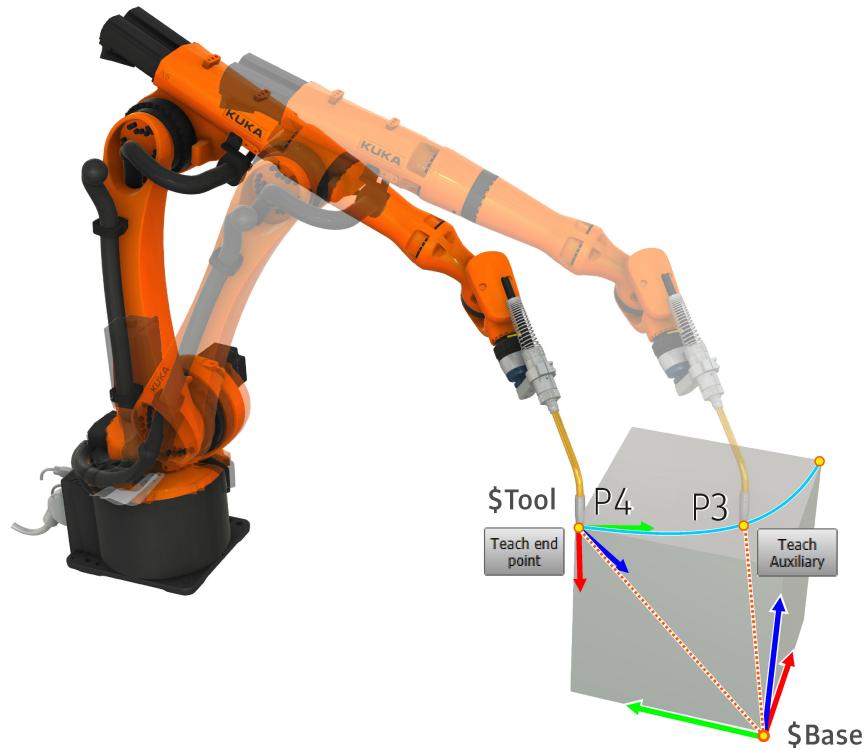


Abb. 7-34: Speicherung der Punktkoordinaten bei "TouchUp HP" und "Touchup ZP"

7.4.3.1 Inlineformular: SCIRC

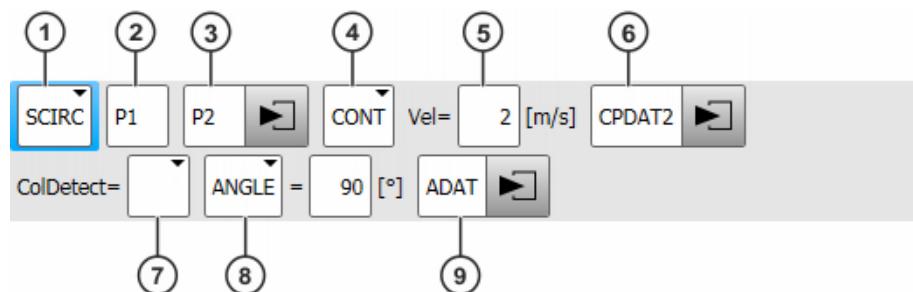


Abb. 7-35: Inline-Formular SCIRC (Einzelbewegung)

Pos.	Beschreibung
1	Bewegungsart SCIRC
2	Punktname für den Hilfspunkt Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden.
3	Punktname für den Zielpunkt Das System vergibt automatisch einen Namen. Der Name kann überschrieben werden. Pfeilsymbol: (>> "Werkzeug und Basis setzen" Seite 206)
4	<ul style="list-style-type: none"> CONT: Zielpunkt wird überschliffen. [leer]: Zielpunkt wird genau angefahren.

Pos.	Beschreibung
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.
7	Kollisionserkennung für diese Bewegung • OFF : Die Kollisionserkennung ist ausgeschaltet • CDSet_Set[Nr.] : Die Kollisionserkennung ist eingeschaltet. Für die Erkennung werden die Werte aus dem Datensatz Nr. verwendet.
8	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.
9	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.

Orientierungsverhalten

- Bei Bahnbewegungen besteht die Möglichkeit die Orientierungsführung genau zu definieren.
- Das Werkzeug kann am Start- und am Zielpunkt einer Bewegung unterschiedliche Orientierungen haben.
 - Orientierungsverhalten des Werkzeugs**
(>>> [7.4.3.3 "Orientierungsführung bei SCIRC" Seite 222](#))
- Die Orientierung des Werkzeugs kann beim Abfahren der Kreisbahn durch weitere Parameter beeinflusst werden:
 - Orientierungsverhalten im Hilfspunkt**
(>>> [7.4.3.4 "SCIRC: Orientierungsverhalten – Beispiel Hilfspunkt" Seite 224](#))
 - Orientierungsverhalten im Zielpunkt**
(>>> [7.4.3.5 "SCIRC: Orientierungsverhalten – Beispiel Zielpunkt" Seite 228](#))

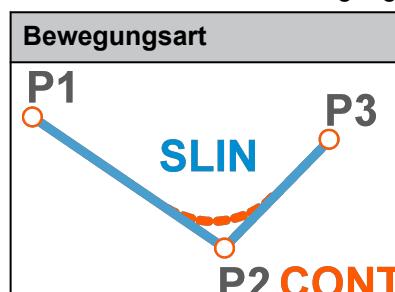
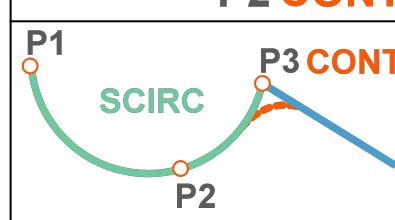
7.4.3.2 Überschleifen bei Bahnbewegungen

Überschleifen von Bahnbewegungen



Die Überschleiffunktion ist nicht zum Erzeugen von Kreisbewegungen geeignet. Es handelt sich lediglich um das Verhindern eines Genauhaltes im Punkt.

Überschleifen in den Bewegungsarten SLIN und SCIRC

Bewegungsart	Merkmal
	<ul style="list-style-type: none"> • Bahnverlauf entspricht zwei Parabelästen • Angabe der Überschleifdistanz in mm
	<ul style="list-style-type: none"> • Bahnverlauf entspricht zwei Parabelästen • der Zielpunkt wird überschliffen • Angabe der Überschleifdistanz in mm

7.4.3.3 Orientierungsführung bei SCIRC

Orientierungsführung: Standard oder Hand-PTP

Die Orientierung des Werkzeugs ändert sich während der Bewegung kontinuierlich. Hand PTP wird dann verwenden, bevor der Roboter mit Standard in eine Handachsensingularität gerät, da die Orientierung durch lineare Überführung (achsspezifisches Verfahren) der Handachswinkel geschieht.

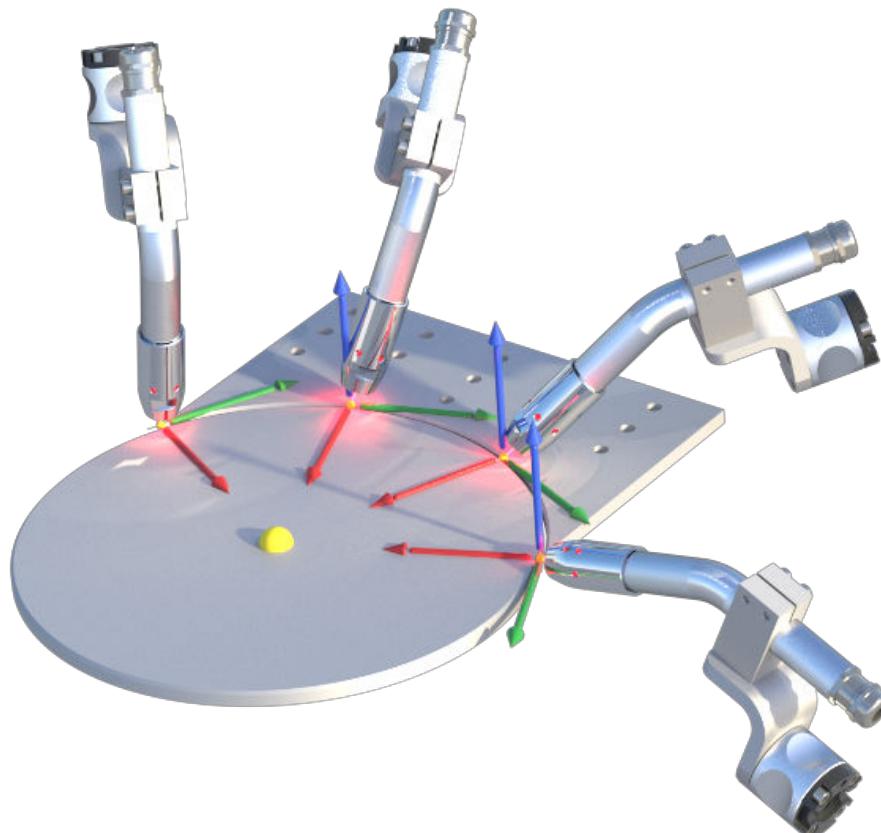
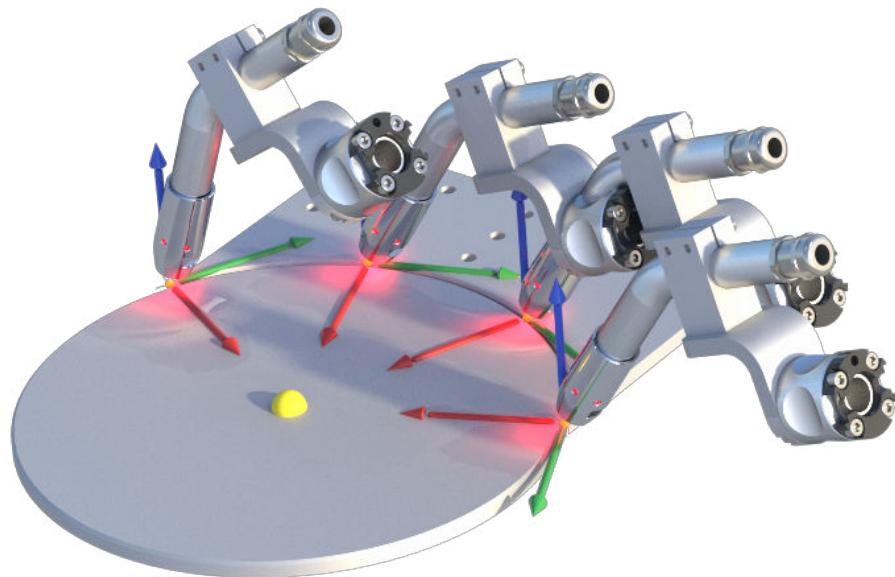
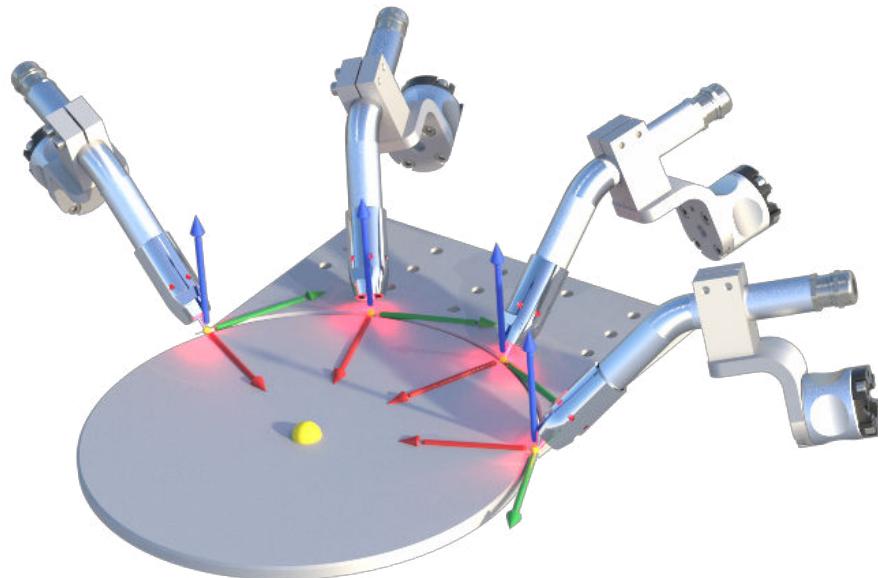


Abb. 7-36: Orientierungs: Standard + Basis-Bezug

Orientierungsführung: Konstant und Basis-Bezug

Die Orientierung des Werkzeugs bleibt während der Bewegung konstant, wie sie am Startpunkt geteacht wurde. Die am Endpunkt geteachte Orientierung wird ignoriert.

**Abb. 7-37: Orientierungs: Konstant + Basis-Bezug****Orientierungsführung: Konstant und Bahn-Bezug****Abb. 7-38: Orientierungs: Konstant + Bahn-Bezug**

Vorgehensweise

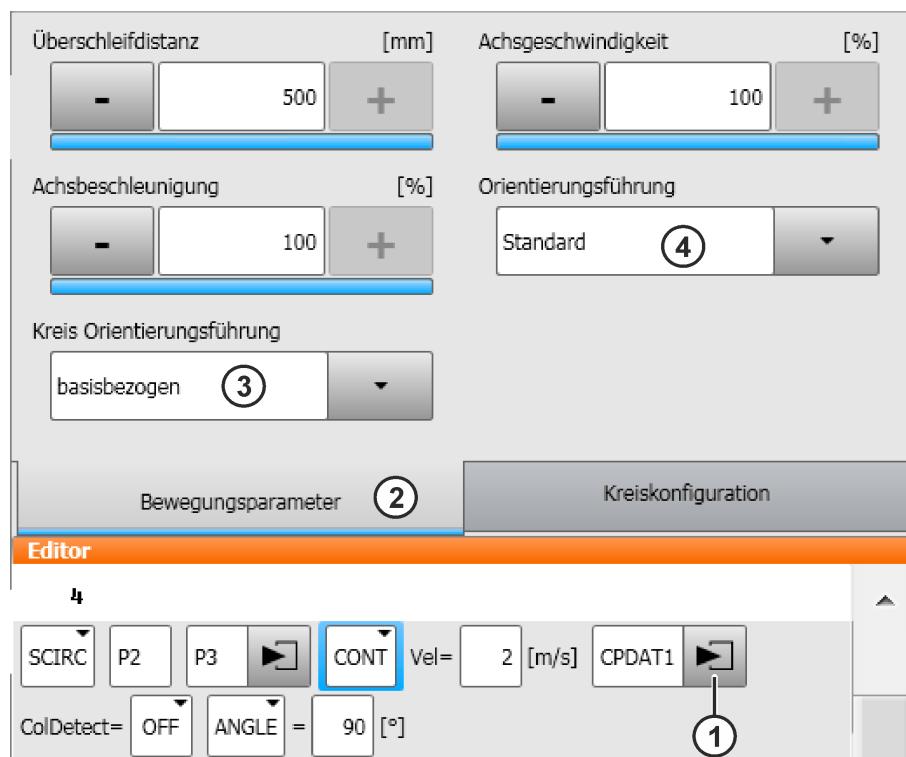


Abb. 7-39: Bewegungsparameter

1. Im geöffneten Inline-Formular auf das Pfeilsymbol im Feld CPDATn (1) tippen.
2. Im eingeblendeten Kontextfenster in den Registerkarte Bewegungsparameter (2) wechseln.

Parameter

3	Kreis Orientierungsführung <ul style="list-style-type: none"> • basisbezogen • bahnbezogen
4	Orientierungsführung <ul style="list-style-type: none"> • Standard • Hand PTP • Konstante Orientierungsführung

7.4.3.4 SCIRC: Orientierungsverhalten – Beispiel Hilfspunkt

Ausgangssituation

- 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 $360^\circ - 197^\circ = 163^\circ$ 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.

Interpolieren #INTERPOLATE

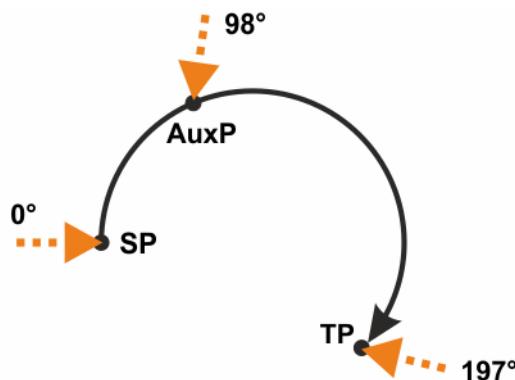


Abb. 7-40: #INTERPOLATE

SP Startpunkt

AuxP Hilfspunkt

TP Zielpunkt

- Der TCP nimmt am Hilfspunkt die programmierte Orientierung von 98° an.
- Die Umorientierung beträgt 197°.

Ignorieren #IGNORE

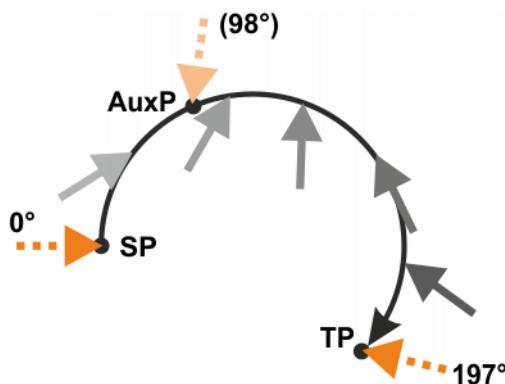
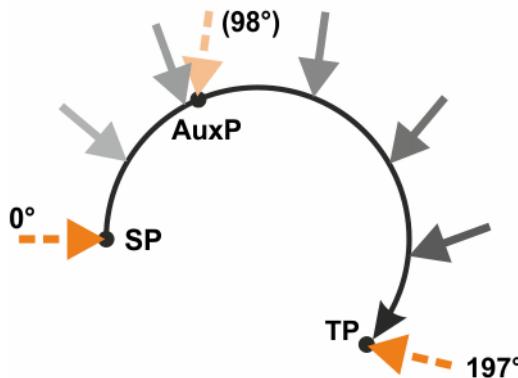


Abb. 7-41: #IGNORE

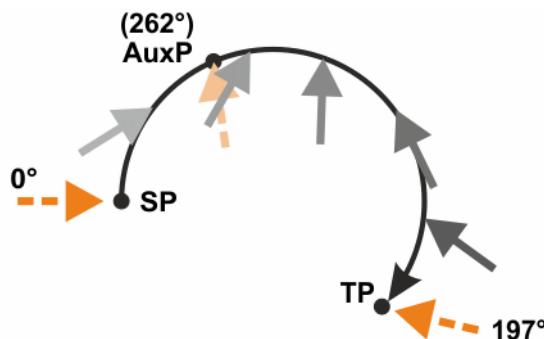
- Die kurze Umorientierung mit 163° wird gefahren.
- Die programmierte Orientierung des Hilfspunkts wird ignoriert.

Berücksichtigen #CONSIDER**Abb. 7-42: #CONSIDER**

- 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.



Berücksichtigen #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.

Weiteres Beispiel für Berücksichtigen #CONSIDER**Abb. 7-43: #CONSIDER, weiteres Beispiel**

- 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.

Vorgehensweise

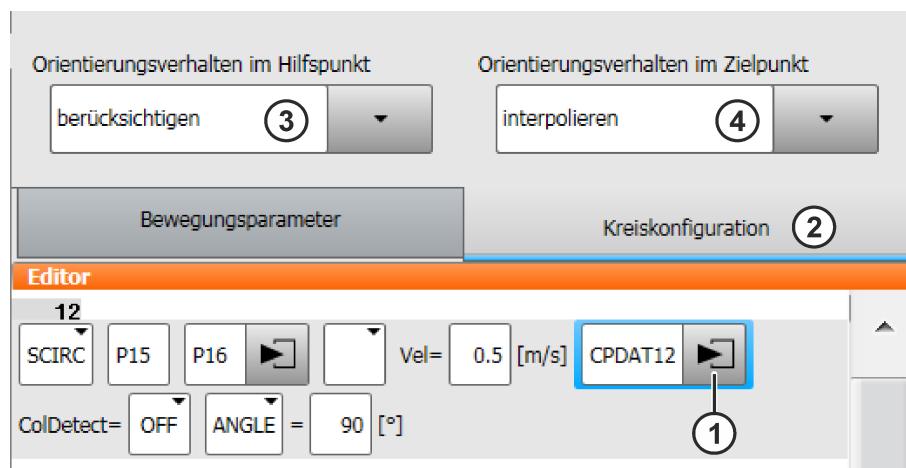


Abb. 7-44: SCIRC Orientierungsverhalten

1. Im geöffneten Inlineformular auf das Pfeilsymbol im Feld CPDATn (1) tippen.
2. Im eingeblendeten Kontextfenster in den Reiter Kreiskonfiguration (2) wechseln.
3. Orientierungsverhalten für den Hilfspunkt (3) und Zielpunkt (4) festlegen.

Element	Beschreibung
Orientierungsverhalten im Hilfspunkt	<p>Pulldown-Menü</p> <ul style="list-style-type: none"> • interpolieren: Im Hilfspunkt nimmt der TCP die programmierte Orientierung an. • ignorieren: Die Robotersteuerung ignoriert die programmierte Orientierung des Hilfspunkts. Die Start-Orientierung des TCP wird auf dem kürzesten Weg in die Ziel-Orientierung überführt. • berücksichtigen (Default): Die Robotersteuerung wählt den Weg, der der programmierten Orientierung des Hilfspunkts näher kommt.
Orientierungsverhalten im Zielpunkt	<p>Pulldown-Menü</p> <ul style="list-style-type: none"> • interpolieren: Am tatsächlichen Zielpunkt wird die programmierte Orientierung des Zielpunkts angenommen. (Einige Möglichkeit für SCIRC ohne Kreiswinkel-Angabe. Wenn EXTRAPOLATE gesetzt wird, wird trotzdem INTERPOLATE ausgeführt.) • extrapolieren(Default für SCIRC mit Kreiswinkel-Angabe): 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.

7.4.3.5 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.

Interpolieren #INTERPOLATE

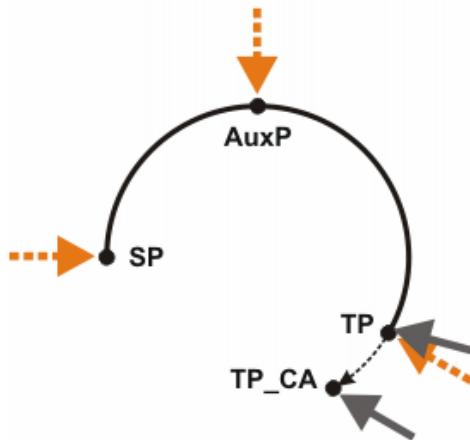


Abb. 7-45: #INTERPOLATE

- | | |
|--------------|---|
| SP | Startpunkt |
| AuxP | Hilfspunkt |
| TP | Programmierter Zielpunkt |
| TP_CA | Tatsächlicher Zielpunkt. Ergibt sich durch den Kreiswinkel. |
- In **TP**, der sich vor **TP_CA** befindet, ist die programmierte Orientierung noch nicht erreicht.
 - In **TP_CA** wird die programmierte Orientierung angenommen.

Extrapolieren #EXTRAPOLATE

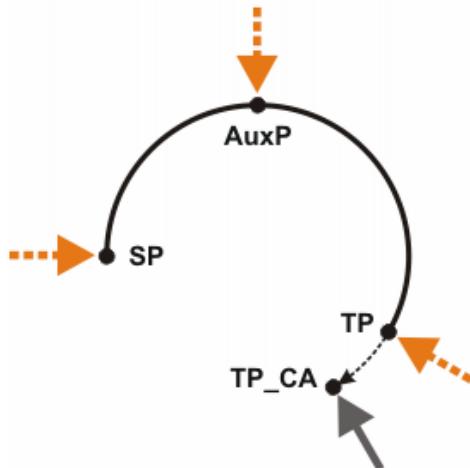


Abb. 7-46: #EXTRAPOLATE

- In **TP** wird die programmierte Orientierung angenommen.
- Für **TP_CA** wird diese Orientierung gemäß dem Kreiswinkel weitergeführt.

Vorgehensweise

Siehe:

SCIRC: Orientierungsverhalten - Beispiel Hilfspunkt > Vorgehensweise
(>>> Abb. 7-44)

7.4.3.6 Einschränkungen bei \$CIRC_MODE

Einschränkungen

Auszug aus der Beschreibung der Systemvariablen

- 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.

Erklärung zu Fehlermeldungen

Manche Fehlermeldungen enthalten den Text "**Fehler wegen Regel x**"

Bei der Programmierung von **\$CIRC_MODE** sind sowohl für die Orientierung als auch die Zusatzachsen zu beachten:

1. Regel 1: **#CONSIDER** ist genau dann erlaubt, wenn Start- und Zielpunkt nicht ignoriert werden.
2. Regel 2: **\$CIRC_TYPE= #PATH** ist genau dann erlaubt, wenn Start- und Zielpunkt nicht ignoriert werden.
3. Regel 3: Wenn **\$ORI_TYPE = #IGNORE** oder **\$EX_AX_IGNORE** gesetzt sind, dann wird **\$CIRC_MODE** nicht mehr ausgewertet.
4. Regel 4: Wenn ein Kreiswinkel programmiert ist, ist Interpolation im Hilfspunkt verboten.
5. Regel 5: Wenn ein Kreiswinkel programmiert ist, darf der Zielpunkt genau dann durch Extrapolation bestimmt werden, wenn Start- und Zielpunkt nicht ignoriert werden.
6. Regel 6: Wenn ein Kreiswinkel programmiert ist, darf der Zielpunkt genau dann übernommen (interpoliert) werden, wenn er nicht ignoriert wird.
7. Regel 7: **#CONSIDER** Weg wirkt nur bei endlos-drehenden Zusatzachsen. Bei andersartigen Achsen wird immer der kurze Weg gewählt, was **#IGNORE** entspricht.
8. Regel 8: Die Komponente **TARGET_PT** wird nur berücksichtigt, wenn ein Kreiswinkel programmiert ist.
9. Regel 9: Das Lesen von **\$CIRC_MODE** ist nirgends erlaubt, das Schreiben nur im WITH-Token eines SCIRC.

7.4.4 **Bahnfahren und Überschleifen**

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bahnfahren und Überschleifen**

7.5 Mit globalen Punkten programmieren

Beschreibung

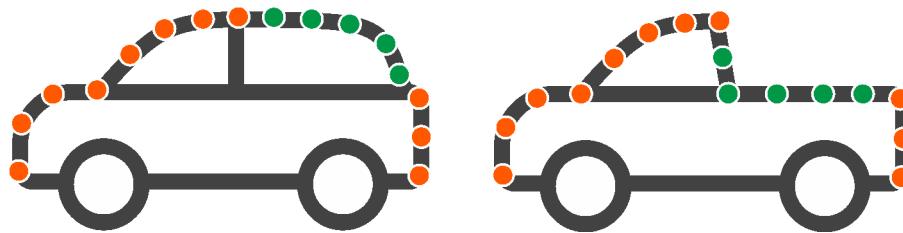


Abb. 7-47: globale und lokale Punkte

- In dem obigen Beispiel müssen Punkte an zwei ähnliche Fahrzeugen angefahren werden.
 - Die **orangen Punkte** sind bei beiden Karosserien **identisch**.
 - Die **grünen Punkte** sind bei beiden Karosserien **unterschiedlich**.
- Für beide Fahrzeuge sind unterschiedliche Programme vorgesehen.
- Um den Programmier- und Pflegeaufwand zu minimieren, werden
 - die orangefarbenen Punkte als **globale Punkte** geteilt.
 - die grünen Punkte als **lokale Punkte** in der Programm-Datenliste.
- Die globalen Punkte werden auf der Steuerung in einer separaten globalen Punkt-Datenliste gespeichert.



Die Funktion **Globale Punkte teachen** muss durch den Administrator freigeschalten sein.

Rechteverwaltung



Damit auch niedrigere Benutzergruppen auf die Funktionalität globale Punkte teachen zurückgreifen können, muss hierzu vom Administrator diese in der Rechteverwaltung freigeschalten werden.

Rechteverwaltung	
Zuordnung der Funktionsgruppen zu Benutzergruppen	
	Funktionsgruppe
	Satzauswahl
	Allgemeine KRL-Programmänderungen
	Kritische KRL-Programmänderungen
	Lokale Punkte teachen
	Globale Punkte teachen
Teachen und Nachteachen von globalen Punkten	

Abb. 7-48: Rechteverwaltung

Einschränkungen

Globale Punkte sind für folgende Bewegungen möglich

- SLIN (sowohl im Spline-Block als auch einzeln)
- SPTP (sowohl im Spline-Block als auch einzeln)

- LIN
- PTP

Globale Punkte sind nicht möglich für

- SPLINE-Block als Ganzes
- SCIRC
- CIRC



HOME-Positionen sind bereits globale Punkte!

Vorgehensweise

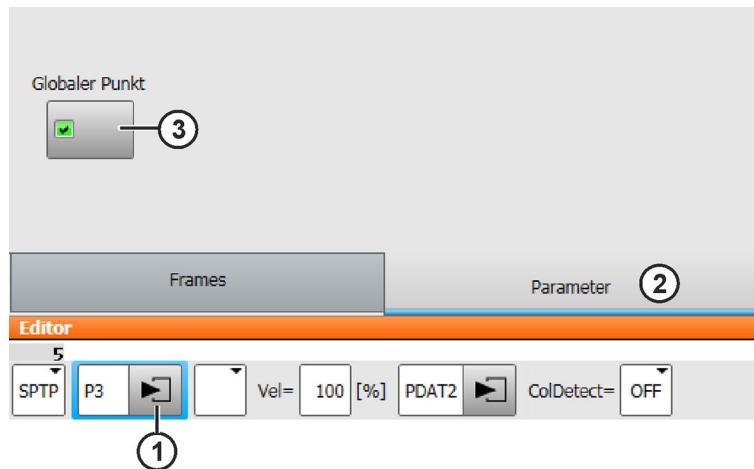


Abb. 7-49: globalen Punkt aktivieren

1. Im Inline-Formular die Punktparameter (1) öffnen.
2. In die Registerkarte **Parameter** (2) wechseln.
3. Über die gleichnamige Schaltfläche den globalen Punkt (3) aktivieren.
4. Markierung eines globalen Punkts im geöffneten Inline-Formulars durch den Zusatz **GLOBAL**

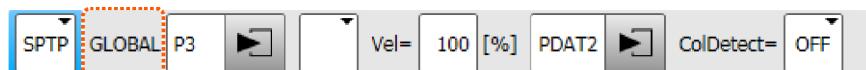


Abb. 7-50: globaler Punkt im Inlineformular

5. Punkt mit der Schaltfläche **OK** übernehmen.
6. Im Editor steht der Name von globalen Punkten in spitzen <> Klammern.
 - **lokal gespeicherter Punkt**

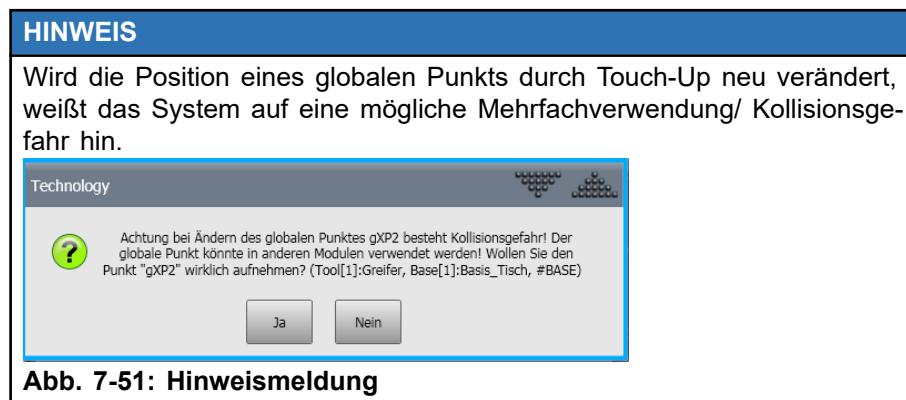
```
1 SPTP P3 Vel = 100% ....
```

- **global gespeicherter Punkt**

```
1 SPTP <P3> Vel = 100% ....
```



Für jeden Punkt kann einzeln entschieden werden, ob er als lokaler oder globaler Punkt abgespeichert wird.



Global_Points.dat

- Punkte, welche im Inlineformular global aktiv gesetzt werden, werden in der Global_Points.dat gespeichert.
- **Menüpfad:** KUKA_DISK C: > KRC > Roboter > KRC > R1 > System > Global_Points.dat

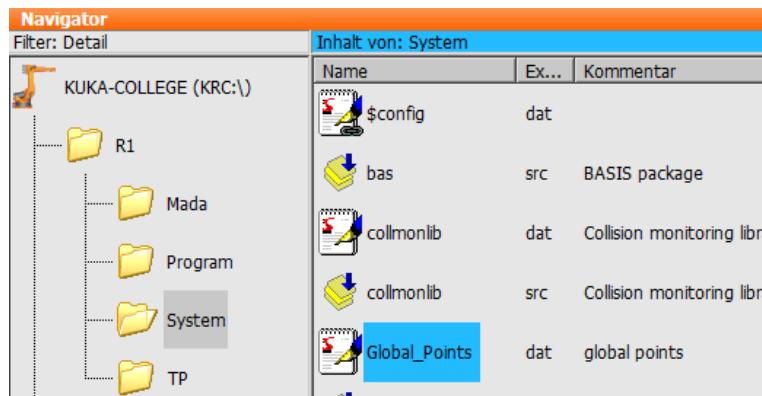


Abb. 7-52: Global_Points.dat

- **Beispiel:** Gespeicherter Punkt in der Global_Points.dat

```
DEFDAT GLOBAL_POINTS PUBLIC

DECL GLOBAL E6POS gXP1={X 1765.00,Y -2.81284883E-07,Z
1784.00,A 0.0,B 90.0000,C 9.13112519E-09,S 6,T 26,E1
0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL GLOBAL FDAT gFP1={TOOL_NO 1,BASE_NO 0,IPO_FRAME
#BASE,POINT2[] " "}
DECL GLOBAL PDAT gPPDAT1={VEL 100.000,ACC
100.000,APO_DIST 100.000,APO_MODE #CPTP,GEAR_JERK
50.0000,EXAX_IGN 0}

ENDDAT
```

- **Syntax:**

- geteachter Punkt über eine Inlineformular in der **lokalen Datenliste**.

```
xP1 = {X 1700, Y 23, Z 1200, A 0, B 90, C 45}
```

- geteachter Punkt über ein Inlineformular in der **Global_Points.dat**.

```
gXP1 = {X 1700, Y 23, Z 1200, A 0, B 90, C 45}
```



Durch die unterschiedlichen Präfixe **xPn** und **gxPn** ist jeder Punkt auf der Steuerung eindeutig.

7.5.1 Globale Punkte in einer Übersicht anzeigen und ändern

Beschreibung

Globale Punkte können in jedem Programmmodul generiert werden. Die Übersicht "Globale Punkte" ist über das smartPAD aufrufbar und listet alle existierenden globalen Punkte der Steuerung auf. In der Regel die in der **Global_Points.dat** enthaltenen. In der Übersicht können die Punkte bearbeitet werden.

Vorgehensweise

1. Im Hauptmenü **Konfiguration > Globale Punkte** wählen.

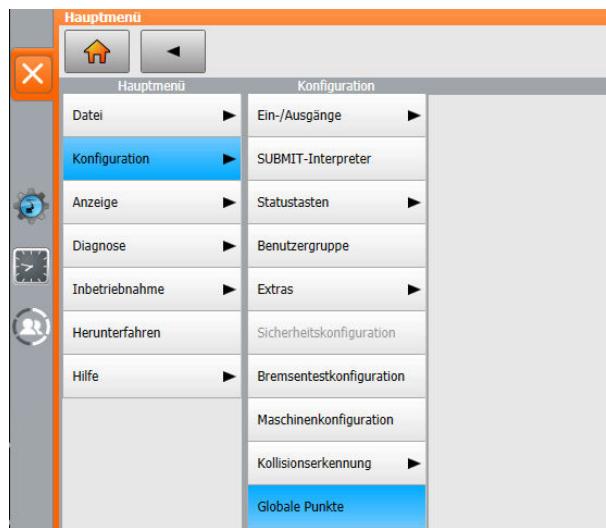


Abb. 7-53: Menüpfad, globale Punkte

2. Das Fenster **Globale Punkte** öffnet sich.



Abb. 7-54: Punktübersicht, Suche

3. Globalen Punkt suchen, nach Bedarf ändern.
4. Auf **Speichern** drücken. Eine Sicherheitsabfrage wird angezeigt.
5. Wenn die Sicherheitsabfrage mit **Ja** beantwortet wird, werden die Änderungen gespeichert.

Fenster Globale Punkte

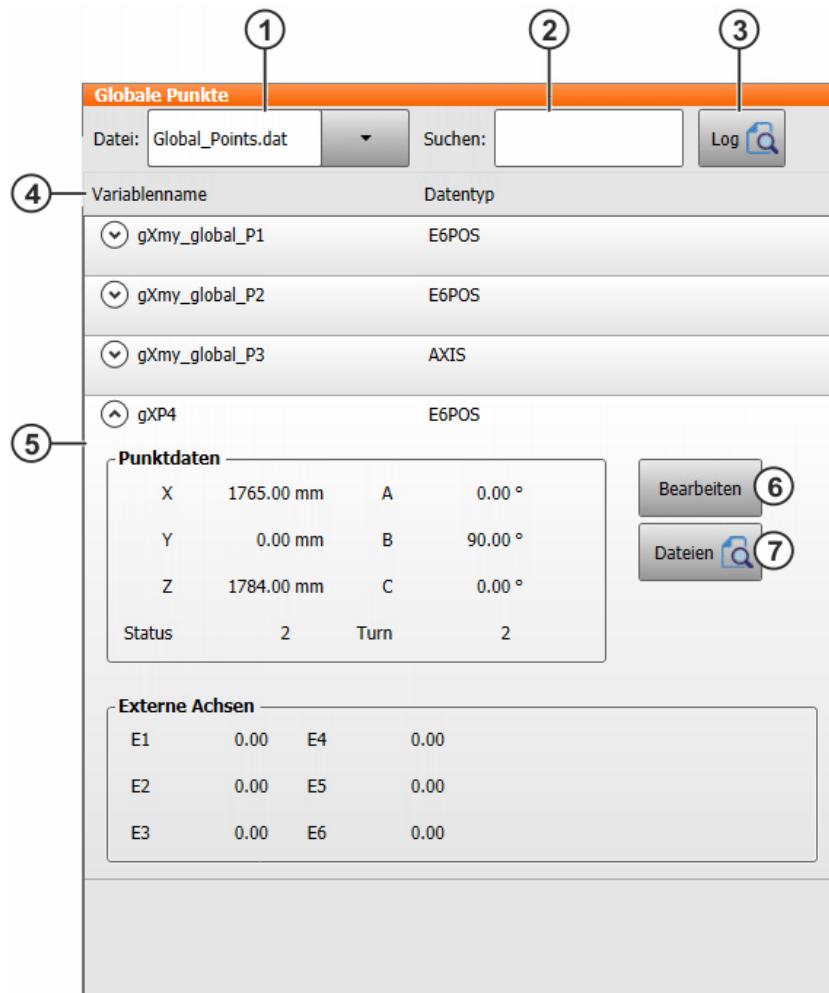


Abb. 7-55: Fenster Globale Punkte

Pos.	Beschreibung
1	Datei, deren globale Punkte angezeigt werden Standardmäßig ist hier nur die Global_Points.dat verfügbar. Weitere Dateien können hinzugefügt werden.
2	Den Namen oder Namensbestandteile eingeben, nach denen die Anzeige gefiltert werden soll
3	Archiviert alle globalen Punkte unter C:\KRC\Roboter\LOG\GlobalPointsLogFile.txt. (Unabhängig davon, welche Datei ggf. unter Pos. 1 ausgewählt ist.)
4	Liste der Punkte mit Angabe des Variablennamens und des Datentyps Listeneinträge können über das Pfeilsymbol aufgeklappt werden.
5	Aufgeklappter Eintrag, zeigt die Punktdaten an
6	Öffnet den Listeneintrag im Bearbeitungsmodus.
7	Zeigt eine Liste aller Dateien an, in denen der jeweilige globale Punkt verwendet wird.

Beispiel: Verwendungsübersicht

Globale Punkte	
Variablenname :	gXStift_Tisch
X	KRC:\R1\Program\REF_P1\ D_Stift_Tisch_ab.src
	KRC:\R1\Program\REF_P1\ D_Stift_Tisch_hol.src

Abb. 7-56: Globale Punkte, Verwendung

7.6 Ändern von Bewegungsbefehlen

Bewegungsbefehle ändern

Es gibt unterschiedlichste Gründe bestehende Bewegungsbefehle zu ändern:

Beispielhafte Gründe	Durchzuführende Änderung
<ul style="list-style-type: none"> Position des zu greifenden Teiles ändert sich. Die Position einer von fünf Bohrungen bei der Bearbeitung ändert sich. Eine Schweißnaht muss verkürzt werden. 	<ul style="list-style-type: none"> Änderung der Positionsdaten
<ul style="list-style-type: none"> Lage der Palette ändert sich. 	<ul style="list-style-type: none"> Änderung der Frame-Daten: Base
<ul style="list-style-type: none"> Eine Position wurde versehentlich mit der falschen Base oder dem falschen TOOL geteacht. 	<ul style="list-style-type: none"> Änderung der Frame-Daten: Base und/oder Tool mit Aktualisierung der Position
<ul style="list-style-type: none"> Die Bearbeitung läuft zu langsam: die Taktzeit muss verbessert werden. 	<ul style="list-style-type: none"> Änderung der Bewegungsdaten: Geschwindigkeit, Beschleunigung Änderung der Bewegungsart

Effekte bei Änderung der Positionsdaten

Nur der Datensatz des Punkts wird geändert: der Punkt erhält neue Koordinaten, da mit "Touch-Up" die Werte aktualisiert wurden.

HINWEIS
Die alten Punktkoordinaten werden überschrieben und sind anschließend nicht mehr verfügbar!

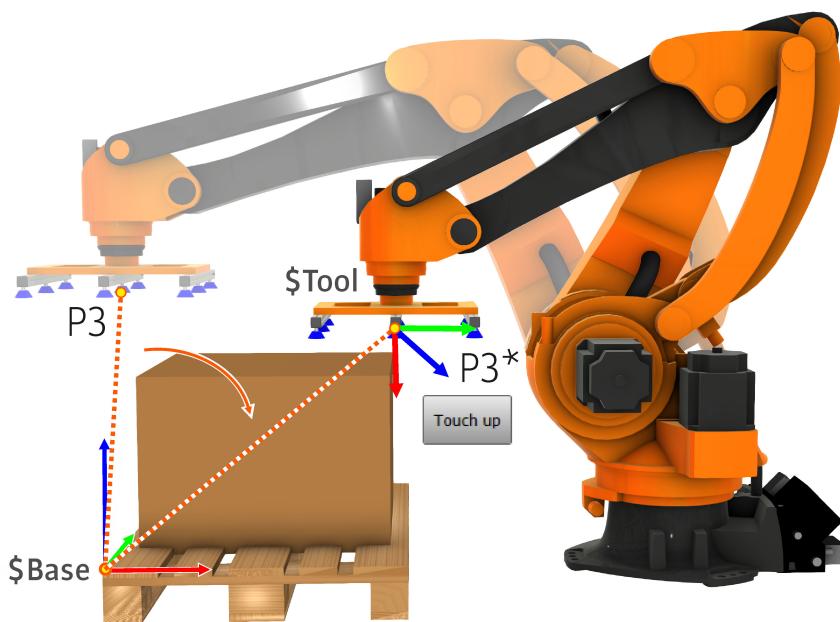


Abb. 7-57: Änderung der Roboterposition mit "Touchup"

Vorgehensweise

1. Betriebsart T1 einstellen und Cursor in die Zeile mit der Anweisung setzen, die geändert werden soll.
2. Roboter in die neue Position bringen.
3. **Ändern** drücken.



Abb. 7-58: Inlineformular, Schaltflächen

Das Inline-Formular zur Anweisung öffnet sich.

4. Für SPTP- und SLIN-Bewegungen:



Abb. 7-59: Inlineformular bearbeiten

- **Touchup** drücken, um die aktuelle Position des TCP als neuen Punkt aufzunehmen.

Für SCIRC-Bewegungen:

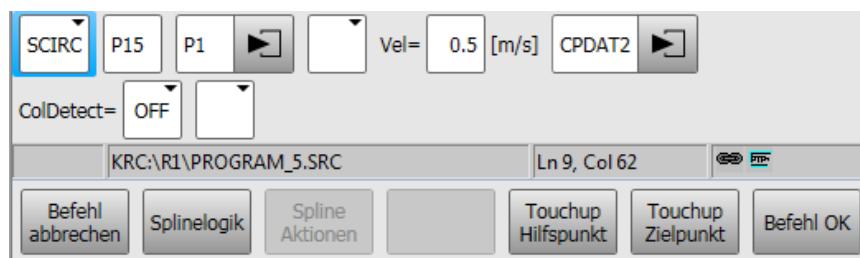


Abb. 7-60: SCIRC, Schaltflächen

- **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. **Dialog:** "Wollen Sie den Punkt "XPnn" wirklich aufnehmen?"

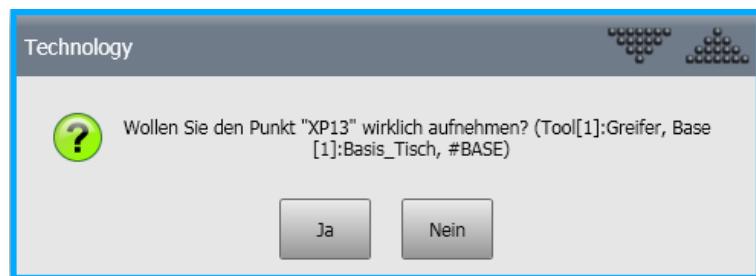


Abb. 7-61: Meldung

Diesen mit **Ja** bestätigen.

6. Änderung mit **Befehl OK** speichern.
7. Punkt im Programm auf Kollisionsfreiheit testen.

Effekte bei Änderung der Framedaten

Werden Frame-Daten (z. B. Tool, Base) geändert, kommt es zu einer Verschiebung der Position (vgl. "Vektorverschiebung"). Es ändert sich die Roboterstellung! (entspricht der Achswinkelstellung des Roboters). Die alten Koordinaten des Punkts sind nach wie vor gespeichert und gültig. Es ändert sich nur der Bezug (z. B. die Basis). Es kann eine Überschreitung des Arbeitsbereichs auftreten! Somit sind bestimmte Roboterstellungen nicht erreichbar. Soll die Roboterstellung identisch bleiben, aber dennoch die Frame-Parameter geändert werden, so muss nach Änderung der Parameter (z. B. Base) in der gewünschten Stellung die Koordinate mit "Touch-Up" aktualisiert werden!

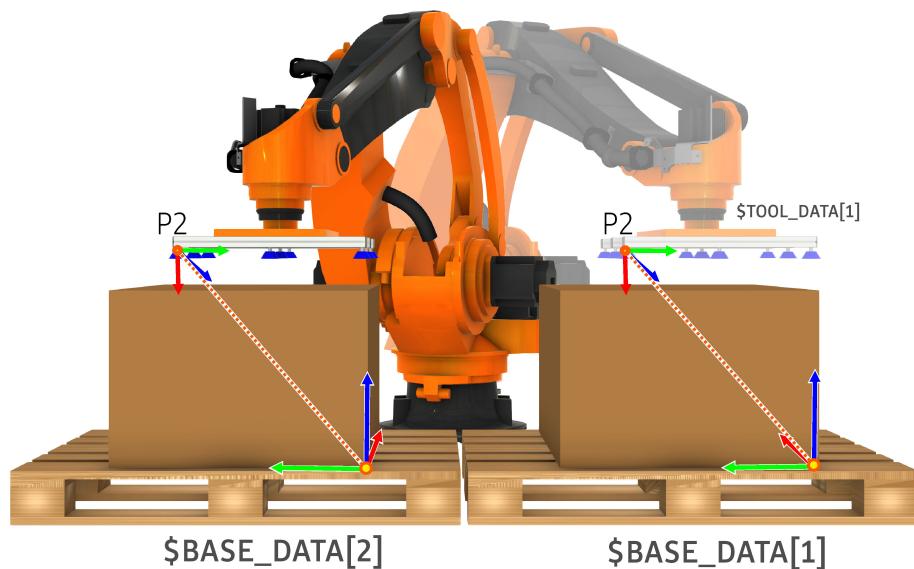


Abb. 7-62: Änderung Framedaten (Beispiel Base)



Ein Benutzerdialog warnt zusätzlich: "Achtung beim Ändern punktbezogener Frameparameter besteht Kollisionsgefahr!".

Vorgehensweise

1. Cursor in die Zeile mit der Anweisung setzen, die geändert werden soll.
2. **Ändern** drücken.



Abb. 7-63: Inlineformular, Schaltflächen

Das Inline-Formular zur Anweisung öffnet sich.

3. Optionsfenster Frames öffnen.
4. Neue Tool oder Base oder Externer TCP einstellen.

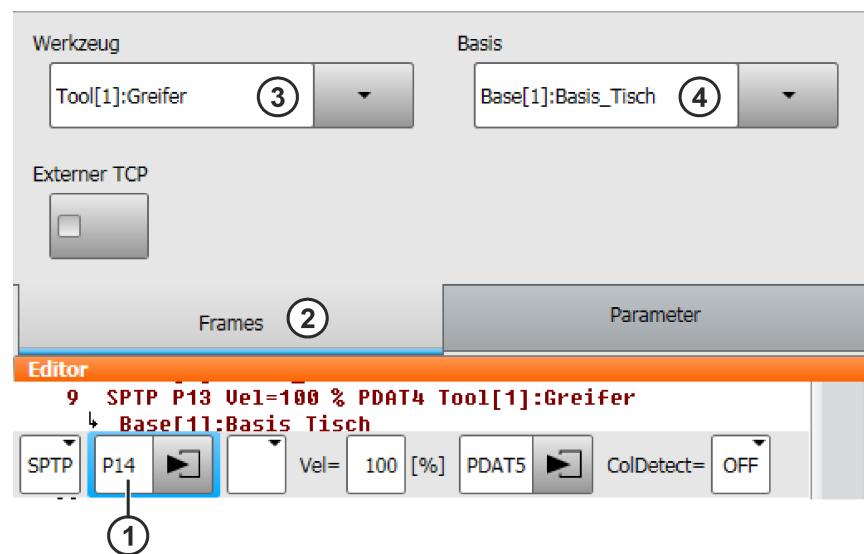


Abb. 7-64: Basis und Werkzeug setzen

5. Benutzerdialog "Achtung beim Ändern der punktbezogenen Frameparameter besteht Kollisionsgefahr!" mit **OK** bestätigen.

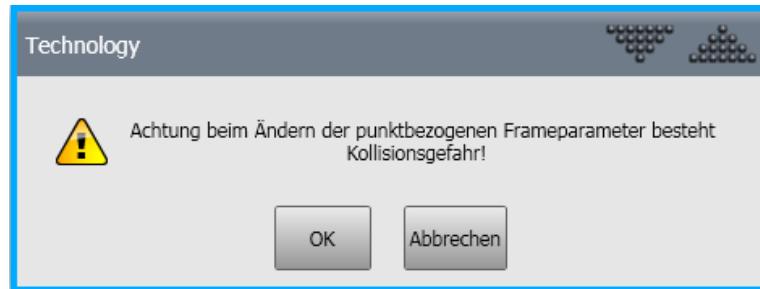


Abb. 7-65: Meldung, Kollisionsgefahr

6. Möchte man die **aktuelle Roboterposition** mit geänderten Tool- und/oder Baseeinstellungen **beibehalten**, muss zwingend **Touch Up** gedrückt werden, um die aktuelle Position neu zu berechnen und zu speichern. Die zu erhaltende Position muss vorher angefahren worden sein.
7. Änderungen mit **Befehl OK** speichern.



Bei Änderung von Frameparametern müssen die Programme erneut auf Kollisionsfreiheit getestet werden.

Effekte bei Ändern von Bewegungsdaten

Bei Änderung der Geschwindigkeit oder der Beschleunigung ändert sich das Fahrprofil. Dies kann Auswirkungen auf den Bearbeitungsprozess haben, gerade bei Bahnapplikationen:

- Dicke einer Kleberaupe
- Güte einer Schweißnaht

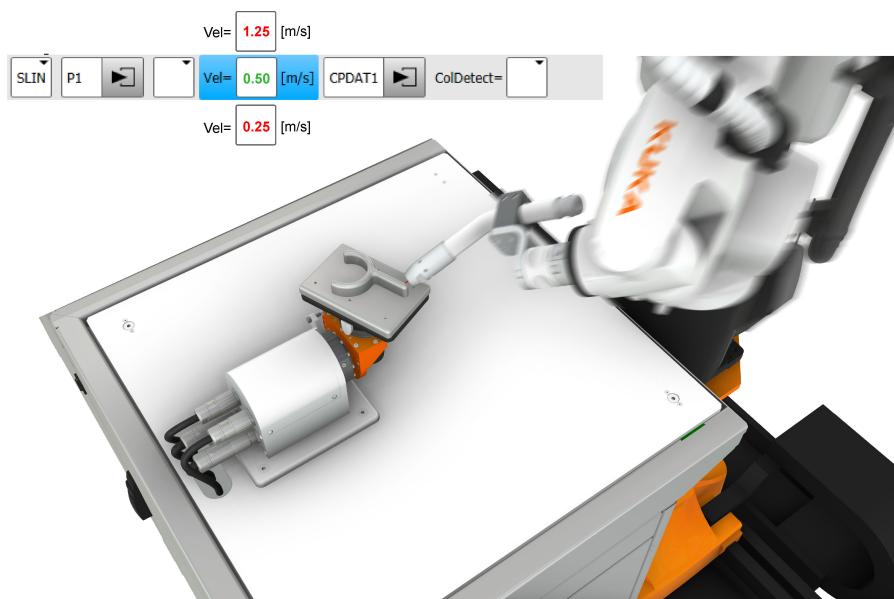


Abb. 7-66: Bewegungsdaten ändern (Beispiel Geschwindigkeit)

Effekte bei Ändern der Bewegungsart

Änderung der Bewegungsart führt immer zu einer Änderung der Bahnplanung! Dies kann in ungünstigen Fällen zu Kollisionen führen, da sich die Bahn unvorhersehbar ändern kann.

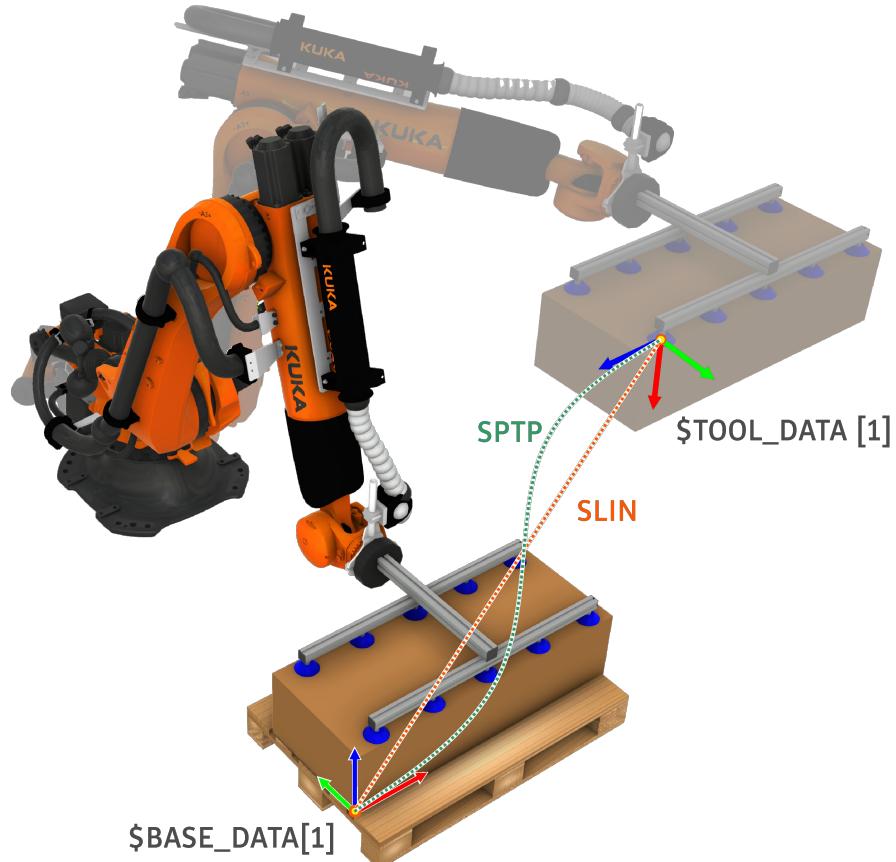


Abb. 7-67: Bewegungsart ändern

Vorgehensweise

1. Cursor in die Zeile mit der Anweisung setzen, die geändert werden soll.

2. Ändern drücken.



Abb. 7-68: Inlineformular, Schaltflächen

Das Inline-Formular zur Anweisung öffnet sich.

3. Parameter ändern.
4. Änderungen mit **Befehl OK** speichern.



Bei Änderung von Bewegungsparametern müssen die Programme erneut auf Kollisionsfreiheit und Prozesssicherheit geprüft werden.

