

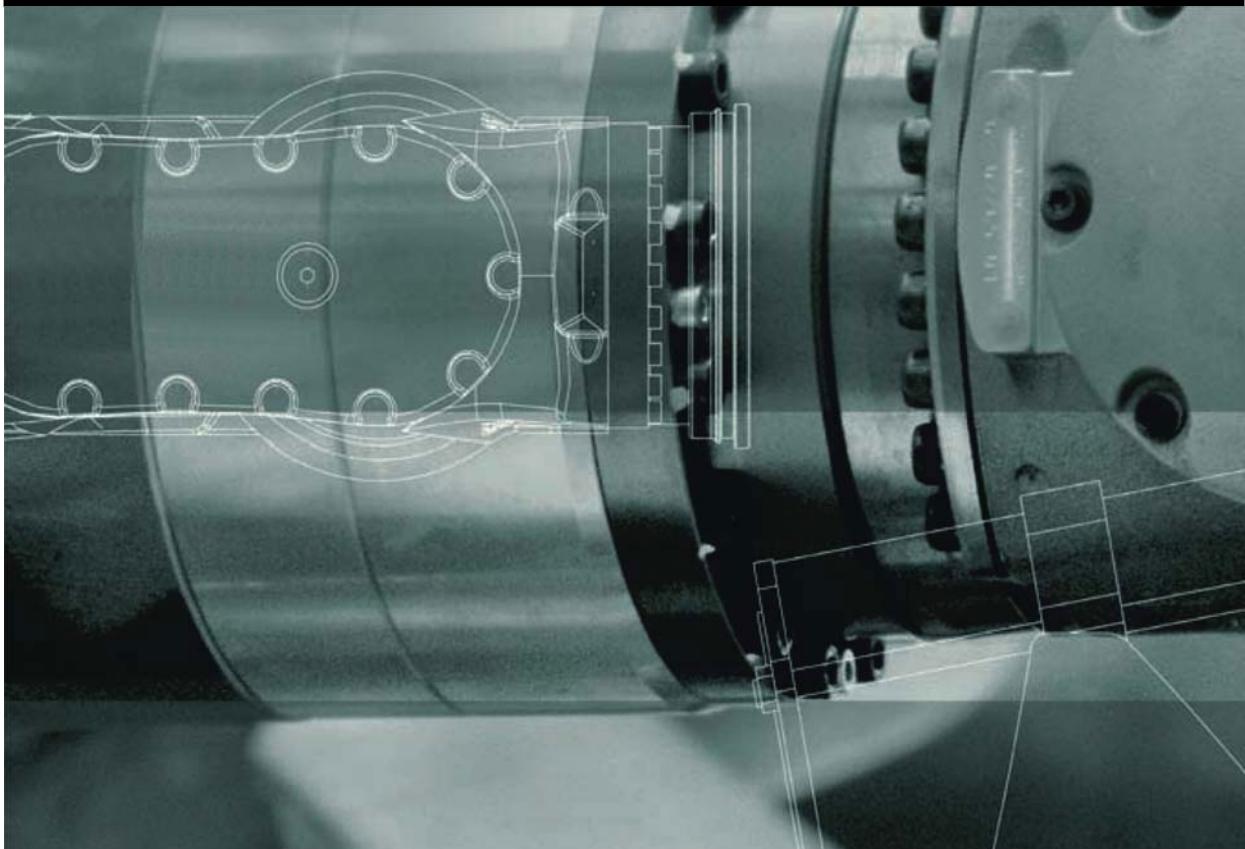
# KUKA

System Software

KUKA Roboter GmbH

## KUKA Sunrise.OS 1.7 KUKA Sunrise.Workbench 1.7

Bedien- und Programmieranleitung



Stand: 02.04.2015

Version: KUKA Sunrise.OS 1.7 V1

© Copyright 2015

KUKA Roboter GmbH  
Zugspitzstraße 140  
D-86165 Augsburg  
Deutschland

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

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

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

Technische Änderungen ohne Beeinflussung der Funktion vorbehalten.

Original-Dokumentation

KIM-PS5-DOC

Publikation: Pub KUKA Sunrise.OS 1.7 (PDF) de

Buchstruktur: KUKA Sunrise.OS 1.7 V1.2

Version: KUKA Sunrise.OS 1.7 V1

## Inhaltsverzeichnis

<b>1 Einleitung</b> .....	15
1.1 Zielgruppe .....	15
1.2 Dokumentation des Industrieroboters .....	15
1.3 Darstellung von Hinweisen .....	15
1.4 Marken .....	16
1.5 Verwendete Begriffe .....	16
<b>2 Produktbeschreibung</b> .....	19
2.1 Übersicht des Robotersystems .....	19
2.2 Übersicht der Software-Komponenten .....	20
2.3 Übersicht KUKA Sunrise.OS .....	20
2.4 Übersicht KUKA Sunrise.Workbench .....	21
2.5 Bestimmungsgemäße Verwendung der System Software .....	22
<b>3 Sicherheit</b> .....	23
3.1 Rechtliche Rahmenbedingungen .....	23
3.1.1 Haftungshinweis .....	23
3.1.2 Bestimmungsgemäße Verwendung des Industrieroboters .....	23
3.1.3 EG-Konformitätserklärung und Einbauerklärung .....	24
3.2 Sicherheitsfunktionen .....	24
3.2.1 Verwendete Begriffe .....	25
3.2.2 Personal .....	27
3.2.3 Arbeits-, Schutz- und Gefahrenbereich .....	28
3.2.4 Sicherheitsgerichtete Funktionen .....	28
3.2.4.1 NOT-HALT-Einrichtung .....	29
3.2.4.2 Zustimmeinrichtung .....	30
3.2.4.3 Bedienerschutz .....	30
3.2.4.4 Externe NOT-HALT-Einrichtung .....	31
3.2.4.5 Externer Sicherheitshalt 1 (bahntreu) .....	31
3.2.4.6 Externe Zustimmeinrichtung .....	31
3.2.4.7 Externer sicherer Betriebshalt .....	32
3.2.5 Auslöser für sicherheitsgerichtete Stopp-Reaktionen .....	32
3.2.6 Nicht sicherheitsgerichtete Funktionen .....	33
3.2.6.1 Betriebsartenwahl .....	33
3.2.6.2 Software-Endschalter .....	35
3.3 Zusätzliche Schutzausstattung .....	35
3.3.1 Tippbetrieb .....	35
3.3.2 Kennzeichnungen am Industrieroboter .....	35
3.3.3 Externe Schutzeinrichtungen .....	35
3.4 Sicherheitsmaßnahmen .....	36
3.4.1 Allgemeine Sicherheitsmaßnahmen .....	36
3.4.2 Transport .....	37
3.4.3 Inbetriebnahme und Wiederinbetriebnahme .....	37
3.4.4 Manueller Betrieb .....	39
3.4.5 Automatikbetrieb .....	40
3.4.6 Wartung und Instandsetzung .....	40
3.4.7 Außerbetriebnahme, Lagerung und Entsorgung .....	41
3.4.8 Sicherheitsmaßnahmen für "Single Point of Control" .....	41

3.5	Angewandte Normen und Vorschriften .....	42
<b>4</b>	<b>Installation KUKA Sunrise.Workbench .....</b>	<b>45</b>
4.1	Systemvoraussetzungen PC .....	45
4.2	Sunrise.Workbench installieren .....	45
4.3	Sunrise.Workbench deinstallieren .....	45
4.4	Sprachpaket in Sunrise.Workbench installieren .....	46
<b>5</b>	<b>Bedienung KUKA Sunrise.Workbench .....</b>	<b>47</b>
5.1	Sunrise.Workbench starten .....	47
5.2	Übersicht Bedienoberfläche Sunrise.Workbench .....	47
5.2.1	Sichten anders anordnen .....	49
5.2.2	Verschiedene Perspektiven der Bedienoberfläche anzeigen .....	49
5.2.3	Symbolleisten .....	50
5.3	Sunrise-Projekt mit Vorlage erstellen .....	51
5.4	Neue Roboter-Applikation erstellen .....	54
5.4.1	Neues Java-Paket erstellen .....	54
5.4.2	Roboter-Applikation mit Paket erstellen .....	54
5.4.3	Roboter-Applikation für bestehendes Paket erstellen .....	55
5.5	Neuen Hintergrund-Task erstellen .....	55
5.5.1	Hintergrund-Task mit Paket erstellen .....	55
5.5.2	Hintergrund-Task für bestehendes Paket erstellen .....	55
5.6	Arbeitsbereich .....	56
5.6.1	Neuen Arbeitsbereich anlegen .....	56
5.6.2	Zu bestehendem Arbeitsbereich wechseln .....	56
5.6.3	Zwischen zuletzt geöffneten Arbeitsbereichen wechseln .....	56
5.6.4	Projekte archivieren .....	57
5.6.5	Projekte aus Archiv in Arbeitsbereich laden .....	57
5.6.6	Projekte aus Verzeichnis in Arbeitsbereich laden .....	57
5.7	Sunrise-Projekte mit referenzierten Java-Projekten .....	58
5.7.1	Neues Java-Projekt erstellen .....	58
5.7.1.1	Roboterspezifische Klassenbibliotheken in Java-Projekt einfügen .....	58
5.7.2	Java-Projekte referenzieren .....	59
5.7.3	Referenzierung zu Java-Projekten aufheben .....	59
5.8	Element im <b>Paket-Explorer</b> umbenennen .....	60
5.8.1	Projekt oder Java-Paket umbenennen .....	60
5.8.2	Java-Datei umbenennen .....	60
5.9	Element aus <b>Paket-Explorer</b> entfernen .....	60
5.9.1	Element aus Projekt löschen .....	60
5.9.2	Projekt im <b>Paket-Explorer</b> entfernen .....	60
5.9.3	Projekt aus Arbeitsbereich löschen .....	61
5.10	Automatische Änderungserkennung aktivieren .....	61
<b>6</b>	<b>Bedienung KUKA smartPAD .....</b>	<b>63</b>
6.1	Bedienhandgerät KUKA smartPAD .....	63
6.1.1	Vorderseite .....	63
6.1.2	Rückseite .....	65
6.2	Robotersteuerung ein-/ausschalten .....	66
6.2.1	Robotersteuerung einschalten und System Software starten .....	66

6.2.2	Robotersteuerung ausschalten .....	66
6.3	Automatisches Update der smartPAD-Software .....	66
6.4	Bedienoberfläche KUKA smartHMI .....	67
6.4.1	Navigationsleiste .....	68
6.4.2	Statusanzeige .....	69
6.4.3	Tastatur .....	70
6.4.4	Stationssicht .....	70
6.4.5	Robotersicht .....	72
6.5	Hauptmenü aufrufen .....	73
6.6	Betriebsart wechseln .....	74
6.7	Koordinatensysteme .....	76
6.8	Roboter manuell verfahren .....	78
6.8.1	Fenster <b>Handverfahroptionen</b> .....	78
6.8.2	Hand-Override (HOV) einstellen .....	80
6.8.3	Mit Verfahrtasten achsspezifisch verfahren .....	80
6.8.4	Mit Verfahrtasten kartesisch verfahren .....	81
6.8.4.1	Nullraum-Bewegung .....	82
6.9	Betriebsart KRF – Roboter kontrolliert freifahren .....	82
6.10	Roboter von Hand führen .....	83
6.11	Sicherheitssteuerung fortsetzen .....	83
6.12	Haltebremsen öffnen .....	84
6.13	Frames teachen und manuell anfahren .....	85
6.13.1	Frames anzeigen .....	85
6.13.2	Frames teachen .....	86
6.13.3	Fenster <b>Verfahrart</b> .....	88
6.13.4	Frames manuell anfahren .....	89
6.14	Programmausführung .....	90
6.14.1	Roboter-Applikation anwählen .....	90
6.14.2	Programmablaufart auswählen .....	92
6.14.3	Manuellen Override einstellen .....	93
6.14.4	Programm vorwärts starten (manuell) .....	94
6.14.5	Programm vorwärts starten (automatisch) .....	94
6.14.6	Roboter nach Verlassen der Bahn rückpositionieren .....	94
6.15	Benutzertasten aktivieren .....	95
6.16	Anzeigefunktionen .....	96
6.16.1	Ziel-Frame der aktuell ausgeführten Bewegung anzeigen .....	96
6.16.2	Achsspezifische Istposition anzeigen .....	97
6.16.3	Kartesische Istposition anzeigen .....	98
6.16.4	Achsspezifische Momente anzeigen .....	98
6.16.5	E/A-Gruppe anzeigen und Wert eines Ausgangs ändern .....	99
6.16.6	IP-Adresse und Software-Version anzeigen .....	101
6.16.7	Robotertyp und Seriennummer anzeigen .....	101
6.16.8	Meldungen des Virenscanners anzeigen .....	101
<b>7</b>	<b>Inbetriebnahme und Wiederinbetriebnahme</b> .....	103
7.1	Positionsjustage .....	103
7.1.1	Achsen justieren .....	103
7.1.2	Achsen manuell dejustieren .....	103
7.2	Vermessen .....	104

7.2.1	Werkzeug vermessen .....	104
7.2.1.1	TCP vermessen: XYZ 4-Punkt-Methode .....	105
7.2.1.2	Orientierung festlegen: ABC 2-Punkt-Methode .....	107
7.2.1.3	Orientierung festlegen: ABC Welt-Methode .....	109
7.2.2	Basis vermessen: 3-Punkt-Methode .....	110
7.3	Werkzeug-Lastdaten ermitteln .....	112
<b>8</b>	<b>Bremsentest .....</b>	<b>117</b>
8.1	Übersicht Bremsentest .....	117
8.2	Bremsentest-Applikation aus Vorlage erstellen .....	120
8.2.1	Bremsentest-Applikation für Test gegen minimales Bremsenhaltemoment anpassen	122
8.2.2	Bewegungsfolge für Momentenwert-Ermittlung ändern .....	123
8.2.3	Ausgangsposition für Bremsentest ändern .....	123
8.3	Programmierschnittstelle für den Bremsentest .....	124
8.3.1	Auftretende Momente auswerten und maximalen Absolutwert ermitteln .....	124
8.3.2	Ergebnis der Auswertung der maximalen absoluten Momente abfragen .....	125
8.3.3	Objekt für den Bremsentest erzeugen .....	127
8.3.4	Ausführung des Bremsentests starten .....	128
8.3.5	Bremsentest auswerten .....	129
8.3.5.1	Ergebnis des Bremsentests abfragen .....	131
8.4	Bremsentest durchführen .....	133
8.4.1	Ergebnis der Auswertung der maximalen absoluten Momente (Anzeige) .....	134
8.4.2	Ergebnis des Bremsentests (Anzeige) .....	134
<b>9</b>	<b>Projektverwaltung .....</b>	<b>137</b>
9.1	Sunrise-Projekte – Übersicht .....	137
9.2	Frame-Verwaltung .....	137
9.2.1	Neuen Frame anlegen .....	138
9.2.2	Frame als Basis kennzeichnen .....	138
9.2.3	Frames verschieben .....	139
9.2.4	Frames löschen .....	140
9.2.5	Eigenschaften eines Frames anzeigen und ändern .....	140
9.2.6	Frame in Bewegungsanweisung einfügen .....	142
9.3	Objektverwaltung .....	142
9.3.1	Geometrischer Aufbau von Werkzeugen .....	143
9.3.2	Geometrischer Aufbau von Werkstücken .....	144
9.3.3	Werkzeug oder Werkstück anlegen .....	144
9.3.4	Frame für Werkzeug oder Werkstück anlegen .....	145
9.3.5	Standard-Frame für Bewegungen festlegen .....	146
9.3.6	Lastdaten .....	147
9.3.6.1	Lastdaten eingeben .....	147
9.3.7	Sicherheitsgerichtetes Werkzeug .....	148
9.3.7.1	Sicherheitsgerichtetes Werkzeug definieren .....	149
9.3.8	Sicherheitsgerichtete Werkstücke .....	152
9.3.8.1	Sicherheitsgerichtete Werkstücke definieren .....	154
9.4	Synchronisation von Projekten .....	155
9.4.1	Projekt auf Robotersteuerung übertragen .....	155
9.4.2	Projekt auf Robotersteuerung oder in Sunrise.Workbench aktualisieren .....	156
9.5	Projekt von Robotersteuerung laden .....	158

<b>10 Stationskonfiguration und Installation .....</b>	159
10.1 Stationskonfiguration öffnen .....	159
10.1.1 Parameter für Vermessung konfigurieren .....	159
10.2 System Software installieren .....	160
10.2.1 Sicherheitskonfiguration in neue Software-Version konvertieren .....	161
10.3 Sprachpaket installieren .....	161
10.4 VirensScanner installieren oder updaten .....	162
<b>11 Buskonfiguration .....</b>	165
11.1 Konfiguration und E/A-Verschaltung in WorkVisual – Übersicht .....	165
11.2 Übersicht Feldbusse .....	166
11.3 Neue E/A-Konfiguration anlegen .....	166
11.4 Bestehende E/A-Konfiguration öffnen .....	166
11.5 Sunrise E/As anlegen .....	167
11.5.1 Fenster "E/A Signale anlegen" .....	168
11.5.2 E/A-Gruppe und Ein-/Ausgänge der Gruppe erstellen .....	169
11.5.3 E/A-Gruppe editieren .....	170
11.5.4 E/A-Gruppe löschen .....	170
11.5.5 Ein-/Ausgang einer Gruppe ändern .....	170
11.5.6 Ein-/Ausgang einer Gruppe löschen .....	170
11.5.7 E/A-Gruppe als Vorlage exportieren .....	170
11.5.8 E/A-Gruppe aus Vorlage importieren .....	171
11.6 Bus verschalten .....	172
11.6.1 Fenster <b>EA-Verschaltung</b> .....	172
11.6.2 Buttons im Fenster <b>EA-Verschaltung</b> .....	173
11.6.3 Sunrise E/As verschalten .....	174
11.7 E/A-Konfiguration in Sunrise-Projekt exportieren .....	174
<b>12 Externe Steuerung .....</b>	177
12.1 Externe Steuerung konfigurieren .....	177
12.1.1 Externe Steuerung Eingänge .....	178
12.1.2 Externe Steuerung Ausgänge .....	178
12.1.3 Signallaufpläne .....	179
12.1.4 Externe Steuerung in den Projekt-Eigenschaften konfigurieren .....	180
12.2 Roboter-Applikation als Default-Applikation markieren .....	181
12.3 Meldeausgänge für nicht extern gesteuertes Projekt festlegen .....	182
<b>13 Sicherheitskonfiguration .....</b>	183
13.1 Übersicht Sicherheitskonfiguration .....	183
13.2 Sicherheitskonzept .....	184
13.3 Permanent Safety Monitoring .....	186
13.4 Event-driven Safety Monitoring .....	188
13.5 Übersicht Atomic Monitoring Functions .....	189
13.5.1 Standard Atomic Monitoring Functions .....	189
13.5.2 Parametrierbare Atomic Monitoring Functions .....	190
13.5.3 Extended Atomic Monitoring Functions .....	192
13.5.4 Verfügbarkeit der AMFs in Abhängigkeit von der Kinematik .....	192
13.6 Sicherheitskonfiguration mit KUKA Sunrise.Workbench .....	193
13.6.1 Übersicht Sicherheitskonfiguration ändern und auf Steuerung aktivieren .....	194

13.6.2	Sicherheitskonfiguration öffnen .....	195
13.6.2.1	Auswertung der Sicherheitskonfiguration .....	195
13.6.2.2	Übersicht Bedienoberfläche Sicherheitskonfiguration .....	196
13.6.3	Sicherheitsfunktionen des PSM-Mechanismus konfigurieren .....	197
13.6.3.1	PSM-Tabelle <b>Anwender PSM</b> öffnen .....	197
13.6.3.2	Sicherheitsfunktionen für PSM-Mechanismus erstellen .....	199
13.6.3.3	Sicherheitsfunktion des PSM-Mechanismus löschen .....	199
13.6.3.4	Bestehende Sicherheitsfunktionen des PSM-Mechanismus bearbeiten .....	199
13.6.4	Sichere Zustände des ESM-Mechanismus konfigurieren .....	200
13.6.4.1	Neuen ESM-Zustand hinzufügen .....	201
13.6.4.2	Tabelle eines ESM-Zustands öffnen .....	201
13.6.4.3	ESM-Zustand löschen .....	203
13.6.4.4	Sicherheitsfunktion für ESM-Zustand erstellen .....	203
13.6.4.5	Sicherheitsfunktion eines ESM-Zustands löschen .....	203
13.6.4.6	Bestehende Sicherheitsfunktion eines ESM-Zustands bearbeiten .....	203
13.6.4.7	ESM-Mechanismus deaktivieren .....	204
13.6.4.8	Umschalten zwischen ESM-Zuständen .....	204
13.7	Sicherheitskonfiguration auf Robotersteuerung aktivieren .....	205
13.7.1	Sicherheitskonfiguration deaktivieren .....	205
13.7.2	Sicherheitskonfiguration wiederherstellen .....	206
13.7.3	Passwort für die Aktivierung der Sicherheitskonfiguration ändern .....	206
13.8	Verwendung und Parametrierung der Atomic Monitoring Functions .....	206
13.8.1	Auswertung der Sicherheitseinrichtungen am KUKA smartPAD .....	206
13.8.2	Auswertung der Betriebsart .....	207
13.8.3	Auswertung der Fahrfreigabe .....	207
13.8.4	Überwachung von sicheren Eingängen .....	208
13.8.5	Handführen mit Zustimmeinrichtung und Geschwindigkeitsüberwachung .....	209
13.8.5.1	Überwachung von Zustimmeinrichtungen an Handführgeräten .....	209
13.8.5.2	Überwachungen während des Handführens .....	211
13.8.5.3	Geschwindigkeitsüberwachung während des Handführens .....	211
13.8.6	Auswertung der Positionsreferenzierung .....	212
13.8.7	Auswertung der Momentenreferenzierung .....	212
13.8.8	Geschwindigkeitsüberwachungen .....	213
13.8.8.1	Achsspezifische Geschwindigkeitsüberwachung definieren .....	213
13.8.8.2	Kartesische Geschwindigkeitsüberwachung definieren .....	214
13.8.8.3	Richtungsabhängige Überwachung der kartesischen Geschwindigkeit .....	215
13.8.9	Überwachungsräume .....	220
13.8.9.1	Kartesische Arbeitsräume definieren .....	222
13.8.9.2	Kartesische Schutzzräume definieren .....	224
13.8.9.3	Achsspezifische Überwachungsräume definieren .....	227
13.8.10	Überwachung der Werkzeugorientierung .....	228
13.8.11	Stillstandsüberwachung (Sicherer Betriebshalt) .....	231
13.8.12	Einschaltverzögerung für Sicherheitsfunktionen .....	232
13.8.13	Überwachung von Kräften und Momenten .....	232
13.8.13.1	Achsmomentenüberwachung .....	233
13.8.13.2	Kollisionserkennung .....	234
13.8.13.3	TCP-Kraftüberwachung .....	235
13.9	Beispiel einer Sicherheitskonfiguration .....	236
13.9.1	Aufgabe .....	236
13.9.2	Anforderung .....	237

13.9.3	Lösungsvorschlag für Aufgabenstellung .....	237
13.10	Positions- und Momentenreferenzierung .....	240
13.10.1	Positionsreferenzierung .....	240
13.10.2	Momentenreferenzierung .....	241
13.10.3	Applikation für die Positions- und Momentenreferenzierung erstellen .....	243
13.11	Übersicht Sicherheitsabnahme .....	243
13.11.1	Checkliste Sicherheitsfunktionen Allgemein .....	244
13.11.2	Checklisten Sicherheitsgerichtes Werkzeug .....	247
13.11.2.1	Geometriedaten des sicherheitsgerichteten Werkzeugs .....	247
13.11.2.2	Lastdaten des sicherheitsgerichteten Werkzeugs .....	248
13.11.3	Checkliste Sicherheitsgerichtete Werkstücke .....	248
13.11.4	Checkliste Verwendete Zeilen in den PSM-Tabellen .....	249
13.11.5	Checklisten ESM-Zustände .....	250
13.11.5.1	Verwendete ESM-Zustände .....	250
13.11.5.2	Nicht verwendete ESM-Zustände .....	251
13.11.6	Checklisten Verwendete AMFs .....	252
13.11.6.1	AMF NOT-HALT smartPAD .....	252
13.11.6.2	AMF Zustimmung smartPAD inaktiv .....	252
13.11.6.3	AMF Zustimmung Panik smartPAD aktiv .....	252
13.11.6.4	AMF Zustimmung Handführgerät inaktiv .....	252
13.11.6.5	AMF Zustimmung Handführgerät aktiv .....	253
13.11.6.6	AMF Betriebsart Test .....	253
13.11.6.7	AMF Betriebsart Automatik .....	253
13.11.6.8	AMF Betriebsart mit reduzierter Geschwindigkeit .....	253
13.11.6.9	AMF Betriebsart mit hoher Geschwindigkeit .....	253
13.11.6.10	AMF Fahrfreigabe .....	253
13.11.6.11	AMF Eingangssignal .....	253
13.11.6.12	AMF Stillstandsüberwachung aller Achsen .....	254
13.11.6.13	AMF Achsmomentenüberwachung .....	254
13.11.6.14	AMF Achsgeschwindigkeitsüberwachung .....	254
13.11.6.15	AMF Positionsreferenzierung .....	254
13.11.6.16	AMF Momentenreferenzierung .....	255
13.11.6.17	AMF Achsbereichsüberwachung .....	255
13.11.6.18	AMF Kartesische Geschwindigkeitsüberwachung .....	255
13.11.6.19	AMF Kartesische Arbeitsraumüberwachung / Kartesische Schutzraumüberwachung .....	255
13.11.6.20	AMF Kollisionserkennung .....	256
13.11.6.21	AMF TCP-Kraftüberwachung .....	256
13.11.6.22	AMF Zeitverzögerung .....	257
13.11.6.23	AMF Werkzeugorientierung .....	257
13.11.6.24	AMF Werkzeugbezogene Geschwindigkeitskomponente .....	258
13.11.7	Bericht zur Sicherheitskonfiguration erstellen .....	259
<b>14</b>	<b>Grundlagen der Bewegungsprogrammierung .....</b>	<b>261</b>
14.1	Bewegungsarten Übersicht .....	261
14.2	Bewegungsart PTP .....	261
14.3	Bewegungsart LIN .....	262
14.4	Bewegungsart CIRC .....	262
14.5	Bewegungsart SPL .....	263
14.6	Bewegungsart Spline .....	263
14.6.1	Geschwindigkeitsprofil bei Spline-Bewegungen .....	264

14.6.2 Änderungen an Spline-Blöcken .....	266
14.6.3 LIN-SPL-LIN-Übergang .....	268
14.7 Bewegungsart Handführen .....	269
14.8 Überschleifen .....	270
14.9 Orientierungsführung LIN, CIRC, SPL .....	272
14.9.1 CIRC – Bezugssystem der Orientierungsführung .....	274
14.9.2 CIRC – Kombinationen von Bezugssystem und Typ der Orientierungsführung .....	275
14.10 Redundanzinformationen .....	276
14.10.1 Redundanzwinkel .....	277
14.10.2 Status .....	277
14.10.3 Turn .....	278
14.11 Singularitäten .....	279
14.11.1 Kinematische Singularitäten .....	279
14.11.2 Systembedingte Singularitäten .....	281
<b>15 Programmierung .....</b>	<b>283</b>
15.1 Java-Editor .....	283
15.1.1 Roboter-Applikation im Java-Editor öffnen .....	283
15.1.2 Aufbau einer Roboter-Applikation .....	283
15.1.3 Bearbeitungsfunktionen .....	284
15.1.3.1 Variablen umbenennen .....	284
15.1.3.2 Auto-Vervollständigen .....	284
15.1.3.3 Schablonen – Schnelleingabe für Java-Anweisungen .....	285
15.1.3.4 Benutzerspezifische Schablonen erstellen .....	286
15.1.3.5 Methoden extrahieren .....	286
15.1.4 Javadoc-Informationen anzeigen .....	287
15.1.4.1 Aufbau des Javadoc-Browsers .....	289
15.2 Zeichen und Schriftarten .....	292
15.3 Datentypen .....	292
15.4 Variablen .....	293
15.5 Netzwerk-Kommunikation über UDP und TCP/IP .....	293
15.6 Versionsinformationen RoboticsAPI .....	294
15.6.1 RoboticsAPI-Version anzeigen .....	294
15.6.2 Aufbau RoboticsAPI-Versionsnummer .....	294
15.7 Bewegungsprogrammierung: PTP, LIN, CIRC .....	294
15.7.1 Aufbau eines Bewegungsbefehls (move/moveAsync) .....	294
15.7.2 PTP .....	295
15.7.3 LIN .....	296
15.7.4 CIRC .....	297
15.7.5 LIN REL .....	297
15.7.6 MotionBatch .....	298
15.8 Bewegungsprogrammierung: Spline .....	299
15.8.1 Programmertipps für Spline-Bewegungen .....	299
15.8.2 CP-Spline-Block anlegen .....	300
15.8.3 JP-Spline-Block anlegen .....	301
15.8.4 Spline in Bewegungsanweisung verwenden .....	302
15.9 Bewegungsparameter .....	302
15.9.1 Achsspezifische Bewegungsparameter programmieren .....	304
15.10 Handführen programmieren .....	304

15.10.1	Achsspezifische Grenzen für das Handführen .....	307
15.11	Werkzeuge und Werkstücke im Programm verwenden .....	309
15.11.1	Werkzeuge und Werkstücke deklarieren .....	309
15.11.2	Werkzeuge und Werkstücke initialisieren .....	310
15.11.3	Werkzeuge und Werkstücke mit Roboter verbinden .....	310
15.11.3.1	Werkzeug mit Roboterflansch verbinden .....	311
15.11.3.2	Werkstück mit anderen Objekten verbinden .....	311
15.11.3.3	Verbindungen lösen .....	313
15.11.4	Werkzeuge und Werkstücke bewegen .....	313
15.11.5	Eigene Objektklassen definieren .....	314
15.11.6	Kommandieren von Lastwechseln an Sicherheitssteuerung .....	317
15.12	Ein-/Ausgänge .....	319
15.12.1	Datenfeld für E/A-Gruppe anlegen .....	320
15.12.2	Datenfeld für E/A-Gruppe initialisieren .....	321
15.12.3	Ein-/Ausgänge lesen .....	321
15.12.4	Ausgänge setzen .....	322
15.13	Achsmomente abfragen .....	322
15.14	Auslesen kartesischer Kräfte und Momente .....	324
15.14.1	Berechnete Kraft-Momenten-Daten abfragen .....	324
15.14.2	Einzelne Kraft-Momenten-Werte abfragen .....	325
15.14.3	Zuverlässigkeit der berechneten Kraft-Momenten-Werte prüfen .....	326
15.14.4	Einzelwerte von einem Vektor abfragen .....	327
15.15	Abfrage der Roboterposition .....	328
15.15.1	Achsspezifische Ist- oder Sollposition abfragen .....	328
15.15.2	Kartesische Ist- oder Sollposition abfragen .....	329
15.15.3	Kartesische Soll-Ist-Differenz abfragen .....	330
15.16	HOME-Position .....	331
15.16.1	HOME-Position ändern .....	332
15.17	Systemzustände abfragen .....	332
15.17.1	HOME-Position abfragen .....	333
15.17.2	Justagezustand abfragen .....	333
15.17.3	Fahrbereitschaft abfragen .....	334
15.17.3.1	Auf Änderung des Fahrbereitschaft-Signals reagieren .....	334
15.17.4	Roboteraktivität abfragen .....	335
15.17.5	Sicherheitssignale abfragen und auswerten .....	335
15.17.5.1	Zustand der Sicherheitssignale abfragen .....	336
15.17.5.2	Auf Zustandsänderung von Sicherheitssignalen reagieren .....	337
15.18	Programmablaufart ändern und abfragen .....	338
15.19	Override ändern und abfragen .....	339
15.19.1	Auf Override-Änderung reagieren .....	340
15.20	Bedingungen .....	341
15.20.1	Bedingungen in der RoboticsAPI .....	341
15.20.2	Komplexe Bedingungen .....	342
15.20.3	Achsmomenten-Bedingung .....	343
15.20.4	Kraftbedingung .....	344
15.20.4.1	Bedingung für kartesische Kraft aus allen Richtungen .....	345
15.20.4.2	Bedingung für Normalkraft .....	346
15.20.4.3	Bedingung für Scherkraft .....	348
15.20.5	Kraft-Komponenten-Bedingung .....	349

15.20.6 Bedingung für kartesisches Moment .....	351
15.20.6.1 Bedingung für kartesisches Moment aus allen Richtungen .....	352
15.20.6.2 Bedingung für Drehmoment .....	353
15.20.6.3 Bedingung für Kippmoment .....	354
15.20.7 Moment-Komponenten-Bedingung .....	355
15.20.8 Bahnbezogene Bedingung .....	356
15.20.9 Bedingung für boolesche Signale .....	359
15.20.10 Bedingung für Wertebereich eines Signals .....	359
15.21 Abbruchbedingungen für Bewegungsbefehle .....	360
15.21.1 Abbruchbedingung festlegen .....	360
15.21.2 Abbruchbedingungen auswerten .....	361
15.21.2.1 Abbruchbedingung abfragen .....	362
15.21.2.2 Roboterposition zum Abbruchzeitpunkt abfragen .....	363
15.21.2.3 Abgebrochene Bewegung abfragen (Spline-Block, MotionBatch) .....	363
15.22 Bahnbezogene Schaltaktionen (Trigger) .....	364
15.22.1 Trigger programmieren .....	364
15.22.2 Bahnbezogene Schaltaktion programmieren .....	365
15.22.3 Trigger-Informationen auswerten .....	366
15.23 Überwachen von Prozessen (Monitoring) .....	368
15.23.1 Listener für das Überwachen von Bedingungen .....	368
15.23.2 Listener-Objekt für Überwachung einer Bedingung erzeugen .....	369
15.23.3 Listener für Benachrichtigung bei Zustandsänderung registrieren .....	370
15.23.4 Benachrichtigungsdienst für Listener aktivieren oder deaktivieren .....	372
15.23.5 Monitoring Programmierbeispiel .....	372
15.24 Blockierendes Warten auf Bedingung .....	372
15.25 Daten aufzeichnen und auswerten .....	374
15.25.1 Objekt für Datenaufzeichnung erstellen .....	374
15.25.2 Daten für Aufzeichnung festlegen .....	375
15.25.3 Datenaufzeichnung starten .....	377
15.25.4 Datenaufzeichnung beenden .....	378
15.25.5 Zustände vom DataRecorder-Objekt abfragen .....	379
15.25.6 Beispielprogramm für Datenaufzeichnung .....	379
15.26 Benutzertasten definieren .....	380
15.26.1 Benutzertasten-Leiste erstellen .....	381
15.26.2 Benutzertasten zur Leiste hinzufügen .....	382
15.26.3 Funktion einer Benutzertaste festlegen .....	384
15.26.4 Beschriftung und grafische Gestaltung der Benutzertasten-Leiste .....	386
15.26.4.1 Textelement zuweisen .....	386
15.26.4.2 LED-Icon zuweisen .....	388
15.26.5 Sicherheitskritische Benutzertasten kennzeichnen .....	389
15.26.6 Benutzertasten-Leiste veröffentlichen .....	390
15.27 Meldungsprogrammierung .....	390
15.27.1 Benutzermeldungen programmieren .....	390
15.27.2 Benutzerdialoge programmieren .....	391
15.28 Programmablaufkontrolle .....	393
15.28.1 Applikation pausieren .....	393
15.28.2 Bewegungsausführung pausieren .....	393
15.28.3 for-Schleife .....	393
15.28.4 while-Schleife .....	394

15.28.5 do-while-Schleife .....	395
15.28.6 if-else-Verzweigung .....	396
15.28.7 switch-Verzweigung .....	398
15.28.8 Beispiele für verschachtelte Schleifen .....	399
15.29 Pausierte Applikation im Automatikbetrieb fortsetzen (Recovery) .....	400
15.30 Fehlerbehandlung .....	402
15.30.1 Behandlung fehlgeschlagener Bewegungsbefehle .....	402
15.30.2 Behandlung fehlgeschlagener synchroner Bewegungsbefehle .....	402
15.30.3 Behandlung fehlgeschlagener asynchroner Bewegungsbefehle .....	404
<b>16 Hintergrund-Tasks .....</b>	<b>409</b>
16.1 Verwendung von Hintergrund-Tasks .....	409
16.2 Zyklischer Hintergrund-Task .....	410
16.3 Nicht-zyklischer Hintergrund-Task .....	413
16.4 Datenaustausch zwischen Tasks .....	414
16.4.1 Task-Funktionalitäten deklarieren .....	415
16.4.2 Task-Funktionalitäten implementieren .....	416
16.4.3 Anbietenden Task erstellen .....	417
16.4.4 Task-Funktionalitäten nutzen .....	420
<b>17 Programmierung mit nachgiebigem Roboter .....</b>	<b>425</b>
17.1 Sensorik und Regelung .....	425
17.2 Verfügbare Regler – Übersicht .....	425
17.3 Regler in Roboter-Applikation verwenden .....	426
17.3.1 Regler-Objekt anlegen .....	426
17.3.2 Regler-Parameter festlegen .....	426
17.3.3 Regler-Objekt als Bewegungsparameter übergeben .....	426
17.4 Positionsregler .....	427
17.5 Kartesischer Impedanzregler .....	427
17.5.1 Berechnung der Kräfte nach dem Federgesetz .....	428
17.5.2 Parametrierung des kartesischen Impedanzreglers .....	430
17.5.2.1 Darstellung der kartesischen Freiheitsgrade .....	430
17.5.2.2 Regler-Parameter für einzelne Freiheitsgrade festlegen .....	431
17.5.2.3 Freiheitsgradspezifische Regler-Parameter .....	431
17.5.2.4 Freiheitsgradunabhängige Regler-Parameter .....	432
17.6 Kartesischer Impedanzregler mit aufgeschalteter Kraftschwingung .....	435
17.6.1 Aufschalten einer einfachen Kraftschwingung .....	435
17.6.2 Aufschalten überlagerter Kraftschwingungen (Lissajous-Figuren) .....	436
17.6.3 Parametrierung des Impedanzreglers mit aufgeschalteter Kraftschwingung .....	437
17.6.3.1 Freiheitsgradspezifische Regler-Parameter .....	438
17.6.3.2 Freiheitsgradunabhängige Regler-Parameter .....	440
17.7 Statische Methoden für Impedanzregler mit überlagerter Kraftschwingung .....	442
17.7.1 Konstante Kraft aufschalten .....	443
17.7.2 Einfache Kraftschwingung aufschalten .....	443
17.7.3 Lissajous-Schwingung aufschalten .....	444
17.7.4 Spiralförmige Kraftschwingung aufschalten .....	445
17.8 Achsspezifischer Impedanzregler .....	447
17.8.1 Parametrierung des achsspezifischen Impedanzreglers .....	447
17.8.2 Methoden des achsspezifischen Impedanzreglers .....	448

17.9 Halten der Position in Regelung .....	449
<b>18 Diagnose .....</b>	<b>451</b>
18.1 Feldbus-Diagnose .....	451
18.1.1 Allgemeine Feldbusfehler anzeigen .....	451
18.1.2 Fehlerzustand von E/As und E/A-Gruppen anzeigen .....	451
18.2 Protokoll anzeigen .....	451
18.2.1 Ansicht <b>Protokoll</b> .....	452
18.2.2 Protokolleinträge filtern .....	453
18.3 Anzeige Fehlermeldungen (Applikationssicht) .....	454
18.4 Diagnoseinformationen sammeln für Fehleranalyse bei KUKA .....	457
18.4.1 Diagnosepaket über smartHMI erstellen .....	457
18.4.2 Diagnosepaket über smartPAD erstellen .....	458
18.4.3 Diagnosepaket über Sunrise.Workbench erstellen .....	458
18.4.4 Bestehende Diagnosepakete von Robotersteuerung laden .....	458
<b>19 KUKA Service .....</b>	<b>461</b>
19.1 Support-Anfrage .....	461
19.2 KUKA Customer Support .....	461
<b>Index .....</b>	<b>469</b>

# 1 Einleitung

## 1.1 Zielgruppe

Diese Dokumentation richtet sich an Benutzer mit folgenden Kenntnissen:

- Fortgeschrittene Systemkenntnisse der Robotersteuerung
- Fortgeschrittene Java-Programmierkenntnisse



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

## 1.2 Dokumentation des Industrieroboters

Die Dokumentation zum Industrieroboter besteht aus folgenden Teilen:

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

Jede Anleitung ist ein eigenes Dokument.