Diese Vorgehensweise kann für folgende Änderungen eingesetzt werden:

- Bewegungsart
- Geschwindigkeit
- Beschleunigung
- Überschleifen
- Überschleifdistanz

Sicherheitshinweise zum Ändern von Bewegungsbefehlen

HINWEIS
<ul style="list-style-type: none"> • Nach jeder Änderung von Bewegungsbefehlen muss das Roboterprogramm bei reduzierter Geschwindigkeit (Betriebsart T1) getestet werden. • Sofortiges Starten des Roboterprogramms bei hoher Geschwindigkeit kann zu Schäden am Robotersystem und an der gesamten Anlage führen, da mit unvorhersehbaren Bewegungen zu rechnen ist. • Sollte sich eine Person im Gefahrenbereich befinden, ist mit lebensgefährlichen Verletzungen zu rechnen.

Umbenennen von Punkten

1. Den zu umbenennenden Punkt mit dem Cursor markieren.
2. Mit der Schaltfläche "Ändern" das Inlineformular öffnen.
3. Den Punktamen im Inlineformular abändern (Feld Punktname).
4. Änderungen mit der Schaltfläche "Befehl OK" übernehmen.
5. Folgender Dialog erscheint: *Vorhergehende Koordinaten für Punkt "Punktname" beibehalten ? (TOOL_DATA[n], BASE_DATA[n], #BASE)*
6. Mit der Schaltfläche "Ja" werden die vorhergehenden Koordinaten des ursprünglichen Punktes übernommen.

Mehrfachverwendung von Punkten

1. Ein neues Inlineformular an der gewünschten Programmposition einfügen.
 2. Den Punktamen des wiederzuverwendenden Punktes im Inlineformular eintragen (Feld Punktname).
- Hier ist auf die exakte Schreibweise des originären Punktamens zu achten.

3. Weitere Einstellungen vornehmen und mit der Schaltfläche "Befehl OK" übernehmen
4. Folgender Dialog erscheint: *Punkt "Punktname" bereits vorhanden - überschreiben ? (TOOL_DATA[n], BASE_DATA[n], #BASE)*
5. Mit der Schaltfläche "Nein" werden die bereits vorhandenen Punktparameter des ursprünglichen Punktes nicht überschrieben.

Einfügen von Programmzeilen

1. Eine vorhandene Programmzeile mit dem Cursor markieren.
2. Mit der Schaltfläche Bewegung ein neues Inlineformular öffnen.



Das neue Inlineformular wird immer nach der markierten Programmzeile eingefügt. Nachfolgende Programmzeilen werden um eine Zeile nach hinten verschoben.

Löschen von Programmzeilen

1. zu lösчende Programmzeile mit dem Cursor markieren
2. **Menüpfad:** Schaltfläche Bearbeiten > Löschen



Im dazugehörigen .dat File bleiben die punktbezogenen Werte erhalten.

8 Programmieren von Spline-Bewegungen

8.1 Lerneinheit: Programmieren von Spline-Bewegungen

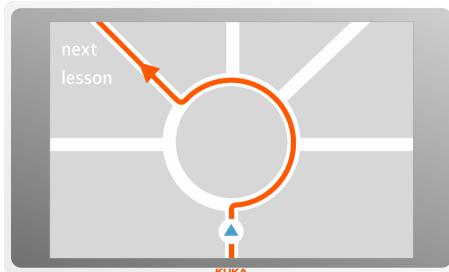


Abb. 8-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Was ist SPLINE?
- Beschreibung des SPLINE-Blocks
- SPLINE-Block mit Bewegung programmieren
- Programmierhinweise
- Überschleifen von SPLINE-Bewegungen

8.2 Was ist ein SPLINE ?

Beschreibung

Programmieren mit Einzelbewegungen

- Ein Roboterprogramm besteht im ersten Schritt aus einzelnen geteachten oder berechneten Punkten.

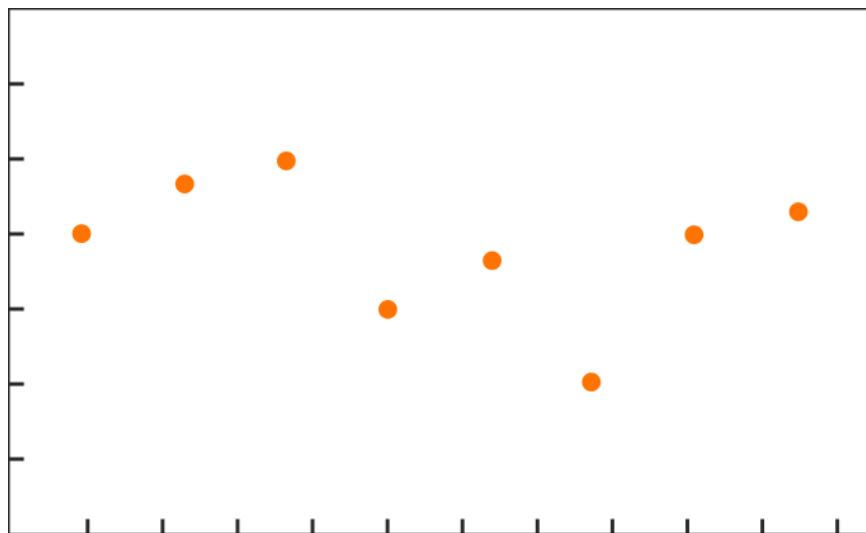


Abb. 8-2: Ausgangssituation Positionen

- Dem Roboter wird durch ein Inlineformular oder durch die KRL-Programmierung zusätzlich mitgeteilt, wie diese Punkte miteinander verbunden werden sollen.
- **Beispiel:** Möglicher Bewegungssatz ist hier (S)LIN. » Die Punkte werden durch eine Strecke miteinander verbunden.

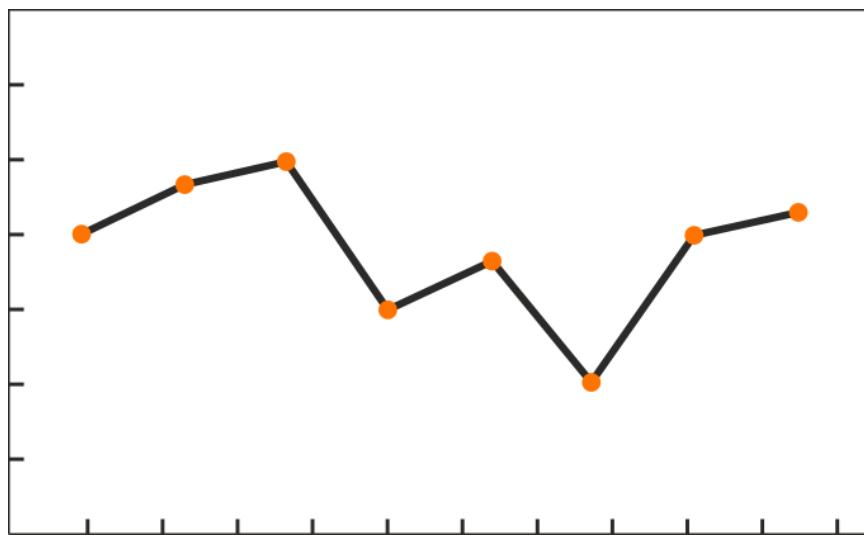


Abb. 8-3: Linerare Interpolation



Nachteil: Der Roboter stoppt in jedem einzelnen Punkt und beschleunigt wieder neu.

- Um den Genauhalt zu vermeiden, können die Punkte mit dem Parameter CONT überschliffen werden.

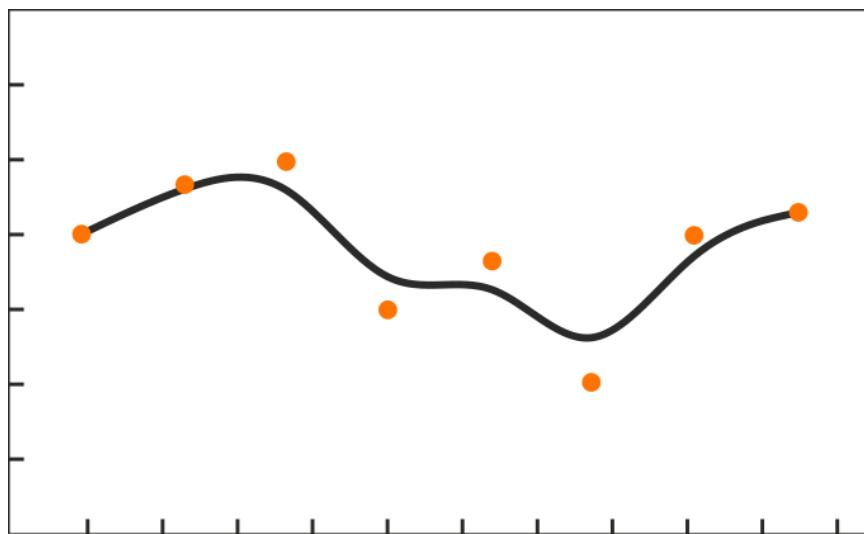


Abb. 8-4: B-SPLINE-Interpolation



Nachteil: Der Roboter überschleift die geteachten/ berechneten Punkte.
Nur der Anfangs- und Endpunkt liegen auf der Bahn.
Vorteil: Der Roboter fährt die gesamte Bahn mit einer flüssigen Bewegung ab.



Genaugenommen handelt es sich hier bereits um eine Sonderform des SPLINE.

Was ist SPLINE?

- Straklatte**
 - Die Bezeichnung "Spline" geht auf das Aufspannen von Holzplatten des "Strakens" zurück, mit der früher im Schiff- und Flugzeugbau glatte Spantkonturen bestimmt wurden.

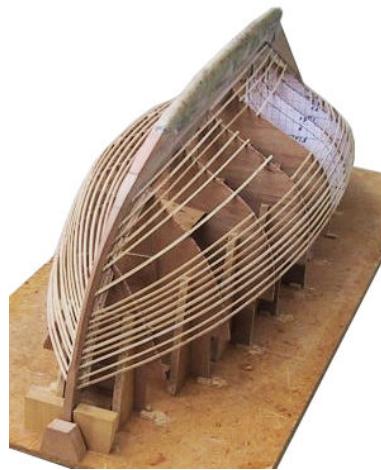


Abb. 8-5: Schiffsrumpf, Plankenbauweise

- Bei dieser Methode wird die Straklatte (elastisches Lineal; engl.: spline) durch Strakgewichte, die an den Interpolationspunkten (im Bild Nägel/Stifte) aufgestellt wurden, gezwungen, die Interpolationsbedingungen (Kontur) zu erfüllen.

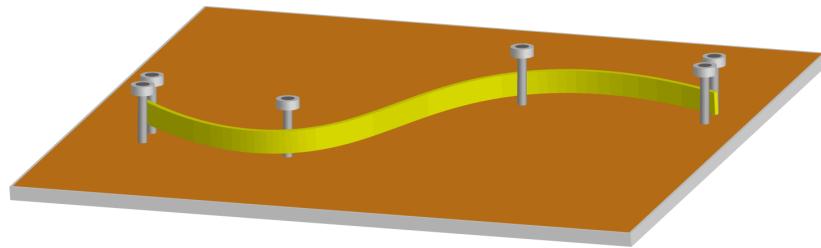


Abb. 8-6: Straklatte

- Dann entsteht aufgrund der Elastizität des Materials eine Biegelinie kleinster Gesamtkrümmung.
- **Stützpunkt verschoben**

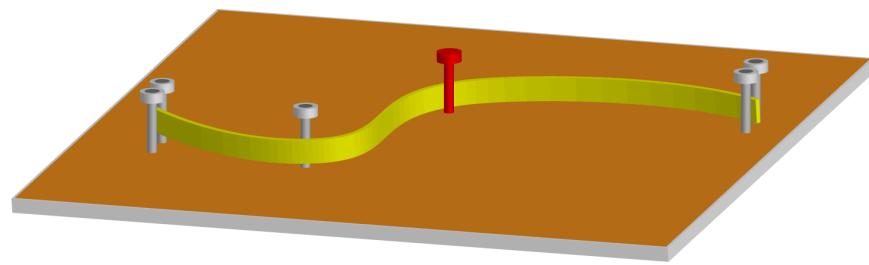


Abb. 8-7: Verschieben eines Stützpunktes

- Das Verschieben eines Stützpunktes (roter Nagel/Stift) hat Einfluss auf die Gesamtbahn der Straklatte.
- Durch mehrere Stützpunkte kann die Bahn präzisiert werden.

Mathematischer Hintergrund

- **Polynominterpolation**

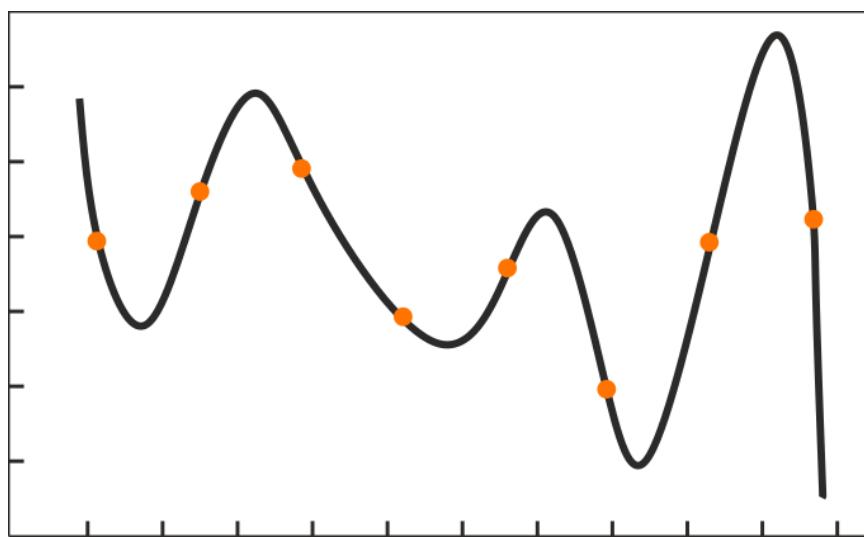


Abb. 8-8: Polynominterpolation

- Polynominterpolation, Funktion n-ten Grades durch die Punkte
- Wird z. B. eine polynomiale Funktion höheren Grades als Interpolationsfunktion verwendet (siehe Bild), so kann es zu **recht extremen Oszillationen** der Kurve kommen.
- Aus 8 gegebenen Positionen wird ein Polynom 7.Grades erstellt ($ax^7+bx^6+\dots$)



In Konstruktionsanwendungen wie dem Schiffbau oder der Flugzeugindustrie sind solche starken Oszillationen allerdings höchst unerwünscht, hier werden nur leicht gebogene, anschmiegsame Kurven zwischen den einzelnen Stützstellen benötigt (man stelle sich einen Schiffsrumph im Sägezahnformat vor...).

- **SPLINE-Interpolation**

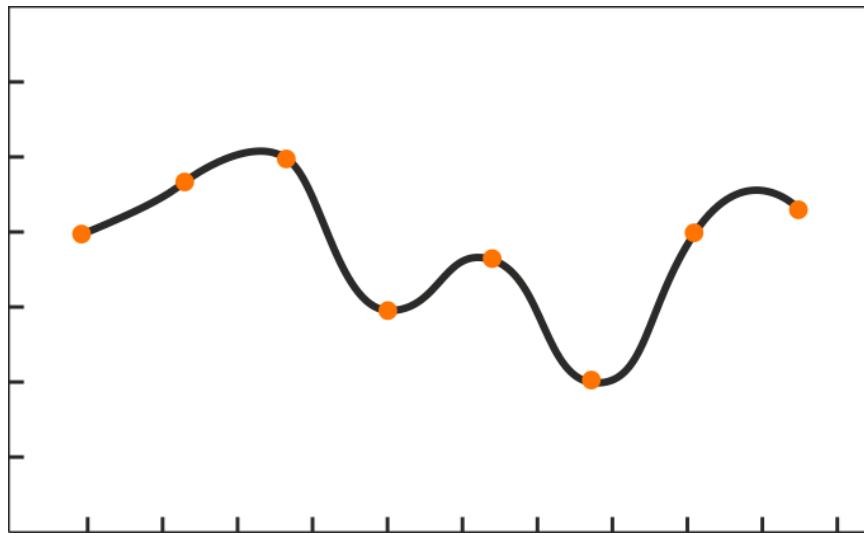


Abb. 8-9: SPLINE-Interpolation

- Splines versuchen nicht eine Funktion über alle Stützstellen zu interpolieren, sondern nur lokal über jeweils zwei Stützstellen.
- Diese Splines wirken glättend und haben nur eine sehr geringe Welligkeit. Sie können Konturen damit wirklichkeitsgetreuer abbilden.

- Am häufigsten werden hierbei lineare oder kubische Splines verwendet, die durch eine lineare Funktion bzw. ein Polynom 3. oder 4. Grades (**KUKA**) interpoliert werden.



Siehe auch (>>> **Abb. 8-4**)

8.3 Beschreibung des SPLINE-Blocks

Beschreibung

Zusätzlich zu den Einzelsätzen mit SPTP, SLIN, SCRIC steht ein "Spline-Block" zur Verfügung. Der Spline-Block ist eine Bewegungsart, die besonders für komplexe geschwungene Bahnen geeignet ist. Der Spline-Block wird als einzelne Bewegung mit einer "komplexen Bahn" gesehen und geplant.



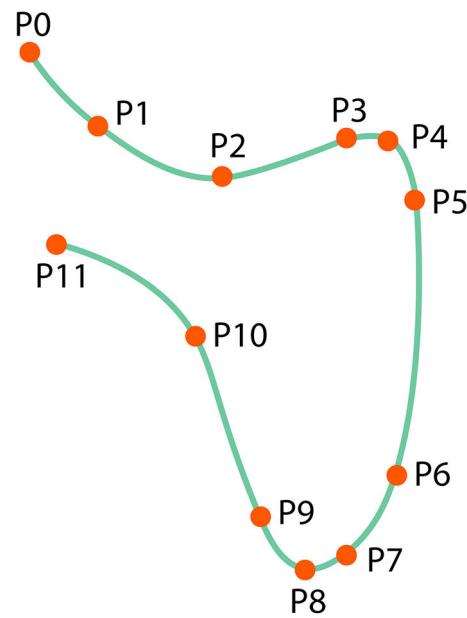
Abb. 8-10

Es gibt zwei Arten von Spline-Blöcken:

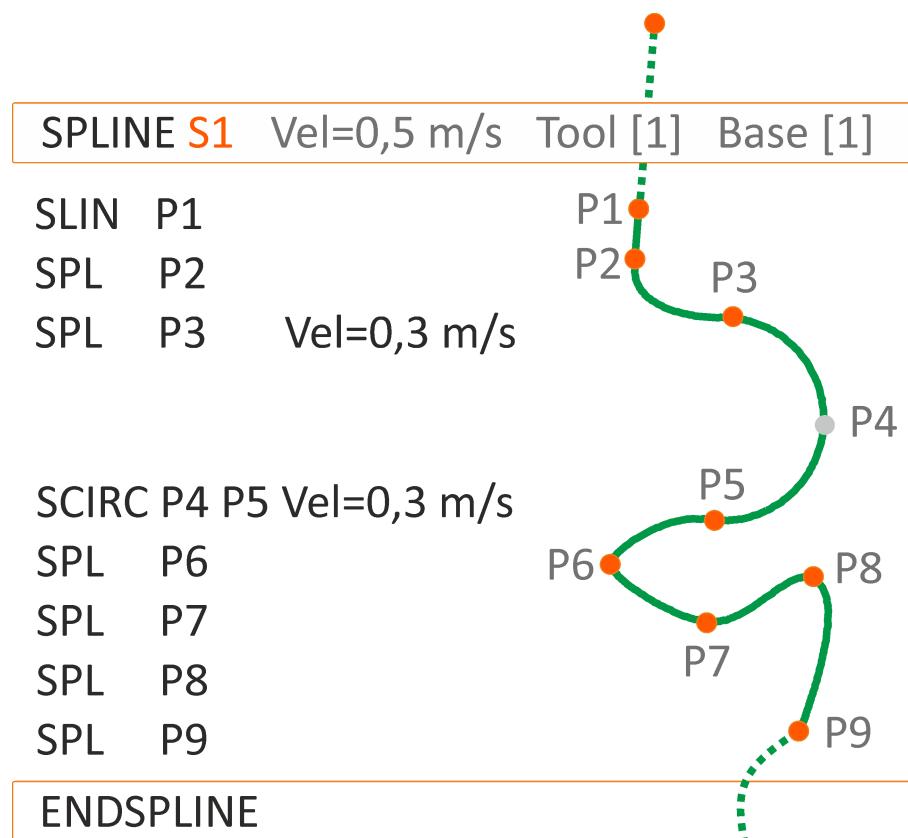
- **Spline-Block**
Spline mit Bahnbewegungen (SPL, SLIN, SCIRC)
- **PTP-SPLINE-Block**
Spline mit Bewegungen ausschließlich im Achsraum (nur SPTP)

Die Bahn wird durch alle Punkte durchgeplant, somit werden alle Punkte durchfahren. Die Bahn des Spline-Blocks wird komplett im Voraus berechnet.

Spline-Bahnen, die sehr enge Konturen haben, werden immer das Herabsetzen der Geschwindigkeit mit sich ziehen, da die Roboterachsen die begrenzenden Elemente sind.

Spline-Bahn**Abb. 8-11: geschwungene Bahn, SPLINE-Block**

Die Bahn wird definiert über Stützpunkte, durch die die Bahn gelegt wird. Der Roboter berechnet die Bahn aus den Stützpunkten. In 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.

Aufbau eines Spline-Blocks**Abb. 8-12: Spline-Block**

- Ein Spline-Block ist ein Fold (Falte) welche von der Steuerung als eine Bewegung interpoliert wird.
- In dieser Falte werden Stützpunkte geteacht, die als Spline-Bewegung interpoliert werden.
- Zusätzlich kann der Roboter in den einzelnen Spline-Segmenten auf eine lineare oder zirkuläre Bahn gezwungen werden. Dies geschieht durch die Wahl der Bewegungsart im Segment hin zum Stützpunkt.
- Hieraus ergeben sich folgende Bewegungsarten für die einzelnen Spline-Segmente:
 - SLIN
 - SCIRC
 - SPL
- Für den gesamten Spline-Block wird in der "Kopfzeile" der Name der Bewegung als auch die globalen Parameter festgelegt.
 - Werkzeug, Basis
 - Geschwindigkeit
 - Überschleifen
 - Bewegungsparameter
 - Kollisionserkennung an/aus
- In jeder einzeln geteachten Bewegungen kann davon abgewichen werden.



Ausnahme: Werkzeug und Basis werden über die "Kopfzeile" festgelegt.

- Außer den Bewegungssegmenten darf ein Spline-Block folgende Elemente enthalten:
 - Inline-Befehle aus Technologiepaketen, die über die Spline-Funktionalität verfügen
 - Kommentare und Leerzeilen
- Splinelogik kann innerhalb des Inline-Formulars eingebunden werden (Trigger, Bedingter Stopp, Konstante Geschwindigkeit). Ein Spline-Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.



Die Punkte vor und nach dem Spline-Block legen die Kontur fest, wie in und aus dem Spline-Block gefahren wird.



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.

8.4 SPLINE-Block mit Bewegung programmieren

1. Spline-Block einfügen

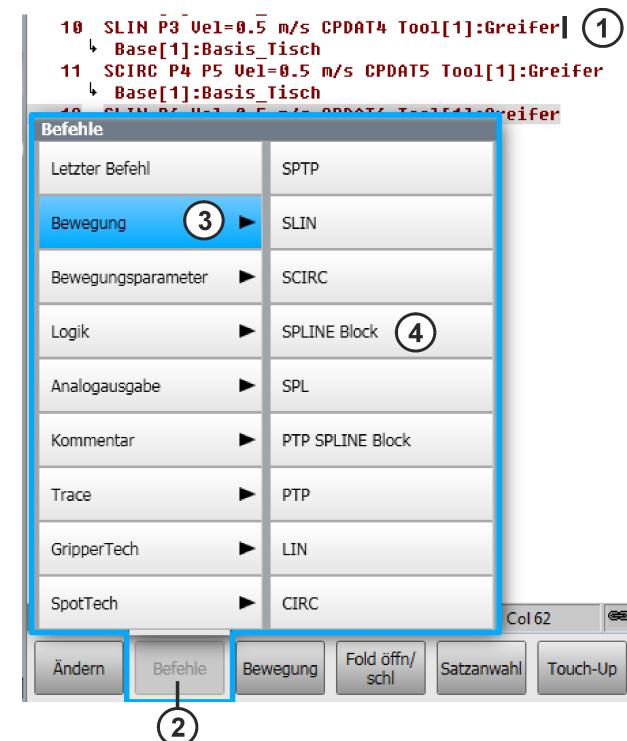


Abb. 8-13: SPLINE-Block einfügen

- Den Cursor auf die Programmzeile (1) setzen, nach der der Spline-Block eingefügt werden soll.
 - Auf die Schaltfläche Befehle (2) tippen.
 - Den Eintrag Bewegung (3) öffnen.
 - Im Kontextmenü Spline-Block (4) auswählen.
2. Der Spline-Block wird eingefügt.

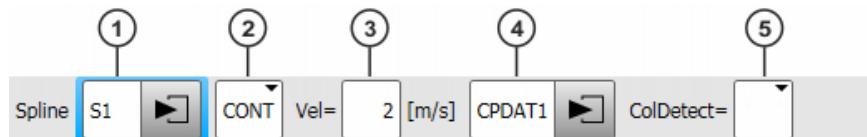


Abb. 8-14: Inline-Formular Spline

- Den Namen des Spline-Blocks (1) festlegen.
- Alle weiteren Parameter entsprechend denen aus der Einzelsatzprogrammierung und gelten übergreifen für den gesamten SPLINE-BLOCK.
(Werkzeug, Basis, Überschleifen, Geschwindigkeit im TCP, Orientierungsführung, Kollisionserkennung, Achsgeschwindigkeit und - Beschleunigung)



In jedem einzeln geteachten Punkt kann individuell von diesen Parametern abgewichen werden.

3. Bei einem Spline-Block handelt es sich programmtechnisch um einen Fold (Falte). In dieser Falte werden die abzufahrenden Punkte gezeigt.

```

    ↴ Base[1]:Basis_Tisch
11 → SPLINE S1 Vel=0.5 m/s CPDAT7 Tool[1]:Greifer
    ↴ Base[1]:Basis_Tisch|—①
12 SCIRC P4 P5 Vel=0.5 m/s CPDAT5 Tool[1]:Greifer
    ↴ Base[1]:Basis_Tisch

```

KRC\R1\PROGRAM_5.SRC Ln 11, Col 64

Ändern Befehle Bewegung Fold öffn/schl Satzanwahl Touch-Up Bearbeiten

Abb. 8-15: Fold öffnen

- Wird ein Spline-Block neu eingefügt, ist der Fold bereits geöffnet.
- Nach längerer Inaktivität schließt sich die Falte und muss neu geöffnet werden.
- Hierzu den Spline-Block markieren (1) und mittels der gleichnamigen Schaltfläche Fold öffnen/schließen (2).

4. Den Roboter an den neu aufzunehmenden Punkt fahren.

```

    ↴ Base[1]:Basis_Tisch|—①
11 → SPLINE S1 Vel=0.5 m/s CPDAT7 Tool[1]:Greifer
    ↴ Base[1]:Basis_Tisch
12 ENDSPINE
13 SCIRC P4 P5 Vel=0.5 m/s CPDAT5 Tool[1]:Greifer
    ↴ Base[1]:Basis_Tisch
14 SLIN P6 Vel=0.5 m/s CPDAT6 Tool[1]:Greifer
    ↴ Base[1]:Basis_Tisch

```

KRC\R1\PROGRAM_5.SRC Ln 11, Col 64

Ändern Befehle Bewegung Fold öffn/schl Satzanwahl Touch-Up Bearbeiten

Abb. 8-16: Punkt aufnehmen

- Den Fokus innerhalb des Spline-Blocks (1) legen.
Bei bereits programmierten Punkten die Stelle, an der der neue Punkt eingefügt werden soll.
- Mittels der Schaltfläche Bewegung (2) einen neuen Punkt in den Spline-Block einfügen.

5. Bewegung einfügen

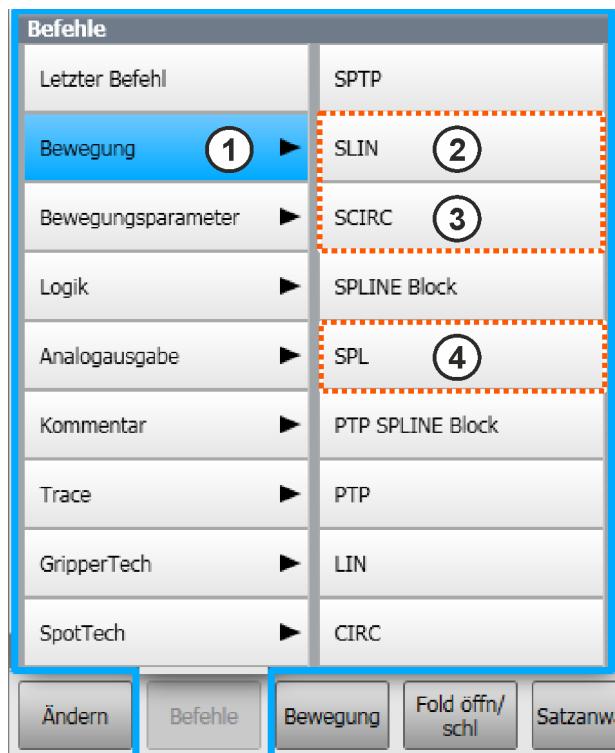


Abb. 8-17: Bewegungen

- **Menüpfad:** Befehle > Bewegung > SLIN, SCIRC oder SPL

Alternativ:

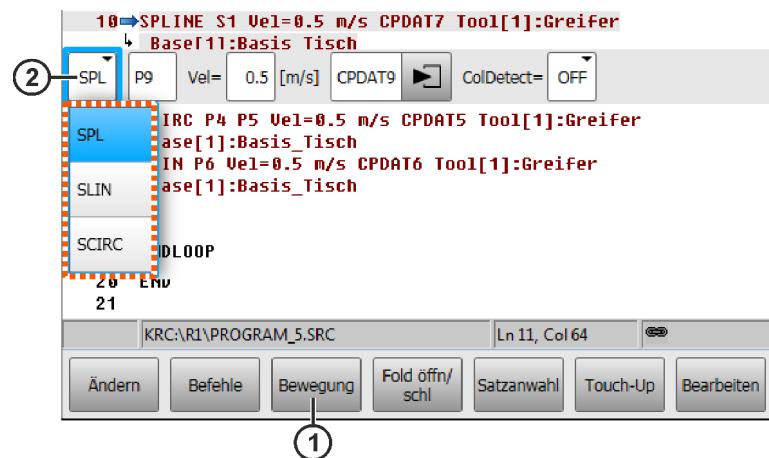


Abb. 8-18: Bewegung einfügen

- Im geöffneten Fold über die gleichnamige Schaltfläche **Bewegung** (1) einfügen.
- Die gewünschte Bewegung über das Pulldown-Fenster (2) ändern.

8.4.1 SLIN im SPLINE-Block

Beschreibung

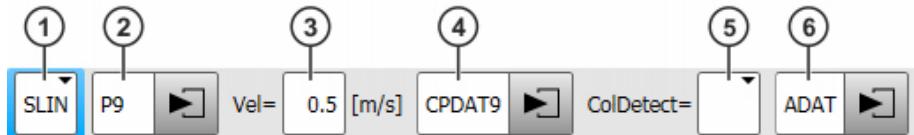


Abb. 8-19: Inline-Formular SLIN-Segment

Pos.	Beschreibung
1	Bewegungsart • SPL, SLIN oder SCIRC
2	Bewegungspunkt • Der vom System automatisch vergebene Name kann geändert werden. • Die geteachten Punktkoordinaten werden in diesem Punkt gespeichert.
3	Kartesische Geschwindigkeit • Standardmäßig gilt für die Bewegung der für den Spline-Block gültige globale Wert. • Der eingegebene Wert gilt für diese Bewegung • 0.001 ... 2 m/s
4	Bewegungsparameter • Der vom System automatisch vergebene Name kann geändert werden. • Standardmäßig gilt für die Bewegung der für den Spline-Block gültige globale Wert. • Der eingegebene Wert gilt für diese Bewegung • Zum Bearbeiten der Daten Pfeil berühren. Das Fenster Bewertungsparameter öffnet sich: (>> 8.4.4 "Bewegungsparameter" Seite 258)
5	Kollisionserkennung für diese Bewegung • OFF: Die Kollisionserkennung ist ausgeschaltet • CDSet_Set[Nr.]: Die Kollisionserkennung ist eingeschaltet. Für die Erkennung werden die Werte aus dem Datensatz-Nr. verwendet.
6	Logikparameter • Der vom System automatisch vergebene Name kann geändert werden. • Der eingegebene Wert gilt für diese Bewegung • Zum Bearbeiten der Daten Pfeil berühren. • Das zugehörige Optionsfenster öffnet sich. Logikparameter – Trigger – Bedingter Stopp – Konstante Geschwindigkeit

Parameter wechseln

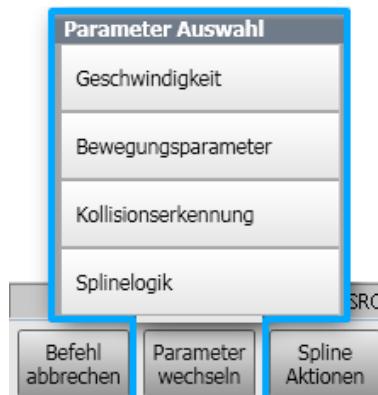


Abb. 8-20: Parameter wechseln

- Standardmäßig werden nicht alle Felder des Inline-Formulars angezeigt.
- Über die Schaltfläche Parameter wechseln lassen sich bei geöffnetem Inline-Formular zusätzliche Parameter ein- und wieder ausblenden.

8.4.2 SCIRC im SPLINE-Block

Beschreibung

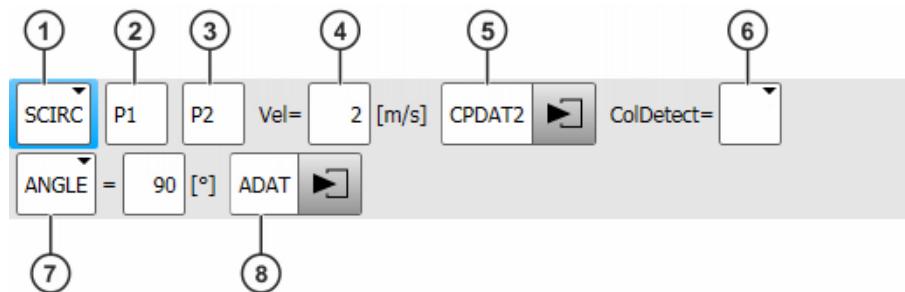


Abb. 8-21: Inline-Formular SCIRC-Spline-Segment

Pos.	Beschreibung
1	Bewegungsart • SPL, SLIN oder SCIRC
2	Hilfs-Punkt • Der vom System automatisch vergebene Name kann geändert werden. • Die geteachten Punktkoordinaten werden in diesem Punkt gespeichert.
3	Ziel-Punkt • Der vom System automatisch vergebene Name kann geändert werden. • Die geteachten Punktkoordinaten werden in diesem Punkt gespeichert.

Pos.	Beschreibung
4	<p>Kartesische Geschwindigkeit</p> <ul style="list-style-type: none"> Defaultmäßig gilt für die Bewegung der für den Spline-Block gültige globale Wert. Der eingegebene Wert gilt für diese Bewegung 0.001 ... 2 m/s
5	<p>Bewegungsparameter</p> <ul style="list-style-type: none"> Der vom System automatisch vergebene Name kann geändert werden. Defaultmäßig gilt für die Bewegung der für den Spline-Block gültige globale Wert. Der eingegebene Wert gilt für diese Bewegung Zum Bearbeiten der Daten Pfeil berühren. Das Fenster Bewegungsparameter öffnet sich: (>>> 8.4.4 "Bewegungsparameter" Seite 258)
6	<p>Kollisionserkennung für diese Bewegung</p> <ul style="list-style-type: none"> OFF: Die Kollisionserkennung ist ausgeschaltet CDSet_Set[Nr.]: Die Kollisionserkennung ist eingeschaltet. Für die Erkennung werden die Werte aus dem Datensatz Nr. verwendet.
7	<p>Kreiswinkel</p> <ul style="list-style-type: none"> Optional aktivierbar, um eine Kreisbahn größer oder kleiner in Bezug auf den Zielpunkt zu fahren. - 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.
8	<p>Logikparameter</p> <ul style="list-style-type: none"> Der vom System automatisch vergebene Name kann geändert werden. Der eingegebene Wert gilt für diese Bewegung Zum Bearbeiten der Daten Pfeil berühren. Das zugehörige Optionsfenster öffnet sich. <p>Logikparameter</p> <ul style="list-style-type: none"> – Trigger – Bedingter Stop – Konstante Geschwindigkeit

8.4.3 SPL im SPLINE-Block

Beschreibung

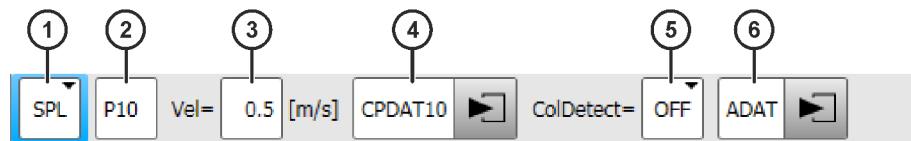


Abb. 8-22: Inline-Formular SPL-Segment

Siehe: **SLIN im SPLINE-Block**

(>>> **8.4.1 "SLIN im SPLINE-Block" Seite 255**)

8.4.4 Bewegungsparameter

Bewegungsparameter

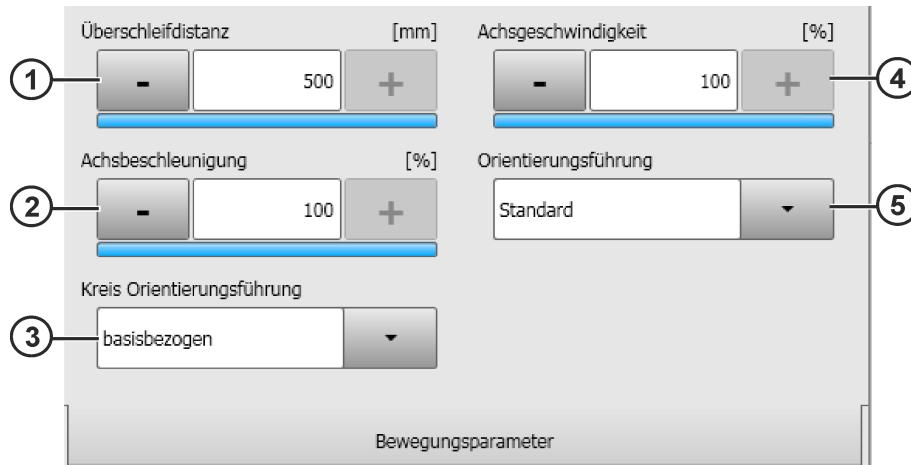


Abb. 8-23: Bewegungsparameter

1	Überschleifdistanz 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 eingegeben wird, wird dieser ignoriert und der Maximalwert verwendet.
2	Achsbeschleunigung Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. <ul style="list-style-type: none">• 1 ... 100 %
3	Kreis Orientierungsführung <ul style="list-style-type: none">• basisbezogen• bahnbezogen

4	Achsgeschwindigkeit. Der Wert bezieht sich auf den in den Maschinendaten angegebenen Maximalwert. <ul style="list-style-type: none">• 1 ... 100 %
5	Orientierungsführung <ul style="list-style-type: none">• Standard• Hand PTP• Konstante Orientierungsführung

8.5 Programmierhinweise

SLIN-SPL-SLIN-Übergang

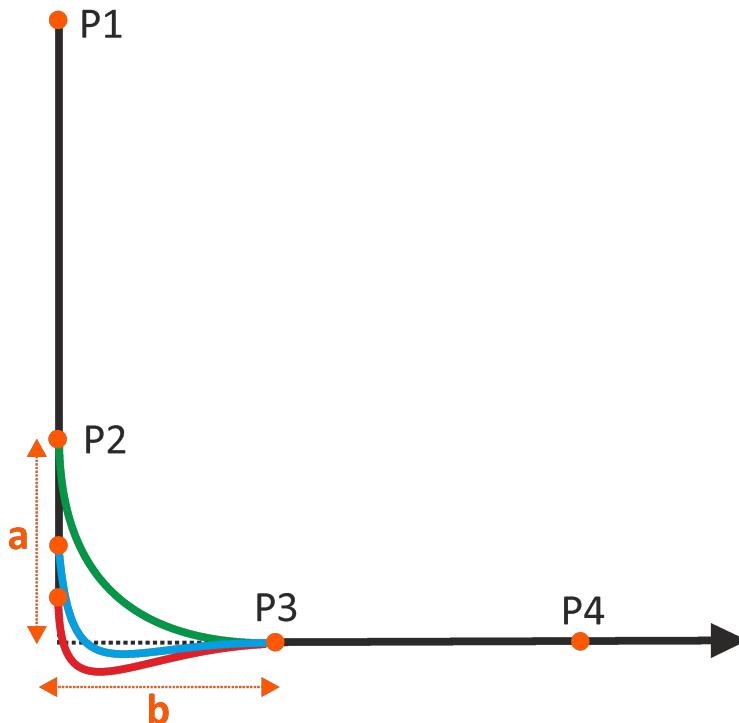


Abb. 8-24: SLIN > SPL > SLIN Übergang

Um eine Kollision bei der Programmierung eines SLIN > SPL > SLIN Übergangs zu vermeiden, ist Folgendes zu beachten:

- Die beiden SLIN-Segmente schneiden sich in ihrer Verlängerung.
 - $\frac{2}{3} \leq a/b \leq \frac{3}{2}$
- a = Abstand vom Startpunkt des SPL-Segments zum Schnittpunkt der SLIN-Segmente
b = Abstand vom Schnittpunkt der SLIN-Segmente zum Zielpunkt des SPL-Segments

8.5.1 Punkte in einem SPLINE-Block verändern

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:**

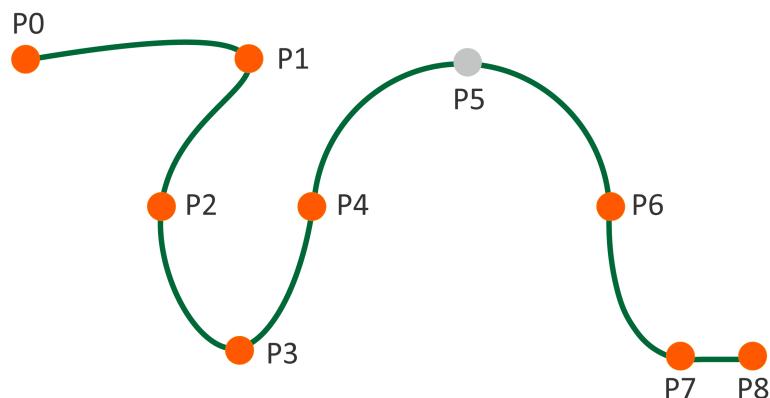


Abb. 8-25: SPLINE, ursprüngliche Bahn

```
SPTP P0
SPLINE S1
    SPL P1
    SPL P2
    SPL P3
    SPL P4
    SCIRC P5, P6
    SPL P7
    SLIN P8
ENDSPLINE
```

- **Gegenüber der ursprünglichen Bahn wird der Punkt 3 verschoben:**

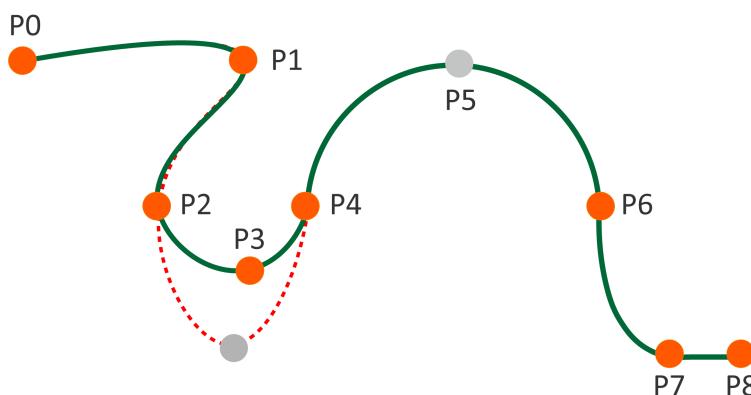


Abb. 8-26: Punkt verschoben

- 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.
- **Gegenüber der ursprünglichen Bahn wird der Bahntyp geändert:**

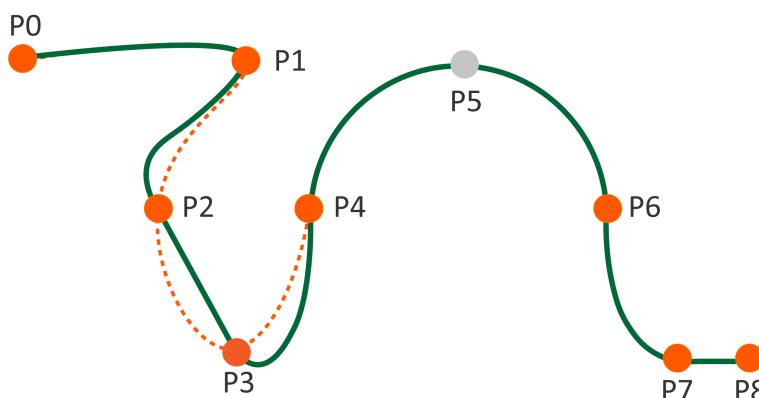


Abb. 8-27: Bahntyp geändert

```

SPTP P0
SPLINE S1
  SPL P1
  SPL P2
  SLIN P3
  SPL P4
  SCIRC P5, P6
  SPL P7
  SLIN P8
ENDSPLINE

```

- 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.

Beispiel 2

In diesem Beispiel wurde eine Strecke ausschließlich mit SPL-Segmenten programmiert.

- Ursprüngliche Bahn:

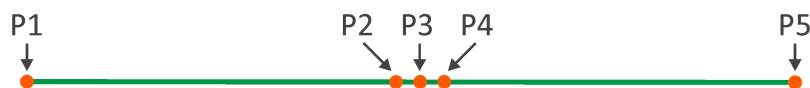


Abb. 8-28: SPL, ursprüngliche Bahn

- Gegenüber der ursprünglichen Bahn wird ein Punkt verschoben:

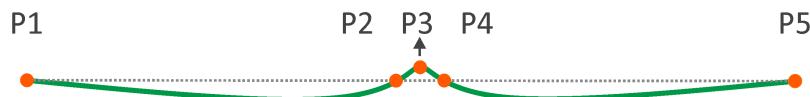


Abb. 8-29: SPL, Punktverschiebung

- 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 eine starke Änderung der Bahn.



- Punktabstände gleichmäßiger verteilen
- Geraden (außer sehr kurze Geraden) als SLIN-Segmente programmieren

8.5.2 Geschwindigkeitsprofil bei Spline-Bewegungen

SPLINE Geschwindigkeit

- Die Bahn verläuft immer gleich, unabhängig von Override, Geschwindigkeit oder Beschleunigung.
- Die Robotersteuerung berücksichtigt bereits bei der Planung die physikalischen Grenzen des Roboters.
- Dies ist ein Vorteil gegenüber den alten 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 ggf. 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.

Absenkung der Geschwindigkeit auf 0

Dies ist der Fall bei:

- Aufeinanderfolgenden Punkten mit gleichen Koordinaten
- Aufeinanderfolgenden **SLIN-** und/oder **SCIRC-Segmenten** innerhalb des Spline-Blocks.

Ursache: Unstetiger Verlauf der Geschwindigkeitsrichtung.

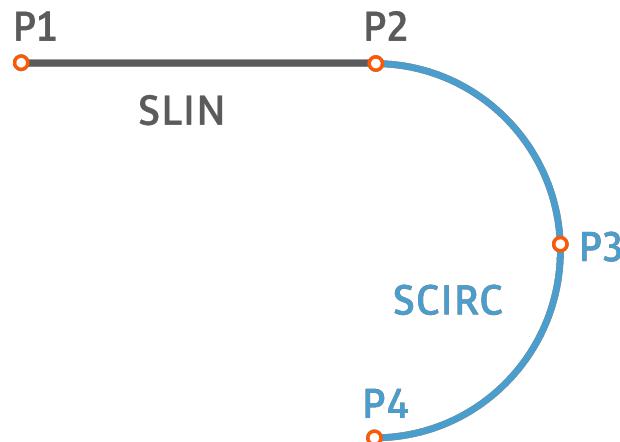


Abb. 8-30: Genauhalt in P2

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 gekrümmmt ist.

- Aufeinanderfolgenden **SLIN-** auf **SLIN-Segmente** innerhalb des Spline-Blocks.

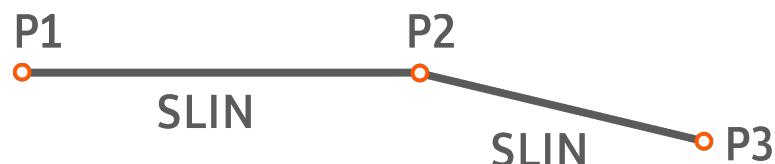


Abb. 8-31: Genauhalt in P2

Abhilfe

Durch das Einfügen eines SPL-Segmentes zwischen SLIN und SCIRC wird ein Genauhalt vermieden und der Roboter kann die Bewegung mit der programmierten Geschwindigkeit an einem Stück abfahren.

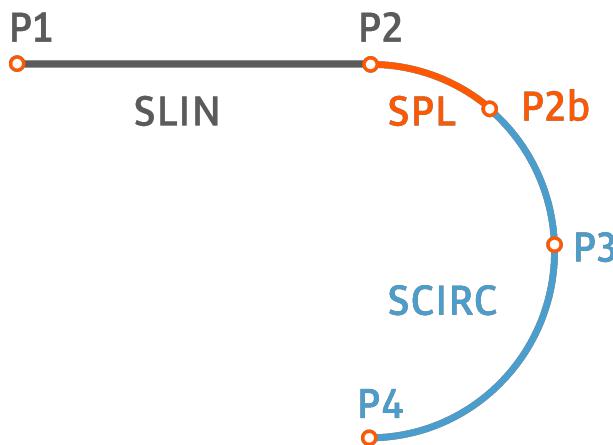


Abb. 8-32: SLIN-SCIRC Übergang mit SPL

Ausnahmen

- Wenn SLIN-Segmente aufeinanderfolgen, die eine Gerade ergeben und bei denen sich die Orientierungen gleichmäßig ändern, wird die Geschwindigkeit nicht reduziert.



Abb. 8-33: P2 wird ohne Genauhalt durchfahren

- Bei einem SCIRC-SCIRC-Übergang wird die Geschwindigkeit nicht reduziert, wenn beide Kreise oder Kreissegmente 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 $\geq 360^\circ$ zu erhalten. Eine einfachere Möglichkeit ist es, einen Kreiswinkel zu programmieren.

Geschwindigkeitsabsenkung durch ungleichmäßiges Teachen

- Bei ungleichmäßiger Verteilung von Orientierungsänderungen, auch in Verbindung mit Zusatzachsen, zur kartesischen Bogenlänge kommt es häufig zu unerwünschten Geschwindigkeitseinbrüchen.
- Bei ungleichmäßiger Verteilung von Orientierungswegen auf den kartesischen Weg (Bogenlänge) muss die Orientierung sehr häufig beschleunigen oder verzögern, was tendenziell auch mit großen Orientierungsrucken einhergeht.
- In der Folge kommt es bei ungleichmäßiger Wegverteilung viel häufiger zu Geschwindigkeitseinbrüchen als bei gleichmäßiger Verteilung der Orientierungswege.

Abhilfe

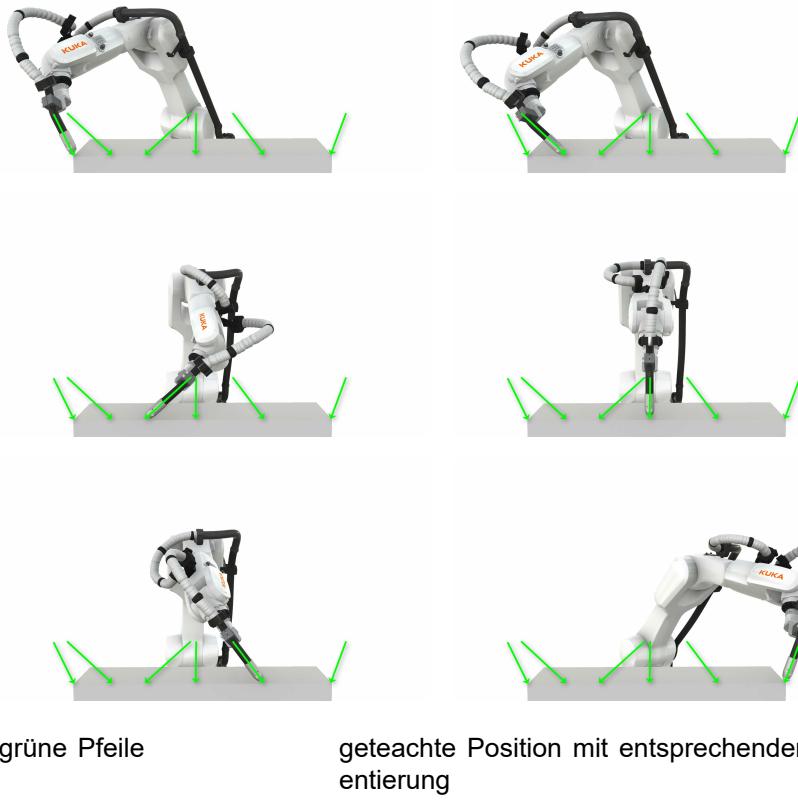


Orientierung und Zusatzachsen möglichst gleichmäßig verteilen

Beispiel einer gleichmäßigen Verteilung:

```
; Startpunkt des Splines
SPTP {x 0, y 0, z 0, A 0, B 0, C 0}
SPLINE S1
; 0,1° Umorientierung pro mm kart. Weg
SPL {x 0, y 100, z 0, A 10, B 0, C 0}
; 0,1° Umorientierung pro mm kart. Weg
SPL {x 0, y 110, z 0, A 11, B 0, C 0}
; 0,1° Umorientierung pro mm kart. Weg
SPL {x 0, y 310, z 0, A 31, B 0, C 0}
ENDSPLINE
```

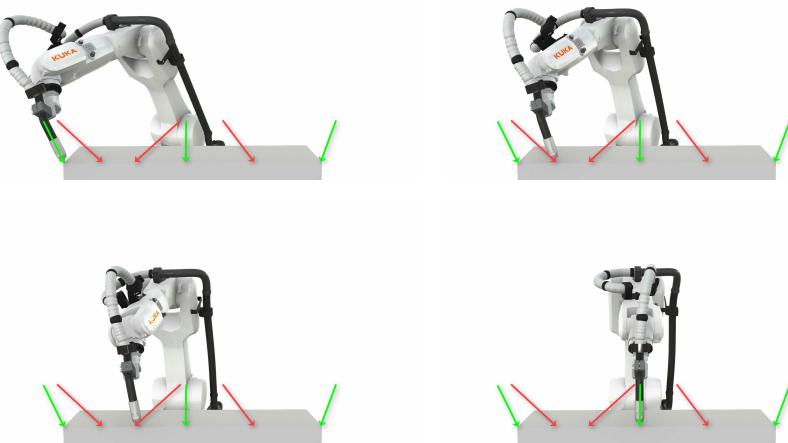
Mit programmierter Orientierung



Fazit

HINWEIS
<ul style="list-style-type: none"> Oft werden viele Punkte mit relativ kleinem Abstand programmiert. Man ist aber hauptsächlich an der kartesischen Bahn (x, y, z) interessiert. Der Spline interpoliert aber auch die programmierte Orientierung, was zu Geschwindigkeitsabsenkungen führen kann. Deswegen ist in diesem Fall bevorzugt im Inline-Formulars IGNORE zu wählen.

Ohne programmierte Orientierung





grüne Pfeile

geteachte Position mit entsprechender Orientierung

rote Pfeile

geteachte Position mit entsprechender Orientierung, deren Orientierung nicht übernommen wird.

8.5.3 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.

- **PTP-Spline-Block:**

Die SAK-Fahrt wird als herkömmliche PTP-Bewegung ausgeführt.

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 worden ist 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 worden ist, ist der Startpunkt nicht bekannt.

Beispiel: Veränderte Bahn bei Satzanwahl auf P1

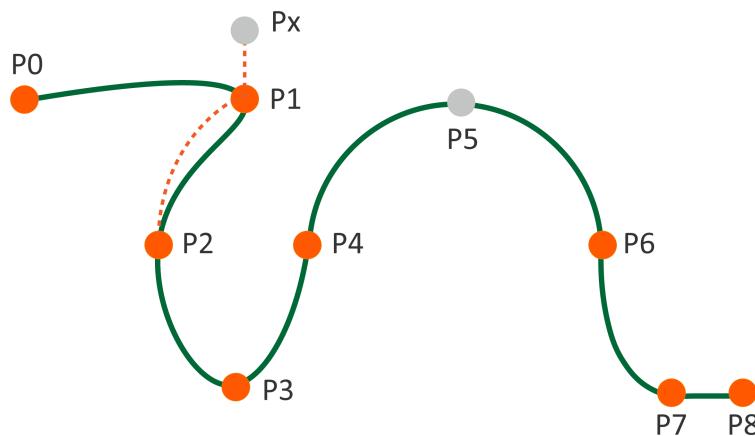


Abb. 8-34: Beispiel: Veränderte Bahn bei Satzanwahl auf P1 mit unbekanntem P0

Zeile	Beschreibung
2	Kopfzeile/Beginn des CP-Spline-Blocks
3 ... 9	Spline-Segmente
10	Ende des CP-Spline-Blocks

```

1   SPTP P0
2   SPLINE S1
3   SPL P1
4   SPL P2
5   SPL P3
6   SPL P4
7   SCIRC P5, P6
8   SPL P7
9   SLIN P8
10 ENDSPLINE

```

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.

8.6 Überschleifen von Spline-Bewegungen

Beschreibung

Alle Spline-Blöcke und alle Spline-Einzelbewegungen können miteinander überschliffen werden.



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 Vorlaufstop: Mit dem Beginn des Überschleifbogens ist das Ende des aktuellen Satzes erreicht. D. h., der Vorlaufstop 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.

8.7 Übung: Bahnkontur mit Spline-Block

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bahnkontur mit Spline-Block**

9 Logische Funktionen im Roboterprogramm nutzen

9.1 Lerneinheit: Logische Funktionen im Roboterprogramm nutzen

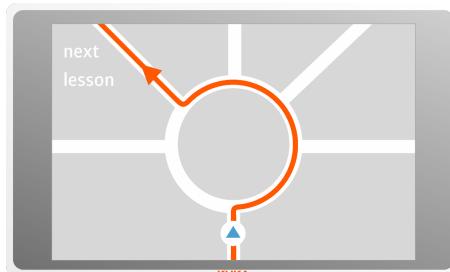


Abb. 9-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Einstieg in die Logikprogrammierung
- Anzeigen von Variablen
- Einfache Logikprogrammierung
- Programmierung von Logik mit SPLINE

9.2 Einstieg in die Logikprogrammierung

Beschreibung

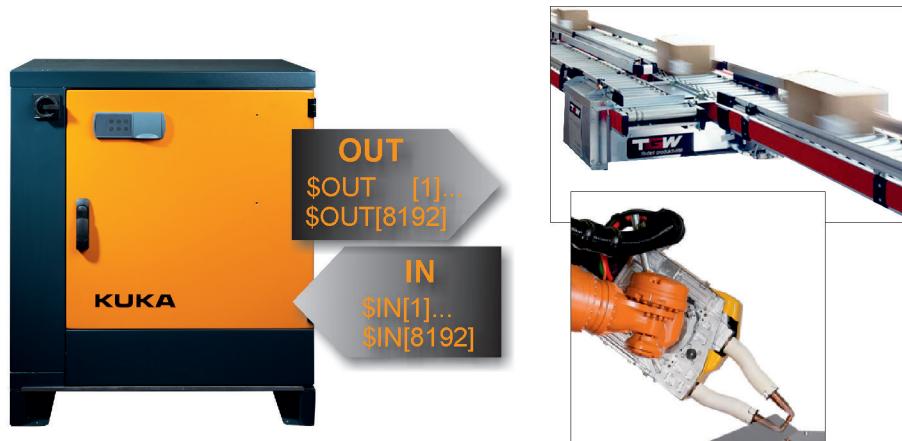


Abb. 9-2: Digitale Ein- und Ausgänge

- Um an einer definierten Programmstelle z. B. ein Werkzeug anzusteuern oder auf einen Signal zu warten, stehen in der Steuerung Feldbus-schnittstellen zur Verfügung.
- Über diese Schnittstelle erfolgt die **Kommunikation** mit der **Peripherie** über welche **digitale und analoge Ein- und Ausgänge** verwendet werden.

Begriffserklärung

Begriff	Erklärung	Beispiel
Kommunikation	Signalaustausch über eine Schnittstelle	Abfrage eines Zustands (Greifer Auf/Zu)
Peripherie	"Umgebung"	Werkzeug (z. B. Greifer, Schweißzange etc.), Sensoren, Materialfördersysteme etc.
digital	digitale Technik: wert- und zeitdiskrete Signale	Sensorsignal: Teil ist da: Wert 1 (TRUE/WAHR), Teil ist nicht da: Wert 0 (FALSE/FALSCH)
analog	Abbildung einer physikalischen Größe	Temperaturmessung
Eingänge	Die über die Feldbus-schnittstelle zur Steuerung <i> kommenden </i> Signale	Sensorsignal: Greifer ist auf/Greifer zu
Ausgänge	Die über die Feldbus-schnittstelle von Steuerung zur Peripherie <i> gesendeten </i> Signale	Befehl zum Schalten eines Ventils, das zum Schließen eines Backengreifers führt.

9.3 Anzeige von Variablen

Wert einer Variablen anzeigen und ändern

1. Menüpfad: Robotertaste > Anzeige > Variable > Einzeln

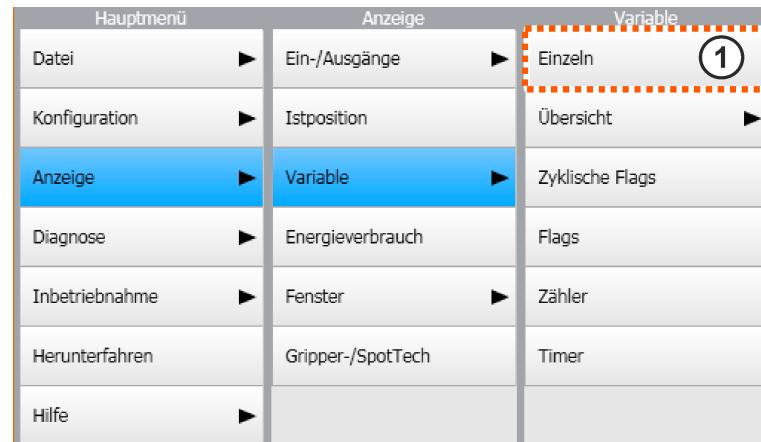


Abb. 9-3: Menüpfad

2. Das Fenster **Variablen Anzeige-Einzeln** öffnet sich.

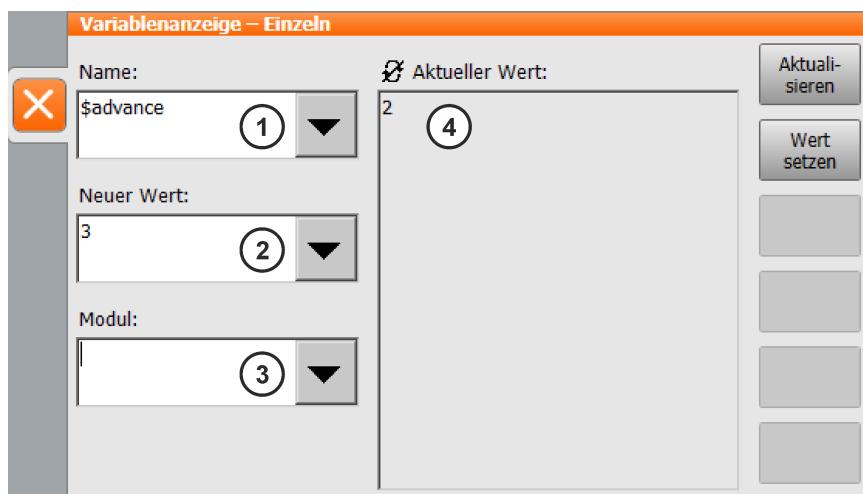


Abb. 9-4: Fenster, Variablen Anzeige-Einzel

Pos.	Beschreibung
1	Name der Variablen, die angezeigt oder 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 irrelevant.
4	Dieses Feld zeigt den Wert der Variablen an. Für die Aktualität der Variablen gibt es zwei Möglichkeiten: – : Der angezeigte Wert wird nicht automatisch aktualisiert. – : Der angezeigte Wert wird automatisch aktualisiert. Zwischen den Zuständen wechseln: – Aktual. drücken.

3. Variablenwert anzeigen lassen

- Im Feld **Name** den Namen der Variablen eingeben.
- 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.

- Die Eingabe-Taste oder die Schaltfläche "Aktualisieren" drücken.
Im Feld **Aktueller Wert** wird der aktuelle Wert der Variablen angezeigt. Wenn nichts angezeigt wird, dann ist der Variablen noch kein Wert zugewiesen worden.
- Im Feld **Neuer Wert** den gewünschten Wert eingeben.
Ausreichende Benutzerrechten müssen vorhanden sein.
- Schaltfläche **Wert stetzen**.
Im Feld **Aktueller Wert** wird der neue Wert angezeigt.

9.3.1 Anzeige von Ein- und Ausgängen

Beschreibung

Der aktuelle Zustand von feldbusbasierten digitalen und analogen Ein- und Ausgängen lässt sich auf dem smartPAD zur Anzeige bringen. Ausgänge können durch manuelle Eingabe gesetzt werden.

Hierzu sind entsprechende Benutzerrechte erforderlich.

Vorgehensweise

- Menüpfad: Robotertaste > Anzeige > Ein-/ Ausgänge

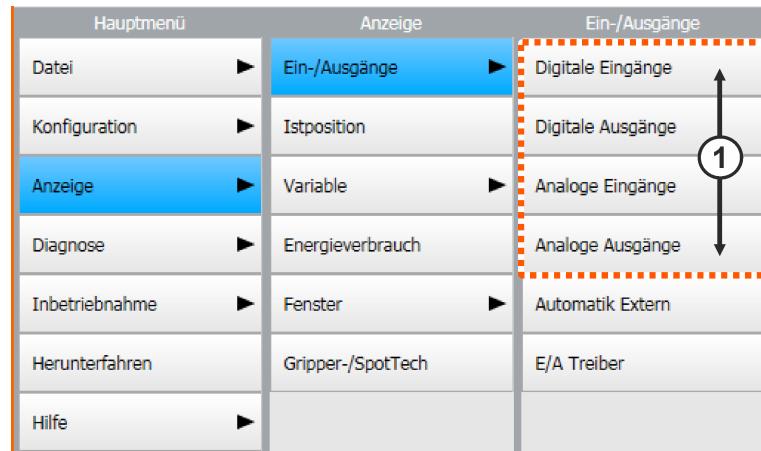


Abb. 9-5: Menüpfad: Ein- & Ausgänge

- In die gewünschte Anzeige wechseln (1).
 - Digitale Ein- und Ausgänge**
(>>> "Digitale Ein- und Ausgänge" Seite 272)
 - Analoge Ein- und Ausgänge**

Digitale Ein- und Ausgänge

Beispiel digitale Ausgänge

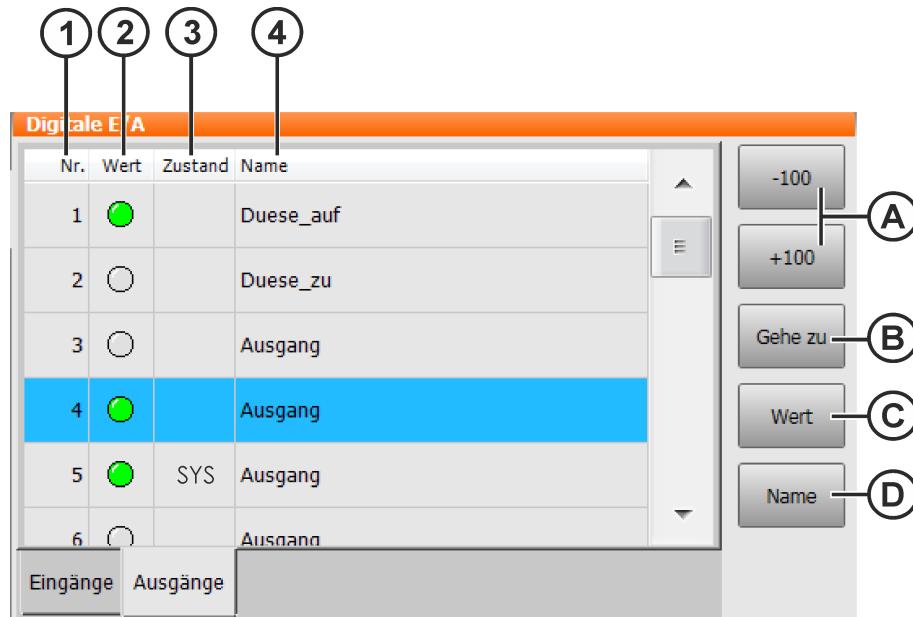


Abb. 9-6: digitale Ausgänge

Pos.	Beschreibung	
1	Nummer des Ein-/Ausgangs	
2	Wert des Ein-/Ausgangs Wenn ein Ein- oder Ausgang TRUE ist, ist er grün 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 Der Name des markierten Ein-/Ausgangs kann geändert werden.	

Nr.	Schaltfläche	Beschreibung
A	-100	Schaltet in der Anzeige 100 Ein- oder Ausgänge zurück.
	+100	Schaltet in der Anzeige 100 Ein- oder Ausgänge weiter.
B	Gehe zu	Die Nummer des gesuchten Ein- oder Ausgangs kann eingegeben werden.
C	Wert	Schaltet den markierten Ein- oder Ausgang zwischen TRUE und FALSE um. Voraussetzung: Der Zustimmungsschalter ist gedrückt. <ul style="list-style-type: none"> In der Betriebsart AUT EXT steht die Schaltfläche Wert nicht zur Verfügung. Für Eingänge steht die Schaltfläche Wert nur zur Verfügung, wenn die Simulation eingeschaltet ist.
D	Name	Der Name des markierten Ein- oder Ausgangs kann geändert werden.

9.3.2 Anzeige von Flags, Zählern und Timern

- Menüpfad: Robotertaste > Anzeige > Variable



Abb. 9-7: Menüpfad

Nun stehen die verschiedenen Systemvariablen zur Auswahl:

- **Zyklische Flags**
- **Flags**
- **Zähler**
- **Timer**

- **Beispiel:** Zähler

Nr.	Wert	Name
1	0	Zähler
2	0	Zähler
3	0	Zähler
4	8	my_counter
5	0	Zähler
6	0	Zähler
7	0	Zähler

Abb. 9-8: Zähler

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

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

9.4 Einfache Logikprogrammierung

Beschreibung

Bei der Programmierung von KUKA Robotern werden Ein- und Ausgangssignale für Logikanweisungen verwendet:

- **OUT**

Schalten eines Ausgangs an einer bestimmten Stelle im Programm

- **WAIT FOR**

Signalabhängige Wartefunktion » Hier wartet die Steuerung auf ein Signal:

- Eingang **IN**
- Ausgang **OUT**
- Zeitsignal **TIMER**
- Steuerungsinterne Speicheradresse (Merker/1-Bit-Speicher) **FLAG** oder **CYCFLAG** (wenn zyklisch kontinuierlich ausgewertet)
- **WAIT**

Zeitabhängige Wartefunktion » Die Steuerung wartet an dieser Stelle im Programm entsprechend einer eingetragenen Zeit.



Die hier beschriebenen Logikbefehle sind Grundbestandteil auch in älteren Softwareversionen. Die Verwendung ist immer ein Kompromiss zwischen exaktem Schaltpunkt und Vorlaufstop.
Es empfieilt sich, bei Neuprojektierungen die Verwendung von Triggerbefehlen (SYNOUT, Schalten auf der Bahn) in Kombination mit SPTP, SLIN, SCIRC und SPLINE-Blöcken zu prüfen.
Hier kann ein exakter Schaltpunkt unabhängig eines Vorlaufstops gewährleistet werden.

9.4.1 Programmieren von Wartefunktionen

Beschreibung

Rechnervorlauf

```

1 DEF Depal_Box1()
2
3INI
4 SPTP HOME VEL= 100 % DEFAULT
5 SPTP P1 VEL= 100 % PDAT1 Tool[5] Base[10]
6 SPTP P2 VEL= 100 % PDAT1 Tool[5] Base[10]
7 SLIN P3 VEL= 1 m/s CPDAT1 Tool[5] Base[10]
8 SLIN P4 VEL= 1 m/s CPDAT2 Tool[5] Base[10]
9 SPTP P5 VEL= 100% PDAT1 Tool[5] Base[10]
10 OUT 26'' State=TRUE
11 SPTP HOME VEL= 100 % DEFAULT
12
13 END

```

Zeile	
6	Position des Hauptlaufzeigers
9	mögliche Position des Vorlaufzeigers (nicht sichtbar)
10	Befehlssatz, der einen Vorlaufstop auslöst

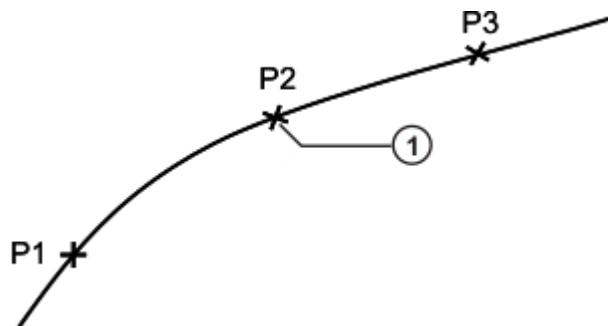
- Der Rechnervorlauf liest (nicht sichtbar für den Bediener) die Bewegungssätze im Vorlauf ein, um der Steuerung die Bahnplanung bei Überschleifbefehlen zu ermöglichen.
- Es werden jedoch nicht nur Bewegungsdaten mit dem Vorlauf abgearbeitet, sondern auch arithmetische und Peripherie steuernde Anweisungen.
- Manche Anweisungen lösen einen Vorlaufstopp aus. Dazu gehören Anweisungen, die die Peripherie beeinflussen, z. B. OUT-Anweisungen (Greifer schließen, Schweißzange öffnen).
- Wird der Vorlaufzeiger angehalten, kann nicht überschliffen werden.

Wartefunktion WAIT**Abb. 9-9: Inline-Formular WAIT**

Pos.	Beschreibung
1	Wartezeit • ≥ 0 s

- Wartefunktionen in einem Bewegungsprogramm lassen sich ganz einfach über Inline-Formulare programmieren.
- Dabei wird zwischen zeitabhängiger Wartefunktion und signalabhängiger Wartefunktion unterschieden.
- Mit **WAIT** wird die Roboterbewegung für die programmierte Zeit angehalten. WAIT löst immer einen Vorlaufstopps aus.

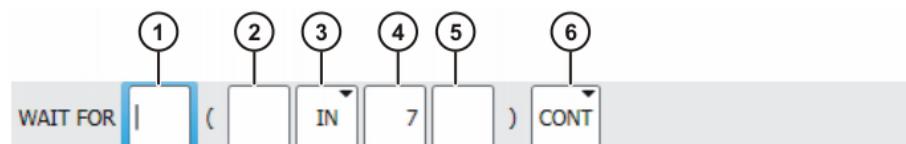
```
SPTP P1 Vel=100% PDAT1 Tool[1] Base[1]
SPTP P2 Vel=100% PDAT2 Tool[1] Base[1]
WAIT Time=2 sec
SPTP P3 Vel=100% PDAT3 Tool[1] Base[1]
```

Beispiel**Abb. 9-10: Beispielbewegung für Logik**

Pos.	Bemerkung
1	Bewegung wird für 2 Sekunden am Punkt P2 unterbrochen.

Wartefunktion WAIT FOR

- 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. 9-11: Inline-Formular WAIT FOR**

Pos.	Beschreibung
1	<p>Äußere Verknüpfung hinzufügen. Der Operator steht zwischen den geklammerten Ausdrücken.</p> <ul style="list-style-type: none"> • AND • OR • EXOR <p>NOT hinzufügen.</p> <ul style="list-style-type: none"> • NOT • [leer] <p>Den gewünschten Operator über die entsprechende Schaltfläche einfügen.</p>
2	<p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks.</p> <ul style="list-style-type: none"> • AND • OR • EXOR <p>NOT hinzufügen.</p> <ul style="list-style-type: none"> • NOT • [leer] <p>Den gewünschten Operator über die entsprechende Schaltfläche einfügen.</p>
3	<p>Signal, auf das gewartet wird</p> <ul style="list-style-type: none"> • IN • OUT • CYCFLAG • TIMER • FLAG
4	<p>Nummer des Signals</p> <ul style="list-style-type: none"> • 1 ... 4096
5	<p>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.</p>
6	<ul style="list-style-type: none"> • CONT: Bearbeitung im Vorlauf • [leer]: Bearbeitung mit Vorlaufstopp

Logische Verknüpfungen

Bei der Verwendung von signalabhängigen Wartefunktionen finden auch logische Verknüpfungen ihren Einsatz.

Mit logischen Verknüpfungen lassen sich die Abfragen verschiedener Signale oder Zustände kombinieren.

- Es können Abhängigkeiten geschaffen werden.
- Es können bestimmte Zustände ausgeschlossen werden.

Das Ergebnis einer Funktion mit einem logischen Operator liefert immer einen Wahrheitswert, d. h. am Ende kommt immer "Wahr" (Wert 1) oder "Falsch" (Wert 0) heraus.



Abb. 9-12: Beispiel und Prinzip einer logischen Verknüpfung

Operatoren für logische Verknüpfungen sind:

NOT	Dieser Operand wird zum Negieren verwendet, d. h. der Wert wird umgekehrt (Aus "Wahr" wird "Falsch").
AND	Das Ergebnis des Ausdrucks ist wahr, wenn beide verknüpften Ausdrücke wahr sind.
OR	Das Ergebnis des Ausdrucks ist wahr, wenn mindestens einer der verknüpften Ausdrücke wahr ist.
EXOR	Das Ergebnis des Ausdruckes ist wahr, wenn beide durch den Operator verknüpften Aussagen unterschiedliche Wahrheitswerte haben.

Signalabhängige Wartefunktionen können sowohl mit als auch ohne deren Bearbeitung im Vorlauf programmiert werden.

Bearbeitung ohne Vorlauf

Signalabhängige Wartefunktionen **Ohne Vorlauf** bedeutet, dass in jedem Falle die Bewegung im Punkt angehalten wird, und dort das Signal überprüft wird: (1) (>>> Abb. 9-13) Der Punkt kann also nicht überschritten werden.

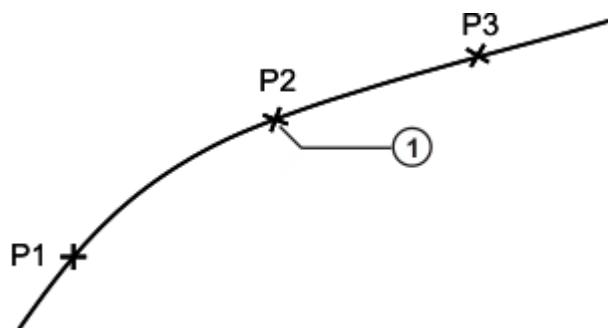


Abb. 9-13: Beispielbewegung für Logik

```
SPTP P1 Vel=100% PDAT1 Tool[1] Base[1]
SPTP P2 CONT Vel=100% PDAT2 Tool[1] Base[1]
WAIT FOR IN 10 'door_signal'
SPTP P3 Vel=100% PDAT3 Tool[1] Base[1]
```



Bei Abarbeitung einer WAIT FOR Zeile ohne CONT erscheint eine Hinweismeldung: "Überschleifen nicht möglich".

Bearbeitung mit Vorlauf (CONT)

- Mit **Vorlauf** programmierte signalabhängige Wartefunktionen erlauben es, dass der vor der Befehlszeile angelegte Punkt überschritten werden kann.

- Jedoch ist die gegenwärtige Lage des Vorlaufzeigers nicht eindeutig (Standardwert: drei Bewegungssätze), somit ist der genaue Zeitpunkt für die Überprüfung des Signals unbestimmt (1)
- Zudem werden Signaländerungen nach der Überprüfung des Signals nicht erkannt!

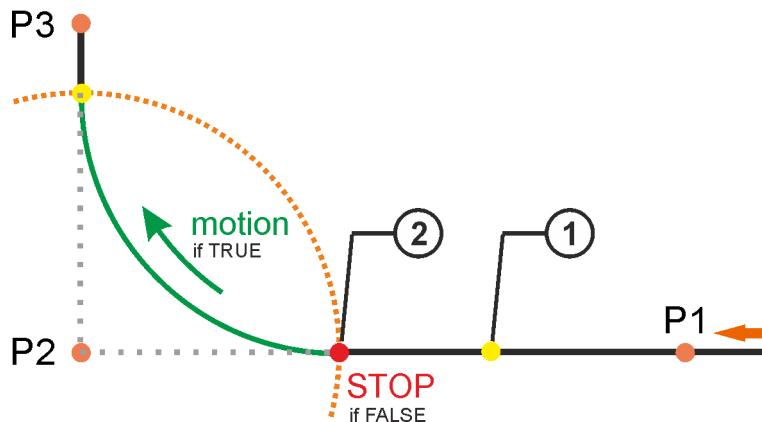


Abb. 9-14: WAIT FOR bei Überschleifbewegungen

```
SPTP P1 Vel=100% PDAT1 Tool[1] Base[1]
SLIN P2 CONT Vel=100% PDAT2 Tool[1] Base[1]
WAIT FOR IN 10 'door_signal' CONT
SLIN P3 Vel=100% PDAT3 Tool[1] Base[1]
```

1. Position (1)

- **WAIT FOR = TRUE** vor Erreichen der Position (2)
- » Roboter kann **ohne** Genauhalt die Überschleifbahn abfahren.

2. Position (2)

- **WAIT FOR = FALSE** bis zum Erreichen der Postion (2)
- » Roboter bleibt **mit einem** Genauhalt am Beginn der Überschleifkontur stehen.
- Änderung auf **WAIT FOR = TRUE** nach Erreichen der Postion (2)
- » Roboter fährt die Kontur auf der Überschleifbahn weiter.

Vorgehensweise

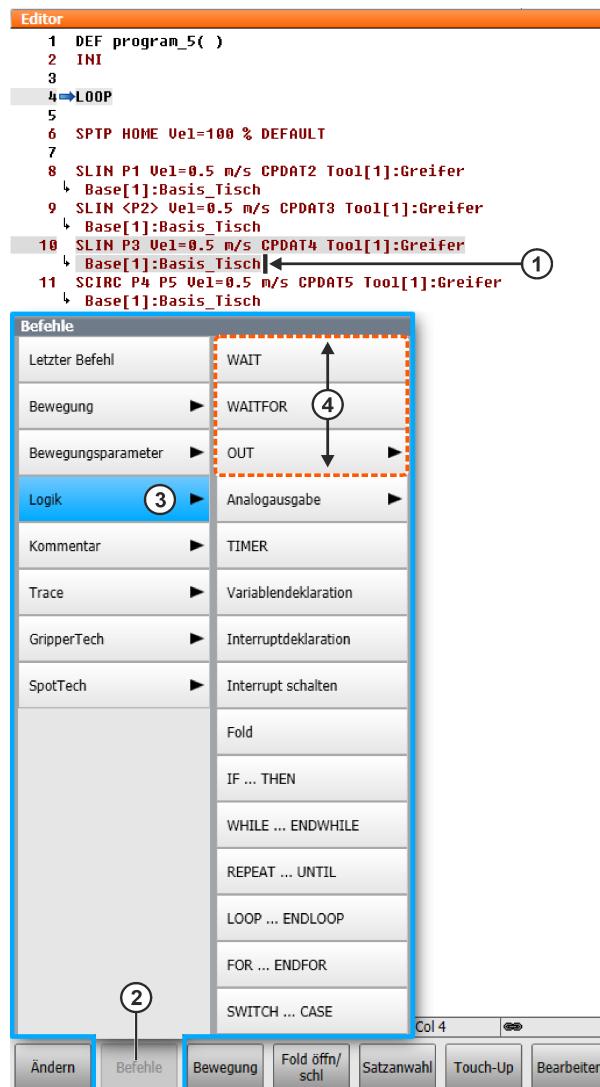


Abb. 9-15: Logik einfügen

1. Cursor in die Zeile setzen, nach der die Logikanweisung eingefügt werden soll (1).
2. Den Reiter **Befehle** (2) öffnen.
3. Über den Eintrag **Logik** (3) den gewünschten Warte- oder Ausgangsbefehl (4) auswählen.
4. Das gewünschte Inlineformular wird an der markierten Stelle im Programm eingefügt.
5. Im Inline-Formular die Parameter einstellen.
6. Anweisung mit **Befehl OK** speichern.

9.4.2 Programmierung von einfachen Schaltfunktionen

Beschreibung

Einfache Schaltfunktion

- Durch eine Schaltfunktion kann ein digitales Signal an die Peripherie gesendet werden.

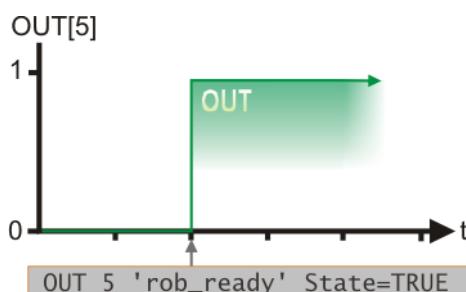


Abb. 9-16: Statisch Schalten

- Dazu wird eine Ausgangsnummer verwendet, die vorher der Schnittstelle entsprechend definiert wurde.
- Das Signal wird statisch gesetzt, d. h. es bleibt so lange bestehen, bis der Ausgang mit einem anderen Wert belegt wird.
- Die Schaltfunktion wird im Programm durch ein Inline-Formular realisiert:

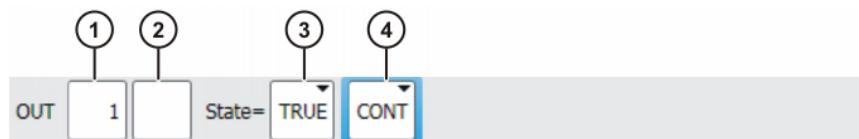


Abb. 9-17: Inline-Formular OUT

Pos.	Beschreibung
1	Nummer des Ausgangs – 1 ... 4096
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



Der einfache Schaltbefehl OUT wird verwendet, um am Anfang eines Programms Feldbusausgänge auf einen definierten Ausgangszustand zu schalten.

Gepulste Schaltfunktionen

- Wie bei der einfachen Schaltfunktion wird auch hier der Wert für einen Ausgang verändert.
- Allerdings wird beim Pulsen das Signal nach einer definierten Zeit wieder zurückgenommen.

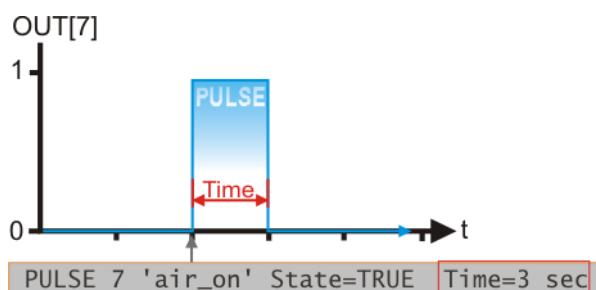


Abb. 9-18: Gepulster Pegel

- Die Programmierung erfolgt ebenfalls mit einem Inline-Formular, bei dem ein Impuls mit definierter Länge gesetzt wird.

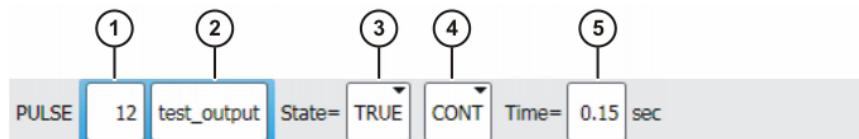


Abb. 9-19: Inline-Formular PULSE

Pos.	Beschreibung
1	Nummer des Ausgangs – 1 ... 4096
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: Pegel "High" – FALSE: Pegel "Low"
4	– CONT: Bearbeitung im Vorlauf – [leer]: Bearbeitung mit Vorlaufstop
5	Länge des Pulses – 0.10 ... 3.00 s

Auswirkungen von CONT bei Schaltfunktionen

- ohne CONT:

Lässt man den Eintrag CONT im OUT-Inlineformular weg, wird beim Schaltvorgang ein **Vorlaufstop** erzwungen und es folgt ein Genauhalt im Punkt vor dem Schaltbefehl. Nach dem Setzen des Ausgangs wird die Bewegung fortgesetzt.

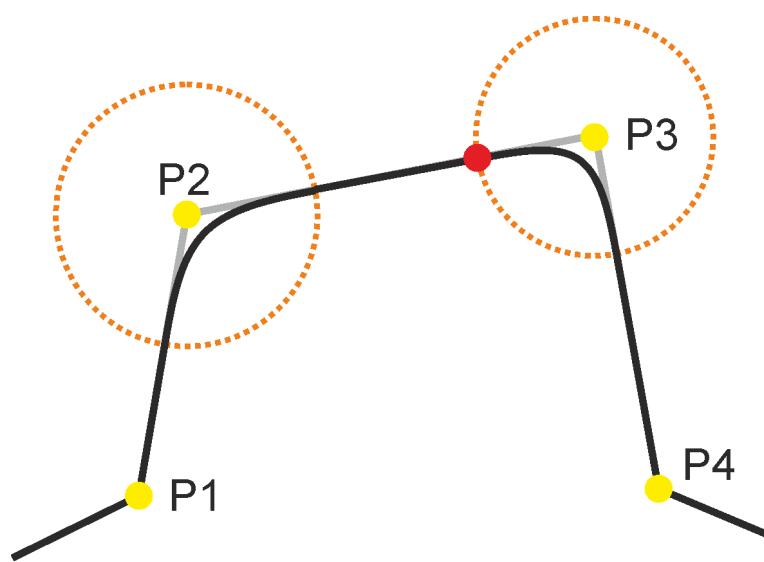


Abb. 9-20: Beispielbewegung mit Schalten mit Vorlaufstopp

```

SLIN P1 Vel=0.2 m/s CPDAT1 Tool[1] Base[1]
SLIN P2 CONT Vel=0.2 m/s CPDAT2 Tool[1] Base[1]
SLIN P3 CONT Vel=0.2 m/s CPDAT3 Tool[1] Base[1]
OUT 5 'rob_ready' State=TRUE
SLIN P4 Vel=0.2 m/s CPDAT4 Tool[1] Base[1]

```

- **mit CONT:**

Das Setzen des Eintrages CONT bewirkt, dass der Vorlaufzeiger nicht angehalten wird (es wird kein Vorlaufstop ausgelöst). Somit kann eine Bewegung vor dem Schaltbefehl überschliffen werden. Das Setzen des Signals erfolgt im **Vorlauf**.

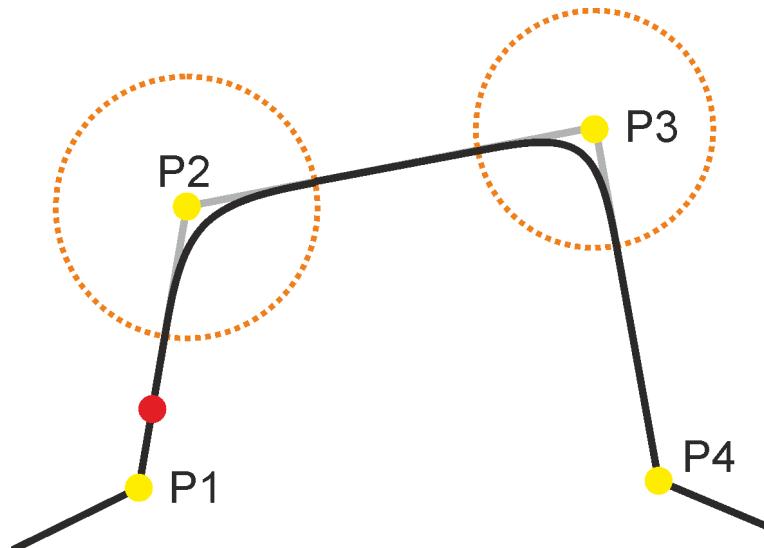


Abb. 9-21: Beispielbewegung mit Schalten im Vorlauf

```

SLIN P1 Vel=0.2 m/s CPDAT1 Tool[1] Base[1]
SLIN P2 CONT Vel=0.2 m/s CPDAT2 Tool[1] Base[1]
SLIN P3 CONT Vel=0.2 m/s CPDAT3 Tool[1] Base[1]
OUT 5 'rob_ready' State=TRUE CONT
SLIN P4 Vel=0.2 m/s CPDAT4 Tool[1] Base[1]

```



Der Standardwert für den Vorlaufzeiger beträgt drei Bewegungssätze. Jedoch kann der Vorlauf variieren, d. h. es muss damit gerechnet werden, dass der Schaltzeitpunkt nicht immer gleich ist!

Vorgehensweise

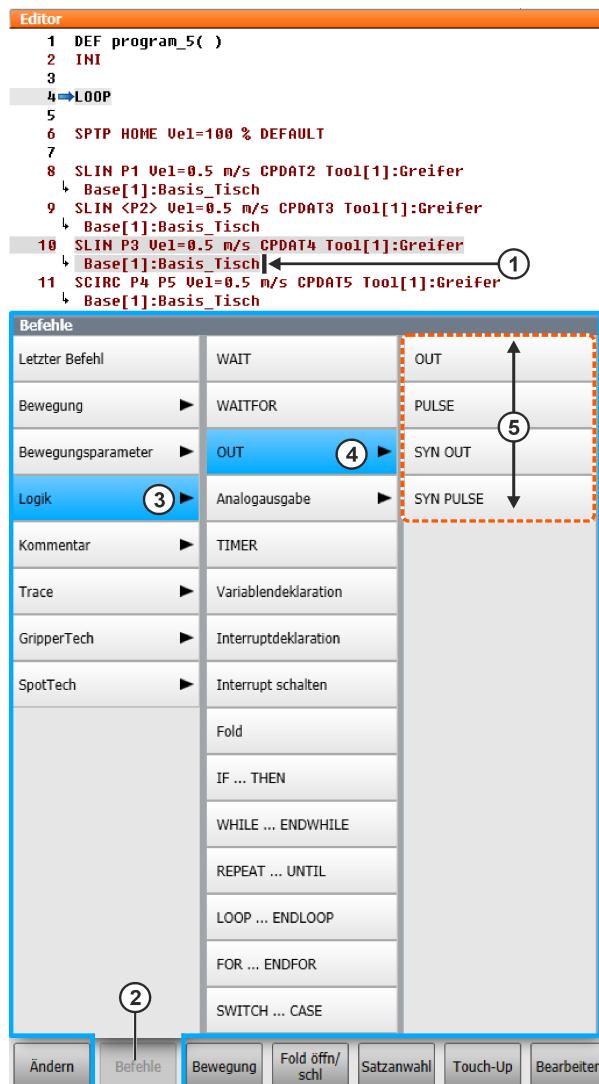


Abb. 9-22: Schaltbefehle einfügen

1. Cursor in die Zeile setzen, nach der die Logikanweisung (1) eingefügt werden soll.
2. **Menüfolge:** Befehle (2) > Logik (3) > OUT (4)> OUT, PULSE, SYN OUT oder SYN PULSE (5)
3. Im Inline-Formular die Parameter einstellen.
4. Anweisung mit **Befehl OK** speichern.

9.4.3 Optionale Übung: Bahnbewegung mit Logik versehen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bahnbewegung mit Logik versehen**

9.5 Programmierung von Logik mit SPLINE und SPLINE-Einzelsätzen

Beschreibung

Bei der Programmierung von Spline-Einzelsätzen (SLIN, SCIRC, SPTP) und Spline-Blöcken kann neben der Bewegung auch Logik eingebunden werden. Programmiert werden kann die Logik über KRL oder auch bequem über Inline-Formulare.

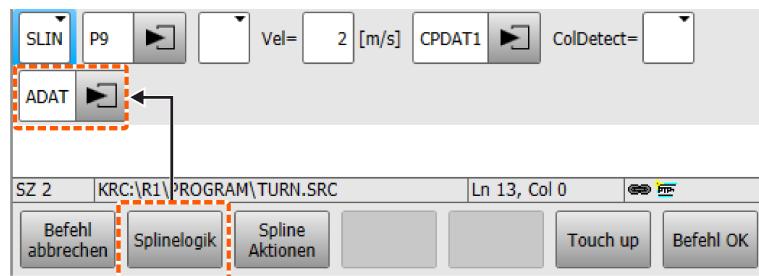


Abb. 9-23: SplineLogik im Inlineformular

Immer verfügbar (ADAT) sind:

- Trigger (Schaltbefehl auf der Bahn)
- Bedingter Stop

Nur im Spline-Block

- Konstantfahrbereich
- Trigger als Inline-Formular

Nur im Spline-Block und nur als KRL-Befehl

- Zeitblock

Nur außerhalb des Spline-Block

- Bedingter Stop als Inline-Formular (Spline Stop Condition)

9.5.1 Spline-Trigger programmieren

Beschreibung

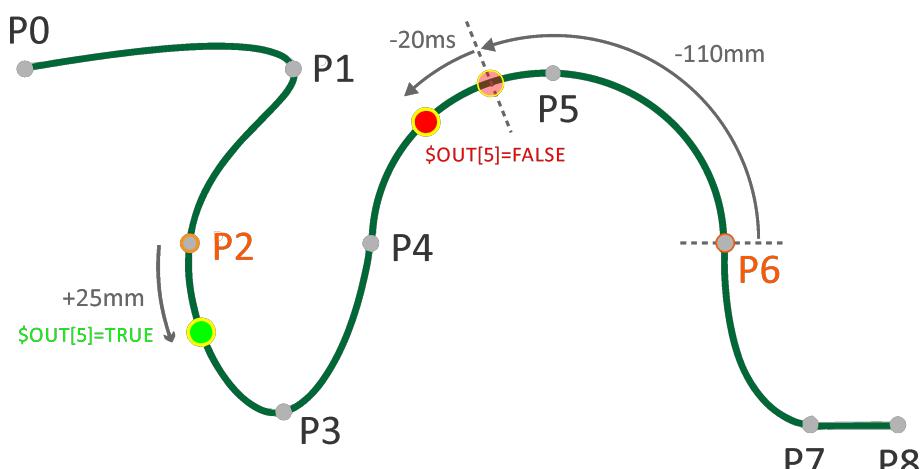


Abb. 9-24: Trigger mit SPLINE

Das Beispiel zeigt eine Bahn, welche mittels **Spline-Block** programmiert wurde. Die **Trigger-Funktion** ist auch auf eine Bahnbewegung übertragbar, welche sich aus **Einzelsätzen** zusammensetzt.

Der Trigger löst eine vom Benutzer definierte Anweisung aus, im Beispiel den digitalen Ausgang 5. 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 löst entweder direkt am Bezugspunkt aus, oder sie wird örtlich und/oder zeitlich verschoben.



Wenn der Trigger in einem Spline-Block verwendet wird, darf er nicht zwischen dem letzten Segment und ENDSPLINE stehen.

Möglichkeiten der Programmierung

- Inlineformular Optionsfenster Logik Trigger
- Inlineformular Spline Trigger
- Programmierung mittels KRL Befehl TRIGGER WHEN PATH

9.5.1.1 SPLINE Trigger programmieren

Vorgehensweise

1. Den Spline-Block mit der Schaltfläche "Fold öffnen/schließen" öffnen
2. Im **Punkt 5** soll der digitale Ausgang \$OUT[5] eingeschaltet werden.

```
4 SPLINE S1 Vel=0.5 m/s CPDAT 2 Tool[1]:pen Base[5]
blue
5 SPL P4
6 SLIN P5 ADAT
7 SPL P6
8 SLIN P7
9 SPL P8
10 ENDSPLINE
```

3. Parameter Auswahl

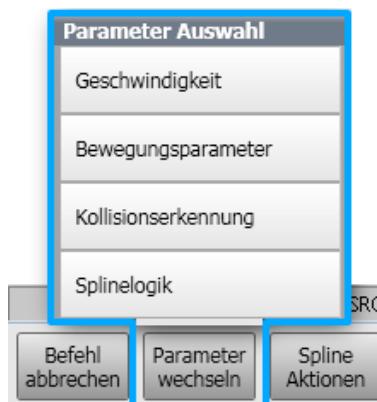


Abb. 9-25: Parameter wechseln

4. Den **Punkt 5** um die Splinelogik erweitern.
- **Schaltflächen:** Ändern > Parameter wechseln > Splinelogik hinzufügen
4. Punkt 5

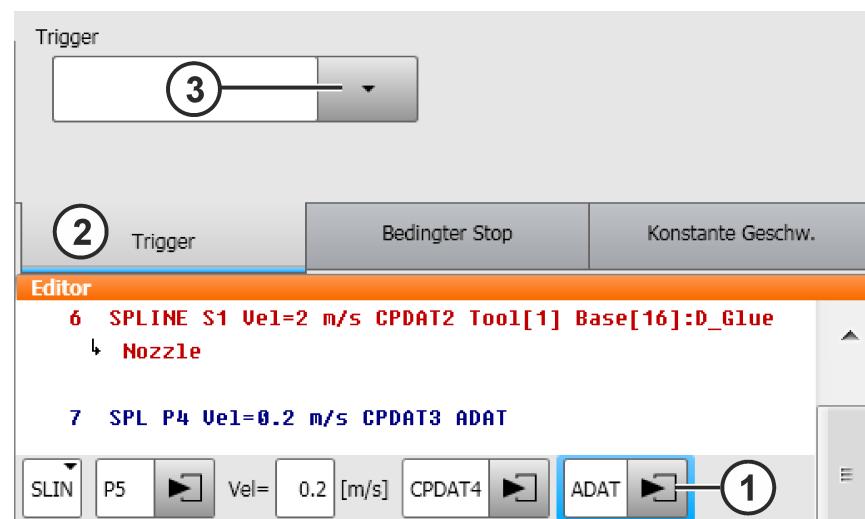


Abb. 9-26: Trigger Auswahl (leer)

- Auf das Pfeilsymbol (1) im Punkt 5 (ADAT) tippen.
- Im eingeblendeten Fenster in die Registerkarte Trigger (2) wechseln.



Über das Pull-Down Fenster Trigger kann aus vordefinierten Trigern/ Schaltaktionen gewählt werden, die in diesem Punkt aktiviert werden sollen. Diese sind aktuell noch nicht vorhanden und müssen erst im nächsten Schritt definiert werden.

5. Triggerbefehl hinzufügen

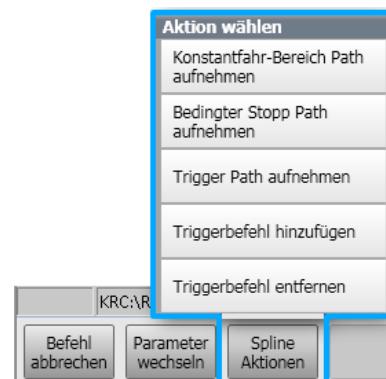


Abb. 9-27: Spline Aktionen

- Im geöffneten Bewegungssatz über die Schaltfläche **Spline Aktionen** einen **Triggerbefehl hinzufügen**.
- Das Trigger Fenster wird durch weitere Eingabefelder erweitert.

6. Variante 1: Trigger im Punkt aktivieren.

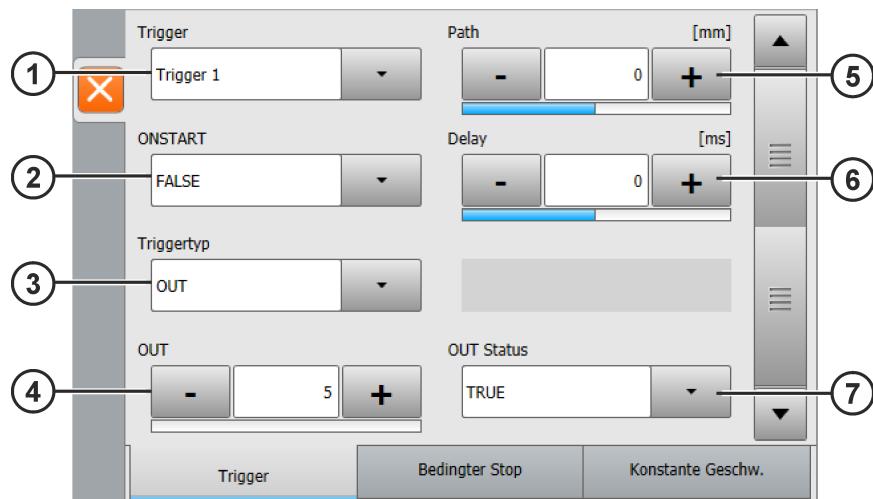


Abb. 9-28: Spline-Trigger definieren

- **Trigger (1)**

Das System legt einen neuen Trigger mit dem Namen Trigger 1 an. Sind bereits andere Trigger definiert, so können diese über diese Fenster aufgerufen und angepasst werden.

- **ONSTART (2)**

Wird der Parameter ONSTART auf **TRUE** gesetzt, bezieht sich der Bezugspunkt für das Auslösen des Triggers auf den Startpunkt. Dieser kann mit den Parametern Path und Delay zusätzlich verschoben werden. In unserem Beispiel der digitale Ausgang 5.

- **Triggertyp (3)**

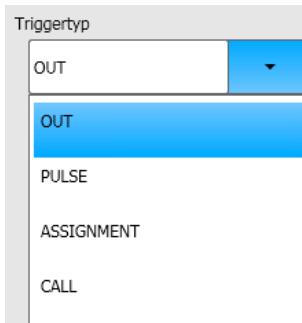


Abb. 9-29: Triggertyp

Definiert, welche Art von Trigger verwendet werden soll. Hier ist teilweise zusätzliche Expertenwissen in KRL notwendig.

In unserem Beispiel ist OUT zu wählen, um einen digitalen Ausgang zu schalten.



Abhängig von der Art des gewählten Triggers PULSE, ASSIGNMENT oder CALL, ändern sich die Eingabe- und Auswahlfenster in den Feldern 3, 4 und 7.

Nähere Informationen sind hier der Softwaredokumentation zu entnehmen.

- **OUT (4)**

Auswahl, welcher digitale Ausgang durch den Trigger geschalten wird.

- **PATH (5)**

Da der Trigger direkt im Punkt geschalten wird und auf TRUE steht, ist der örtliche Versatz inaktiv.

- **Delay (6)**

In Bezug auf den Punkt kann der Punkt abhängig von der gefahrenen Geschwindigkeit zeitlich verschoben werden. Hilfreich bei z. B. Klebeapplikationen

- **OUT-Status (7)**

Der gewünschte Schaltzustand des digitalen Ausgangs kann hier definiert werden. In diesem Beispiel wird dieser durch den booleschen Ausdruck TRUE eingeschaltet.

7. **Variante 2:** Triggerwert Path (Distanz) vom System aufnehmen lassen.

Im Beispiel: Örtliche und zeitliche Verschiebung in Bezug auf den Punkt 5.



Abb. 9-30: örtlicher und zeitlicher Versatz

- Den Roboter mittels der Programmtasten vorwärts oder rückwärts auf der Spline-Bahn bewegen, bis der "Triggerpunkt" erreicht ist.
- Über die Schaltfläche *Spline Aktionen > Trigger Path aufnehmen* die aktuelle Position in das Feld Path übernehmen.

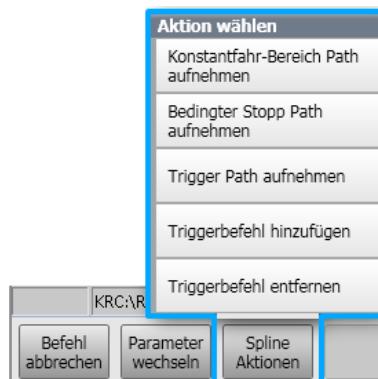


Abb. 9-31: Spline Aktionen



Wird die Bahn durch manuelles Verfahren des Roboters verlassen, ist eine Aufnahme mittel Path nicht möglich. Hier ist ein erneute Satzanwahl notwendig.



Durch das Teachen wird ONSTART immer auf FALSE gesetzt und die örtliche Verschiebung in Bezug auf den Zielpunkt (negativ) angeben.



Eine Verschiebung mit Bezug auf den Startpunkt ist mit ONSTART auf TRUE prinzipiell möglich. Hier ist jedoch der Offset (positiv) nur mittels numerischer Eingabe möglich.

- Bei Bedarf die zeitliche Verschiebung numerisch im Feld Delay [ms] eintragen.

9.5.1.2 SPLINE-Trigger mittels Inline-Formulars programmieren

Beschreibung

Ergänzend zur Programmierung des SPLINE Triggers direkt im Bewegungspunkt des Spline-Blocks gibt es noch die Möglichkeit, diesen in einem separaten Inline-Formulars zu programmieren.



Das Inlineformular SPLINE-Trigger kann nur innerhalb eines SPLINE-Blocks verwendet werden.

Vorgehensweise

- Bewegungssatz markieren, nachdem Logik-Befehl eingefügt werden soll.
- Spline-Logik einfügen



Abb. 9-32: Spline Logik einfügen

Menüpfad: Schaltfläche Befehle (1) > Bewegungsparameter (2) > Spline Trigger oder Spline Stop Condition (3)

- Das eingeblendete Inlineformular konfigurieren.

Inline-Formular

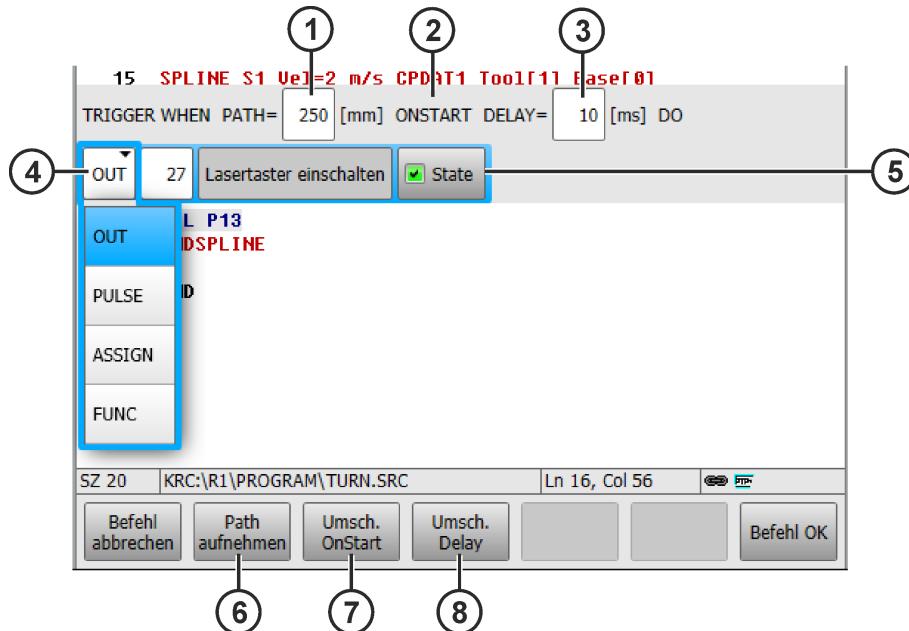


Abb. 9-33: Inlineformular, Spline Trigger

Pos.	Bezeichnung	Beschreibung
1	Path aufnehmen	Die örtliche Verschiebung (Path) des Schaltpunkts auf der Bahn kann mit Path aufnehmen "geteacht" werden.
2	ONSTART	Punkt, auf den sich der SPLINE-Trigger bezieht <ul style="list-style-type: none"> mit ONSTART: letzter Punkt vor dem Spline-Trigger ohne ONSTART: Punkt nach dem Spline-Trigger ONSTART kann über die Schaltfläche Umsch. OnStart gesetzt oder entfernt werden.
3	DELEY	zeitlicher Versatz des SPLINE Triggers
4	OUT	Auslösereaktion des Triggers <ul style="list-style-type: none"> OUT (Setze Ausgang) PULSE (Setze Pulsausgang) ASSIGN (Trigger Zuweisung) FUNC (Trigger Funktionsaufruf)
5	State	Angabe des anzunehmenden Zustands des Ausgangs TRUE = grüner Hacken auf Schaltfläche FALSE = graue Schaltfläche

Pos.	Bezeichnung	Beschreibung
6	Path aufnehmen	Automatische Ermittlung des PATH-Werts durch Anfahren des Trigger-Punkts auf der Bahn
7	Umsch. OnStart	Siehe Pos. 2
8	Umsch Delay	Siehe Pos. 3

Beispiel: Aufruf eines Unterprogramms

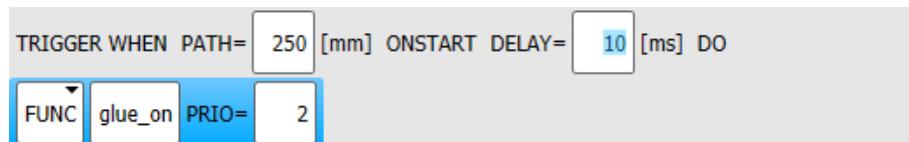


Abb. 9-34: Beispiel Interrupt

9.5.1.3 Programmierhinweise

Frage

Wo liegt der Bezugspunkt eines PATH-Triggers, wenn der Start- oder Zielpunkt überschlichen ist?



Dies ist in erster Linie abhängig davon, ob es sich um ein homogenes oder ein gemischtes Überschleifen handelt.

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 ein Spline-Block oder Einzelsatz sein.