## 1.3 Darstellung von Hinweisen

### Sicherheit

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



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



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



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



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



Diese Hinweise enthalten Verweise auf sicherheitsrelevante Informationen oder allgemeine Sicherheitsmaßnahmen.

Diese Hinweise beziehen sich nicht auf einzelne Gefahren oder einzelne Vorsichtsmaßnahmen.

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



Mit diesem Hinweis gekennzeichnete Vorgehensweisen **müssen** genau eingehalten werden.

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



Hinweis zur Arbeitserleichterung oder Verweis auf weiterführende Informationen.

## 1.4 Marken

**Java** ist eine Marke von Sun Microsystems (Oracle Corporation).

**Windows** ist eine Marke der Microsoft Corporation.



ist eine Marke der Beckhoff Automation GmbH.

## 1.5 Verwendete Begriffe

Begriff	Beschreibung
AMF	Atomic Monitoring Function Kleinste Einheit einer Überwachungsfunktion (deutsch: Elementarüberwachung)
API	Application Programming Interface Schnittstelle zur Programmierung von Applikationen
CIB-SR	Cabinet Interface Board Small Robot
ESM	Event-Driven Safety Monitoring Sicherheitsüberwachungen, die über definierte Ereignisse aktiviert werden
Exception	Ausnahme oder Ausnahmesituation (engl. exception) Eine Ausnahme bezeichnet ein Verfahren, um Informationen über bestimmte Programmzustände, meist Fehlerzustände, an andere Programmebenen zur Weiterbehandlung weiterzugeben.
Frame	Ein Frame ist ein 3-dimensionales Koordinatensystem, das durch seine Position und Orientierung bezüglich eines Referenzsystems beschrieben wird. Mithilfe von Frames lassen sich Punkte im Raum einfach definieren. Frames sind oft hierarchisch in einer Baumstruktur angeordnet.
FSoE	Fail Safe over EtherCAT FSoE ist ein Protokoll zur Übertragung von sicherheitsrelevanten Daten über EtherCAT unter Verwendung eines FSoE Masters und FSoE Slaves.
Javadoc	Javadoc ist eine aus speziellen Java-Kommentaren generierte Dokumentation.
JRE	Java Runtime Environment Laufzeitumgebung der Programmiersprache Java
KLI	KUKA Line Interface Ethernet-Schnittstelle der Robotersteuerung (nicht-echtzeitfähig) für die externe Kommunikation
KMP	KUKA Mobile Platform Bezeichnung für mobile Plattformen von KUKA

<b>Begriff</b>	<b>Beschreibung</b>
KRF	<p><b>Kontrollierte Roboterfahrt</b></p> <p>KRF ist eine Betriebsart, die zur Verfügung steht, wenn der Industrieroboter von der Sicherheitssteuerung aufgrund einer der folgenden Ursachen gestoppt wird:</p> <ul style="list-style-type: none"> <li>■ Industrieroboter verletzt einen sicher überwachten Raum.</li> <li>■ Orientierung des sicherheitsgerichteten Werkzeugs liegt außerhalb des sicher überwachten Bereichs.</li> <li>■ Industrieroboter verletzt eine sicher überwachte Kraft- oder Momentengrenze.</li> <li>■ Ein Positionssensor ist bei aktiver kartesischer Geschwindigkeitsüberwachung nicht justiert.</li> </ul> <p>In der Betriebsart KRF kann der Roboter manuell verfahren und in eine Position zurückgebracht werden, in der die stoppauslösende Überwachung nicht mehr verletzt ist.</p>
KUKA RoboticsAPI	<p>Java-Programmierschnittstelle für KUKA Roboter</p> <p>RoboticsAPI ist eine objektorientierte Java-Schnittstelle zur Steuerung von Robotern und Peripheriegeräten.</p>
KUKA smartHMI	<p>KUKA smart Human-Machine Interface</p> <p>Bezeichnung der Bedienoberfläche der Robotersteuerung</p>
KUKA smartPAD	<p>Das smartPAD ist das Bedienhandgerät für die Roboterzelle (Station). Es bietet alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung der Station benötigt werden.</p>
KUKA Sunrise Cabinet	<p>Steuerungshardware zum Betrieb des Industrieroboters LBR iiwa</p>
KUKA Sunrise.OS	<p>KUKA Sunrise.Operating System</p> <p>System Software für Industrieroboter, die mit der Robotersteuerung KUKA Sunrise Cabinet betrieben werden</p>
MRK	<p>Mensch-Roboter Kooperation</p>
PROFINET	<p>PROFINET ist ein auf Ethernet basierender Feldbus.</p>
PROFIsafe	<p>PROFIsafe ist eine auf PROFINET basierende Sicherheitsschnittstelle zur Anbindung einer Sicherheits-SPS an die Robotersteuerung. (SPS = Master, Robotersteuerung = Slave)</p>
PSM	<p>Permanent Safety Monitoring</p> <p>Sicherheitsüberwachungen, die permanent aktiv sind</p>
SPS	<p>Speicherprogrammierbare Steuerung</p>
TCP	<p>Tool Center Point</p> <p>Der TCP ist der Arbeitspunkt eines Werkzeugs. Es können mehrere Arbeitspunkte für ein Werkzeug definiert werden.</p>



## 2 Produktbeschreibung

### 2.1 Übersicht des Robotersystems

Ein Robotersystem ([>>> Abb. 2-1](#)) umfasst alle Baugruppen eines Industrieroboters wie Manipulator (Robotermechanik und Elektro-Installation), Steuerung, Verbindungsleitungen, Werkzeug und Ausrüstungsteile.

Der Industrieroboter besteht aus folgenden Komponenten:

- Manipulator
- Robotersteuerung KUKA Sunrise Cabinet
- Bedienhandgerät KUKA smartPAD
- Verbindungsleitungen
- Software
- Optionen, Zubehör



**Abb. 2-1: Übersicht Robotersystem**

- 1 Verbindungsleitung zum smartPAD
- 2 Bedienhandgerät KUKA smartPAD
- 3 Manipulator
- 4 Verbindungsleitung zur Robotersteuerung KUKA Sunrise Cabinet
- 5 Robotersteuerung KUKA Sunrise Cabinet

## 2.2 Übersicht der Software-Komponenten

Folgende Software-Komponenten werden eingesetzt:

- KUKA Sunrise.OS 1.7
- KUKA Sunrise.Workbench 1.7
- WorkVisual 3.0

## 2.3 Übersicht KUKA Sunrise.OS

**Beschreibung** Mit KUKA Sunrise.OS wird eine strikte Trennung von Bedienung und Programmierung des Robotersystems realisiert.

- Mit KUKA Sunrise.Workbench werden Roboter-Applikationen programmiert.
- Über das Bedienhandgerät KUKA smartPAD wird eine Roboterzelle (Station) bedient.
- Eine Station besteht aus einer Robotersteuerung, einem Manipulator und weiteren Geräten.
- Eine Station kann mehrere Applikationen (Aufgaben) ausführen.

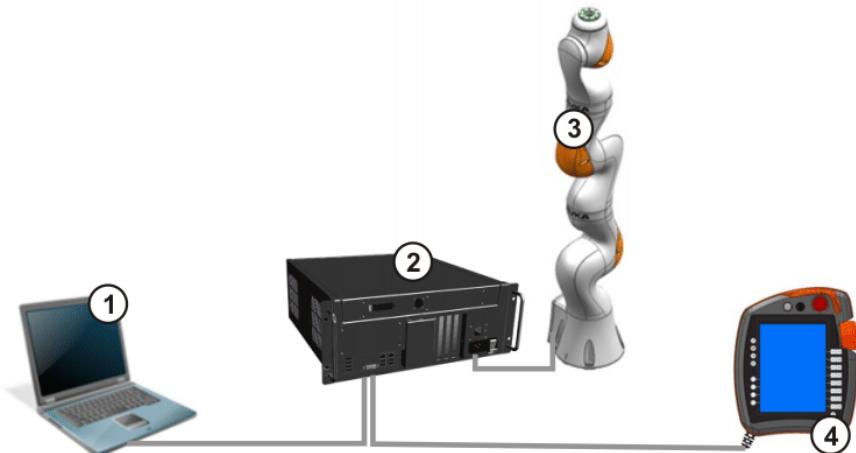


Abb. 2-2: Trennung von Bedienung und Programmierung

- 1 Entwicklungsrechner mit KUKA Sunrise.Workbench (Anschluss über das KLI der Robotersteuerung)
- 2 Robotersteuerung KUKA Sunrise Cabinet
- 3 Manipulator
- 4 Bedienhandgerät KUKA smartPAD



Der Entwicklungsrechner ist nicht im Lieferumfang des Industrieroboters enthalten.

### Aufgabenteilung

KUKA Sunrise.Workbench ist das Werkzeug zur Inbetriebnahme einer Station und zur Entwicklung von Roboter-Applikationen. Zur Buskonfiguration und Busverschaltung wird WorkVisual verwendet.

Das smartPAD wird in der Inbetriebnahme-Phase nur für Aufgaben benötigt, die aus praktischen oder sicherheitstechnischen Gründen nicht mit KUKA Sunrise.Workbench ausgeführt werden können, z. B. die Justage, das Vermessen und das Teachen von Punkten.

Nach Inbetriebnahme und Applikationsentwicklung kann der Bediener über das smartPAD einfache Instandhaltungs- und Bedienaufgaben ausführen. Die

Stations- und Sicherheitskonfiguration sowie die Programmierung kann der Bediener nicht ändern.

## Übersicht

Aufgabe	WorkVisual	Workbench	smartPAD
Stationskonfiguration		✓	
Software-Installation		✓	
Buskonfiguration/-diagnose	✓		
Busverschaltung	✓		
Sicherheitseinstellungen konfigurieren		✓	
Sicherheitskonfiguration aktivieren			✓
Programmierung		✓	
Debugging		✓	
Laufzeitdaten verwalten/editieren		✓	
Frames teachen			✓
Betriebsartenwahl			✓
Handverfahren			✓
Justage			✓
Vermessen			✓
Lastdatenermittlung			✓
Ausgänge setzen/abfragen			✓
Eingänge abfragen			✓
Roboter-Applikationen starten/stoppen			✓
Diagnosepaket erstellen		✓	✓

## 2.4 Übersicht KUKA Sunrise.Workbench

KUKA Sunrise.Workbench ist die Entwicklungsumgebung für die Roboterzelle (Station). Sie verfügt über folgende Funktionalitäten zur Inbetriebnahme und Applikationsentwicklung:

- |                                |   |
|--------------------------------|---|
| <b>Inbetriebnahme</b>          | <ul style="list-style-type: none"> <li>■ System Software installieren</li> <li>■ Roboterzelle (Station) konfigurieren</li> <li>■ Sicherheitskonfiguration bearbeiten</li> <li>■ E/A-Konfiguration anlegen</li> <li>■ Projekt auf die Robotersteuerung übertragen</li> </ul> |
| <b>Applikationsentwicklung</b> | <ul style="list-style-type: none"> <li>■ Roboter-Applikationen in Java programmieren</li> <li>■ Projekte und Programme verwalten</li> <li>■ Laufzeitdaten editieren und verwalten</li> <li>■ Projektsynchronisation</li> </ul>  |

## 2.5 Bestimmungsgemäße Verwendung der System Software

### Verwendung

Die System Software ist ausschließlich zum Betreiben von KUKA Achsen in der Industrie in Verbindung mit KUKA Sunrise Cabinet bestimmt. KUKA Achsen sind z. B. Industrieroboter oder mobile Plattformen.

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

### Fehlanwendung

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

Zu den Fehlanwendungen zählen z. B.:

- Betreiben von Achsen, die keine KUKA Achsen sind
- Betreiben der System Software unter anderen als den spezifizierten Systemvoraussetzungen
- Einsatz eines anderen Debuggers als den von Sunrise.Workbench
- Einsatz außerhalb industrieller Anwendungen, bei denen spezifische Produktanforderungen/Normen existieren (z. B. Medizin)

## 3 Sicherheit

### 3.1 Rechtliche Rahmenbedingungen

#### 3.1.1 Haftungshinweis

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

Komponenten des Industrieroboters:

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

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

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

#### Sicherheitsinformation

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

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

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

#### 3.1.2 Bestimmungsgemäße Verwendung des Industrieroboters

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

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

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

Der Betreiber ist für die Durchführung einer Risikoanalyse verantwortlich. Aus dieser ergeben sich die erforderlichen zusätzlichen Schutzeinrichtungen, für deren Installation der Betreiber ebenfalls verantwortlich ist.

<b>Fehlanwendung</b>	Alle von der bestimmungsgemäßen Verwendung abweichenden Anwendungen gelten als Fehlanwendung und sind unzulässig. Dazu zählen z. B.:
	<ul style="list-style-type: none"><li>■ Transport von Menschen und Tieren</li><li>■ Benutzung als Aufstiegshilfen</li><li>■ Einsatz außerhalb der spezifizierten Betriebsgrenzen</li><li>■ Einsatz in explosionsgefährdeter Umgebung</li><li>■ Einsatz ohne erforderliche zusätzliche Schutzeinrichtungen</li><li>■ Einsatz im Freien</li><li>■ Einsatz unter Tage</li></ul>

### 3.1.3 EG-Konformitätserklärung und Einbauerklärung

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

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

<b>Konformitätserklärung</b>	Der Systemintegrator muss eine Konformitätserklärung gemäß der Maschinenrichtlinie für die gesamte Anlage erstellen. Die Konformitätserklärung ist Grundlage für die CE-Kennzeichnung der Anlage. Der Industrieroboter darf nur nach landesspezifischen Gesetzen, Vorschriften und Normen betrieben werden.
<b>Einbauerklärung</b>	Die Robotersteuerung besitzt eine CE-Zertifizierung gemäß der EMV-Richtlinie und der Niederspannungsrichtlinie.

<b>Einbauerklärung</b>	Der Industrieroboter als unvollständige Maschine wird mit einer Einbauerklärung nach Anhang II B der Maschinenrichtlinie 2006/42/EG ausgeliefert. Bestandteile der Einbauerklärung sind eine Liste mit den eingehaltenen grundlegenden Anforderungen nach Anhang I und die Montageanleitung.  Mit der Einbauerklärung wird erklärt, dass die Inbetriebnahme der unvollständigen Maschine solange unzulässig bleibt, bis die unvollständige Maschine in eine Maschine eingebaut, oder mit anderen Teilen zu einer Maschine zusammengebaut wurde, diese den Bestimmungen der EG-Maschinenrichtlinie entspricht und die EG-Konformitätserklärung gemäß Anhang II A vorliegt.
------------------------	---

## 3.2 Sicherheitsfunktionen

Sicherheitsfunktionen werden nach den Sicherheitsanforderungen unterschieden, die sie erfüllen:

- Sicherheitsgerichtete Funktionen zum Schutz von Personen  
Die sicherheitsgerichteten Funktionen des Industrieroboters erfüllen folgende Sicherheitsanforderungen:
  - **Kategorie 3** und **Performance Level d** nach EN ISO 13849-1:2008

■ **SIL 2 nach EN 62061**

Die Anforderungen werden jedoch nur unter folgender Voraussetzung erfüllt:

- Alle sicherheitsrelevanten mechanischen und elektromechanischen Komponenten des Industrieroboters werden bei der Inbetriebnahme und mindestens alle 12 Monate auf Funktionsfähigkeit geprüft, sofern nach Gefährdungsbeurteilung am Arbeitsplatz nicht abweichend bestimmt. Dazu gehören:
  - NOT-HALT-Einrichtung am smartPAD
  - Zustimmeinrichtung am smartPAD
  - Zustimmeinrichtung am Medien-Flansch Touch (wenn vorhanden)
  - Schlüsselschalter am smartPAD
  - Sichere Ausgänge der diskreten Sicherheitsschnittstelle
- Nicht sicherheitsgerichtete Funktionen zum Schutz von Maschinen  
Die nicht sicherheitsgerichteten Funktionen des Industrieroboters erfüllen keine spezifischen Sicherheitsanforderungen.



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



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

### 3.2.1 Verwendete Begriffe

Begriff	Beschreibung
Achsbereich	Bereich jeder Achse in Grad oder Millimeter, in dem sie sich bewegen darf. Der Achsbereich muss für jede Achse definiert werden.
Anhalteweg	Anhalteweg = Reaktionsweg + Bremsweg  Der Anhalteweg ist Teil des Gefahrenbereichs.
Arbeitsbereich	Im Arbeitsbereich darf sich der Manipulator bewegen. Der Arbeitsbereich ergibt sich aus den einzelnen Achsbereichen.
Automatik (AUT)	Betriebsart für den Programmbetrieb. Der Manipulator wird mit der programmierten Geschwindigkeit verfahren.
Betreiber (Benutzer)	Der Betreiber eines Industrieroboters kann der Unternehmer, Arbeitgeber oder die delegierte Person sein, die für die Benutzung des Industrieroboters verantwortlich ist.
Gefahrenbereich	Der Gefahrenbereich beinhaltet den Arbeitsbereich und die Anhaltewege.
Gebrauchsdauer	Die Gebrauchsdauer eines sicherheitsrelevanten Bauteils beginnt ab dem Zeitpunkt der Lieferung des Teils an den Kunden.  Die Gebrauchsdauer wird nicht beeinflusst davon, ob das Teil in einer Robotersteuerung oder anderweitig betrieben wird oder nicht, da sicherheitsrelevante Bauteile auch während der Lagerung altern.

Begriff	Beschreibung
KRF	<p><b>Kontrollierte Roboterfahrt</b></p> <p>KRF ist eine Betriebsart, die zur Verfügung steht, wenn der Industrieroboter von der Sicherheitssteuerung aufgrund einer der folgenden Ursachen gestoppt wird:</p> <ul style="list-style-type: none"> <li>■ Industrieroboter verletzt einen sicher überwachten Raum.</li> <li>■ Orientierung des sicherheitsgerichteten Werkzeugs liegt außerhalb des sicher überwachten Bereichs.</li> <li>■ Industrieroboter verletzt eine sicher überwachte Kraft- oder Momen-tengrenze.</li> <li>■ Ein Positionssensor ist bei aktiver kartesischer Geschwindigkeits-überwachung nicht justiert.</li> </ul> <p>In der Betriebsart KRF kann der Roboter manuell verfahren und in eine Position zurückgebracht werden, in der die stoppauslösende Überwa-chung nicht mehr verletzt ist.</p>
KUKA smartPAD	Das smartPAD ist das Bedienhandgerät für die Roboterzelle (Station). Das smartPAD hat alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung benötigt werden.
Manipulator	Die Robotermechanik und die zugehörige Elektroinstallation
Schutzbereich	Im Schutzbereich darf sich der Manipulator nicht bewegen. Der Schutz-bereich ist der Bereich außerhalb des Gefahrenbereichs.
Sicherheitshalt	<p>Der Sicherheitshalt wird von der Sicherheitssteuerung ausgelöst, unter-bricht den Arbeitsablauf und bewirkt den Stillstand aller Roboterbewe-gungen. Die Programmdaten bleiben bei einem Sicherheitshalt erhalten und das Programm kann an der unterbrochenen Stelle fortgesetzt wer-den.</p> <p>Der Sicherheitshalt kann als Stopp-Kategorie 0, Stopp-Kategorie 1 oder Stopp-Kategorie 1 (bahntreu) ausgeführt werden.</p> <p><b>Hinweis:</b> Ein Sicherheitshalt der Stopp-Kategorie 0 wird im Dokument als Sicherheitshalt 0, ein Sicherheitshalt der Stopp-Kategorie 1 als Sicherheitshalt 1 und ein Sicherheitshalt der Stopp-Kategorie 1 (bahntreu) als Sicherheitshalt 1 (bahntreu) bezeichnet.</p>
Stopp-Kategorie 0	Die Antriebe werden sofort abgeschaltet und die Bremsen fallen ein.
Stopp-Kategorie 1	<p>Der Manipulator bremst nicht bahntreu. Der Manipulator wird mit den Antrieben in den Stillstand überführt. Sobald eine Achse stillsteht, wird der Antrieb abgeschaltet und die Bremse fällt ein.</p> <p>Der Bremsvorgang wird von der roboterinternen Antriebselektronik sicherheitsgerichtet überwacht. Im Fehlerfall wird die Stopp-Kategorie 0 ausgeführt.</p>
Stopp-Kategorie 1 (bahntreu)	<p>Der Manipulator bremst bahntreu. Im Stillstand werden die Antriebe abgeschaltet und die Bremsen fallen ein.</p> <p>Wird die Stopp-Kategorie 1 (bahntreu) von der Sicherheitssteuerung ausgelöst, wird der Bremsvorgang von der Sicherheitssteuerung über-wacht. Nach spätestens 1 s werden die Antriebe abgeschaltet und die Bremsen fallen ein. Im Fehlerfall wird die Stopp-Kategorie 0 ausgeführt.</p>
Systemintegrator (Anlagenintegrator)	Systemintegratoren sind Personen, die den Industrieroboter sicherheits-gerecht in eine Anlage integrieren und inbetriebnehmen.
T1	<p>Test-Betriebsart Manuell Reduzierte Geschwindigkeit (&lt;= 250 mm/s)</p> <p><b>Hinweis:</b> Beim Handführen in T1 ist die Geschwindigkeit nicht reduziert. Bei einer mobilen Plattform gilt die maximale Geschwindigkeit von 250 mm/s nicht.</p>
T2	Test-Betriebsart Manuell Hohe Geschwindigkeit (> 250 mm/s zulässig)

### 3.2.2 Personal

Folgende Personen oder Personengruppen werden für den Industrieroboter definiert:

- Betreiber
- Personal



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

#### Betreiber

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

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

#### Personal

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

Zum Personal zählen:

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



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

#### Systemintegrator

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

Der Systemintegrator ist für folgende Aufgaben verantwortlich:

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

#### Anwender

Der Anwender muss folgende Voraussetzungen erfüllen:

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



Arbeiten an der Elektrik und Mechanik des Manipulators dürfen nur von der KUKA Roboter GmbH vorgenommen werden.

### 3.2.3 Arbeits-, Schutz- und Gefahrenbereich

Arbeitsbereiche müssen auf das erforderliche Mindestmaß beschränkt werden, um eine Gefährdung von Personen und Sachen auszuschließen. Sichere Achsbereichsbegrenzungen, die zum Personenschutz benötigt werden, sind konfigurierbar.



Weitere Informationen zur Konfiguration sicherer Achsbereichsbegrenzungen sind in der Bedien- und Programmieranleitung im Kapitel "Sicherheitskonfiguration" zu finden. (>>> 13 "Sicherheitskonfiguration" Seite 183)

Der Gefahrenbereich beinhaltet den Arbeitsbereich und die Anhaltewege des Manipulators. Bei einem Stopp bremst der Manipulator und kommt im Gefahrenbereich zu stehen. Der Schutzbereich ist der Bereich außerhalb des Gefahrenbereichs.

Der Gefahrenbereich ist durch trennende Schutzeinrichtungen zu sichern, z. B. durch Lichtschranken, Lichtvorhänge oder Sperrzäune. Sind keine trennenden Schutzeinrichtungen vorhanden, müssen die Anforderungen an den kollaborierenden Betrieb nach EN ISO 10218 erfüllt werden. An Einlege- und Übergabebereichen dürfen keine Scher- und Quetschstellen entstehen.

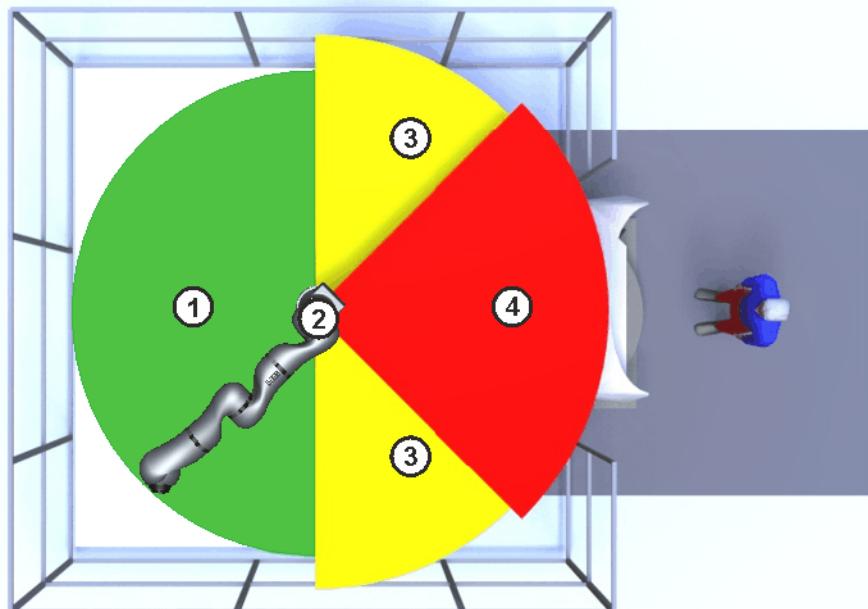


Abb. 3-1: Beispiel Achsbereich A1

1 Arbeitsbereich  
2 Manipulator

3 Anhalteweg  
4 Schutzbereich

### 3.2.4 Sicherheitsgerichtete Funktionen

Folgende sicherheitsgerichteten Funktionen sind am Industrieroboter vorhanden und fest definiert:

- NOT-HALT-Einrichtung

- Zustimmeinrichtung
- Verriegelung der Betriebsart (durch Schlüsselschalter)

Folgende sicherheitsgerichteten Funktionen sind vorkonfiguriert und können über die Sicherheitsschnittstelle der Robotersteuerung in die Anlage integriert werden:

- Bedienerschutz (= Anschluss für die Verriegelung von trennenden Schutzeinrichtungen)
- Externe NOT-HALT-Einrichtung
- Externer Sicherheitshalt 1 (bahntreu)

Weitere nicht defaultmäßig vorhandene sicherheitsgerichtete Funktionen sind konfigurierbar, z. B.:

- Externe Zustimmeinrichtung
- Externer sicherer Betriebshalt
- Achsspezifische Arbeitsraumüberwachung
- Kartesische Arbeitsraumüberwachung
- Kartesische Schutzraumüberwachung
- Geschwindigkeitsüberwachung
- Stillstandsüberwachung
- Achsmomentenüberwachung
- Kollisionserkennung



Weitere Informationen zur Konfiguration der Sicherheitsfunktionen sind in der Bedien- und Programmieranleitung im Kapitel "Sicherheitskonfiguration" zu finden. ([>>> 13 "Sicherheitskonfiguration" Seite 183](#))

In den folgenden Abschnitten zur Sicherheit wird die Default-Konfiguration der vorkonfigurierten Sicherheitsfunktionen beschrieben.

### 3.2.4.1 NOT-HALT-Einrichtung

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

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

- Der Manipulator stoppt mit einem Sicherheitshalt 1 (bahntreu).

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



**WARNUNG** Werkzeuge oder andere Einrichtungen, die mit dem Manipulator verbunden sind, müssen anlagenseitig in den NOT-HALT-Kreis eingebunden werden, wenn von ihnen Gefahren ausgehen können.

Wenn dies nicht beachtet wird, können Tod, schwere Verletzungen oder erheblicher Sachschaden die Folge sein.

Wenn für das smartPAD eine Halterung verwendet wird, die das NOT-HALT-Gerät am smartPAD verdeckt, muss eine externe NOT-HALT-Einrichtung installiert werden, die immer erreichbar ist.

([>>> 3.2.4.4 "Externe NOT-HALT-Einrichtung" Seite 31](#))

### 3.2.4.2 Zustimmeinrichtung

Die Zustimmeinrichtung des Industrieroboters sind die Zustimmungsschalter am smartPAD.

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

- Nicht gedrückt
- Mittelstellung
- Durchgedrückt (Panikstellung)

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

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

Bei einer Fehlfunktion eines Zustimmungsschalters (z. B. Klemmen in Mittelstellung) kann der Industrieroboter mit folgenden Methoden gestoppt werden:

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



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

### 3.2.4.3 Bedienerschutz

Das Signal Bedienerschutz dient zur Verriegelung trennender Schutzeinrichtungen, z. B. Schutztüren. In der Default-Konfiguration ist ohne dieses Signal kein Automatikbetrieb möglich. Alternativ müssen die Anforderungen an den kollaborierenden Betrieb nach EN ISO 10218 erfüllt werden.

Reaktion des Industrieroboters bei einem Signalverlust während des Automatikbetriebs, z. B. Schutztür wird geöffnet (Default-Konfiguration):

- Der Manipulator stoppt mit einem Sicherheitshalt 1 (bahntreu).

In den Betriebsarten Manuell Reduzierte Geschwindigkeit (T1) und KRF ist der Bedienerschutz defaultmäßig nicht aktiv, d. h. das Signal wird nicht ausgewertet. In der Betriebsart Manuell Hohe Geschwindigkeit (T2) ist der Bedienerschutz aktiv.

**⚠️ WARNUNG**

Nach einem Signalverlust darf der Automatikbetrieb nicht allein durch das Schließen der Schutzeinrichtung wieder fortgesetzt werden, sondern erst, wenn das Signal für den Bediener-schutz durch eine zusätzliche Einrichtung gesetzt wird, z. B. durch einen Quittierungstaster. Der Systemintegrator muss hierfür Sorge tragen. Dies soll verhindern, dass der Automatikbetrieb versehentlich fortgesetzt wird, während sich Personen im Gefahrenbereich befinden, z. B. durch Zufallen der Schutztür.

- Diese zusätzliche Einrichtung muss so gestaltet sein, dass vorher eine tatsächliche Prüfung des Gefahrenbereichs stattfinden kann. Einrichtungen, die dies nicht zulassen (z. B. weil sie automatisch auf das Schließen der Schutzeinrichtung folgen) sind unzulässig.
- Wenn dies nicht beachtet wird, können Tod von Personen, schwere Verletzungen oder erheblicher Sachschaden die Folge sein.

#### 3.2.4.4 Externe NOT-HALT-Einrichtung

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.

Reaktion des Industrieroboters, wenn das externe NOT-HALT-Gerät gedrückt wird (Default-Konfiguration):

- Der Manipulator stoppt mit einem Sicherheitshalt 1 (bahntreu).

Es können mehrere externe NOT-HALT-Einrichtungen über die Sicherheits-schnittstelle der Robotersteuerung angeschlossen werden. Externe NOT-HALT-Einrichtungen sind nicht im Lieferumfang des Industrieroboters enthalten.

#### 3.2.4.5 Externer Sicherheitshalt 1 (bahntreu)

Der externe Sicherheitshalt 1 (bahntreu) kann über einen Eingang an der Sicherheitsschnittstelle ausgelöst werden (Default-Konfiguration). Der Zustand bleibt erhalten, so lange das externe Signal FALSE ist. Wenn das externe Signal TRUE ist, kann der Manipulator wieder verfahren werden. Es ist keine Quittierung notwendig.

#### 3.2.4.6 Externe Zustimmeinrichtung

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

Es können mehrere externe Zustimmeinrichtungen über die Sicherheits-schnittstelle der Robotersteuerung angeschlossen werden. Externe Zustimm-einrichtungen sind nicht im Lieferumfang des Industrierobters enthalten.

Eine externe Zustimmeinrichtung kann zum Handführen des Roboters ver-wendet werden. Bei aktiver Zustimmung darf der Roboter nur mit reduzierter Geschwindigkeit bewegt werden.

Für das Handführen ist eine sicherheitsgerichtete Geschwindigkeitsüberwa-chung mit einer maximal zulässigen Geschwindigkeit von 250 mm/s vorkonfi-guriert. Die maximal zulässige Geschwindigkeit kann angepasst werden.

Der maximal zulässige Geschwindigkeitswert muss im Rahmen einer Risiko-beurteilung ermittelt werden.

### 3.2.4.7 Externer sicherer Betriebshalt

Der sichere Betriebshalt ist eine Stillstandsüberwachung. Er stoppt die Roboterbewegung nicht, sondern überwacht, ob die Roboterachsen still stehen.

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

### 3.2.5 Auslöser für sicherheitsgerichtete Stopp-Reaktionen

Stopp-Reaktionen werden aufgrund von Bedienhandlungen oder als Reaktion auf Überwachungen und Fehler ausgeführt. Die folgenden Tabellen zeigen die Stopp-Reaktionen in Abhängigkeit der eingestellten Betriebsart.

#### Übersicht

In KUKA Sunrise werden folgende Auslöser unterschieden:

- Fest definierte Auslöser

Fest definierte Auslöser für Stopp-Reaktionen und die zugehörige Stopp-Kategorie werden vom System vorgegeben und können nicht verändert werden. Im Rahmen der anwenderspezifischen Sicherheitskonfiguration ist jedoch eine Verschärfung der verwendeten Stopp-Reaktion möglich.

- Anwenderspezifische Auslöser

Zusätzlich zu den fest definierten Auslösern kann der Anwender weitere Auslöser für Stopp-Reaktionen inklusive der zugehörigen Stopp-Kategorie konfigurieren.



Weitere Informationen zur Konfiguration der Sicherheitsfunktionen sind in der Bedien- und Programmieranleitung im Kapitel "Sicherheitskonfiguration" zu finden. (>>> 13 "Sicherheitskonfiguration" Seite 183)

#### Fest definierte Auslöser

Folgende Auslöser für Stopp-Reaktionen sind fest definiert:

Auslöser	T1, T2, KRF	AUT
Betriebsart wechseln während Betrieb	Sicherheitshalt 1 (bahntreu)	
Zustimmung lösen	Sicherheitshalt 1 (bahntreu)	-
Zustimmung durchdrücken (Panikstellung)	Sicherheitshalt 1 (bahntreu)	-
Lokalen NOT-HALT betätigen	Sicherheitshalt 1 (bahntreu)	
Fehler in Sicherheitssteuerung	Sicherheitshalt 1	

#### Anwenderspezifische Auslöser

Die Robotersteuerung wird mit einer Sicherheitskonfiguration ausgeliefert, die bei der Erstinbetriebnahme aktiv ist. Diese enthält folgende von KUKA vorkonfigurierte anwenderspezifische Auslöser für Stopp-Reaktionen (zusätzlich zu den fest definierten Auslösern).

Auslöser	T1, KRF	T2, AUT
Schutztür öffnen (Bedienerschutz)	-	Sicherheitshalt 1 (bahntreu)

Beim Erstellen eines neuen Sunrise-Projekts wird automatisch eine projektspezifische Sicherheitskonfiguration erzeugt. Diese enthält folgende von

KUKA vorkonfigurierte anwenderspezifische Auslöser für Stopp-Reaktionen (zusätzlich zu den fest definierten Auslösern).



Beim Übertragen des Sunrise-Projekts auf die Robotersteuerung wird die werkseitige Sicherheitskonfiguration durch die projektspezifische Sicherheitskonfiguration überschrieben. Dadurch wird eine Aktivierung der Sicherheitskonfiguration erforderlich.

Weitere Informationen zur Aktivierung der Sicherheitskonfiguration sind in der Bedien- und Programmieranleitung im Kapitel "Sicherheitskonfiguration" zu finden. ([>>> 13 "Sicherheitskonfiguration" Seite 183](#))

Auslöser	T1, KRF	T2, AUT
Schutztür öffnen (Bedienerschutz)	-	Sicherheitshalt 1 (bahntreu)
Externen NOT-HALT betätigen		Sicherheitshalt 1 (bahntreu)
Externer Sicherheitshalt		Sicherheitshalt 1 (bahntreu)

## Auslöser Handführen

Ist eine Zustimmeinrichtung für das Handführen konfiguriert, sind zusätzlich folgende Auslöser für Stopp-Reaktionen fest definiert:

Auslöser	T1, KRF	T2, AUT
Zustimmung Handführgerät lösen	Sicherheitshalt 1 (bahntreu)	-
Zustimmung Handführgerät durchdrücken (Panikstellung)	Sicherheitshalt 1 (bahntreu)	-
Überschreiten der maximal zulässigen Geschwindigkeit bei erteilter Zustimmung Handführgerät		Sicherheitshalt 1 (bahntreu)

Die maximal zulässige Geschwindigkeit für das Handführen ist mit 250 mm/s vorkonfiguriert. Die maximal zulässige Geschwindigkeit kann angepasst werden.

Der maximal zulässige Geschwindigkeitswert muss im Rahmen einer Risiko-beurteilung ermittelt werden.

([>>> 13.8.5.3 "Geschwindigkeitsüberwachung während des Handführens" Seite 211](#))

## 3.2.6 Nicht sicherheitsgerichtete Funktionen

### 3.2.6.1 Betriebsartenwahl

Der Industrieroboter kann in folgenden Betriebsarten betrieben werden:

- Manuell Reduzierte Geschwindigkeit (T1)
- Manuell Hohe Geschwindigkeit (T2)
- Automatik (AUT)
- Kontrollierte Roboterfahrt (KRF)

Betriebsart	Verwendung	Geschwindigkeiten
T1	Programmieren, Teachen und Testen von Programmen	<ul style="list-style-type: none"> <li>■ Programmverifikation: Reduzierte programmierte Geschwindigkeit, maximal 250 mm/s</li> <li>■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s</li> <li>■ Handführen: Keine Begrenzung der Geschwindigkeit, sondern sicherheitsgerichtete Geschwindigkeitsüberwachung gemäß Sicherheitskonfiguration</li> </ul> <p><b>Hinweis:</b> Bei einer mobilen Plattform gilt die maximale Geschwindigkeit von 250 mm/s nicht.</p>
T2	Testen von Programmen  Nur mit geschlossener Schutztür möglich	<ul style="list-style-type: none"> <li>■ Programmverifikation: Programmierte Geschwindigkeit</li> <li>■ Handbetrieb: Nicht möglich</li> </ul>
AUT	Automatisches Ausführen von Programmen  Für Industrieroboter mit und ohne übergeordnete Steuerung	<ul style="list-style-type: none"> <li>■ Programmbetrieb: Programmierte Geschwindigkeit</li> <li>■ Handbetrieb: Nicht möglich</li> </ul>
KRF	<ul style="list-style-type: none"> <li>■ Herausfahren des Industrieroboters aus einem verletzten kartesischen oder achsspezifischen Bereich</li> <li>■ Herausfahren des Industrieroboters aus einem verletzten Bereich der Werkzeugorientierung</li> <li>■ Freifahren des Industrieroboters aus Klemmsituationen bei Verletzung von Kraft- oder Momentengrenzen</li> <li>■ Verfahren des Industrieroboters, wenn bei aktiver kartesischer Geschwindigkeitsüberwachung ein Justageverlust für mindestens einen Positionssensor vorliegt</li> </ul> <p>KRF ist eine Betriebsart, die zur Verfügung steht, wenn der Industrieroboter von der Sicherheitssteuerung aufgrund einer der folgenden Ursachen gestoppt wird:</p> <ul style="list-style-type: none"> <li>■ Industrieroboter verletzt einen sicher überwachten Raum.</li> <li>■ Orientierung des sicherheitsgerichteten Werkzeugs liegt außerhalb des sicher überwachten Bereichs.</li> <li>■ Industrieroboter verletzt eine sicher überwachte Kraft- oder Momentengrenze.</li> <li>■ Ein Positionssensor ist bei aktiver kartesischer Geschwindigkeitsüberwachung nicht justiert.</li> </ul>	<ul style="list-style-type: none"> <li>■ Programmbetrieb: Reduzierte programmierte Geschwindigkeit, maximal 250 mm/s</li> <li>■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s</li> </ul>

### 3.2.6.2 Software-Endschalter

Die Achsbereiche aller Manipulatorachsen sind über nicht sicherheitsgerichtete Software-Endschalter begrenzt. Diese Software-Endschalter dienen nur als Maschinenschutz und sind so voreingestellt, dass der Manipulator bei Überfahren der Achsgrenze geregelt angehalten und die Mechanik nicht beschädigt wird.

## 3.3 Zusätzliche Schutzausstattung

### 3.3.1 Tippbetrieb

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

- Das Loslassen des Zustimmungsschalters am smartPAD löst einen Sicherheitshalt aus. (=> 3.2.5 "Auslöser für sicherheitsgerichtete Stopp-Reaktionen" Seite 32)
- Das Durchdrücken des Zustimmungsschalters am smartPAD löst einen Sicherheitshalt 1 aus.
- Das Loslassen der Start-Taste löst einen Stopp der Stopp-Kategorie 1 (bahntreu) aus.

### 3.3.2 Kennzeichnungen am Industrieroboter

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

Kennzeichnungen am Industrieroboter sind:

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



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

### 3.3.3 Externe Schutzeinrichtungen

Der Zutritt von Personen in den Gefahrenbereich des Industrieroboters ist durch Schutzeinrichtungen zu verhindern. Alternativ müssen die Anforderungen an den kollaborierenden Betrieb nach EN ISO 10218 erfüllt werden. Der Systemintegrator hat hierfür Sorge zu tragen.

Trennende Schutzeinrichtungen müssen folgende Anforderungen erfüllen:

- Sie entsprechen den Anforderungen von EN 953.
- Sie verhindern den Zutritt von Personen in den Gefahrenbereich und können nicht auf einfache Weise überwunden werden.
- Sie sind ausreichend befestigt und halten den vorhersehbaren Betriebs- und Umgebungskräften stand.

- Sie stellen nicht selbst eine Gefährdung dar und können keine Gefährdungen verursachen.
- Der vorgeschriebene Mindestabstand zum Gefahrenbereich wird eingehalten.

Schutztüren (Wartungstüren) müssen folgende Anforderungen erfüllen:

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



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

## Andere Schutzeinrichtungen

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

### 3.4 Sicherheitsmaßnahmen

#### 3.4.1 Allgemeine Sicherheitsmaßnahmen

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

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



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



Der Aufenthalt unter der Robotermechanik kann zum Tod oder zu schweren Verletzungen führen. Insbesondere, wenn mit dem Industrieroboter Objekte bewegt werden, die sich lösen können (z. B. aus einem Greifer). Aus diesem Grund ist der Aufenthalt unter der Robotermechanik verboten!

## smartPAD

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

## Änderungen

Nach Änderungen am Industrieroboter muss geprüft werden, ob das erforderliche Sicherheitsniveau gewährleistet ist. Für diese Prüfung sind die geltenden

staatlichen oder regionalen Arbeitsschutzzvorschriften zu beachten. Zusätzlich sind alle Sicherheitsfunktionen auf ihre sichere Funktion zu testen.

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

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

Der Roboter darf bei laufender Robotersteuerung nicht an- und abgesteckt werden.

#### **Störungen**

Bei Störungen am Industrieroboter ist wie folgt vorzugehen:

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

#### **3.4.2 Transport**

##### **Manipulator**

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

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

##### **Robotersteuerung**

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

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

#### **3.4.3 Inbetriebnahme und Wiederinbetriebnahme**

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

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



Das Passwort für die Aktivierung der Sicherheitskonfiguration muss vor der Inbetriebnahme geändert werden. Dieses Passwort darf nur geschulten Sicherheitsinbetriebnehmern bekanntgegeben werden, die autorisiert sind, die Sicherheitskonfiguration zu aktivieren.  
 (>>> 13.7.3 "Passwort für die Aktivierung der Sicherheitskonfiguration ändern" Seite 206)

**GEFAHR**

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



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

**HINWEIS**

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

**Funktionsprüfung**

Vor der Inbetriebnahme und Wiederinbetriebnahme sind folgende Prüfungen durchzuführen:

**Prüfung allgemein:**

Sicherzustellen ist:

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

**Prüfung der Sicherheitsfunktionen:**

Bei allen sicherheitsgerichteten Funktionen muss durch einen Funktionstest sichergestellt werden, dass sie korrekt arbeiten.

(>>> 13.11 "Übersicht Sicherheitsabnahme" Seite 243)

**Prüfung der sicherheitsrelevanten mechanischen und elektromechanischen Komponenten:**

Folgende Prüfungen sind vor der Inbetriebnahme und mindestens alle 12 Monate durchzuführen, sofern nach Gefährdungsbeurteilung am Arbeitsplatz nicht abweichend bestimmt:

- NOT-HALT-Einrichtung am smartPAD drücken. Es muss eine Meldung am smartPAD angezeigt werden, dass der NOT-HALT ausgelöst wurde und es darf keine Fehlermeldung zur NOT-HALT-Einrichtung angezeigt werden.
- Bei allen 3 Zustimmungsschaltern am smartPAD und bei Zustimmungsschalter am Medien-Flansch Touch (wenn vorhanden) Roboter im Testbetrieb verfahren und Zustimmungsschalter loslassen. Die Roboterbewegung muss gestoppt werden und es darf keine Fehlermeldung zur Zustimmungseinrichtung angezeigt werden. Wenn der Zustand

- des Zustimmungsschalters auf einen Ausgang konfiguriert ist, kann die Prüfung auch über den Ausgang durchgeführt werden.
- Bei allen 3 Zustimmungsschaltern am smartPAD und bei Zustimmungsschalter am Medien-Flansch Touch (wenn vorhanden)
  - Roboter im Testbetrieb verfahren und Zustimmungsschalter durchdrücken. Die Roboterbewegung muss gestoppt werden und es darf keine Fehlermeldung zur Zustimmmeinrichtung angezeigt werden. Wenn der Zustand des Zustimmungsschalters auf einen Ausgang konfiguriert ist, kann die Prüfung auch über den Ausgang durchgeführt werden.
  - Schlüsselschalter am smartPAD nach rechts und wieder zurückdrehen. Es darf keine Fehlermeldung am smartPAD angezeigt werden.
  - Abschaltbarkeit der sicheren Ausgänge durch Ausschalten und Wiedereinschalten der Robotersteuerung prüfen. Nach dem Einschalten darf keine Fehlermeldung zu einem sicheren Ausgang angezeigt werden.



Bei einer unvollständigen Inbetriebnahme der Anlage sind zusätzliche risikominderende Ersatzmaßnahmen zu ergreifen und zu dokumentieren, z. B. Anbringen eines Schutzaufs oder Warnschildes, Verriegelung des Hauptschalters etc. Eine unvollständige Inbetriebnahme liegt beispielsweise vor, wenn noch nicht alle notwendigen Sicherheitsüberwachungen implementiert wurden oder die Sicherheitsfunktionen noch nicht auf ihre sichere Funktion getestet wurden.

#### **Prüfung der Funktionsfähigkeit der Bremsen:**

Für den KUKA LBR iiwa (alle Varianten) steht ein Bremsentest zur Verfügung, mit dem geprüft werden kann, ob die Bremsen aller Achsen ein ausreichendes Bremsmoment aufbringen.

(>>> 8 "Bremsentest" Seite 117)

Sofern durch eine Risikobetrachtung nicht anders bestimmt, muss der Bremsentest regelmäßig durchgeführt werden:

- Der Bremsentest ist bei der Inbetriebnahme und Wiederinbetriebnahme des Industrieroboters für jede Achse durchzuführen.
- Während des Betriebs ist der Bremsentest täglich durchzuführen.

Der Benutzer kann durch eine Risikobetrachtung ermitteln, ob es in seinem konkreten Anwendungsfall erforderlich ist, den Bremsentest durchzuführen und in welcher Regelmäßigkeit er durchzuführen ist.

#### **3.4.4 Manueller Betrieb**

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

- Tippbetrieb
- Teachen
- Programmverifikation

Beim manuellen Betrieb ist Folgendes zu beachten:

- Neue oder geänderte Programme müssen immer zuerst in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden.
- Werkzeuge und Manipulator dürfen niemals den Absperrzaun berühren oder über den Absperrzaun hinausragen.
- Werkstücke, Werkzeuge und andere Gegenstände dürfen durch das Verfahren des Industrieroboters weder eingeklemmt werden, noch zu Kurzschlüssen führen oder herabfallen.

- Alle Einrichtarbeiten müssen so weit wie möglich von außerhalb des durch Schutzeinrichtungen abgegrenzten Raumes durchgeführt werden.

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

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

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

Wenn es notwendig ist, dass sich mehrere Personen im durch Schutzeinrichtungen abgegrenzten Raum aufhalten, muss Folgendes beachtet werden:

- Jede Person muss eine Zustimmeinrichtung zur Verfügung haben.
- Alle Personen müssen ungehinderte Sicht auf den Industrieroboter haben.
- Zwischen allen Personen muss immer Möglichkeit zum Blickkontakt bestehen.

- Der Bediener muss eine Position einnehmen, aus der er den Gefahrenbereich einsehen kann und einer Gefahr ausweichen kann.

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

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

### 3.4.5 Automatikbetrieb

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

- Alle Sicherheits- und Schutzeinrichtungen sind vorhanden und funktionsfähig.
- Es befinden sich keine Personen in der Anlage. Oder die Anforderungen an den kollaborierenden Betrieb nach EN ISO 10218 sind erfüllt.
- Die festgelegten Arbeitsverfahren werden befolgt.

Wenn der Manipulator ohne ersichtlichen Grund stehen bleibt, darf der Gefahrenbereich erst betreten werden, wenn ein NOT-HALT ausgelöst wurde.

### 3.4.6 Wartung und Instandsetzung

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

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

Sicherheitsmaßnahmen bei Tätigkeiten am Industrieroboter sind:

- Tätigkeiten außerhalb des Gefahrenbereichs durchführen. Wenn Tätigkeiten innerhalb des Gefahrenbereichs durchzuführen sind, muss der Betreiber

ber zusätzliche Schutzmaßnahmen festlegen, um einen sicheren Personenschutz zu gewährleisten.

- Industrieroboter ausschalten und gegen Wiedereinschalten (z. B. mit einem Vorhangeschloss) sichern. Wenn die Tätigkeiten bei eingeschalteter Robotersteuerung durchzuführen sind, muss der Betreiber zusätzliche Schutzmaßnahmen festlegen, um einen sicheren Personenschutz zu gewährleisten.
- Wenn die Tätigkeiten bei eingeschalteter Robotersteuerung durchzuführen sind, dürfen diese nur in der Betriebsart T1 durchgeführt werden.
- Tätigkeiten mit einem Schild an der Anlage kennzeichnen. Dieses Schild muss auch bei zeitweiser Unterbrechung der Tätigkeiten vorhanden sein.
- Die NOT-HALT-Einrichtungen müssen aktiv bleiben. Wenn Sicherheitsfunktionen oder Schutzeinrichtungen aufgrund Wartungs- oder Instandsetzungsarbeiten deaktiviert werden, muss die Schutzwirkung anschließend sofort wiederhergestellt werden.

**GEFAHR**

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

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

Reinigungs- und Pflegearbeiten sind gemäß der Betriebsanleitung durchzuführen.

#### Robotersteuerung

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

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

Nach Ausschalten der Robotersteuerung kann an verschiedenen Komponenten mehrere Minuten eine Spannung von über 60 V anliegen. Um lebensgefährliche Verletzungen zu verhindern, dürfen in diesem Zeitraum keine Tätigkeiten am Industrieroboter durchgeführt werden.

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

#### 3.4.7 Außerbetriebnahme, Lagerung und Entsorgung

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

#### 3.4.8 Sicherheitsmaßnahmen für "Single Point of Control"

##### Übersicht

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

## Komponenten:

- Tools zur Konfiguration von Bussystemen mit Online-Funktionalität



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

Da die sicheren Zustände von Aktoren in der Peripherie der Robotersteuerung nur dem Systemintegrator bekannt sind, obliegt es ihm diese Aktoren in einen sicheren Zustand zu versetzen.

**T1, T2, KRF**

In den Betriebsarten T1, T2 und KRF kann eine Roboterbewegung nur ausgelöst werden, wenn ein Zustimmungsschalter am smartPAD gedrückt ist.

**Tools zur Konfiguration von Bussystemen**

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

- KUKA Sunrise.Workbench
- WorkVisual von KUKA
- Tools anderer Hersteller

## Sicherheitsmaßnahmen:

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

**3.5 Angewandte Normen und Vorschriften**

Name	Definition	Ausgabe
2006/42/EG	<b>Maschinenrichtlinie:</b> Richtlinie 2006/42/EG des Europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen und zur Änderung der Richtlinie 95/16/EG (Neufassung)	2006
2004/108/EG	<b>EMV-Richtlinie:</b> Richtlinie 2004/108/EG des Europäischen Parlaments und des Rates vom 15. Dezember 2004 zur Angleichung der Rechtsvorschriften der Mitgliedstaaten über die elektromagnetische Verträglichkeit und zur Aufhebung der Richtlinie 89/336/EWG	2004
EN ISO 13850	<b>Sicherheit von Maschinen:</b> NOT-HALT-Gestaltungsleitsätze	2008
EN ISO 13849-1	<b>Sicherheit von Maschinen:</b> Sicherheitsbezogene Teile von Steuerungen; Teil 1: Allgemeine Gestaltungsleitsätze	2008
EN ISO 13849-2	<b>Sicherheit von Maschinen:</b> Sicherheitsbezogene Teile von Steuerungen; Teil 2: Validierung	2012

<b>EN ISO 12100</b>	<b>Sicherheit von Maschinen:</b> Allgemeine Gestaltungsleitsätze, Risikobeurteilung und Risikominderung	2010
<b>EN ISO 10218-1</b>	<b>Industrieroboter:</b> Sicherheit <b>Hinweis:</b> Inhalt entspricht <b>ANSI/RIA R.15.06-2012, Teil 1</b>	2011
<b>EN 614-1 + A1</b>	<b>Sicherheit von Maschinen:</b> Ergonomische Gestaltungsgrundsätze; Teil 1: Begriffe und allgemeine Leitsätze	2009
<b>EN 61000-6-2</b>	<b>Elektromagnetische Verträglichkeit (EMV):</b> Teil 6-2: Fachgrundnormen; Störfestigkeit für Industriebereich	2005
<b>EN 61000-6-4 + A1</b>	<b>Elektromagnetische Verträglichkeit (EMV):</b> Teil 6-4: Fachgrundnormen; Störaussendung für Industriebereich	2011
<b>EN 60204-1 + A1</b>	<b>Sicherheit von Maschinen:</b> Elektrische Ausrüstung von Maschinen; Teil 1: Allgemeine Anforderungen	2009



## 4 Installation KUKA Sunrise.Workbench

### 4.1 Systemvoraussetzungen PC

<b>Hardware</b>	<b>Mindestanforderungen</b>
	<ul style="list-style-type: none"> <li>■ PC mit Pentium IV-Prozessor, mindestens 1500 MHz</li> <li>■ 1 GB Arbeitsspeicher</li> <li>■ 1 GB freier Festplattenspeicher</li> <li>■ DirectX8-kompatible Grafikkarte mit einer Auflösung von 1024x768 Pixel</li> </ul>
	<b>Empfohlene Anforderungen</b>
	<ul style="list-style-type: none"> <li>■ PC mit Pentium IV-Prozessor und 2500 MHz</li> <li>■ 2 GB Arbeitsspeicher</li> <li>■ DirectX8-kompatible Grafikkarte mit einer Auflösung von 1280x1024 Pixel</li> </ul>
<b>Software</b>	<ul style="list-style-type: none"> <li>■ Windows 7</li> </ul> <p>Zur Buskonfiguration wird folgende Software benötigt:</p> <ul style="list-style-type: none"> <li>■ WorkVisual 3.0</li> </ul>

### 4.2 Sunrise.Workbench installieren

<b>Voraussetzung</b>	<ul style="list-style-type: none"> <li>■ Lokale Administratorrechte</li> </ul>
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. Installation über die Datei <b>Setup.exe</b> starten.</li> <li>2. Sprache auswählen und mit <b>Ok</b> bestätigen. Der Installations-Assistent <b>Sunrise Workbench Setup</b> öffnet sich.</li> <li>3. Mit <b>Weiter &gt;</b> zur nächsten Seite.</li> <li>4. Die Lizenzbedingungen akzeptieren. Mit <b>Weiter &gt;</b> zur nächsten Seite.</li> <li>5. Im Feld <b>Verzeichnis</b>: ist das Default-Verzeichnis für die Installation angegeben. Über <b>Wählen...</b> kann ein anderes Installationsverzeichnis ausgewählt werden. Mit <b>Weiter &gt;</b> zur nächsten Seite.</li> <li>6. Wenn keine Desktop-Verknüpfung für Sunrise.Workbench erstellt werden soll: Option <b>Desktop</b> deaktivieren (Häkchen entfernen).</li> <li>7. Mit <b>Weiter &gt;</b> zur nächsten Seite und auf <b>Installieren</b> klicken. Sunrise.Workbench wird installiert.</li> <li>8. Wenn die Installation abgeschlossen ist, auf <b>Fertigstellen</b> klicken, um den Installations-Assistenten zu schließen.</li> </ol> <p>Im Anschluss wird Sunrise.Workbench direkt gestartet. Wenn dies nicht gewünscht ist, die Option <b>Sunrise Workbench ausführen</b> deaktivieren (Häkchen entfernen).</p>

### 4.3 Sunrise.Workbench deinstallieren



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

<b>Voraussetzung</b>	<ul style="list-style-type: none"> <li>■ Lokale Administratorrechte</li> </ul>
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. Im Windows Startmenü <b>Systemsteuerung</b> (<b>&gt; Programme</b>) &gt; <b>Programme und Funktionen</b> wählen. Die Liste der installierten Programme wird angezeigt.</li> </ol>

2. Den Eintrag **Sunrise Workbench** markieren. Auf **Deinstallieren** klicken und Sicherheitsabfrage mit **Ja** beantworten.

**Alternative  
Vorgehensweise**

- Im Windows Startmenü das beim Installieren angegebene Installationsverzeichnis öffnen und auf **Uninstall** klicken.

#### 4.4 Sprachpaket in Sunrise.Workbench installieren

**Beschreibung**

Die Bedienoberfläche von Sunrise.Workbench steht aktuell in folgenden Sprachen zur Verfügung:

Deutsch	Italienisch
Englisch	Spanisch
Französisch	

Sprachen, die erst nach Auslieferung einer Software zur Verfügung stehen, können bei Bedarf nachträglich installiert werden.

**Voraussetzung**

- Lokale Administratorrechte
- Sprachpaket, das die gewünschte Sprache enthält, ist verfügbar.

**Vorgehensweise**

1. Sunrise.Workbench schließen.
2. Datei **SWB\_LanguagePack-<Version>-Setup.exe** ausführen. Das Sprachpaket wird installiert.
3. Sunrise.Workbench neu starten. Sunrise.Workbench startet automatisch in der lokal eingestellten Windows-Sprache.

## 5 Bedienung KUKA Sunrise.Workbench

### 5.1 Sunrise.Workbench starten

#### Vorgehensweise

1. Das **Sunrise.Workbench**-Icon auf dem Desktop doppelklicken.  
Alternative:  
Im Windows-Startmenü das Installationsverzeichnis öffnen und auf **Sunrise Workbench** doppelklicken.  
Das Fenster **Startprogramm für Arbeitsbereich** öffnet sich.
2. Im Feld **Arbeitsbereich** das Verzeichnis für den Arbeitsbereich angeben, in das Projekte gespeichert werden.
  - Es wird ein Verzeichnis vorgegeben. Über **Durchsuchen...** kann das Verzeichnis geändert werden.
  - Wenn beim nächsten Start von Sunrise.Workbench der Arbeitsbereich nicht mehr abgefragt werden soll, die Option **Diesen Wert als Standardwert verwenden[...]** aktivieren (Häkchen setzen).
3. Beim ersten Start von Sunrise.Workbench öffnet sich ein Willkommens-Bildschirm. Hier stehen verschiedene Optionen zur Verfügung.
  - Auf **Arbeitsbereich** klicken, um die Bedienoberfläche von Sunrise.Workbench zu öffnen.  
(>>> 5.2 "Übersicht Bedienoberfläche Sunrise.Workbench" Seite 47)
  - Auf **Neues Sunrise-Projekt** klicken, um direkt ein neues Sunrise-Projekt zu erstellen. Der Assistent für die Projekterstellung öffnet sich.  
(>>> 5.3 "Sunrise-Projekt mit Vorlage erstellen" Seite 51)

### 5.2 Übersicht Bedienoberfläche Sunrise.Workbench

Die Bedienoberfläche von KUKA Sunrise.Workbench besteht aus mehreren Sichten. Der Zusammenschluss mehrerer Sichten wird als Perspektive bezeichnet. KUKA Sunrise.Workbench bietet unterschiedliche vorkonfigurierte Perspektiven an.

Defaultmäßig wird die Perspektive **Programmierung** geöffnet. Weitere Perspektiven können eingeblendet werden.

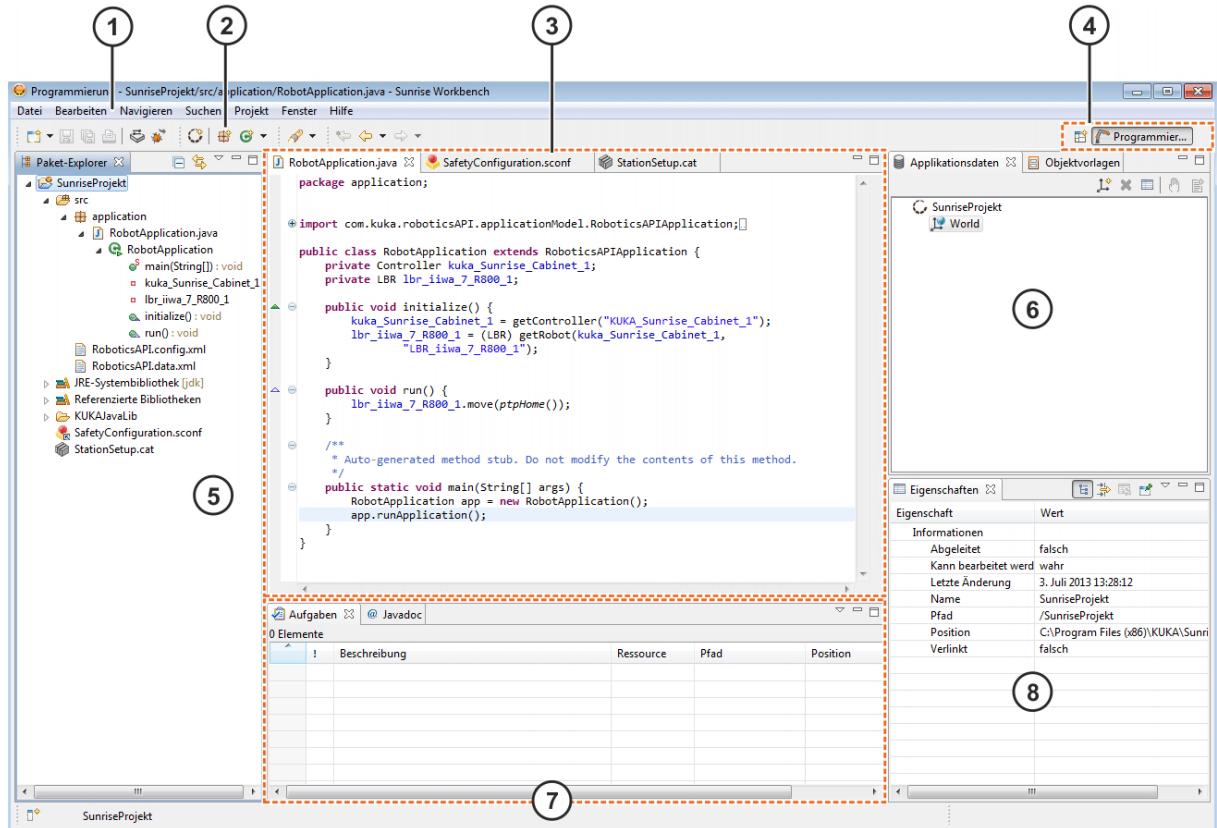


Abb. 5-1: Übersicht Bedienoberfläche - Perspektive "Programmierung"

Pos.	Beschreibung
1	Menüleiste
2	Symbolleisten (>>> 5.2.3 "Symbolleisten" Seite 50)
3	Editoren-Bereich Hier werden geöffnete Dateien angezeigt und bearbeitet, z. B. Roboter-Applikationen.
4	Auswahl Perspektive Hier kann zwischen verschiedenen bereits verwendeten Perspektiven gewechselt werden, indem man den Namen der entsprechenden Perspektive anklickt oder über das Symbol <b>Perspektive öffnen</b> auswählt. (>>> 5.2.2 "Verschiedene Perspektiven der Bedienoberfläche anzeigen" Seite 49)
5	Sicht <b>Paket-Explorer</b> Die Sicht enthält die erstellten Projekte und die zugehörigen Dateien.
6	Sichten <b>Applikationsdaten</b> und <b>Objektvorlagen</b> <ul style="list-style-type: none"> <li>■ <b>Applikationsdaten:</b> Die Sicht zeigt die für ein Projekt angelegten Frames in einer Baumstruktur an.</li> <li>■ <b>Objektvorlagen:</b> Die Sicht zeigt die für ein Projekt angelegten geometrischen Objekte, Werkzeuge und Werkstücke in einer Baumstruktur an.</li> </ul>

<b>Pos.</b>	<b>Beschreibung</b>
7	Sichten <b>Aufgaben</b> und <b>Javadoc</b> <ul style="list-style-type: none"> <li>■ <b>Aufgaben:</b> Anzeige der Aufgaben, die ein Benutzer angelegt hat</li> <li>■ <b>Javadoc:</b> Anzeige der Javadoc-Informationen zu den markierten Elementen einer Java-Applikation</li> </ul>
8	Sicht <b>Eigenschaften</b> Wird ein Objekt, z. B. ein Projekt, Frame oder Werkzeug, in einer Sicht angewählt, werden dessen Eigenschaften hier angezeigt.

### 5.2.1 Sichten anders anordnen

- Vorgehensweise**
1. Die Sicht an der Titelleiste mit gedrückter linker Maustaste greifen und auf der Bedienoberfläche verschieben.  
Die möglichen Positionen für die Sicht werden dabei durch einen grauen Rahmen angezeigt.
  2. Maustaste loslassen, wenn die Position markiert ist, an der die Sicht eingefügt werden soll.

### 5.2.2 Verschiedene Perspektiven der Bedienoberfläche anzeigen

- Beschreibung** Die Bedienoberfläche kann in unterschiedlichen Perspektiven angezeigt werden. Diese können über die Menüfolge **Fenster > Perspektive öffnen** oder über das Symbol **Perspektive öffnen** ausgewählt werden.

Die Perspektiven sind abgestimmt auf verschiedene Arbeitsschwerpunkte:

<b>Perspektive</b>	<b>Schwerpunkt</b>
<b>Programmierung</b>	Die Perspektive enthält geeignete Sichten für die Bearbeitung von Sunrise-Projekten. Beispielsweise für die Stationskonfiguration, Sicherheitskonfiguration und Applikationsentwicklung.
<b>Debuggen</b>	Die Perspektive enthält geeignete Sichten für die Fehlersuche und Behebung von Programmierfehlern.

Perspektiven können an die Bedürfnisse des Benutzers angepasst werden.  
Beispiele:

- Erstellen eigener Perspektiven
- Sichten ein-/ausblenden
- Menüs ein-/ausblenden
- Menüpunkte ein-/ausblenden

Es ist möglich, die angepasste Perspektive als Default-Einstellung der Perspektive oder unter einem eigenen Namen zu speichern.

- Vorgehensweise** Sichten in der aktuellen Perspektive einblenden:

- Menüfolge **Fenster > Sicht anzeigen** und die gewünschte Sicht auswählen.  
Über den Menüpunkt **Andere...** können weitere Sichten ausgewählt werden.

Die aktuelle Perspektive wieder auf die Default-Einstellung zurücksetzen:

- Menüfolge **Fenster > Perspektive zurücksetzen...** wählen und Sicherheitsabfrage mit **Ja** beantworten.

Benutzerdefinierte Perspektive speichern:

1. Menüfolge **Fenster > Perspektive speichern als...** wählen.
2. Im Feld **Name** einen Namen für die Perspektive eingeben und mit **OK** bestätigen.

Wenn eine bereits vorhandene Perspektive ausgewählt und überschrieben wird, wird die Perspektive künftig mit diesen Einstellungen geöffnet.

### 5.2.3 Symbolleisten

Die Buttons, die defaultmäßig in den Symbolleisten zur Verfügung stehen, sind abhängig von der aktiven Perspektive. Hier sind die Buttons der Perspektive **Programmierung** beschrieben.

Programmierung	Symbol	Name / Beschreibung
		<b>Neu</b> Öffnet den Assistenten zum Erstellen von neuen Dokumenten.
		<b>Speichern</b> Speichert die aktuell geöffnete und angewählte Datei.
		<b>Alle speichern</b> Speichert alle Dateien und Projekte, die seit dem letzten Speichern bearbeitet wurden.
		<b>Drucken</b> Öffnet das Menü zum Drucken einer Datei.
		<b>Projekt synchronisieren</b> Synchronisiert das angewählte Projekt mit der Robotersteuerung.
		<b>Projekt debuggen</b> Stellt eine Remote-Verbindung zur Robotersteuerung her, um eine Applikation im laufenden Betrieb zu debuggen.
		<b>Sunrise-Projekt</b> Öffnet den Assistenten zum Erstellen eines neuen Sunrise-Projekts.
		<b>Neues Java-Paket</b> Öffnet den Assistenten zum Erstellen eines neuen Java-Pakets im angewählten Projekt.
		<b>Neue Java-Klasse</b> Öffnet den Assistenten zum Erstellen einer neuen Java-Klasse im angewählten Projekt.
		<b>Suchen</b> Öffnet den Assistenten zum Suchen von Wörtern oder Textbausteinen.
		<b>Letzte Bearbeitungsposition</b> Schaltet zur letzten Bearbeitungsposition in der aktuell geöffneten und angewählten Datei.

Symbol	Name / Beschreibung
	<b>Zurück zu ...</b> Schaltet zurück zu den vorherigen Bearbeitungsschritten.
	<b>Weiter zu ...</b> Schaltet wieder vorwärts zu den nachfolgenden Bearbeitungsschritten.

### 5.3 Sunrise-Projekt mit Vorlage erstellen

#### Vorgehensweise

1. Menüfolge **Datei > Neu > Sunrise-Projekt** wählen. Der Assistent für die Projekterstellung öffnet sich.
2. Im Feld **Steuerungs-IP-Adresse**: die IP-Adresse der Robotersteuerung eingeben, für die das Projekt erstellt werden soll.  
Später bei der Projektkonfiguration besteht die Möglichkeit, diese Adresse nochmal zu ändern.



Folgende Adress-Bereiche werden defaultmäßig von der Robotersteuerung für interne Zwecke genutzt. IP-Adressen aus diesen Bereichen können deshalb vom Benutzer nicht vergeben werden.

- 192.168.0.0 ... 192.168.0.255
- 172.16.0.0 ... 172.16.255.255
- 172.17.0.0 ... 172.17.255.255

3. Die Einstellung **Neues Projekt erstellen (Offline)** beibehalten.  
Mit **Weiter >** zur nächsten Seite.
4. Im Feld **Projektname**: einen Namen für das Projekt eingeben.
5. Im Feld **Position**: ist das Default-Verzeichnis für Projekte angegeben.  
Es kann ein anderes Verzeichnis ausgewählt werden: Dazu das Häkchen bei **Standardposition verwenden** entfernen und **Durchsuchen...** wählen. Dann über den Dialog **Ordner suchen** den gewünschten Speicherort wählen und mit **OK** bestätigen.  
Mit **Weiter >** zur nächsten Seite.
6. In der Liste **Topologievorlage** eine Vorlage auswählen.  
Die Vorlage bestimmt, welche Elemente später in der Stationskonfiguration in der Registerkarte **Topologie** vorausgewählt sind. Unabhängig davon, welche Vorlage hier gewählt wird, stehen jedoch in **Topologie** immer alle Elemente zur Verfügung, und die Vorauswahl kann ganz nach Bedarf wieder verändert werden.  
Mit **Weiter >** zur nächsten Seite.
7. Wenn die ausgewählte Vorlage ein Roboter mit Medien-Flansch ist, den passenden Medien-Flansch auswählen.



Das Gewicht und die Höhe des ausgewählten Medien-Flansches werden von der System Software automatisch berücksichtigt.

8. Defaultmäßig ist die Montagerichtung eines am Boden montierten Roboters eingestellt. ( $A=0^\circ$ ,  $B=0^\circ$ ,  $C=0^\circ$ )  
Bei einer Decken- oder Wandmontage des Roboters die Montagerichtung bezogen auf den am Boden montierten Roboter eingeben:
  - a. **Drehung um Z-Achse in ° (A-Winkel)**: Verdrehung des Winkels A um die Z-Achse des Roboterfuß-Koordinatensystems ( $-180^\circ \leq A \leq 180^\circ$ )
  - b. **Drehung um Y-Achse in ° (B-Winkel)**: Verdrehung des Winkels B um die Y-Achse ( $-90^\circ \leq B \leq 90^\circ$ ). Die Verdrehung um die Y-Achse bezieht sich auf das gedrehte Koordinatensystem aus Schritt a.

- c. **Drehung um X-Achse in ° (C-Winkel):** Verdrehung des Winkels C um die X-Achse ( $-180^\circ \leq C \leq 180^\circ$ ). Die Verdrehung um die X-Achse bezieht sich auf das gedrehte Koordinatensystem aus Schritt b.

(>>> 6.7 "Koordinatensysteme" Seite 76)



Die Montagerichtung des Roboters muss korrekt eingegeben werden. Eine falsche eingegebene Montagerichtung kann sich beispielsweise folgendermaßen auswirken:

- Unerwartetes Roboterverhalten in Impedanzregelung
- Veränderte Position bereits geteachter Frames
- Verhinderung der Fahrerlaubnis durch Kollisionserkennung und TCP-Kraftüberwachung
- Unerwartetes Verhalten beim Handverfahren im Welt- oder Basis-Koordinatensystem

Mit **Weiter >** zur nächsten Seite.

9. Eine Zusammenfassung mit Informationen zum Projekt wird angezeigt. Das Häkchen bei **Applikation erstellen (startet weiteren Wizard)** entfernen und auf **Fertigstellen** klicken. Das Projekt wird erstellt und im **Paket-Explorer** eingefügt.

Wenn das Häkchen bei **Applikation erstellen (startet weiteren Wizard)** gesetzt ist, öffnet sich der Assistent für die Applikationserstellung. Es kann direkt eine erste Roboter-Applikation für das eben erstellte Projekt angelegt werden.

(>>> 5.4.2 "Roboter-Applikation mit Paket erstellen" Seite 54)

## Beschreibung

Die Abbildung zeigt die Struktur eines neu erstellten Sunrise-Projekts, in dem noch keine Roboter-Applikationen erstellt oder andere Änderungen vorgenommen wurden. Der für das Sunrise-Projekt konfigurierte Roboter verfügt über einen Medien-Flansch.

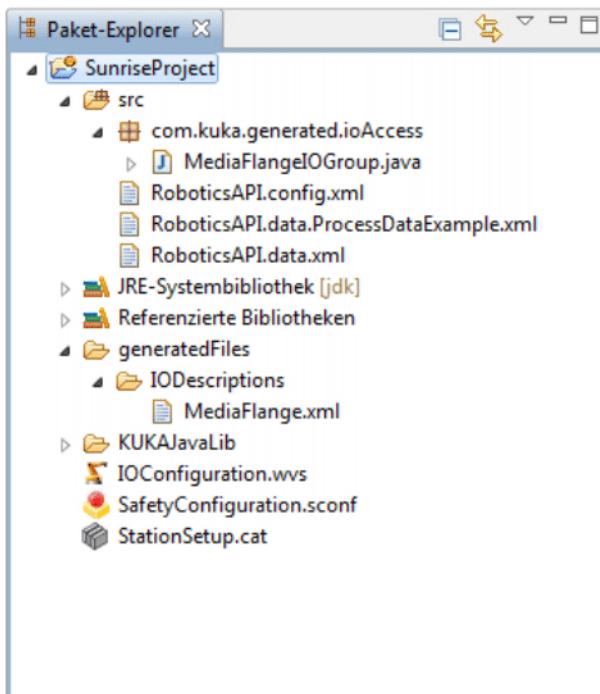


Abb. 5-2: Übersicht Projektstruktur

<b>Element</b>	<b>Beschreibung</b>
<b>src</b>	<p>Quellenordner des Projekts</p> <p>Im Quellenordner werden die erstellten Roboter-Applikationen und Java-Klassen abgelegt.</p> <p>Das Java-Paket <b>com.kuka.generated.ioAccess</b> enthält die Java-Klasse <b>MediaFlangelIOGroup.java</b>. Die Klasse enthält bereits die für die Programmierung benötigten Methoden, um auf die Ein-/Ausgänge des Medien-Flansches zuzugreifen.</p> <p>Weitere Informationen zur Verwendung von Ein-/Ausgängen beim Programmieren sind hier zu finden: (<a href="#">&gt;&gt;&gt; 15.12 "Ein-/Ausgänge" Seite 319</a>)</p> <p>Außerdem sind im Quellenordner verschiedene XML-Dateien enthalten, in denen neben Konfigurationsdaten auch Laufzeitdaten gespeichert werden, beispielsweise die vom Benutzer angelegten Frames und Werkzeuge.</p> <p>Die XML-Dateien können angezeigt, dürfen aber nicht bearbeitet werden.</p>
<b>JRE-Systembibliothek</b>	<p>Java Runtime Environment-Systembibliothek</p> <p>Die Systembibliothek enthält die Java-Klassenbibliotheken, die für die Standard-Java-Programmierung genutzt werden können.</p>
<b>Referenzierte Bibliotheken</b>	<p>Referenzierte Bibliotheken</p> <p>Die referenzierten Bibliotheken können im Projekt genutzt werden. Defaultmäßig werden die roboterspezifischen Java-Klassenbibliotheken beim Anlegen eines Sunrise-Projekts automatisch hinzugefügt. Der Benutzer hat die Möglichkeit, weitere Bibliotheken hinzuzufügen.</p>
<b>generatedFiles</b>	<p>Ordner mit Unterordner <b>IODescriptions</b></p> <p>Die Daten der für den Medien-Flansch konfigurierten Ein-/Ausgänge sind in einer XML-Datei gespeichert.</p> <p>Die XML-Datei kann angezeigt, darf aber nicht bearbeitet werden.</p>
<b>KUKAJavaLib</b>	<p>Ordner mit speziellen Bibliotheken, die zur Roboterprogrammierung benötigt werden.</p>
<b>IOConfiguration.wvs</b>	<p>E/A-Konfiguration für den Medien-Flansch</p> <p>Die E/A-Konfiguration enthält den vollständigen Bus-Aufbau des Medien-Flansches inklusive der E/A-Verschaltung.</p> <p>Die E/A-Konfiguration kann in WorkVisual geöffnet, bearbeitet und wieder in das Sunrise-Projekt exportiert werden.</p> <p><b>Hinweis:</b> Die E/A-Konfiguration wird nur für die am ausgewählten Medien-Flansch vorhandenen Ein- und Ausgänge automatisch vorgenommen. Weitere am Medien-Flansch angeschlossene EtherCAT-Geräte müssen mit WorkVisual konfiguriert werden.</p> <p>(<a href="#">&gt;&gt;&gt; 11 "Buskonfiguration" Seite 165</a>)</p>

Element	Beschreibung
<b>SafetyConfiguration.sconf</b>	Die Datei enthält die von KUKA vorkonfigurierten Sicherheitsfunktionen. Die Konfiguration kann angezeigt und bearbeitet werden.  (>>> 13 "Sicherheitskonfiguration" Seite 183)
<b>StationSetup.cat</b>	Die Datei enthält die Stationskonfiguration für die bei der Projekterstellung ausgewählte Station (Steuerung). Die Konfiguration kann angezeigt und bearbeitet werden.  Über die Stationskonfiguration kann die System Software auf der Robotersteuerung installiert werden.  (>>> 10 "Stationskonfiguration und Installation" Seite 159)



Der Ordner **generatedFiles** wird vom System genutzt und darf nicht für die Ablage eigener Dateien verwendet werden.