Gemischtes Überschleifen

- Von einer CP-Spline-Bewegung in eine PTP-Spline-Bewegung oder umgekehrt
Jede Spline-Bewegung kann Spline-Block oder Einzelsatz sein.
- Von einer PTP-Bewegung in eine LIN- oder CIRC-Bewegung oder umgekehrt



Hier ist die Position des Bezugspunkts zusätzlich davon abhängig, ob es sich um Spline-Bewegungen handelt oder um herkömmliche Bewegungen.

Expertendokumentation

Nähere Information zum Thema "Bahnbezogene Schaltaktionen (Trigger)", finden Sie in der Expertendokumentation.



Xpert Suche:	KUKA System Software 8.6
	Bedien- und Programmieranleitung für Systemintegratoren
Xpert Filter:	Dokumentation > Bedien- und Programmieranleitung
Kapitel:	KRL-Programmierung > Bahnbbezogene Schaltaktionen (=Trigger)

9.5.2 Bedingten Stopp programmieren

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 Robotesteuerung 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 sich die Bedingung für den Stopp erst erfüllt, wenn der Roboter den Bremspunkt bereits passiert hat, ist es zu spät, um mit einer normalen Bremsrampe am Stopp-Punkt 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 Stopp-Punkt zum Stehen kommt, kann das Programm erst fortgesetzt werden, wenn die Bedingung nicht mehr erfüllt ist.

Falls der Roboter durch den bahntreuen NOT-HALT vor dem Stopp-Punkt zum Stehen kommt, geschieht Folgendes, wenn das Programm fortgesetzt wird:

- Wenn die Stoppbedingung **nicht mehr erfüllt ist**:
Der Roboter fährt weiter.
- Wenn die Stoppbedingung **noch erfüllt ist**:
Der Roboter fährt bis zum Stopp-Punkt und bleibt dort stehen.

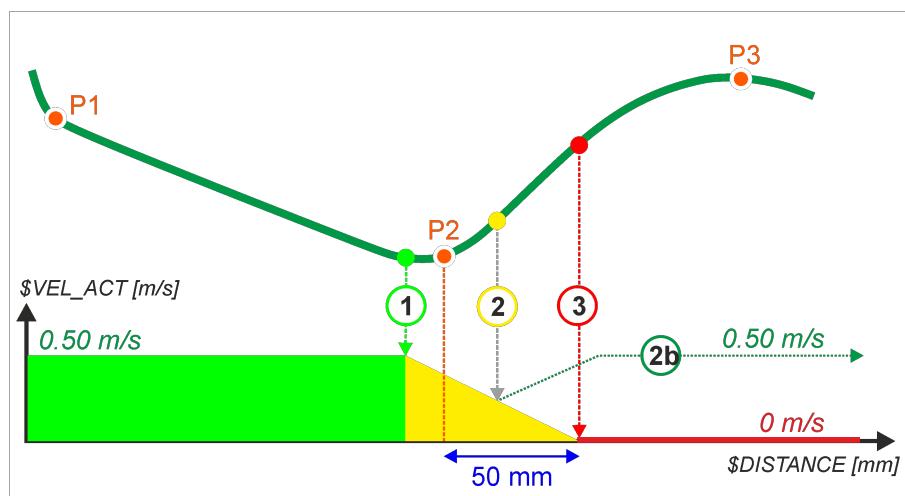


Abb. 9-35: Spline mit Bedingten Stopp

Pos.	Erklärung
1	berechneter Bremspunkt, abhängig von Weg und Geschwindigkeit
2	Es kommt möglicherweise zu einer Zustandsänderung (Bedingung ist nicht mehr erfüllt) während des Bremsvorgangs. Der Roboter beschleunigt wieder auf seine programmierte Geschwindigkeit.
2 b	Erreichen der programmierten Geschwindigkeit nach Wiederbeschleunigung.
3	Vorgegebener Stopp-Punkt (durch Inline-Formular)



Grundsätzlich können beliebig viele Bedingter 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})*.

Möglichkeiten der Programmierung

- Im Spline-Block oder im Spline-Einzelsatz:
Inlineformular Optionsfenster Logik Bedingter Stop
- Vor einem Spline-Block:
Inlineformular Spline Conditional Stop

9.5.2.1 Programmierhinweise

Digitalen Eingang überprüfen

```
$IN[1]==TRUE
```

\$IN	Hier handelt es sich um ein vom System festgelegtes Feld (Variable). In dieses Feld spiegelt der Inbetriebnehmer den aktuellen Zustand digitaler Eingänge von Feldbusteilnehmern. Der Zustand kann TRUE (EIN) oder FALSE (AUS) sein.
[1]	Die Zahl entspricht dem Feldindex. In diesem Fall der digitale Eingang-Nr. 1.

==	Vergleichsoperation; Der tatsächliche physikalische Zustand der Eingangsklemme (Zeile 1) wird mit dem Zustand rechts neben diesen Ausdruck (Zeile 4) verglichen.
TRUE	Vorgabewert, der digitale Eingang soll EIN sein. Wenn JA wird der Bedingte Stop ausgelöst. Soll der digitale Eingang auf den Zustand AUS verglichen werden, muss der Vergleichsoperator FALSE sein.
i	Um einen digitalen Eingang auf TRUE zu überwachen, kann auch folgende verkürzte Eingabe verwendet werden. \$IN[1]

9.5.2.2 Bedingten Stop programmieren

Vorgehensweise

1. Den SPLINE-Block mit der Schaltfläche "**Fold öffnen/schließen**" öffnen
2. Der **Bedingte Stop** soll im **Punkt 4** ausgelöst werden.

```
4 SPLINE S1 Vel=0.5 m/s CPDAT 2 Tool[1]:pen Base[5]
blue
5 SPL P1
6 SLIN P2
7 SPL P2
8 SLIN P4
9 SPL P5
10 ENDSPLINE
```

3. Splinelogik einfügen

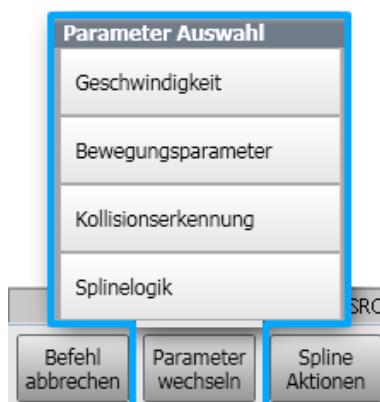


Abb. 9-36: Paramter wechseln

- Der Bewegungspunkt P4 wird um die Splinelogik erweitert werden (ADAT).
- **Schaltflächen:** Ändern > Paramter wechseln > Splinelogik hinzufügen

```
4 SPLINE S1 Vel=0.5 m/s CPDAT 2 Tool[1]:pen Base[5]
blue
5 SPL P1
6 SLIN P2
7 SPL P2
8 SLIN P4 ADAT
9 SPL P5
```

10 ENDSPLINE

4. Bedingten Stop aktivieren

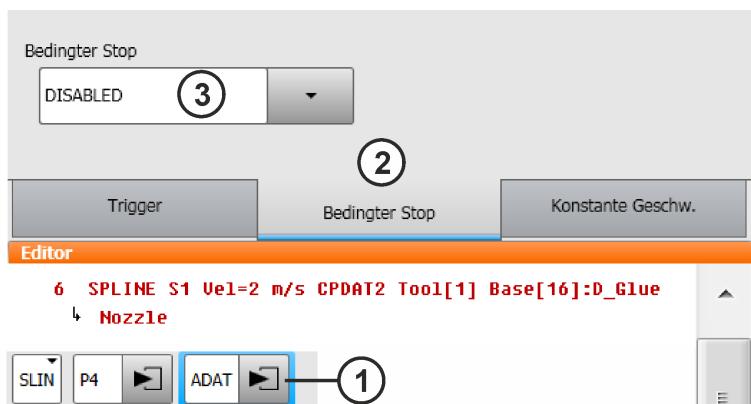


Abb. 9-37: Bedingten Stop aktivieren

- Auf das Pfeilsymbol (1) im Punkt 4 tippen.
- Im eingeblendeten Fenster in den Reiter Bedingter Stop (2) wechseln.
- Über das Pull-Down Fenster die Bedingter Stop (3) aktivieren.
Hierzu **ENABLED** auswählen.

5. Bedingung programmieren und händischen Versatz eingeben.

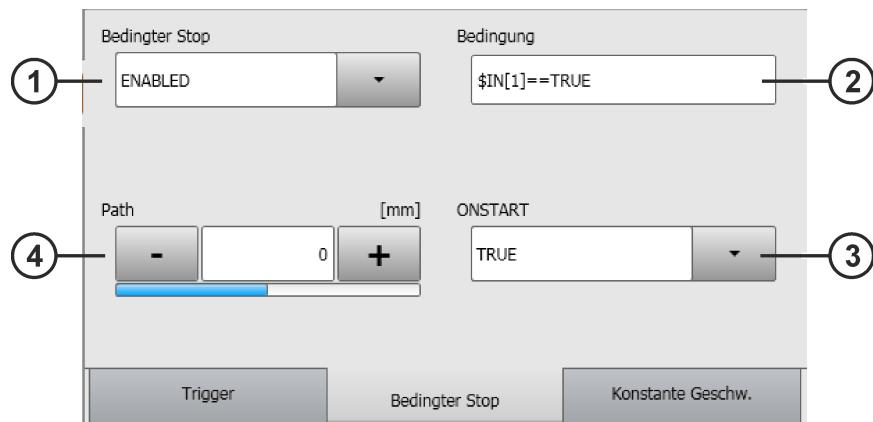


Abb. 9-38: Bedingter Stop programmieren

Pos	Paramter	Wert	Beschreibung
1	Bedingter Stop	ENABLED	Bedingter Stop ist in Bezug auf diesen Punkt aktiviert
2	Bedingung	KRL	<p>Bedingung, welche abgefragt wird, um beim Anfahren dieses Bewegungspunktes den Bedingten Stop auszulösen oder ohne Stop weiterzufahren.</p> <p>Die Bedingung wird in KRL programmiert.</p> <p>(>>> "Digitalen Eingang überprüfen" Seite 295)</p>

Pos	Parameter	Wert	Beschreibung
3	ONSTART	TRUE	Start bezieht sich auf den Startpunkt Wenn der Startpunkt überschritten ist, ergibt sich der Bezugspunkt auf die gleiche Weise wie beim homogenen Überschleifen beim PATH-Trigger.
4	Parth	0 mm	keine örtliche Verschiebung des Beginns für den Konstantfahrbereich

Variante

Örtliche Verschiebung des Beginns für den Bedingten Stop in Bezug auf den Punkt 4.

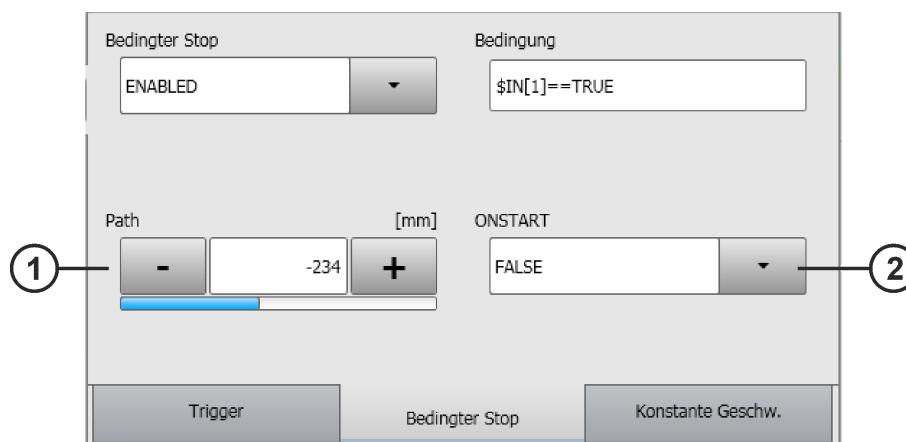


Abb. 9-39: Bezugspunkt verschoben

Pos	Parameter	Wert	Beschreibung
1	Parth	-234 mm	Die Bezugspunkt für den Bedingten Stop wird 234mm vor Erreichen des Zielpunktes festgelegt.
2	ONSTART	FALSE	Start bezieht sich auf den Zielpunkt Wenn der Zielpunkt überschritten ist, bezieht sich Start bzw. End auf den Anfang des Überschleifbogens.

- Den Roboter mittels der Programmtasten vorwärts oder rückwärts auf der Splinebahn bewegen, bis der Bezugspunkt für den Bedingten Stop erreicht ist.
- Über die Schaltfläche *Spline Aktionen > Bedingter Stopp Path aufnehmen* die aktuelle Position in das Feld Path übernehmen.



Abb. 9-40: Spline Aktionen



Wird die Bahn durch manuelles Verfahren des Roboters verlassen, ist eine Aufnahme mittel Path nicht möglich. Hier ist ein erneute Satzanwahl notwendig.



Durch das Teachen wird ONSTART immer auf FALSE gesetzt und die örtliche Verschiebung in Bezug auf den Zielpunkt (negativ) angeben.



Eine Verschiebung mit Bezug auf den Startpunkt ist mit ONSTART auf TRUE prinziell möglich. Hier ist jedoch der Offset (positiv) nur mittels numerischer Eingabe möglich.

9.5.2.3 Bedingten Stopp mittels Inlineformular programmieren

Beschreibung

Neben der Programmierung des Bedingten Stopps direkt im Bewegungspunkt des Spline-Blocks gibt es noch die Möglichkeit diesen in einem separaten Inlineformular zu programmieren.



Dieses Inline-Formular darf nur außerhalb eines Spline-Block verwendet werden.

Vorgehensweise

1. Bewegungssatz markieren, nachdem Logik-Befehl eingefügt werden soll.
2. Spline-Logik einfügen

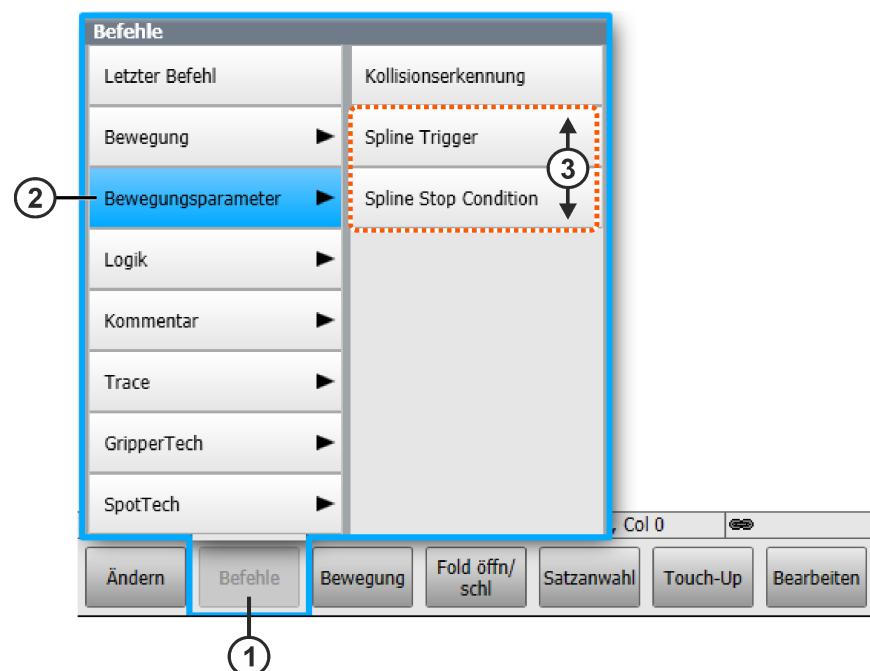


Abb. 9-41: Spline Logik einfügen

Menüpfad: Schalftäche Befehle (1) > Bewegungsparameter (2) > Spline Trigger oder Spline Stop Condition (3)

3. Das eingeblendete Inlineformular konfigurieren.

Inline-Formular



Abb. 9-42: Inline-Formular Spline Stop Condition

Pos.	Beschreibung
1	<p>Punkt, auf den sich der Bedingte Stopp bezieht</p> <ul style="list-style-type: none"> mit ONSTART: letzter Punkt vor dem Spline-Block ohne ONSTART: letzter Punkt im Spline-Block <p>Wenn der Spline überschliffen ist, gelten die gleichen Regeln wie beim PATH-Trigger.</p> <p>Hinweis: Informationen zum Überschleifen beim PATH-Trigger sind in der Bedien-/Programmieranleitung für Systemintegratoren zu finden.</p> <p>ONSTART kann über die Schaltfläche Umsch. OnStart gesetzt oder entfernt werden.</p>
2	<p>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.</p> <ul style="list-style-type: none"> Positiver Wert: Verschiebung in Richtung Bewegungsende Negativer Wert: Verschiebung in Richtung Bewegungsanfang <p>Der Stopp-Punkt kann nicht beliebig weit verschoben werden. Es gelten die gleichen Grenzen wie beim PATH-Trigger.</p> <p>Die örtliche Verschiebung kann auch geteacht werden.</p>
3	<p>Stopp-Bedingung</p> <p>Zulässig sind:</p> <ul style="list-style-type: none"> eine globale boolesche Variable ein Signalname ein Vergleich eine einfache logische Verknüpfung: NOT, OR, AND oder EXOR
	<p>Führt zu einer Fehlermeldung: $(\\$IN[29]==\text{FALSE}) \text{ OR } (\\$IN[22]==\text{TRUE})$</p> <p>Richtig: $\text{NOT } \\$IN[29] \text{ OR } \\$IN[22]$</p>

Schaltflächen

Im geöffnetet Zustand des Inlineformulars werden ergänzend Schaltflächen eingeblendet.

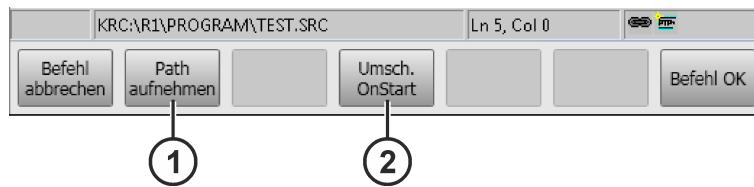


Abb. 9-43: Schaltflächen

Pos.	Bezeichnung	Beschreibung
1	Path aufnehmen	Die örtliche Verschiebung (Path) des Schaltpunktes auf der Bahn kann mit Path aufnehmen "geteacht" werden.
2	Umsch. OnStart	Festlegung des Schaltpunktes im Punkt ohne örtliche Verschiebung

9.5.3 Konstantfahrbereich

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 Spline-Block kann 1 Konstantfahrbereich definiert werden.
- Ein Konstantfahrbereich ist definiert durch eine Startanweisung und eine Endanweisung.
- Der Bereich kann sich nicht über den Spline-Block hinaus erstrecken.
- Der Bereich kann beliebig klein sein.

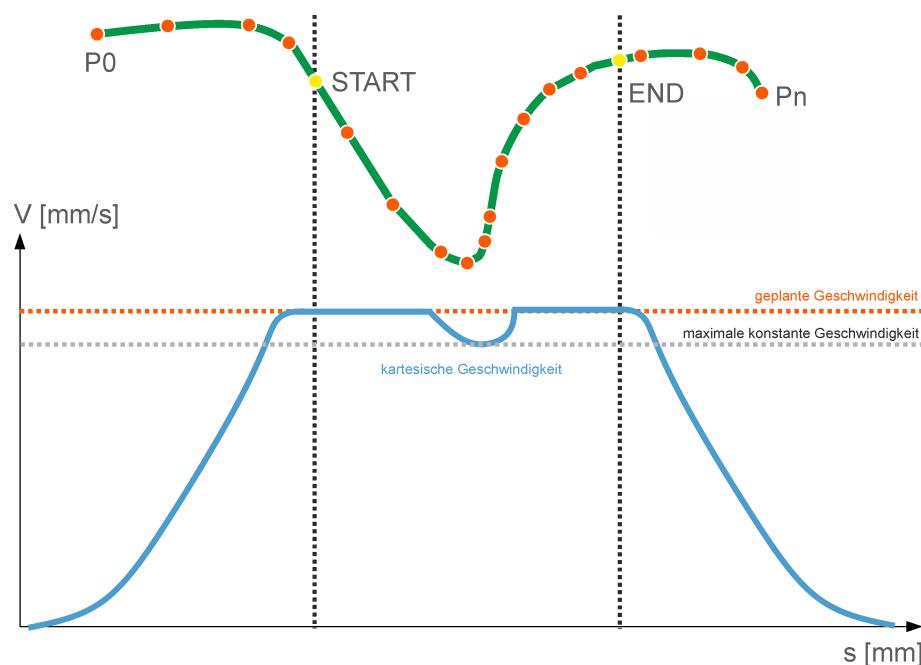


Abb. 9-44: Konstante Geschwindigkeit, Prinzip

Wenn es nicht möglich ist, die programmierte Geschwindigkeit konstant zu halten, zeigt die Robotersteuerung dies beim Programmlauf innerhalb der Testbetriebsarten 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 programmierten Geschwindigkeiten für den gesamten Bereich.

- Auch in den Segmenten mit höherer programmierte Geschwindigkeit wird in diesem Fall mit der niedrigsten Geschwindigkeit verfahren.
- Hier wird keine Meldung wegen Geschwindigkeitsunterschreitung ausgetragen.
- Dies geschieht nur, wenn die niedrigste programmierten Geschwindigkeit nicht gehalten werden kann.

9.5.3.1 Konstantfahrbereich programmieren

Vorgehensweise

1. Den SPLINE-Block mit der Schaltfläche "Fold öffnen/schließen" öffnen
2. Die Geschwindigkeit soll zwischen **P5** und **P12** konstantgehalten werden.

```

4 SPLINE S1 Vel=0.5 m/s CPDAT 2 Tool[1]:pen Base[5]
blue
5 SPL P4 CPDATA4
6 SLIN P5 ADAT
7 SPL P6
8 SLIN P7
9 SPL P8
10 SPL P9
11 SPL P10
12 SPL P11
13 SLIN P12ADAT
14 SPL P13
15 ENDSPLINE

```

3. Die Konstantfahrt wird am Punkt 5 gestartet und am Punkt 12 beendet.

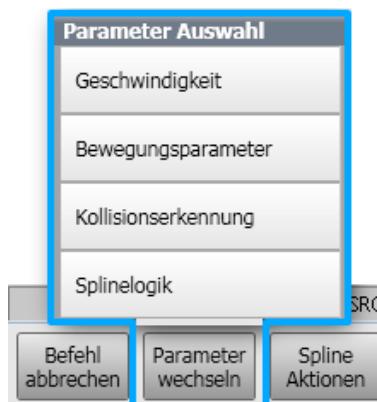


Abb. 9-45: Parameter wechseln

- D.h. muss an diesen beiden Stellen (P5 und P12) der Bewegungspunkt um die Splinelogik erweitert werden (ADAT).
 - **Schaltflächen:** Ändern > Paramter wechseln > Splinelogik hinzufügen
4. Punkt 5

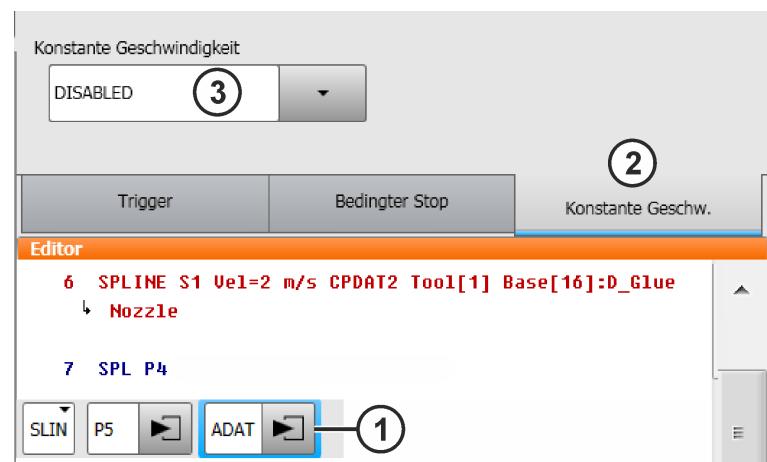


Abb. 9-46: Konstante Geschwindigkeit aktivieren

- Auf das Pfeilsymbol (1) im Punkt 5 tippen.
- Im eingeblendeten Fenster in den Reiter Konstante Geschwindigkeit (2) wechseln.
- Über das Pull-Down Fenster die Konstante Geschwindigkeit (3) aktivieren.

Hierzu **START** auswählen.

5. Programmieren des händischen Versatzes.

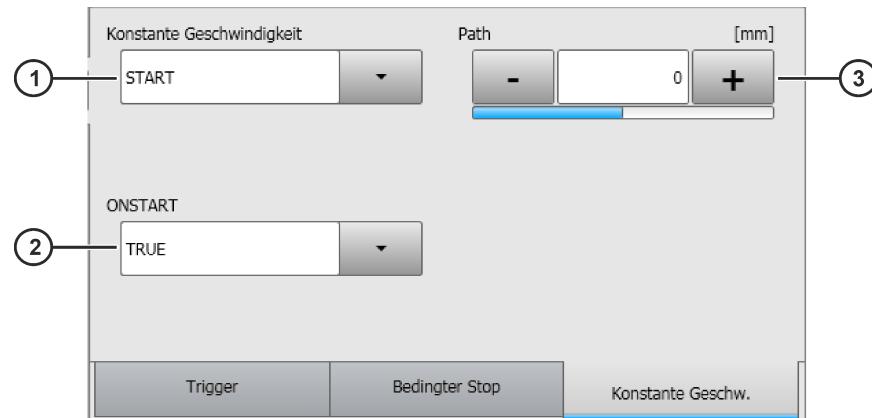


Abb. 9-47: Konstante Geschwindigkeit aktivieren

Parameter	Wert	Beschreibung
Konstante Geschwindigkeit	Start	beginnt mit dem Anfang des Konstantfahrbereichs
ONSTART	TRUE	Start bezieht sich auf den Startpunkt Wenn der Startpunkt überschritten ist, ergibt sich der Bezugspunkt auf die gleiche Weise wie beim homogenen Überschleifen beim PATH-Trigger.
Path	0 mm	keine örtliche Verschiebung des Beginns für den Konstantfahrbereich

Variante:

Örtliche Verschiebung des Beginns für den Konstantfahrbereich in Bezug auf den Punkt 5.

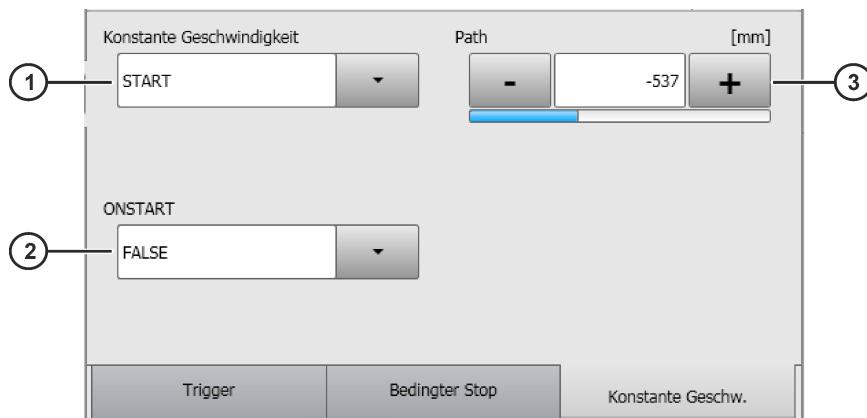


Abb. 9-48: Startpunkt verschoben

Parameter	Wert	Beschreibung
Konstante Geschwindigkeit	Start	Start des Konstantfahrbereichs
ONSTART	FALSE	Start bezieht sich auf den Zielpunkt Wenn der Zielpunkt überschritten ist, bezieht sich Start bzw. End auf den Anfang des Überschleifbogens.
Path	-537 mm	Die Konstantfahrt beginnt 537mm vor Erreichen des Zielpunktes.

- Den Roboter mittels der Programmtasten vorwärts oder rückwärts auf der Splinebahn bewegen, bis der Startpunkt der Konstantfahrt erreicht ist.
- Über die Schaltfläche *Spline Aktionen > Konstantfahr-Bereich Path aufnehmen* die aktuelle Position in das Feld Path übernehmen.



Abb. 9-49: Spline Aktionen



Wird die Bahn durch manuelles Verfahren des Roboters verlassen, ist eine Aufnahme mittel Path nicht möglich. Hier ist ein erneute Satzanwahl notwendig.



Durch das Teachen wird ONSTART immer auf FALSE gesetzt und die örtliche Verschiebung in Bezug auf den Zielpunkt (negativ) angeben.



Eine Verschiebung mit Bezug auf den Startpunkt ist mit ONSTART auf TRUE prinzipiell möglich. Hier ist jedoch der Offset (positiv) nur mittels numerischer Eingabe möglich.

6. Konstantfahrt im Punkt 7 beenden

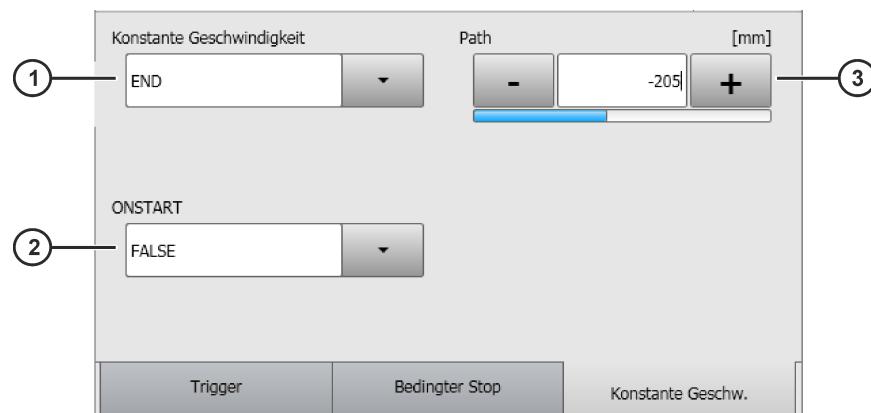


Abb. 9-50: Konstante Geschwindigkeit beenden

- Die Vorgehensweise entspricht der Aufnahme des Startpunktes.
- Hier ist im Feld Konstante Geschwindigkeit (1) END auszuwählen.
- Auch hier kann der Punkt örtlich durch *Spline Aktionen > Konstantfahr-Bereich Path aufnehmen* verschoben und aufgenommen werden.



Alternativ kann die Distanz auch direkt unter Path eingetragen werden.

Konstantfahrt prüfen

Die programmierte Konstantfahrt in der Betriebsart T1 testen.

Obwohl der Roboter mit reduzierter Geschwindigkeit die SPLINE-Kontur abfährt, signalisiert er durch eine entsprechende Meldung, welche Geschwindigkeit er abweichend zur programmierten Geschwindigkeit maximal in einer Konstantfahrt erreichen kann.

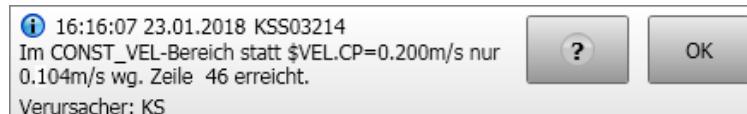


Abb. 9-51: Geschwindigkeitsreduktion

Es wird die Zeilennummer angezeigt, welcher Punkt hier die limitierende Größe ist.



Hierzu müssen alle Folds geöffnet sein, und in der Ansicht zusätzlich die DEF-Zeile, Detailansicht (ASCII) aktiviert sein, um die Zeilennummer zuzuordnen können.



Durch gezieltes Optimieren des Punkts (Position oder Orientierung) kann die zu erreichende Geschwindigkeit maximiert werden.



Das initiale Testen direkt im T2 oder Automatik ist zu vermeiden, da es hier für jeden Stützpunkt eine Meldung ausgegeben wird. Dies eignet sich nicht für das Optimieren einzelner Bewegungspunkte.

9.5.3.2 Programmierhinweise

Satzanwahl in den Konstantfahrbereich



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.

Was gilt als Satzanwahl?



Abb. 9-52: Beispiel Konstantfahrbereich (Bahn)

Satzanwahl auf Punkt ...	P1	P2	P3	P4
= Im Konstantfahrbereich?	Nein	Nein	Ja	Nein

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.

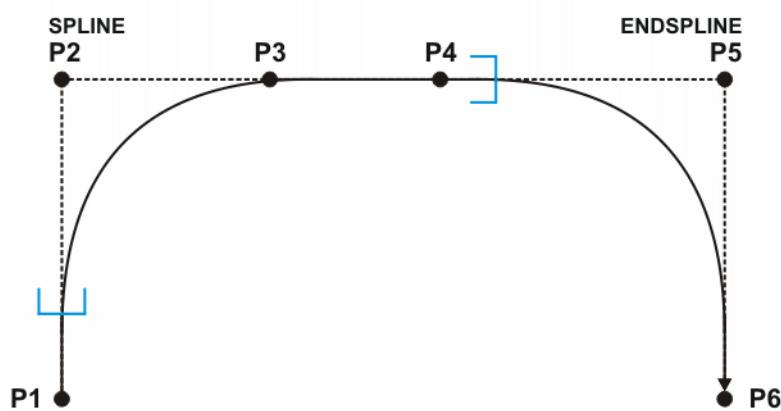


Abb. 9-53: Maximale Grenzen bei überschliffenem SPLINE/ENDSPLINE



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.

9.5.4 Übung: Splinebewegung mit Logik versehen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Splinebewegung mit Logik versehen**

9.5.5 Übung: Konstantfahrbereich und bedingter Stop

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- Konstantfahrbereich und bedingter Stop

10 Technologiepakete nutzen

10.1 Lerneinheit: Technologiepakete nutzen

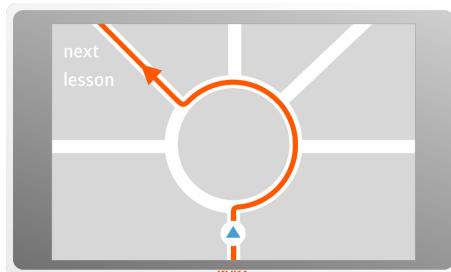


Abb. 10-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- KUKA.GripperTech kennenlernen
- KUKA.GripperTech bedienen
- KUKA.GripperTech konfigurieren
- KUKA.GripperTech programmieren

10.2 GripperTech kennenlernen

Beschreibung

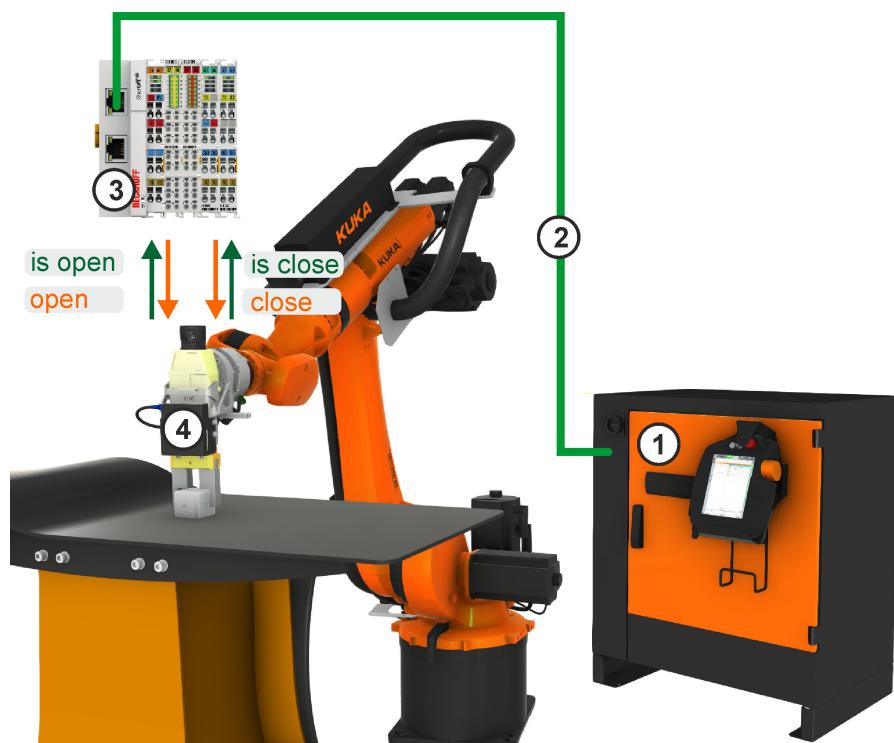


Abb. 10-2: Greifermansteuerung

Pos.	Beschreibung	Pos.	Beschreibung
1	KR C4	3	Feldbusmodul mit Ein- und Ausgangsmodulen
2	Feldbus	4	Greifer mit digitalen Ein- und Ausgängen

Belegung am Beispiel der College-Schulungszelle

Greifer öffnen	\$OUT[25]	Greifer ist offen	\$IN[25]
Greifer schließen	\$OUT[26]	Greifer ist geschlossen	\$IN[26]

- Werkzeuge, wie z. B. der Greifer der Schulungszelle (3), müssen zur Programmlaufzeit öffnen oder schließen.
- Eine Zustandsänderung wird über konfigurierte Ausgänge (\$OUT) erreicht.
- Über Eingänge (\$IN) meldet das Werkzeug den aktuellen Zustand an die Steuerung zurück.
- Die Übertragung zum Greifer erfolgt in der Regel über ein konfiguriertes Feldbusystem mittels digitaler Ein- und Ausgangsmodule (2).
- Eine Hilfe für die Programmierung und Konfiguration stellt hier das KUKA.GripperTech dar.

Folgende Inhalte werden nachfolgend behandelt

- Greifer über das smartPAD bedienen
(>>> [10.3 "Greiferbedienung mit KUKA.GripperTech" Seite 312](#))
- Greifesignale auf dem smartPAD anzeigen.
(>>> [10.3.1 "Greifesignale am smartPAD anzeigen" Seite 313](#))
- Greifer konfigurieren
(>>> [10.4 "Greifer konfigurieren" Seite 315](#))
- Greifer parametrieren
(>>> [10.4.1 "Greifer parametrieren" Seite 321](#))
- Greifer mittels Inline-Formulare programmieren.
(>>> [10.5 "Greifer mittels Inlineformularen programmieren" Seite 323](#))

10.3 Greiferbedienung mit KUKA.GripperTech

Beschreibung

Für die einfache Bedienung des Greifers oder der Punktschweißzange stehen auf dem smartPAD 4 separate Statustasten zur Verfügung. Diese können vom Kunden mittels der Konfiguration angepasst werden. Auf dem smartPAD, rechts neben der Statustaste befindet sich ein Icon, das die Funktion verdeutlicht.

Vorgehensweise

Statustasten einblenden



Abb. 10-3: Statustasten einblenden

- Sollten nach einer Neuinstallation oder nach einem Kaltstart die Statustasten nicht zur Verfügung stehen, so können diese manuell eingeblendet werden.
- Menüpfad:** Robotertaste > Konfiguration > Statustasten > Gripper-/SpotTech oder ServoTech
(je nach verwendeter Technologie)

Greifer bedienen

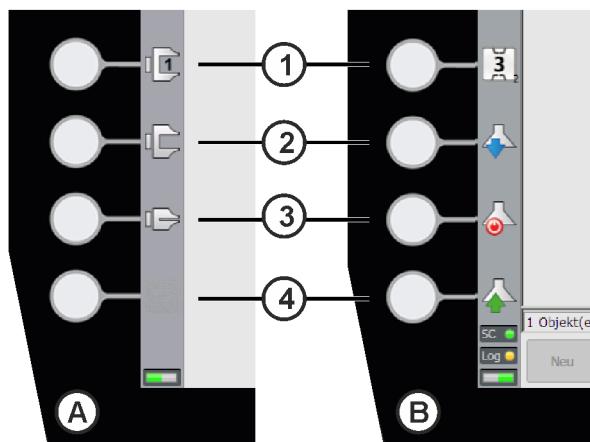


Abb. 10-4: Greifertasten

- Über die Statustaste (1) kann durch mehrfaches Drücken zwischen den vorkonfigurierten Greifern gewechselt werden.
- Bild A: Standardgreifer**
Greifer mit den Statustasten für die Funktionen Öffnen (2) und Schließen (3).
Hierzu parallel den Zustimmungsschalter betätigen. Die Symbole werden "aktiviert".
- Bild B: Sauggreifer**
Greifer mit den Statustasten für die Funktionen Blasen (2), Aus (3) und Saugen (4).
Hierzu parallel den Zustimmungsschalter betätigen. Die Symbole werden "aktiviert".



WARNUNG

Warnung!

Beim Umgang mit dem Greifersystem besteht Quetsch- und Schneidegefahr. Personal, das den Greifer bedient, muss sicherstellen, dass kein Körperteil vom Greifer gequetscht werden kann.

10.3.1 Greifersignale am smartPAD anzeigen

Beschreibung

Mittels der Gripper-Anzeige lässt sich der aktuelle Greiferzustand auf dem smartPAD visualisieren. Dadurch ist eine sichere Bedienung, insbesondere bei komplexeren Greifern (z.B. Sauggreifern), möglich.

Vorgehensweise

1. Menüpfad: Robotertaste > Anzeige > Gripper-/SpotTech



Abb. 10-5: Anzeige Greifersignale

2. Beispiel: COLLEGE-Greifer

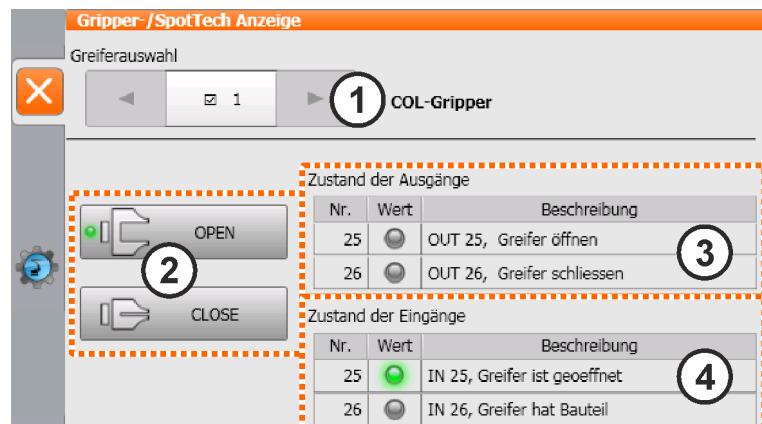


Abb. 10-6: Greifersignale anzeigen

Pos.	Beschreibung
1	Auswahl des anzuzeigenden Greifers
2	<p>Schaltzustände Öffnen und Schließen Über die Schaltflächen kann der Greifer alternativ zu den Greifertasten bedient werden. Über ein LED-Icon an den Statustasten wird der zuletzt gesetzte Schaltzustand angezeigt:</p> <ul style="list-style-type: none"> - Grün: Schaltzustand ist in Ordnung - Rot: Schaltzustand ist nicht in Ordnung, die fehlerhaften Eingänge werden ebenfalls mit einem roten LED-Icon dargestellt
3	<p>Zustand Ausgänge</p> <ul style="list-style-type: none"> - 25 Grau: Greifertaste öffnen ist nicht gedrückt > Ausgang FALSE - 26 Grau: Greifertaste schließen ist nicht gedrückt > Ausgang FALSE

Pos.	Beschreibung
4	Zustand der Eingänge <ul style="list-style-type: none"> – 25 grün: Greifer hat den Zustand geöffnet > Eingang TRUE – 26 grau: Greifer ist nicht geschlossen (hat kein Bauteil) > Eingang FALSE

3. Beispiel eines Sauggreifers



Abb. 10-7: Greifersignale eines Sauggreifers

10.4 Greifer konfigurieren

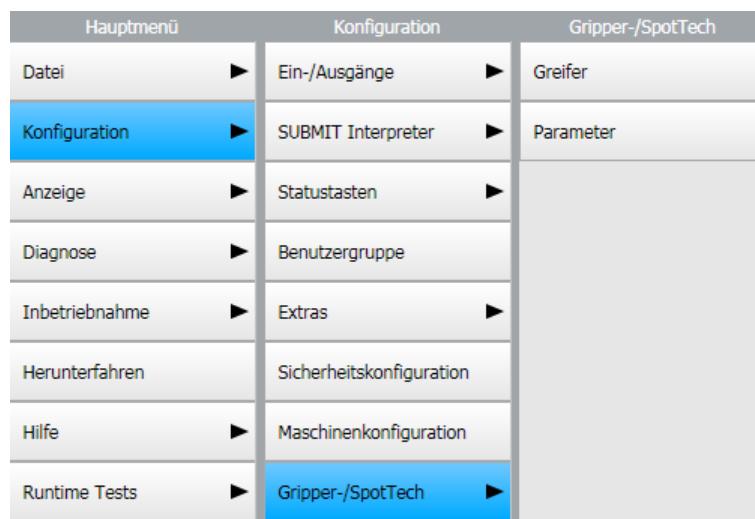
Beschreibung

Folgende Funktionen unterstützt KUKA.GripperTech:

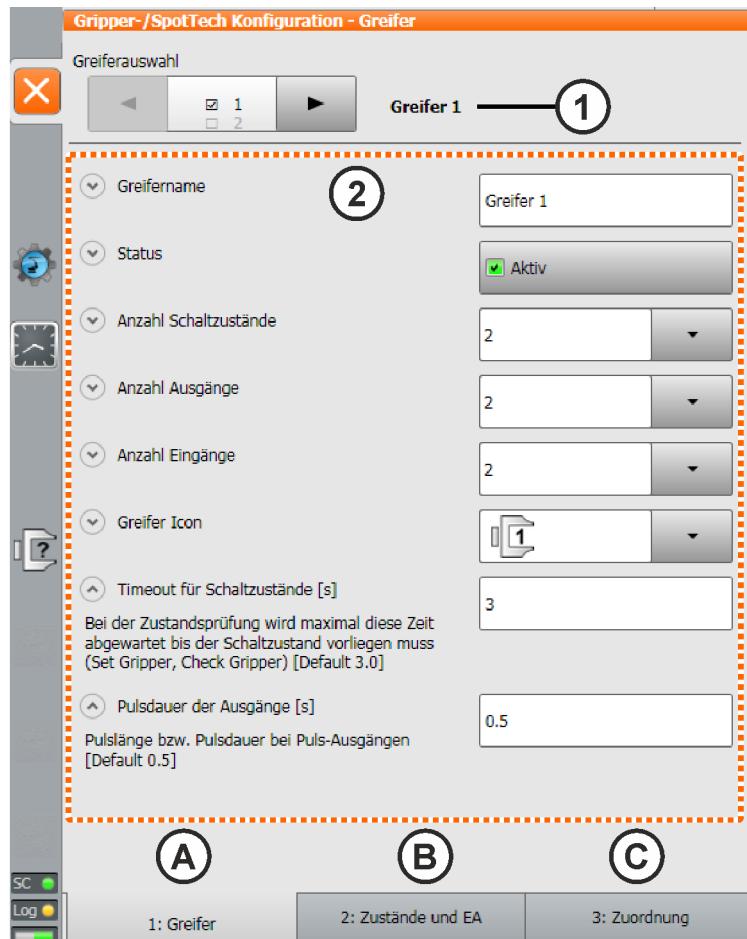
- Ansteuern verschiedener Greifertypen
- Abfragen und Anzeigen der aktuellen Greiferstellung.
- Festlegen von Bedingungen, um eine bestimmte Greiferstellung anusteueren. (kein Bestandteil dieser Schulung)
- Programmierung einer eigenen Anwenderlogik (kein Bestandteil dieser Schulung)
- Es können bis zu 16 verschiedene Greifer auf der Steuerung konfiguriert werden.

Vorgehensweise

Menüpfad: Robotertaster > Konfiguration > Gripper-/SpotTech > Greifer

**Abb. 10-8: Menüpfad****Registerkarte: Greifer**

- In der Registerkarte **Greifer (A)** werden greiferspezifische Einstellungen konfiguriert.
- In der darauffolgenden Registerkarte **Zustände und EA** werden die hier festgelegten Schaltzustände sowie die Ein- und Ausgänge definiert.

**Abb. 10-9: Registerkarte: Greifer**

Parameter	Beschreibung
Greiferauswahl (1)	Festlegung des zu konfigurierenden Greifers. Es stehen 16 Speicherplätze zur Verfügung
Greifername (2)	Name des Greifers (max. 24 Zeichen) Der Name wird im Inline-Formular angezeigt. Der vordefinierte Name kann geändert werden.
Status (2)	<ul style="list-style-type: none"> Mit Häkchen: Greifer können über die Statustaste bedient und über das ILF programmiert werden. Ohne Häkchen: Greifer ist nicht aktiv.
Anzahl Schaltzustände (2)	Anzahl der Schaltzustände des Greifers <ul style="list-style-type: none"> 1 ... 6
Anzahl Ausgänge (2)	Anzahl der Ausgänge, mit denen der Greifer gesteuert wird. <ul style="list-style-type: none"> 1 ... 6
Anzahl Eingänge (2)	Anzahl der Eingänge, mit denen der Greifer überwacht wird. <ul style="list-style-type: none"> 1 ... 6
Greifer Icon (2)	Auswahl der Icons für die Statustaste und für das Anzeige-Plugin Default: Icons mit Num
Timeout für Schaltzustände (2)	<p>Bei der Zustandsprüfung wird maximal diese Zeit abgewartet, bis der Schaltzustand vorliegen muss (Set Gripper mit Überwachung, Check Gripper).</p> <p>Wenn diese Zeit abgelaufen ist und der Schaltzustand immer noch nicht vorliegt, wird die Fehlerstrategie gestartet.</p> <p>Liegt der Schaltzustand vor, wird das Programm fortgesetzt ohne diese Zeit abzuwarten.</p> <ul style="list-style-type: none"> 0...10 s <p>Default: 3 s</p>
Pulsdauer der Ausgänge (2)	Pulsdauer der Ausgänge, die als Pulse definiert sind. <ul style="list-style-type: none"> 0...3 s <p>Default: 0,5 s</p>



Mittels den Pfeilsymbolen neben den Parametern lässt sich eine nähere Beschreibung einblenden.
Siehe als Beispiel im Bild *Timeout für Schaltzustände [s]* und *Pulsdauer der Ausgänge [s]*.

Registerkarte: Zustände und EA

- In der Registerkarte **Zustände und EA (B)** werden die Schaltzustände (2) sowie die Ausgänge (3) und Eingänge (4) zugeordnet.
- Hier werden für die Ein- und Ausgänge definierten Schaltzustände sowie die Voraussetzungen für die manuelle Bedienung zugeordnet.

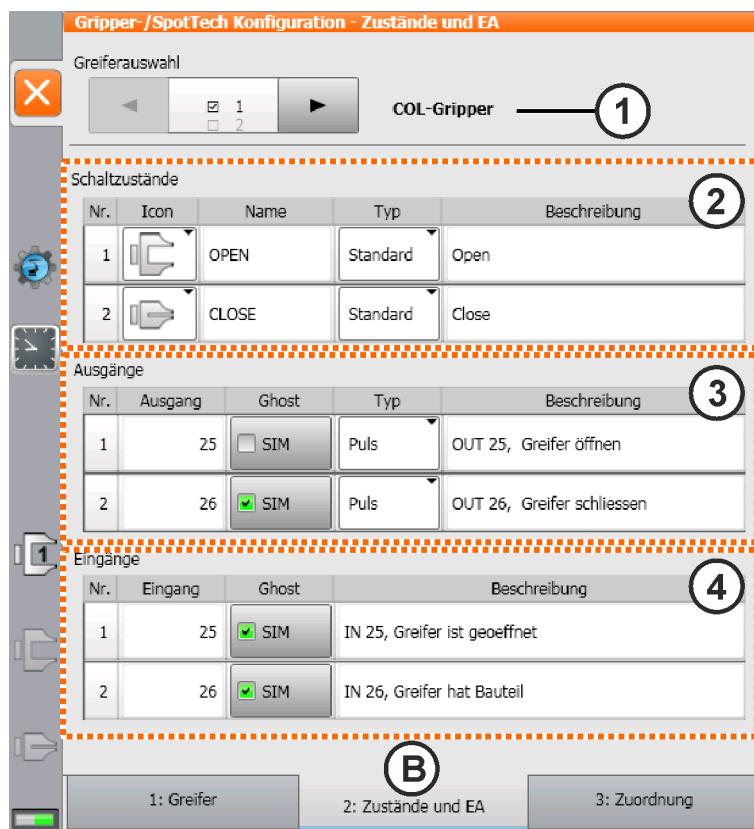


Abb. 10-10: Registerkarte: Zustände und EA

- Schaltzustände (2)

Parameter	Beschreibung
Nr.	Nummer des Schaltzustandes Interner Index für die Steuerung Beginnt mit 1 und ist fortlaufend. Kann nicht verändert werden
Bild	Schaltzustand Icon Auswahl von Icons, die an den Statustasten und im Anzeige-Plugin für den Schaltzustand angezeigt werden sollen.
Name	Name des Schaltzustands Wird in den Inline-Formularen angezeigt, wenn der zugehörige Greifer ausgewählt ist.
Typ	Typ der Statustaste für diesen Schaltzustand <ul style="list-style-type: none"> – Standardtaste: Aktion wird sofort ausgeführt. – Kritische Taste: Taste muss 2 mal hintereinander innerhalb von 4 s betätigt werden, damit die Aktion ausgeführt wird. Nur die ersten 3 Statustasten können als kritische Taste definiert werden. Die übrigen 3 können nur im Anzeige-Plugin angezeigt werden und deshalb nicht als kritisch definiert werden.
Beschreibung	Beschreibung des Schaltzustands

- Ausgänge (3) und Eingänge (4)

Parameter	Beschreibung
Nr.	Interner Index für die Steuerung des Ausgangs Interner Index für die Auswertung des Eingangs Beginnt mit der 1 und ist fortlaufend. Kann nicht verändert werden
Ausgang/Eingang	Ausgang und Eingang Default: 0
Ghost/SIM	Einstellung, ob die Ein-/Ausgänge im Ghostmode gesetzt oder ausgewertet werden sollen. Eingänge: <ul style="list-style-type: none"> – Sim mit Häkchen: Eingang wird nicht ausgewertet. – Sim ohne Häkchen: Eingang wird auch im GhostMode ausgewertet. Ausgänge: <ul style="list-style-type: none"> – Sim mit Häkchen: Ausgang wird nicht gesetzt. – Sim ohne Häkchen: Ausgang wird auch im GhostMode gesetzt
Typ	Typ des Ausgangs <ul style="list-style-type: none"> – Standard: Der Ausgang ist als Standardausgang definiert. Der Zustand bleibt erhalten. – Puls: Der Ausgang ist als Pulsausgang definiert. Der gesetzte Zustand bleibt nur für eine bestimmte Zeit erhalten.
Beschreibung	Langtext des Ausgangs und Eingangs

Registerkarte: Zuordnung

- In der Registerkarte **Zuordnung (3)** werden den definierten Schaltzuständen, die definierten Aus- und Eingänge sowie die Voraussetzungen für die manuelle Bedienung zugeordnet.
- Damit die Schaltzustände sowie die Ein- und Ausgänge in dieser Registerkarte zur Verfügung stehen, müssen sie in der vorherigen Registerkarte **Zustände und EA** definiert werden.

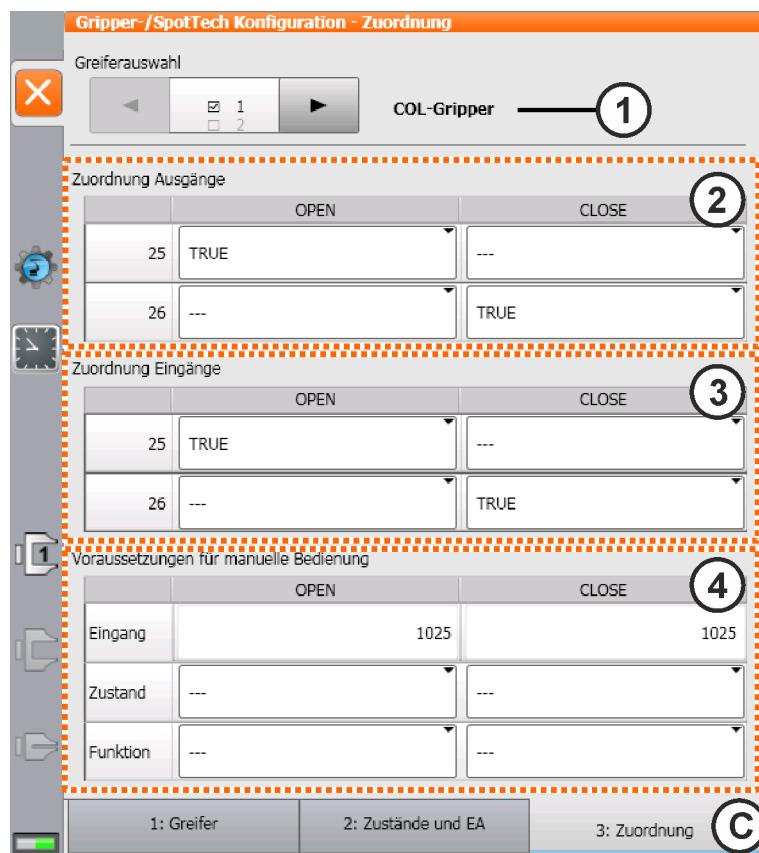


Abb. 10-11: Registerkarte: Zuordnung

Zuordnungen	Beschreibung
Zuordnung Ausgänge (2)	Setzen von Ausgängen für einen Zustand <ul style="list-style-type: none"> TRUE: Ausgang wird auf TRUE gesetzt. FALSE: Ausgang wird auf FALSE gesetzt. [/er]: Ausgang wird nicht gesetzt, der aktuelle Zustand bleibt erhalten.
Zuordnung Eingänge (3)	Auswertung der Eingänge für einen Schaltzustand <ul style="list-style-type: none"> TRUE: Eingang muss gesetzt sein. FALSE: Eingang darf nicht gesetzt sein. [/er]: Eingang ist nicht relevant und wird nicht ausgewertet.