## 5.4 Neue Roboter-Applikation erstellen

Roboter-Applikationen sind Java-Programme. Diese beschreiben Aufgaben, die in einer Station ausgeführt werden sollen. Sie werden mit dem Projekt auf die Robotersteuerung übertragen und können über das smartPAD angewählt und ausgeführt werden.

Roboter-Applikationen werden zu Paketen zusammengefasst. Dies macht die Programmierung übersichtlicher und erleichtert die spätere Verwendung eines Pakets in anderen Projekten.

### 5.4.1 Neues Java-Paket erstellen

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. Menüfolge **Datei > Neu > Paket** wählen. Das Fenster **Neues Java-Paket** öffnet sich.
3. Im Feld **Name** einen Namen für das Paket eingeben.
4. Auf **Fertigstellen** klicken. Das Paket wird erstellt und im Ordner "src" des Projekts eingefügt.

Das Paket enthält noch keine Dateien. Ein leeres Paket wird durch ein weißes Paket-Symbol angezeigt. Sobald ein Paket Dateien enthält, ist die Symbolfarbe braun.

### 5.4.2 Roboter-Applikation mit Paket erstellen

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. Menüfolge **Datei > Neu > Roboter-Applikation** wählen. Das Fenster **Neue Roboter-Applikation** öffnet sich.
3. Im Feld **Paket** den Namen des Pakets eingeben, in dem die Applikation erstellt werden soll.
4. Im Feld **Name** einen Namen für die Applikation eingeben.
5. Auf **Fertigstellen** klicken. Applikation und Paket werden erstellt und in das Projekt eingefügt.

Die Applikation *Name.java* wird im Editoren-Bereich geöffnet.

### 5.4.3 Roboter-Applikation für bestehendes Paket erstellen

- Vorgehensweise**
1. Paket im **Paket-Explorer** markieren.
  2. Menüfolge **Datei > Neu > Roboter-Applikation** wählen. Das Fenster **Neue Roboter-Applikation** öffnet sich.
  3. Im Feld **Name** einen Namen für die Applikation eingeben.
  4. Auf **Fertigstellen** klicken. Die Applikation wird erstellt und in das Paket eingefügt.
- Die Applikation *Name.java* wird im Editoren-Bereich geöffnet.

## 5.5 Neuen Hintergrund-Task erstellen

Hintergrund-Tasks sind Java-Programme, die parallel zur Roboter-Applikation auf der Robotersteuerung ausgeführt werden. Sie können beispielsweise Steuerungsaufgaben für Peripheriegeräte erfüllen.

Die Verwendung und Programmierung der Hintergrund-Tasks wird hier beschrieben: ([>>> 16 "Hintergrund-Tasks" Seite 409](#))

Folgende Eigenschaften werden beim Erstellen des Tasks festgelegt:

- Starttyp des Task
  - **Automatisch**  
Der Task wird nach einem Hochlauf der Robotersteuerung automatisch gestartet. (Default)
  - **Manuell**  
Der Task muss manuell über das smartPAD gestartet werden. (Funktion wird zur Zeit noch nicht unterstützt.)
- Task-Vorlage
  - **Zyklischer Hintergrund-Task**  
Vorlage für Tasks, die zyklisch ausgeführt werden sollen (Default)
  - **Nicht-zyklischer Hintergrund-Task**  
Vorlage für Tasks, die einmalig ausgeführt werden sollen

### 5.5.1 Hintergrund-Task mit Paket erstellen

- Vorgehensweise**
1. Projekt im **Paket-Explorer** markieren.
  2. Menüfolge **Datei > Neu > Hintergrund-Task** wählen. Das Fenster **Neuer Hintergrund-Task** öffnet sich.
  3. Im Feld **Paket** den Namen des Pakets eingeben, in dem der Task erstellt werden soll.
  4. Im Feld **Name** einen Namen für den Task eingeben.
  5. Auf **Weiter >** klicken und den Starttyp des Tasks auswählen.
  6. Auf **Weiter >** klicken und die Task-Vorlage auswählen.
  7. Auf **Fertigstellen** klicken. Task und Paket werden erstellt und in das Projekt eingefügt.
- Der Task *Name.java* wird im Editoren-Bereich geöffnet.

### 5.5.2 Hintergrund-Task für bestehendes Paket erstellen

- Vorgehensweise**
1. Paket im **Paket-Explorer** markieren.
  2. Menüfolge **Datei > Neu > Hintergrund-Task** wählen. Das Fenster **Neuer Hintergrund-Task** öffnet sich.
  3. Im Feld **Name** einen Namen für den Task eingeben.

4. Auf **Weiter >** klicken und den Starttyp des Tasks auswählen.
5. Auf **Weiter >** klicken und die Task-Vorlage auswählen.
6. Auf **Fertigstellen** klicken. Der Task wird erstellt und in das Paket eingefügt.  
Der Task *Name.java* wird im Editoren-Bereich geöffnet.

## 5.6 Arbeitsbereich

Das Verzeichnis, in das die angelegten Projekte und benutzerdefinierten Einstellungen von Sunrise.Workbench gespeichert werden, wird als Arbeitsbereich bezeichnet. Das Verzeichnis für den Arbeitsbereich muss beim ersten Start von Sunrise.Workbench durch den Benutzer festgelegt werden. Es können weitere Arbeitsbereiche in Sunrise.Workbench angelegt und zwischen diesen gewechselt werden.

### 5.6.1 Neuen Arbeitsbereich anlegen

- Vorgehensweise**
1. Menüfolge **Datei > Arbeitsbereich wechseln > Andere...** wählen. Das Fenster **Startprogramm für Arbeitsbereich** öffnet sich.
  2. Im Feld **Arbeitsbereich** den Pfad zum neuen Projekt-Verzeichnis manuell eingeben.  
Alternative:
    - Über **Durchsuchen...** zu dem Verzeichnis navigieren, in dem der neue Arbeitsbereich angelegt werden soll.
    - Über **Neuen Ordner erstellen** das neue Projekt-Verzeichnis anlegen. Mit **OK** bestätigen.  
Der Pfad zum neuen Projekt-Verzeichnis wird im Feld **Arbeitsbereich** übernommen.
  3. Neuen Arbeitsbereich mit **OK** bestätigen. Sunrise.Workbench startet neu und der Willkommens-Bildschirm öffnet sich.

### 5.6.2 Zu bestehendem Arbeitsbereich wechseln

- Voraussetzung**
- Es sind weitere Arbeitsbereiche vorhanden.
- Vorgehensweise**
1. Menüfolge **Datei > Arbeitsbereich wechseln > Andere...** wählen. Das Fenster **Startprogramm für Arbeitsbereich** öffnet sich.
  2. Über **Durchsuchen...** zu dem Verzeichnis des gewünschten Arbeitsbereichs navigieren und auswählen.
  3. Mit **OK** bestätigen. Der Pfad zum neuen Projekt-Verzeichnis wird im Fenster **Startprogramm für Arbeitsbereich** übernommen.
  4. Ausgewählten Arbeitsbereich mit **OK** bestätigen. Sunrise.Workbench startet neu und öffnet den ausgewählten Arbeitsbereich.

### 5.6.3 Zwischen zuletzt geöffneten Arbeitsbereichen wechseln

- Voraussetzung**
- Es sind weitere Arbeitsbereiche vorhanden.
- Vorgehensweise**
1. Menüfolge **Datei > Arbeitsbereich wechseln** wählen. Die zuletzt verwendeten Arbeitsbereiche werden in einer Liste angezeigt (maximal 4).
  2. Gewünschten Arbeitsbereich in der Liste auswählen. Sunrise.Workbench startet neu und öffnet den ausgewählten Arbeitsbereich.

#### 5.6.4 Projekte archivieren

**Vorgehensweise**

1. Menüfolge **Datei > Exportieren** wählen. Der Assistent zum Exportieren von Dateien öffnet sich.
2. Im Ordner **Allgemein** die Option **Archivdatei** markieren und auf **Weiter >** klicken.
3. Im linken oberen Bereich sind alle Projekte aufgelistet, die sich im Arbeitsbereich befinden. Projekte auswählen, die archiviert werden sollen (Häkchen in Checkbox setzen).
4. Über **Durchsuchen...** zum gewünschten Speicherort navigieren, den Dateinamen für das Archiv eingeben und auf **Speichern** klicken.
5. Auf **Fertigstellen** klicken. Die Archiv-Datei wird erstellt.

#### 5.6.5 Projekte aus Archiv in Arbeitsbereich laden

**Voraussetzung**

- Archiv-Datei (z. B. eine ZIP-Datei) mit den zu ladenden Projekten ist vorhanden.
- Der Arbeitsbereich enthält kein Projekt mit dem Namen eines zu ladenden Projekts.

**Vorgehensweise**

1. Menüfolge **Datei > Importieren** wählen. Der Assistent zum Importieren von Dateien öffnet sich.
2. Im Ordner **Allgemein** die Option **Vorhandene Objekte in Arbeitsbereich** markieren und auf **Weiter >** klicken.
3. Den Radio-Button **Archivdatei auswählen** aktivieren und über **Durchsuchen...** zur gewünschten Archiv-Datei navigieren und diese markieren.
4. Auf **Öffnen** klicken. Unter **Projekte** werden alle Projekte aufgelistet, die sich im Archiv befinden.
5. Projekte auswählen, die in den Arbeitsbereich geladen werden sollen (Häkchen in Checkbox muss gesetzt sein).
6. Auf **Fertigstellen** klicken. Die ausgewählten Projekte werden geladen.

#### 5.6.6 Projekte aus Verzeichnis in Arbeitsbereich laden

**Voraussetzung**

- Eines oder mehrere Projekte sind in einem beliebigen Verzeichnis vorhanden.
- Der Arbeitsbereich enthält kein Projekt mit dem Namen eines zu ladenden Projekts.

**Vorgehensweise**

1. Menüfolge **Datei > Importieren** wählen. Der Assistent zum Importieren von Dateien öffnet sich.
2. Im Ordner **Allgemein** die Option **Vorhandene Objekte in Arbeitsbereich** markieren und auf **Weiter >** klicken.
3. Den Radio-Button **Stammverzeichnis auswählen** aktivieren und über **Durchsuchen...** zum gewünschten Verzeichnis navigieren und dieses markieren.
4. Auf **OK** klicken. Unter **Projekte** werden alle Projekte aufgelistet, die sich im ausgewählten Verzeichnis befinden.
5. Projekte auswählen, die in den Arbeitsbereich geladen werden sollen (Häkchen in Checkbox muss gesetzt sein).
6. Auf **Fertigstellen** klicken. Die ausgewählten Projekte werden geladen.

## 5.7 Sunrise-Projekte mit referenzierten Java-Projekten

Innerhalb eines Sunrise-Projekts können ein oder mehrere Java-Projekte referenziert werden. Die Referenzierbarkeit von Java-Projekten ermöglicht es, diese in beliebig vielen Sunrise-Projekten und somit auf verschiedenen Robotersteuerungen zu verwenden.

Die referenzierten Java-Projekte können selbst weitere Java-Projekte referenzieren. Unter allen miteinander referenzierten Projekten darf sich nur ein einziges Sunrise-Projekt befinden.



Beim Synchronisieren von Sunrise-Projekten werden referenzierte Java-Projekte mit auf die Robotersteuerung übertragen. Falls innerhalb eines Sunrise-Projekts ein weiteres Sunrise-Projekt referenziert ist, wird die Synchronisation mit einer Fehlermeldung abgebrochen.

### 5.7.1 Neues Java-Projekt erstellen

- Vorgehensweise**
1. Menüfolge **Datei > Neu > Projekt...** wählen. Der Assistent für die Projektgestaltung öffnet sich.
  2. Im Ordner **Java** die Option **Java-Projekt** markieren und auf **Weiter >** klicken.
  3. Im Feld **Projektname:** den Namen des Java-Projekts eingeben.
  4. Im Bereich **JRE** die JRE-Version auswählen, die der JRE-Version des Sunrise-Projekts entspricht. Dies ist in der Regel die JavaSE-1.6.
  5. Auf **Weiter >** klicken und dann auf **Fertigstellen** klicken.
  6. Wenn das erste Mal ein Java-Projekt im Arbeitsbereich erstellt wird oder die Entscheidung bei vorherigen Java-Projekten noch nicht gemerkt wurde, wird eine Abfrage angezeigt, ob die Perspektive **Java** geöffnet werden soll.
    - Je nach Bedarf **Ja** oder **Nein** wählen.
    - Wenn die Abfrage beim nächsten Java-Projekt, das im Arbeitsbereich erstellt wird, nicht mehr angezeigt werden soll, die Option **Entscheidung merken** aktivieren (Häkchen setzen).



In den Java-Projekten müssen alle Klassen, die von außen referenziert werden sollen, in ein definiertes Java-Paket abgelegt werden. Werden referenzierte Klassen im Standardpaket erstellt, können sie im Sunrise-Projekt nicht gefunden werden.

#### 5.7.1.1 Roboterspezifische Klassenbibliotheken in Java-Projekt einfügen

##### Beschreibung

Wenn ein Java-Projekt zur Roboterprogrammierung verwendet wird, müssen die speziellen KUKA Bibliotheken, die dafür benötigt werden, in das Projekt eingefügt werden. Defaultmäßig sind diese Bibliotheken in einem Java-Projekt nicht enthalten.

Die KUKA Bibliotheken müssen aus einem kompatiblen Sunrise-Projekt kopiert werden. Idealerweise ist dies ein Sunrise-Projekt, in dem das Java-Projekt referenziert ist oder wird. Voraussetzung für die Kompatibilität referenzierter Projekte ist, dass die RoboticsAPI-Versionen übereinstimmen.

##### Voraussetzung

- Im Arbeitsbereich ist mindestens ein kompatibles Sunrise-Projekt vorhanden.

##### Vorgehensweise

1. Den Ordner **KUKAJavaLib** eines kompatiblen Sunrise-Projekts kopieren: Im **Paket-Explorer** auf den Ordner rechtsklicken und im Kontextmenü **Kopieren** wählen.

2. Den Ordner **KUKAJavaLib** im Java-Projekt einfügen: Im **Paket-Explorer** auf das gewünschte Java-Projekt rechtsklicken und im Kontextmenü **Einfügen** wählen.
3. Erneut auf das Java-Projekt rechtsklicken und im Kontextmenü **Erstellungspfad** > **Erstellungspfad konfigurieren...** wählen. Das Fenster **Eigenschaften für Projekt** öffnet sich.
4. Im **Java-Erstellungspfad** die Registerkarte **Bibliotheken** auswählen und auf den Button **JARs hinzufügen ...** klicken. Das Fenster **JAR-Auswahl** öffnet sich.
5. Es sind alle Projekte aufgelistet, die sich im Arbeitsbereich befinden. Das Java-Projekt aufklappen, in das die referenzierten Bibliotheken eingefügt werden sollen.
6. Den Ordner **KUKAJavaLib** aufklappen und die vorhandenen JAR-Dateien markieren.
7. Auswahl mit **OK** bestätigen. Die JAR-Dateien werden in die Registerkarte **Bibliotheken** des Erstellungspfads übernommen.
8. Fenster mit **OK** schließen. Die referenzierten Bibliotheken werden in das Java-Projekt eingefügt.

### **5.7.2 Java-Projekte referenzieren**

#### **Voraussetzung**

- Die referenzierten Klassen sind in einem definierten Java-Paket abgelegt (nicht im Standardpaket).
- Für Java-Projekte, die referenzierte KUKA Bibliotheken verwenden: In den referenzierten Projekten müssen die RoboticsAPI-Versionen übereinstimmen.

#### **Vorgehensweise**

1. Im **Paket-Explorer** auf das Projekt rechtsklicken, für das Java-Projekte referenziert werden sollen.
2. Im Kontextmenü **Erstellungspfad** > **Erstellungspfad konfigurieren...** wählen. Das Fenster **Eigenschaften für Projekt** öffnet sich.
3. Im **Java-Erstellungspfad** die Registerkarte **Projekte** auswählen und auf den Button **Hinzufügen ...** klicken. Das Fenster **Erforderliche Projekt-auswahl** öffnet sich.
4. Es sind alle Projekte aufgelistet, die sich im Arbeitsbereich befinden. Java-Projekte auswählen, die referenziert werden sollen (Häkchen in Checkbox setzen).
5. Auswahl mit **OK** bestätigen. Die ausgewählten Projekte werden in die Registerkarte **Projekte** des Erstellungspfads übernommen.
6. Fenster mit **OK** schließen.

### **5.7.3 Referenzierung zu Java-Projekten aufheben**

#### **Beschreibung**

Referenzierungen zu fehlerhaft hinzugefügten oder nicht (mehr) benötigten Projekten können entfernt werden.

#### **Vorgehensweise**

1. Im **Paket-Explorer** auf das Projekt rechtsklicken, aus dem referenzierte Projekte entfernt werden sollen.
2. Im Kontextmenü **Eigenschaften** wählen. Das Fenster **Eigenschaften für Projekt** öffnet sich.
3. Im **Java-Erstellungspfad** die Registerkarte **Projekte** auswählen.
4. Die nicht benötigten Projekte markieren und auf **Entfernen** klicken.
5. Fenster mit **OK** schließen.

## 5.8 Element im Paket-Explorer umbenennen

In der Sicht **Paket-Explorer** können die Namen eingefügter Elemente geändert werden, z. B. die Namen von Projekten, Java-Paketen und Java-Dateien.

### 5.8.1 Projekt oder Java-Paket umbenennen

- |                       |  |
|-----------------------|--|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"><li>1. Auf das gewünschte Projekt oder Java-Paket rechtsklicken. Im Kontextmenü <b>Refactoring &gt; Umbenennen</b> wählen. Das Fenster <b>Java-Projekt umbenennen</b> oder <b>Java-Paket umbenennen</b> öffnet sich.</li><li>2. Im Feld <b>Neuer Name</b> den gewünschten Namen eintragen. Mit <b>OK</b> bestätigen.</li></ol> |
|-----------------------|--|

### 5.8.2 Java-Datei umbenennen

- |                       |   |
|-----------------------|---|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"><li>1. Auf die gewünschte Java-Datei rechtsklicken. Im Kontextmenü <b>Refactoring &gt; Umbenennen</b> wählen. Das Fenster <b>Kompilierseinheit umbenennen</b> öffnet sich.</li><li>2. Im Feld <b>Neuer Name</b> den gewünschten Namen eintragen. Auf <b>Fertigstellen</b> klicken.</li><li>3. Vor dem Fertigstellen wird auf mögliche Konflikte hingewiesen. Wurden diese zur Kenntnis genommen und überprüft, erneut auf <b>Fertigstellen</b> klicken.</li></ol> |
|-----------------------|---|

## 5.9 Element aus Paket-Explorer entfernen

In der Sicht **Paket-Explorer** können eingefügte Elemente wieder entfernt werden, z. B. ganze Projekte oder einzelne Java-Pakete und Java-Dateien eines Projekts.

### 5.9.1 Element aus Projekt löschen

- |                     |   |
|---------------------|---|
| <b>Beschreibung</b> | Für ein Projekt erstellte Elemente können wieder gelöscht werden. Die Elemente werden dauerhaft aus dem Arbeitsbereich gelöscht und können nicht wiederhergestellt werden.<br><br>Manche defaultmäßig vorhandenen Elemente eines Projekts können ebenfalls gelöscht werden, andere nicht. |
|---------------------|---|

- |                       |   |
|-----------------------|---|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"><li>1. Auf das Element rechtsklicken. Im Kontextmenü <b>Löschen</b> wählen.</li><li>2. Sicherheitsabfrage mit <b>OK</b> beantworten. Das Element wird gelöscht.</li></ol> |
|-----------------------|---|

### 5.9.2 Projekt im Paket-Explorer entfernen

- |                     |  |
|---------------------|--|
| <b>Beschreibung</b> | Mit dieser Vorgehensweise wird ein Projekt nur im <b>Paket-Explorer</b> entfernt und bleibt im Verzeichnis für den Arbeitsbereich auf dem Datenträger erhalten.<br><br>Bei Bedarf kann das Projekt wieder aus dem Verzeichnis in den Arbeitsbereich geladen werden. Das Projekt steht dann wieder im <b>Paket-Explorer</b> zur Verfügung.<br><br>(>>> 5.6.6 "Projekte aus Verzeichnis in Arbeitsbereich laden" Seite 57) |
|---------------------|--|

- |                       |   |
|-----------------------|---|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"><li>1. Auf das gewünschte Projekt rechtsklicken. Im Kontextmenü <b>Löschen</b> wählen. Es wird abgefragt, ob das Projekt wirklich gelöscht werden soll.</li></ol> |
|-----------------------|---|

2. Bei **Projektinhalt auf dem Datenträger löschen (kann nicht rückgängig gemacht werden)** ist defaultmäßig kein Häkchen gesetzt. So belassen.
3. Abfrage mit **OK** bestätigen.

### 5.9.3 Projekt aus Arbeitsbereich löschen

**Beschreibung** Mit dieser Vorgehensweise wird ein Projekt im **Paket-Explorer** entfernt und dauerhaft aus dem Verzeichnis für den Arbeitsbereich auf dem Datenträger gelöscht. Das Projekt kann nicht wiederhergestellt werden.

- Vorgehensweise**
1. Auf das gewünschte Projekt rechtsklicken. Im Kontextmenü **Löschen** wählen. Es wird abgefragt, ob das Projekt wirklich gelöscht werden soll.
  2. Bei **Projektinhalt auf dem Datenträger löschen (kann nicht rückgängig gemacht werden)** ein Häkchen setzen.
  3. Abfrage mit **OK** bestätigen.

## 5.10 Automatische Änderungserkennung aktivieren

**Beschreibung** Die automatische Änderungserkennung in Sunrise.Workbench ist defaultmäßig aktiv. Wurde sie deaktiviert, führt dies beispielsweise dazu, dass beim Exportieren einer E/A-Konfiguration aus WorkVisual die für die Verwendung der Signale erforderlichen Java-Klassen und Dateien im Sunrise-Projekt nicht erstellt werden.

- Vorgehensweise**
1. Menüfolge **Fenster > Benutzervorgaben** wählen. Das Fenster **Benutzervorgaben** öffnet sich.
  2. In der Verzeichnisstruktur im linken Bereich des Fensters **Allgemein > Arbeitsbereich** wählen.
  3. Das Häkchen bei **Aktualisieren durch native Hooks oder Polling** setzen, um die automatische Änderungserkennung zu aktivieren.



## 6 Bedienung KUKA smartPAD

### 6.1 Bedienhandgerät KUKA smartPAD

#### 6.1.1 Vorderseite

##### Funktion

Das smartPAD ist das Bedienhandgerät für den Industrieroboter. Das smartPAD hat alle Bedien- und Anzeigemöglichkeiten, die für die Bedienung benötigt werden.

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

##### Übersicht

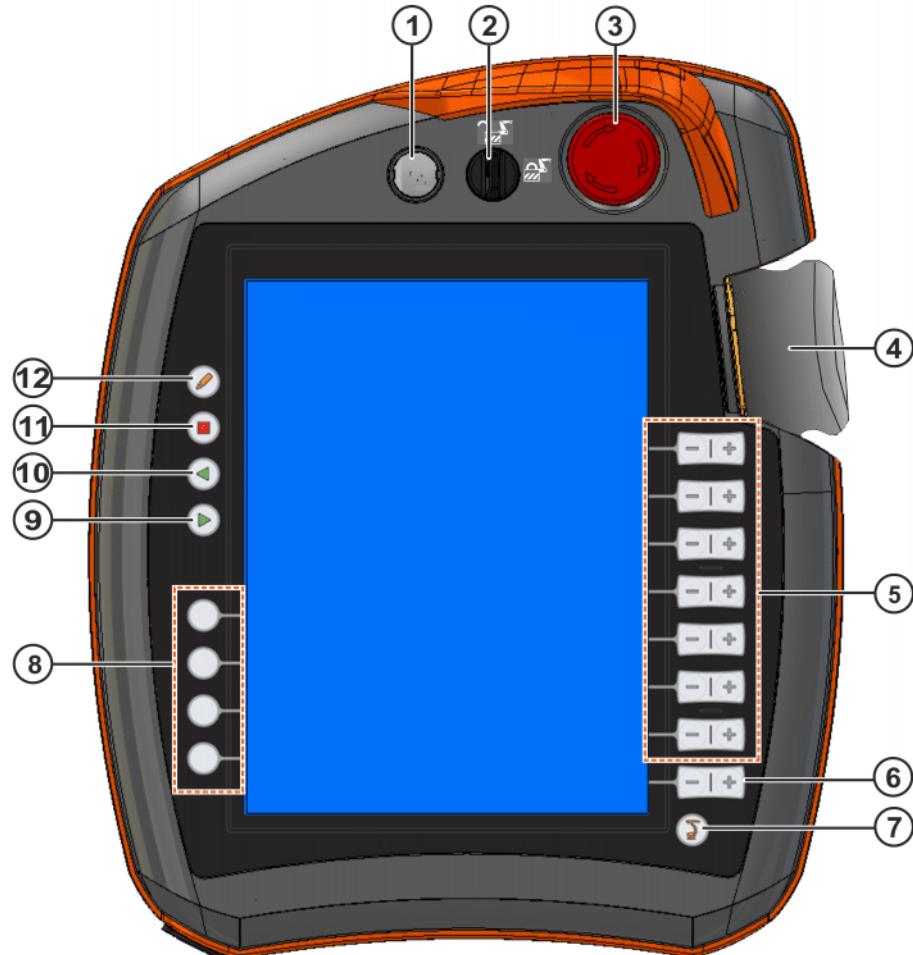
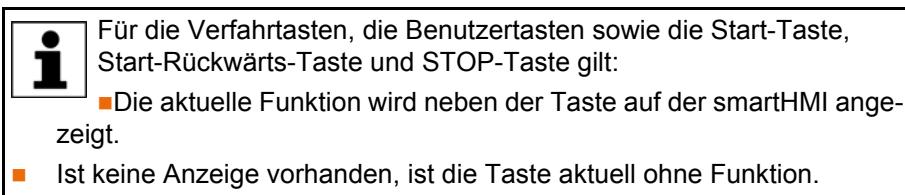


Abb. 6-1: KUKA smartPAD Vorderseite

Pos.	Beschreibung
1	Knopf zum Abstecken des smartPADs Zur Zeit ohne Funktion
2	Schlüsselschalter  Mit dem Schlüsselschalter wird der Verbindungs-Manager aufgerufen. Der Schalter kann nur umgelegt werden, wenn der Schlüssel steckt.  Über den Verbindungs-Manager kann die Betriebsart gewechselt werden.  (>>> 6.6 "Betriebsart wechseln" Seite 74)
3	NOT-HALT-Einrichtung  Mit der NOT-HALT-Einrichtung wird der Roboter in Gefahrensituationen gestoppt. Die NOT-HALT-Einrichtung verriegelt sich, wenn sie gedrückt wird.
4	Space Mouse  Zur Zeit ohne Funktion
5	Verfahrtasten  Mit den Verfahrtasten wird der Roboter manuell verfahren.  (>>> 6.8 "Roboter manuell verfahren" Seite 78)
6	Taste zum Einstellen des Hand-Overrides
7	Hauptmenü-Taste  Die Hauptmenü-Taste blendet das Hauptmenü auf der smartHMI ein und aus.
8	Benutzertasten  Die Funktion der Benutzertasten ist frei programmierbar. Die Benutzertasten können beispielsweise dazu verwendet werden, Peripheriegeräte zu steuern oder applikationsspezifische Aktionen auszulösen.  (>>> 15.26 "Benutzertasten definieren" Seite 380)
9	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.  (>>> 6.14 "Programmausführung" Seite 90) (>>> 6.13.4 "Frames manuell anfahren" Seite 89) (>>> 6.14.6 "Roboter nach Verlassen der Bahn rückpositionieren" Seite 94)
10	Start-Rückwärts-Taste  Zur Zeit ohne Funktion
11	STOP-Taste  Mit der STOP-Taste wird eine laufende Applikation angehalten.
12	Tastatur-Taste  Zur Zeit ohne Funktion



## 6.1.2 Rückseite

## Übersicht



Abb. 6-2: KUKA smartPAD Rückseite

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

<b>Beschreibung</b>	<b>Element</b>	<b>Beschreibung</b>
	<b>Typenschild</b>	Typenschild
	<b>Start-Taste</b>	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.

Element	Beschreibung
Zustimmungs-schalter	<p>Der Zustimmungsschalter hat 3 Stellungen:</p> <ul style="list-style-type: none"> <li>■ Nicht gedrückt</li> <li>■ Mittelstellung</li> <li>■ Durchgedrückt</li> </ul> <p>Der Zustimmungsschalter muss in den Betriebsarten T1, T2 und KRF in der Mittelstellung gehalten werden, damit der Manipulator verfahren kann.</p> <p>In der Betriebsart Automatik hat der Zustimmungsschalter defaultmäßig keine Funktion.</p>
USB-An-schluss	<p>Der USB-Anschluss wird z. B. für die Archivierung ver-wendet.</p> <p>Nur für FAT32 formatierte USB-Sticks.</p>

## 6.2 Robotersteuerung ein-/ausschalten

### 6.2.1 Robotersteuerung einschalten und System Software starten

- Vorgehensweise**
- Hauptschalter an der Robotersteuerung in Stellung "I" bringen.  
Die System Software startet automatisch.

Die Robotersteuerung ist betriebsbereit, wenn die Statusanzeige, die den Boot-Zustand der Robotersteuerung anzeigt, grün leuchtet (Stationssicht / Kachel **KUKA\_Sunrise\_Cabinet**).

### 6.2.2 Robotersteuerung ausschalten

- Vorgehensweise**
- Hauptschalter an der Robotersteuerung in Stellung "0" bringen.

**HINWEIS** Wenn beim Ausschalten der Robotersteuerung noch eine Applikation läuft, werden laufende Bewegungen gestoppt. Dies kann zu einer Beschädigung des Roboters führen. Die Robotersteuerung darf daher erst ausgeschalten werden, wenn keine Applikation mehr läuft und der Roboter stillsteht.

## 6.3 Automatisches Update der smartPAD-Software

Bei einem Neustart der Robotersteuerung sowie beim Anstecken des smartPADs an eine laufende Robotersteuerung wird automatisch eine Versionsprüfung der smartPAD-Software durchgeführt. Bei Konflikten zwischen der smartPAD-Software und der System Software auf der Robotersteuerung ist ein Update der smartPAD-Software erforderlich.

Eigenschaften des Updates der smartPAD-Software:

- In den Betriebsarten T1, T2 und KRF wird das Update automatisch durchgeführt.
- In der Betriebsart Automatik ist kein automatisches Update der smartPAD-Software möglich.

Wird beim Anstecken des smartPADs in der Betriebsart Automatik ein Versionskonflikt erkannt, wird das smartPAD für Benutzereingaben gesperrt. In diesem Fall in die Betriebsart T1 oder T2 wechseln, um das automatische Update zu ermöglichen.

- Während des Updates ist das smartPAD für Benutzereingaben gesperrt.

**HINWEIS** Das Update darf nicht unterbrochen werden, da sonst das smartPAD beschädigt werden kann. Es muss darauf geachtet werden, dass das smartPAD während des Updates nicht von der Robotersteuerung getrennt oder die Robotersteuerung von der Stromversorgung getrennt wird.

- Nach dem Update wird das smartPAD automatisch neu gestartet. Der obere Bereich der smartHMI wird durch einen Balken verdeckt.

**i** Nach dem automatischen Update muss die Robotersteuerung neu gestartet werden, um die smartHMI vollständig anzuzeigen und das System nutzen zu können.

## 6.4 Bedienoberfläche KUKA smartHMI

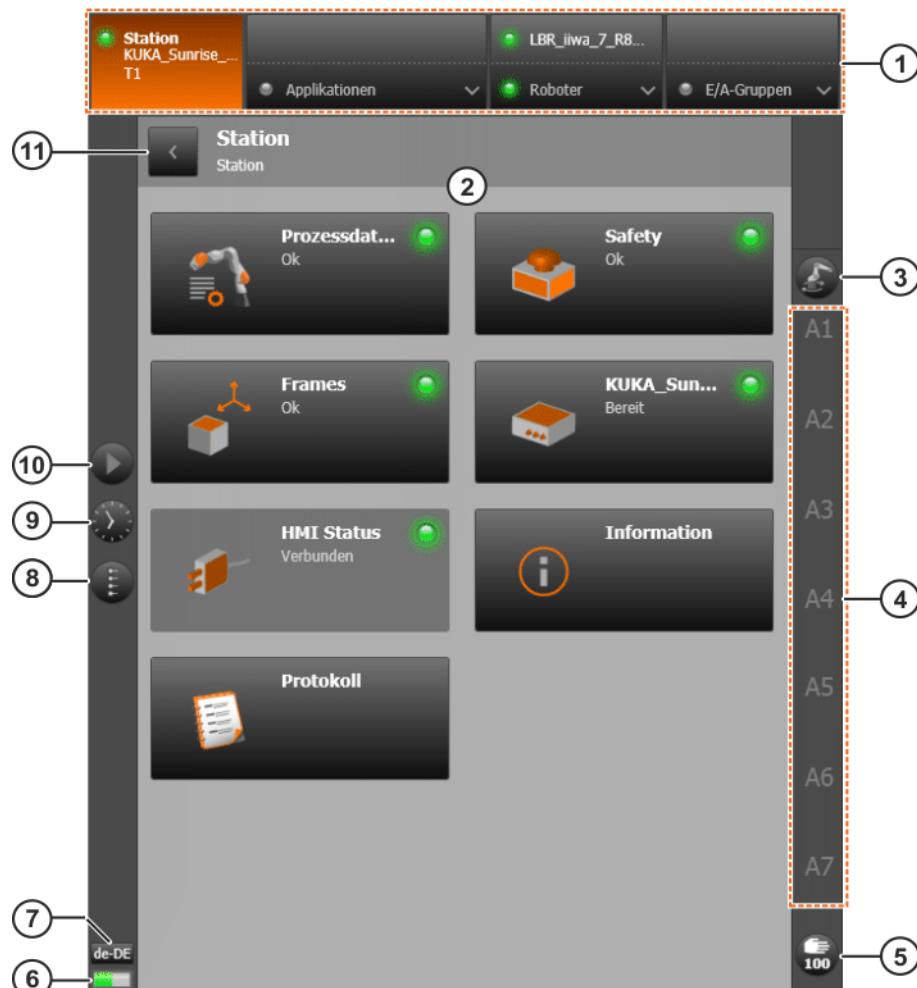


Abb. 6-3: Bedienoberfläche KUKA smartHMI

Pos.	Beschreibung
1	Navigationsleiste: Hauptmenü und Statusanzeige (>>> 6.4.1 "Navigationsleiste" Seite 68)
2	Anzeige-Bereich Anzeige der in der Navigationsleiste ausgewählten Ebene, hier die Stationssicht

Pos.	Beschreibung
3	<p><b>Button Handverfahroptionen</b></p> <p>Zeigt das aktuelle Koordinatensystem für das manuelle Verfahren mit den Verfahrtasten an. Das Berühren des Buttons öffnet das Fenster <b>Handverfahroptionen</b>, in dem das Bezugs-Koordinatensystem und weitere Parameter für das manuelle Verfahren eingestellt werden können.</p> <p>(&gt;&gt;&gt; 6.8.1 "Fenster Handverfahroptionen" Seite 78)</p>
4	<p><b>Anzeige Verfahrtasten</b></p> <p>Wenn das achsspezifische Verfahren ausgewählt ist, werden hier die Achsnummern angezeigt (A1, A2 etc.). Wenn das kartesische Verfahren ausgewählt ist, werden hier die Richtungen des Koordinatensystems angezeigt (X, Y, Z, A, B, C) sowie der Ellenbogen-Winkel (R) für das Ausführen einer Nullraum-Bewegung.</p> <p>(&gt;&gt;&gt; 6.8 "Roboter manuell verfahren" Seite 78)</p>
5	<p><b>Button HOV</b></p> <p>Zeigt den aktuellen Hand-Override an. Das Berühren des Buttons öffnet das Fenster <b>Handverfahrgeschwindigkeit</b>, in dem der Hand-Override eingestellt werden kann.</p> <p>(&gt;&gt;&gt; 6.8.2 "Hand-Override (HOV) einstellen" Seite 80)</p>
6	<p><b>Anzeige Lebenszeichen</b></p> <p>Ein gleichmäßig blinkendes Lebenszeichen zeigt an, dass die smartHMI aktiv ist.</p>
7	<p><b>Button Sprachauswahl</b></p> <p>Das Berühren des Buttons öffnet das Menü <b>Sprachauswahl</b>, in dem die Anzeigesprache der smartHMI geändert werden kann.</p>
8	<p><b>Button Auswahl Benutzertasten</b></p> <p>Das Berühren des Buttons öffnet das Fenster <b>Auswahl Benutzertasten</b>, in dem die aktuell verfügbaren Benutzertasten-Leisten ausgewählt werden können.</p> <p>(&gt;&gt;&gt; 6.15 "Benutzertasten aktivieren" Seite 95)</p>
9	<p><b>Button Uhr</b></p> <p>Die Uhr zeigt die Systemzeit an. Das Berühren des Buttons blendet die Systemzeit in digitaler Darstellung sowie das aktuelle Datum ein.</p>
10	<p><b>Button Verfahrtart</b></p> <p>Zeigt den aktuell eingestellten Modus der Start-Taste an. Das Berühren des Buttons öffnet das Fenster <b>Verfahrtart</b>, in dem der Modus geändert werden kann.</p> <p>(&gt;&gt;&gt; 6.13.3 "Fenster Verfahrtart" Seite 88)</p>
11	<p><b>Zurück-Button</b></p> <p>Über diesen Button kann man zur vorherigen Ansicht zurückkehren.</p>

#### 6.4.1 Navigationsleiste

Die Navigationsleiste ist das Hauptmenü der Bedienoberfläche und in 4 Ebenen aufgeteilt. Sie dient zur Navigation zwischen den verschiedenen Ebenen. Einige der Ebenen sind zweigeteilt:

- Auswahlliste unten: Öffnet eine Liste, über die je nach Ebene eine Applikation, ein Roboter oder eine E/A-Gruppe ausgewählt werden kann.
- Button oben: Wurde über die Liste eine Auswahl getroffen, kann über diesen Button die entsprechende Applikationssicht, Robotersicht oder E/A-Gruppe angezeigt werden.

Alternativ kann das Hauptmenü über die Hauptmenü-Taste auf dem smartPAD aufgerufen werden. Dieses enthält weitere Menüs, die nicht über die Menüführung der Navigationsleiste ausgewählt werden können.

(>>> 6.5 "Hauptmenü aufrufen" Seite 73)

## Übersicht



Abb. 6-4: KUKA smartHMI Navigationsleiste

Pos.	Beschreibung
1	Ebene <b>Station</b> Anzeige des Steuerungsnamens und der gewählten Betriebsart (>>> 6.4.4 "Stationssicht" Seite 70)
2	Ebene <b>Applikationen</b> Anzeige der angewählten Applikation (>>> 6.14.1 "Roboter-Applikation anwählen" Seite 90)
3	Ebene <b>Roboter</b> Anzeige des ausgewählten Roboters (>>> 6.4.5 "Robotersicht" Seite 72)
4	Ebene <b>E/A-Gruppen</b> Anzeige der ausgewählten E/A-Gruppe (>>> 6.16.5 "E/A-Gruppe anzeigen und Wert eines Ausgangs ändern" Seite 99)

### 6.4.2 Statusanzeige

Der Status der Systemkomponenten ist auf der smartHMI durch farbige Kreise gekennzeichnet.

In der Navigationsleiste (>>> Abb. 6-4 ) wird im unteren Teil der Leiste der "Sammelstatus" angezeigt. Im oberen Teil wird jeweils der Status der gewählten Komponente angezeigt. Es kann z. B. eine Applikation ausgeführt werden und sich gleichzeitig eine andere Applikation im Fehlerzustand befinden."