Zuordnungen	Beschreibung
Voraussetzungen für manuelle Bedienung (4)	<p>Für das manuelle Setzen des Schaltzustands können bis zu 3 Voraussetzungen festgelegt werden. Der Schaltzustand wird erst dann gesetzt, wenn die festgelegten Voraussetzungen erfüllt sind.</p> <p>Eingang:</p> <ul style="list-style-type: none"> • Nummer des Eingangs wird festgelegt. • Der definierte Eingang muss TRUE sein. • Default: 1025 <p>Zustand:</p> <ul style="list-style-type: none"> • Auswahl von definierten Schaltzuständen • Neuer Schaltzustand wird gesetzt, wenn in der Registerkarte "Zustände und EA" der definierte Schaltzustand vorliegt. • Default: [/eer] <p>Funktion:</p> <ul style="list-style-type: none"> • Auswahl von bis zu 10 verschiedenen Userroutinen. Diese können in der Grp_User.src (R1\TP\GripperSpotTech) programmiert werden. Jede Bedingung sendet einen Rückgabewert zurück. • Rückgabewert: <ul style="list-style-type: none"> – TRUE: Schaltzustand wird gesetzt. – FALSE: Schaltzustand wird nicht gesetzt. • Default: [/eer]

10.4.1 Greifer parametrieren

Beschreibung

Ghostbetrieb

Wird eine Anlage in Betrieb genommen oder nach Ende eines Produktionsabschnitt "leergefahren" so geschieht dies in der Regel ohne Bauteile. Das Leerefahren wird in der Konfiguration als **Ghostbetrieb** bezeichnet. Um eine ungewollte Aktion des Greifer vorzubeugen, z. B. der Greifer nimmt eine Bauteil auf, so kann dies in der Konfiguration explizit deaktiviert werden. Bei Sauggreifern geht man in der Regel einen etwas anderen Weg. Hier wird der Greifer statt des Zustands Saugen mit dem Zustand Blasen angesteuert. Die Zustände werden vertauscht. Bei einer Blechaufnahme wird damit sicher ein Verrutschen oder Verschieben verhindert. Der Ghostbetrieb kann manuell über das Konfigurationsmenü aktiv und inaktiv gesetzt werden. Alternativ kann der Ghostbetrieb auch über Ein- und Ausgänge der SPS ein- und ausgeschalten werden.

Vorgehensweise

Greiferparameter

- **Menüpfad:** Robotertaster > Konfiguration > Gripper-/SpotTech > Parameter

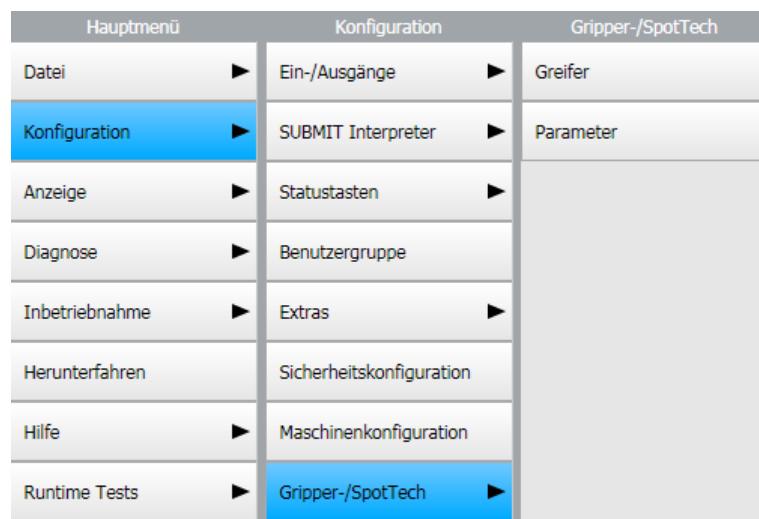


Abb. 10-12: Menüpfad

- Greiferparameter

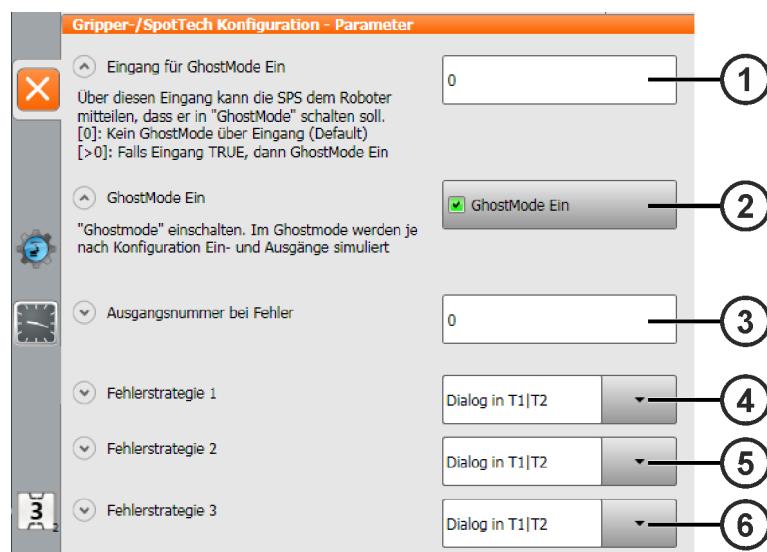


Abb. 10-13: Greifer-Parameter

Parameter	Beschreibung
Eingang für GhostMode Ein (1)	Nummer des Eingangs, über den der GhostMode eingeschaltet wird <ul style="list-style-type: none"> – 0: GhostMode wird nicht über einen Eingang geschaltet. – 1 ...4096/8192: Wenn Eingang TRUE ist, dann ist der GhostMode aktiv. Default: 0
GhostMode Ein (2)	Schaltet den GhostMode ein (nicht über Eingang) <ul style="list-style-type: none"> – Mit Häkchen: GhostMode ist eingeschaltet. – Ohne Häkchen: GhostMode ist ausgeschaltet. Default: GhostMode ist ausgeschaltet.

Parameter	Beschreibung
Ausgangsnummer bei Fehler (3)	Nummer des Ausgangs der bei einem Fehler gesetzt wird (Fehlerinformation an übergeordnete Steuerung). <ul style="list-style-type: none"> – 0: Es wird kein Ausgang gesetzt. – 1 ... 4096/8192: Ausgang wird mit dieser Nummer gesetzt. Default: 0
Fehlerstrategie 1...3 (4-6)	Es können 3 Fehlerstrategien konfiguriert werden, die später in den Inline-Formularen mit Greiferüberwachung ausgewählt werden können. Die ausgewählten Fehlerstrategien werden ausgeführt, wenn der Schaltzustand nicht vorliegt.

10.5 Greifer mittels Inlineformularen programmieren

Beschreibung

Mit dem Technologiepaket KUKA.GripperTech ist es möglich, Greiferbefehle über vorgefertigte Inline-Formulare direkt im angewählten Programm zu programmieren. Zwei Befehle zum Ansteuern des Greifers stehen dabei zur Verfügung.

- **Gripper SET**

Befehl zum Öffnen und Schließen des College-Greifers im Programm.

- **Gripper SYN SET**

Befehl zum Öffnen und Schließen des College-Greifers im Programm.

Zusätzlich kann der Schaltpunkt zeitlich sowie bahnbezogen verschoben werden.

Funktionen der Greiferprogrammierung

Es ist grundsätzlich möglich den Greiferbefehl so zu programmieren, dass der Befehl relativ zum Start- oder zum Zielpunkt ausgeführt wird.

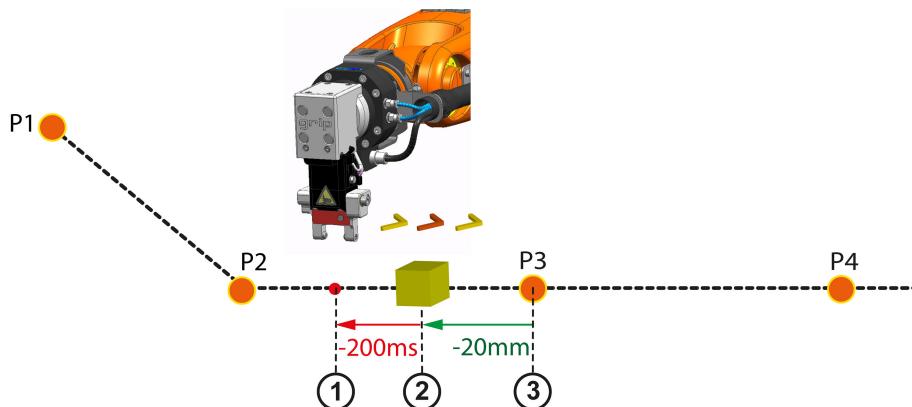


Abb. 10-14: Beispiel Greiferfunktionen mit örtlicher und zeitlicher Verschiebung

Pos.	Beschreibung
1	Greifer öffnen
2	Position des Greifers
3	Programmierter Punkt

- Im Beispiel befindet sich ein aufzunehmender Würfel (2) 20mm vor dem programmierten Punkt P3 (3).
- 200ms vor Erreichen des Würfels (1), soll der Greifer des Roboters geöffnet werden.



WARNUNG

Ein Greiferbefehl mit Bearbeitung während der Bewegung ist sorgfältig auszuwählen, da es bei unbedachter Verwendung zu Personen- oder Sachschaden durch fliegende Teile oder Kollision kommen kann!

Vorgehensweise

1. Im angewählten Programm die Programmzeile markieren, unter der der Greiferbefehl eingefügt werden soll.

```

1 INI
2
3 SPTP HOME Vel=100% % DEFAULT
4 SLIN P1 Vel=2m/s CPDAT1 Tool[1] Base[1]
5 SLIN P2 Vel=0.2m/s CPDAT2 Tool[1] Base[1]6
6 ;### Hier Greiferbefehl einfügen ###
7 SLIN P3 Vel=0.2m/s CPDAT3 Tool[1] Base[1]
8 SLIN P4 Vel=2m/s CPDAT3 Tool[1] Base[1]
9 ...

```

2. Menüpfad: Befehle > GripperTech > Gripper....

Den gewünschten Greiferbefehl einfügen.

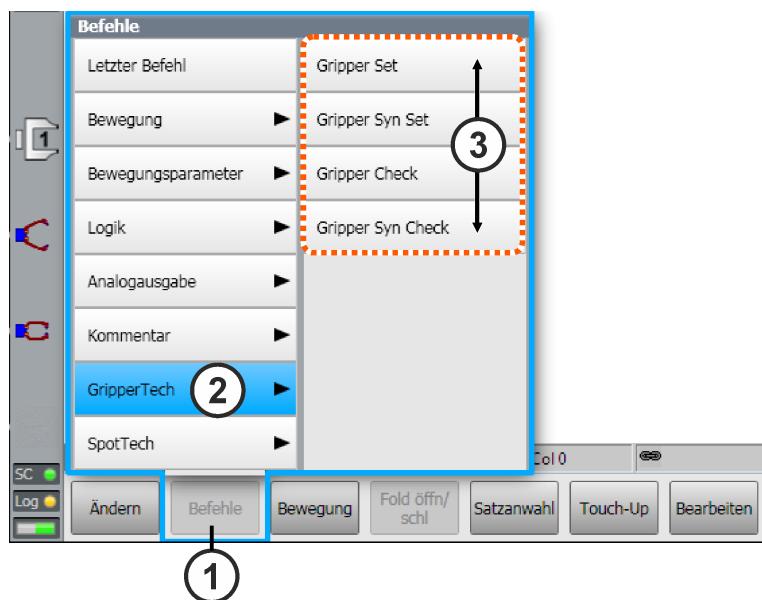


Abb. 10-15: Greifer Befehle

3. Im Inline-Formular den gewünschten Zustand des Greifers, die Wartezeit oder eine Fehlerstrategie einstellen.

Beispiel: **Gripper SET**

```

1 INI
2
3 SPTP HOME Vel=100% % DEFAULT
4 SLIN P1 Vel=2m/s CPDAT1 Tool[1] Base[1]
5 SLIN P2 Vel=0.2m/s CPDAT2 Tool[1] Base[1]
6 ; ### Greiferbefehl ###

```

```

7 Gripper SET [1]Gripper State=[1]OPEN Wait 0.2[s]Check
with No Check
7 SLIN P3 Vel=0.2m/s CPDAT3 Tool[1] Base[1]
8 SLIN P4 Vel=2m/s CPDAT3 Tool[1] Base[1]
9 ...

```

Gripper SET

- Über das Inline-Formular wird der Schaltzustand des Greifers gesetzt.
- Der Schaltzustand kann im Hauptlauf als auch im Vorlauf geschalten werden.
- Bei der Verwendung im Hauptlauf kann nach der Greiferaktion eine festgelegte Zeit gewartet werden, bis der nächste Bewegungssatz angefahren wird.

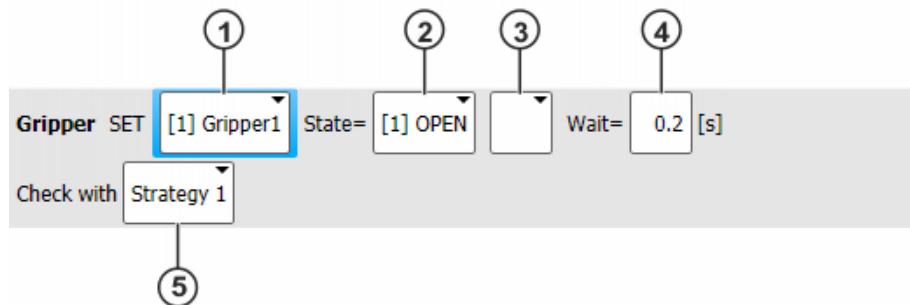


Abb. 10-16: Inline-Formular Gripper SET

Pos.	Beschreibung
1	Aktive Greifer Nur aktive Greifer werden angezeigt.
2	Schaltzustand des Greifers, der gesetzt wird. Die Anzahl der auswählbaren Schaltzustände und ihre Bezeichnung ist abhängig von der Konfiguration.
3	<ul style="list-style-type: none"> CONT: Der Schaltzustand wird über den Vorlauf gesetzt. [/er]: Der Schaltzustand wird mit dem Hauptlauf gesetzt.
4	Wartezeit Wenn die angegebene Wartezeit abgelaufen ist, wird das Programm fortgesetzt. <ul style="list-style-type: none"> 0.0 ... 10.0 s Default: 0.2 s Dieses Feld wird nur angezeigt, wenn mit dem Hauptlauf geschalten wird.
5	Fehlerstrategie Wenn ein Zustand nicht vorliegt, kann hierfür eine Fehlerstrategie eingestellt werden. <ul style="list-style-type: none"> No Check: Programm wird fortgesetzt, ohne zu prüfen, ob der Schaltzustand vorliegt. Strategy 1...3: Konfigurierte Fehlerstrategien

Gripper SYN SET

- Mit dem Inline-Formular kann am Start- oder am Zielpunkt der Bewegung ein Schaltzustand des Greifers gesetzt werden.

- Der Schaltzustand kann zeitlich und/oder örtlich verschoben werden.

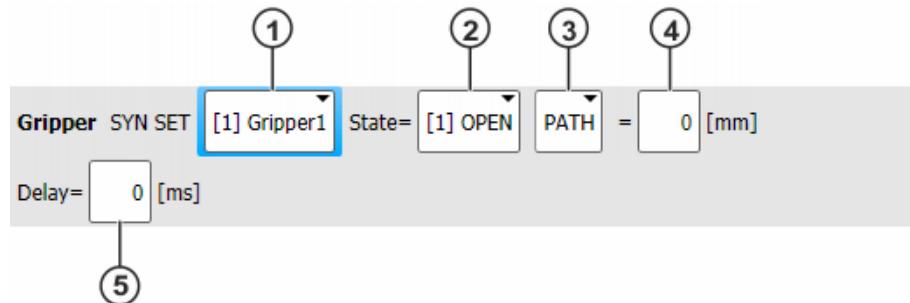


Abb. 10-17: Inline-Formular Gripper SYN SET

Pos.	Beschreibung
1	Aktive Greifer Nur aktive Greifer werden angezeigt.
2	Schaltzustand des Greifers, der gesetzt wird Die Anzahl der auswählbaren Schaltzustände und ihre Bezeichnung sind abhängig von der Konfiguration.
3	Punkt, auf den sich Gripper SYN SET bezieht <ul style="list-style-type: none"> START: Startbezugspunkt der zeitlichen Verschiebung END: Zielbezugspunkt der zeitlichen Verschiebung PATH: Setzen des Schaltzustands bezieht sich auf den Zielpunkt der Bewegung. Zusätzlich zur örtlichen Verschiebung ist eine zeitliche Verschiebung möglich.
4	Dieses Feld wird nur angezeigt, wenn PATH ausgewählt ist. Entfernung des Schaltpunkts vom Zielpunkt <ul style="list-style-type: none"> -2 000 ... +2 000 mm
5	Zeitliche Verschiebung der Schaltaktion <ul style="list-style-type: none"> -1000 ... 1000 ms Die Zeitangabe ist absolut. Der Schaltpunkt ändert sich je nach Geschwindigkeit des Roboters.

Beispiel SYN SET

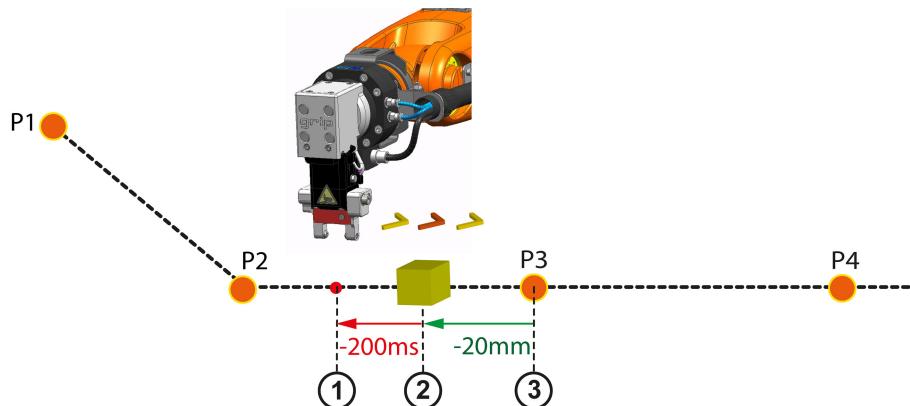


Abb. 10-18: Beispiel Greiferfunktionen mit örtlicher und zeitlicher Verschiebung

```

1 INI
2
3 SPTP HOME Vel=100% % DEFAULT
4 SLIN P1 Vel=2m/s CPDAT1 Tool[1] Base[1]
5 SLIN P2 Vel=0.2m/s CPDAT2 Tool[1] Base[1]
6 ; *** Greifer öffnen ***
7 Gripper Syn SET [1]Gripper State=[1]OPEN Path=-20mm
Delay=-200ms
8 ; *** Greifer schließen ***
9 Gripper Syn SET [1]Gripper State=[1]CLOSE Path=-20mm
Delay=0ms
10 SLIN P3 Vel=0.2m/s CPDAT3 Tool[1] Base[1]
16 ...

```

10.5.1 Greiferzustände mittels Inlineformular prüfen

Beschreibung

Mit dem Technologiepaket KUKA.GripperTech ist es möglich, Greiferzustände über vorgefertigte Inline-Formulare direkt im angewählten Programm zu prüfen.

Zwei Befehle stehen dabei zur Verfügung

- **Gripper CHECK**

Die Prüfung des Schaltzeitpunktes zum Haupt- oder im Vorlauf.

- **Gripper SYN CHECK**

Die Prüfung des Schaltzeitpunktes kann zeitlich, sowie örtlich verschoben werden.

Vorgehensweise

Gripper CHECK

Über das Inline-Formular wird der Schaltzustand des Greifers geprüft.

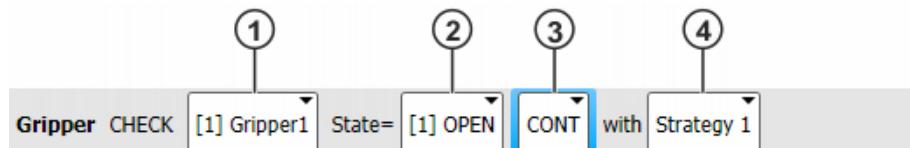


Abb. 10-19: Inline-Formular Gripper CHECK

Pos.	Beschreibung
1	Aktive Greifer Nur aktive Greifer werden angezeigt.
2	Schaltzustand des Greifers, der geprüft wird Die Anzahl der auswählbaren Schaltzustände und ihre Bezeichnung ist abhängig von der Konfiguration.
3	<ul style="list-style-type: none"> • CONT: Der Schaltzustand wird über den Vorlauf geprüft. • [/er]: Der Schaltzustand wird mit dem Hauptlauf geprüft.

Pos.	Beschreibung
4	<p>Fehlerstrategie</p> <p>Wenn ein Zustand nicht vorliegt, kann hierfür eine Fehlerstrategie eingestellt werden.</p> <ul style="list-style-type: none"> • No Check: Programm wird fortgesetzt, ohne zu prüfen, ob der Schaltzustand vorliegt. • Strategy 1...3: Konfigurierte Fehlerstrategien

Gripper SYN CHECK

- Mit dem Inline-Formular kann am Start- oder am Zielpunkt der Bewegung ein Schaltzustand des Greifers geprüft werden.
- Die Prüfung des Schaltzustands kann zeitlich und/oder örtlich verschoben werden.

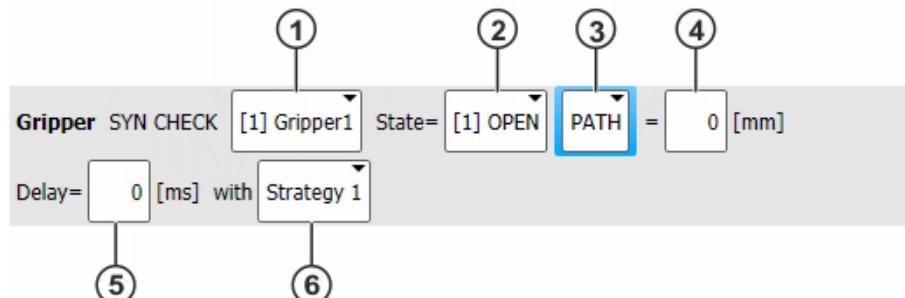


Abb. 10-20: Inline-Formular Gripper SYN CHECK

Pos.	Beschreibung
1	Aktive Greifer Nur aktive Greifer werden angezeigt.
2	Schaltzustand des Greifers, der geprüft wird Die Anzahl der auswählbaren Schaltzustände und ihre Bezeichnung sind abhängig von der Konfiguration.
3	Punkt, auf den sich Gripper SYN CHECK bezieht <ul style="list-style-type: none"> • START: Zeitliche Verschiebung in Bezug zum Startpunkt • END: Zeitliche Verschiebung in Bezug zum Zielpunkt • PATH: Prüfung bezieht sich auf den Zielpunkt der Bewegung. Neben der zeitlichen ist auch eine örtliche Verschiebung möglich.
4	Dieses Feld wird nur angezeigt, wenn PATH ausgewählt ist. Entfernung des Schaltpunkts vom Zielpunkt <ul style="list-style-type: none"> • -2 000 ... +2 000 mm
5	Zeitliche Verschiebung der Prüfung <ul style="list-style-type: none"> • -1000 ... 1000 ms Die Zeitangabe ist absolut. Der Zeitpunkt der Prüfung ändert sich je nach Geschwindigkeit des Roboters.

Pos.	Beschreibung
6	<p>Fehlerstrategie</p> <p>Wenn ein Zustand nicht vorliegt, kann hierfür eine Fehlerstrategie eingestellt werden.</p> <ul style="list-style-type: none">• No Check: Programm wird fortgesetzt, ohne zu prüfen, ob der Schaltzustand vorliegt.• Strategy 1...3: Konfigurierte Fehlerstrategien

10.5.2 Übung: Greiferprogrammierung Schild

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Greiferprogrammierung Schild**

10.5.3 Übung: Greiferprogrammierung Stift

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- Greiferprogrammierung Stift

11 Konfiguration und Programmierung von externen Werkzeugen

11.1 Lerneinheit: Konfiguration und Programmierung von externen Werkzeugen

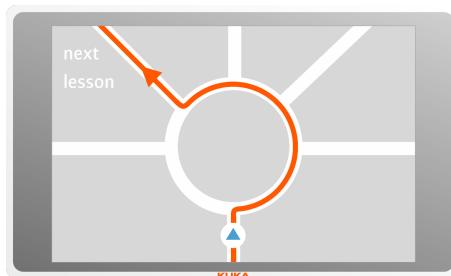


Abb. 11-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Inbetriebnehmen des Roboters
 - Vermessung eines feststehenden Werkzeugs
 - Vermessung eines robotergeführten Werkstücks
- Handverfahren mit einem feststehenden Werkzeug
- Bewegungsprogrammierung mit einem externen TCP

11.2 Robotergeführtes Werkstück und feststehendes Werkzeug

Beschreibung

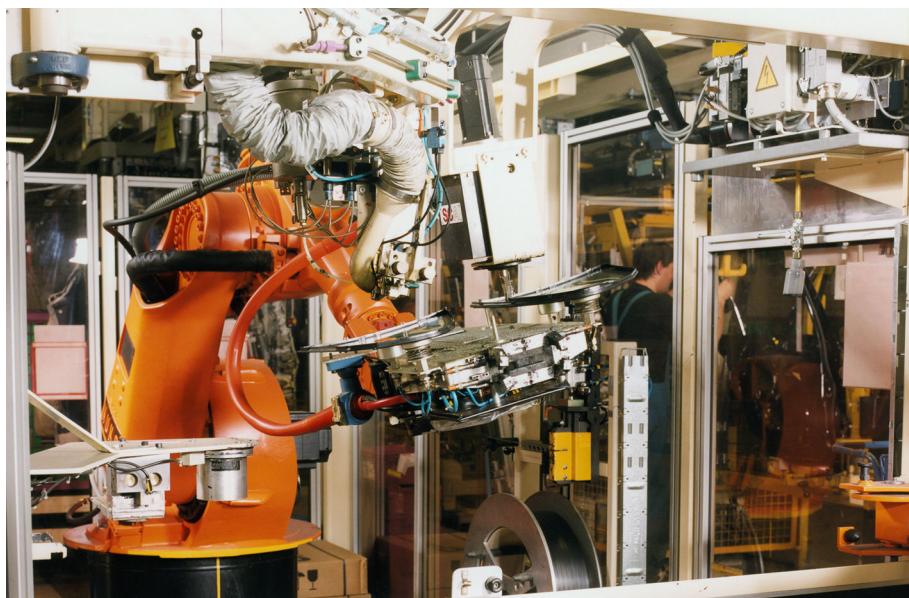


Abb. 11-2: Kleben von Seitenscheiben



In diesem Beispiel werden Seitenscheiben von einem Roboter geführt und der Kleber an einem feststehenden Werkzeug aufgetragen.

- Aufgrund von prozessspezifischen Gründen kann es notwendig sein,

- dass das zu bearbeitende Werkstück vom Roboter geführt werden muss, um z. B. einen optimierten Bewegungsablauf zu erreichen.
- dass das Werkzeug, z. B. aufgrund der Baugröße, stationär aufgestellt werden muss.
- **Durch die Verwendung eines feststehenden Werkzeugs und einem robotergeführten Werkstücks**
 - spielt das Werkzeuggewicht keine Rolle mehr.
 - ist die Medienversorgung wesentlich einfacher umzusetzen.
Es können auch Versorgungsleitungen verwendet werden, die nicht kabelschlepptauglich sind.
- **Folgende Vermessungsschritte sind notwendig:**
 - Vermessung des feststehenden Werkzeugs
(>>> *11.2.1 "Vermessen eines feststehenden Werkzeugs" Seite 334*)
 - Vermessung des robotergeführten Werkstücks
(>>> *11.2.2 "Vermessen eines robotergeführten Werkstücks" Seite 339*)

11.2.1 Vermessen eines feststehenden Werkzeugs

Beschreibung

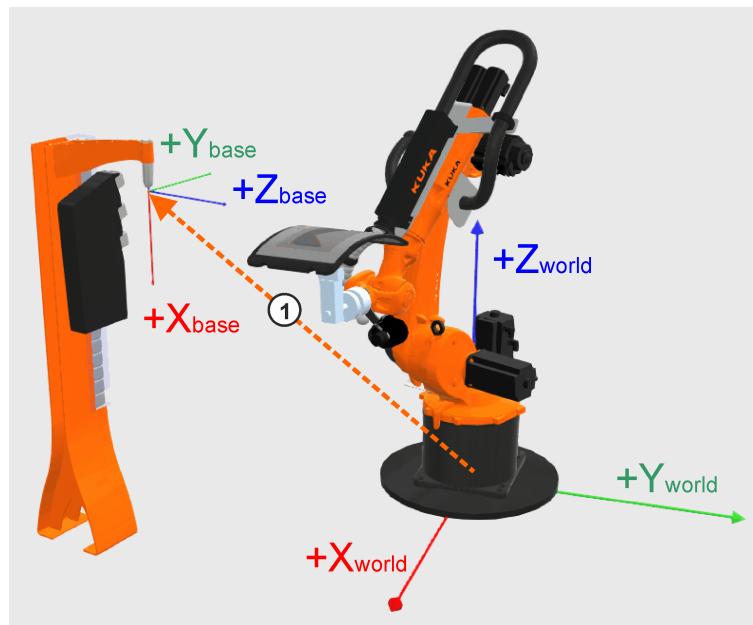


Abb. 11-3: Grundlagen Vermessung des feststehenden Werkzeugs

- 1 Abstand (XYZ) und Ausrichtung (ABC) des externen TCP in Bezug auf das Weltkoordinatensystem



Der externe TCP (1) bezogen auf das Welt-Koordinatensystem (\$WORLD) wird als Base-Koordinatensystem verwaltet.

Die Vermessung des feststehenden (externen) Werkzeugs besteht aus zwei Schritten:

1. Ermittlung der Position des externen TCPs zum Ursprung des Weltkoordinatensystems
2. Orientierung des Koordinatensystems am externen TCP

Voraussetzung

- Vor der Vermessung müssen die Richtungen des Flanschkoordinatensystems bekannt sein.
- Zur Vermessung des externen TCPs wird ein bereits vermessenes, robotergeführtes Werkzeug benötigt.
- Dies gilt insbesondere bei Verwendung der 6D-Methode, da sich +Z der Basis (des externen TCPs) aus +X Flansch ergibt.



Vor der Vermessung das Werkzeug 0 (Flansch) und das Werkzeugkoordinatensystem einstellen. In +X-Richtung verfahren und die Richtung am Flansch markieren.

Vorgehensweise

Ermittlung des externen TCPs

1. Ein neues feststehendes Werkzeug wurde über die Werkzeug- und Basisverwaltung hinzugefügt.
(>>> [4.8.1 "Neue Basis/festes Werkzeug hinzufügen" Seite 151](#))



Abb. 11-4: Feststehendes Werkzeug anlegen

2. Über die Schaltfläche **Vermessen** die **XYZ-Methode** auswählen.

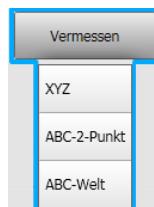


Abb. 11-5: Vermessungsmethode

3. **Externen TCP mit Referenzwerkzeug anfahren**

Mit der Werkzeugspitze des vermessenen, robotergeführten Werkzeugs, wird der externe TCP angefahren.



Abb. 11-6: Erster Punkt: Ursprung, Koordinatensystem, externer TCP

4. Externen TCP teachen



Abb. 11-7: TCP vermessen

- Das Referenzwerkzeug im PullDown-Fenster (1) auswählen.
- Wenn der TCP des Referenzwerkzeugs und der TCP des feststehenden Werkzeugs deckungsgleich sind, die Zeile Kalibrierungspunkt TCP (2) antippen.
Die Zeile wird blau hinterlegt.
- Mit der Schaltfläche Touch-Up (3) den Punkt aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.
- Der ermittelte Werkzeugoffset wird im Fenster **Vermessungsergebnis** (4) angezeigt.
- Die Werte mittels der gleichnamigen Schaltfläche **Speichern** (5).



Mittels der Schaltfläche **Anfahren (PTP)** können bereits geteachte Punkte wiederholt angefahren werden.

5. Lage des neuen feststehenden Werkzeugs

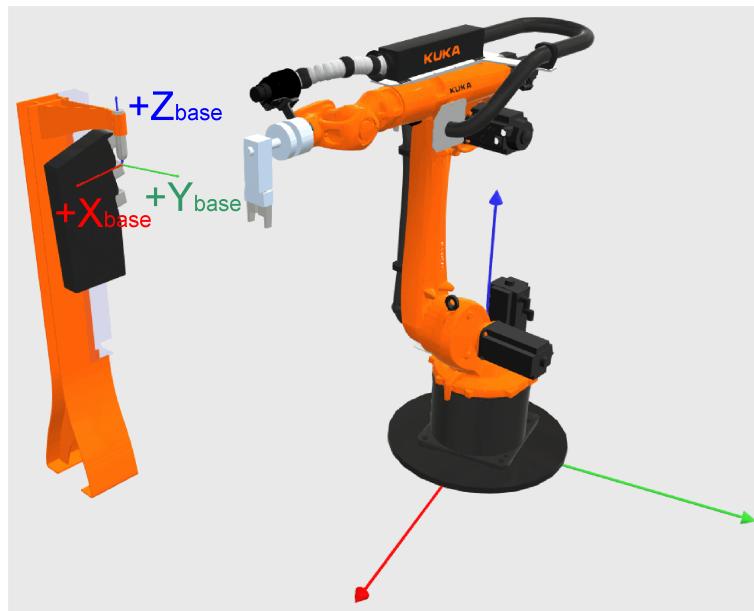


Abb. 11-8: Vermessenes, externes Werkzeug

Ermittlung der Orientierung

1. Ein neues feststehendes Werkzeug wurde über die Werkzeug- und Basisverwaltung hinzugefügt.
(>>> *4.8.1 "Neue Basis/festes Werkzeug hinzufügen" Seite 151*)
Der TCP des feststehenden Werkzeugs wurde ermittelt und gespeichert.
(>>> *"Ermittlung des externen TCPs" Seite 335*)
2. Das feststehende Werkzeug in der Werkzeugverwaltung öffnen.
Über die Schaltfläche **Vermessen** die **ABC-Welt** Methode auswählen.

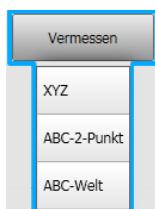


Abb. 11-9: Vermessungsmethode

3. Stoßrichtung festlegen

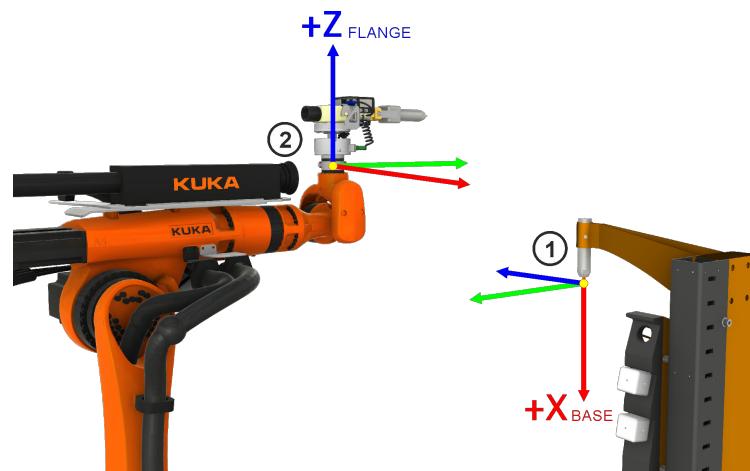


Abb. 11-10: Vermessung festes Werkzeug ABC-Welt-Methode 5D

- 1 +X-Festes Werkzeug
- 2 +Z-Flange

- Zur Ermittlung der Orientierung wird das FLANGE-Koordinatensystem parallel zum neuen Koordinatensystem ausgerichtet.
- Bei dieser Vermessung ist auf die Zentralhand zu achten. Bei vielen Robotertypen kann die Zentralhand als Orientierungshilfe verwendet werden.

4. Stoßrichtung teachen

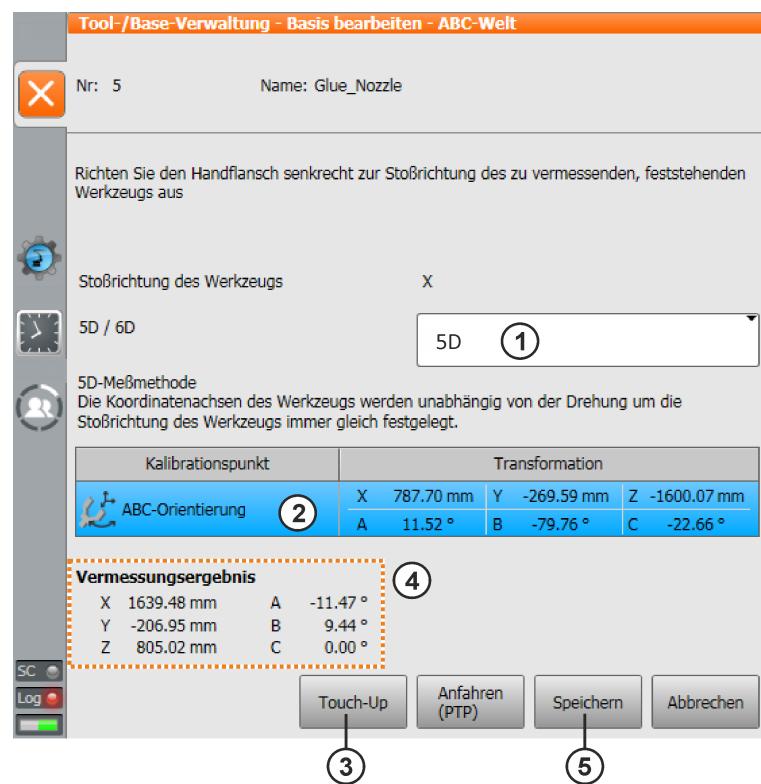


Abb. 11-11: Stoßrichtung des Werkzeugs

- Die Art der Vermessung im Auswahlfenster 5D/6D (1) festlegen
Es gibt 2 Varianten:
 - **5D:** Der Robotersteuerung wird nur die Stoßrichtung des feststehenden Werkzeugs bekanntgegeben. Die Stoßrichtung ist

standardmäßig die X-Achse. Die Orientierung der anderen Achsen wird vom System festgelegt und ist für den Benutzer nicht ohne Weiteres zu erkennen.

- **6D:** Der Robotersteuerung werden die Orientierungen aller 3 Achsen des Koordinatensystems bekanntgegeben.
- Die Zeile Kalibrierungspunkt, ABC-Orientierung (2) antippen.
Die Zeile wird blau hinterlegt.
- Die Roboterposition mittels der Schaltfläche Touch-Up (3) übernehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.
- Der ermittelte Werkzeugoffset wird im Fenster **Vermessungsergebnis** (4) angezeigt.
- Die Werte mittels der gleichnamigen Schaltfläche **Speichern** (5).



Mittels der Schaltfläche **Anfahren (PTP)** können bereits geteachte Punkte wiederholt angefahren werden.

11.2.2 Vermessen eines robotergerührten Werkstücks

Beschreibung

Vermessung des robotergerührten Werkstücks

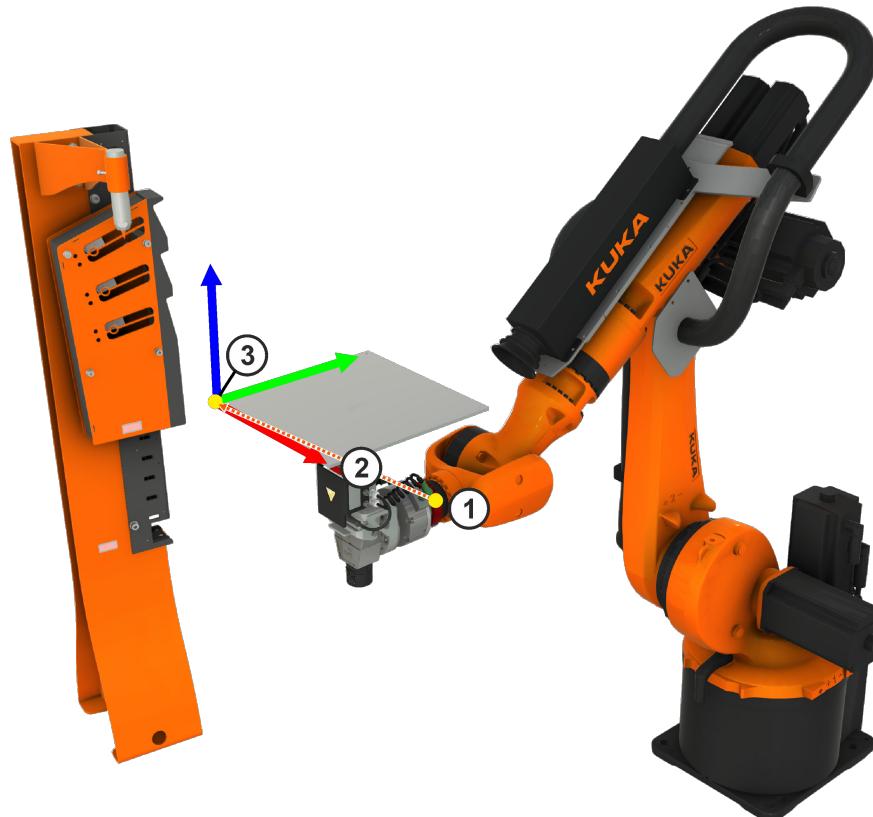


Abb. 11-12: Vermessung Werkstück über direkte Vermessung

- 1 Flanschmittelpunkt
- 2 Abstand zwischen 1 und 3 und Verdrehung
- 3 Koordinatensystemursprung vermessenes Werkstück

Expertendokumentation

Im Folgenden wird nur auf die direkte Vermessungsmethode eingegangen. Die indirekte Vermessung ist äußerst selten und wird in der Dokumentation *KUKA System Software 8.6* beschrieben.



Xpert Suche:	KUKA System Software 8.6
	Bedien- und Programmieranleitung für Systemintegratoren
Xpert Filter:	Dokumentation > Bedien- und Programmieranleitung
Kapitel:	Vermessen > Übersicht Vermessungsmethoden > Methode indirekt

Vorgehensweise

1. Das feststehende Werkzeug wurde vermessen.
(>>> *11.2.1 "Vermessen eines feststehenden Werkzeugs" Seite 334*)
2. Das für die Vermessung notwendige Werkstück mittels z. B. Greifer aufnehmen.
3. Ein neues robotergeführtes Werkstück wurde über die Werkzeug- und Basisverwaltung hinzugefügt.
(>>> *4.6.1 "Neues Werkzeug/Werkstück hinzufügen" Seite 125*)
4. Im Fenster Werkzeug bearbeiten im PullDown-Fenster **Werkstück** auswählen.



Abb. 11-13: Robotergeführtes Werkstück definieren

5. Über die Schaltfläche **Vermessen** die Methode **3-Punkt** auswählen.



Abb. 11-14: Basis vermessen, Auswahl

6. **Koordinatenursprung anfahren**

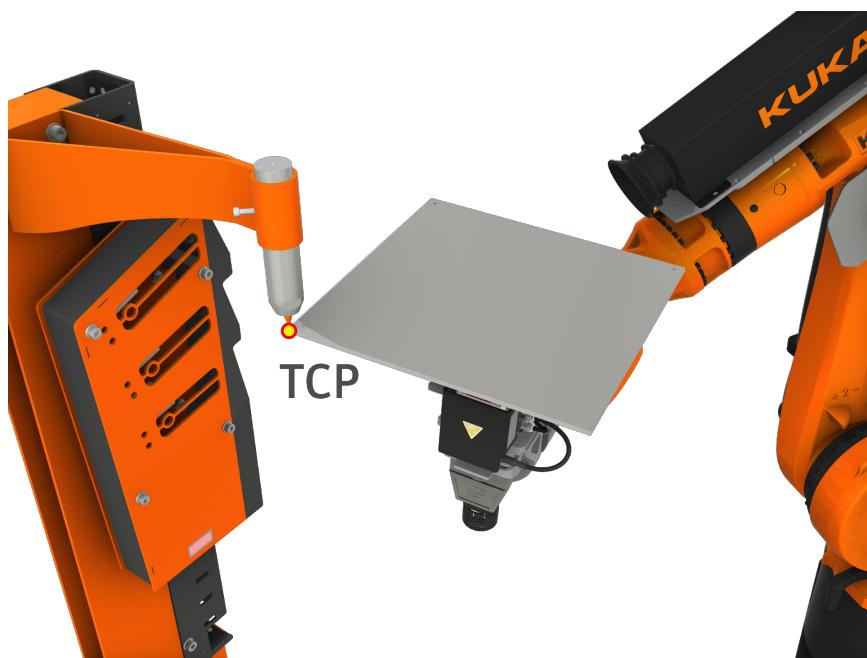


Abb. 11-15: Koordinatenursprung, Werkstück

- Mit dem Werkstück an den TCP des feststehenden Werkzeugs fahren.
- Hierbei ist der Ursprung des zu definierenden Koordinatensystems mit dem TCP des feststehenden Werkzeugs in Übereinstimmung zu bringen.

7. Koordinatenursprung vermessen

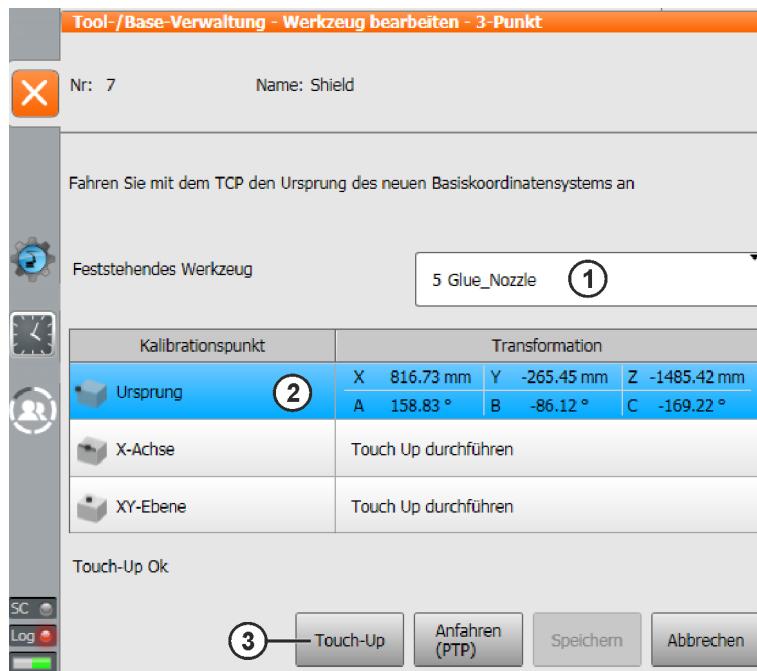


Abb. 11-16: Koordinatenursprung vermessen

- Das feststehende Werkzeug im PullDown-Fenster (1) auswählen.
- Wenn der Koordinatenursprung und der TCP des feststehenden Werkzeugs deckungsgleich sind, die Zeile Kalibrierungspunkt Ursprung (2) antippen.
- Mit der Schaltfläche Touch-Up (3) den Punkt aufnehmen.

Die ermittelten Werte werden im Feld Transformation (2) angezeigt.

8. Punkt auf der X-Orientierung anfahren

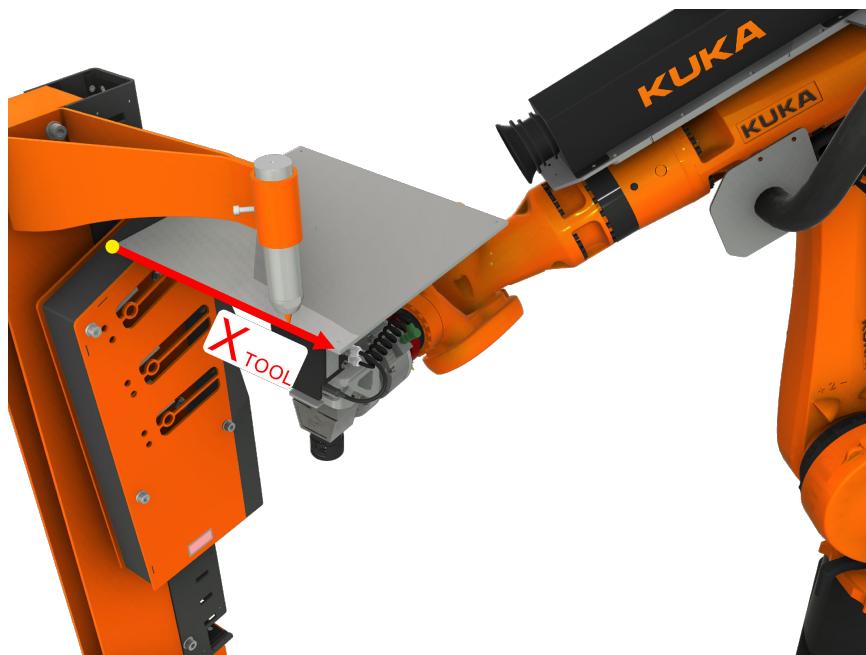


Abb. 11-17: X-Orientierung

- Mit dem TCP des feststehenden Werkzeugs einen Punkt auf der positiven X-Achse des zu definierenden Werkstück-Koordinatensystems anfahren.

9. Punkt auf der X-Orientierung teachen

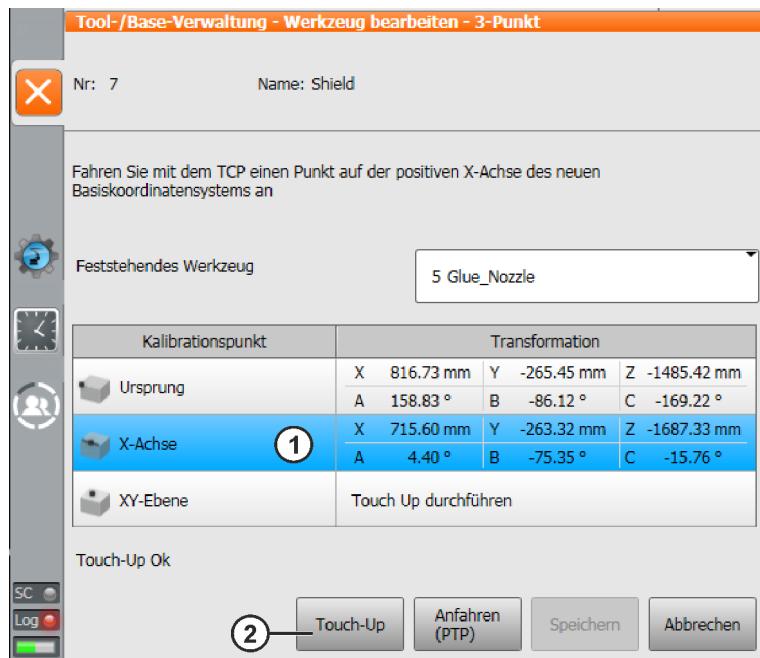


Abb. 11-18: X-Achse vermessen

- Die Zeile Kalibrierungspunkt X-Achse (2) antippen.
Die Zeile wird blau hinterlegt.
- Mit der Schaltfläche Touch-Up (2) den Punkt aufnehmen.
Die ermittelten Werte werden im Feld Transformation (2) angezeigt.

10. Punkt auf der XY-Ebene anfahren

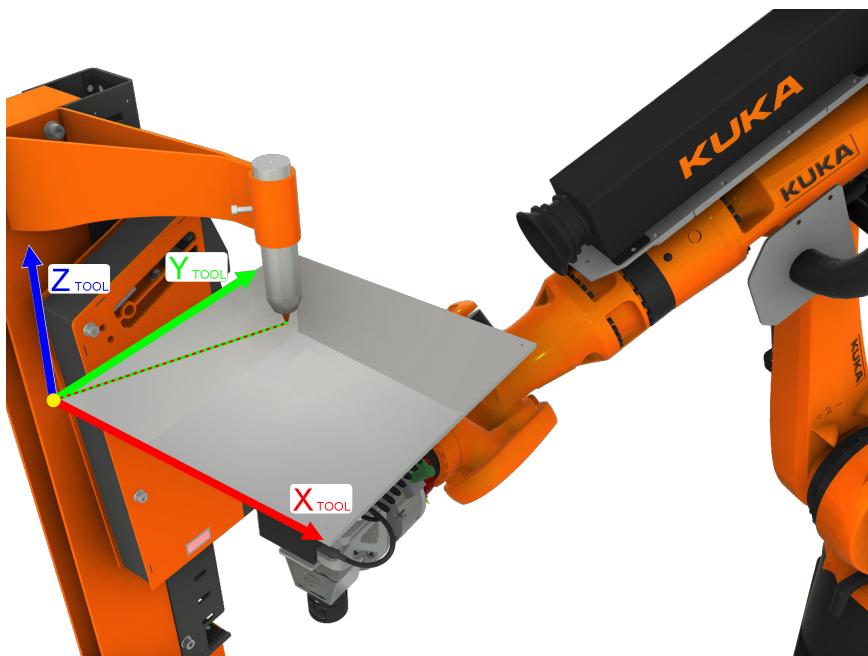


Abb. 11-19: XY-Ebene

- Mit dem TCP des feststehenden Werkzeugs einen Punkt auf der XY-Ebene des zu definierenden Werkstück-Koordinatensystems anfahren.

11. Punkt auf der XY-Ebene teachen

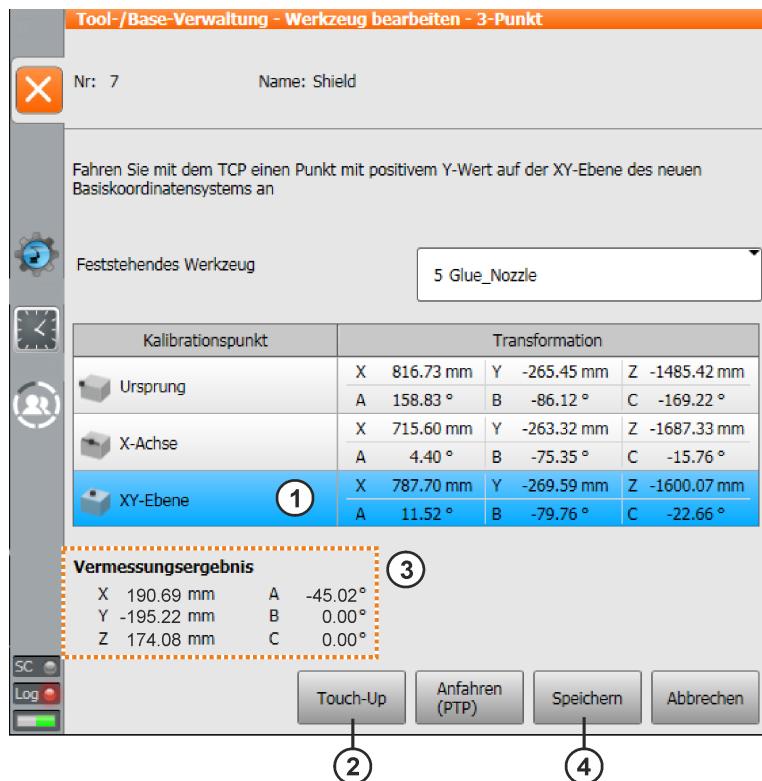


Abb. 11-20: XY-Ebene vermessen

- Auf die Zeile **XY-Ebene** (2) tippen.
- Die Position mittels der Schaltfläche **Touch-Up** (2) aufnehmen.
Die ermittelten Werte werden im Feld Transformation angezeigt.

- Die Vermessung der Basis ist durch Aufnahme des letzten Punkts abgeschlossen. Die Offset-Werte werden im Feld **Vermessungsergebnis** (3) angezeigt.
- Offset-Werte und Verdrehung des Bezugspunkts mit der gleichnamigen Schaltfläche **Speichern** (4).

11.2.3 Übung: Externes Werkzeug und robotergeführtes Werkstück vermessen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Externes Werkzeug und robotergeführtes Werkstück vermessen**

11.3 Handverfahren mit einem feststehenden Werkzeug

Beschreibung

Vorteile und Anwendungsbereiche

Manche Produktions- und Bearbeitungsprozesse erfordern es, dass der Roboter das **Werkstück** (Bauteil) (1) und nicht das **Werkzeug** (2) handhabt. In unserem Beispiel wird Klebstoff auf ein Glas für eine Taucherbrille aufgetragen.

Der Vorteil liegt darin, dass das Bauteil nicht erst zur Bearbeitung abgelegt werden muss - somit können Spannvorrichtungen eingespart werden.