Status	Beschreibung
	<p>Schwerwiegender Fehler</p> <p>Die Systemkomponente kann nicht genutzt werden. Grund dafür kann sowohl ein Bedienfehler als auch ein Fehler der Systemkomponente sein.</p>
	<p>Warnung</p> <p>Für die Systemkomponente liegt ein Warnhinweis vor. Die Betriebsfähigkeit der Komponente kann eingeschränkt sein. Es wird daher empfohlen, die Ursache zu beheben.</p> <p>Bei Applikationen bedeutet die gelbe Statusanzeige, dass die Applikation pausiert ist.</p>
	<p>Status i.O.</p> <p>Für die Systemkomponente liegen weder Warnungen noch Störungen vor.</p>
	<p>Status unbekannt</p> <p>Über den Status der Systemkomponente ist keine Aussage möglich.</p>

### 6.4.3 Tastatur

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



**Abb. 6-5: Beispiel Tastatur**



**SYM** muss gedrückt werden, um die Zweitbelegungen der Tasten aktivieren, z. B. das "=" -Zeichen auf der S-Taste. Die Taste ist für einen Tastendruck selbsthaltend. Sie braucht also nicht gedrückt gehalten werden.

#### 6.4.4 Stationssicht

Die Stationssicht bietet den Zugriff auf Informationen und Funktionalitäten, die die gesamte Station betreffen.



Abb. 6-6: Stationssicht

Pos.	Beschreibung
1	<b>Kachel Prozessdaten</b> Hier werden Informationen zu den Prozessdaten angezeigt. Die Konfiguration von Prozessdaten ist aktuell noch nicht möglich.
2	<b>Kachel Safety</b> Öffnet die Ansicht <b>Safety</b> und zeigt den Sicherheitsstatus der Station an. <i>(&gt;&gt;&gt; 13.7 "Sicherheitskonfiguration auf Robotersteuerung aktivieren" Seite 205)</i>
3	<b>Kachel Frames</b> Öffnet die Frames-Sicht. Die Sicht enthält die für die Station angelegten Frames aus dem Sunrise-Projekt. <i>(&gt;&gt;&gt; 6.13 "Frames teachen und manuell anfahren" Seite 85)</i>
4	<b>Kachel KUKA_Sunrise_Cabinet</b> Zeigt den Zustand der Robotersteuerung an und öffnet eine Ansicht, die die Kacheln <b>Bootzustand</b> und <b>Feldbusse</b> enthält. Die Kachel <b>Bootzustand</b> zeigt den Boot-Zustand der Robotersteuerung an. Die Kachel <b>Feldbusse</b> zeigt den Zustand der Feldbusse an. Sie wird nur angezeigt, wenn E/A-Gruppen für das Sunrise-Projekt angelegt und zugehörige Signale mit WorkVisual verschaltet wurden.
5	<b>Kachel HMI Status</b> Zeigt den Verbindungsstatus zwischen smartHMI und Robotersteuerung an.

Pos.	Beschreibung
6	<b>Kachel Information</b> Zeigt Systeminformationen an, z. B. die IP-Adresse der Robotersteuerung.
7	<b>Kachel Protokoll</b> Öffnet die Ansicht <b>Protokoll</b> und zeigt die protokollierten Ereignisse und Zustandsänderungen des Systems an. Die Anzeige kann anhand unterschiedlicher Kriterien gefiltert werden. (>>> 18.2 "Protokoll anzeigen" Seite 451)

#### 6.4.5 Robotersicht

Die Robotersicht bietet Zugriff auf Informationen und Funktionalitäten, die den ausgewählten Roboter betreffen.

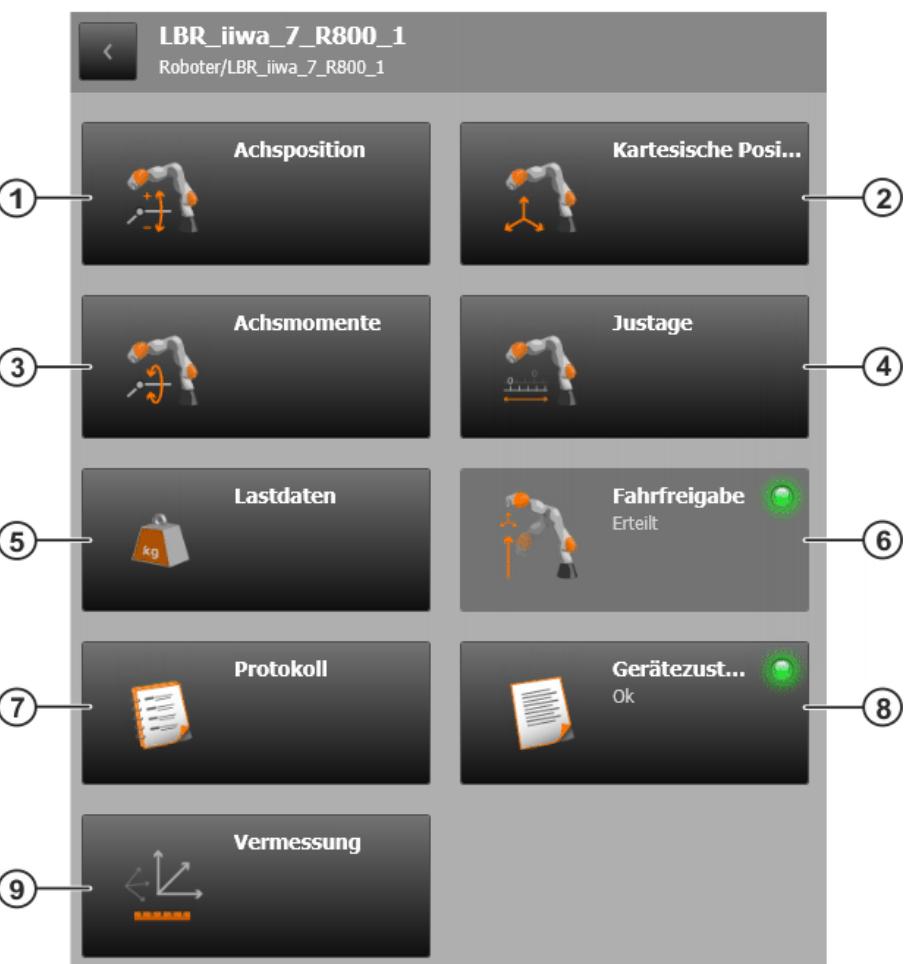


Abb. 6-7: Robotersicht

<b>Pos.</b>	<b>Beschreibung</b>
1	<p><b>Kachel Achsposition</b></p> <p>Öffnet die Ansicht <b>Achsposition</b>. Die achsspezifische Istposition des Roboters wird angezeigt.</p> <p>(&gt;&gt;&gt; 6.16.2 "Achsspezifische Istposition anzeigen" Seite 97)</p>
2	<p><b>Kachel Kartesische Position</b></p> <p>Öffnet die Ansicht <b>Kartesische Position</b>. Die kartesische Istposition des Roboters wird angezeigt.</p> <p>(&gt;&gt;&gt; 6.16.3 "Kartesische Istposition anzeigen" Seite 98)</p>
3	<p><b>Kachel Achsmomente</b></p> <p>Öffnet die Ansicht <b>Achsmomente</b>. Die Achsmomente des Roboters werden angezeigt.</p> <p>(&gt;&gt;&gt; 6.16.4 "Achsspezifische Momente anzeigen" Seite 98)</p>
4	<p><b>Kachel Justage</b></p> <p>Öffnet die Ansicht <b>Justage</b>. Der Justagestatus der Roboterachsen wird angezeigt. Die Achsen können einzeln justiert oder dejustiert werden.</p> <p>(&gt;&gt;&gt; 7.1 "Positionsjustage" Seite 103)</p>
5	<p><b>Kachel Lastdaten</b></p> <p>Öffnet die Ansicht <b>Lastdaten</b> für die automatische Lastdatenermittlung.</p> <p>(&gt;&gt;&gt; 7.3 "Werkzeug-Lastdaten ermitteln" Seite 112)</p>
6	<p><b>Kachel Fahrfreigabe</b></p> <p>Es wird angezeigt, ob der Roboter die Fahrfreigabe erhalten hat.</p>
7	<p><b>Kachel Protokoll</b></p> <p>Öffnet die Ansicht <b>Protokoll</b> und zeigt die protokollierten Ereignisse und Zustandsänderungen des Systems an. Die Anzeige kann anhand unterschiedlicher Kriterien gefiltert werden. Defaultmäßig ist der Filter <b>Quelle(n)</b> bereits auf den betreffenden Roboter gesetzt.</p> <p>(&gt;&gt;&gt; 18.2 "Protokoll anzeigen" Seite 451)</p>
8	<p><b>Kachel Gerätezustand</b></p> <p>Der Status des Antriebssystems des Roboters wird angezeigt.</p>
9	<p><b>Kachel Vermessung</b></p> <p>Öffnet die Ansicht <b>Vermessung</b>, die die Kacheln <b>Basisvermessung</b> und <b>Werkzeugvermessung</b> enthält.</p> <p>(&gt;&gt;&gt; 7.2 "Vermessen" Seite 104)</p>

## 6.5 Hauptmenü aufrufen

**Vorgehensweise** ■ Hauptmenü-Taste auf dem smartPAD drücken. Die Ansicht **Hauptmenü** öffnet sich.

**Beschreibung** Eigenschaften Ansicht **Hauptmenü**:

- In der linken Spalte wird das Hauptmenü angezeigt.
- Die ersten 4 Buttons von oben aus gesehen sind identisch mit den Ebenen der Navigationsleiste.

- Das Berühren eines Buttons mit Pfeil blendet die zur Ebene gehörenden Bereiche ein, z. B. **Station**.  
Die weiteren Navigationsmöglichkeiten sind in der folgenden Tabelle beschrieben.



**Abb. 6-8: Beispielansicht Haupts menü**

Pos.	Beschreibung
1	Zurück-Button Über diesen Button kehrt man zur Ansicht zurück, die vor dem Öffnen des Haupts menüs sichtbar war.
2	Home-Button Blendet alle geöffneten Bereiche aus.
3	Button zum Schließen der Ebene Blendet die tiefste geöffnete Ebene aus.
4	Hier werden die zuletzt über das Haupts menü aufgerufenen Ansichten angezeigt (maximal 3). Durch Berühren der jeweiligen Anzeige kann direkt in diese Ansichten gewechselt werden, ohne durch das Haupts menü navigieren zu müssen.

## 6.6 Betriebsart wechseln

### Beschreibung

Die Betriebsart kann über den Verbindungs-Manager mit dem smartPAD eingestellt werden.



Es ist möglich die Betriebsart zu wechseln, wenn auf der Robotersteuerung eine Applikation läuft. Der Industrieroboter stoppt dann mit einem Sicherheitshalt 1 und die Applikation wird pausiert. Wenn die neue Betriebsart eingestellt ist, kann die Applikation fortgesetzt werden.

### Voraussetzung

- Schlüssel steckt im Schalter zum Aufrufen des Verbindungs-Managers

### Vorgehensweise

1. Am smartPAD den Schalter für den Verbindungs-Manager nach rechts drehen. Der Verbindungs-Manager wird angezeigt.
2. Die Betriebsart wählen.
3. Den Schalter für den Verbindungs-Manager nach links drehen.  
Die gewählte Betriebsart ist jetzt aktiv und wird in der Navigationsleiste der smartHMI angezeigt.

Betriebsart	Verwendung	Geschwindigkeiten
T1	Programmieren, Teachen und Testen von Programmen	<ul style="list-style-type: none"> <li>■ Programmverifikation: Reduzierte programmierte Geschwindigkeit, maximal 250 mm/s</li> <li>■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s</li> <li>■ Handführen: Keine Begrenzung der Geschwindigkeit, sondern sicherheitsgerichtete Geschwindigkeitsüberwachung gemäß Sicherheitskonfiguration</li> </ul> <p><b>Hinweis:</b> Bei einer mobilen Plattform gilt die maximale Geschwindigkeit von 250 mm/s nicht.</p>
T2	Testen von Programmen  Nur mit geschlossener Schutztür möglich	<ul style="list-style-type: none"> <li>■ Programmverifikation: Programmierte Geschwindigkeit</li> <li>■ Handbetrieb: Nicht möglich</li> </ul>

Betriebsart	Verwendung	Geschwindigkeiten
AUT	Automatisches Ausführen von Programmen  Für Industrieroboter mit und ohne übergeordnete Steuerung	<ul style="list-style-type: none"> <li>■ Programmabetrieb: Programmierte Geschwindigkeit</li> <li>■ Handbetrieb: Nicht möglich</li> </ul>
KRF	<ul style="list-style-type: none"> <li>■ Herausfahren des Industrieroboters aus einem verletzten kartesischen oder achsspezifischen Bereich</li> <li>■ Herausfahren des Industrieroboters aus einem verletzten Bereich der Werkzeugorientierung</li> <li>■ Freifahren des Industrieroboters aus Klemmsituationen bei Verletzung von Kraft- oder Momentengrenzen</li> <li>■ Verfahren des Industrieroboters, wenn bei aktiver kartesischer Geschwindigkeitsüberwachung ein Justageverlust für mindestens einen Positionssensor vorliegt</li> </ul> <p>KRF ist eine Betriebsart, die zur Verfügung steht, wenn der Industrieroboter von der Sicherheitssteuerung aufgrund einer der folgenden Ursachen gestoppt wird:</p> <ul style="list-style-type: none"> <li>■ Industrieroboter verletzt einen sicher überwachten Raum.</li> <li>■ Orientierung des sicherheitsgerichteten Werkzeugs liegt außerhalb des sicher überwachten Bereichs.</li> <li>■ Industrieroboter verletzt eine sicher überwachte Kraft- oder Momentengrenze.</li> <li>■ Ein Positionssensor ist bei aktiver kartesischer Geschwindigkeitsüberwachung nicht justiert.</li> </ul>	<ul style="list-style-type: none"> <li>■ Programmabetrieb: Reduzierte programmierte Geschwindigkeit, maximal 250 mm/s</li> <li>■ Handbetrieb: Handverfahrgeschwindigkeit, maximal 250 mm/s</li> </ul>

## 6.7 Koordinatensysteme

Koordinatensysteme oder Frames bestimmen die Position und Orientierung eines Objekts im Raum.

### Übersicht

Für die Robotersteuerung sind folgende Koordinatensysteme relevant:

- Welt
- Roboterfuß
- Basis
- Flansch
- Werkzeug

### Beschreibung

#### Welt-Koordinatensystem

Das Welt-Koordinatensystem ist ein fest definiertes kartesisches Koordinatensystem. Es ist das Ursprungs-Koordinatensystem für alle anderen Koordinatensysteme, insbesondere für Basis-Koordinatensysteme und das Roboterfuß-Koordinatensystem.

Defaultmäßig liegt das Welt-Koordinatensystem im Roboterfuß.

### **Roboterfuß-Koordinatensystem**

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

Defaultmäßig ist das Roboterfuß-Koordinatensystem mit dem Welt-Koordinatensystem deckungsgleich. Es kann eine Verschiebung des Roboters zum Welt-Koordinatensystem definiert werden, indem man beim Erstellen des Sunrise-Projekts die Montagerichtung ändert. Defaultmäßig ist die Montagerichtung des am Boden montierten Roboters eingestellt. ( $A=0^\circ$ ,  $B=0^\circ$ ,  $C=0^\circ$ ).

(>>> 5.3 "Sunrise-Projekt mit Vorlage erstellen" Seite 51)

### **Basis-Koordinatensystem**

Um Bewegungen im kartesischen Raum zu definieren, ist die Angabe eines Bezugs-Koordinatensystems (Basis) nötig.

Defaultmäßig wird das Welt-Koordinatensystem als Basis-Koordinatensystem für eine Bewegung verwendet. Es können weitere Basis-Koordinatensysteme relativ zum Welt-Koordinatensystem definiert werden.

(>>> 7.2.2 "Basis vermessen: 3-Punkt-Methode" Seite 110)

### **Flansch-Koordinatensystem**

Das Flansch-Koordinatensystem beschreibt die aktuelle Position und Orientierung des Roboterflansch-Mittelpunktes. Es ist nicht ortsfest und wird mit dem Roboter bewegt.

Das Flansch-Koordinatensystem wird als Ursprung für Koordinatensysteme verwendet, die am Flansch montierte Werkzeuge beschreiben.

### **Werkzeug-Koordinatensystem**

Das Werkzeug-Koordinatensystem ist ein kartesisches Koordinatensystem, das im Arbeitspunkt des montierten Werkzeugs liegt. Dieser wird als Tool Center Point (TCP) bezeichnet.

Für ein Werkzeug können beliebig viele Frames definiert sein, die als TCP gewählt werden können. Der Ursprung des Werkzeug-Koordinatensystems ist in der Regel identisch mit dem Flansch-Koordinatensystem.

(>>> 9.3.1 "Geometrischer Aufbau von Werkzeugen" Seite 143)

Das Werkzeug-Koordinatensystem wird vom Benutzer in den Arbeitspunkt des Werkzeugs verschoben.

(>>> 7.2.1 "Werkzeug vermessen" Seite 104)

### **Position und Orientierung**

Um die Position und Orientierung eines Objektes zu bestimmen, werden Translation und Rotation bezogen auf ein Bezugs-Koordinatensystem angegeben. Dafür werden 6 Koordinaten verwendet.

#### **Translation**

<b>Koordinate</b>	<b>Beschreibung</b>
Strecke X	Verschiebung entlang der X-Achse des Bezugssystems
Strecke Y	Verschiebung entlang der Y-Achse des Bezugssystems
Strecke Z	Verschiebung entlang der Z-Achse des Bezugssystems

#### **Rotation**

<b>Koordinate</b>	<b>Beschreibung</b>
Winkel A	Drehung um die Z-Achse des Bezugssystems

Koordinate	Beschreibung
Winkel B	Drehung um die Y-Achse des Bezugssystems
Winkel C	Drehung um die X-Achse des Bezugssystems

## 6.8 Roboter manuell verfahren

### Übersicht

Es gibt 2 Arten, den Roboter manuell zu verfahren:

- Kartesisch verfahren  
Der eingestellte TCP wird in positiver oder negativer Richtung entlang der Achsen eines Koordinatensystems verfahren oder um diese Achsen gedreht.
- Achsspezifisch verfahren  
Jede Achse kann einzeln in positiver und negativer Richtung verfahren werden.

### 6.8.1 Fenster Handverfahroptionen

#### Vorgehensweise

Fenster **Handverfahroptionen** öffnen:

- Den Button **Handverfahroptionen** berühren.

Fenster **Handverfahroptionen** schließen:

- Den Button **Handverfahroptionen** oder einen Bereich außerhalb des Fensters berühren.

#### Beschreibung

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

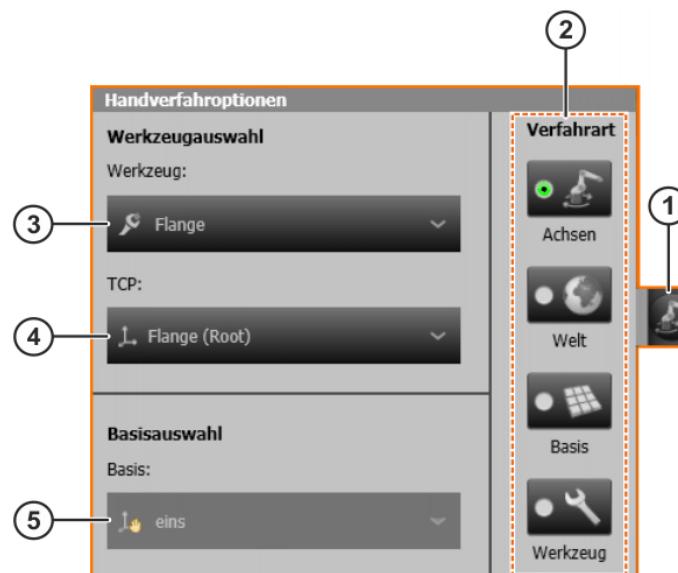


Abb. 6-9: Fenster Handverfahroptionen

<b>Pos.</b>	<b>Beschreibung</b>
1	Button <b>Handverfahroptionen</b> Das angezeigte Symbol ist abhängig von der eingestellten Verfahrtart.
2	Verfahrtart auswählen. Es kann achsspezifisch und in unterschiedlichen Koordinatensystemen kartesisch verfahren werden. Die ausgewählte Verfahrtart ist grün markiert und wird auf dem Button <b>Handverfahroptionen</b> angezeigt. <ul style="list-style-type: none"> <li>■ <b>Achsen:</b> Es wird achsspezifisch verfahren.</li> <li>■ <b>Welt:</b> Der ausgewählte TCP wird kartesisch im Welt-Koordinatensystem verfahren.</li> <li>■ <b>Basis:</b> Der ausgewählte TCP wird kartesisch im ausgewählten Basis-Koordinatensystem verfahren.</li> <li>■ <b>Werkzeug:</b> Der ausgewählte TCP wird kartesisch im eigenen Werkzeug-Koordinatensystem verfahren.</li> </ul>
3	Roboterflansch oder montiertes Werkzeug auswählen. Nicht möglich, während eine Applikation ausgeführt wird.  Die Frames des ausgewählten Werkzeugs können als TCP für das kartesische Handverfahren ausgewählt werden. Die eingesetzten Lastdaten des Werkzeugs werden berücksichtigt.  Wird eine Roboter-Applikation pausiert, steht das aktuell in der Applikation verwendete Werkzeug unter dem Namen <b>Applikationswerkzeug</b> zur Auswahl.  (>>> "Applikationswerkzeug" Seite 79)
4	TCP auswählen.  Als TCP stehen alle Frames des ausgewählten Werkzeugs zur Verfügung.  Der hier manuell eingestellte TCP bleibt erhalten. Dies ist auch dann der Fall, wenn eine Applikation pausiert wird, die einen anderen TCP verwendet hat.  <b>Ausnahmen:</b> <ul style="list-style-type: none"> <li>■ In den Handverfahroptionen ist das Applikationswerkzeug ausgewählt. Wird in diesem Fall das verwendete Werkzeug in der Applikation geändert und anschließend ein Bewegungsbefehl ausgeführt, wird auch das gewählte Applikationswerkzeug in den Handverfahroptionen angepasst. Der eingestellte TCP in den Handverfahroptionen ist dann automatisch derjenige Frame des Applikationswerkzeugs, mit dem die zuletzt kommandierte Bewegung ausgeführt wurde.</li> <li>■ In den Handverfahroptionen ist der aktive Programm-TCP des Applikationswerkzeugs gewählt. In diesem Fall ändert sich der TCP für das Handverfahren entsprechend dem aktuell verwendeten TCP in der Applikation.</li> </ul>
5	Basis auswählen. Nur möglich, wenn die Verfahrtart <b>Basis</b> ausgewählt ist.  Als Basis stehen alle Frames zur Verfügung, die in Sunrise.Workbench als Basis gekennzeichnet wurden.

**Applikationswerkzeug**

Das Applikationswerkzeug setzt sich aus allen Frames zusammen, die zur Laufzeit unterhalb des Roboterflansches liegen. Das können Frames eines mit dem attachTo-Befehl mit dem Roboterflansch verbundenen Werkzeugs

oder Werkstücks sein sowie in der Applikation erzeugte und mit dem Flansch direkt oder indirekt verknüpfte Frames.

Das Applikationswerkzeug steht nur dann in den Handverfahroptionen zur Auswahl, wenn eine Roboter-Applikation pausiert ist und vor dem Pausieren ein Bewegungsbefehl an die Robotersteuerung geschickt wurde.

Wird das Applikationswerkzeug ausgewählt, wird in den Handverfahroptionen automatisch der Frame als TCP gesetzt, mit dem der aktuelle Bewegungsbefehl in der Applikation ausgeführt wird. Alle anderen Frames, die sich zur Laufzeit hierarchisch unter dem Flansch-Koordinatensystem befinden, können ebenfalls als TCP für das Handverfahren ausgewählt werden.

Der Roboterflansch-Frame steht unter dem Namen **ApplicationTool(Root)** als TCP für das Handverfahren zur Auswahl.

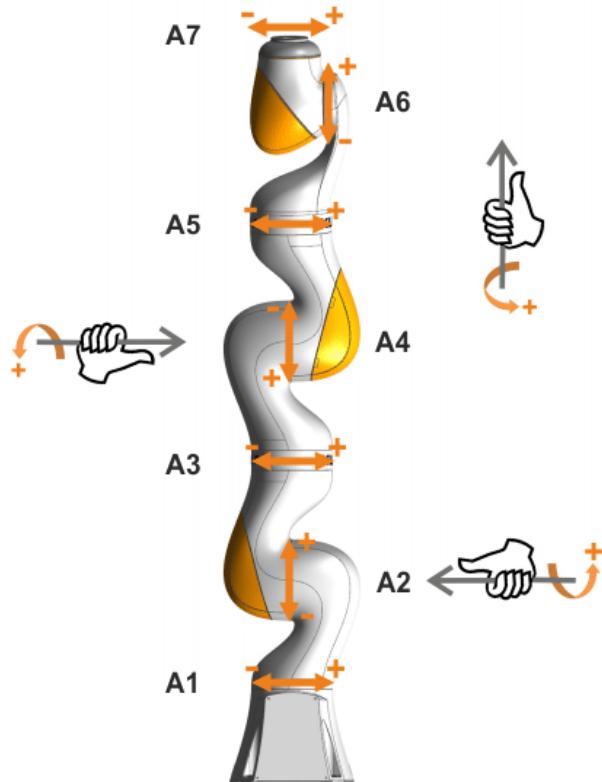
### 6.8.2 Hand-Override (HOV) einstellen

<b>Beschreibung</b>	Der Hand-Override bestimmt die Geschwindigkeit des Roboters beim manuellen Verfahren. Welche Geschwindigkeit der Roboter bei 100 % Hand-Override tatsächlich erreicht, ist abhängig von verschiedenen Faktoren, u. a. vom Robotertyp. Die Geschwindigkeit des eingestellten Arbeitspunkts kann jedoch 250 mm/s nicht übersteigen.
<b>Vorgehensweise</b>	<ol style="list-style-type: none"><li>Den Button <b>HOV</b> berühren. Das Fenster <b>Handverfahrgeschwindigkeit</b> öffnet sich.</li><li>Den gewünschten Hand-Override einstellen. Er kann entweder über die Plus-Minus-Tasten oder über den Regler eingestellt werden.<ul style="list-style-type: none"><li>Plus-Minus-Tasten: Der Override kann schrittweise auf folgende Werte gesetzt werden: 100%, 75%, 50%, 30%, 10%, 5%, 3%, 1%, 0%.</li><li>Regler: Der Override kann in 1%-Schritten geändert werden.</li></ul></li><li>Den Button <b>HOV</b> oder einen Bereich außerhalb des Fensters berühren, um das Fenster zu schließen.</li></ol>
<b>Alternative Vorgehensweise</b>	Alternativ kann der Override mit der Plus-Minus-Taste rechts am smartPAD eingestellt werden. Einstellung in den Schritten 100%, 75%, 50%, 30%, 10%, 5%, 3%, 1%, 0%.

### 6.8.3 Mit Verfahrtasten achsspezifisch verfahren

<b>Voraussetzung</b>	<ul style="list-style-type: none"><li>■ Betriebsart T1</li></ul>
<b>Vorgehensweise</b>	<ol style="list-style-type: none"><li>In den Handverfahroptionen <b>Achsen</b> als Verfahrtart auswählen. Neben den Verfahrtasten werden die Achsen A1 bis A7 angezeigt.</li><li>Hand-Override einstellen.</li><li>Zustimmungsschalter drücken und halten. Ist die Fahrerlaubnis erteilt, werden die Anzeigen neben den Verfahrtasten weiß hervorgehoben.</li><li>Auf die Plus- oder Minus-Verfahrtaste drücken, um eine Achse in positiver oder negativer Richtung zu bewegen.</li></ol>

## Beschreibung



**Abb. 6-10: Achsspezifisch verfahren**

Die positive Drehrichtung der Roboterachsen ist mit der Rechte-Hand-Regel bestimmbar. Dazu stellt man sich den Kabelstrang vor, der innerhalb des Roboters vom Fuß bis zum Flansch verläuft. Die Finger der rechten Hand umschließen – gedanklich – an der Stelle der betrachteten Achse den Kabelstrang. Wichtig ist, dass der Daumen dabei ausgestreckt bleibt. Der Daumen liegt so auf dem Kabelstrang, dass er in die gleiche Richtung zeigt, wie der Kabelstrang im Inneren der Achse weiter zum Flansch läuft. Die übrigen Finger der rechten Hand zeigen dann in die positive Drehrichtung der Roboterachse.

### 6.8.4 Mit Verfahrtasten kartesisch verfahren

#### Voraussetzung

- Betriebsart T1

#### Vorgehensweise

1. Das Koordinatensystem für das kartesische Handverfahren auswählen. Zur Verfügung stehen **Welt**, **Basis** und **Werkzeug**.  
Neben den Verfahrtasten werden folgende Bezeichnungen angezeigt:
  - **X, Y, Z**: für die linearen Bewegungen entlang der Achsen des gewählten Koordinatensystems
  - **A, B, C**: für die rotatorischen Bewegungen um die Achsen des gewählten Koordinatensystems
  - **R**: für die Nullraum-Bewegung
2. Gewünschtes Werkzeug und gewünschten TCP des Werkzeugs auswählen.
3. Wenn das Koordinatensystem **Basis** für das kartesische Handverfahren ausgewählt ist, den gewünschten Basis-Frame auswählen.



Die Frames, die als Basis für das Handverfahren auswählbar sein sollen, müssen zuvor in Sunrise.Workbench als Basis gekennzeichnet werden.

(>>> 9.2.2 "Frame als Basis kennzeichnen" Seite 138)

4. Hand-Override einstellen.
5. Zustimmungsschalter drücken und halten.  
Ist die Fahr freigabe erteilt, werden die Anzeigen neben den Verfahrtasten weiß hervorgehoben.
6. Auf die Plus- oder Minus-Verfahrtaste drücken, um den Roboter in positiver oder negativer Richtung zu bewegen.

#### 6.8.4.1 Nullraum-Bewegung

##### Beschreibung

Der Leichtbauroboter besitzt 7 Achsen und ist dadurch kinematisch redundant. Das bedeutet, dass er jeden Punkt im Arbeitsraum theoretisch mit unendlich vielen Achskonfigurationen anfahren kann.

Aufgrund der kinematischen Redundanz kann beim kartesischen Verfahren eine sogenannte Nullraum-Bewegung durchgeführt werden. Bei der Nullraum-Bewegung werden die Achsen so gedreht, dass die Position und Orientierung des eingestellten TCP während der Bewegung beibehalten werden.



**Abb. 6-11: Nullraum-Bewegung**

##### Eigenschaften

- Die Nullraum-Bewegung wird über den "Ellenbogen" des Roboterarms ausgeführt.
- Die Stellung des Ellenbogens wird durch den Ellenbogen-Winkel ( $R$ ) beschrieben.
- Beim kartesischen Verfahren mit den Verfahrtasten kann die Stellung des Ellenbogen-Winkels ( $R$ ) geändert werden.

##### Anwendungsbereiche

- Für eine gegebene Position und Orientierung des TCP kann die optimale Achskonfiguration eingenommen werden. Das ist vor allem bei einem eingeschränkten Arbeitsraum nützlich.
- Bei Erreichen eines Software-Endschalters kann man versuchen, den Roboter durch eine Änderung des Ellenbogen-Winkels aus den Endschaltern zu fahren.

## 6.9 Betriebsart KRF – Roboter kontrolliert freifahren

##### Beschreibung

KRF ist eine Betriebsart, die zur Verfügung steht, wenn der Roboter von der Sicherheitssteuerung gestoppt wurde und die auslösende Zeile in der Sicherheitskonfiguration eine der folgenden Überwachungen enthält:

- Achsbereichsüberwachung

- Kartesische Geschwindigkeitsüberwachung
- Kartesische Arbeitsraumüberwachung
- Kartesische Schutzraumüberwachung
- Werkzeugorientierung
- Achsmomentenüberwachung
- Kollisionserkennung
- TCP-Kraftüberwachung

In der Betriebsart KRF kann der Roboter verfahren werden, wenn er von der Sicherheitssteuerung aufgrund einer der folgenden Ursachen gestoppt wird

- Roboter verletzt einen sicher überwachten Raum.
- Orientierung des sicherheitsgerichteten Werkzeugs liegt außerhalb des sicher überwachten Bereichs.
- Roboter verletzt eine sicher überwachte Kraft- oder Momentengrenze.
- Ein Positionssensor ist bei aktiver kartesischer Geschwindigkeitsüberwachung nicht justiert.

Die Verfahrgeschwindigkeit des eingestellten Arbeitspunkts in der Betriebsart KRF entspricht der Handverfahrgeschwindigkeit in der Betriebsart T1, maximal 250 mm/s.

Der Roboter lässt sich in der Betriebsart KRF unabhängig von den aktivierten Überwachungen verfahren. Beim Überfahren weiterer Überwachungsgrenzen wird kein Stopp ausgelöst. Geschwindigkeitsüberwachungen sind im KRF-Betrieb weiterhin aktiv.

#### **Vorgehensweise**

1. In die Betriebsart KRF wechseln.
  2. Den Roboter manuell verfahren und in eine Position bringen, in der die stoppauslösende Überwachung nicht mehr verletzt ist.
- Wenn die überwachten Parameter wieder außerhalb des verletzten Bereichs liegen und sich nach 4 Sekunden immer noch in einem erlaubten Bereich befinden, wird automatisch in die Betriebsart T1 gewechselt.

### **6.10 Roboter von Hand führen**



**VORSICHT** Der Bediener muss sicherstellen, dass das smartPAD während des Handführens von keiner anderen Person unbemerkt bedient werden kann. Wenn dies nicht beachtet wird, können Verletzungen die Folge sein.

### **6.11 Sicherheitssteuerung fortsetzen**

#### **Beschreibung**

Bei Verbindungs- oder Peripheriefehlern wird die Sicherheitssteuerung pausiert (je nach Fehler nach ein- oder mehrmaligem Auftreten). Durch das Pausieren der Sicherheitssteuerung, wird der Roboter gestoppt und alle sicheren Ausgänge werden abgeschaltet. Wenn der Fehler behoben ist, kann die Sicherheitssteuerung fortgesetzt werden.

#### **Vorgehensweise**

1. In der Stationssicht **Safety > Status** wählen. Die Ansicht **Status** öffnet sich.  
In der Ansicht wird die Fehlerursache angezeigt. Der Button **Sicherheitssteuerung fortsetzen** ist inaktiv.
2. Den Fehler beheben. Der Button **Sicherheitssteuerung fortsetzen** ist jetzt aktiv.
3. Auf **Sicherheitssteuerung fortsetzen** drücken. Die Sicherheitssteuerung wird fortgesetzt.

## 6.12 Haltebremsen öffnen

### Beschreibung

Wurde der Roboter in eine Klemmsituation gebracht, durch die das Maximaldrehmoment des Gelenkmomenten-Sensors in mindestens einer Achse überschritten wird, wird die Sicherheitssteuerung pausiert. Der Roboter kann nicht mehr verfahren werden.

In diesem Fall ist es möglich, durch ein kurzzeitiges Öffnen der Haltebremsen das auf die betroffenen Achsen wirkende Moment zu reduzieren.



#### VORSICHT

Nach der Überschreitung des Maximaldrehmoments wird die Referenzierung der Positions- und Momentensensoren der betroffenen Achsen verworfen. Um die Sicherheitsintegrität der positions- und achsmomentabhängigen Überwachungen zu gewährleisten, müssen die Positions- und Momentensensoren der betroffenen Achsen nach dem Beseitigen der Klemmung erneut referenziert werden.

### Maximaldrehmoment

Das Maximaldrehmoment liegt zwischen Nenn- und Grenzdrehmoment des Sensors. Der Sensor darf bis zum Grenzdrehmoment belastet werden, ohne bleibenden Schaden zu nehmen. Um erhöhte Messungenauigkeiten jenseits des Nenndrehmoments zu berücksichtigen, wird die Sicherheitssteuerung bereits bei Erreichen des Maximaldrehmoments pausiert.

Solange der vom Sensor ermittelte Wert das Maximaldrehmoment nicht überschreitet ist sichergestellt, dass das Grenzdrehmoment nicht überschritten ist.

### Voraussetzung

- Das Maximaldrehmoment des Gelenkmomenten-Sensors in mindestens einer Achse ist überschritten.

### Vorgehensweise

1. In der Stationssicht **Safety > Status** wählen. Die Ansicht **Status** öffnet sich.  
In der Ansicht wird eine Fehlermeldung für das Überschreiten des Maximaldrehmoments angezeigt.
2. Zustimmungsschalter drücken und halten. Die Haltebremsen der betroffenen Achsen werden kurzzeitig geöffnet.



Beim Öffnen der Bremsen bewegt sich der Roboter geringfügig.

3. Wenn die Bremsen wieder geschlossen sind, Zustimmungsschalter loslassen.
4. Falls in mindestens einer Achse das Maximaldrehmoment des Gelenkmomenten-Sensors weiterhin überschritten ist: Schritt 1 und 2 so oft wiederholen bis das Moment in allen Achsen unterhalb des Maximaldrehmoments liegt.
5. Auf **Sicherheitssteuerung fortsetzen** drücken. Die Sicherheitssteuerung wird fortgesetzt.
6. Falls erforderlich, Roboter per Handverfahren in eine unkritische Position verfahren.
7. Wenn Sicherheitsfunktionen konfiguriert sind, die achsmomentbasierte oder positionsbasierte AMFs verwenden: Momenten- und/oder Positionsreferenzierung durchführen.  
(>>> 13.10.2 "Momentenreferenzierung" Seite 241)  
(>>> 13.10.1 "Positionsreferenzierung" Seite 240)

## 6.13 Frames teachen und manuell anfahren

### 6.13.1 Frames anzeigen

**Vorgehensweise** ■ In der Stationssicht **Frames** wählen. Die Frames-Sicht öffnet sich.

**Beschreibung** Die Sicht enthält die für das Sunrise-Projekt angelegten Frames. Hier werden die Frames geteacht. Durch das Teachen werden Position und Orientierung eines Frames im Raum sowie zugehörige Redundanzinformationen aufgenommen.