Dies ist u. a. der Fall bei:

- Klebeapplikationen
- Schweißapplikationen

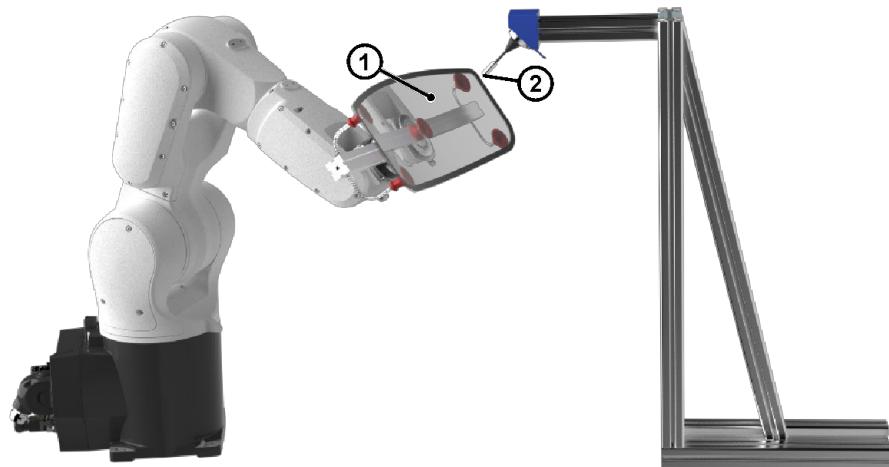


Abb. 11-21: Beispiel feststehendes Werkzeug

HINWEIS

Um eine solche Applikation erfolgreich zu programmieren, muss sowohl der externe Tool Center Point des feststehenden Werkzeugs als auch das Werkstück vermessen werden.

Geänderter Bewegungsablauf bei feststehendem Werkzeug

Robotergeführtes Werkzeug

In den meisten Fällen führt der Roboter das Werkzeug und bearbeitet oder befördert Werkstücke. Für jeden programmierten Bewegungssatz (teachen) wird der Abstand vom aktiven Basekoordinatensystem zum aktiven Tool-Koordinatensystem (TCP) gespeichert. Die hinterlegte Bewegung (Interpolation) bezieht sich hierbei immer auf den TCP des robotergeführten Werkzeugs.

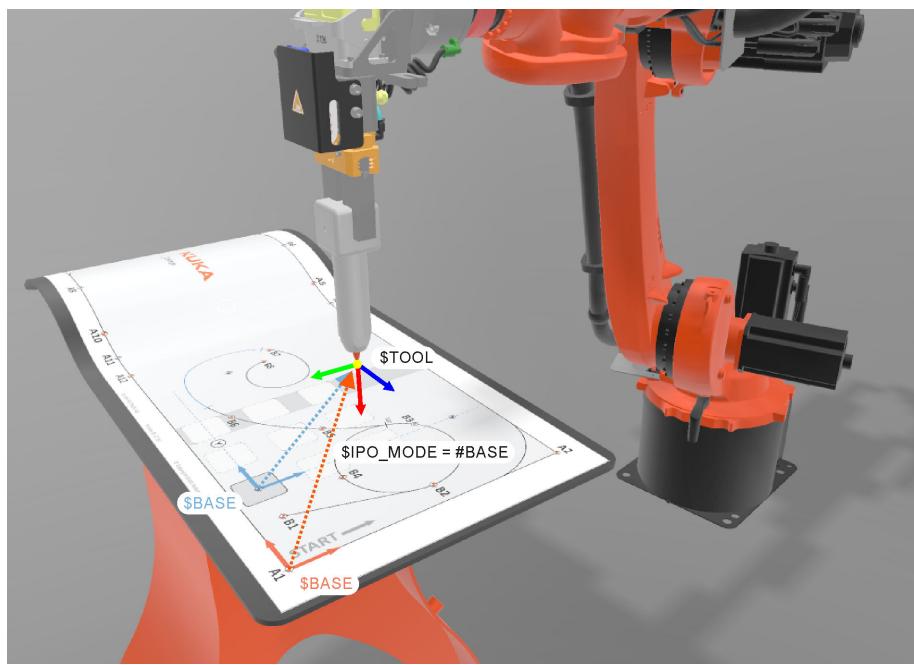


Abb. 11-22: \$IPO_MODE = #BASE

Robotergeführtes Werkstück/ Feststehendes Werkzeug

Bei einer Applikation mit feststehendem, nicht beweglichen Werkzeug wird das Werkstück vom Roboter geführt. Das Werkzeug ist in seiner Position unveränderlich und wird innerhalb der Steuerung als Basis gespeichert. Das Werkstück wird vom Roboter geführt und wird als Werkzeug verwaltet. Werkzeug und Basis sind jedoch in ihrer Rolle vertauscht. Der Steuerung muss mittels der Variablen \$IPO_MODE = #TCP im Bewegungssatz, als auch im Handverfahrbetrieb mitgeteilt werden, dass sich die Bewegung (z. B. SLIN) auf den sogenannten **externen Tool Center Point** bezieht. Das Verfahren entlang von Konturen des robotergeführten Werkstücks ist somit problemlos möglich.

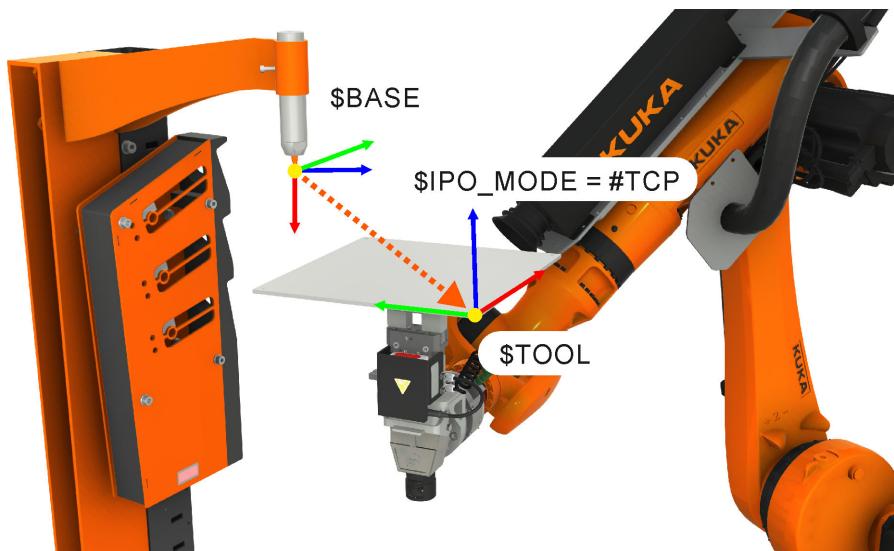


Abb. 11-23: \$IPO_MODE = #TCP

Vorgehensweise

Handverfahren mit feststehendem Werkzeug

1. Das Fenster aktuelle Basis/ Werkzeug (1) auf dem smartPAD öffnen.

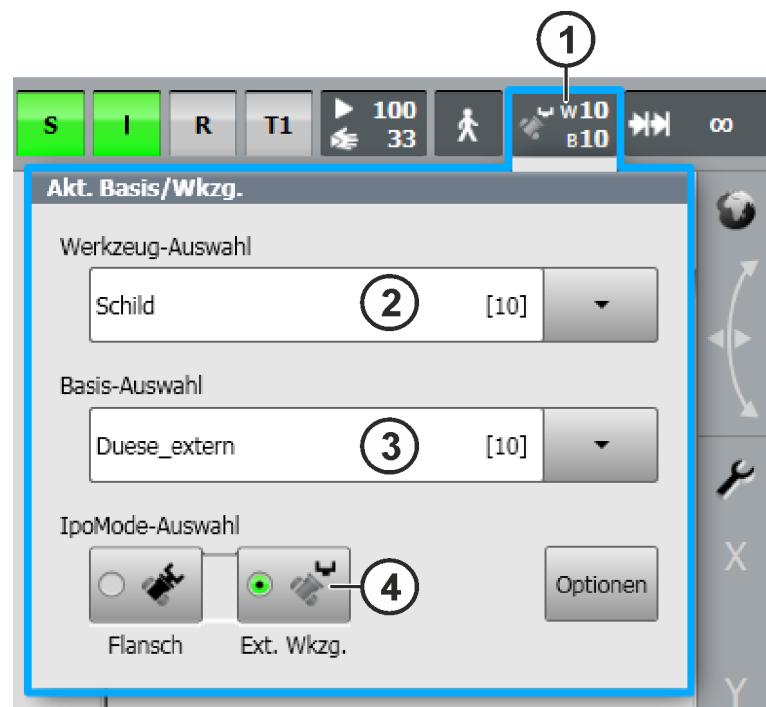
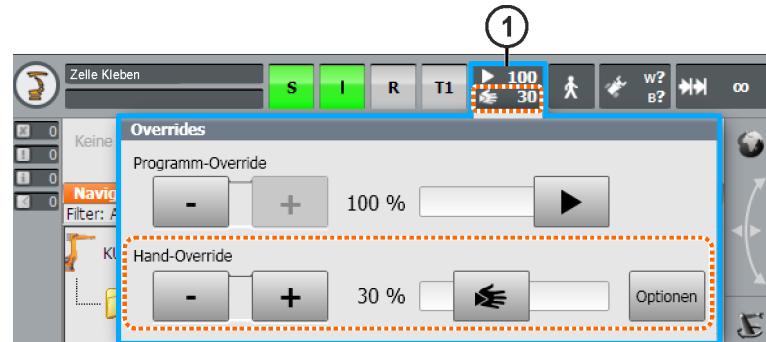


Abb. 11-24: Auswahl ext. TCP im Optionsmenü

- Das vom Roboter geführte Werkstück im Fenster Werkzeug-Auswahl (2)aktivieren.
 - Das feststehendes Werkzeug im Fenster Basis-Auswahl (3) aktivieren.
 - Die IpoMode-Auswahl auf **Externes Werkzeug** (4)setzen.
2. Den Hand-Override einstellen (1).



3. Nach eigener Preferenz das Handverfahren auf dem smartPAD einstellen.

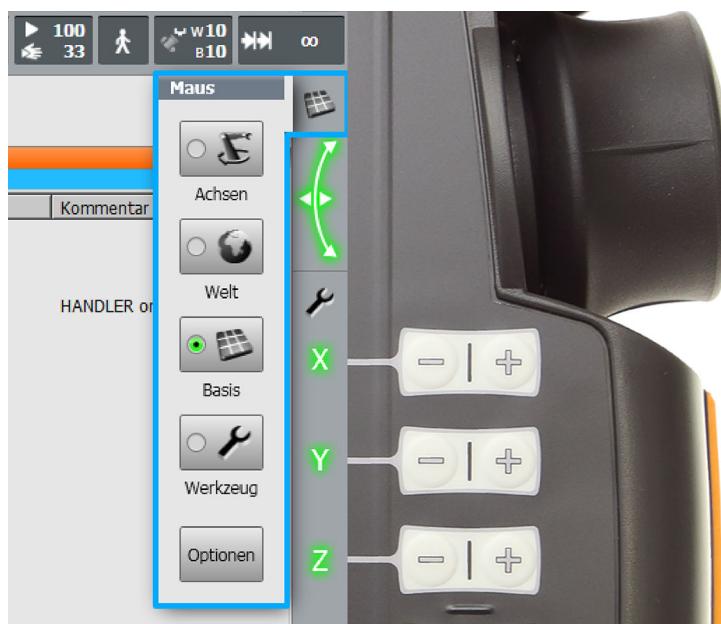


Abb. 11-25: Handverfahren, externes Werkzeug, 6D-Maus

- Werkzeug einstellen um im Koordinatensystem des Werkstückes zu verfahren,
 - Basis einstellen um im Koordinatensystem des Ext. Werkzeugs zu verfahren,
4. Einen der vier Zustimmmtaster auf Mittelstellung drücken und halten.



5. Verfahren mit den Verfahrtasten/Space Mouse in die gewünschte Richtung.



Abb. 11-26: Handverfahren

Durch die Auswahl **Ext. Wkzg.** im Optionsfenster **Handverfahroptionen** schaltet die Steuerung um: alle Bewegungen erfolgen jetzt relativ zum externen TCP und nicht zu einem robotergeführten Werkzeug.

11.3.1 Übung: Handverfahren mit feststehendem Werkzeug

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Handverfahren mit feststehendem Werkzeug**

11.4 Bewegungsprogrammierung mit externem TCP

Bewegungsprogrammierung mit externem TCP

Bei der Bewegungsprogrammierung mit einem feststehenden Werkzeug ergeben sich für den Bewegungsablauf gegenüber einer Standardbewegung folgende Unterschiede:

- Kennzeichnung im Inline-Formular: Im Optionsfenster **Frames** muss der Eintrag **Externer TCP** auf TRUE stehen.

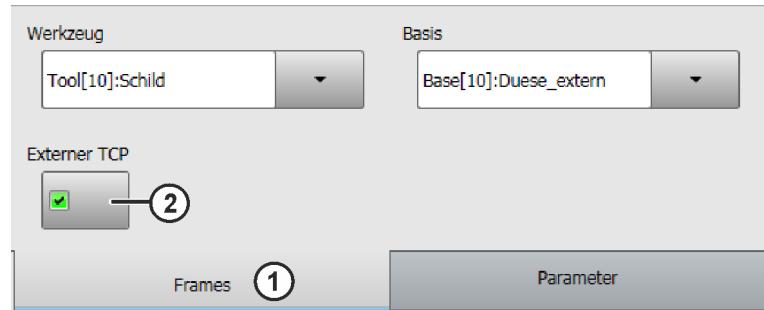


Abb. 11-27: Optionsfenster "Frames": ext. TCP

- Die **Bewegungsgeschwindigkeit** bezieht sich dann auf den externen TCP.
- Die **Orientierung** entlang der Bahn bezieht sich dann ebenfalls auf das externe Werkzeug.
- Es ist sowohl das passende Base-Koordinatensystem (feststehendes Werkzeug/externer TCP) als auch das passende Tool-Koordinatensystem (robotergeführtes Werkstück) anzugeben.

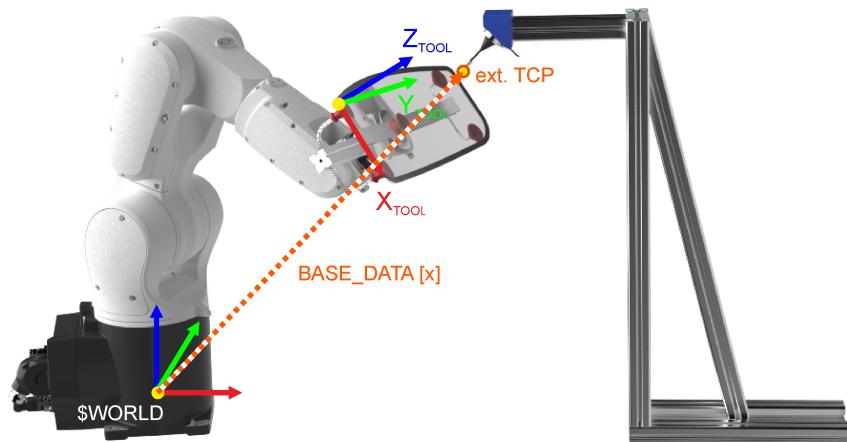


Abb. 11-28: Koordinatensysteme bei feststehendem Werkzeug

11.4.1 Übung: Bewegungsprogrammierung mit externen TCP

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Bewegungsprogrammierung mit externem TCP**

12 Einführung in die Expertenebene

12.1 Lerneinheit: Einführung in die Expertenebene

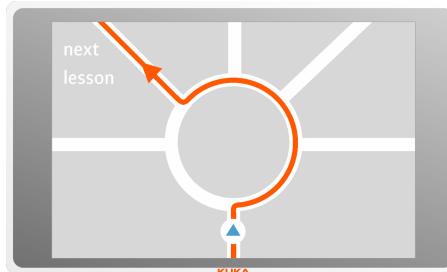


Abb. 12-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Programme erstellen mittels Templates
- Ansichten im Explorer anpassen
- Ansichten im Editor anpassen
- Fehler im Programm finden und beheben
- Roboterprogramme strukturieren

12.2 Programme erstellen mittels Templates

Beschreibung

- In der Benutzergruppe Experte werden beim der Neuerstellung eines Programms Templates zur Verfügung gestellt.
- Templates sind Programmrümpfe ("Baumuster"), die als Kopiervorlagen zur Erstellung von neuen Programmen verwendet werden.
- Folgende Templates stehen grundsätzlich zur Verfügung.

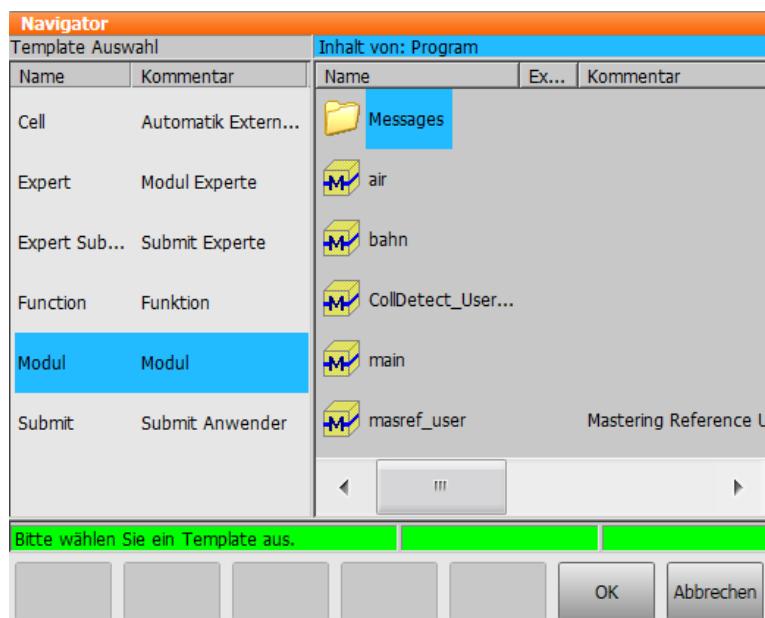


Abb. 12-2: Template Auswahl

- **Cell:** Vorhandenes Cell Programm kann nur ersetzt oder bei gelöschttem Cell neu erstellt werden.
- **Expert:** Modul bestehend aus SRC und DAT-Datei, in denen nur der Programmkopf und -ende vorhanden sind.
- **Expert Submit:** Zusätzliche Submitdatei (SUB) bestehend aus Programmkopf und -ende.
- **Function:** Funktionserstellung SRC, in dem nur der Funktionskopf mit einer BOOL Variablen angelegt wird. Das Ende der Funktion ist vorhanden, jedoch muss die Rückgabe programmiert werden.
- **Modul:** Modul bestehend aus SRC und DAT-Datei, in denen der Programmkopf, -ende und das Grundgerüst (INI und 2x SPTP HOME) vorhanden sind.
- **Submit:** Zusätzliche Submitdatei (SUB) bestehend aus Programmkopf, -ende und Grundgerüst (DECLARATION,INI,LOOP/ENDLOOP).

Templates modifizieren

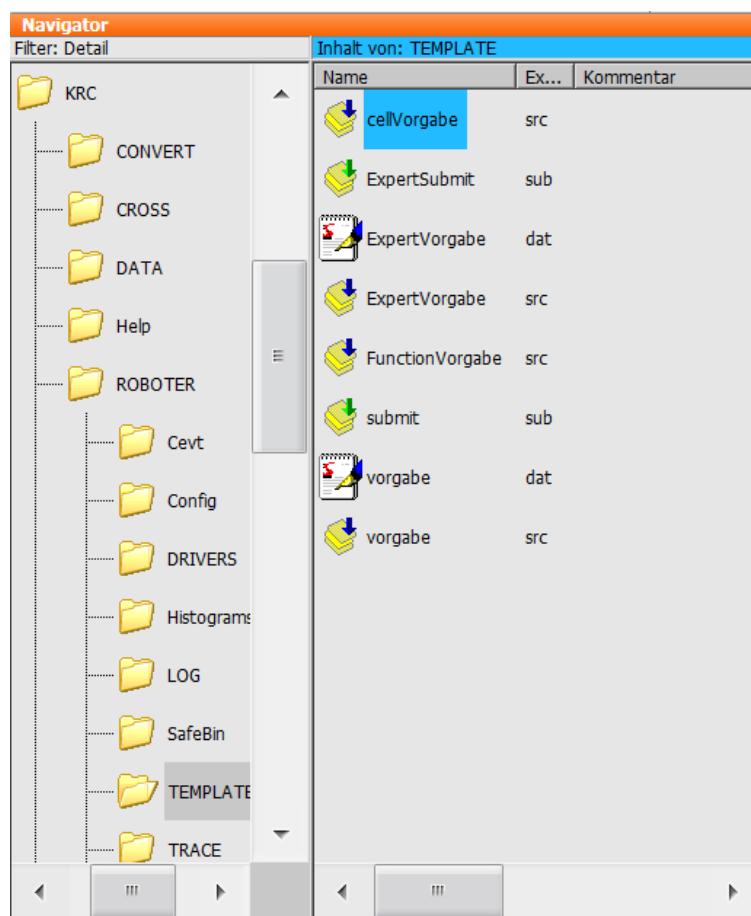


Abb. 12-3: Templates



Vorgabe \triangleq predefined

- Templates, die zur Programmerstellung zur Auswahl stehen, finden sich auf der Steuerung unter folgenden Dateipfad: C:\KRC\ROBOTER\TEMPLATE
- Templates sind nichts anderes als vordefinierte Programmrümpfe, die als Kopiervorlagen fungieren.
- Templates können wie "normale" Programme, Funktionen etc. modifiziert werden.

HINWEIS

Wird ein Template verändert, so wirkt sich diese Veränderung unmittelbar bei jeder Neuerstellung eines Programms aus! Auf bereits erstellte Programme hat dies keine Auswirkung.

12.3 Ansichten im Explorer anpassen

Beschreibung

Detailansicht nutzen

In der Benutzergruppe Experte kann neben der Modulansicht auch die Detailansicht genutzt werden.

- **Modulansicht**

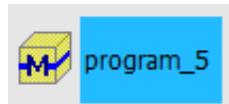


Abb. 12-4: Modulansicht

- In der Modulansicht wird ein Programm immer als ein Eintrag angezeigt.
- Das eigentliche Programm *.src und die dazugehörige Datenliste *.dat sind einzeln bearbeitbar.



Wird das Modul dupliziert, werden immer alle notwendigen Dateien mitgeführt.

- **Detailansicht**



Abb. 12-5: Detailansicht

- In der Detailansicht wird das eigentliche Programm *.src und die dazugehörige Datenliste *.dat getrennt dargestellt.
- Beide Dateien können getrennt voneinander direkt bearbeitet, kopiert, gelöscht und archiviert werden.



Wird das Modul dupliziert, muss darauf geachtet werden, dass immer beide Dateien *.src und *.dat mitgeführt werden.

Vorgehensweise

- Über die Schaltfläche Bearbeiten das Kontextmenü öffnen. Die Funktion Filter aufrufen.



Abb. 12-6: Bearbeiten, Filter

- Im eingeblendeten Auswahlfenster kann zwischen der Modul- und Detailansicht gewechselt werden.

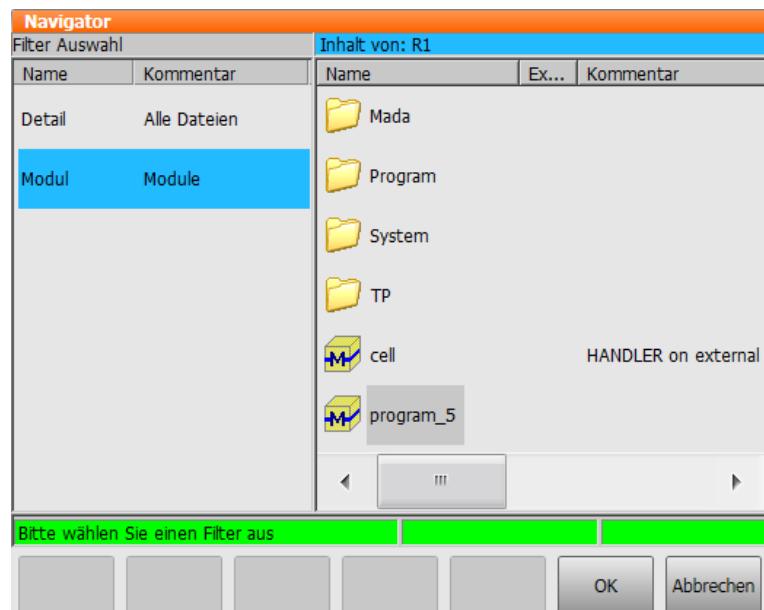


Abb. 12-7: Filter Auswahl

12.4 Ansicht im Editor anpassen

Beschreibung

- Die Ansicht im Editor kann hinsichtlich der Bedienung und Darstellung angepasst werden.
- Auch die Detailtiefe lässt sich an die Bedürfnisse des Nutzers anpassen.

Vorgehensweise

1. Ein Programm im smartPAD-Editor öffnen.

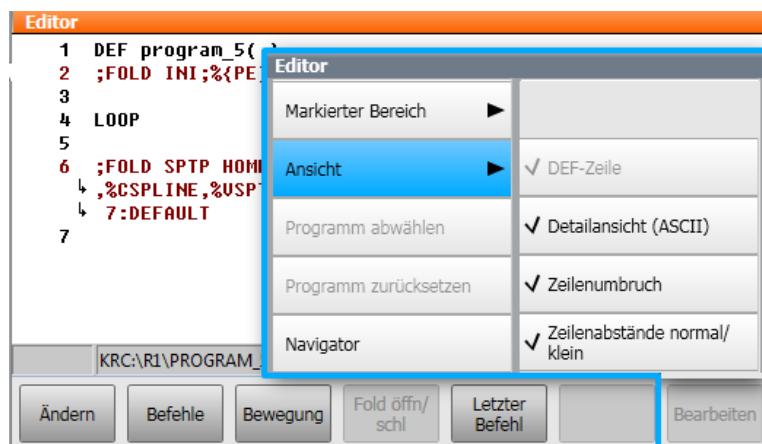


Abb. 12-8: Ansicht

Menüpfad: Schaltfläche *Bearbeiten* > *Ansicht*

- **DEF-Zeile**

Der Programmrumppf (Header und END) wird angezeigt.

```
DEF program()
...
END
```

- **Detailansicht (ASCII)**

Ausgeblendete ASCII-Zeichen werden eingeblendet.

```
...
;FOLD SPTP HOME VEL=100% DEFAULT; %{ PE }
%NJZJATOBASUS....
...
```

- **Zeilenumbruch**

Der Bildschirm ist in der Standardeinstellung in der Horizontalen scrollbar.

Ist der Zeilenumbruch aktiv, werden längere Programmzeilen alternativ umgebrochen dargestellt.

- **Zeilenabstände normal/klein**

Der Zeilenabstand zwischen den Programmzeilen ist größer oder kleiner.

2. **Beispiel:** Erweitere Ansicht im Editor mit DEF-Zeile, Detailansicht und Zeilenumbruch klein.

```

Editor
1 DEF program_5( )
2 ;FOLDINI; %{PE}
3
4 LOOP
5
6 ;FOLD SPTP HOME Vel=100 % DEFAULT; %{PE} %MKUKATPBASIS
↓ ,%CSPLINE,%USPTP_SB,%P 1:SPTP_SB, 2:HOME, 3:, 5:100,
↓ 7:DEFAULT
7
8 ;FOLD SLIN P1 Vel=0.5 m/s CPDAT2 Tool[1]:Greifer
↓ Base[1]:Basis_Tisch ; %{PE}
9 ;FOLD SLIN P2 Vel=0.5 m/s CPDAT3 Tool[1]:Greifer
↓ Base[1]:Basis_Tisch ; %{PE}
10 ;FOLD SLIN P3 Vel=0.5 m/s CPDAT4 Tool[1]:Greifer
↓ Base[1]:Basis_Tisch ; %{PE}
11
12 ;FOLD SPTP HOME Vel=100 % DEFAULT ; %{PE}
13
14 ENDOOP
15
16
17 END
18 |

```

Abb. 12-9: Detailansicht, Editor



Die erweiterte Ansicht (Einblenden der DEF-Zeile, Einschalten der Detailansicht) hilft bei der Suche von Syntaxfehlern anhand der Zeilennummer. (>>> [12.5 "Fehler im Programm finden und beheben" Seite 360](#)) Zusätzlich müssen alle evtl. vorhanden Folds geöffnet sein.

12.5 Fehler im Programm finden und beheben

Beschreibung

Bei der Erstellung von Programmen mit Inline-Formularen können kaum Programmierfehler entstehen. Wird jedoch in der Hochsprache KRL-KUKA Robot Language programmiert, kann es bei der textuellen Eingabe zu Syntaxfehlern kommen. Diese sind für die Steuerung nicht interpretierbar und werden als Fehler angezeigt.

- Fehlerhafte Programme/ Dateien werden im Datei-Explorer mit einem roten X gekennzeichnet.



Abb. 12-10: Fehlerhaftes Programm

- Der übergeordneten Ordner des fehlerhaften Programm/ Datei wird auch mit einem kleinem roten X gekennzeichnet.

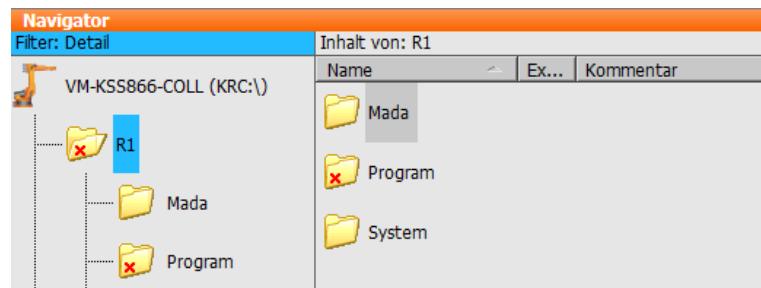


Abb. 12-11: Fehlerkennzeichnung im Ordner

Vorgehensweise

1. fehlerhaftes Modul im Navigator auswählen
(>> Abb. 12-10)
2. Auf die Schaltfläche **Fehlerliste** tippen.

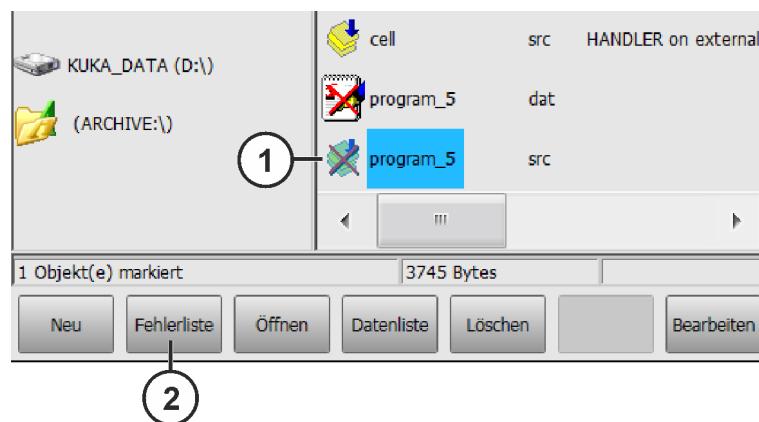


Abb. 12-12: Fehlerliste, Aufruf

3. Fehleranzeige (*programmname.ERR*) öffnet sich.

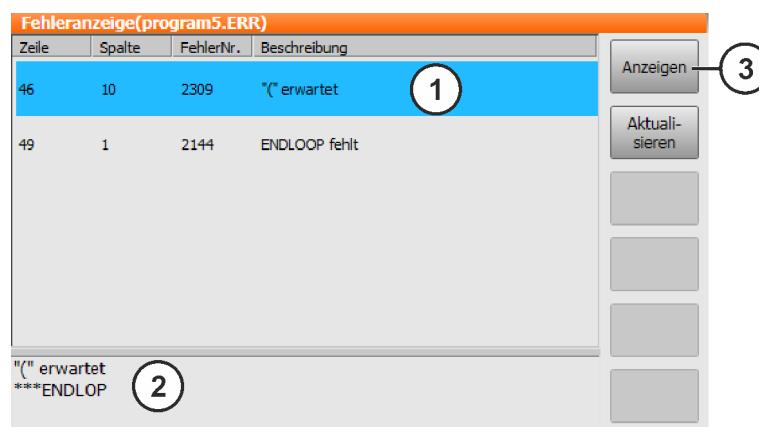


Abb. 12-13: Fehleranzeige

- In der Fehleranzeige werden die erkannten Syntaxfehler tabellarisch dargestellt.
 - Zeile mit der Maus markieren (1). Die Zeile wird blau hinterlegt.
 - Nähere Informationen werden in einem separaten Fenster angezeigt (2).
 - Über die Schaltfläche **Anzeigen** (3) öffnet sich das Programm.
4. Der Cursor springt in die fehlerbehaftete Syntaxzeile. Der Fehler kann analysiert und korrigiert werden.

```

Editor
 1  INI
 2
 3  LOOP
 4
 5  SPTP HOME Vel=100 % DEFAULT
 6
 7  SLIN P1 Vel=0.5 m/s CPDATA2 Tool[1]:Greifer
 8  SLIN P2 Vel=0.5 m/s CPDATA3 Tool[1]:Greifer
 9  SLIN P3 Vel=0.5 m/s CPDATA4 Tool[1]:Greifer
10
11  SPTP HOME Vel=100 % DEFAULT
12
13  ENDLOOP
14

```

Abb. 12-14: Syntaxfehler Endlosschleife, "ENDLOOP"



Die in der Fehleranzeige eingeblendete Zeilennummer 46 ist in diesem Beispiel nicht deckungsgleich mit der im Programm 13. Das liegt daran, das die Ansicht in der reduziert Darstellung erfolgt.
 (>>> [12.4 "Ansicht im Editor anpassen" Seite 358](#))

5. Editor verlassen und speichern

12.6 Roboterprogramme strukturieren

Möglichkeiten zur Strukturierung eines Roboterprogrammes

- Die Struktur eines Roboterprogramms ist ein wichtiger Faktor für seinen Gebrauchswert.
- Je strukturierter ein Programm ist, desto verständlicher, effektiver, lesbarer und wirtschaftlicher ist es.
- Um ein Programm strukturiert zu gestalten, können folgende Techniken verwendet werden:
 - **Kommentieren**
Kommentar und Stempel
(>>> [12.6.1 "Programme kommentieren" Seite 362](#))
 - **Einrücken**
Leerzeichen
(>>> [12.6.2 "Programme einrücken" Seite 365](#))
 - **Ausblenden**
Folds ("Falten")
(>>> [12.6.3 "Programmzeilen ausblenden" Seite 366](#))
 - **Modultechnik**
Verwenden von Unterprogrammen (Module)
(>>> [12.6.4 "Roboterprogramme verknüpfen" Seite 366](#))

12.6.1 Programme kommentieren

Beschreibung

Das Hinzufügen eines Kommentars bietet die Möglichkeit einen Text im Roboterprogramm zu hinterlegen, der für den Leser des Programms be-

stimmt ist. Der Kommentar-Text wird nicht vom Roboterinterpretierer eingelesen, sondern erhöht lediglich die Lesbarkeit eines Programms.

Kommentare können in Roboterprogrammen in vielerlei Hinsicht verwendet werden:

- **Informationen** zum Programmtext: Autor, Version, Erstellungsdatum



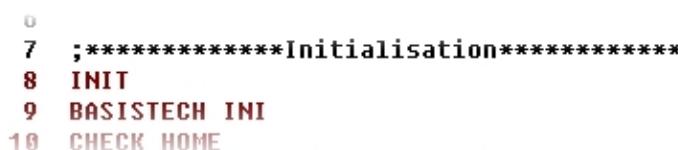
```

Editor
1 DEF welding1( )
2 ; Programmed by JACK SPARROW
3 ; Version 1.5 (10/10/2010)
4INI
5
6 PTP HOME Vel= 100 % DEFAULT
7

```

Abb. 12-15: Kommentar Beispiel: Informationen

- **Gliederung** des Programmtextes: Vor allem unter Verwendung grafischer Mittel (Sonderzeichen #, *, ~,.)



```

D
7 ;*****Initialisation*****
8 INIT
9 BASISTECHINI
10 CHECK HOME

```

Abb. 12-16: Kommentar Beispiel: Gliederung

- **Auskommentierung** (Experten-Ebene): Durch setzen eines Semikolons an den Beginn einer Programmzeile wird diese Zeile "auskommentiert", d. h. der Text wird als Kommentar erkannt und fließt nicht mit in die Programmabarbeitung ein.



```

10 CHECK HOME
11 PTP HOME Vel= 100 % DEFAULT
12 ;$OUT[33]=TRUE
13 AUTOEXTINI
14 LOOP

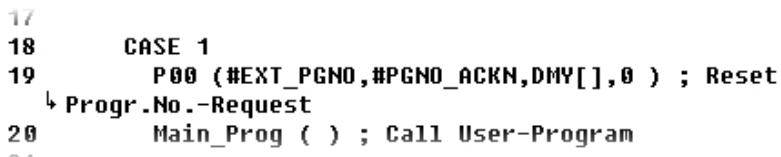
```

Abb. 12-17: Kommentar Beispiel: Auskommentieren



Inlineformulare lassen sich **nicht** mit einem ; auskommentieren.

- **Erläuterungen** zu einzelnen Zeilen sowie **Hinweis auf zu erledigende Arbeit**: Kennzeichnung unzureichender Programmteile



```

17
18 CASE 1
19 P00 (#EXT_PENO,#PGNO_ACKN,DMY[],0) ; Reset
   ↴ Prog.No.-Request
20 Main_Prog( ) ; Call User-Program

```

Abb. 12-18: Kommentar Beispiel: Erläuterungen

HINWEIS

Kommentare sind nur sinnvoll, wenn sie auf aktuellem Stand gehalten werden. Es ist zwingend notwendig, dass nachträglich auf Änderungen an den Anweisungen auch die Aktualisierung der Kommentare folgt!

Arten der Kommentierung

- **Setzen eines Semikolon** (Experten-Ebene): Durch das Einfügen eines Semikolons (" ; ") wird der in der Zeile nachfolgende Teil zum Kommentar.

```
...
SPTP HOME Vel=100% DEFAULT
; ##### Wuerfel holen #####
SLIN P1 Vel=0.5 m/s CPDAT1 Tool[1]....
```

- **Einfügen des Inline-Formulars "Kommentar"**



Abb. 12-19: Inline-Formular Kommentar

Pos.	Beschreibung
1	Beliebiger Text

- **Einfügen des Inline-Formulars "Stempel":**

Hier wird zusätzlich ein Zeitstempel eingefügt. Außerdem besteht die Möglichkeit den Namen des Bearbeiters einzufügen.



Abb. 12-20: Inline-Formular Stempel

Pos.	Beschreibung
1	Systemdatum (kann aktualisiert werden)
2	Systemzeit (kann aktualisiert werden)
3	Name oder Kennung des Benutzers
4	Beliebiger Text

Vorgehensweise

1. Die Zeile markieren, nach der der Kommentar oder der Stempel einge-fügt werden soll.
2. **Menüfolge: Befehle > Kommentar > Normal oder Stempel**



Abb. 12-21: Kommentar einfügen

3. Gewünschte Daten eingeben.

```

Editor
1 DEF program_5( )
2INI
3
4LOOP
5
6SPTP HOME Vel=100 % DEFAULT
7⇒
; 7.12.17 10:59 NAME: picking cubes CHANGES: start position
8SLIN P1 Vel=0.5 m/s CPDAT2 Tool[1]:Greifer Base[1]:Basis_T
9SLIN P2 Vel=0.5 m/s CPDAT3 Tool[1]:Greifer Base[1]:Basis_T
KRC\R1\PROGRAM_5.SRC Ln 7, Col 0
Befehl abbrechen Normal Zeit NEU Name NEU Text NEU Befehl OK

```

Abb. 12-22: Stempel einfügen

- 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.



Wenn bereits vorher ein Kommentar oder Stempel eingefügt worden ist, enthält das Inline-Formular noch die gleichen Angaben.

4. Mit der Schaltfläche **OK** den Kommentar/ Stempel speichern.

12.6.2 Programme einrücken

Einrücken von Programmzeilen

Ein effektives Mittel die Lesbarkeit eines Roboterprogramms zu erhöhen, ist das Einrücken von Programmzeilen. Dadurch wird erreicht, dass der Zusammenhang von Programmbausteinen verdeutlicht wird. Das Einrücken geschieht durch die Verwendung eines oder mehrerer Leerzeichen.

```

13 AUTOEXTINI
14 LOOP
15     P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
16     SWITCH PGNO ; Select with Programnumber
17         CASE 1
18             P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
19             Main_Prog ( ) ; Call User-Program
20         CASE 2
21             P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
22             Sub_Prog1 ( ) ; Call User-Program
23         DEFAULT
24             P00 (#EXT_PGNO,#PGNOFAULT,DMY[],0 )
25     END SWITCH
26 END LOOP

```

Abb. 12-23: Einrücken von Programmzeilen



Der Effekt des Einrückens ist rein optisch. Eingerückte Programmteile werden beim Programmablauf genauso abgearbeitet wie nicht-eingerückte.

12.6.3 Programmzeilen ausblenden

Programmzeilen ausblenden durch Folds

Die KUKA Robot Language bietet die Möglichkeit, Programmzeilen in *Folds* zusammenzufassen und auszublenden. Dadurch werden Programmteile für den Anwender nicht sichtbar. Die Einfachheit beim Lesen des Programms wird erhöht. In der Benutzergruppe Experte können die Folds dann geöffnet und auch geschlossen werden.

```
13
14
15 CHECK HOME
16
17
18
19
20
```

Abb. 12-24: Geschlossenes Fold

```
14
15 CHECK HOME
16 $H_POS=XHOME
17 IF CHECK_HOME==TRUE THEN
18 P00 (#CHK_HOME,#PGNO_GET,DMY[],0) ;Test HPos
19 ENDIF
20
```

Abb. 12-25: Geöffnetes Fold

Farbkennzeichnung bei Folds:

Farbe	Beschreibung
Dunkelrot	Geschlossener Fold
Hellrot	Geöffneter Fold
Dunkelblau	Geschlossener Unterfold
Hellblau	Geöffneter Unterfold
Grün	Inhalt des Folds

12.6.4 Roboterprogramme verknüpfen

Beschreibung

Unterprogrammtechnik bietet die Möglichkeit Roboterprogramme modular aufzubauen und somit strukturell effizient zu gestalten. Ziel ist es nicht alle Befehle in ein Programm zu schreiben, sondern bestimmte Abläufe, Berechnungen oder Prozesse in separate Programme auszulagern.

Durch die Nutzung von Unterprogrammen entstehen eine Vielzahl von Vorteilen:

- Das Hauptprogramm bekommt eine klare Struktur und ist leichter lesbar, da die Programmlänge reduziert wird.
- Unterprogramme können getrennt voneinander entwickelt werden: der Programmieraufwand kann geteilt werden, die Fehlerquellen minimieren sich.
- Unterprogramme können mehrfach wiederverwendet werden.
- Bessere Wartbarkeit und Pflege der einzelnen Programmmodulen/ Unterprogramme.

Es gibt grundsätzlich zwei verschiedene Unterprogrammtypen:

- **globale Unterprogramme**
(>>> "Globale Unterprogramme" Seite 367)
- **lokale Unterprogramme**
(>>> "Lokale Unterprogramme" Seite 367)

Globale Unterprogramme

Ein globales Unterprogramm ist ein eigenständiges Roboterprogramm (eines *.SRC und *.DAT Datei), das von einem anderen Roboterprogramm aus aufgerufen wird. Die Verzweigung der Programme kann anforderungsspezifisch verlaufen, d. h. einmal kann ein Programm ein Hauptprogramm sein, ein anderes mal fungiert es als Unterprogramm.

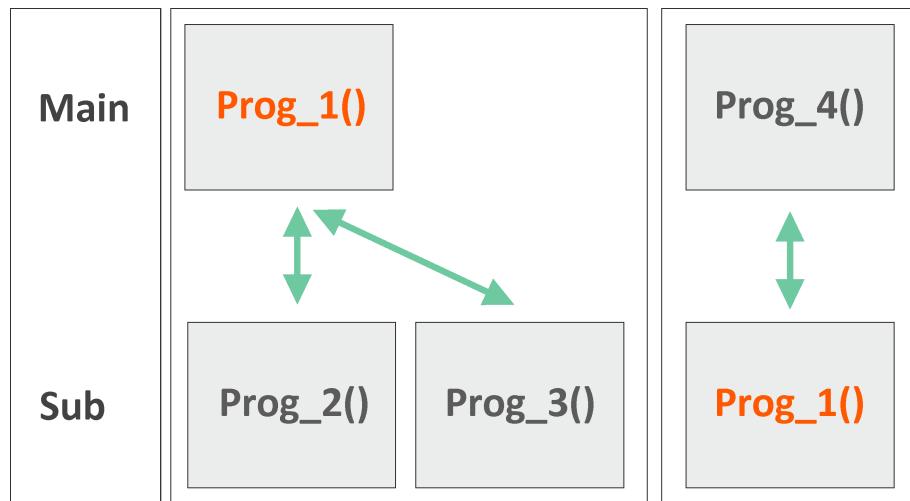


Abb. 12-26: Beispielschema für globale Unterprogramme

Lokale Unterprogramme

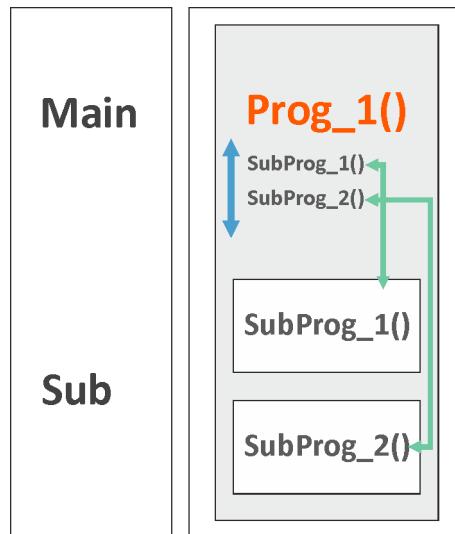


Abb. 12-27: Schema: lokale Unterprogramme

- Lokale Unterprogramme sind Programme, die in einem Hauptprogramm integriert sind, d. h. die Befehle sind in der gleichen SRC-Datei enthalten.
- Punktkoordinaten des Unterprogramms werden dementsprechend in der gleichen DAT-Datei gespeichert.

Ablauf eines Unterprogramm-Aufrufs

```

1 DEF main( )
2INI
3 PTP HOME Vel= 100 % DEFAULT
4 PTP P1 Vel=100 % PDAT1 Tool[2] Base[2]
5 PTP P2 Vel=100 % PDAT2 Tool[2] Base[2]
6 PTP P3 Vel=100 %
7
8 sub_prog()
9
10 PTP P4 Vel=100 %
11 PTP P5 Vel=100 %
12 PTP P6 Vel=100 % PDAT6 Tool[2] Base[2]
13 PTP HOME Vel= 100 % DEFAULT
14 END
15

```

Abb. 12-28: Ablauf eines Unterprogrammaufrufes

- Jedes Programm beginnt mit einer DEF- und endet mit einer END-Zeile.
- Wird in einem Hauptprogramm ein Unterprogramm aufgerufen, wird standardmäßig das Unterprogramm von DEF bis END abgearbeitet.
- Nach Erreichen der END-Zeile springt der Programmlaufzeiger wieder in das aufrufende Programm (Hauptprogramm).

HINWEIS

Um ein Unterprogramm frühzeitig (also vor der END-Zeile) zu verlassen, ist es möglich im Unterprogramm den Befehl RETURN zu programmieren. Das Ausführen dieser Programmzeile bewirkt dann den vorzeitigen Abbruch des Unterprogramms.

Unterprogramm aufrufen

- Die Benutzergruppe Experte ist erforderlich.
(>>> [4.2.1 "Benutzergruppe wechseln" Seite 84](#))
 - Syntax: Name()
1. Gewünschtes Hauptprogramm mit **Öffnen** in den Editor laden.
- ```

INI
SPTP HOME Vel= 100% DEFAULT

```
2. Cursor in die gewünschte Zeile positionieren.
  3. Eingabe des Unterprogrammnamens mit den beiden Klammern - z. B. **myprog()**.
- ```

INI
SPTP HOME Vel= 100% DEFAULT
myprog()
SPTP HOME Vel= 100% DEFAULT

```
4. Editor mittels Schließen-Symbol schließen und die Änderungen speichern.

Anwendung von Unterprogrammen

```
DEF MAIN()  
  
INI  
  
    GET_PEN()  
    PAINT_PATH()  
    PEN_BACK()  
    GET_PLATE()  
    GLUE_PLATE()  
    PLATE_BACK()  
  
END
```

12.6.5 Übung: Unterprogrammaufruf programmieren

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Unterprogrammaufruf programmieren**

13 Variablen und Vereinbarungen

13.1 Lerneinheit: Variablen und Vereinbarungen

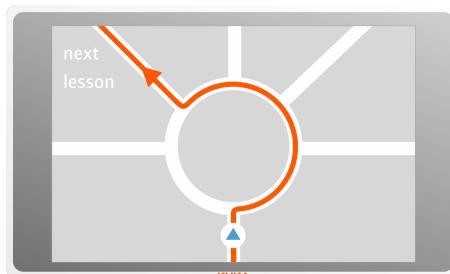


Abb. 13-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Datenhaltung in KRL
- Arbeiten mit einfachen Datentypen

13.2 Datenhaltung in KRL

Allgemeines zu Variablen



- In der Roboterprogrammierung mit KRL ist eine Variable im allgemeinsten Sinne einfach ein Behälter für Rechengrößen („Werte“), die im Verlauf eines Roboterprozesses auftreten.
- Eine Variable hat eine bestimmte zugewiesene Adresse im Speicher des Rechners.
- Eine Variable wird durch einen Namen bezeichnet, welches kein KUKA-Schlüsselwort ist.
- Jede Variable ist mit einem bestimmten Datentyp verbunden
- Die Deklaration des Datentyps ist vor der Verwendung notwendig.
- In KRL wird zwischen lokalen und globalen Variablen unterschieden.

13.2.1 Namenskonventionen

Folgende Regeln sind bei der Wahl der Variablennamens zu berücksichtigen:

- Namen in KRL dürfen maximal 24 Zeichen lang sein.
- 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.

- Groß- und Kleinschreibung ist nicht relevant.

Tipps

- sinnvolle, selbsterklärende Variablennamen benutzen,
- keine Abkürzungen einsetzen,
- sinnvolle Namenslänge nutzen, also nicht jedes Mal 24 Zeichen verbrauchen.

13.2.2 Datentypen

Vordefinierte Standarddatentypen

- **BOOL:** klassische "JA"/"NEIN" Ergebnisse
- **REAL:** Gleitkommazahl, Ergebnis von Rechenoperationen, um Rundungsfehler vermeiden zu können
- **INT:** Ganzzahl, klassische Zählvariable für Zählschleifen oder Stückzähler
- **CHAR:** nur ein Zeichen
String oder Text kann nur als CHAR-Feld realisiert werden

Einfache Datentypen	Ganzzahl	Gleitkommazahl	Logische Werte	Einzelnes Zeichen
Schlüsselwort	INT	REAL	BOOL	CHAR
Wertebereich	$-2^{31} \dots (2^{31}-1)$	$\pm 1.1 \cdot 10^{-38} \dots \pm 3.4 \cdot 10^{38}$	TRUE / FALSE	ASCII-Zeichensatz
Beispiele	-199 oder 56	-0.0000123 oder 3.1415	TRUE oder FALSE	"A" oder "q" oder "7"

Ausblick

Auf Grundlage der Standarddatentypen gibt es weitere abgeleitete komplexe Datentypen, welche in einem weiterführenden Seminar verwendet werden.

Felder/Array

Mehrere Variablen vom selben Datentyp werden mittels Index gespeichern.

```
Voltage[10] = 12.75
Voltage[11] = 15.59
```

Aufzählungsdatentyp

Alle Werte des Aufzählungstyps werden beim Anlegen mit Namen (Klar-text) definiert.

```
color = #red
```

Zusammengesetzter Datentyp/Struktur

Eine Struktur ist ein Zusammengesetzter Datentyp aus Komponenten verschiedener Datentypen.

```
Date = {day 14, month 12, year 1996}
```

13.2.3 Anlegen von Variablen

Deklaration von Variablen

- die Deklaration muss immer vor der Verwendung erfolgen.
- jeder Variable muss ein Datentyp zugeordnet werden.
- bei der Namensvergabe sind die Namenskonventionen einzuhalten.
- das Schlüsselwort für die Deklaration lautet **DECL**.
- das Schlüsselwort **DECL** kann bei den vier einfachen Datentypen entfallen.
- Wertzuweisungen erfolgen im Vorlauf.
- die Variablen-deklaration kann an unterschiedlichen Stellen erfolgen.
Das wirkt sich auf die Lebensdauer und Gültigkeit der jeweiligen Variablen aus.

13.2.4 Doppeldeklaration von Variablen

Beschreibung

Eine Doppeldeklaration entsteht immer bei Verwendung gleicher Variablennamen (Zeichenfolgen).

Es handelt sich um **keine** Doppeldeklaration, wenn der gleiche Variablenname in einem *.SRC/*.DAT-File und in der \$config.dat verwendet wird. In diesem Fall hat die lokale Variable Vorrang vor der globalen Variablen sofern die Variable aus dem Programm heraus beschrieben wird in welchem die Variable deklariert wurde.

Wird die Variable aus einem anderen Programm heraus beschrieben wird die globale Variable beschrieben.

13.2.5 Lebensdauer und Gültigkeit von Variablen

Lebensdauer

- Unter der Lebensdauer versteht man den Zeitraum, in dem für die Variable Speicherplatz reserviert ist.
- Laufzeitvariablen geben beim Verlassen des Programmes oder der Funktion ihren Speicherplatz wieder frei.
- Variablen in einer Datenliste behalten ihren aktuellen (letzten) Wert in ihrem Speicherplatz dauerhaft.

Gültigkeit

- Lokal deklarierte Variablen sind nur in dem Programm verfügbar und sichtbar, in dem sie deklariert wurden.
- Globale Variablen sind in einer zentralen (globalen) Datenliste angelegt.
- Globale Variablen können auch in einer lokalen Datenliste angelegt und beim Deklarieren mit dem Schlüsselwort **global** versehen werden.

13.2.6 Variablen-deklaration in Abhängigkeit des Speicherortes

Variable in der *.SRC-Datei

- Eine Variable welche in der *.SRC-Datei erstellt wird, heißt Laufzeitvariable.

- Diese kann nicht immer angezeigt werden.
- Sie ist nur in der Programmroutine verfügbar, in der sie deklariert worden ist. Somit sind die Variablen während der Programmausführung verfügbar.
- Sie gibt ihren Speicherplatz beim Erreichen der letzten Programmzeile (END-Zeile) wieder frei.

Variable im lokalen *.DAT- File

- kann während des Programmablaufes des dazugehörigen *.SRC- File immer angezeigt werden,
- der Wert der Variable bleibt nach Beendigung des Programmes erhalten,
- ist im kompletten *.SRC- File , also auch in lokalen Unterprogrammen, verfügbar,
- kann auch als globale Variable erstellt werden,
- erhält den aktuellen Wert im *.DAT- File und beginnt beim erneuten Aufruf mit dem gespeicherten Wert.
- Wird die Variable global deklariert, ist diese auch global verfügbar. Es ist in allen Programmrountinen ein Lese- und Schreibzugriff möglich, wenn das DAT-File mit dem Schlüsselwort PUBLIC und bei der Deklaration zusätzlich das Schlüsselwort GLOBAL verwendet wird.

Variable in der Systemdatei \$CONFIG.DAT

- ist in allen Programmen verfügbar (global),
- kann immer angezeigt werden, selbst wenn kein Programm aktiv ist,
- ist global verfügbar, d.h. in allen Programmrountinen ist ein Lese- und Schreibzugriff möglich,
- speichert den aktuellen Wert in der \$CONFIG.DAT.

13.2.7 KUKA Systemdaten

Beschreibung

- KUKA Systemdaten kommen in allen Datentypen vor, beispielsweise als
 - Aufzählungdatentyp, wie z.B. Betriebsart
 - Struktur, wie z.B. Datum/Uhrzeit
- Systeminformationen erhält man aus KUKA Systemvariablen. Diese
 - lesen die aktuelle Systeminformation aus,
 - verändern aktuelle Systemkonfigurationen,
 - sind vordefiniert und beginnen mit dem "\$"-Zeichen
 - \$DATE (aktuelle Uhrzeit und Datum)
 - \$POS_ACT (aktuelle Roboterposition)
 - \$MODE_OP(aktuelle Betriebsart)
 - ...

13.3 Arbeiten mit einfachen Datentypen

Im Folgenden wird das Erstellen, Initialisieren und Verändern von Variablen erläutert. Hierbei werden nur die einfachen Datentypen verwendet.

Einfache Datentypen mit KRL

- ganze Zahlen (INT)
- Gleitkommazahlen (REAL)
- logische Werte (BOOL)
- einzelnes Zeichen (CHAR)

13.3.1 Logikbefehle und Schleifen mittels Inline-Formulare programmieren

Beschreibung

Ab der KSS 8.6 wurden für die Logikprogrammierung über das smartPAD weitere Inline-Formulare ergänzt.



Abb. 13-2: Logikbefehle über Inlineformulare programmieren

Die Inline-Formulare erleichtern das Programmieren und verhindern Syntaxfehler. Hinter dem Inline-Formulars verbirgt sich die bekannte KRL-Syntax, die natürlich nicht verändert wurde. Für komplexere Strukturprogrammierung, z. B. verschachteln von Schleifen, empfiehlt es sich weiterhin auf die native KRL-Programmierung zurückzugreifen.