- Geteachte Frames können manuell angefahren werden.
  - Geteachte Frames können als Zielpunkte von Bewegungen verwendet werden. Wird eine Applikation ausgeführt und der Ziel-Frame einer Bewegung angefahren, wird er in der Frames-Sicht markiert.
- (>>> 6.16.1 "Ziel-Frame der aktuell ausgeführten Bewegung anzeigen"  
Seite 96)

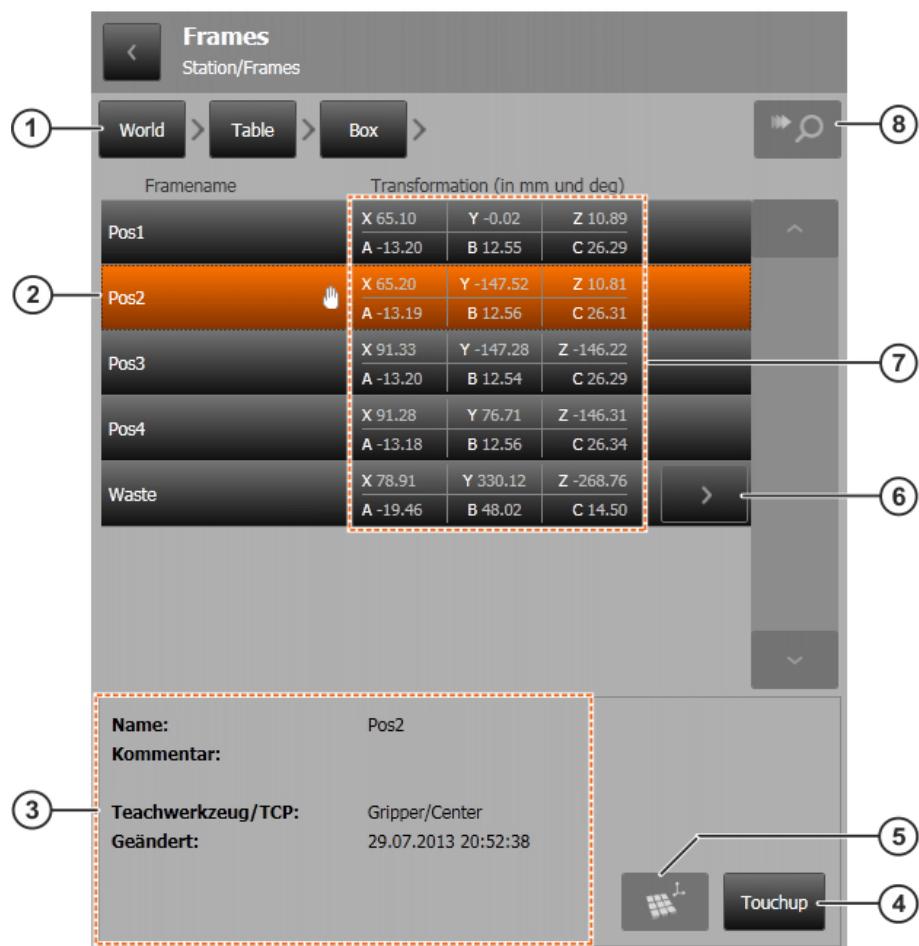


Abb. 6-12: Frames-Sicht

Pos.	Beschreibung
1	Frame-Pfad Pfad zu den Frames der aktuell angezeigten Hierarchieebene. Ausgehend von <b>World</b> bis zum direkten Eltern-Frame (hier <b>Box</b> ).
2	Frames der aktuellen Hierarchieebene Ein Frame kann durch Berühren markiert werden. Der hier markierte Frame ist mit einem Handsymbol gekennzeichnet. Das Handsymbol bedeutet, dass dieser Frame als Basis für das Handverfahren verwendet und vermessen werden kann.
3	Eigenschaften des markierten Frame <ul style="list-style-type: none"> <li>■ Name des Frames</li> <li>■ Kommentar</li> <li>■ Verwendetes Werkzeug beim Teachen des Frames</li> <li>■ Datum und Uhrzeit der letzten Änderung</li> </ul>
4	Button <b>Touchup</b> Ein markierter Frame kann geteacht werden. Ist kein Frame markiert, ist der Button inaktiv.
5	Button zum Setzen der Basis für das Handverfahren Über den Button wird der markierte Frame in den Handverfahroptionen als Basis für das Handverfahren gesetzt. (>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78) Der Button ist nur aktiv, wenn in den Handverfahroptionen die Verfahrart <b>Basis</b> ausgewählt ist und der markierte Frame in Sunrise.Workbench als Basis gekennzeichnet wurde.
6	Button zum Einblenden von Kind-Frames Der Button steht nur zur Verfügung, wenn ein Frame Kindelemente besitzt. Über den Button werden die direkten Kindelemente eines Frames eingeblendet.
7	Frame-Koordinaten bezogen auf den Eltern-Frame
8	Suche-Button Der Suche-Button ist nur aktiv, wenn eine Applikation ausgeführt und der Ziel-Frame einer Bewegung angefahren wird. Über den Button kann zu diesem Ziel-Frame geschalten werden, falls er noch nicht angezeigt wird.

### 6.13.2 Frames teachen

#### Beschreibung

Die Koordinaten eines Frames können über die smartHMI geändert werden. Dazu fährt man die neue Position des Frames mit dem gewünschten TCP an und teacht den Frame. Dabei werden die neue Position und Orientierung übernommen.



Es wird empfohlen, das Projekt sofort nach dem Teachen der Frames zu synchronisieren, damit die neuen Frame-Daten auch im entsprechenden Projekt in Sunrise.Workbench aktualisiert werden.

#### Voraussetzung

- Das Werkzeug mit dem gewünschten TCP ist in den Handverfahroptionen eingestellt.  
(>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78)

**i** Es wird empfohlen, zum Teachen von Frames nicht das Applikationswerkzeug zu verwenden, da das Applikationswerkzeug nur in Applikationspausen in den Handverfahroptionen zur Verfügung steht. Stattdessen kann das dem aktuellen Applikationswerkzeug entsprechende, in den Objektvorlagen des Projekts angelegte Werkzeug verwendet werden.

■ Betriebsart T1

### Vorgehensweise

1. Die gewünschte Position des Frames mit dem TCP anfahren.
2. In der Frames-Sicht den Frame markieren, dessen Position geteacht werden soll.
3. Auf **Touchup** drücken, um die aktuellen TCP-Koordinaten für den markierten Frame zu übernehmen.
4. Die Koordinaten und Redundanzinformationen des geteachten Punkts werden im Dialog **Touchup Daten übernehmen** angezeigt. Auf **Übernehmen** drücken, um die neuen Werte zu speichern.

**⚠️ WARNUNG** Wird ein Frame geändert, wirkt sich die Änderung auf alle Applikationen aus, in denen der Frame Verwendung findet. Geänderte Programme müssen immer zuerst in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden.

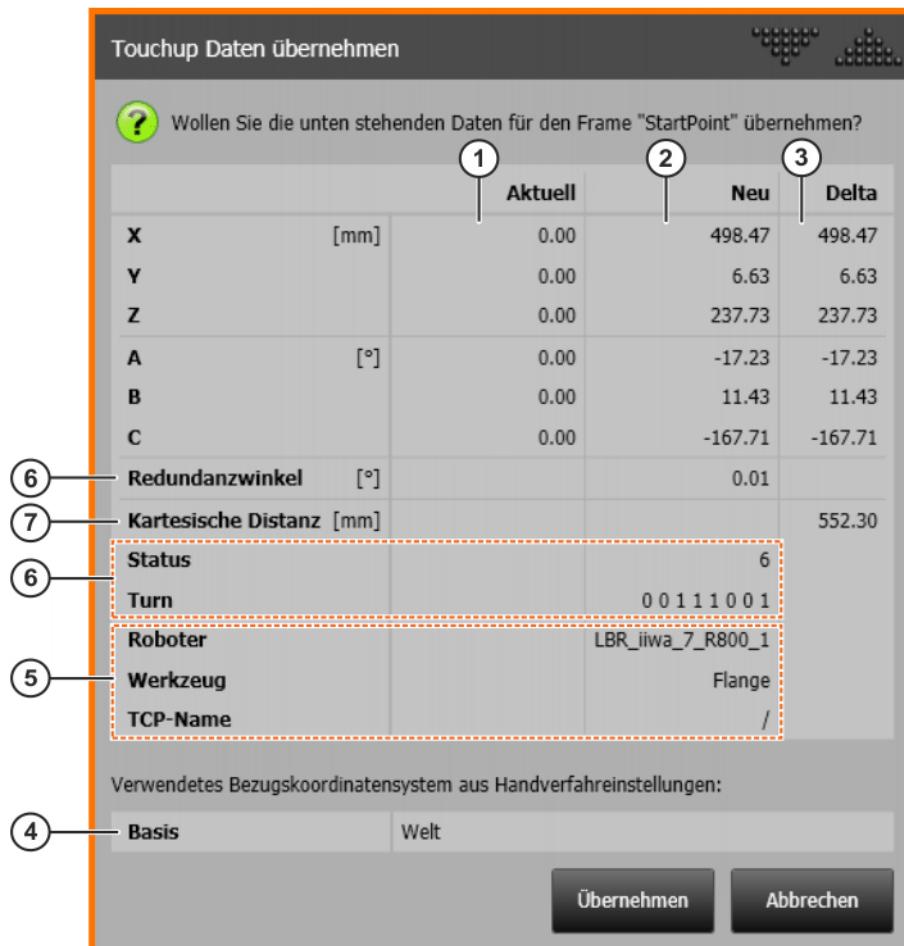


Abb. 6-13: Touchup Daten übernehmen

Pos.	Beschreibung
1	Bisher gespeicherte Werte
2	Neue Werte

Pos.	Beschreibung
3	Änderungen zwischen den bisher gespeicherten und neuen Werten
4	Basis für das Handverfahren Alle im Dialog angezeigten Koordinatenwerte des Frames beziehen sich auf die in den Handverfahroptionen eingestellte Basis für das Handverfahren. Diese Werte unterscheiden sich in der Regel von den Koordinatenwerten des Frames bezogen auf seinen Eltern-Frame. (>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78)
5	Informationen zu Roboter und Werkzeug, die beim Teachen verwendet wurden (>>> 9.2.5 "Eigenschaften eines Frames anzeigen und ändern" Seite 140)
6	Redundanzinformationen zu geteachtem Punkt (>>> 9.2.5 "Eigenschaften eines Frames anzeigen und ändern" Seite 140)
7	Kartesische Distanz zwischen der aktuellen und neuen Position des Frames

### 6.13.3 Fenster Verfahrart

- Vorgehensweise** Fenster **Verfahrart** öffnen:
- Den Button **Verfahrart** neben der Start-Taste berühren.
- Fenster **Verfahrart** schließen:
- Den Button **Verfahrart** oder einen Bereich außerhalb des Fensters berühren.
- Beschreibung** Im Fenster **Verfahrart** kann die Funktionalität der Start-Taste konfiguriert werden.

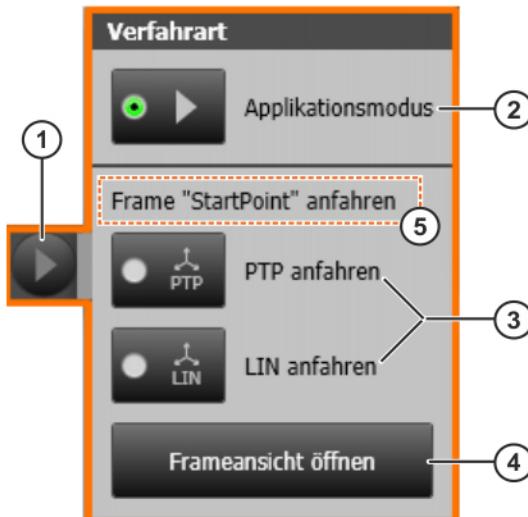


Abb. 6-14: Fenster Verfahrart

<b>Pos.</b>	<b>Beschreibung</b>
1	Button <b>Verfahrtart</b> Das angezeigte Symbol ist abhängig von der eingestellten Verfahrtart.
2	Verfahrtart <b>Applikationsmodus</b> In dieser Verfahrtart dient die Start-Taste zum Starten von Applikationen. <b>Hinweis:</b> Beim Wechsel in die Betriebsart T2 oder Automatik wird automatisch die Verfahrtart <b>Applikationsmodus</b> eingestellt.
3	Verfahrtarten <b>PTP anfahren / LIN anfahren</b> In diesen Verfahrtarten dient die Start-Taste zum manuellen Anfahren von Frames mit einer PTP- oder LIN-Bewegung. Sie können nur ausgewählt werden, wenn ein Frame in der Frames-Sicht markiert und die Betriebsart T1 eingestellt ist. <b>Hinweis:</b> In der Verfahrtart <b>PTP anfahren</b> wird der Status des Ziel-Frames berücksichtigt. Dadurch kann es zu einer Bewegung der Achsen kommen, auch wenn der Zielpunkt kartesisch bereits erreicht ist. <b>Hinweis:</b> In der Verfahrtart <b>LIN anfahren</b> wird der Status des Ziel-Frames nicht berücksichtigt.
4	Button <b>Frameansicht öffnen</b> Über den Button kann zur Frames-Sicht gewechselt werden.
5	Frame-Anzeige Hier wird der Name des Frames angezeigt, der aktuell in der Frames-Sicht markiert ist.

**Symbole**

Folgende Symbole werden je nach eingestellter Verfahrtart auf dem Button **Verfahrtart** angezeigt:

<b>Symbol</b>	<b>Beschreibung</b>
	Verfahrtart <b>Applikationsmodus</b>
	Verfahrtart <b>PTP anfahren</b>
	Verfahrtart <b>LIN anfahren</b>

**6.13.4 Frames manuell anfahren****Beschreibung**

Geteachte Frames können mit einer PTP- oder einer LIN-Bewegung manuell angefahren werden. Bei einer PTP-Bewegung wird der Frame auf dem schnellsten Weg, bei einer LIN-Bewegung auf einer vorhersehbaren Bahn angefahren.

Beim Anfahren von Frames wird in folgenden Fällen eine Warnmeldung angezeigt:

- Das gewählte Werkzeug entspricht nicht dem Werkzeug, mit dem der Frame geteacht wurde.
- Der gewählte TCP entspricht nicht dem TCP, mit dem der Frame geteacht wurde.

- Die Transformation des TCP-Frames wurde geändert.

Ist der Frame dennoch erreichbar, ist ein Anfahren möglich.

#### Voraussetzung

- Der Frame ist mit dem ausgewählten TCP anfahrbar.
- Betriebsart T1

#### Vorgehensweise

- In der Frames-Sicht den gewünschten Frame markieren.
- Im Fenster **Verfahrtart** die gewünschte Verfahrtart auswählen.
- Zustimmungsschalter drücken und halten.
- Start-Taste drücken und halten bis der Frame erreicht ist.



Befindet sich der gewählte Arbeitspunkt bereits an der Zielposition oder ist der Frame mit den aktuellen Einstellungen nicht erreichbar, führt der Roboter keine Bewegung aus.

## 6.14 Programmausführung

### 6.14.1 Roboter-Applikation anwählen

#### Beschreibung

- In der Navigationsleiste unter **Applikationen** die gewünschte Applikation auswählen. Die Applikationssicht öffnet sich und die Applikation geht in den Zustand **Angewählt**.

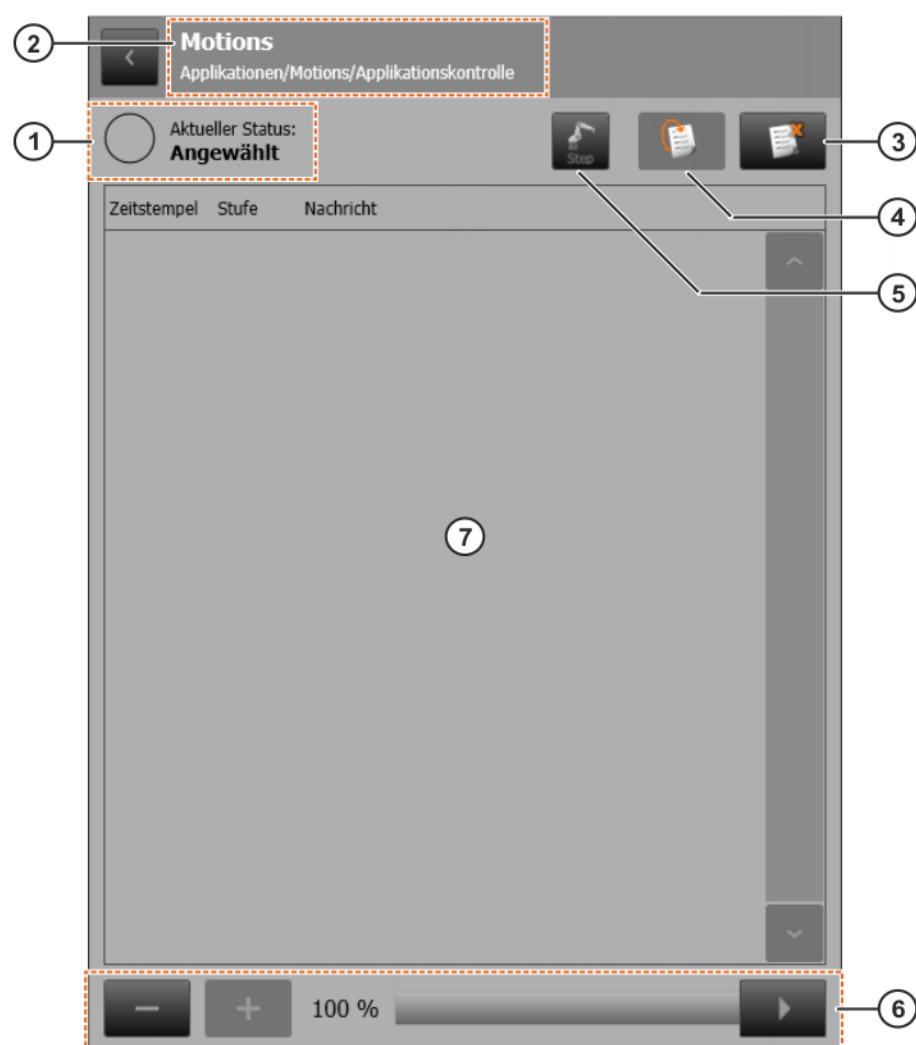


Abb. 6-15: Applikationssicht - Applikation angewählt

<b>Pos.</b>	<b>Beschreibung</b>
1	Aktueller Status der Applikation Der Status wird in Textform und als Symbol angezeigt. (>>> "Statusanzeige Applikation" Seite 91)
2	Anzeige Applikation Der Name der ausgewählten Applikation wird angezeigt, hier <b>Motions</b> .
3	Button <b>Abwählen</b> Wählt die angewählte Applikation ab und schließt die Applikationsansicht. Eine pausierte Applikation wird vor der Abwahl zurückgesetzt. Der Button ist nur aktiv, wenn die Applikation im Zustand <b>Angewählt, Bewegung pausiert</b> oder <b>Fehler</b> ist.
4	Button <b>Zurücksetzen</b> Setzt eine pausierte Applikation zurück. Zurücksetzen bedeutet, die Applikation wird wieder auf den Programmanfang gesetzt und geht in den Zustand <b>Angewählt</b> . Der Button ist nur aktiv, wenn die Applikation pausiert ist.
5	Button <b>Step</b> Durch Drücken des Buttons kann zwischen Step- und Standard-Betrieb gewechselt werden. (>>> 6.14.2 "Programmablaufart auswählen" Seite 92)
6	Manueller Override Der manuelle Override kann über die Plus-Minus-Tasten oder über den Regler eingestellt werden.
7	Meldungsfenster Hier werden Fehlermeldungen und in der Applikation programmierte Benutzermeldungen angezeigt.

### Statusanzeige Applikation

<b>Symbol</b>	<b>Zustand</b>	<b>Beschreibung</b>
	<b>Angewählt</b>	Die Applikation ist angewählt.
	<b>Starten</b>	Die Applikation wird initialisiert.
	<b>Ausführung</b>	Die Applikation wird ausgeführt.
	<b>Bewegung pausiert</b>	Die Applikation ist pausiert. Wird die Applikation über das smartPAD pausiert, z. B. durch Drücken der STOP-Taste, wird nur die Bewegungsausführung angehalten. Andere Befehle, beispielsweise das Schalten von Ausgängen, werden im Zustand <b>Bewegung pausiert</b> ausgeführt bis ein synchroner Bewegungsbefehl erreicht ist.

Symbol	Zustand	Beschreibung
	<b>Fehler</b>	Beim Ausführen der Applikation ist ein Fehler aufgetreten.
	<b>Rückpositionierung</b>	Der Roboter wird rückpositioniert. Die Applikation ist pausiert, da der Roboter die Bahn verlassen hat.
	<b>Stoppen</b>	Die Applikation wird auf den Programmanfang zurückgesetzt und geht in den Zustand <b>Angewählt</b> .

**Start-Taste**

Im Applikationsmodus stehen über die Start-Taste folgende Funktionalitäten zur Verfügung:

Symbol	Beschreibung
	Applikation starten. Eine angewählte Applikation kann gestartet oder eine pausierte Applikation fortgesetzt werden.
	Roboter rückpositionieren. Hat der Roboter die Bahn verlassen, muss er rückpositioniert werden, um die Applikation fortsetzen zu können.



Wenn eine Applikation pausiert ist, kann der Roboter manuell verfahren werden. Das in einer pausierten Applikation verwendete Werkzeug und der aktuelle TCP sind nicht automatisch als Werkzeug und TCP für das kartesische Handverfahren eingestellt.  
(=> 6.8.1 "Fenster Handverfahroptionen" Seite 78)

**STOP-Taste**

Über die STOP-Taste steht folgende Funktionalität zur Verfügung:

Symbol	Beschreibung
	Applikation pausieren. Eine laufende Applikation kann im Automatikbetrieb pausiert werden.

**6.14.2 Programmablaufart auswählen**

- Voraussetzung**
- Applikation ist angewählt.
  - Betriebsart T1 oder T2

- Vorgehensweise**
- Programmablaufart über den Button **Step** auswählen.

Beschreibung	Programmablaufart	Beschreibung
	<b>Continuous</b>	Standard-Betrieb  Das Programm läuft ohne Stopp bis zum Programmende ab.
	<b>Step</b>	Step-Betrieb  Das Programm läuft mit einem Stopp nach jedem Bewegungsbefehl ab. Die Start-Taste muss für jeden Bewegungsbefehl neu gedrückt werden. <ul style="list-style-type: none"> <li>■ Der Zielpunkt einer überschliffenen Bewegung wird nicht überschliffen, sondern mit einem Genauhalt angefahren. Ausnahme: Überschliffene Bewegungen, die bereits vor Aktivierung des Step-Betriebs asynchron an die Robotersteuerung geschickt wurden und dort auf Ausführung warten, stoppen am Überschleifpunkt. Für diese Bewegungen wird beim Fortsetzen noch der Überschleifbogen gefahren.</li> <li>■ Bei einer Spline-Bewegung wird der gesamte Spline-Block als eine Bewegung abgefahren und dann gestoppt.</li> <li>■ Bei einem MotionBatch wird nicht der gesamte Batch abgefahren, sondern nach jeder einzelnen Bewegung des Batches ein Genauhalt ausgeführt.</li> </ul>



Die Programmablaufart kann auch im Quellcode der Applikation gesetzt und abgefragt werden. ([>>> 15.18 "Programmablaufart ändern und abfragen" Seite 338](#))

#### 6.14.3 Manuellen Override einstellen

**Beschreibung** Der manuelle Override bestimmt die Geschwindigkeit des Roboters beim Programmablauf.

Der manuelle Override ist nicht zwangsläufig identisch mit dem effektiven Override, mit dem der Roboter tatsächlich verfährt. Ist ein Applikations-Override programmiert, der von der Applikation gesetzt wird, ergibt sich der effektive Override wie folgt:

Effektiver Override = Manueller Override · Applikations-Override

([>>> 15.19 "Override ändern und abfragen" Seite 339](#))

Der Override wird in Prozent angegeben und bezieht sich auf die programmierte Geschwindigkeit. In der Betriebsart T1 ist die maximale Geschwindigkeit 250 mm/s, unabhängig vom eingestellten Override.

**Voraussetzung** ■ Applikation ist geöffnet.

**Vorgehensweise** ■ Den gewünschten manuellen Override einstellen. Er kann entweder über die Plus-Minus-Tasten oder über den Regler eingestellt werden.

- Plus-Minus-Tasten: Der Override kann schrittweise auf folgende Werte gesetzt werden: 100%, 75%, 50%, 30%, 10%, 5%, 3%, 1%, 0%.
- Regler: Der Override kann in 1%-Schritten geändert werden.

#### 6.14.4 Programm vorwärts starten (manuell)

- |                       |   |
|-----------------------|---|
| <b>Voraussetzung</b>  | <ul style="list-style-type: none"><li>■ Applikation ist angewählt.</li><li>■ Betriebsart T1 oder T2</li></ul>   |
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"><li>1. Programmablaufart wählen.</li><li>2. Zustimmungsschalter drücken und halten.</li><li>3. Start-Taste drücken und gedrückt halten. Die Applikation wird ausgeführt.</li></ol> <p>Um ein manuell gestartetes Programm zu pausieren, die Start-Taste loslassen. Ist die Applikation pausiert, kann sie über den Button <b>Zurücksetzen</b> zurückgesetzt werden.</p> |

#### 6.14.5 Programm vorwärts starten (automatisch)

- |                       |   |
|-----------------------|---|
| <b>Voraussetzung</b>  | <ul style="list-style-type: none"><li>■ Applikation ist angewählt.</li><li>■ Betriebsart Automatik</li><li>■ Projekt wird nicht extern gesteuert.</li></ul>   |
| <b>Vorgehensweise</b> | <ul style="list-style-type: none"><li>■ Start-Taste drücken. Die Applikation wird ausgeführt.</li></ul> <p>Um ein im Automatikbetrieb gestartetes Programm zu pausieren, die STOP-Taste drücken. Ist die Applikation pausiert, kann sie über den Button <b>Zurücksetzen</b> zurückgesetzt werden.</p> |

#### 6.14.6 Roboter nach Verlassen der Bahn rückpositionieren

- |                     |  |
|---------------------|--|
| <b>Beschreibung</b> | <p>Der Roboter kann durch folgende Ereignisse von seiner geplanten Bahn abgebracht werden:</p> <ul style="list-style-type: none"><li>■ Auslösen eines nicht bahntreuen Stopps</li><li>■ Handverfahren bei pauserter Applikation</li></ul> <p>Mithilfe der Start-Taste kann der Roboter rückpositioniert werden. Rückpositionieren bedeutet, dass der Roboter zu der kartesischen Position zurückgefahren wird, an der zuvor die Bahn verlassen wurde. Von dort aus kann die Applikation fortgesetzt werden.</p> <p>Eigenschaften der Bewegung, mit der auf die Bahn zurückgekehrt wird:</p> <ul style="list-style-type: none"><li>■ Es wird eine PTP-Bewegung ausgeführt.<br/>Der Bahnverlauf beim Zurückkehren auf die Bahn ist ein anderer als beim Verlassen der Bahn.</li><li>■ Es wird mit 20 % der maximal möglichen Achsgeschwindigkeit und dem effektiven Override verfahren (Effektiver Override = Manueller Override · Applikations-Override).</li></ul> |
|---------------------|--|



Der aktuell eingestellte Hand-Override ist beim Rückpositionieren irrelevant.

- Es wird mit den Lastdaten verfahren, die zu dem Zeitpunkt gesetzt waren, als die Applikation unterbrochen wurde.
  - Es wird mit dem Reglermodus verfahren, der zu dem Zeitpunkt gesetzt war, als die Applikation unterbrochen wurde.
- Durch einen Impedanzregler zusätzlich aufgeschaltete Kräfte oder Kraftmodulationen werden beim Rückpositionieren weggenommen.

**HINWEIS** Wird ein impedanzgeregelter Roboter rückpositioniert, kann es zu unerwarteten Roboterbewegungen kommen. Da immer auf die Sollposition rückpositioniert wird, stimmt bei einem impedanzgeregelten Roboter die Istposition nach dem Rückpositionieren nicht zwangsläufig mit der Istposition überein, an der die Bahn verlassen wurde. Dies kann in Kontaktsituationen zu unerwartet hohen Kräften führen. Einen impedanzgeregelten Roboter vor dem Rückpositionieren manuell an eine Position verfahren, die der Position, an der die Bahn verlassen wurde, möglichst nahe kommt. Wenn dies nicht beachtet wird, kann es zu Sachschaden kommen.

**HINWEIS** Es darf nur rückpositioniert werden, wenn es auf dem Weg zurück zur Bahn zu keiner Kollision kommen kann. Ist dies nicht sichergestellt, den Roboter zuerst in eine geeignete Position verfahren, von der aus er gefahrlos rückpositioniert werden kann.

#### Vorgehensweise

1. In der Betriebsart T1 oder T2: Zustimmungsschalter drücken und halten.
2. Start-Taste drücken und gedrückt halten. Der Roboter fährt zurück auf die Bahn.

### 6.15 Benutzertasten aktivieren

#### Beschreibung

Die Benutzertasten auf dem smartPAD können mit Funktionen belegt sein. Dem Bediener stehen alle Benutzertasten-Funktionen zur Verfügung, die in einer laufenden Roboter-Applikation oder einem Hintergrund-Task definiert sind. Um die gewünschten Funktionen nutzen zu können, muss der Bediener die zugehörige Benutzertasten-Leiste aktivieren.

#### Vorgehensweise

1. Den Button **Auswahl Benutzertasten** berühren.  
Das Fenster **Auswahl Benutzertasten** öffnet sich. Die aktuell verfügbaren Benutzertasten-Leisten werden angezeigt.
2. Die gewünschte Benutzertasten-Leiste über den zugehörigen Namen-Button auswählen.  
Die Beschriftung oder Grafik auf der smartHMI neben den Benutzertasten ändert sich entsprechend der ausgewählten Leiste. Die Benutzertasten besitzen jetzt die zugehörigen Funktionen.
3. Den Button **Auswahl Benutzertasten** oder einen Bereich außerhalb des Fensters berühren.  
Das Fenster **Auswahl Benutzertasten** schließt sich.

## Beispiel

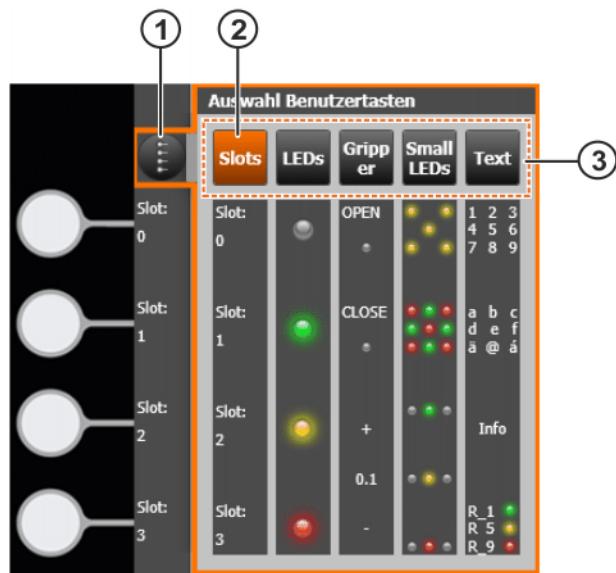


Abb. 6-16: Fenster Auswahl Benutzertasten

- 1 Button **Auswahl Benutzertasten**
- 2 Aktuell aktive Benutzertasten-Leiste
- 3 Namen der verfügbaren Benutzertasten-Leisten

## 6.16 Anzeigefunktionen

### 6.16.1 Ziel-Frame der aktuell ausgeführten Bewegung anzeigen

#### Beschreibung

Wird ein Frame aus dem Frame-Baum in einer Applikation angefahren, wird dieser in der Frames-Sicht gekennzeichnet. Liegt der Ziel-Frame der aktuell ausgeführten Bewegung in der angezeigten Hierarchieebene, ist der Frame-Name mit einem Pfeilsymbol (3 Pfeilspitzen) markiert:

Start	X 586.45	Y 0.02	Z 555.34
	A 180.00	B 47.24	C 179.99

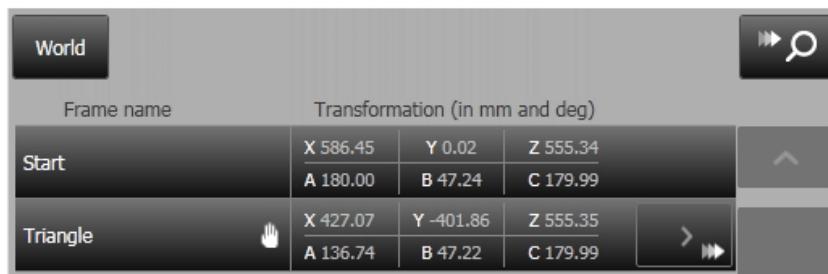
Abb. 6-17: Pfeilsymbol markiert aktuellen Ziel-Frame

Liegt der Ziel-Frame hierarchisch unter einem angezeigten Frame, ist der Button zum Einblenden von Kind-Frames mit einem zusätzlichen Pfeilsymbol (3 Pfeilspitzen) gekennzeichnet:

Triangle		X 427.07	Y -401.86	Z 555.35	
		A 136.74	B 47.22	C 179.99	

Abb. 6-18: Button schaltet bis zum aktuellen Ziel-Frame

Mithilfe des Suche-Buttons im rechten oberen Bereich der Frames-Sicht kann direkt zum aktuellen Ziel-Frame geschalten werden:



The screenshot shows a table titled "Frame name" and "Transformation (in mm and deg)". It lists two frames: "Start" and "Triangle". The "Start" frame has coordinates X: 586.45, Y: 0.02, Z: 555.34; A: 180.00, B: 47.24, C: 179.99. The "Triangle" frame has coordinates X: 427.07, Y: -401.86, Z: 555.35; A: 136.74, B: 47.22, C: 179.99. There are navigation buttons for the table: a magnifying glass icon at the top right, and arrows (up, down, left, right) at the bottom right.

Frame name	Transformation (in mm and deg)		
Start	X 586.45	Y 0.02	Z 555.34
	A 180.00	B 47.24	C 179.99
Triangle	X 427.07	Y -401.86	Z 555.35
	A 136.74	B 47.22	C 179.99

**Abb. 6-19: Suche-Button schaltet direkt zum aktuellen Ziel-Frame**

Der Suche-Button ist inaktiv, wenn kein Frame angefahren wird.

#### Voraussetzung

- Applikation ist angewählt.
- Applikationsstatus **Ausführung** oder **Bewegung pausiert**.
- Die Bewegung verwendet einen in den Applikationsdaten angelegten Ziel-Frame.

#### Vorgehensweise

1. In der Stationssicht **Frames** wählen. Die Frames-Sicht öffnet sich.
2. Über den Button zum Einblenden von Kind-Frames oder über den Suche-Button zum Ziel-Frame schalten.

#### 6.16.2 Achsspezifische Istposition anzeigen

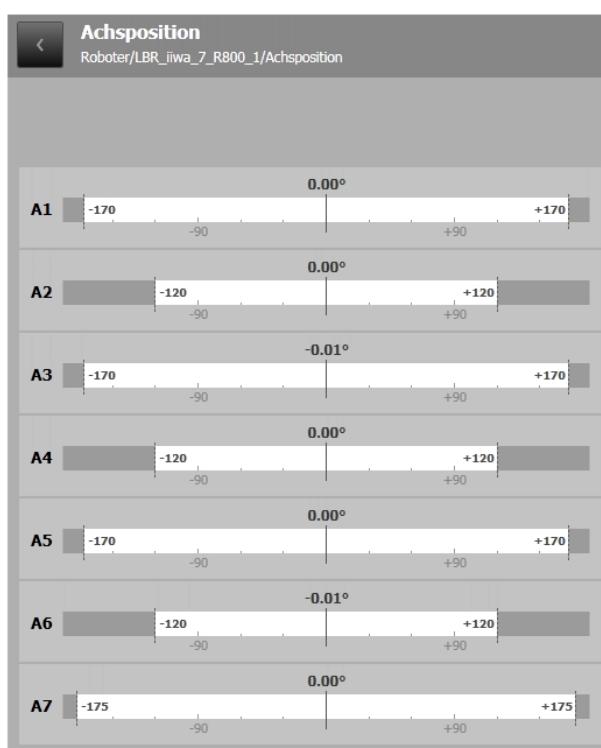
##### Vorgehensweise

- In der Robotersicht **Achsposition** wählen.

##### Beschreibung

Die aktuelle Position der Achsen A1 bis A7 wird angezeigt. Zusätzlich wird für jede Achse der Bereich angezeigt (weißer Balken), in dem sie bewegt werden kann (Begrenzung durch Endanschläge).

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



**Abb. 6-20: Achsspezifische Istposition**

### 6.16.3 Kartesische Istposition anzeigen

- Vorgehensweise**
1. In der Robotersicht **Kartesische Position** wählen.
  2. TCP und Basis im Fenster **Handverfahroptionen** einstellen.

**Beschreibung** Die kartesische Istposition des ausgewählten TCP wird angezeigt. Die Werte beziehen sich auf die in den Handverfahroptionen eingestellte Basis.

Die Anzeige enthält folgende Daten:

- Aktuelle Position (X, Y, Z)
- Aktuelle Orientierung (A, B, C)
- Aktuelle Redundanzinformationen: Status, Turn, Redundanzwinkel (E1)
- Aktuelles Werkzeug, aktueller TCP und aktuelle Basis

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

Ist-Position				
	X	-0.02 mm	Y	0.00 mm
	A	-0.01 °	B	-0.01 °
	Status	7	Turn	126
	E1	-0.01 °		
<b>Verwendetes Werkzeug:</b> Flange				
<b>Verwendeter TCP:</b> Flange (Root)				
<b>Verwendete Basis:</b> Welt				

Abb. 6-21: Kartesische Istposition

### 6.16.4 Achsspezifische Momente anzeigen

- Vorgehensweise**
- In der Robotersicht **Achsmomente** wählen.

**Beschreibung** Die aktuellen Momentenwerte für die Achsen A1 bis A7 werden angezeigt. Zusätzlich wird für jede Achse der Sensor-Messbereich angezeigt (weißer Balken).

Wenn das absolute zulässige Moment an einem Gelenk überschritten wird, färbt sich der dunkelgraue Bereich des Balkens der betroffenen Achse orange. Es wird nur der verletzte Bereich, d. h. entweder der negative oder der positive Teil, eingefärbt.

**i** Die Aktualisierungsrate der angezeigten Werte ist begrenzt. Kurzzeitig anliegende Spitzenwerte werden daher unter Umständen nicht angezeigt.

Die Anzeige enthält folgende Daten:

- Aktuelle absolute Drehmomente
- Aktuelle externe Drehmomente

**i** Die externen Drehmomente werden nur dann korrekt angezeigt, wenn das korrekte Werkzeug angegeben wurde.

- Aktuelles Werkzeug

Die achsspezifischen Momente können auch angezeigt werden, während der Roboter verfährt.

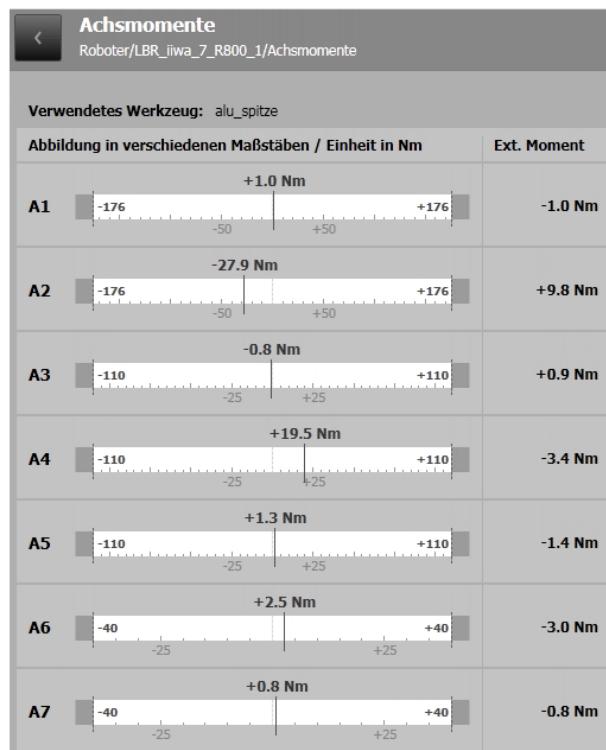


Abb. 6-22: Achsspezifische Momente

### 6.16.5 E/A-Gruppe anzeigen und Wert eines Ausgangs ändern

#### Voraussetzung

- E/A-Gruppe wurde für das Sunrise-Projekt angelegt und zugehörige Signale mit WorkVisual verschaltet.
- Um einen Ausgang zu ändern: Betriebsart T1, T2 oder KRF



Die Ausgänge können unabhängig vom Zustand der Sicherheitssteuerung geändert werden, beispielsweise auch bei einem gedrückten NOT-HALT.

#### Vorgehensweise

1. In der Navigationsleiste unter **E/A-Gruppen** die gewünschte E/A-Gruppe auswählen. Die Ein-/Ausgänge der ausgewählten Gruppe werden angezeigt.
2. Den Ausgang markieren, der geändert werden soll.
3. Bei numerischen Ausgängen wird ein Eingabefeld angezeigt: Den gewünschten Wert eingeben.
4. Zustimmungsschalter drücken und halten. Wert des Ausgangs über den zugehörigen Button ändern.