Die Inline-Formulare lassen sich nur in der SRC-Datei einfügen. Dazu muss das Programm, wie bei der Syntaxprogrammierung, geöffnet sein.

13.3.2 Deklarieren von Variablen

Prinzip der Variablendeclaration

Programmaufbau exemplarisch im SRC-File

- im Deklarationsteil müssen Variablen deklariert werden
- der Initialisierungsteil beginnt mit der ersten Wertzuweisung, meistens jedoch mit der "INI"-Zeile
- im Anweisungsteil werden Werte zugewiesen oder verändert

```

DEF main( )
; Deklarationsteil
...
; Initialisierungsteil
INI
...
; Anweisungsteil
SPTP HOME Vel=100% DEFAULT
...
END

```

Standardansicht ändern

- Das Einblenden der DEF- Zeile ist nur als Experte möglich
- Der Vorgang ist notwendig, um bei Modulen vor die "INI"-Zeile in den Deklarationsteil zu gelangen
- Um die DEF- und END-Zeile sehen zu können, wichtig auch bei der Variablenübergabe in Unterprogramme

Variablen-deklaration planen

- Lebensdauer festlegen
- Gültigkeit/Verfügbarkeit festlegen
- Datentyp festlegen
- Namensvergabe und Deklaration

Vorgehensweise bei der Deklaration von Variablen mit einfachem Datentyp

Variable im SRC-File erstellen

1. Benutzergruppe Experte
2. DEF-Zeile einblenden lassen
3. SRC-File im Editor öffnen
4. Deklaration der Variablen durchführen

```

DEF MY_PROG ( )
DECL INT counter
DECL REAL price
DECL BOOL error
DECL CHAR symbol
INI
...
END

```

5. Programm schließen und abspeichern

Variable im DAT-File erstellen

1. Benutzergruppe Experte
2. DAT-File im Editor öffnen
3. Deklaration der Variablen durchführen

```

DEFDAT MY_PROG
EXTERNAL DECLARATIONS
DECL INT counter1
DECL REAL price1
DECL BOOL error1
DECL CHAR symbol1
...
ENDDAT

```

HINWEIS

Keine Variablen verwenden, die in der in der SRC-Datei verwendet werden. Dies führt zu einer Doppeldeklaration.

4. Datenliste schließen und abspeichern

Variable in der \$CONFIG.DAT erstellen

1. Benutzergruppe Experte
2. im Ordner SYSTEM die \$CONFIG.DAT im Editor öffnen

```
DEFDAT $CONFIG
BASISTECH GLOBALS
AUTOEXT GLOBALS
USER GLOBALS
BackupManagerConfig
Conveyor
ENDDAT
```

3. Fold "USER GLOBALS" auswählen und mit Softkey "Fold öffn/sch" öffnen
4. Deklaration der Variablen durchführen

```
DEFDAT $CONFIG
...
;=====
; Userdefined Types
;=====
; Userdefined Externals
;=====
; Userdefined Variables
;=====
DECL INT counter
DECL REAL price
DECL BOOL error
DECL CHAR symbol
...
; ****
; Make your modifications -ONLY- here
; ****
ENDDAT
```



Die **\$CONFIG.DAT** kann im Zuge von Softwareupdates und über Technologiepaketen modifiziert werden. Ausschließlich der gesondert markierte Teil "*Make your modifications -ONLY- here*" ist davon ausgenommen.

5. Datenliste schließen und abspeichern

Globale Variable im DAT-File erstellen

1. Benutzergruppe Experte
2. DAT-File im Editor öffnen
3. Datenliste im Programmkopf mit dem Schlüsselwort PUBLIC erweitern

```
DEFDAT MY_PROG PUBLIC
```

4. Deklaration der Variablen durchführen

```
DEFDAT MY_PROG PUBLIC
EXTERNAL DECLARATIONS
DECL GLOBAL INT counter
DECL GLOBAL REAL price
DECL GLOBAL BOOL error
DECL GLOBAL CHAR symbol
...
ENDDAT
```

HINWEIS

Keine Namen verwenden, die in der CONFIG.DAT verwendet werden. Dies führt zu einer Doppeldeklaration.

5. Datenliste schließen und abspeichern

13.3.3 Initialisierung von Variablen mit einfachen Datentypen

Beschreibung der Initialisierung mit KRL

- Nach der Deklaration hat eine Variable nur einen Speicherplatz reserviert, ihr Wert ist immer ein ungültiger Wert.
- Im SRC-File erfolgt die Deklaration und die Initialisierung immer in zwei getrennten Zeilen.
- Im DAT-File erfolgt die Deklaration und die Initialisierung immer in einer Zeile (sofern eine Initialisierung gewünscht ist). Eine Konstante kann nur in einer Datenliste deklariert werden und muss dort sofort initialisiert werden.
Soll der zuletzt zugewiesene Variableninhalte in der Datenliste "sichtbar" sein, so muss dort auch eine "Erstinitialisierung" stattfinden.
- Der Initialisierungsteil beginnt in der *.src Datei mit der ersten Wertzuweisung.

Prinzip der Initialisierung

Initialisierung von Ganzzahlen

- Initialisierung als Dezimalzahl

```
value = 58
```

- Initialisierung als Binär-Zahl

```
value = 'B111010'
```

Rechnung: $1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 58$

Binär	2^5	2^4	2^3	2^2	2^1	2^0
Dez	32	16	8	4	2	1

- Initialisierung Hexadezimal

```
value = 'H3A'
```

HEX	Digit		Berech-nung		Summen											
A	10	*	16^0	=	10											
3	3	*	16^1	=	48											
Rechnung: $3 \cdot 16 + 10 = 58$												58				
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dez	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Vorgehenweise bei der Initialisierung mit KRL

Deklaration und Initialisierung im SRC-File

1. SRC-File im Editor öffnen
2. Deklaration ist durchgeführt worden
3. Initialisierung durchführen

```
DEF MY_PROG ( )
DECL INT counter
DECL REAL price
DECL BOOL error
DECL CHAR symbol
INI
counter = 10
price = 0.0
error = FALSE
symbol = "X"
...
END
```

4. Programm schließen und abspeichern

Deklaration und Initialisierung im DAT-File

1. DAT-File im Editor öffnen
2. Deklaration ist durchgeführt worden
3. Initialisierung durchführen

```
DEFDAT MY_PROG
EXTERNAL DECLARATIONS
DECL INT counter = 10
DECL REAL price = 0.0
DECL BOOL error = FALSE
DECL CHAR symbol = "X"
...
ENDDAT
```

4. Datenliste schließen und abspeichern

Deklaration im DAT-File und Initialisierung im SRC-File

1. DAT-File im Editor öffnen
2. Deklaration durchführen

```
DEFDAT MY_PROG
EXTERNAL DECLARATIONS
DECL INT counter
DECL REAL price
DECL BOOL error
```

```
DECL CHAR symbol  
...  
ENDDAT
```

3. Datenliste schließen und abspeichern
4. SRC-File im Editor öffnen
5. Initialisierung durchführen

```
DEF MY_PROG ( )  
...  
INI  
counter = 10  
price = 0.0  
error = FALSE  
symbol = "X"  
...  
END
```

6. Programm schließen und abspeichern

Deklaration und Initialisierung einer Konstanten

Beschreibung

- Konstanten werden mit dem Schlüsselwort **CONST** erstellt.
- Konstanten dürfen nur in Datenlisten angelegt werden.

Erstellung von Konstanten

1. DAT-File im Editor öffnen.
2. Deklaration und Initialisierung durchführen.

```
DEFDAT MY_PROG  
EXTERNAL DECLARATIONS  
DECL CONST INT max_size = 99  
DECL CONST REAL PI = 3.1415  
...  
ENDDAT
```

3. Datenliste schließen und abspeichern.

13.3.4 Variable über ein Inlineformular deklarieren und initialisieren



Abb. 13-3: Logikbefehle über Inlineformulare programmieren

1. Die Stelle im Programm markieren, an der das Inline-Formular einge-fügt werden soll
2. Auf die Schaltflächen Befehle (1) und Logik (2) tippen.
3. Im Auswahlfenster den gewünschten Logikbefehl (3) auswählen Im Beispiel die Variablen-deklaration-
4. Das eingefügte Inline-Formular anpassen und je nach Funktion mittels eingebender Schaltflächen erweitern.

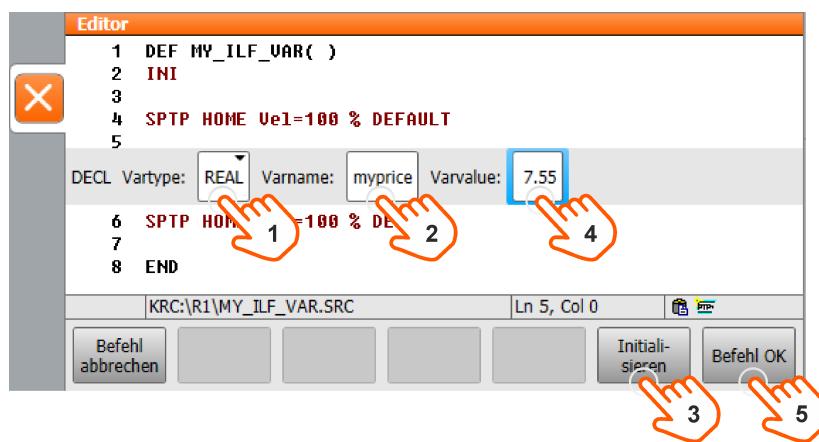
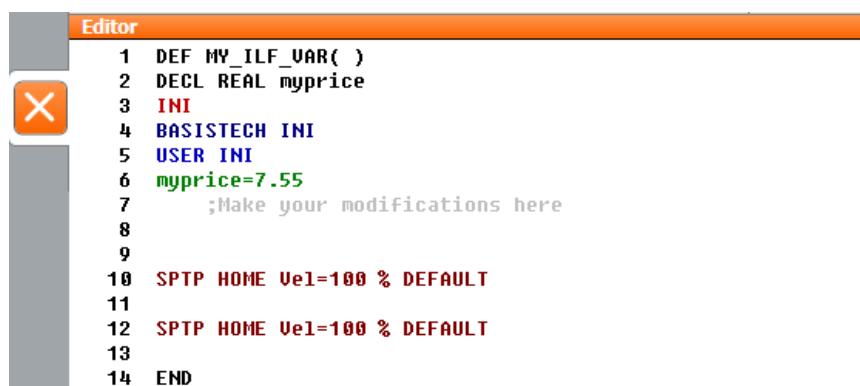


Abb. 13-4: Variable, Deklaration und Initialisierung im Inlineformu-lar

Für unser Beispiel "Variablen-deklaration" den Datentyp (Vartype) (1) auswählen und den Variablennamen (Varname) (2) festlegen. Soll die Variable gleichzeitig erstinitialisiert werden mit der Schaltfläche Initialisieren (3) das Inline-Formular erweitern Im Feld Varvalue (4) den Initialisierungswert eingeben und die Eingaben mit der Schaltfläche Befehl OK (5) speichern.

5. Die Variablen Deklaration und Initialisierung wird in das bestehende Programm übernommen. Die Initialisierung findet sich in der **Falte: INI > User INI** wieder.



```

Editor
1 DEF MY_ILF_VAR( )
2 DECL REAL myprice
3 INI
4 BASISTECH INI
5 USER INI
6 myprice=7.55
7 ;Make your modifications here
8
9
10 SPTP HOME Vel=100 % DEFAULT
11
12 SPTP HOME Vel=100 % DEFAULT
13
14 END

```

Abb. 13-5: Übernahme in KRL

Einschränkungen bei "Ändern"

Über die Schaltfläche **Ändern** kann keine Initialisierung geändert oder hinzugefügt werden.

Mögliche Änderungen: Namen und Typ der Variablen ändern

13.3.5 Manipulation von Variablenwerten einfacher Datentypen mit KRL

Auflistung der Möglichkeiten zur Veränderung von Variablenwerten mit KRL

Die Variablenwerte werden in den Programm Routinen (SRC-File) je nach Aufgabenstellung unterschiedlich verändert. Im Folgenden werden die gängigsten Methoden besprochen. Die Manipulation mittels Bit-Operationen und Standardfunktionen sind möglich, werden aber hier nicht vertieft.

Datenmanipulation mittels

- Grundrechenarten
 - (+) Addition
 - (-) Subtraktion
 - (*) Multiplikation
 - (/) Division
- Vergleichsoperationen
 - (==) identisch/Gleichheit
 - (<>) ungleich
 - (>) größer
 - (<) kleiner
 - (>=) größer gleich
 - (<=) kleiner gleich
- Logische Operationen
 - (**NOT**) Invertierung
 - (**AND**) Logisches UND
 - (**OR**) Logisches ODER
 - (**EXOR**) Exklusives ODER
- Bit-Operationen
 - (**B_NOT**) Bitweise Invertierung
 - (**B_AND**) Bitweise UND-Verkäpfung

- **(B_OR)** Bitweise ODER-Verkäpfung
- **(B_EXOR)** Bitweise Exklusiv-ODER-Verkäpfung

Standardfunktionen

- Absolutfunktion
- Wurzelfunktion
- Sinus und Kosinus-Funktion
- Tangens-Funktion
- Arcuskosinus- Funktion
- Arcustangens-Funktion
- mehrere Funktionen zur Stringmanipulation

Zusammenhänge bei der Datenmanipulation

Wertveränderung unter Verwendung des Datentyps REAL und INT

- aufrunden/abrunden

```
; Deklaration
DECL INT A,B,C
DECL REAL R,S,T
; Initialisierung
A = 3           ; A=3
B = 5.5         ; B=6 (ab x.5 wird aufgerundet)
C = 2.25        ; C=2 (abrunden)
R = 4           ; R=4.0
S = 6.5         ; S=6.5
T = C           ; T=2.0 (es wird der abgerundete Wert
genommen)
```



Faustregel:

Schritt 1: Das Zwischenergebnis (rechts vom $=$) wird errechnet.

Schritt 2: Danach erfolgt die Wertzuweisung, bei der je nach Ziellistentyp gerundet werden muss (INT als Ziellistentyp)

- Ergebnisse von arithmetischen Operationen (+;-;*)

Ergebnis	Operand 1	Operation	Operand 2
INT	INT	+/-*	INT
REAL	INT	+/-*	REAL
REAL	REAL	+/-*	INT
REAL	REAL	+/-*	REAL



Faustregel:

Sobald ein Operand den Datentyp REAL aufweist, ist auch das resultierende Ergebnis REAL.

```
; Deklaration
DECL INT D,E
DECL REAL U,V
; Initialisierung
D = 2
E = 5
U = 0.5
V = 10.6
; Anweisungsteil (Datenmanipulation)
```

```
D = D*E ; D = 2 * 5 = 10
E = E+V ; E= 5 + 10.6 = 15.6 -> aufrunden E=16
U = U*V ; U= 0.5 * 10.6 = 5.3
V = E+V ; V= 16 + 10.6 = 26.6
```

- Ergebnisse von arithmetischen Operationen (/)

Besonderheiten bei arithmetischen Operationen mit Integerwerten:

- Bei Zwischenergebnissen von reinen Integeroperationen werden alle Nachkommastellen abgeschnitten.
- Bei Wertzuweisungen an eine Integervariable wird das Ergebnis nach den normalen Rechenregeln gerundet.

```
; Deklaration
DECL INT F
DECL REAL W
; Initialisierung
F = 10
W = 10.0
; Anweisungsteil (Datenmanipulation)
; INT / INT -> INT
F = F/2 ; F=5
F = 10/4 ; F=2 (10/4 = 2.5 -> Kommastelle abschneiden)
; REAL / INT -> REAL
F = W/4 ; F=3 (10.0/4=2.5 -> aufrunden)
W = W/4 ; W=2.5
```

Vergleichsoperationen

Mit Vergleichsoperatoren können logische Ausdrücke gebildet werden. Das Ergebnis eines Vergleichs ist immer vom Datentyp BOOL.

Operator/KRL	Beschreibung	zulässige Datentypen
==	identisch/ Gleichheit	INT, REAL, CHAR, BOOL
<>	ungleich	INT, REAL, CHAR, BOOL
>	größer	INT, REAL, CHAR
<	kleiner	INT, REAL, CHAR
>=	größer gleich	INT, REAL, CHAR
<=	kleiner gleich	INT, REAL, CHAR

```
; Deklaration
DECL BOOL G,H
; Initialisierung/Anweisungsteil
G = 10>10.1 ; G=FALSE
H = 10/3 == 3 ; H=TRUE
G = G<>H ; G=TRUE
```

Logische Operationen

Mit logischen Operationen können logische Ausdrücke gebildet werden. Das Ergebnis einer solchen Operation ist immer vom Datentyp BOOL.

Operationen		NOT A	A AND B	A OR B	A EXOR B
A=FALSE	B=FALSE	TRUE	FALSE	FALSE	FALSE
A=FALSE	B=TRUE	TRUE	FALSE	TRUE	TRUE

Operationen		NOT A	A AND B	A OR B	A EXOR B
A=TRUE	B=FALSE	FALSE	FALSE	TRUE	TRUE
A=TRUE	B=TRUE	FALSE	TRUE	TRUE	FALSE

```
; Deklaration
DECL BOOL K,L,M
; Initialisierung/Anweisungsteil
K = TRUE
L = NOT K ; L=FALSE
M = (K AND L) OR (K EXOR L) ; M=TRUE
L = NOT (NOT K) ; L=TRUE
```

Operatoren werden in der Reihenfolge ihrer Prioritäten ausgeführt

Priorität	Operator
1	NOT (B_NOT)
2	Multiplikation (*); Division (/)
3	Addition (+), Subtraktion (-)
4	AND (B_AND)
5	EXOR (B_EXOR)
6	OR (B_OR)
7	jeglicher Vergleich (==; <>; ...)

```
; Deklaration
DECL BOOL X, Y
DECL INT Z
; Initialisierung/Anweisungsteil
X = TRUE
Z = 4
Y = (4*Z+16 <> 32) AND X ; Y=FALSE
```

Vorgehensweise bei der Datenmanipulation

1. Datentyp für die Variable oder Variablen festlegen
2. Gültigkeit und Lebensdauer der Variablen ermitteln
3. Variablen Deklaration durchführen
4. Variable initialisieren
5. in den Programm Routinen, also immer im *.SRC-File, die Variable manipulieren
6. *.SRC-File schließen und abspeichern

HALT Befehl

HALT wird hauptsächlich während der Programmierungsphase zum Testen verwendet. Beispielsweise um den Inhalt einer Laufzeitvariablen zur Anzeige zu bringen.

- Der HALT Befehl 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.

Beispiel:

```
DEF program()
DECL BOOL a,b
INI
...
SPTP XP1
a=$IN[1]
b=$IN[2]
HALT
IF ((a == TRUE) AND (b == FALSE)) THEN
..
ENDIF
...
```

13.3.6 Übung: Einfache Datentypen

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Einfache Datentypen**

14 Nutzen von Programmablaufkontrollen

14.1 Lerneinheit: Nutzen von Programmablaufkontrollen

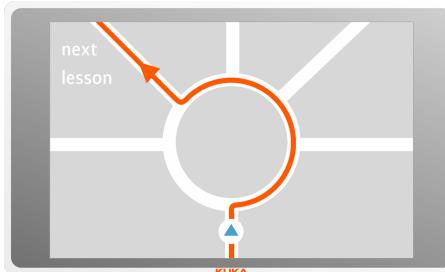


Abb. 14-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Schleifen programmieren
- Verzweigungen programmieren
- Verteiler programmieren
- Sprungbefehl programmieren
- Wartefunktionen in KRL programmieren

14.2 Schleifen programmieren

Neben den reinen Bewegungsbefehlen und den Kommunikationsbefehlen (Schalt- und Wartefunktionen) finden sich in Roboterprogrammen auch eine Vielzahl von Routinen, die der Programmablaufkontrolle dienen.

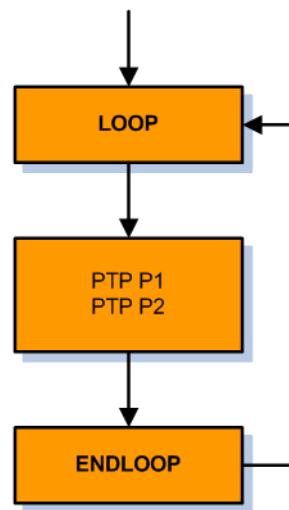
Allgemeines zu Schleifen

- Schleifen sind Kontrollstrukturen.
- Sie wiederholen Programmanweisungen so lange, bis eine Abbruchbedingung eintritt.
- Ein Sprung von außen in einen Schleifenkörper ist nicht erlaubt
- Schleifen können ineinander verschachtelt werden
- Es gibt verschiedene Schleifentypen
 - Endlosschleife
 - Zählschleife
 - bedingte Schleifen
 - abweisende Schleife
 - nicht abweisende Schleife

14.2.1 Endlosschleife programmieren

Beschreibung

Die Endlosschleife ist eine Schleife, die nach jeder Durchlauf erneut abgearbeitet wird. Die Ausführung kann durch äußere Einflüsse abgebrochen werden.

**Abb. 14-2: Programmablaufplan: Endlosschleife**

- die Endlosschleife kann mittels `EXIT` verlassen werden
- beim Verlassen einer Endlosschleife mit `EXIT` muss auf Kollisionsfreiheit geachtet werden
- sind zwei Endlosschleifen ineinander geschachtelt, dann werden auch zwei `EXIT` Befehle benötigt, um beide Schleifen zu verlassen

Syntax

```

LOOP
;   Anweisung
...
;   Anweisung
ENDLOOP
  
```

Beispiele beim Programmieren einer Endlosschleife

Endlosschleife ohne Abbruch

```

DEF MY_PROG( )
INI
SPTP HOME Vel=100% DEFAULT

LOOP
    SPTP P1
    SPTP P2
    SPTP P3
    SPTP P4
ENDLOOP

SPTP P5 Vel=30% PDAT5 Tool[1] Base[1]
SPTP HOME Vel=100% DEFAULT
END
  
```



Der Punkt P5 wird programmtechnisch nie angefahren

Endlosschleife mit Abbruch

```

DEF MY_PROG( )
INI
  
```

```

SPTP HOME Vel=100% DEFAULT

LOOP
    SPTP P1
    SPTP P2
; Bedingung für Abbruch
    IF $IN[3]==TRUE THEN ;
        EXIT
    ENDIF
    SPTP P3
    SPTP P4
ENDLOOP

SPTP P5 Vel=30% PDAT5 Tool[1] Base[1]
SPTP HOME Vel=100% DEFAULT
END

```



Der Punkt P5 wird angefahren sobald der Eingang 3 aktiv ist.
Wichtig: Die Fahrt zwischen P2 und P5 muss auf Kollisionsfreiheit geprüft werden

Endlosschleifen über das Inline-Formular

```
LOOP
```

Abb. 14-3: Inline-Formular LOOP ... ENDOLOOP ohne Felder

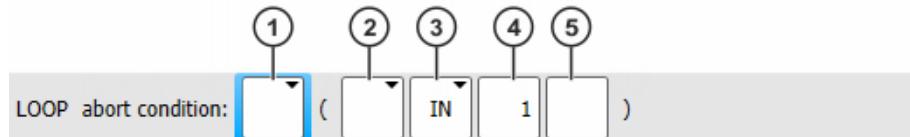


Abb. 14-4: Inline-Formular LOOP ... ENDOLOOP mit Feldern

Um die Felder anzuzeigen, die Schaltfläche **Hinzufügen Abbr. Bdg.** berühren.

Pos.	Beschreibung
1	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Äußere Verknüpfung hinzufügen. Der Term steht zwischen den geklammerten Ausdrücken.</p> <p>Den gewünschten Term über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
2	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks.</p> <p>Den gewünschten Operator über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR

Pos.	Beschreibung
3	<p>Signal, das ausgewertet wird. Default-Auswahl:</p> <ul style="list-style-type: none"> IN, OUT, CYCFLAG, TIMER oder FLAG <p>Auch andere Signale können programmiert werden: Über die Schaltfläche Freitext kann statt der Default-Auswahl ein Feld eingeblendet werden, im dem KRL eingegeben werden kann.</p> <p>System Liste blendet wieder die Default-Auswahl ein. Durch Berühren der Schaltflächen kann zwischen ihnen gewechselt werden.</p>
4	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde.</p> <p>Nummer des Signals eingeben.</p>
5	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde. Wenn für das Signal ein Name existiert, wird er angezeigt.</p> <p>Ab Benutzergruppe Experte: Ein Name kann eingegeben werden oder der bestehende Name geändert werden. Solange das Inline-Formular noch nicht gespeichert wurde, können diese Änderungen durch Drücken auf Langtext wieder zurückgesetzt werden.</p>

Einschränkungen bei "Ändern"

Das Inline-Formular kann über die Schaltfläche **Ändern** nicht geändert werden!

14.2.2 Zählschleife programmieren

Beschreibung

Die **FOR**-Schleife ist eine Kontrollstruktur, mit der eine oder mehrere Anweisungen mit einer festgelegten Anzahl von Wiederholungen ausgeführt werden können.

Für eine Zählschleife ist ein vorher deklarierter Schleifenzähler vom Datentyp Integer notwendig.

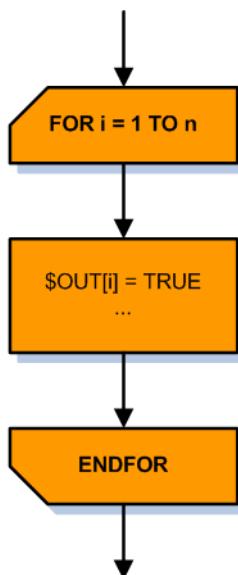


Abb. 14-5: Programmablaufplan: Zählschleife



Der Wert für *increment* muss als Ganzzahl eingegeben werden und kann keine Variable sein.

Syntax mit der Standard-Schrittweite + 1

```

FOR counter = start TO last
;Anweisung
ENDFOR
  
```

Die Zählschleife beginnt beim Wert `start` und endet spätestens beim Wert `last`.

Syntax für abweichende Schrittweiten

```

FOR counter = start TO last STEP increment
;Anweisung
ENDFOR
  
```

Die Schrittweite (*increment*) kann auch als Ganzzahl mit dem Schlüsselwort `STEP` angegeben werden.

Der Wert für *increment* muss als Ganzzahl eingegeben werden und kann **keine** Variable sein.

Die Zählschleife kann mittels `EXIT` sofort verlassen werden.

Funktionsweise

```

DECL INT counter

FOR counter = 1 TO 3 Step 1
;Anweisung
ENDFOR
  
```

1. Schleifenzähler wird mit dem Startwert initialisiert: `counter = 1`
2. Schleifenzähler wird bei `ENDFOR` um die Schrittweite `STEP` hochgezählt
3. Schleife beginnt wieder bei der `FOR`-Zeile

4. Prüfung der Eintrittsbedingung: Schleifenzähler muss kleiner gleich dem angegebenen Endwert sein, ansonsten wird die Schleife beendet
5. je nach Prüfung wird der Schleifenzähler erneut hochgezählt oder die Schleife ist beendet und das Programm wird nach der ENDFOR-Zeile fortgesetzt

Einfache Zählschleife ohne Angabe der Schrittweite

```
DECL INT counter

FOR counter = 1 TO 50
    $OUT[counter] = FALSE
ENDFOR
```



Ohne Angabe der Schrittweite mittels STEP wird automatisch die Schrittweite +1 verwendet.

Einfache Zählschleife mit Angabe der Schrittweite

```
DECL INT counter

FOR counter = 1 TO 4 STEP 2
    $OUT[counter] = TRUE
ENDFOR
```



Diese Schleife wird nur zweimal durchlaufen. Einmal mit dem Startwert counter=1 und das zweite Mal mit counter=3. Beim Zählerwert 5 wird die Schleife sofort abgebrochen.

Rückwärtzählen

Zählschleife mit Angabe einer negativen Schrittweite

```
DECL INT counter

FOR counter = 10 TO 1 STEP -1
    value = 100 - 2 * counter
ENDFOR
```



Der Initialwert oder Startwert der Schleife muss größer als der Endwert sein, damit die Schleife mehrmals durchlaufen werden kann.

Verschachtelte Zählschleifen

verschachtelte Zählschleife mit Angabe der Schrittweite

```
DECL INT counter1, counter2

FOR counter1 = 1 TO 21 STEP 2
    FOR counter2 = 20 TO 2 STEP -2
        ...
    ENDFOR
ENDFOR
```



Es wird immer zuerst die innere Schleife durchlaufen, hier mit counter2 und anschließend die Äußere (counter1).

Zählschleife mittels Inline-Formular programmieren

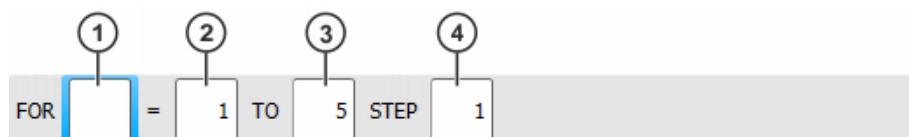


Abb. 14-6: Inline-Formular FOR ... ENDFOR

Pos.	Beschreibung
1	INT-Variable, die die Durchläufe zählt, die "Zählvariable". Die Variable kann, muss aber nicht vorher deklariert worden sein. Der Wert der Variablen kann in Anweisungen innerhalb und außerhalb der Schleife benutzt werden. Nach dem Verlassen der Schleife hat die Variable den zuletzt angenommenen Wert.
2	Startwert Die Zählvariable wird automatisch mit dem Startwert vorbesetzt.
3	Endwert Wenn der Endwert über- oder unterschritten ist, ist die Schleife beendet.
4	Schrittweite Nach jedem Schleifendurchlauf verändert sich die Zählvariable automatisch um die Schrittweite. Der Wert darf negativ sein. Default: 1. <ul style="list-style-type: none"> • Wert positiv: Die Schleife ist beendet, wenn die Zählvariable größer als der Endwert ist. • Wert negativ: Die Schleife ist beendet, wenn die Zählvariable kleiner als der Endwert ist. Der Wert darf weder Null noch eine Variable sein.

14.2.3 Abweisende Schleife programmieren

Beschreibung

Eine WHILE- Schleife wird auch *kopfgesteuerte Schleife* genannt. Sie ist eine *abweisende* oder *vorprüfende* Schleife, bei der die Ausführungsbedingung geprüft wird, bevor der Anweisungsteil der Schleife durchlaufen wird.

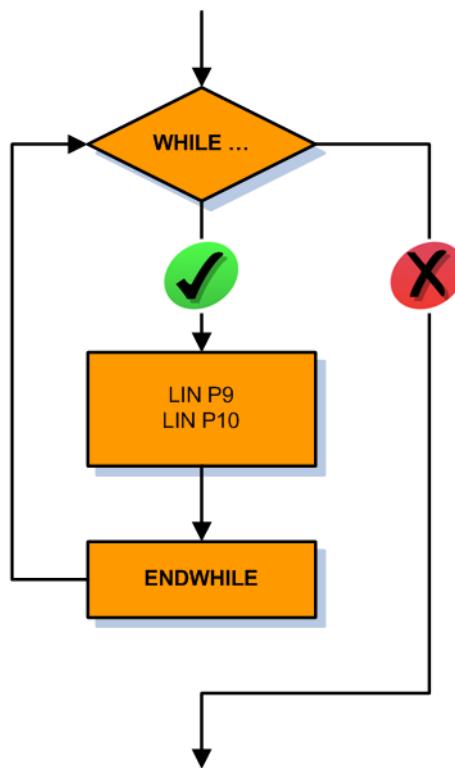


Abb. 14-7: Programmablaufplan: Abweisende Schleife

Syntax

Eine abweisende Schleife wiederholt Vorgänge, solange eine bestimmte Ausführbedingung (condition) erfüllt ist. Eine nicht erfüllte Ausführbedingung hat zur Folge, dass die Schleife sofort beendet wird und die Anweisungen nach ENDWHILE abgearbeitet werden

```

WHILE condition
  ; Anweisung
ENDWHILE
  
```

Die abweisende Schleife kann mittels EXIT sofort verlassen werden.

Programmieren mit einer abweisenden Schleife

- **abweisende Schleife mit einfacher Ausführbedingung**

```

...
WHILE $IN[41]==TRUE ; Teil liegt im Magazin bereit
  PICK_PART( )
ENDWHILE
...
  
```



Der Ausdruck WHILE \$IN[41]==TRUE kann auch auf WHILE \$IN[41] reduziert werden. Ein Weglassen bedeutet immer einen Vergleich auf TRUE.

- **abweisende Schleife mit einfacher, negierter Ausführbedingung**

```

...
WHILE $IN[41]==FALSE ; Magazin ist leer
  PICK_PART( )
  
```

ENDWHILE...

oder

```
...
WHILE NOT $IN[41]==TRUE ; Magazin ist leer
    PICK_PART( )
ENDWHILE...
```

- **abweisende Schleife mit komplexer Ausführbedingung**

```
...
WHILE  (($IN[40]==TRUE) AND ($IN[41]==FALSE) OR
(counter>20))
    PALLET( )
ENDWHILE
...
```

Abweisende Schleife mittels Inline-Formular programmieren

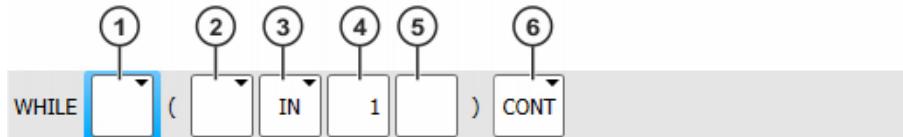


Abb. 14-8: Inline-Formular WHILE ... ENDWHILE

Pos.	Beschreibung
1	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Äußere Verknüpfung hinzufügen. Der Term steht zwischen den geklammerten Ausdrücken.</p> <p>Den gewünschten Term über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
2	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks.</p> <p>Den gewünschten Operator über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
3	<p>Signal, das ausgewertet wird. Default-Auswahl:</p> <ul style="list-style-type: none"> • IN, OUT, CYCFLAG, TIMER oder FLAG <p>Auch andere Signale können programmiert werden. Hierfür stehen die folgenden Schaltflächen zur Verfügung. Das Berühren der Schaltfläche zeigt die jeweils nächste an.</p> <ul style="list-style-type: none"> • Freitext: Blendet ein Feld ein, in dem KRL eingegeben werden kann. • Anwender Liste: Blendet eine Liste mit nutzerdefinierten Variablen ein. Voraussetzung: Die Liste wurde konfiguriert. • System Liste: Blendet wieder die Default-Auswahl ein.

Pos.	Beschreibung
4	<ul style="list-style-type: none"> Wenn ein Signal aus der Default-Auswahl ausgewählt wurde: Nummer eingeben. Wenn eine Variable aus der Anwenderliste ausgewählt wurde: Wert eingeben.
5	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde. Wenn für das Signal ein Name existiert, wird er angezeigt.</p> <p>Ab Benutzergruppe Experte: Ein Name kann eingegeben werden oder der bestehende Name geändert werden. Solange das Inline-Formular noch nicht gespeichert wurde, können diese Änderungen durch Drücken auf Langtext wieder zurückgesetzt werden.</p>
6	<ul style="list-style-type: none"> CONT: Bearbeitung im Vorlauf [leer]: Bearbeitung mit Vorlaufstopp

Einschränkungen bei "Ändern"

Über die Schaltfläche **Ändern** kann nur die Bedingung geändert werden. CONT kann nicht hinzugefügt oder entfernt werden.

14.2.4 Nicht abweisende Schleife programmieren

Beschreibung

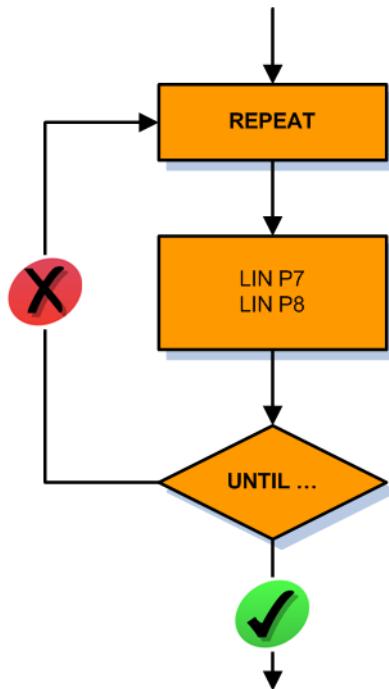


Abb. 14-9: Programmablaufplan: Nicht abweisende Schleife

- Die nicht abweisende Schleife wird auch *fußgesteuerte Schleife* genannt.
- Eine REPEAT-Schleife ist eine *nicht-abweisende* oder *nachprüfende* Schleife, bei der die Abbruchbedingung erst geprüft wird, nachdem der Anweisungsteil der Schleife erstmals durchlaufen wurde.

- Nachdem der Anweisungsteil abgearbeitet wurde wird geprüft, ob eine Bedingung (condition) erfüllt ist, um die Schleife verlassen zu können.
 - Bei einem positiven Resultat der Bedingung wird die Schleife verlassen und es werden die Anweisungen nach UNTIL abgearbeitet .
 - Bei einem negativen Resultat der Bedingung wird die Schleife erneut bei REPEAT gestartet.
- Syntax

```
REPEAT
  ; Anweisung
UNTIL condition
```

- Die nicht abweisende Schleife kann mittels EXIT sofort verlassen werden.

Programmieren einer nicht abweisenden Schleife

- Nicht abweisende Schleife mit einfacher Ausführbedingung.

```
...
REPEAT
  PICK_PART( )
UNTIL $IN[42]==TRUE ; Eingang 42: Magazin ist leer
...
```



Der Ausdruck UNTIL \$IN[42]==TRUE kann auch auf UNTIL \$IN[42] reduziert werden. Ein Weglassen bedeutet immer einen Vergleich auf TRUE

- Nicht abweisende Schleife mit komplexer Ausführbedingung.

```
...
REPEAT
  PALLET( )
UNTIL (($IN[40]==TRUE) AND ($IN[41]==FALSE) OR
(counter>20))
...
```



Bei einem positiven Ergebnis wird die Schleife beendet!

Nicht abweisende Schleife mittels Inline-Formular programmieren

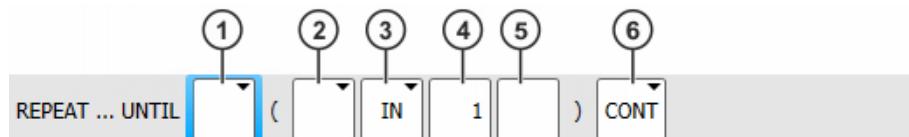


Abb. 14-10: Inline-Formular REPEAT ... UNTIL

Pos.	Beschreibung
1	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Äußere Verknüpfung hinzufügen. Der Term steht zwischen den geklammerten Ausdrücken. Den gewünschten Term über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
2	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks. Den gewünschten Operator über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
3	<p>Signal, das ausgewertet wird. Default-Auswahl:</p> <ul style="list-style-type: none"> • IN, OUT, CYCFLAG, TIMER oder FLAG <p>Auch andere Signale können programmiert werden. Hierfür stehen die folgenden Schaltflächen zur Verfügung. Das Berühren der Schaltfläche zeigt die jeweils nächste an.</p> <ul style="list-style-type: none"> • Freitext: Blendet ein Feld ein, in dem KRL eingegeben werden kann. • Anwender Liste: Blendet eine Liste mit nutzerdefinierten Variablen ein. Voraussetzung: Die Liste wurde konfiguriert. • System Liste: Blendet wieder die Default-Auswahl ein.
4	<ul style="list-style-type: none"> • Wenn ein Signal aus der Default-Auswahl ausgewählt wurde: Nummer eingeben. • Wenn eine Variable aus der Anwenderliste ausgewählt wurde: Wert eingeben.
5	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde. Wenn für das Signal ein Name existiert, wird er angezeigt.</p> <p>Ab Benutzergruppe Experte: Ein Name kann eingegeben werden oder der bestehende Name geändert werden. Solange das Inline-Formular noch nicht gespeichert wurde, können diese Änderungen durch Drücken auf Langtext wieder zurückgesetzt werden.</p>
6	<ul style="list-style-type: none"> • CONT: Bearbeitung im Vorlauf • [leer]: Bearbeitung mit Vorlaufstopp

Einschränkungen bei "Ändern"

Über die Schaltfläche **Ändern** kann nur die Bedingung geändert werden. CONT kann nicht hinzugefügt oder entfernt werden.

14.3 Abfragen oder Verzweigungen programmieren

Beschreibung

Mit der Verwendung von Verzweigungen kann ermöglicht werden, dass Programmabschnitte nur unter einer bestimmten Bedingung ausgeführt werden. Sie wird dazu benutzt, um ein Programm in mehrere Pfade aufzuteilen. Eine *bedingte Verzweigung* (IF- Abfrage) besteht aus einer Bedingung und zwei Anweisungsteilen.

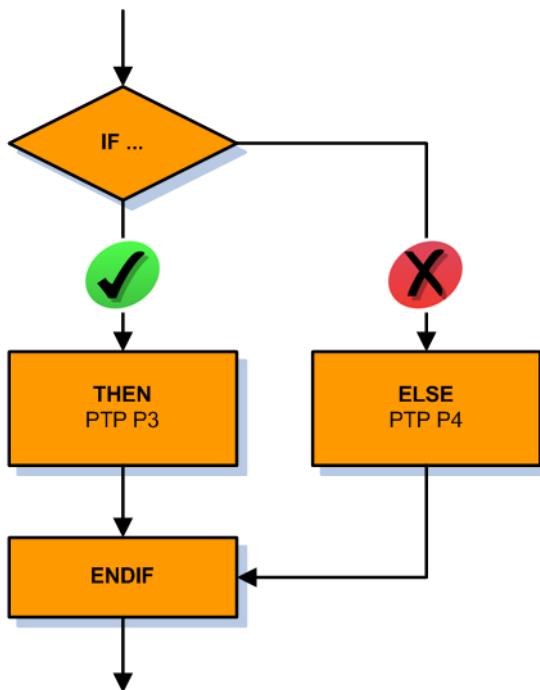


Abb. 14-11: Programmablaufplan: IF-Verzweigung

Die IF-Anweisung überprüft, ob diese Bedingung wahr (TRUE) oder falsch (FALSE) ist.

- Wenn die Bedingung erfüllt ist, kann die erste Anweisung abgearbeitet werden.
- Ist die Bedingung nicht erfüllt, wird die zweite, alternative Anweisung abgearbeitet.

Varianten der IF- Abfrage

- Der zweite Anweisungsteil kann weggelassen werden: IF-Abfrage ohne ELSE. Dem zufolge wird bei Nichterfüllung der Bedingung direkt nach der Verzweigung (ENDIF) das Programm fortgeführt.
- Mehrere IF-Abfragen können ineinander geschachtelt werden (*Mehrfachverzweigung*): Die Abfragen werden der Reihe nach abgearbeitet und geprüft, ob eine Bedingung erfüllt ist.

Programmieren einer Verzweigung

Syntax

- mit Alternativzweig

```

IF condition THEN
  Anweisung
ELSE

```

```
Anweisung
ENDIF
```

- **ohne Alternativzweig (Abfrage)**

```
IF condition THEN
Anweisung
ENDIF
```

Beispiele von Verzweigungen

Verzweigung mit Alternativzweig

```
DEF MY_PROG( )
DECL INT error_nr
...
INI
error_nr = 4
...
; nur bei error_nr = 5 wird XP21 angefahren, ansonsten XP22
IF error_nr == 5 THEN
    SPTP XP21Anweisung 1
ELSE
    SPTP XP22Anweisung 2
ENDIF
...
END
```

Verzweigung ohne Alternativzweig

```
DEF MY_PROG( )
DECL INT error_nr
...
INI
error_nr = 4
...
; nur bei error_nr = 5 wird XP21 angefahren
IF error_nr == 5 THEN
    SPTP XP21
ENDIF
...
END
```

Verzweigung mit komplexen Ausführbedingungen

```
DEF MY_PROG( )
DECL INT error_nr
...
INI
error_nr = 4
...
; nur bei error_nr = 1 oder 10 oder groesser 99 wird XP21
angefahren
IF ((error_nr == 1) OR (error_nr == 10) OR (error_nr >
99)) THEN
    SPTP XP21
ENDIF
...
END
```

Verzweigung mit boolschen Ausdrücken

```

DEF MY_PROG( )
DECL BOOL no_error
...
INI
no_error = TRUE
...
; nur ohne Fehler (no_error) wird XP21 angefahren
IF no_error == TRUE THEN
    SPTP XP21
ENDIF
...
END

```



Der Ausdruck `IF no_error==TRUE THEN` kann auch auf `IF no_error THEN` reduziert werden. Ein Weglassen bedeutet immer einen Vergleich auf `TRUE`.

Abfragen oder Verzweigungen mit einem Inline-Formular programmieren

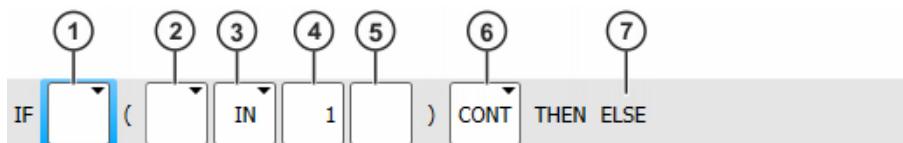


Abb. 14-12: Inline-Formular IF ... THEN

Pos.	Beschreibung
1	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Äußere Verknüpfung hinzufügen. Der Term steht zwischen den geklammerten Ausdrücken.</p> <p>Den gewünschten Term über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
2	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks.</p> <p>Den gewünschten Operator über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
3	<p>Signal, das ausgewertet wird. Default-Auswahl:</p> <ul style="list-style-type: none"> • IN, OUT, CYCFLAG, TIMER oder FLAG <p>Auch andere Signale können programmiert werden. Hierfür stehen die folgenden Schaltflächen zur Verfügung. Das Berühren der Schaltfläche zeigt die jeweils nächste an.</p> <ul style="list-style-type: none"> • Freitext: Blendet ein Feld ein, in dem KRL eingegeben werden kann. • Anwender Liste: Blendet eine Liste mit nutzerdefinierten Variablen ein. Voraussetzung: Die Liste wurde konfiguriert. • System Liste: Blendet wieder die Default-Auswahl ein.

Pos.	Beschreibung
4	<ul style="list-style-type: none"> Wenn ein Signal aus der Default-Auswahl ausgewählt wurde: Nummer eingeben. Wenn eine Variable aus der Anwenderliste ausgewählt wurde: Wert eingeben.
5	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde. Wenn für das Signal ein Name existiert, wird er angezeigt.</p> <p>Ab Benutzergruppe Experte: Ein Name kann eingegeben werden oder der bestehende Name geändert werden. Solange das Inline-Formular noch nicht gespeichert wurde, können diese Änderungen durch Drücken auf Langtext wieder zurückgesetzt werden.</p>
6	<ul style="list-style-type: none"> CONT: Bearbeitung im Vorlauf [leer]: Bearbeitung mit Vorlaufstopp
7	ELSE über die entsprechende Schaltfläche hinzufügen. Fokus muss auf Feld 1 oder 2 sein.

Einschränkungen bei "Ändern"

Über die Schaltfläche **Ändern** kann nur die Bedingung geändert werden. ELSE oder CONT können nicht hinzugefügt oder entfernt werden.

14.4 Verteiler programmieren (SWITCH- CASE)

Beschreibung des Verteilers (SWITCH- CASE)

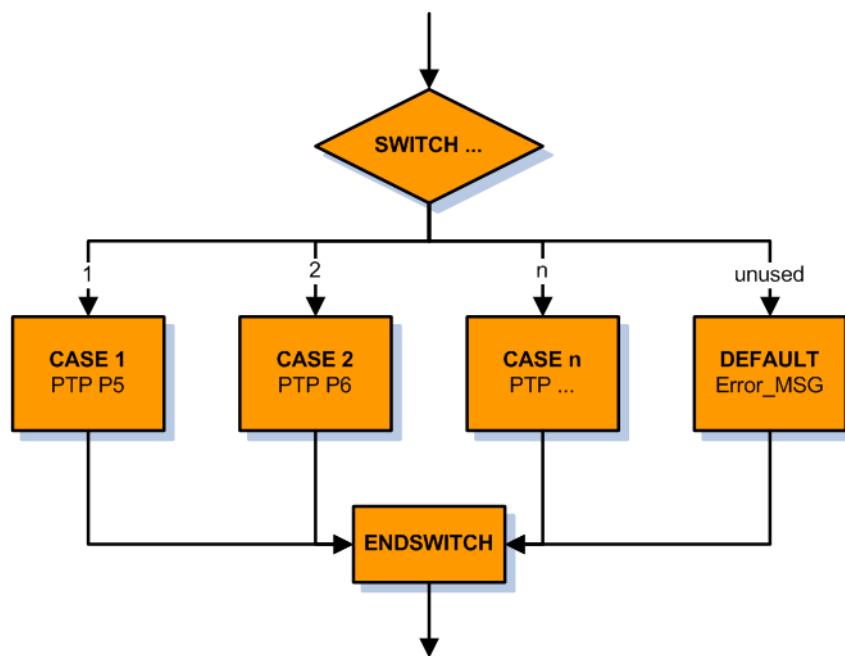


Abb. 14-13: Programmablaufplan: Verteiler SWITCH - CASE

- Will man viele Fälle unterscheiden und für jeden Fall unterschiedliche Aktionen ausführen, so kann das mit einer SWITCH - CASE Anweisung erreicht werden.

- Eine SWITCH- CASE Verzweigung ist ein *Verteiler* bzw. *Mehrfachverzweigung* und dient der Fallunterscheidung.
- Eine übergebene Variable in der SWITCH-Anweisung wird als Schalter genutzt und springt im Anweisungsblock in die vordefinierten CASE-Anweisungen.
- Findet die SWITCH-Anweisung keinen vordefinierten CASE, so wird der DEFAULT-Abschnitt durchlaufen, falls dieser vorher definiert worden ist.

Syntax

```

SWITCH Auswahlkriterium
    CASE Wert
        Anweisung
    CASE Wert
        Anweisung
    CASE Wert
        Anweisung
    ...
    DEFAULT
        Anweisung
ENDSWITCH

```

Werteübergabe über einfache Datentypen

- **SWITCH- CASE** kann in Kombination mit folgenden Datentypen verwendet werden:
 - **INT (Ganzzahl)**
- Beispiel mit definierten Verteilern und ohne Alternativpfad**

```

DEF MY_PROG( )
DECL INT my_number
...
INI
my_number = 2
...
SWITCH my_number
    CASE 1
        SPTP P21 Vel=30% PDAT5 Tool[1] Base[1]
    CASE 2
        SPTP P22 Vel=30% PDAT5 Tool[1] Base[1]
    CASE 3
        SPTP P23 Vel=30% PDAT5 Tool[1] Base[1]
ENDSWITCH
...

```



Bei *number* ungleich 1 oder 2 oder 3 wird direkt zu END-SWITCH gesprungen, ohne eine Anweisung auszuführen.

- **CHAR (ein Zeichen)**

Beispiel mit definierten Verteilern und einem Alternativpfad

```

DEF MY_PROG( )
DECL CHAR my_sign
...
INI
my_sign = "a"
...
SWITCH my_sign

```

```

CASE "a"
    SPTP P21 Vel=30% PDAT5 Tool[1] Base[1]
CASE "b"
    SPTP P22 Vel=30% PDAT5 Tool[1] Base[1]
CASE "c"
    SPTP P23 Vel=30% PDAT5 Tool[1] Base[1]
DEFAULT
    SPTP P24 Vel=30% PDAT5 Tool[1] Base[1]
ENDSWITCH
...

```



Bei *my_sign* ungleich "a" oder "b" oder "c" wird in den "DE-FAULT-Fall" gesprungen um diese Anweisung(en) auszuführen.

Mit mehreren Lösungen in einem Verteiler

Beispiel **mit mehreren Lösungen in einem Verteiler und einem Alternativpfad**

```

SWITCH number
CASE 1,2
    SPTP P21 Vel=30% PDAT5 Tool[1] Base[1]
CASE 3,4,5
    SPTP P22 Vel=30% PDAT5 Tool[1] Base[1]
CASE 6
    SPTP Px3Vel=30% PDAT5 Tool[1] Base[1]
DEFAULT
    SPTP P24 Vel=30% PDAT5 Tool[1] Base[1]
ENDSWITCH

```



Bei *number* 3,4 oder 5 wird in den zweiten "CASE" gesprungen um diese Anweisung(en) auszuführen.

Verteiler mit dem Inline-Formular programmieren (SWITCH- CASE)

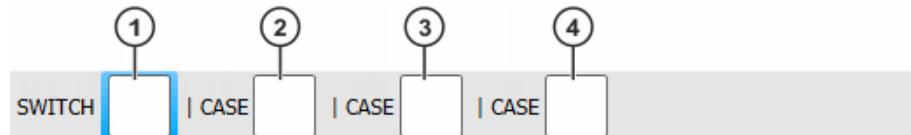


Abb. 14-14: Inline-Formular SWITCH ... CASE

Pos.	Beschreibung
1	Variable für das Auswahlkriterium <ul style="list-style-type: none"> Eine INT- oder CHAR-Variable kann, muss aber nicht vorher deklariert worden sein. Eine ENUM-Variable muss vorher oder nachher händisch deklariert werden.
2	Kennung für den Anweisungsblock Der Datentyp der Kennung muss mit dem Datentyp der Variablen (Pos. 1) übereinstimmen. Ein Anweisungsblock kann beliebig viele Kennungen haben. Mehrere Kennungen in einem CASE-Feld müssen durch Komma voneinander getrennt werden.

Pos.	Beschreibung
3	Wie Pos. 2.
4	Vorhandene CASE-Felder müssen entweder mit Kennungen gefüllt oder entfernt werden. Über die entsprechenden Schaltflächen können CASE-Felder entfernt oder hinzugefügt werden.

Einschränkungen bei "Ändern"

Über die Schaltfläche **Ändern** kann kein CASE hinzugefügt oder entfernt werden.

Mögliche Änderungen:

- Wenn der Cursor in der Zeile mit SWITCH steht, kann die Variable geändert werden.
- Wenn der Cursor in einer Zeile mit CASE steht, kann der Wert geändert werden.

14.5 Sprungbefehl programmieren

Beschreibung

Unbedingter Sprung an spezifizierte Stelle im Programm. Das Programm wird an dieser Stelle fortgesetzt.

Das Sprungziel muss sich im selben Programmteil 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.



Bei einem unbedingten Sprung wird seitens des Programms nicht geprüft, ob eine bestimmte Bedingung erfüllt oder nicht erfüllt wurde. Der Sprung wird immer ausgeführt. Ein evtl. auftretender, ungewünschter Sprung muss seitens der Programmierung abgefangen werden. Siehe auch zweites Programmbeispiel. Das Programmieren mit GOTO kann Programme unübersichtlich machen. Stattdessen besser mit Verzweigungen, Verteilern oder Schleifen arbeiten.

Syntax

```
...
GOTO Marke
...
Marke:
...
```

Element	Beschreibung
Marke	Stelle, an die gesprungen wird. An der Zielstelle muss Marke am Ende einen Doppelpunkt haben.