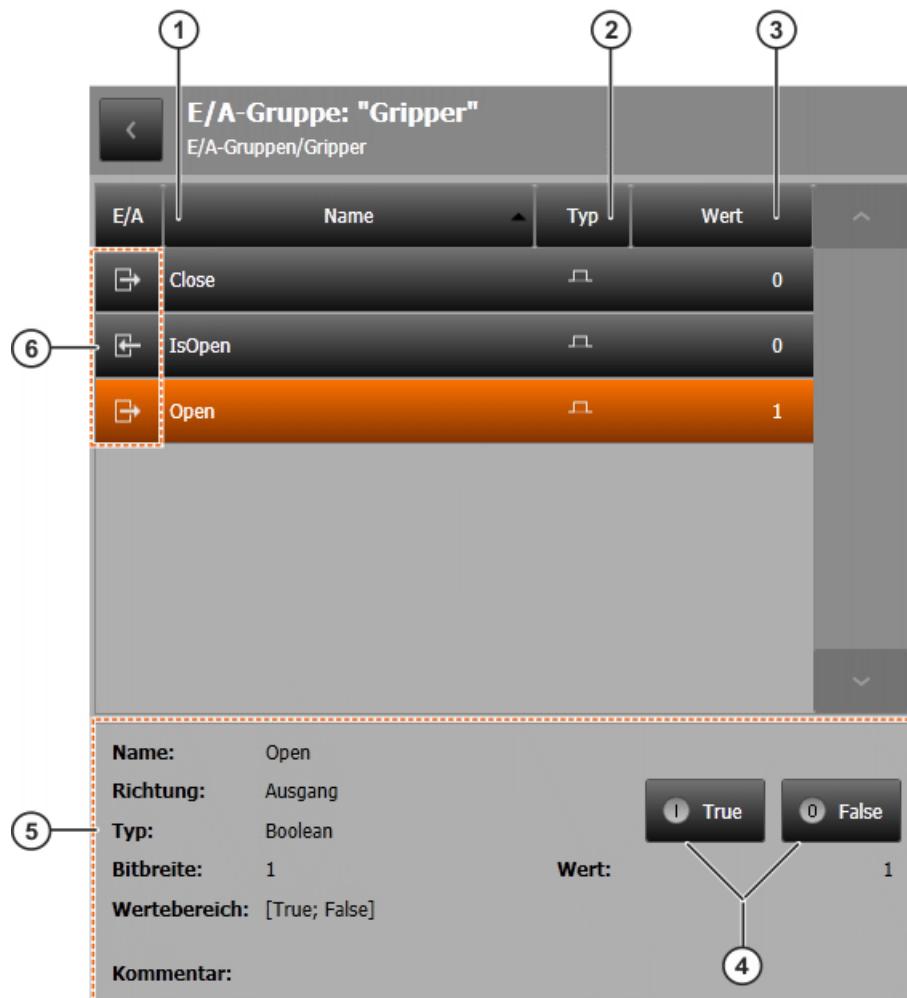
**Beschreibung**

Abb. 6-23: Ein-/Ausgänge einer E/A-Gruppe

Pos.	Beschreibung
1	Name des Ein-/Ausgangs
2	Typ des Ein-/Ausgangs
3	Wert des Ein-/Ausgangs Der Wert wird als Dezimalzahl angezeigt.
4	Buttons zum Ändern von Ausgängen Ist ein Ausgang markiert, kann sein Wert geändert werden. Voraussetzung: Der Zustimmungsschalter ist gedrückt. Die zur Verfügung stehenden Buttons sind abhängig vom Typ des Ausgangs.
5	Signaleigenschaften Die Eigenschaften und der aktuelle Wert des markierten Ein-/Ausgangs werden angezeigt.
6	Signalrichtung Die Symbole geben an, ob es sich bei dem Signal um einen Ein- oder Ausgang handelt.

Folgende Buttons stehen je nach Typ des markierten Ausgangs zur Verfügung:

<b>Button</b>	<b>Beschreibung</b>
<b>True</b>	Buttons zum Ändern boolescher Ausgänge
<b>False</b>	Setzen den markierten booleschen Ausgang auf den Wert True (1) oder False (0).
<b>Setzen</b>	Button zum Ändern numerischer Ausgänge Setzt den markierten numerischen Ausgang auf den eingegebenen Wert.

**Signalrichtung**

Folgende Symbole kennzeichnen die Richtung eines Signals:

<b>Symbol</b>	<b>Beschreibung</b>
	Symbol für einen Ausgang
	Symbol für einen Eingang

**E/A-Typen**

Folgende Symbole kennzeichnen den Typ eines Ein-/Ausgangs:

<b>Symbol</b>	<b>Beschreibung</b>
	Symbol für ein analoges Signal
	Symbol für ein binäres Signal
	Symbol für ein vorzeichenbehaftetes digitales Signal
	Symbol für ein nicht vorzeichenbehaftetes digitales Signal

**6.16.6 IP-Adresse und Software-Version anzeigen****Vorgehensweise**

- In der Stationssicht die Kachel **Information** wählen.

Unter dem Knoten **Station** werden die aktuell installierte Version der System Software und die IP-Adresse der Robotersteuerung angezeigt.

**6.16.7 Robotertyp und Seriennummer anzeigen****Vorgehensweise**

- In der Stationssicht die Kachel **Information** wählen.

Unter dem Knoten **Robotename/Type Plate** werden Informationen zum aktuell verwendeten Roboter angezeigt.

- **Seriennummer:** Seriennummer des angeschlossenen Roboters
- **Angeschlossener Roboter:** Robotertyp des angeschlossenen Roboters
- **Installierter Roboter:** In der Stationskonfiguration des Sunrise-Projekts angegebener Robotertyp

**6.16.8 Meldungen des Virenscaners anzeigen****Voraussetzung**

- Virenschanner ist installiert.

(>>> 10.4 "Virenschanner installieren oder updaten" Seite 162)

**Vorgehensweise**

- In der Stationssicht **KUKA\_Sunrise\_Cabinet** > **VirensScanner** wählen.  
Die Ansicht **VirensScanner** öffnet sich.



Die Meldungen des Virensanners können auch über die Kachel **Protokoll** angezeigt werden.

**Beschreibung**

Die Ansicht **VirensScanner** enthält folgende Daten:

- **VirensScanner Status:** aktiv / inaktiv
- **Version der Virendefinitionsdatei:** Version des Virensanners
- Meldungen zu gefundenen Viren: Die Meldung zu einem Virenfund beinhaltet den Namen des Virus, den Namen der Datei, in der sich der Virus befindet, inklusive Pfadangabe, sowie Datum und Uhrzeit des Fundes.  
Wenn Viren gefunden werden, wechselt die Statusanzeige der Kachel **VirensScanner** in den Zustand "Warnung". Die von den Viren betroffenen Dateien werden automatisch in den Quarantäne-Zustand versetzt.



Wenn der Roboter wegen eines Virenfundes nicht mehr verfahren werden kann, hat der Benutzer folgende Möglichkeiten zur Abhilfe:

- System Software neu auf Robotersteuerung installieren.
- Wenn der Roboter weiterhin nicht verfährt, Diagnosepaket **KRCDiag** erstellen und KUKA Service kontaktieren.

## 7 Inbetriebnahme und Wiederinbetriebnahme

### 7.1 Positionsjustage

Bei der Positionsjustage wird einer festgelegten mechanischen Roboterachsstellung ein Motorwinkel zugeordnet. Nur mit einem justierten Roboter können eingelernte Positionen wiederholgenau angefahren werden. Ein dejusterter Roboter kann nur manuell (achsspezifisch) verfahren werden (Betriebsart T1 oder KRF).

#### 7.1.1 Achsen justieren

**Beschreibung** Der LBR iiwa besitzt in jeder Achse einen auf dem Hall-Effekt basierenden Justagesensor. Die Justageposition der Achse (Nullstellung) befindet sich in der Mitte einer definierten Magnetfolge. Sie wird vom Justagesensor automatisch erkannt, wenn er bei einer Drehung der Achse über die Magnetfolge bewegt wird.

Vor der eigentlichen Justage wird eine automatische Suchfahrt durchgeführt, um eine definierte Vorjustageposition zu finden.

Ist die Suchfahrt erfolgreich, wird die Achse in die Vorjustageposition gefahren. Anschließend wird die Achse so verfahren, dass der Justagesensor über die Magnetfolge bewegt wird. Dabei wird die Motorposition zum Zeitpunkt der Erkennung der Justageposition der Achse als Nullstellung des Motors gespeichert.



Nur eine stets gleiche Vorgehensweise gewährleistet die Wiederholgenauigkeit und Reproduzierbarkeit der Justage. Folgende Regeln sind bei der Justage zu beachten:

- Beim Justieren einer Achse sollten sich alle Achsen in der Kerzenstellung befinden. Ist das nicht möglich, muss die Justage immer in der gleichen Achsstellung durchgeführt werden.
- Die einzelnen Achsen immer in der gleichen Reihenfolge justieren.
- Die Justage immer ohne Last durchführen. Die Justage mit Last wird derzeit nicht unterstützt.

**Voraussetzung**

- Betriebsart T1 oder KRF

**Vorgehensweise**

1. In der Robotersicht **Justage** wählen. Die Ansicht **Justage** öffnet sich.

2. Zustimmungsschalter drücken und halten.

3. Auf den Button **Justiere** der dejustierten Achse drücken.

Zuerst wird durch eine Suchfahrt die Vorjustageposition ermittelt. Anschließend wird die Justagefahrt durchgeführt. Nach erfolgreicher Justage fährt die Achse in die ermittelte Justageposition (Nullstellung).



Schlägt die Suchfahrt oder die Justage fehl, wird der Vorgang abgebrochen und der Roboter bleibt stehen.

#### 7.1.2 Achsen manuell dejustieren

**Beschreibung**

Die gespeicherte Justageposition einer Achse kann gelöscht werden. Dadurch wird die Achse dejustiert. Beim Dejustieren wird keine Bewegung ausgeführt.

**Voraussetzung**

- Betriebsart T1

- Vorgehensweise**
1. In der Robotersicht **Justage** wählen. Die Ansicht **Justage** öffnet sich.
  2. Auf den Button **Dejustiere** der justierten Achse drücken. Die Achse wird dejustiert.

## 7.2 Vermessen

### 7.2.1 Werkzeug vermessen

#### Beschreibung

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

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

Vorteile der Werkzeugvermessung:

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

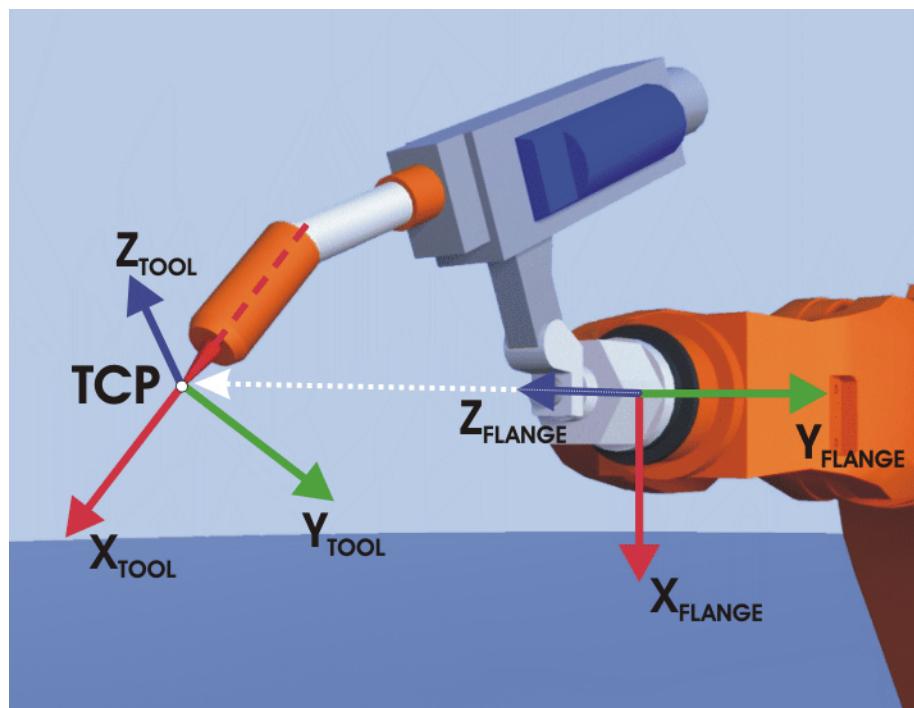


Abb. 7-1: Prinzip der TCP-Vermessung

#### Übersicht

Die Werkzeugvermessung besteht aus 2 Schritten:

Schritt	Beschreibung
1	<p><b>Ursprung des Werkzeug-Koordinatensystems festlegen</b></p> <p>Folgende Methoden stehen zur Auswahl:</p> <ul style="list-style-type: none"> <li>■ XYZ-4-Punkt (&gt;&gt;&gt; 7.2.1.1 "TCP vermessen: XYZ 4-Punkt-Methode" Seite 105)</li> </ul>
2	<p><b>Orientierung des Werkzeug-Koordinatensystems festlegen</b></p> <p>Folgende Methoden stehen zur Auswahl:</p> <ul style="list-style-type: none"> <li>■ ABC-2-Punkt (&gt;&gt;&gt; 7.2.1.2 "Orientierung festlegen: ABC 2-Punkt-Methode" Seite 107)</li> <li>■ ABC-World (&gt;&gt;&gt; 7.2.1.3 "Orientierung festlegen: ABC Welt-Methode" Seite 109)</li> </ul>

### 7.2.1.1 TCP vermessen: XYZ 4-Punkt-Methode

#### Beschreibung

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

Die 4 Flanschpositionen, mit denen der Referenzpunkt angefahren wird, müssen einen bestimmten Mindestabstand einhalten. Liegen die Punkte zu nah beieinander, können die Positionsdaten nicht gespeichert werden. Eine entsprechende Fehlermeldung wird ausgegeben.

Die Qualität der Vermessung kann über den translatorischen Berechnungsfehler beurteilt werden, der beim Vermessen ermittelt wird. Überschreitet dieser Fehler einen bestimmten Grenzwert, wird empfohlen, den TCP noch einmal zu vermessen.

Es ist möglich, Mindestabstand und maximalen Berechnungsfehler in Sunrise.Workbench zu ändern. (>>> 10.1.1 "Parameter für Vermessung konfigurieren" Seite 159)

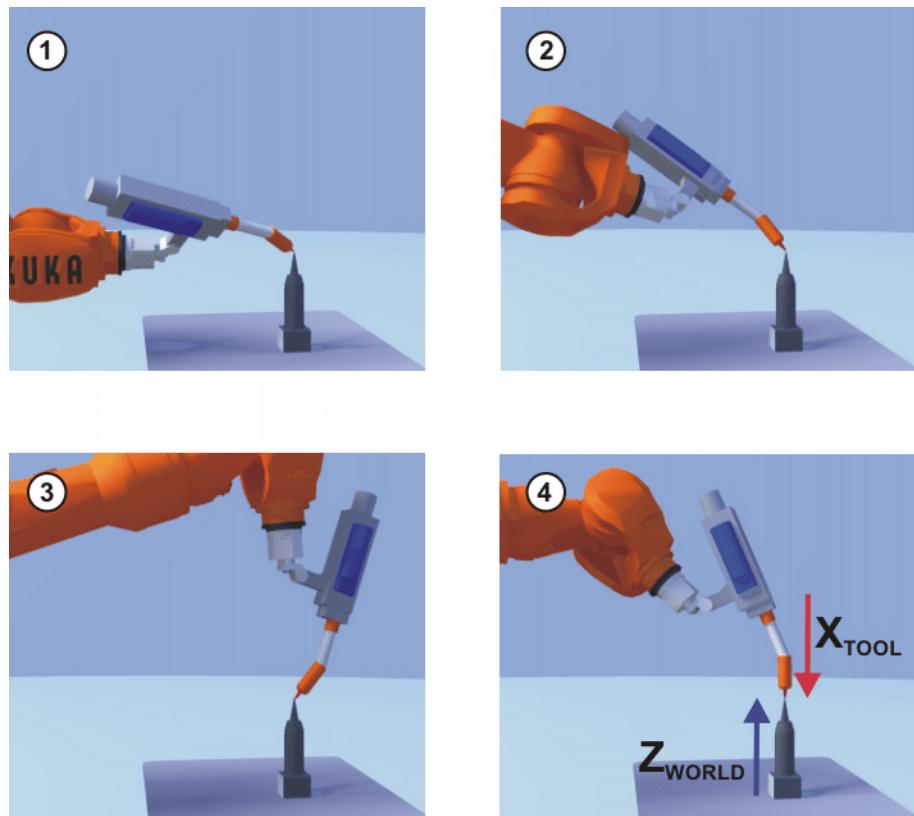


Abb. 7-2: XYZ-4-Punkt-Methode

#### Voraussetzung

- Das zu vermessende Werkzeug ist am Anbauflansch montiert.
- Das zu vermessende Werkzeug und der Frame, der als TCP verwendet wird, wurden in den Objektvorlagen des Projekts angelegt und per Synchronisation auf die Robotersteuerung übertragen.
- Betriebsart T1

#### Vorgehensweise

1. In der Robotersicht **Vermessung > Werkzeugvermessung** wählen. Die Ansicht **Werkzeugvermessung** öffnet sich.
2. Das zu vermessende Werkzeug und den zugehörigen TCP auswählen.
3. Die Methode **TCP Vermessung(XYZ 4-Punkt)** auswählen. Die Messpunkte der Methode werden als Schaltflächen angezeigt:
  - **Messpunkt 1 ... Messpunkt 4**
 Um einen Messpunkt aufnehmen zu können, muss er markiert sein (Schaltfläche ist orange).
4. Mit dem TCP einen beliebigen Referenzpunkt anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
5. Mit dem TCP den Referenzpunkt aus einer anderen Richtung anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
6. Schritt 5 2-mal wiederholen.
7. **Werkzeugdaten bestimmen** drücken. Die Vermessungsdaten und der ermittelte Berechnungsfehler werden im Dialog **Werkzeugdaten übernehmen** angezeigt.
8. Überschreitet der Berechnungsfehler den maximal zulässigen Wert wird eine Warnung angezeigt. **Abbrechen** drücken und TCP neu vermessen.
9. Liegt der Berechnungsfehler unter der konfigurierten Grenze, **Übernehmen** drücken, um die Vermessungsdaten zu speichern.

10. Entweder die Vermessungsansicht schließen oder die Orientierung des Werkzeug-Koordinatensystems mit der ABC 2-Punkt- oder ABC Welt-Methode festlegen.  
(>>> 7.2.1.2 "Orientierung festlegen: ABC 2-Punkt-Methode" Seite 107)  
(>>> 7.2.1.3 "Orientierung festlegen: ABC Welt-Methode" Seite 109)
11. Projekt synchronisieren, um die Vermessungsdaten inklusive Berechnungsfehler in Sunrise.Workbench zu übernehmen.

### **7.2.1.2 Orientierung festlegen: ABC 2-Punkt-Methode**

#### **Beschreibung**

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

Die Punkte müssen einen bestimmten Mindestabstand einhalten. Liegen die Punkte zu nah beieinander, können die Positionsdaten nicht gespeichert werden. Eine entsprechende Fehlermeldung wird ausgegeben.

Es ist möglich, den Mindestabstand in Sunrise.Workbench zu ändern.

(>>> 10.1.1 "Parameter für Vermessung konfigurieren" Seite 159)

Diese Methode wird bei Werkzeugen verwendet, die Kanten und Ecken besitzen, an denen sich der Benutzer orientieren kann. Des Weiteren wird sie benutzt, wenn die Achsrichtungen besonders exakt festgelegt werden müssen.

Für sicherheitsgerichtete Werkzeuge steht diese Methode nicht zur Verfügung.

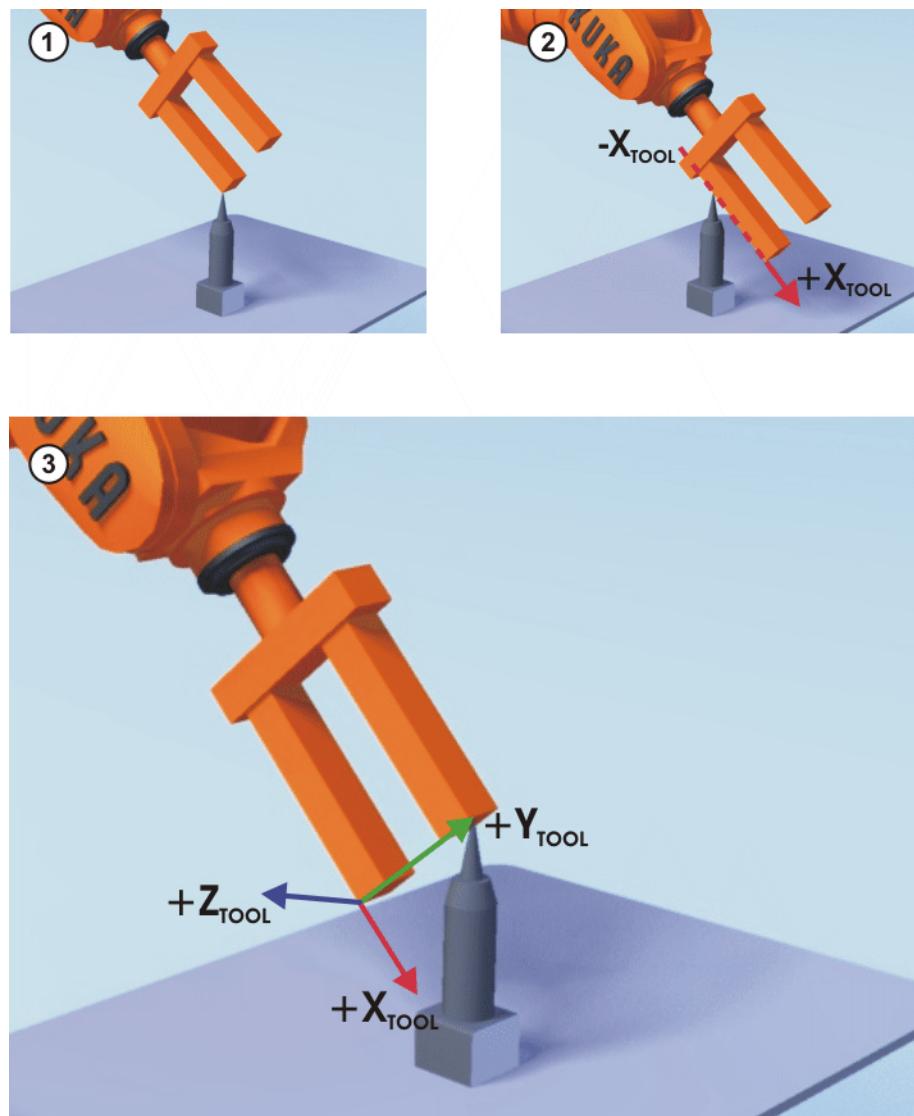


Abb. 7-3: ABC 2-Punkt-Methode

#### Voraussetzung

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

#### Vorgehensweise

1. Nur wenn die Vermessungsansicht nach der TCP-Vermessung geschlossen wurde:  
In der Robotersicht **Vermessung > Werkzeugvermessung** wählen. Die Ansicht **Werkzeugvermessung** öffnet sich.
2. Nur wenn die Vermessungsansicht nach der TCP-Vermessung geschlossen wurde:  
Das montierte Werkzeug und den zugehörigen TCP des Werkzeugs auswählen.
3. Die Methode **Orientierung festlegen(ABC 2-Punkt)** auswählen. Die Messpunkte der Methode werden als Schaltflächen angezeigt:
  - **TCP**
  - **Negative X-Achse**
  - **Positiver Y-Wert auf XY-Ebene**
 Um einen Messpunkt aufnehmen zu können, muss er markiert sein (Schaltfläche ist orange).

4. Mit dem TCP einen beliebigen Referenzpunkt anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
5. Das Werkzeug so verfahren, dass der Referenzpunkt auf der X-Achse auf einem Punkt mit negativem X-Wert (d. h. entgegen der Stoßrichtung) zu liegen kommt. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
6. Das Werkzeug so verfahren, dass der Referenzpunkt auf der XY-Ebene auf einem Punkt mit positivem Y-Wert zu liegen kommt. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
7. **Werkzeugdaten bestimmen** drücken. Die Vermessungsdaten werden im Dialog **Werkzeugdaten übernehmen** angezeigt.
8. **Übernehmen** drücken, um die Vermessungsdaten zu speichern.
9. Projekt synchronisieren, um die Vermessungsdaten in Sunrise.Workbench zu übernehmen.

### 7.2.1.3 Orientierung festlegen: ABC Welt-Methode

<b>Beschreibung</b>	Der Benutzer richtet die Achsen des Werkzeug-Koordinatensystems parallel zu den Achsen des Welt-Koordinatensystems aus. Dadurch wird der Robotersteuerung die Orientierung des Werkzeug-Koordinatensystems bekanntgegeben.  Die Methode hat 2 Varianten:
	<ul style="list-style-type: none"> <li>■ <b>5D:</b> Der Benutzer gibt der Robotersteuerung die Stoßrichtung des Werkzeugs bekannt. Die Stoßrichtung ist defaultmäßig die X-Achse. Die Orientierung der anderen Achsen wird vom System festgelegt und kann vom Benutzer nicht beeinflusst werden.  Das System legt die Orientierung der anderen Achsen immer gleich fest. Falls das Werkzeug später ein weiteres Mal vermessen werden muss, z. B. nach einem Crash, reicht es deshalb, die Stoßrichtung erneut festzulegen. Auf die Verdrehung um die Stoßrichtung muss nicht geachtet werden.</li> <li>■ <b>6D:</b> Der Benutzer gibt der Robotersteuerung die Richtung aller 3 Achsen bekannt.</li> </ul> <p>Diese Methode wird bei Werkzeugen verwendet, die keine Kanten haben, an denen sich der Benutzer orientieren kann, z. B. rundliche Werkzeuge wie Klebe- oder Schweißdüsen.</p>
<b>Voraussetzung</b>	<ul style="list-style-type: none"> <li>■ Das zu vermessende Werkzeug ist am Anbauflansch montiert.</li> <li>■ Der TCP des Werkzeugs ist bereits vermessen.</li> <li>■ Betriebsart T1</li> </ul>
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. Nur wenn die Vermessungsansicht nach der TCP-Vermessung geschlossen wurde:  In der Robotersicht <b>Vermessung &gt; Werkzeugvermessung</b> wählen. Die Ansicht <b>Werkzeugvermessung</b> öffnet sich.</li> <li>2. Nur wenn die Vermessungsansicht nach der TCP-Vermessung geschlossen wurde:  Das montierte Werkzeug und den zugehörigen TCP des Werkzeugs auswählen.</li> <li>3. Die Methode <b>Orientierung festlegen(ABC Welt)</b> auswählen.</li> <li>4. Die Option <b>ABC Welt 5D</b> oder <b>ABC Welt 6D</b> auswählen.</li> <li>5. Wenn <b>ABC Welt 5D</b> gewählt wurde:  +X<sub>TOOL</sub> parallel zu -Z<sub>WORLD</sub> ausrichten. (+X<sub>TOOL</sub> = Stoßrichtung)</li> </ol>

Wenn **ABC Welt 6D** gewählt wurde:

Die Achsen des Werkzeug-Koordinatensystems folgendermaßen ausgerichtet.

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

6. **Werkzeugdaten bestimmen** drücken. Die Vermessungsdaten werden im Dialog **Werkzeugdaten übernehmen** angezeigt.
7. **Übernehmen** drücken, um die Vermessungsdaten zu speichern.
8. Projekt synchronisieren, um die Vermessungsdaten in Sunrise.Workbench zu übernehmen.

### 7.2.2 Basis vermessen: 3-Punkt-Methode

#### Beschreibung

Bei der Basisvermessung weist der Benutzer einem Frame, der als Basis markiert ist, ein kartesisches Koordinatensystem (Basis-Koordinatensystem) zu. Das Basis-Koordinatensystem hat seinen Ursprung in einem vom Benutzer festgelegten Punkt.

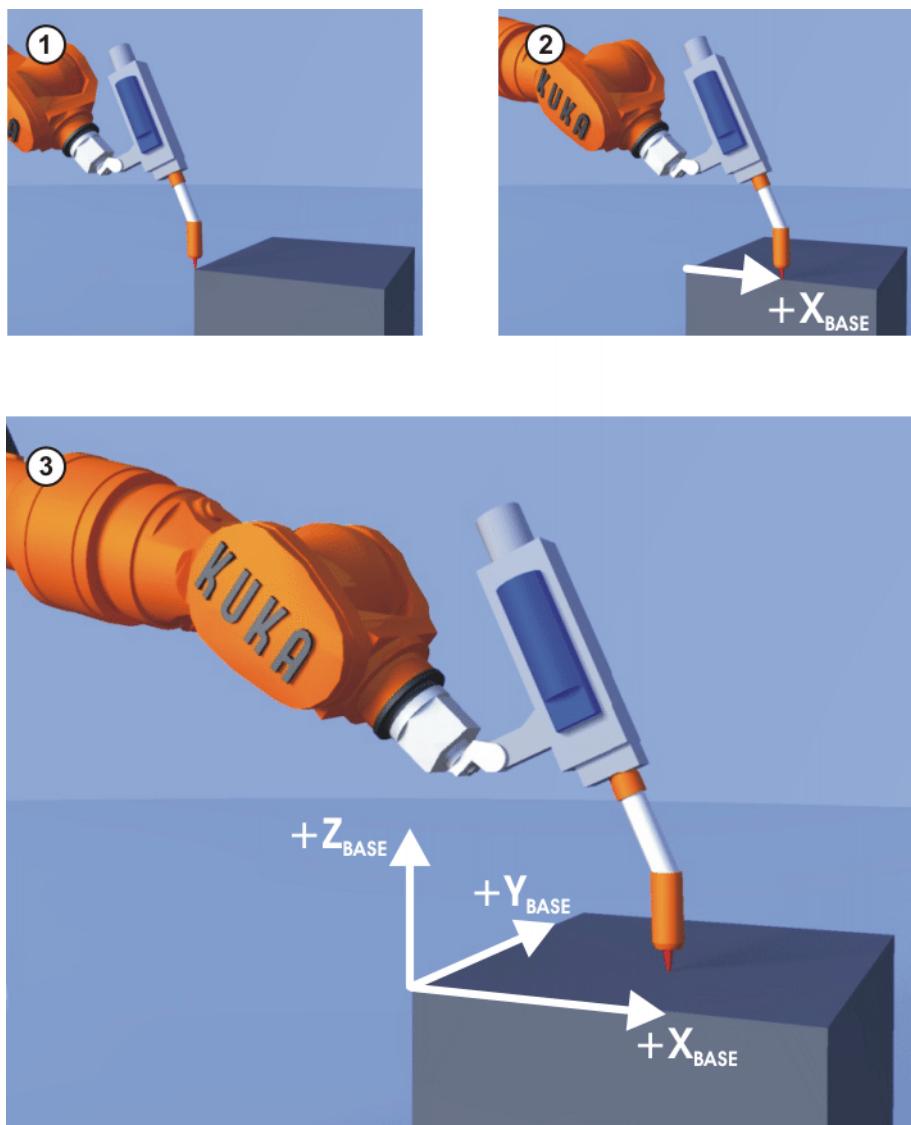
Vorteile der Basisvermessung:

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

Bei der 3-Punkt-Methode werden der Ursprung und 2 weitere Punkte einer Basis angefahren. Diese 3 Punkte definieren die Basis.

Die Punkte müssen einen bestimmten Mindestabstand zum Ursprung und Mindestwinkel zwischen den Geraden (Ursprung – X-Achse und Ursprung – XY-Ebene) einhalten. Liegen die Punkte zu nah beieinander oder ist der Winkel zwischen den Geraden zu klein, können die Positionsdaten nicht gespeichert werden. Eine entsprechende Fehlermeldung wird ausgegeben.

Es ist möglich, Mindestabstand und -winkel in Sunrise.Workbench zu ändern.  
(>>> 10.1.1 "Parameter für Vermessung konfigurieren" Seite 159)



**Abb. 7-4: 3-Punkt-Methode**

#### Voraussetzung

- Ein bereits vermessenes Werkzeug ist am Anbauflansch montiert.
- Der zu vermessende Frame wurde in den Applikationsdaten des Projekts als Basis markiert und per Synchronisation auf die Robotersteuerung übertragen.
- Betriebsart T1

#### Vorgehensweise

1. In der Robotersicht **Vermessung > Basisvermessung** wählen. Die Ansicht **Basisvermessung** öffnet sich.
  2. Die zu vermessende Basis auswählen.
  3. Das montierte Werkzeug und den TCP des Werkzeugs auswählen, mit denen die Messpunkte der Basis angefahren werden.
- Die Messpunkte der 3-Punkt-Methode werden als Schaltflächen angezeigt:
- **Ursprung**
  - **Positive X-Achse**
  - **Positiver Y-Wert auf XY-Ebene**

Um einen Messpunkt aufnehmen zu können, muss er markiert sein (Schaltfläche ist orange).

4. Mit dem TCP den Ursprung der Basis anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
5. Mit dem TCP einen Punkt auf der positiven X-Achse der Basis anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
6. Mit dem TCP auf der XY-Ebene einen Punkt mit positivem Y-Wert anfahren. **Messpunkt aufnehmen** drücken. Die Positionsdaten werden für den markierten Messpunkt übernommen und angezeigt.
7. **Basisdaten bestimmen** drücken. Die Vermessungsdaten werden im Dialog **Basisdaten übernehmen** angezeigt.
8. **Übernehmen** drücken, um die Vermessungsdaten zu speichern.
9. Projekt synchronisieren, um die Vermessungsdaten in Sunrise.Workbench zu übernehmen.

### 7.3 Werkzeug-Lastdaten ermitteln

#### Beschreibung

Bei der Lastdatenermittlung führt der Roboter mehrere Messfahrten mit unterschiedlichen Orientierungen der Handachsen A5, A6 und A7 durch. Die Lastdaten werden aus den Daten berechnet, die während der Messfahrten aufgezeichnet werden.

Aktuell können die Masse und die Lage des Massen-Schwerpunkts des am Roboterflansch montierten Werkzeugs ermittelt werden. Es ist auch möglich, die Masse vorzugeben und auf Basis der bereits bekannten Masse die Lage des Massen-Schwerpunkts zu ermitteln.

Zu Beginn der Lastdatenermittlung wird die Achse A7 in die Nullposition gefahren und die Achse A5 so eingestellt, dass die Achse A6 senkrecht zur Gewichtskraft ausgerichtet ist. Während der Messfahrten muss die Achse A6 zwischen -95° und +95° bewegt werden können, die Achse A7 von 0° bis -90°.

Die restlichen Roboterachsen (A1 bis A4) werden bei der Lastdatenermittlung nicht bewegt. Sie bleiben während der Messung in Ausgangsstellung.

Die Qualität der Lastdatenermittlung kann durch folgende Randbedingungen beeinflusst werden:

- Masse des Werkzeugs

Die Lastdatenermittlung wird mit zunehmender Werkzeugmasse zuverlässiger, da sich Messunsicherheiten bei einer geringeren Masse stärker auswirken.



Die Lastdatenermittlung kann derzeit für Massen von unter einem Kilogramm noch nicht zuverlässig eingesetzt werden.

- Zusatzlasten

Die Lastdatenermittlung ist nur für Lasten geeignet, die fest mit dem Roboterflansch verbunden sind. Zusatzlasten, d. h. Lasten, die auf der Roboterstruktur angebracht sind, z. B. Schlauchpakete, werden von der Lastdatenermittlung nicht unterstützt.

- Startposition, von der aus die Lastdatenermittlung gestartet wird

Eine geeignete Startposition sollte vorab ermittelt werden und folgende Kriterien erfüllen:

- Die Achsen A1 bis A5 sind möglichst weit entfernt von singulären Stellungen.

Das Kriterium ist relevant, wenn bei der Lastdatenermittlung die Masse ermittelt wird. Ist die Lastdatenermittlung nur in Posen möglich, bei denen die Achsen A1 bis A5 in der Nähe singulärer Stellungen sind, kann die Masse vorgegeben werden. Wird nur der Massen-Schwer-

punkt auf Basis der vorgegebenen Masse ermittelt, ist das Kriterium der Achsstellung irrelevant.

- Die Eignung der Startposition zur Lastdatenermittlung für den Roboter, mit dem eine automatische Ermittlung durchgeführt werden soll, muss vor der Montage der zu ermittelnden Last am Roboter geprüft werden.

Eine Roboterpose ist als Startposition für die Lastdatenermittlung geeignet, wenn in dieser Position nur sehr geringe externe Momente (idealerweise < 0,5 Nm) auf den lastfreien Roboter wirken. Dies kann über die Anzeige der externen Achsmomente überprüft werden.

(>>> 6.16.4 "Achsspezifische Momente anzeigen" Seite 98)

Wenn bei der Lastdatenermittlung die Masse ermittelt wird, sind alle externen Achsmomente relevant und möglichst vorab für den lastfreien Roboter zu überprüfen. Wird nur der Massen-Schwerpunkt auf Basis einer vorgegebenen Masse ermittelt, ist nur das externe Achsmoment der Achse A6 relevant.

- Störmomente während der Messfahrten
  - Während der Messfahrten dürfen keine Störmomente, z. B. durch Ziehen oder Drücken am Roboter, aufgebracht werden.
  - Bewegliche Teile, z. B. Schlauchpakete, generieren störende Drehmomente, die den Massen-Schwerpunkt während der Messfahrt verschieben.



Das Aufbringen von Störmomenten bei der Lastdatenermittlung führt zu falschen Lastdaten. (>>> 9.3.6 "Lastdaten" Seite 147)

Bei der Lastdatenermittlung für sicherheitsgerichtete Werkzeuge muss beachtet werden, dass die geänderten Lastdaten nicht automatisch in die Sicherheitskonfiguration, die sich auf der Robotersteuerung befindet, übernommen werden. (>>> 9.3.7 "Sicherheitsgerichtetes Werkzeug" Seite 148)

Per Projekt-Synchronisation müssen die für ein sicherheitsgerichtetes Werkzeug ermittelten Lastdaten zunächst in Sunrise.Workbench aktualisiert werden. Dadurch ändert sich die Sicherheitskonfiguration des Projekts in Sunrise.Workbench, das dann durch eine erneute Synchronisation des Projekts auf die Robotersteuerung rückübertragen werden muss.



Wird eine geänderte Sicherheitskonfiguration auf der Robotersteuerung aktiviert, muss der Sicherheitsinbetriebnehmer eine Sicherheitsabnahme durchführen. (>>> 13.11 "Übersicht Sicherheitsabnahme" Seite 243)

Um unnötigen Verifikationsaufwand zu vermeiden, wird empfohlen, ein Werkzeug erst dann als sicherheitsgerichtet zu markieren, wenn die Lastdaten korrekt eingegeben oder ermittelt und in Sunrise.Workbench übertragen wurden.

## Vorbereitung

- Startposition ermitteln, von der aus die Lastdatenermittlung gestartet wird.

## Voraussetzung

- Werkzeug ist am Anbauflansch montiert.  
Werkzeug wurde in den Objektvorlagen des Projekts angelegt und per Synchronisation auf die Robotersteuerung übertragen.
- Roboter steht in der gewünschten Startposition.
- Im Handachsen-Bereich ist ein ausreichend großer Arbeitsraum verfügbar.
- Betriebsart T1, T2 oder AUT

**HINWEIS**

Reichen Teile des montierten Werkzeugs hinter die Flanschebene (in negative Z-Richtung bezüglich Flansch-Koordinatensystem) besteht die Gefahr, dass das Werkzeug während der Messfahrten mit dem Manipulator kollidiert. Ist dem Bediener die Bewegung der Achsen bei der Lastdatenermittlung nicht bekannt oder kann, z. B. bei erstmaliger Lastdatenermittlung für ein Werkzeug, eine Kollision zwischen Werkzeug und Manipulator nicht ausgeschlossen werden, wird empfohlen, die Lastdatenermittlung in der Betriebsart T1 durchzuführen. Die Güte der Messwerte wird dadurch nicht beeinflusst.