Beispiele

- Unbedingter Sprung zur Programmstelle `GLUE_STOP`.

```
GOTO GLUE_STOP  
...  
GLUE_STOP:
```

- Umwandlung eines unbedingten Sprunges in einen bedingten Sprung durch die Erweiterung mit einer IF- Anweisung. Der Sprung erfolgt zur Programmstelle `GLUE_END`.

```
IF X>100 THEN  
    GOTO GLUE_END  
ELSE  
    X=X+1  
ENDIF  
A=A*X  
...  
GLUE_END:  
END
```

14.6 Wartefunktionen in KRL programmieren



Abb. 14-15: WAIT, GO

Programmierung in KRL von

- zeitabhängiger Wartefunktion
- signalabhängiger Wartefunktion

14.6.1 Zeitabhängige Wartefunktion

Beschreibung einer zeitabhängigen Wartefunktion mit KRL



Abb. 14-16: Stoppuhr

- die zeitabhängige Wartefunktion wartet die angegebene Zeit (`time`) bevor der Prozess fortgeführt werden kann
- Syntax

```
WAIT SEC time
```

Zeitabhängige Wartefunktion mit dem Inline-Formular programmieren

Die Wartezeit lässt sich bequem mit dem Inline-Formular programmieren.. Die Angabe erfolgt in Sekunden.

```
WAIT Time= 2.5 [sec]
```

Abb. 14-17: Inline-Formular WAIT

Das eingefügte Inline-Formular selbst ist eine Falte und lässt sich öffnen. In grün wird der entsprechende KRL-Befehl angezeigt.

```
7 ➔ WAIT Time= 2.5 sec
8   WAIT SEC 2.5
```

Abb. 14-18: geöffneter Fold

Prinzip der zeitabhängigen Wartefunktion

- die zeitabhängige Wartefunktion hat die Zeitbasis Sekunden (s)
- die maximale Zeit beträgt 2147484 Sekunden, das sind mehr als 24 Tage



Das Inlineformular für die zeitabhängige Wartefunktion kann maximal 30 Sekunden warten

- der Zeitwert kann auch mit einer geeigneten Variable übermittelt werden
- die kleinste sinnvolle Zeiteinheit ist 0.012 Sekunden (IPO-Takt)
- ist die angegebene Zeit negativ, so wird nicht gewartet

- eine zeitabhängige Wartefunktion löst einen Vorlaufstop aus, somit ist ein Überschleifen nicht möglich
- um gezielt nur einen Vorlaufstop zu generieren, wird der Befehl WAIT SEC 0 verwendet

Programmieren einer zeitabhängigen Wartefunktion

- zeitabhängige Wartefunktion mit einer festen Zeit

```
SPTP P1 Vel=100% PDAT1
SPTP P2 Vel=100% PDAT2
WAIT SEC 5.25
SPTP P3 Vel=100% PDAT3
```

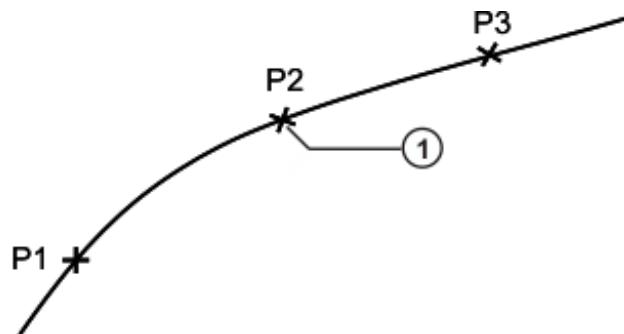


Abb. 14-19: Beispielbewegung für Logik

Pos.	Bemerkung
1	Bewegung wird für 5.25 Sekunden am Punkt P2 unterbrochen

- zeitabhängige Wartefunktion mit einer berechneten Zeit

```
WAIT SEC 3*0.25
```

- zeitabhängige Wartefunktion mit einer Variablen

```
DECL REAL time
time = 12.75
WAIT SEC time
```

14.6.2 Signalabhängige Wartefunktion

Beschreibung

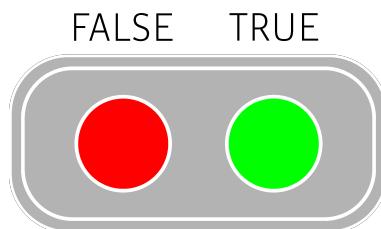


Abb. 14-20: TRUE, FALSE

- die signalabhängige Wartefunktion schaltet bei erfüllter Bedingung (condition) weiter und der Prozess wird fortgeführt
- Syntax

WAIT FOR condition

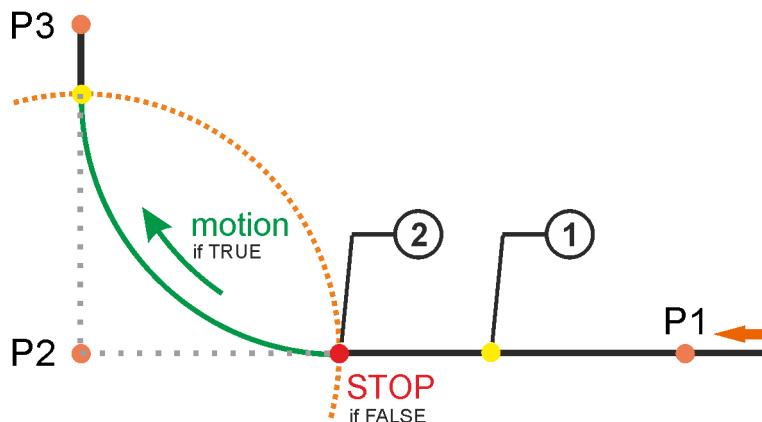
Prinzip der signalabhängigen Wartefunktion

Abb. 14-21: WAIT FOR bei Überschleifbewegungen

- die signalabhängige Wartefunktion löst einen Vorlaufstop aus, somit ist ein Überschleifen nicht möglich
- trotz schon erfüllter Bedingung wird ein Vorlaufstop generiert
- wird in der Zeile unmittelbar vor dem Wartebefehl, der Befehls CONTINUE programmiert, so kann bei rechtzeitig erfüllter Bedingung ein Vorlaufstop verhindert werden



CONTINUE bezieht sich immer auf die nächste Zeile, auch wenn diese eine Leerzeile ist.

Programmieren einer signalabhängigen Wartefunktion

- WAIT FOR mit Vorlaufstop

```
SPTP P1 Vel=100% PDAT1
SPTP P2 CONT Vel=100% PDAT2
WAIT FOR $IN[20]==TRUE
SPTP P3 Vel=100% PDAT3
```

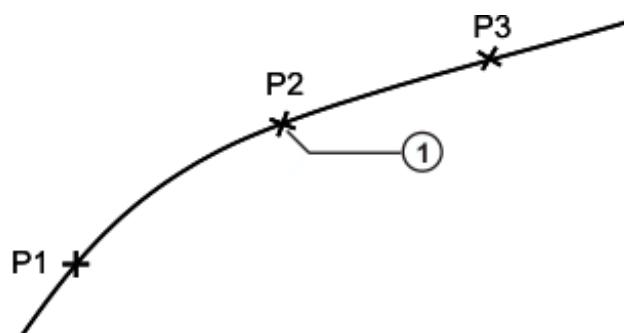


Abb. 14-22: Beispielbewegung für Logik

	Bemerkung
1	Bewegung wird am Punkt P2 unterbrochen. Nach dem Genuholt wird der Eingang 20 geprüft. Ist der Eingangszustand zutreffend kann direkt weitergefahren werden, ansonsten wird auf den Zustand gewartet.

- WAIT FOR mit Bearbeitung im Vorlauf (Einsatz von CONTINUE)

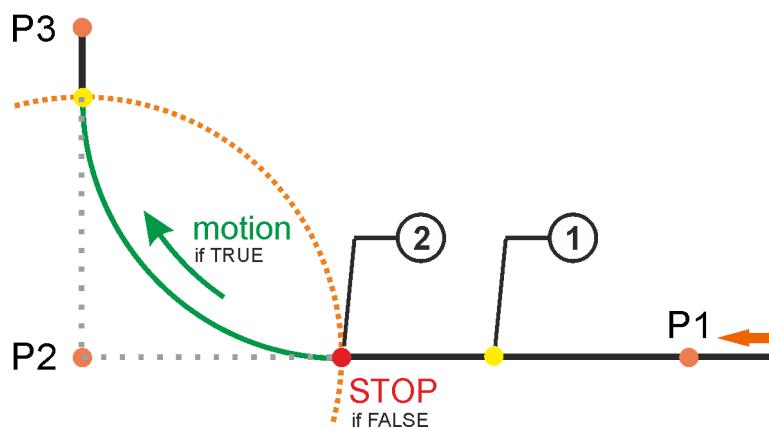


Abb. 14-23: WAIT FOR bei Überschleifbewegungen

```

SPTP P1 Vel=100% PDAT1
SPTP P2 CONT Vel=100% PDAT2
CONTINUE
WAIT FOR ($IN[10] OR $IN[20])
SPTP P3 Vel=100% PDAT3

```

	Aktion
1	Eingang 10 oder Eingang 20 sind oder waren bereits im Vorlauf auf TRUE. Somit wird überschliffen
2	<ul style="list-style-type: none"> - Bedingung kurz vorher erfüllt, somit wird die Bewegung überschliffen - Bedingung zu spät erfüllt, damit kann die Bewegung nicht überschliffen werden und es wird an der Überschleifgrenze angehalten. Der Roboter wartet an der Überschleifgrenze solange, bis die Bedingung erfüllt ist. Die Anzeige hierzu im Meldungsfenster lautet "Warte auf (Eingang 10 oder Eingang 20). In den Testbetriebsarten (T1 oder T2) hat man die Schaltfläche Simuliere zur Verfügung.

Signalabhängige Wartefunktion mit dem Inline-Formular programmieren

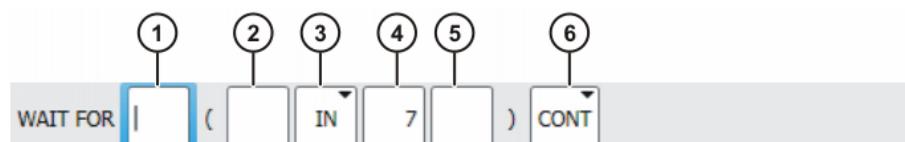


Abb. 14-24: Inline-Formular WAIT FOR

Pos.	Beschreibung
1	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer] <p>Äußere Verknüpfung hinzufügen. Der Term steht zwischen den geklammerten Ausdrücken.</p> <p>Den gewünschten Term über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR

Pos.	Beschreibung
2	<ul style="list-style-type: none"> • NOT: NOT hinzufügen. • [leer]
	<p>Innere Verknüpfung hinzufügen. Der Operator steht innerhalb eines geklammerten Ausdrucks.</p> <p>Den gewünschten Operator über die entsprechende Schaltfläche hinzufügen:</p> <ul style="list-style-type: none"> • AND, OR oder EXOR
3	<p>Signal, auf das gewartet wird. Default-Auswahl:</p> <ul style="list-style-type: none"> • IN, OUT, CYCFLAG, TIMER oder FLAG <p>Über die Schaltfläche Anwender Liste kann statt der Default-Auswahl eine Liste mit nutzerdefinierten Variablen eingeblendet werden. Voraussetzung: Die Liste wurde konfiguriert.</p> <p>System Liste blendet wieder die Default-Auswahl ein. Durch Berühren der Schaltflächen kann zwischen ihnen gewechselt werden.</p>
4	<ul style="list-style-type: none"> • Wenn ein Signal aus der Default-Auswahl ausgewählt wurde: Nummer eingeben. • Wenn eine Variable aus der Anwenderliste ausgewählt wurde: Wert eingeben.
5	<p>Dieses Feld ist nur vorhanden, wenn ein Signal aus der Default-Auswahl gewählt wurde. Wenn für das Signal ein Name existiert, wird er angezeigt.</p> <p>Ab Benutzergruppe Experte: Ein Name kann eingegeben werden oder der bestehende Name geändert werden. Solange das Inline-Formular noch nicht gespeichert wurde, können diese Änderungen durch Drücken auf Langtext wieder zurückgesetzt werden.</p>
6	<ul style="list-style-type: none"> • CONT: Bearbeitung im Vorlauf • [leer]: Bearbeitung mit Vorlaufstopp

14.7 Übung: Schleifentechniken

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- **Schleifentechniken**

15 Arbeiten mit einer übergeordneten Steuerung

15.1 Lerneinheit: Arbeiten mit einer übergeordneten Steuerung

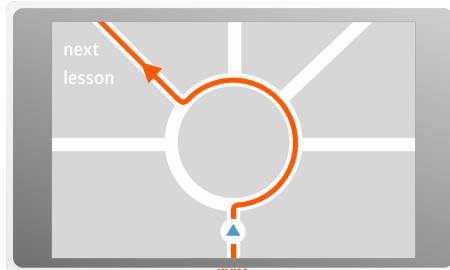


Abb. 15-1: Navigator

Folgende Inhalte werden in dieser Lerneinheit vermittelt:

- Vorbereitung zum Programmstart von SPS
- SPS Anbindung anpassen (Cell.src)

15.2 Vorbereitung zum Programmstart über die SPS

Roboter im Anlagenverbund

Wenn Roboterprozesse von zentraler Stelle aus gesteuert werden sollen (von einem Leitrechner oder von einer SPS), dann geschieht dies über die Schnittstelle *Automatik Extern*.



Abb. 15-2: SPS Anbindung

Für die Kommunikation zwischen SPS und Roboter ist ein konfigurierter Feldbus, z. B. PROFINET notwendig. Über diesen werden Signale für Roboterprozesse als digitale Ein- und Ausgänge übertragen. Roboterseitig erfolgt die Schnittstellenvereinbarung über die Schnittstelle "Automatik Extern".

Schnittstelle "Automatik Extern"

- **Steuerungssignale zum Roboter (Eingänge):**

Die übergeordnete Steuerung (SPS) übermittelt an die Robotersteuerung Signale für Roboterprozesse (z. B. Fahrerlaubnis, Fehlerquittierung, Programmstart etc.).

- **Roboterstatus (Ausgänge):**

Die Robotersteuerung übermittelt an die übergeordnete Steuerung Informationen über Betriebs- und Störzustände (z.B. NOT-HALT, Betriebsart, HOME-Position, HomePosition etc.).

Außerdem ist

- ein angepasstes **CELL.SRC**

(Organisationsprogramm zur Anwahl der Roboterprogramme von außen)

- und die Anwahl der **Automatik Extern Betriebsart** notwendig.
(Betriebsart, in der ein Leitrechner oder eine SPS die Ansteuerung des Robotersystems übernimmt)

Sicherheitshinweise Programmstart extern

Nach Anwahl des CELL-Programms muss eine SAK-Fahrt in Betriebsart T1 oder T2 durchgeführt werden.



WARNUNG

- Eine SAK-Fahrt erfolgt als PTP-Bewegung von der Istposition zur Zielposition wenn der angewählte Bewegungssatz den Fahrbefehl PTP enthält.
- Enthält der angewählte Bewegungssatz LIN oder CIRC wird die SAK-Fahrt als LIN-Bewegung ausgeführt.
- Die Bewegung beobachten, um Kollisionen zu vermeiden. Bei der SAK-Fahrt wird die Geschwindigkeit automatisch reduziert.

Wenn die SAK-Fahrt einmal erfolgt ist, wird beim externen Start **keine** SAK-Fahrt mehr durchgeführt.



WARNUNG

- 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.

Vorgehensweise Programmstart extern

Voraussetzungen

- Betriebsart T1 oder T2
- Die Ein-/Ausgänge für Automatik Extern und das Programm CELL.SRC sind konfiguriert.

1. Im Navigator das Programm **CELL.SRC** anwählen.

Das CELL-Programm befindet sich immer im Verzeichnis **KRC:\R1**

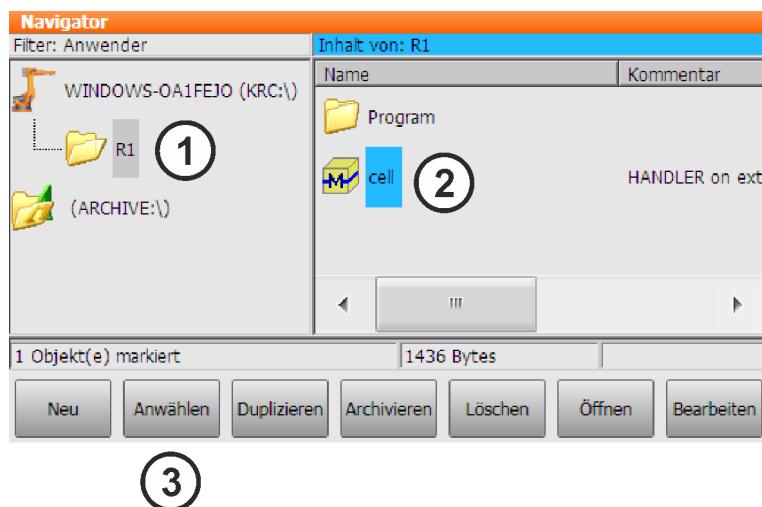


Abb. 15-3: cell.src

2. Programm-Override auf 100 % einstellen. Dies ist die empfohlene Einstellung.

Der Wert kann variabel an die Situation angepasst werden.

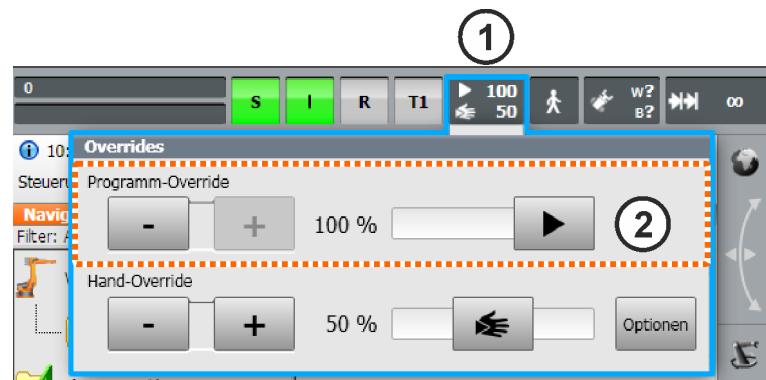


Abb. 15-4: Programm-Override

3. SAK-Fahrt durchführen:
Zustimmungsschalter drücken und halten. Dann Start-Taste drücken und halten, bis im Meldungsfenster "SAK erreicht" angezeigt wird.
4. Betriebsart "Automatik Extern" wählen.
5. Das Programm von einer übergeordneten Steuerung (SPS) aus starten.

15.3 SPS-Anbindung anpassen (Cell.src)

Cell.src Organisationsprogramm

- Zur Verwaltung der von der SPS übertragenen Programmnummern wird das Organisationsprogramm Cell.src verwendet.
- Es befindet sich immer im Ordner "KRC:\R1".
- Wie jedes gewöhnliche Programm kann auch das Cell-Programm individuell angepasst werden, wobei die Grundstruktur des Programmes erhalten bleiben muss.

Aufbau und Funktionalität des Cell-Programms

```

1 ;EXT EXAMPLE1 ( )
2 ;EXT EXAMPLE2 ( )
3 ;EXT EXAMPLE3 ( )
4
5 →INIT
6 BASISTECHINI
7 CHECK HOME
8 PTP HOME Vel= 100 % DEFAULT
9 AUTOEXTINI
10 LOOP
11 P00 (#EXT PGNO,#PGNO_GET,DMY[],0 ) 
12 SWITCH PGNO ; Select with Programnumber
13
14 CASE 1
15 P00 (#EXT PGNO,#PGNO_ACKN,DMY[],0 ) ; Reset
16 Progr.No.-Request
17 ;EXAMPLE1 ( ) ; Call User-Program
18
19 CASE 2
20 P00 (#EXT PGNO,#PGNO_ACKN,DMY[],0 ) ; Reset
21 Progr.No.-Request
22 ;EXAMPLE2 ( ) ; Call User-Program
23
24 CASE 3
25 P00 (#EXT PGNO,#PGNO_ACKN,DMY[],0 ) ; Reset
26 Progr.No.-Request
27 ;EXAMPLE3 ( ) ; Call User-Program
28
29 DEFAULT
30 P00 (#EXT PGNO,#PGNO_FAULT,DMY[],0 )
31 END_SWITCH
32 ENDLOOP
33

```

Abb. 15-5: Cell-Programm

1	Initialisierung und Home-Position <ul style="list-style-type: none"> Initialisierung der Basisparameter Überprüfung der Roboterposition nach der "Home"-Stellung Initialisierung der Automatik-Extern-Schnittstelle Fahrt zur "Home-Position"
2	Endlos-Schleife: <ul style="list-style-type: none"> Abfrage der Programmnummer durch das Modul "P00" Einstieg in die Auswahlschleife mit der ermittelten Programmnummer.
3	Auswahlsschleife Programmnummer <ul style="list-style-type: none"> Entsprechend der Programmnummer (abgelegt in Variable "PGNO") erfolgt der Sprung in den jeweiligen Zweig ("CASE"). Das im Zweig eingetragene Roboterprogramm wird dann abgearbeitet. Ungültige Programmnummern haben zur Folge, dass in den "Default"-Zweig gesprungen wird. Nach erfolgter Abarbeitung wiederholt sich die Schleife.

Vorgehensweise

Aufruf vorhandener Roboterprogramme

1. In die Benutzergruppe "Experte" wechseln.
2. CELL.SRC öffnen.
3. In den Abschnitten "CASE" die Bezeichnung "EXAMPLE" durch den Namen des Programms ersetzen, das von der jeweiligen Programmnummer aufgerufen werden soll. Das Semikolon vor dem Namen löschen.

```
LOOP
    P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
    SWITCH PGNO ; Select with Programnumber

    CASE 1
        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
        main()

    CASE 2
        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
        body_38()
        body_515()

    DEFAULT
        P00 (#EXT_PGNO,#PGNOFAULT,DMY[],0 )
    END_SWITCH
ENDLOOP
```

Abb. 15-6: Beispiel für ein angepasstes Cell-Programm

4. Das Programm schließen und die Änderungen speichern.



Die Roboterprogramme, sowie die damit verknüpften Programmnummern müssen mit dem SPS-Programmierer abgeglichen sein.

15.4 Fragen: Arbeiten an einer übergeordneten Steuerung

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und beantworten Sie folgende Fragen:

- **Arbeiten an einer übergeordneten Steuerung**

16 Anhang

16.1 Beschreibung der Automatik Extern Schnittstelle

Beschreibung

Verwendung der Ein-/Ausgänge der Automatik Extern Schnittstelle

Übersicht über die wichtigsten Signale der Schnittstelle

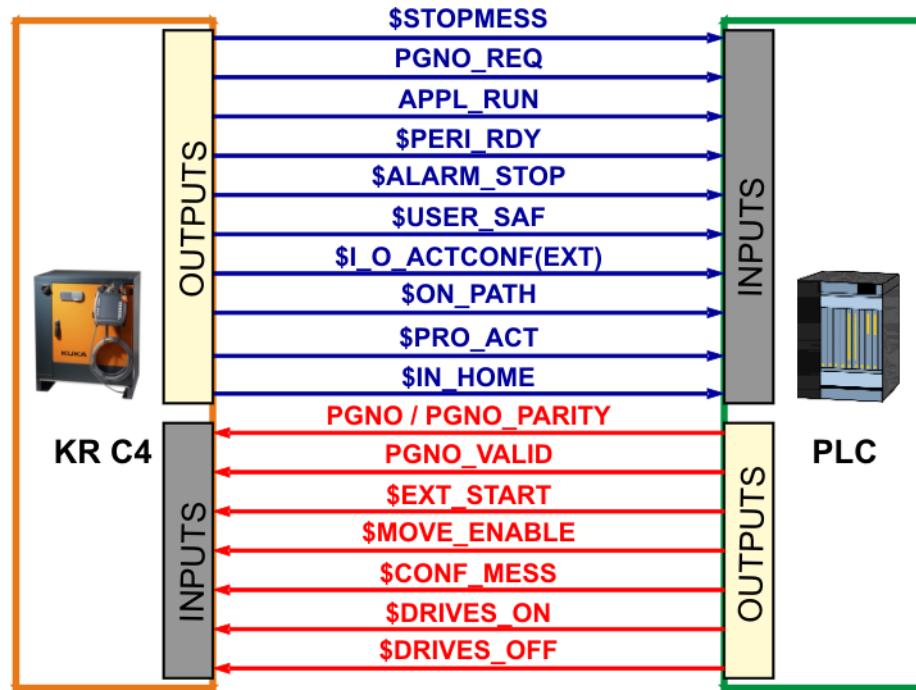


Abb. 16-1: Übersicht der wichtigsten Signale AutomatikExtern

Eingänge (aus Sicht der Robotersteuerung)

- PGNO_TYPE - Programmnummern-Typ

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
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

Wert	Beschreibung	Beispiel
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 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.

- **PGNO_LENGTH** - Programmnummernlänge

Diese Variable legt die Bitbreite der von der übergeordneten Steuerung übermittelten Programmnummer fest. Wertebereich: 1 ... 16.

Wenn PGNO_TYPE den Wert 2 hat, sind nur die Bitbreiten 4, 8, 12 und 16 zugelassen.

- **PGNO_PARITY** - Programmnummer-Paritätsbit

Eingang, auf den das Paritätsbit von der übergeordneten Steuerung übertragen wird.

Eingang	Funktion
Negativer Wert	Ungerade Parität
0	Keine Auswertung
Positiver Wert	Gerade Parität

Wenn PGNO_TYPE den Wert 3 hat, wird PGNO_PARITY nicht ausgewertet.

- **PGNO_VALID** - Programmnummer gültig

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.
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.

- **\$EXT_START** - Externer Start

Mit dem Setzen dieses Eingangs kann bei aktiver E/A-Schnittstelle ein Programm (normalerweise CELL.SRC) gestartet oder fortgesetzt werden.



Es wird nur die ansteigende Flanke des Signals ausgewertet.



WARNUNG

Im Automatik Extern-Betrieb gibt es **keine SAK-Fahrt**. Dies bedeutet, dass der Roboter die erste programmierte Position nach dem Start mit programmierten (nicht reduzierten) Geschwindigkeit anfährt und dort nicht stoppt.

- **\$MOVE_ENABLE** - Fahrfreigabe

Dieser Eingang wird zur Kontrolle der Roboterantriebe durch die übergeordnete Steuerung verwendet.

Signal	Funktion
TRUE	Handverfahren in der Betriebsart T1 und T2 möglich. 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 projektiieren, ist kein externer Start möglich.

- **\$CONF_MESS** - Meldungsquittierung

Durch Setzen dieses Eingangs quittiert die übergeordnete Steuerung Fehlermeldungen selbst, sobald die Störungsursache beseitigt wurde.



Es wird nur die ansteigende Flanke des Signals ausgewertet.

- **\$DRIVES_ON** - Antriebe ein

Wenn an diesem Eingang ein High-Impuls von mindestens 20 ms Dauer anliegt, schaltet die übergeordnete Steuerung die Roboterantriebe ein.

- **\$DRIVES_OFF** - Antriebe aus

Wenn an diesem Eingang ein Low-Impuls von mindestens 20 ms Dauer anliegt, schaltet die übergeordnete Steuerung die Roboterantriebe ab.

Ausgänge (aus Sicht der Robotersteuerung)

- **\$ALARM_STOP** - Not-Halt

Dieser Ausgang wird bei folgenden NOT-HALT-Situationen zurückgesetzt:

- Der NOT-HALT-Taster am KCP wird gedrückt. (Int. NotAus)
- 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 smart-PAD ausgelöst
- **\$ALARM_STOP** ist FALSE, **\$ALARM_STOP_INTERN** ist TRUE: Externer NOT-HALT

- **\$ALARM_STOP_INTERN** - Signalvereinbarung für den internen NOT-HALT

Wenn dieser Ausgang TRUE ist, liegt kein interner NOT-HALT vor. Bei einem internen NOT-HALT setzt die Robotersteuerung den Ausgang auf FALSE.



wie bei **\$ALARM_STOP**

- **\$USER_SAF** - Bedienerschutz/Schutztür
Dieser Ausgang wird beim Öffnen des Schutzgitter-Abfrageschalters (Betriebsart AUT) oder beim Loslassen eines Zustimmungsschalters (Betriebsart T1 oder T2) zurückgesetzt.
- **\$PERI_RDY** - Antriebe bereit
Mit Setzen dieses Ausgangs teilt die Robotersteuerung der übergeordneten Steuerung mit, dass die Roboterantriebe eingeschaltet sind.
- **\$ROB_CAL**
Dieser Ausgang ist immer dann gesetzt, wenn alle Roboterachsen justiert sind. Sobald eine Achse des Roboters dejustiert ist, wird der Ausgang zurückgesetzt.
- **\$STOPMESS** - Stoppmeldungen
Dieser Ausgang wird von der Robotersteuerung gesetzt, um der übergeordneten Steuerung das Auftreten einer Meldung anzuzeigen, die das Anhalten des Roboters erforderlich macht. (Beispiele: NOT-HALT, Fahrfreigabe oder Bedienerschutz)
- **\$I_O_ACTCONF** - Automatik Extern aktiv
Dieser Ausgang ist TRUE, wenn die Betriebsart *Automatik Extern* ausgewählt ist und der Eingang \$I_O_ACT TRUE (normalerweise immer auf \$IN[1025]) ist.
- **\$PRO_ACT** - Programm ist aktiv/läuft
Dieser Ausgang ist immer dann gesetzt, wenn ein Prozess auf Roboterbene 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.
- **PGNO_REQ** - Programmnummernanfrage
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** - Applikationsprogramm läuft
Mit dem Setzen dieses Ausgangs teilt die Robotersteuerung der übergeordneten Steuerung mit, dass gerade ein Programm abgearbeitet wird.
- **\$IN_HOME** - Roboter in HOME-Position
Dieser Ausgang teilt der übergeordneten Steuerung mit, ob sich der Roboter in seiner HOME-Position befindet.
- **\$ON_PATH** - Roboter ist auf der Bahn
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 so lange 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** - Signalvereinbarung für das Toleranzfenster um \$POS_RET
Mit dem Setzen des Ausgangs teilt die Robotersteuerung der übergeordneten Steuerung mit, dass 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 Variable \$NEARPATHTOL definiert werden.

- **\$ROB_STOPPED** - Signalvereinbarung für den Stillstand des Roboters
Dieser Ausgang 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.

Prinzip der Automatik Extern Kommunikation

Übersicht Gesamtablauf

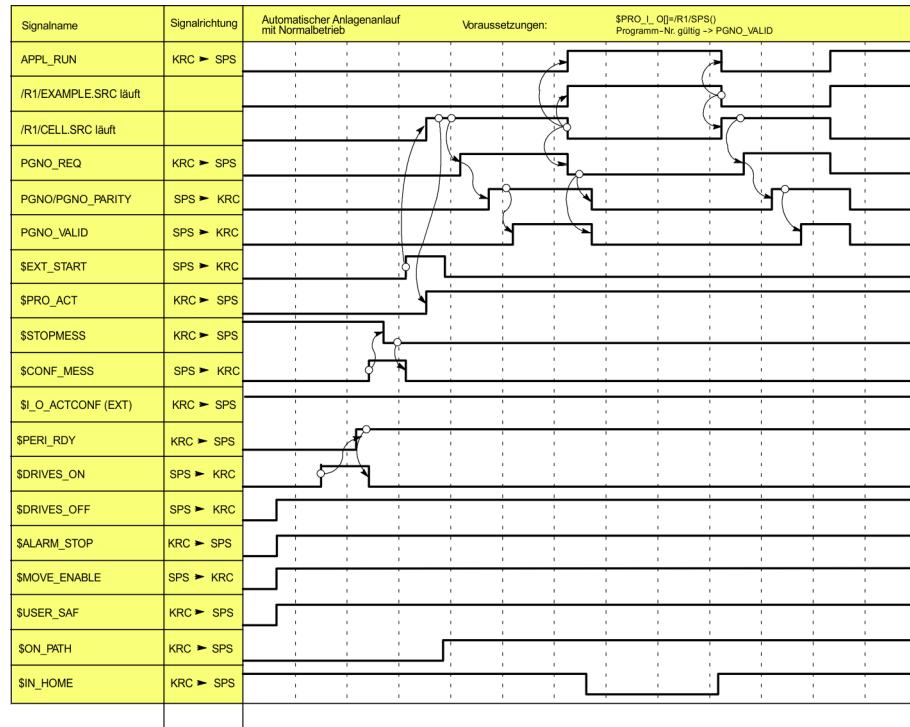


Abb. 16-2: Automatischer Anlagenanlauf und Normalbetrieb mit Quittierung der Programmnummer durch PGNO_VALID

Handshake

Untergliederung in Teilbereiche

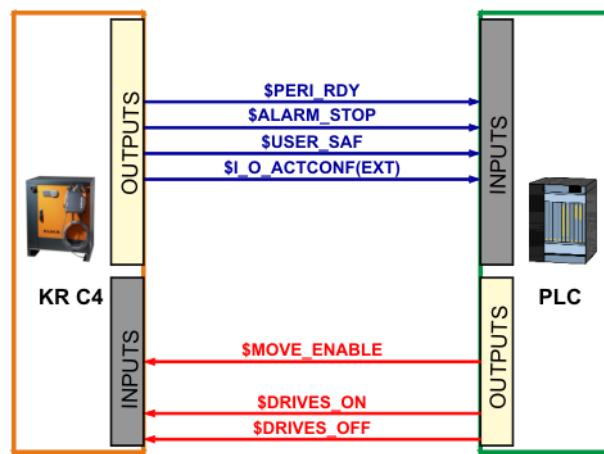
1. Antriebe einschalten
(>>> "Antriebe einschalten" Seite 428)
2. Meldungen quittieren
(>>> "Meldungen quittieren" Seite 429)
3. Cell-Programm starten
(>>> "Programm (CELL.SRC) von extern starten" Seite 429)
4. Programmnummer übergeben und Applikation abarbeiten
(>>> "Programmnummernübergabe und Applikationsprogramm abarbeiten" Seite 430)

Jeder dieser Bereiche ist als Handshake ausgeführt. Es sind Bedingungen zu erfüllen, die SPS schickt Signale und die Robotersteuerung meldet die jeweiligen Roboterzustände zur SPS.

**Abb. 16-3: Handshake**

Es ist sinnvoll diese vorgegebenen Handshakes zu nutzen.

Antriebe einschalten

**Abb. 16-6**

- Voraussetzungen
 - \$USER_SAF - Schutztüre geschlossen
 - \$ALARM_STOP - es liegt kein Not-Halt an
 - \$I_O_ACTCONF - Automatik Extern ist aktiv
 - \$MOVE_ENABLE - Fahrerlaubnis vorhanden
 - \$DRIVES_OFF - Antriebe aus ist nicht aktiviert
- Antriebe einschalten
\$DRIVES_ON - Signal für Antriebe für mind. 20 ms einschalten
- Antriebe bereit
\$PERI_RDY - sobald die Rückmeldung für Antriebe kommt, wird das Signal \$DRIVES_ON zurückgenommen

Meldungen quittieren

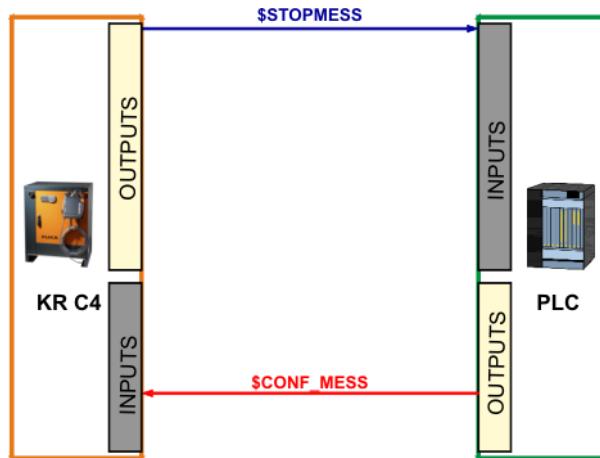


Abb. 16-10

- Voraussetzungen
\$STOPMESS - Stopmeldung steht an
- Meldung quittieren
\$CONF_MESS - Meldung quittieren
- Quittierbare Meldungen sind gelöscht
\$STOPMESS - Stopmeldung steht nicht mehr an \$CONF_MESS kann jetzt zurückgenommen werden

Programm (CELL.SRC) von extern starten

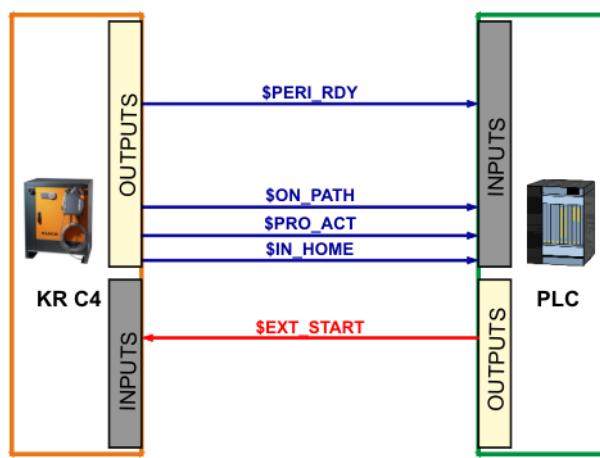


Abb. 16-15

- Voraussetzungen
 - \$PERI_RDY - Antriebe sind bereit
 - \$IN_HOME - Roboter ist in HOME-Position
 - keine \$STOPMESS - keine Stopmeldung steht an
- Externer Start
\$EXT_START - externer Start einschalten (positive Flanke)
- CELL Programm läuft
 - \$PRO_ACT - meldet CELL Programm läuft
 - \$ON_PATH - sobald sich der Roboter auf seiner programmierten Bahn befindet, wird das Signal \$EXT_START zurückgenommen

Programmnummernübergabe und Applikationsprogramm abarbeiten

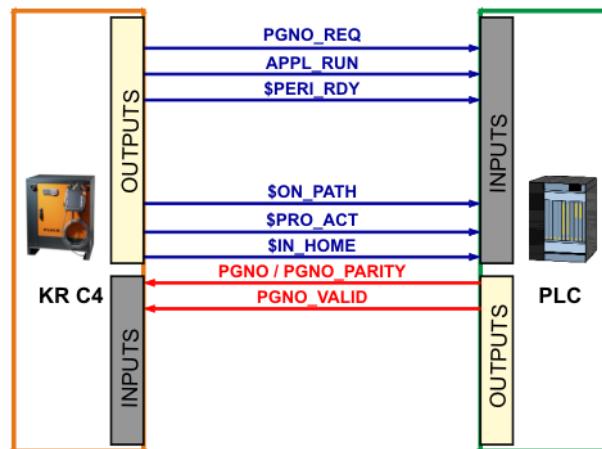


Abb. 16-22

- Voraussetzungen
 - \$PERI_RDY - Antriebe sind bereit
 - \$PRO_ACT - CELL Programm läuft
 - \$ON_PATH - Roboter ist auf der Bahn
 - \$IN_HOME - Roboter ist in HOME-Position, bei Wiederanlauf nicht notwendig
 - PGNO_REQ - Programmnummernanfrage steht an
- Programmnummernübertrag und Bestätigung
 - Programmnummerübergabe
(richtiger Datentyp (PGNO_TYPE), Programmnummerlänge (PGNO_LENGTH) und erstes Bit für Programmnummer (PGNO_FBIT) sind eingestellt)
 - PGNO_VALID- Programmnummer gültig (Bestätigung)schalten (positive Flanke)
- Applikationsprogramm läuft
 - APPL_RUN - meldet Applikationsprogramm läuft
 - Roboter verlässt die HOME-Position. Bei Beendigung des Applikationsprogramms kommt Roboter wieder auf die HOME-Position zurück.

16.1.1 Konfiguration der Automatik-Extern-Schnittstelle

Vorgehensweise

Automatik Externkonfiguration: Eingänge

1. **Menüpfad:** Robotertaste > Konfiguration > Ein-/Ausgänge > Automatik Extern
2. In der Spalte **Wert** die Zelle markieren, die bearbeitet werden soll und auf **Bearbeiten** (C) 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.

Automatik Extern-Konfiguration: Eingänge				
	Bezeichnung	Typ	Name	Wert
1	Typ Programm Nr.	Var	PGNO_TYPE	1
2	Programmnummernspiegelung	Var	REFLECT_PROG_I_0	
3	Bitbreite Programm Nr.	Var	PGNO_LENGTH	4
4	Erstes Bit Programm Nr.	IO	PGNO_FBIT	1
5	Paritätsbit	IO	PGNO_PARITY	5
6	Programm Nr. gültig	IO	PGNO_VALID	6
7	Programmstart	IO	\$EXT_START	17
8	Fahrfreigabe	IO	\$MOVE_ENABLE	8
9	Fehlerquittung	IO	\$CONF_MESS	7
10	Antriebe aus (invers)	IO	\$DRIVES_OFF	10
11	Antriebe ein	IO	\$DRIVES_ON	9
12	Schnittstelle aktivieren	IO	\$I_O_ACT	1025

Legend: SC (selected), Log (yellow), Anzeige (grey), Ausgänge (grey), Bearbeiten (grey), **A**, **B**, **C**.

Abb. 16-25: Automatik Extern-Konfiguration: Eingänge

Pos.	Beschreibung
1	Nummer
2	Langtextname des Ein-/Ausgangs
3	Typ <ul style="list-style-type: none"> • Grün: Ein-/Ausgang • Gelb: Variable oder Systemvariable (\$...)
4	Name des Signals oder der Variable
5	Ein-/Ausgangsnummer oder Kanalnummer
B	Die Ausgänge sind thematisch in Registerkarten eingeteilt.

Konfiguration Automatik Extern: Ausgänge

1. Menüpfad: *KUKA Taste > Konfiguration > Ein-/Ausgänge > Automatik Extern*
2. Über die Schaltfläche **Ausgänge** (B) (>> Abb. 16-25) in die entsprechende Übersicht wechseln.
3. In der Spalte **Wert** die Zelle markieren, die bearbeitet werden soll und auf **Bearbeiten** drücken.
4. Den gewünschten Wert eingeben und mit **Ok** speichern.

5. Schritte 2. und 3. für alle zu bearbeitenden Werte wiederholen.
6. Alle Werte sind in Gruppen sortiert.

- **Startbedingungen (A)**

The screenshot shows a table titled "Automatik Extern-Konfiguration: Ausgänge". The table has columns: Bezeichnung (Description), Typ (Type), Name (Name), and Wert (Value). There are 9 rows of data:

Bezeichnung	Typ	Name	Wert
1 Steuerung bereit	NO	\$RC_RDY1	137
2 Notauskreis geschlossen	NO	\$ALARM_STOP	7
3 Bedienerschutz geschlossen	NO	\$USER_SAF	6
4 Antriebe bereit	NO	\$PERI_RDY	1012
5 Roboter justiert	NO	\$ROB_CAL	1001
6 Schnittstelle aktiv	NO	\$I_O_ACTCONF	140
7 Sammelstörung	NO	\$STOPMESS	1010
8 Erstes Bit für Programm spiegelung	NO	PGNO_FBIT_REFL 999	
9 Interner Not-Halt	NO	\$ALARM_STOP_I	1002

Below the table are navigation buttons: SC, Log, Anzeige, Eingänge, Bearbeiten, and tabs for Startbedingungen (A), Programmstatus (B), Roboterstellung (C), and Betriebsart (D).

Abb. 16-26: Automatik Extern-Konfiguration: Ausgänge

- **Programmstatus (B)**

The screenshot shows a table titled "Automatik Extern-Konfiguration: Ausgänge". The table has columns: Bezeichnung (Description), Typ (Type), Name (Name), and Wert (Value). There are 4 rows of data:

Bezeichnung	Typ	Name	Wert
1 Programm aktiv	NO	\$PRO_ACT	4
2 Programm Nr. Anforderung	NO	PGNO_REQ	3
3 Applikation läuft	NO	APPL_RUN	5
4 Programm bewegung aktiv	NO	\$PRO_MOVE	1022

Below the table are navigation buttons: SC, Log, Anzeige, Eingänge, Bearbeiten, and tabs for Startbedingungen (A), Programmstatus (B), Roboterstellung (C), and Betriebsart (D).

Abb. 16-27: Ausgänge, Programmstatus

- **Roboterstellung (C)**

Automatik Extern-Konfiguration: Ausgänge				
	Bezeichnung	Typ	Name	Wert
1	In Home-Position	IO	\$IN_HOME	1
2	1. Home-Position	IO	\$IN_HOME1	FALSE
3	2. Home-Position	IO	\$IN_HOME2	FALSE
4	3. Home-Position	IO	\$IN_HOME3	FALSE
5	4. Home-Position	IO	\$IN_HOME4	FALSE
6	5. Home-Position	IO	\$IN_HOME5	FALSE
7	Roboter auf der Bahn	IO	\$ON_PATH	2
8	Roboter nahe der Bahn	IO	\$NEAR_POSRET	147
9	Roboter nicht in Bewegung	IO	\$ROB_STOPPED	1023

Abb. 16-28: Ausgänge, Roboterstellung

- Betriebsart (D)

Automatik Extern-Konfiguration: Ausgänge				
	Bezeichnung	Typ	Name	Wert
1	Test1-Betrieb	IO	\$T1	21
2	Test2-Betrieb	IO	\$T2	22
3	Automatik-Betrieb	IO	\$AUT	23
4	Automatik Extern-Betrieb	IO	\$EXT	24

Abb. 16-29: Ausgänge, Betriebsart

Die einzelnen Reiter nacheinander öffnen und in bekannter Weise die Werte anpassen.

7. Das Fenster schließen. Die Änderungen werden übernommen.

16.1.2 Übung: Konfiguration der Automatik Extern Schnittstelle (manuell)

Aufgabenstellung

Öffnen Sie Ihr kursbegleitendes Übungsheft und führen Sie folgende Übung aus:

- Konfiguration der Automatik Extern Schnittstelle (manuell)

16.2 Abkürzungen

Um Ihnen die Arbeit mit der Steuerung zu erleichtern, finden Sie hier einen Auszug der wichtigsten Abkürzungen kurz erläutert.

Begriff	Beschreibung
CCU	Cabinet Control Unit
CCU_SR	Cabinet Control Unit small robot
CIB	Cabinet Interface Board
CIB_SR	Cabinet Interface Board small robot
CBK	Customer Build Kinematic
CSP	Controler System Panel
Dual-NIC-Karte	Zweifache Netzwerkkarte
EDS	Electronic Data Storage (Speicherkarte)
EGB	Elektrostatisch Gefährdete Bauteile
EMD	Electronic Mastering Device (früher EMT) für die Roboterjustage
EMV	Elektromagnetische Verträglichkeit
FSoE	Fail Safe over EtherCAT
GBE	Giga Bit Ethernet
Katalog	Kann verschiedene Elemente beinhalten, z. B. Vorlagen, Komponenten, Kinematiken.
KCB	KUKA Controller Bus
KEB	KUKA Extension Bus
KCP	KUKA Control Panel (Programmierhandgerät), neuer Begriff: smartPAD
KLI	KUKA Line Interface
KOI	KUKA Operator Panel Interface
KONI	KUKA Optional Network Interface
KPC	KUKA Steuerungs-PC
KPP	KUKA Power-Pack
KPP_SR	KUKA Power-Pack small robot
KRF	Kontrollierte Roboterfahrt
KRL	KUKA Robot Language (KUKA Roboter Programmiersprache)
KSB	KUKA System Bus
KSP	KUKA Servo-Pack
KSP_SR	KUKA Servo-Pack small robot
KSI	KUKA Service Interface
HMI	Human Machine Interface KUKA.HMI ist die KUKA-Bedienoberfläche.

Begriff	Beschreibung
LWL	Lichtwellenleiter
MEMD	Micro Electronic Mastering Device
MCC	Multi Core Computer
MCFB	Motion Control Function Block Programmbaustein zur Programmierung von Bewegungsaufgaben Diese Bausteine sind SPS-konform und KUKA-spezifisch.
MGU	Motor Gear Unit KUKA Motorgetriebekombination für Kinematiken
OPC	OLE (Object Link Embedding) for Process Control
OPI	Operator Panel Interface (Anschluß smartPAD)
PDS	Power Drive System
PMB	Power Management Board
RCD	Residual Current Device; Fehlerstromschutzschalter (FI)
RDC	Resolver Digital Converter
SATA	Serial Advanced Technology Attachment (Datenbus zwischen Prozessor und Festplatte)
SEMD	Standard Electronic Mastering Device
SIB	Safety Interface Board
SBC	Single Brake Control
STO	Safe Torque Off
SION	Safety Input Output Node
USB	Universal Serial Bus (Bussystem zur Verbindung eines Computers mit Zusatzgeräten)
USV	Unterbrechungsfreie Stromversorgung
WorkVisual Katalogeditor	Software für die Erstellung von Katalogelementen für WorkVisual

Index

Bedienung und Handverfahren im Weltkoordinatensystem mittels Verfahrtasten.....	70	Dual-NIC-Karte.....	435
robotergeführtes Werkstück.....	339		
A			
Abfragen.....	401	E	
Abkürzungen.....	435	EDS.....	435
Achsen einzeln bewegen.....	51	einfache Datentypen.....	374, 387
Achsspezifisches Handverfahren.....	51	Einrücken.....	365
Ändern, Bewegungsbefehle.....	236	EMD.....	435
Anhang.....	423	EMV.....	435
Arbeiten an einer übergeordneten Steuerung.....	421	Endlosschleife.....	389
Archivieren.....	187	Erstellung.....	195
Automatik Extern.....	415, 423	Expertenebene.....	355
B			
Bahnfahren und Überschleifen.....	230	Externes Werkzeug und robotergeführtes Werkstück vermessen.....	345
Base Koordinatensystem.....	60		
Basis.....	148		
Basisvermessung Tisch.....	157		
bearbeiten.....	184		
Bedienerschutz.....	33		
Bedienung und achsspezifisches Handverfahren.....	59		
Bedienung und Handverfahren im Weltkoordinatensystem mittels 6D-Maus.....	71		
bedingter Stopp.....	294		
Benutzergruppen.....	83		
ändern.....	86		
wechseln.....	84		
Beschreibung der Automatik Extern Schnittstelle.....	423		
Betriebsart wechseln.....	50		
Betriebsarten der Robotersteuerung.....	47		
Bewegungsbefehle.....	195		
Bit-Operationen.....	382		
C			
CBK.....	435	G	
CCU.....	435	GBE.....	435
CCU_SR.....	435	GOTO.....	407
CIB.....	435	Greifer programmieren.....	323
CIB_SR.....	435	Greifer prüfen	
CIRC-Bewegung.....	208	Inlineformular.....	327
CSP.....	435	Greiferprogrammierung Schild.....	330
D			
Datenhaltung in KRL.....	371	Greiferprogrammierung Stift.....	331
DECL.....	373, 375	Greifersignale.....	313
Deklaration.....	376	GripperTech.....	311
Dialogmeldung.....	44	Grundrechenarten.....	382
Dokumentation des Robotersystems.....	35		
I			
Initialisierung.....	174, 378		
Inkrement.....	56		
Inkrementelles Handverfahren.....	55		
Inline-Formular.....	197		
J			

Justage, Lastjustage mit Offset.....	106	LWL.....	436																																																		
Justieren.....	88																																																				
K																																																					
Katalog.....	435	Manipulation.....	382, 383																																																		
KCB.....	435	MCC.....	436																																																		
KCP.....	435	MCFB.....	436																																																		
KEB.....	435	Mechanik eines KUKA Roboters.....	10																																																		
KLI.....	435	Meldungen der Robotersteuerung lesen und interpretieren.....	43																																																		
KOI.....	435	MEMD.....	436																																																		
Konfiguration der Automatik- Extern-Schnittstelle.....	430	MGU.....	436																																																		
KONI.....	435																																																				
Koordinatensystem.....	60																																																				
KPC.....	435																																																				
KPP.....	435																																																				
KRF.....	435																																																				
KRL.....	435																																																				
KSB.....	435																																																				
KSI.....	435																																																				
KSP.....	435																																																				
KUKA Xpert.....	35																																																				
Anmeldung.....	38																																																				
nutzen.....	40																																																				
Registrierung.....	37																																																				
URL.....	36																																																				
L																																																					
Lasten am Roboter.....	116	Panikstellung.....	53																																																		
Lerneinheit		PDS.....	436																																																		
Arbeiten mit einer übergeordneten Steuerung.....	415	PMB.....	436																																																		
Aufbau und Funktion eines KUKA-Robotersystems.....	9	Prgorammmhandhabung.....	207																																																		
Einführung in die Expertenebene.....	355	Prinzip.....	88																																																		
Inbetriebnahme des Roboters.....	83	Priorität.....	385																																																		
Konfiguration und Programmierung von externen Werkzeugen.....	333	Programm: duplizieren.....	185																																																		
Logische Funktionen im Roboterprogramm nutzen.....	269	Programm: löschen.....	186																																																		
Nutzen von Programmablaufkontrollen.....	389	Programm: umbenennen.....	187																																																		
Programmieren von Spline-Bewegungen.....	245	Programmablaufkontrolle.....	389																																																		
Programmierte Bewegungen erstellen und ändern.....	195	Programmerstellung																																																			
Roboter bewegen.....	43	Ansichten im Editor.....	358	Roboterprogramme ausführen.....	171	Ansichten im Explorer.....	357	Technologiepakete nutzen.....	311	Fehler im Programm finden und beheben.....	360	Umgang mit Programmdateien.....	181	Templates.....	355	Variablen und Vereinbarungen.....	371	Programmierung, externer TCP.....	352	LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436
Ansichten im Editor.....	358																																																				
Roboterprogramme ausführen.....	171	Ansichten im Explorer.....	357	Technologiepakete nutzen.....	311	Fehler im Programm finden und beheben.....	360	Umgang mit Programmdateien.....	181	Templates.....	355	Variablen und Vereinbarungen.....	371	Programmierung, externer TCP.....	352	LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436				
Ansichten im Explorer.....	357																																																				
Technologiepakete nutzen.....	311	Fehler im Programm finden und beheben.....	360	Umgang mit Programmdateien.....	181	Templates.....	355	Variablen und Vereinbarungen.....	371	Programmierung, externer TCP.....	352	LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436								
Fehler im Programm finden und beheben.....	360																																																				
Umgang mit Programmdateien.....	181	Templates.....	355	Variablen und Vereinbarungen.....	371	Programmierung, externer TCP.....	352	LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436												
Templates.....	355																																																				
Variablen und Vereinbarungen.....	371	Programmierung, externer TCP.....	352	LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																
Programmierung, externer TCP.....	352																																																				
LIN-Bewegung.....	208	Programmmodule.....	184	Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																				
Programmmodule.....	184																																																				
Logbuch.....	191	Programmstart.....	176	Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																								
Programmstart.....	176																																																				
Logik.....	285	Programmstart von SPS.....	415	Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																												
Programmstart von SPS.....	415																																																				
Logik, Allgemein.....	269	Q				Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																																
Q																																																					
Logikprogrammierung		einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																																						
einfache Logik.....	274	Quittiermeldung.....	44	Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																																								
Quittiermeldung.....	44																																																				
Logische Operationen.....	382	R				Luftprogramm.....	207	RCD.....	436																																												
R																																																					
Luftprogramm.....	207	RCD.....	436																																																		
RCD.....	436																																																				

M

Manipulation.....	382, 383
MCC.....	436
MCFB.....	436
Mechanik eines KUKA Roboters.....	10
Meldungen der Robotersteuerung lesen und interpretieren.....	43
MEMD.....	436
MGU.....	436

N

NOT-HALT-Taster.....	33
----------------------	----

O

OPC.....	436
OPI.....	436

P

Panikstellung.....	53
PDS.....	436
PMB.....	436
Prgorammmhandhabung.....	207
Prinzip.....	88
Priorität.....	385
Programm: duplizieren.....	185
Programm: löschen.....	186
Programm: umbenennen.....	187
Programmablaufkontrolle.....	389
Programmerstellung	
Ansichten im Editor.....	358
Ansichten im Explorer.....	357
Fehler im Programm finden und beheben.....	360
Templates.....	355
Programmierung, externer TCP.....	352
Programmmodule.....	184
Programmstart.....	176
Programmstart von SPS.....	415

Q

Quittiermeldung.....	44
----------------------	----

R

RCD.....	436
RDW.....	436
Roboter Basics.....	9
Roboterposition	
Anzeige.....	76
smartPAD.....	78
Roboterprogramm.....	173
Robotersicherheit.....	32

Robotersteuerung (V)KR C4.....	12
Roboterprogramm.....	171
Robroot.....	60
S	
SAK.....	174
SATA.....	436
Satzanwahl.....	266
SBC.....	436
Schaltfunktion, einfach.....	280
Schleife, abweisend	395
Schleife, nicht abweisend.....	398
Schleifen.....	389
Schleifentechniken.....	414
Schlüsselwort.....	373
Schutztür.....	33
SEMD.....	436
SIB.....	436
Sicherer Betriebshalt.....	49
Sicherheitshalt STOP 0.....	49
Sicherheitshalt STOP 1.....	50
Sicherheitshalt STOP 2.....	50
Sicherheitshalt 0.....	49
Sicherheitshalt 1.....	50
Sicherheitshalt 2.....	50
Sicherheitshalt, extern.....	34
SION.....	436
smartPAD.....	15
smartPAD an- und abstecken.....	23
smartPAD Grundeinstellungen.....	27
SPLINE, Bewegungsart.....	245
Splinebewegung.....	245
Splineblock.....	245
Sprung.....	407
Sprungbefehl.....	407
SPS-Anbindung	
Cell.src.....	418
SPTP-Bewegung.....	197
Standardfunktionen.....	383
Steuerung herunterfahren.....	167
STO.....	436
STOP 0.....	50
STOP 1.....	50
STOP 2.....	50
Stopp-Kategorie 0.....	50
Stopp-Kategorie 1.....	50
Stopp-Kategorie 2.....	50
Stopp-Reaktionen.....	48
Strichmarkierung, für Justage.....	108
T	
Tool Koordinatensystem.....	60
U	
Überschleifen: SCIRC.....	221
Überschleifen: SLIN.....	221
Überschleifen: SPTP.....	200
Übung....	59, 70, 71, 106, 138, 142, 147, 157, 162, 207, 230, 285, 330, 331, 345, 387, 414
KUKA Xpert Basic nutzen.....	41
Übung, Bewegungsprogrammierung mit externen TCP.....	353
Übung, Handverfahren mit feststehendem Werkzeug.....	351
Übung, Konstantfahrbereich und bedingter Stop.....	310
Übung, Roboterprogramm ausführen.....	179
Übung, Spline.....	309
Übung, Splineblock.....	268
Übung: Konfiguration der Automatik Extern Schnittstelle (manuell).....	434
Unterprogrammaufruf programmieren	370
USB.....	436
USV.....	436
V	
Variable ändern.....	270
Variable, einzeln anzeigen.....	270
Variablen.....	371, 375
Vereinbarungen.....	371
Vergleichsoperationen.....	382
Vermessen.....	148, 334
Vermessung.....	339
Verteiler (SWITCH- CASE).....	404
Verzweigung.....	401
Vorlauf.....	373
W	
WAIT.....	276
Wartefunktion, signalabhängig.....	410
Wartefunktion, zeitabhängig.....	409
Wartemeldung.....	44
Welt Koordinatensystem.....	60
Werkzeug- und Basisverwaltung.....	111
Werkzeug vermessen.....	120
Werkzeuglastdaten.....	116
Werkzeugvermessung Greifer.....	142
Werkzeugvermessung Stift.....	138
WHILE.....	395
Wiederherstellen.....	187
WorkVisual Katalogeditor.....	436
World Koordinatensystem.....	60
Z	
Zählschleife.....	392
Zusatzzlast am Roboter.....	118
Zustandsmeldung.....	44