**Vorgehensweise**

1. In der Robotersicht **Lastdaten** wählen. Die Ansicht **Lastdaten** öffnet sich.
2. Das montierte Werkzeug in der Auswahliste auswählen.
3. Wenn die Betriebsart T1 oder T2 eingestellt ist, Zustimmungsschalter drücken und halten bis die Lastdatenermittlung abgeschlossen ist.
4. **Lastdaten ermitteln** drücken.
5. Besitzt das Werkzeug bereits eine Masse, wird der Bediener gefragt, ob die Masse neu ermittelt werden soll.
  - **Masse beibehalten** wählen, wenn die aktuell gespeicherte Masse beibehalten werden soll.
  - **Masse neu ermitteln** wählen, wenn die Masse neu ermittelt werden soll.



Wird die aktuell gespeicherte Masse bei der Lastdatenermittlung beibehalten, ist darauf zu achten, dass der angegebene Wert der Masse korrekt ist. Andernfalls kann der Masseschwerpunkt nicht genau ermittelt werden.

6. Der Roboter startet mit den Messfahrten und die Lastdaten werden ermittelt. Ein Fortschrittsbalken wird angezeigt.



Wenn die Fahrfreigabe während der Lastdatenermittlung entzogen wird, z. B. durch einen NOT-HALT oder durch Loslassen des Zustimmungsschalters (T1, T2), wird die Lastdatenermittlung abgebrochen und muss neu gestartet werden.

Wenn die Lastdatenermittlung abgeschlossen ist, werden die ermittelten Lastdaten im Dialog **Lastdaten übernehmen** angezeigt.

**Übernehmen** drücken, um die ermittelten Lastdaten zu speichern.

7. Projekt synchronisieren, damit die Lastdaten in Sunrise.Workbench übernommen werden.  
Wenn die Lastdaten für ein sicherheitsgerichtetes Werkzeug ermittelt wurden, ändert sich durch die Projekt-Synchronisation die Sicherheitskonfiguration.
8. Bei Bedarf Projekt synchronisieren, um die geänderte Sicherheitskonfiguration auf die Robotersteuerung zu übertragen.

## Übersicht

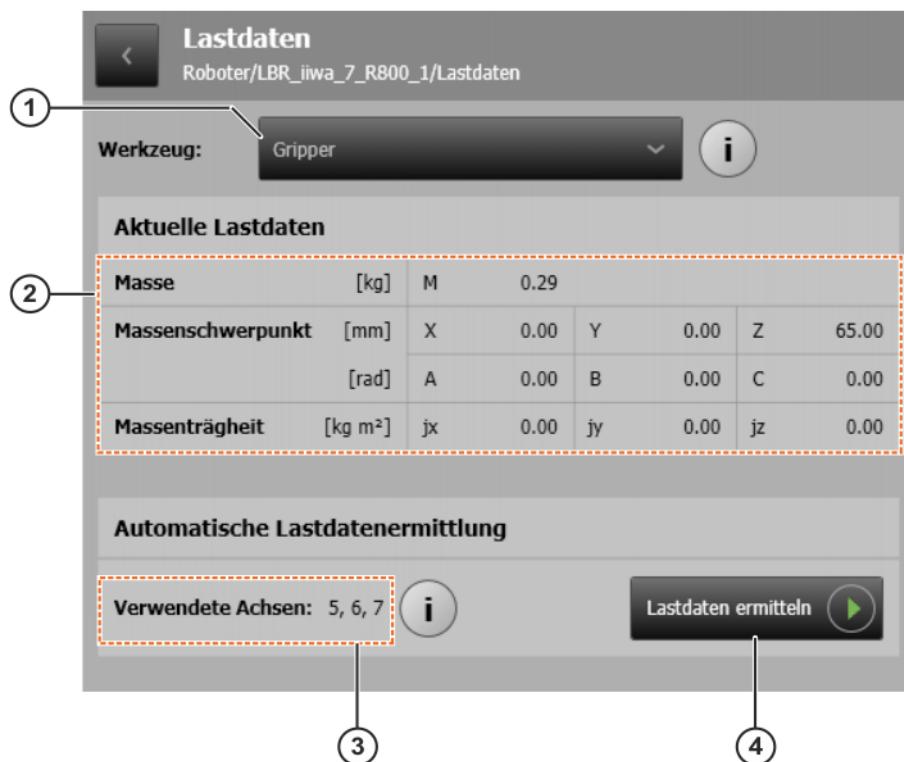


Abb. 7-5: Lastdaten ermitteln

Pos.	Beschreibung
1	Auswahlliste Werkzeug Hier stehen die in den Objektvorlagen angelegten Werkzeuge zur Auswahl.
2	Anzeige Lastdaten Zeigt die aktuellen Lastdaten des ausgewählten Werkzeugs an.
3	Anzeige verwendete Achsen Zeigt die Achsen an, die für die Lastdatenermittlung bewegt werden.
4	Button <b>Lastdaten ermitteln</b> Startet die Lastdatenermittlung. Der Button ist nur aktiv, wenn ein Werkzeug ausgewählt und die Fahrfreigabe erteilt wurde.



## 8 Bremsentest

### 8.1 Übersicht Bremsentest

**Beschreibung** Jede Roboterachse hat eine in den Antriebsstrang integrierte Haltebremse. Die Bremsen haben 2 Funktionen:

- Halten des Roboters bei abgeschalteter Regelung oder im spannungsfreien Zustand.
- Überführen des Roboters in den sicheren Zustand "Stillstand" im Falle eines Fehlers.

Mit dem Bremsentest wird für jede Bremse geprüft, ob das von ihr aufgebrachte Bremsenhaltemoment ausreichend hoch ist, d. h. ob es ein bestimmtes Referenzmoment überschreitet. Dieses Referenzmoment kann vom Programmierer vorgegeben werden oder wird aus den Motordaten ausgelesen.



Sofern durch eine Risikobetrachtung nicht anders bestimmt, muss der Bremsentest regelmäßig durchgeführt werden:

- Der Bremsentest ist bei der Inbetriebnahme und Wiederinbetriebnahme des Industrieroboters für jede Achse durchzuführen.
- Während des Betriebs ist der Bremsentest täglich durchzuführen.

Der Benutzer kann durch eine Risikobetrachtung ermitteln, ob es in seinem konkreten Anwendungsfall erforderlich ist, den Bremsentest durchzuführen und in welcher Regelmäßigkeit er durchzuführen ist.

**Durchführung** Voraussetzung für die Durchführung des Bremsentests ist, dass der Roboter Betriebstemperatur hat.

Der Bremsentest wird manuell über eine Applikation ausgeführt. KUKA Sunrise.Workbench stellt eine vorbereitete Bremsentest-Applikation für den LBR iiwa zur Verfügung.

Wird die vorbereitete Bremsentest-Applikation verwendet, wird der Roboter vor dem eigentlichen Bremsentest verfahren und das dabei auftretende maximale absolute Drehmoment für jede Achse ermittelt. Das ermittelte Moment wird in der Bremsentest-Applikation als Referenzhaltemoment an den Bremsentest übergeben.

Die Ermittlung der maximalen absoluten Drehmomente wird im Weiteren als Momentenwert-Ermittlung bezeichnet.



Es wird empfohlen, die Momentenwert-Ermittlung aus der Applikation zu entfernen und die Bremsen gegen das minimale Bremsenhaltemoment zu testen. Wird gegen ein vom Programmierer vorgegebenes Drehmoment getestet, muss im Voraus durch eine Risikoanalyse beurteilt werden, ob Gefährdungen entstehen, wenn die Bremsen ein niedrigeres Moment als das minimale Bremsenhaltemoment aufweisen.

**Ablauf** Beim Test einer Bremse werden defaultmäßig folgende Schritte ausgeführt:

1. Die Achse verfährt mit konstanter Geschwindigkeit über einen kleinen Achswinkel von maximal 5° (abtriebsseitig). Dabei werden Reibung und Gravitation ermittelt.
2. Wenn die Achse wieder ihre Ausgangsposition erreicht hat und der Antrieb der Achse stillsteht, wird die Bremse geschlossen.
3. Als das zu testende Haltemoment wird einer von folgenden Werten verwendet: Das ermittelte Referenzhaltemoment, das minimale Bremsenhaltemoment oder das Motorhaltemoment.

Das zu testende Haltemoment wird systemintern nach folgenden Regeln festgelegt:

- a. Wenn das Referenzhaltemoment größer als das Minimum aus minimalem Bremsenhaltemoment und Motorhaltemoment ist, wird das Minimum von minimalen Bremsenhaltemoment und Motorhaltemoment als zu testendes Haltemoment verwendet.
- b. Wenn das Referenzhaltemoment kleiner als 20 % des Minimums aus minimalem Bremsenhaltemoment und Motorhaltemoment ist, werden 20 % des Minimums aus minimalem Bremsenhaltemoment und Motorhaltemoment als zu testendes Haltemoment verwendet.
- c. In allen anderen Fällen wird das Referenzhaltemoment verwendet.

Der Bremsentest setzt zu Beginn des Tests das Sollmoment des Antriebs bei geschlossener Bremse auf 80 % des zu testenden Haltemoments.



Das minimale und maximale Bremsenhaltemoment ist in den Motordaten gespeichert. Das Motorhaltemoment wird aus den Motordaten abgeleitet.

4. Das Antriebsmoment wird sukzessive erhöht bis eine Positionsänderung festgestellt wird oder das maximale Bremsenhaltemoment (Wert wird aus den Motordaten ermittelt) erreicht ist. Bei Erreichen des maximalen Bremsenhaltemoments endet der Bremsentest.
5. Das gegen die Bremse aufgebrachte Moment bei Feststellung einer Positionsänderung wird gemessen. Dies ist das gemessene Haltemoment.
6. Das gemessene Haltemoment wird bezogen auf das zu testende Haltemoment ausgewertet.

Der Bremsentest ist erfolgreich, wenn das gemessene Haltemoment in folgendem Bereich liegt:

- $\geq 105\% \text{ des zu testenden Haltemoments} \dots \leq \text{maximales Bremsenhaltemoment}$

Liegt das gemessene Haltemoment unter dem zu testenden Haltemoment, ist der Bremsentest fehlgeschlagen, d. h. die Bremse wurde als defekt erkannt.

Das Testergebnis wird auf der smartHMI angezeigt.

(>>> 8.4.2 "Ergebnis des Bremsentests (Anzeige)" Seite 134)

7. Wenn der Bremsentest beendet ist und der Roboter stillsteht, wird die Bremse kurz geöffnet und wieder geschlossen. Mögliche verbleibende Verspannungen in der Bremse werden dadurch gelöst und ungewollte Bewegungen des Roboters vermieden.



Ist der Bremsentest für eine Achse fehlgeschlagen, d. h. eine Bremse wurde als defekt erkannt, kann er zur Kontrolle wiederholt werden.



Wird die Applikation während des Bremsentests pausiert oder ein Sicherheitshalt ausgelöst, z. B. durch einen NOT-HALT, wird der Bremsentest abgebrochen. Nach dem Fortsetzen der Applikation wird der Bremsentest für die Achse wiederholt.



Der Bremsentest ist unabhängig von am Roboter montierten Lasten, da Gravitation und Reibung beim Ausführen des Tests berücksichtigt werden.

## Übersicht

Hier sind die Schritte beschrieben, um den Bremsentest mit der in Sunrise.Workbench zur Verfügung gestellten Vorlage durchzuführen.

Die Bremsentest-Applikation kann angepasst und erweitert werden. Die in der Vorlage enthaltenen Kommentare sind dabei zu beachten.

**HINWEIS** Wenn eine Bremse defekt ist, kann die zugehörige Achse beim Bremsentest durchrutschen und der Roboter zusammensacken. Der Bremsentest muss in einer Position durchgeführt werden, in der nach Möglichkeit kein Schaden durch ein Zusammensacken entstehen kann. Die Ausgangsposition für den Bremsentest ist entsprechend zu wählen.

**HINWEIS** Ist der Bremsentest für eine Achse fehlgeschlagen, d. h. eine Bremse wurde als defekt erkannt, muss über die Applikation sichergestellt werden, dass der Roboter automatisch in eine sichere Position gefahren wird. Eine sichere Position ist eine Position, in der der Roboter so abgestützt ist, dass er entweder nicht zusammensacken kann oder durch ein Zusammensacken kein Schaden entstehen kann.

Schritt	Beschreibung
1	Bremsentest-Applikation aus Vorlage erstellen.  (>>> 8.2 "Bremsentest-Applikation aus Vorlage erstellen" Seite 120)
2	Ermittlung der applikationspezifisch auftretenden maximalen absoluten Momente in Bremsentest-Applikation entfernen oder anpassen.  Zu Beginn der Bremsentest-Applikation werden defaultmäßig 2 vordefinierte Achspositionen angefahren, um das entstehende maximale absolute Drehmoment für jede Achse zu ermitteln und als Referenzhaltemoment an den Bremsentest zu übergeben.  Es wird empfohlen, die Bremsen gegen das minimale Bremsenhaltemoment zu testen, das in den Motordaten abgelegt ist. Dafür muss die vorbereitete Bremsentest-Applikation angepasst werden.  (>>> 8.2.1 "Bremsentest-Applikation für Test gegen minimales Bremsenhaltemoment anpassen" Seite 122)  Werden für den Bremsentest die maximalen absoluten Momente benötigt, die beim Ablauf einer benutzerspezifischen Roboter-Applikation auftreten, kann die benutzerspezifische Roboter-Applikation in die Bremsentest Applikation eingefügt werden. Da die Bremsen in diesem Fall nicht gegen das minimale Bremsenhaltemoment getestet werden, ist im Voraus eine Risikoanalyse durchzuführen.  (>>> 8.2.2 "Bewegungsfolge für Momentenwert-Ermittlung ändern" Seite 123)
3	Ausgangsposition für den Bremsentest ändern.  Die Ausgangsposition ist defaultmäßig die Kerzenstellung. Bei Bedarf kann eine andere Ausgangsposition angefahren werden.  (>>> 8.2.3 "Ausgangsposition für Bremsentest ändern" Seite 123)

Schritt	Beschreibung
4	<p>Bei Bedarf weitere benutzerspezifische Anpassungen in der Bremsentest-Applikation vornehmen.</p> <p>Beispiele:</p> <ul style="list-style-type: none"> <li>■ Ausgang bei einem fehlgeschlagenen Bremsentest setzen.</li> <li>■ Testergebnisse in einer Datei speichern.</li> </ul> <p>(&gt;&gt;&gt; 8.3 "Programmierschnittstelle für den Bremsentest" Seite 124)</p>
5	Projekt synchronisieren, um die Bremsentest-Applikation auf die Robotersteuerung zu übertragen.
6	Bremsentest-Applikation ausführen. (>>> 8.4 "Bremsentest durchführen" Seite 133)

## 8.2 Bremsentest-Applikation aus Vorlage erstellen

- |                       |   |
|-----------------------|---|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"> <li>1. Sunrise-Projekt im <b>Paket-Explorer</b> markieren.</li> <li>2. Menüfolge <b>Datei &gt; Neu &gt; Andere...</b> wählen.</li> <li>3. Im Ordner <b>Sunrise</b> die Option <b>Applikation für den Bremsentest des LBR iiwa</b> markieren und auf <b>Fertigstellen</b> klicken.</li> </ol> <p>Die Applikation <b>BrakeTestApplication.java</b> wird im Quellenordner des Projekts erstellt und im Editoren-Bereich von Sunrise.Workbench geöffnet.</p> |
| <b>Beschreibung</b>   | <p>In der Methode <code>run()</code> der Applikation <b>BrakeTestApplication.java</b> (hier auf die relevanten Befehlszeilen beschränkt) ist die Ausführung des Bremsentests für alle Achsen des LBR iiwa implementiert.</p> <p>Ebenfalls implementiert ist eine optionale Datenauswertung, die dem eigentlichen Bremsentest vorausgeht. Es werden 2 vordefinierte Achspositionen angefahren, um für jede Achse das maximale absolute Drehmoment zu ermitteln.</p>                          |

```

1 public void run() {
2     ...
3     lbr_iiwa.move(ptpHome());
4     ...
5     TorqueEvaluator evaluator = new TorqueEvaluator(lbr_iiwa);
6     ...
7     evaluator.setTorqueMeasured(false);
8
9     evaluator.startEvaluation();
10    ...
11    lbr_iiwa.move(new PTP(new JointPosition(
12        0.5, 0.8, 0.2, 1.0, -0.5, -0.5, -1.5)).
13        setJointVelocityRel(relVelocity));
14    lbr_iiwa.move(new PTP(new JointPosition(
15        -0.5, -0.8, -0.2, -1.0, 0.5, 0.5, 1.5)).
16        setJointVelocityRel(relVelocity));
17    ...
18    TorqueStatistic maxTorqueData = evaluator.stopEvaluation();
19
20    boolean allAxesOk = true;
21
22    for (int axis : axes) {
23        try {

```

```

24         BrakeTest brakeTest = new BrakeTest(axis,
25             maxTorqueData.getMaxAbsTorqueValues()[axis]);
26         IMotionContainer motionContainer = lbr_iowa.move(brakeTes
27         t);
28         BrakeTestResult brakeTestResult =
29             BrakeTest.evaluateResult(motionContainer);
30         switch(brakeTestResult.getState().getLogLevel())
31         {
32             case Info:
33                 getLogger().info(brakeTestResult.toString());
34                 break;
35             case Warning:
36                 getLogger().warn(brakeTestResult.toString());
37                 break;
38             case Error:
39                 getLogger().error(brakeTestResult.toString());
40                 allAxesOk = false;
41                 break;
42             default:
43                 break;
44         }
45         catch (CommandInvalidException ex) {
46             ...
47             allAxesOk = false;
48         }
49     }
50     if (allAxesOk) {
51         getLogger().info("Brake test was successful for all axes.");
52     }
53     else{
54         getLogger().error("Brake test failed for at least one
55         axis.");
56     }

```

Zeile	Beschreibung
3	Ausgangsposition anfahren, von der aus der Roboter verfahren wird, um das maximale absolute Drehmoment für jede Achse zu ermitteln.  Die Ausgangsposition ist defaultmäßig die Kerzenstellung.
5	Datenauswertung vorbereiten.  Um die Drehmomente, die während einer Bewegungsfolge ermittelt werden, achsspezifisch auswerten zu können, muss eine Instanz der Klasse TorqueEvaluator erzeugt werden.
7	Momente auswählen, die für die Datenauswertung verwendet werden.  Es werden nicht die gemessenen Drehmomente verwendet, sondern die Drehmomente, die während der Berwegungsfolge mithilfe des Robotermodeells berechnet werden. Messungen sind störanfällig. Die Berechnung der Drehmomentwerte stellt sicher, dass keine Störmomente in die Datenauswertung einfließen, die aufgrund dynamischer Effekte (z. B. Roboterbeschleunigung) entstehen.
9	Datenauswertung starten.  Über den Befehl startEvaluation() der Klasse TorqueEvaluator wird die Datenauswertung gestartet.

Zeile	Beschreibung
11 ... 16	Bewegungsfolge zur Ermittlung der maximalen absoluten Drehmomente  Es werden 2 vordefinierte Achspositionen jeweils mit einer PTP-Bewegung angefahren.
18	Datenauswertung beenden und Daten abfragen.  Der Befehl stopEvaluation() der Klasse TorqueEvaluator beendet die Datenauswertung und gibt das Ergebnis als Wert vom Typ TorqueStatistic zurück. Das Ergebnis wird in der Variablen maxTorqueData gespeichert.
20	Variable für die Auswertung des Bremsentests  Über die Variable allAxesOk wird das Ergebnis der Bremsentests für die spätere Auswertung gespeichert. Sie wird auf false gesetzt, wenn der Bremsentest einer Achse fehlschlägt oder durch einen Fehler abgebrochen wird. Andernfalls wird der Wert true beibehalten.
22 ... 54	Ausführung des Bremsentests  Die Bremsen werden, beginnend bei der Bremse der Achse A1, nacheinander getestet. <ol style="list-style-type: none"> <li>Zeile 24, 25: Es wird ein Objekt vom Typ BrakeTest erzeugt und dabei die jeweilige Achse sowie das zuvor ermittelte maximale absolute Drehmoment als Referenzhaltemoment übergeben.</li> <li>Zeile 26: Der Bremsentest wird als Bewegungsbefehl ausgeführt.</li> <li>Zeile 27 ... 54: Das Ergebnis des Bremsentests wird ausgewertet und auf der smartHMI ausgegeben.</li> </ol>

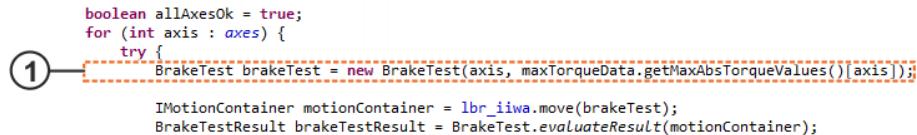
### 8.2.1 Bremsentest-Applikation für Test gegen minimales Bremsenhaltemoment anpassen

#### Beschreibung

Beim Bremsentest soll geprüft werden, ob die Bremsen das minimale Bremsenhaltemoment aufbringen. Daher wird empfohlen, die vorbereitete Bremsentest-Applikation gemäß der folgenden Beschreibung anzupassen.

Soll der Bremsentest ausgeführt werden, ohne dass Referenzhaltemomente ermittelt und an den Bremsentest übergeben werden, müssen alle für die Momentenwert-Ermittlung relevanten Befehlszeilen aus der Bremsentest-Applikation entfernt werden. Die Bremsentest-Applikation startet dann folglich mit der Bewegung zur Ausgangsposition für den Bremsentest.

Zusätzlich muss bei der Erzeugung der BrakeTest-Instanz der Parameter entfernt werden, mit dem das Referenzmoment übergeben wird.



```

boolean allAxesOk = true;
for (int axis : axes) {
    try {
        BrakeTest brakeTest = new BrakeTest(axis, maxTorqueData.getMaxAbsTorqueValues()[axis]);
        IMotionContainer motionContainer = lbr_iwa.move(brakeTest);
        BrakeTestResult brakeTestResult = BrakeTest.evaluateResult(motionContainer);
    }
}
  
```

Abb. 8-1: Übergabe Referenzmoment für Bremsentest

- Konstruktor der Klasse BrakeTest mit Übergabe Referenzmoment

#### Vorgehensweise

- Bremsentest-Applikation in Sunrise.Workbench öffnen.
- In der run()-Methode der Applikation folgende Änderungen vornehmen:
  - Alle für die Momentenwert-Ermittlung relevanten Befehlszeilen löschen.

- Im Aufruf des Konstruktors der Klasse BrakeTest folgenden Parameter löschen:

```
maxTorqueData.getMaxAbsTorqueValues() [axis]
```

In der Zeile bleibt folgender Code erhalten:

```
BrakeTest brakeTest = new brakeTest(axis);
```

3. Änderungen speichern.

### 8.2.2 Bewegungsfolge für Momentenwert-Ermittlung ändern

#### Beschreibung

Die aus der Vorlage erstellte Bremsentest-Applikation, enthält eine vorbereitete Bewegungsfolge zur Ermittlung der maximalen absoluten Drehmomente, die in jeder Achse entstehen.

Der Roboter wird defaultmäßig von der Kerzenstellung aus verfahren. Es kann eine andere Ausgangsposition gewählt werden.

Von der Ausgangsposition aus werden 2 vordefinierte Achspositionen jeweils mit einer PTP Bewegung angefahren. Um die in einer spezifischen Roboter-Applikation maximalen absoluten Drehmomente zu ermitteln und als Referenzhaltemomente für den Bremsentest nutzen zu können, muss die Roboter-Applikation in die Bremsentest-Applikation eingefügt werden.

```

public void run() {
    // initial position
    getLogger().info("Moving into initial position.");
    lbr_iiwa.move(ptpHome);

    // start monitoring for maximum torques
    getLogger().info("Start evaluation of torque statistic.");
    TorqueEvaluator evaluator = new TorqueEvaluator(lbr_iiwa);

    // select static model torque values
    evaluator.setTorqueMeasured(false);

    evaluator.startEvaluation();

    getLogger().info("Perform desired robot motion:");
    lbr_iiwa.move(new PTP(new JointPosition(0.5, 0.8, 0.2, 1.0, -0.5, -0.5, -1.5)).setJointVelocityRel(relVelocity));
    lbr_iiwa.move(new PTP(new JointPosition(-0.5, -0.8, -0.2, -1.0, 0.5, 0.5, 1.5)).setJointVelocityRel(relVelocity));

    // stop monitoring for maximum torques and store results
    getLogger().info("Stop evaluation of torque statistic.");
    TorqueStatistic maxTorqueData = evaluator.stopEvaluation();
    getLogger().info("The result of evaluation:\n" + maxTorqueData.toString());
}

```

**Abb. 8-2: Bewegungsfolge für Momentenwert-Ermittlung**

- 1 ptpHome()-Bewegung zur Ausgangsposition
- 2 Vordefinierte Bewegungsfolge für Momentenwert-Ermittlung

#### Vorgehensweise

1. Bremsentest-Applikation in Sunrise.Workbench öffnen.
2. In der run()-Methode der Applikation bei Bedarf folgende Änderungen vornehmen:
  - Die ptpHome()-Bewegung, die den Roboter in die Ausgangsposition bringt, durch eine Bewegung zur gewünschten Ausgangsposition ersetzen.
  - Die vordefinierte Bewegungsfolge durch eigenen Applikationscode ersetzen.
3. Änderungen speichern.

### 8.2.3 Ausgangsposition für Bremsentest ändern

#### Beschreibung

Die aus der Vorlage erstellte Bremsentest-Applikation führt den Bremsentest defaultmäßig an der Endposition der Bewegungsfolge zur Ermittlung des maximalen absoluten Drehmoments aus. Ist diese Position als Ausgangsposition für den Bremsentest nicht geeignet, muss vor der Ausführung des Bremsentests eine Bewegung an die gewünschte Ausgangsposition programmiert werden.



Zur Ermittlung von Gravitation und Reibung werden die Achsen eines LBR iiwa in Richtung mechanische Nullstellung verfahren. Der maximale Verfahrweg beträgt dabei 5° abtriebsseitig.

## 8.3 Programmierschnittstelle für den Bremsentest

Die RoboticsAPI bietet mit der Klasse BrakeTest eine Programmierschnittstelle für die Durchführung des Bremsentests. Der Bremsentest wird als Bewegungskommando ausgeführt.

Außerdem können mithilfe der Klasse TorqueEvaluator die Drehmomente, die während einer Bewegungsfolge gemessen werden, ausgewertet und das maximale absolute Drehmoment für jede Achse ermittelt werden. Dieses Moment kann als Referenzhaltemoment für den Bremsentest genutzt werden.

### 8.3.1 Auftretende Momente auswerten und maximalen Absolutwert ermitteln

#### Beschreibung

Um die Drehmomente, die während einer Bewegungsfolge ermittelt werden, achsspezifisch auswerten zu können, muss zunächst ein Objekt der Klasse TorqueEvaluator erzeugt werden. Dem Konstruktor der Klasse TorqueEvaluator wird die LBR-Instanz übergeben, für deren Achsen die maximalen absoluten Drehmomentwerte ermittelt werden sollen.

Mit folgenden Methoden der Klasse TorqueEvaluator kann die Auswertung gestartet und wieder beendet werden:

- `startEvaluation()`: Startet die Auswertung.

Nach dem Aufruf der Methode muss die auszuwertende Bewegungsfolge kommandiert werden.

- `stopEvaluation()`: Beendet die Auswertung.

Die Methode gibt ein Objekt vom Typ TorqueStatistic zurück, über das das Ergebnis der Auswertung abgefragt werden kann.

Die während der Bewegungsabfolge auftretenden Drehmomentwerte können auf verschiedene Arten ermittelt werden:

- Gemessene Momente: Es werden die Drehmomente verwendet, die von den Gelenkmomentensensoren gemessen werden.
- Statische Momente (modellbasiert): Es werden die Drehmomente verwendet, die mithilfe des statischen Robotermodells berechnet werden.

Mit der Methode `setTorqueMeasured(...)` der Klasse TorqueEvaluator kann festgelegt werden, ob für die Auswertung die gemessenen Momente oder die statischen Momente (modellbasiert) verwendet werden sollen.

#### Syntax

```
TorqueEvaluator evaluator = new TorqueEvaluator(lbr_iwa);  
evaluator.setTorqueMeasured(isTorqueMeasured);  
evaluator.startEvaluation();  
// Bewegungsfolge  
TorqueStatistic maxTorqueData = evaluator.stopEvaluation();
```

## Erläuterung der Syntax

Element	Beschreibung
<code>evaluator</code>	Typ: TorqueEvaluator  Variable, der die erzeugte TorqueEvaluator-Instanz zugewiesen wird. Über die Variable wird die Auswertung der Drehmomente während einer Bewegungsfolge gestartet und beendet.
<code>isTorqueMeasured</code>	Typ: boolean  Eingangsparameter der Methode <code>setTorqueMeasured(...)</code> . Legt fest, ob für die Auswertung die gemessenen oder die mithilfe des statischen RobotermodeLLs berechneten Drehmomentwerte verwendet werden sollen.  <ul style="list-style-type: none"> <li>■ <b>true</b>: Verwendung der gemessenen Momente</li> <li>■ <b>false</b>: Verwendung der statischen Momente (modellbasiert)</li> </ul> <p><b>Hinweis:</b> Bei Verwendung der statischen Momente (modellbasiert) haben dynamische Effekte, die beispielsweise durch die Beschleunigung des Roboters entstehen, keinen Einfluss auf die ermittelten Werte.</p>
<code>lbr_iiwa</code>	Typ: LBR  LBR-Instanz der Applikation. Repräsentiert den Roboter, für den die maximalen absoluten Drehmomentwerte ermittelt werden sollen.
<code>maxTorqueData</code>	Typ: TorqueStatistic  Variable für den Rückgabewert von <code>stopEvaluation()</code> . Der Rückgabewert enthält die ermittelten maximalen absoluten Drehmomentwerte und weitere Informationen zur Auswertung.

### 8.3.2 Ergebnis der Auswertung der maximalen absoluten Momente abfragen

Wenn die Auswertung der maximalen absoluten Drehmomentwerte beendet ist, kann das Ergebnis der Auswertung abgefragt werden.

#### Übersicht

Folgende Methoden der Klasse `TorqueStatistic` stehen zur Verfügung:

Methode	Beschreibung
<code>getMaxAbsTorqueValues()</code>	Rückgabetyp: <code>double[]</code> ; Einheit: Nm  Gibt ein double-Array zurück, das die ermittelten maximalen absoluten Drehmomentwerte (abtriebsseitig) aller Achsen enthält.
<code>getSingleMaxAbsTorqueValue(...)</code>	Rückgabetyp: <code>double</code> ; Einheit: Nm  Gibt den ermittelten maximalen absoluten Drehmomentwert (abtriebsseitig) für die Achse zurück, die als Parameter (Typ: <code>JointEnum</code> ) übergeben wird.
<code>areDataValid()</code>	Rückgabetyp: <code>boolean</code>  Es wird abgefragt, ob die ermittelten Daten gültig sind (= <code>true</code> ).  Die Daten sind gültig, wenn bei der Befehlsverarbeitung keine Fehler aufgetreten sind.
<code>getStartTimeStamp()</code>	Rückgabetyp: <code>java.util.Date</code>  Gibt den Zeitpunkt zurück, zu dem die Auswertung gestartet wurde.

Methode	Beschreibung
getStopTimestamp()	Rückgabetyp: java.util.Date Gibt den Zeitpunkt zurück, zu dem die Auswertung beendet wurde.
isTorqueMeasured()	Rückgabetyp: boolean Es wird abgefragt, ob bei der Auswertung des maximalen absoluten Drehmoments die gemessenen oder die mithilfe des statischen Robotermodells berechneten Drehmomentwerte verwendet wurden. <ul style="list-style-type: none"> <li>■ <b>true:</b> Verwendung der gemessenen Momente</li> <li>■ <b>false:</b> Verwendung der statischen Momente (modellbasiert)</li> </ul>

**Beispiel** Die bei einer Fügeaufgabe maximal auftretenden Momente sollen als Referenzmomente bei einem Bremsentest verwendet werden. Dazu werden die Drehmomente, die beim Ausführen der Fügeaufgabe gemessen werden, ausgewertet und das maximale absolute Drehmoment für jede Achse ermittelt.

Nach dem Start der Auswertung werden die Bewegungsbefehle des Fügeprozesses kommandiert. Ist der Fügevorgang abgeschlossen, wird die Auswertung beendet und das Ergebnis der Auswertung für die Achsen A2 und A4 in den Prozessdaten gespeichert. Sind die ermittelten Daten ungültig, wird ein Ausgang gesetzt.

```
testEvaluator.setTorqueMeasured(true);
```

```
private LBR testLBR;
private BrakeTestIOGroup brakeTestIOS;
private Tool testGripper;
private Workpiece testWorkpiece;
...
public void run() {

    testGripper.attachTo(testLBR.getFlange());
    testWorkpiece.attachTo(testGripper.getFrame("/GripPoint"));

    // create TorqueEvaluator
    TorqueEvaluator testEvaluator = new TorqueEvaluator(testLBR);

    // select measured torque values
    testEvaluator.setTorqueMeasured(true);

    // start evaluation
    testEvaluator.startEvaluation();

    // performs assembly task
    testAssemblyTask();

    // finish evaluation and store result in variable testMaxTrqData
    TorqueStatistic testMaxTrqData = testEvaluator.stopEvaluation();

    // get maximum absolute measured torque value for joint 2
    double maxTrqA2 = testMaxTrqData
        .getSingleMaxAbsTorqueValue(JointEnum.J2);

    // save result
    getApplicationData().getProcessData("maxTrqA2").setValue(maxTrqA2);

    // get maximum absolute measured torque value for joint 4
    double maxTrqA4 = testMaxTrqData
        .getSingleMaxAbsTorqueValue(JointEnum.J4);
```

```

// save result
getApplicationData().getProcessData("maxTrqA4").setValue(maxTrqA4);

// check if evaluated data is valid
boolean areDataValid = testMaxTrqData.areDataValid();
if(areDataValid == false){
    // if data is not valid, set output signal
    brakeTestIOs.setEvaluatedTorqueInvalid(true);
}

...
}

public void exampleAssemblyTask() {

    testLBR.move(ptp(getFrame("/StartAssembly")));

    ForceCondition testForceCondition =
        ForceCondition.createNormalForceCondition
            (testWorkpiece.getDefaultMotionFrame(), CoordinateAxis.Z, 15.0);
    testWorkpiece.move(linRel(0.0, 0.0, 100.0)
        .breakWhen(testForceCondition));

    CartesianSineImpedanceControlMode testAssemblyMode =
        CartesianSineImpedanceControlMode.createLissajousPattern(
            CartPlane.XY, 5.0, 10.0, 500.0);

    testWorkpiece.move(positionHold(
        testAssemblyMode, 3.0, TimeUnit.SECONDS));

    openGripper();
    testWorkpiece.detach();

    testGripper.move(linRel(0.0, 0.0, -100.0));
}

```

### 8.3.3 Objekt für den Bremsentest erzeugen

#### Beschreibung

Um den Bremsentest ausführen zu können, muss zunächst ein Objekt der Klasse BrakeTest erzeugt werden. Dem Konstruktor der Klasse BrakeTest wird der Index der Achse übergeben, für die der Bremsentest durchgeführt werden soll.

Optional kann über den Parameter *torque* ein Referenzhaltemoment übergeben werden, z. B. das maximale absolute Drehmoment einer Achse, das in einer bestimmten Applikation auftritt.

Beim Bremsentest muss in der Regel geprüft werden, ob die Bremsen das minimale Bremsenhaltemoment aufbringen. Daher wird empfohlen, den Parameter *torque* nicht anzugeben.

#### Syntax

```
BrakeTest brakeTest = new BrakeTest(axis, <torque>);
```

## Erläuterung der Syntax

Element	Beschreibung
<code>brakeTest</code>	<p>Typ: BrakeTest</p> <p>Variable, der die erzeugte BrakeTest-Instanz zugewiesen wird. Über die Variable wird die Ausführung des Bremsentests als Bewegungsbefehl kommandiert.</p>
<code>axis</code>	<p>Typ: int</p> <p>Index der Achse, deren Bremse geprüft werden soll</p> <p>■ 0 ... 6: Achse A1 ... A7</p>
<code>torque</code>	<p>Typ: double; Einheit: Nm</p> <p>Vom Anwender vorgegebenes Referenzhaltemoment (abtriebsseitig), z. B. das maximale absolute Drehmoment, das im Voraus für eine applikationsspezifische Bewegungsfolge ermittelt wurde.</p> <p>Wird kein Referenzhaltemoment angegeben, wird beim Bremsentest der niedrigste von folgenden Werten als zu testendes Haltemoment verwendet: Minimales Bremsenhaltemoment oder Motorhaltemoment.</p> <p>Wird ein Referenzhaltemoment angegeben, wird als das zu testende Haltemoment einer von folgenden Werten verwendet: Das vorgegebene Referenzhaltemoment (<code>torque</code>), das minimale Bremsenhaltemoment oder das Motorhaltemoment.</p> <p>Das zu testende Haltemoment wird systemintern nach folgenden Regeln festgelegt:</p> <ol style="list-style-type: none"> <li>1. Wenn das Referenzhaltemoment größer als das Minimum aus minimalem Bremsenhaltemoment und Motorhaltemoment ist, wird das Minimum von minimalen Bremsenhaltemoment und Motorhaltemoment als zu testendes Haltemoment verwendet.</li> <li>2. Wenn das Referenzhaltemoment kleiner als 20 % des Minimums aus minimalem Bremsenhaltemoment und Motorhaltemoment ist, werden 20 % des Minimums aus minimalem Bremsenhaltemoment und Motorhaltemoment als zu testendes Haltemoment verwendet.</li> <li>3. In allen anderen Fällen wird das Referenzhaltemoment verwendet.</li> </ol> <p><b>Hinweis:</b> Das minimale und maximale Bremsenhaltemoment ist in den Motordaten gespeichert. Das Motorhaltemoment wird aus den Motordaten abgeleitet.</p>

### 8.3.4 Ausführung des Bremsentests starten

#### Beschreibung

Der Bremsentest wird durch ein Bewegungskommando ausgeführt, das über die Klasse BrakeTest zur Verfügung gestellt wird. Um den Bremsentest auszuführen, wird über die in der Applikation verwendete Roboter-Instanz die `move(...)`- oder `moveAsync(...)`-Methode aufgerufen und das für den Bremsentest erzeugte Objekt übergeben.

Um das Ergebnis des Bremsentests auszuwerten, muss der Rückgabewert des Bewegungskommandos in einer Variable vom Typ IMotionContainer gespeichert werden.

Wird bei der Ausführung des Bremsentests ein Fehler festgestellt, wird der Bremsentest abgebrochen. Um im Programm auf einen Fehlerfall reagieren

zu können wird empfohlen, die Ausführung und Auswertung des Bremsentests innerhalb eines try-Blocks zu kommandieren und die im Fehlerfall auftretende CommandInvalidException zu behandeln.

**Syntax**

```
try{
    BrakeTest brakeTest = ...;
    IMotionContainer brakeTestMotionContainer =
        robot.moveToMoveAsync(brakeTest);
    ...
} catch(CommandInvalidException ex){
    ...
}
```

**Erläuterung der Syntax**

<b>Element</b>	<b>Beschreibung</b>
<i>brakeTest</i>	Typ: BrakeTest  Variable, der die erzeugte BrakeTest-Instanz zugewiesen wird. Die Instanz legt die Achse fest, für die der Bremsentest ausgeführt werden soll sowie optional ein vom Programmierer vorgegebenes Referenzhaltemoment.
<i>brakeTest Motion Container</i>	Typ: IMotionContainer  Variable für den Rückgabewert des Bewegungskommandos move(...) oder moveAsync(...), mit dem der Bremsentest ausgeführt wird. Wenn der Bremsentest beendet ist, kann das Ergebnis über die Variable ausgewertet werden.
<i>robot</i>	Typ: Robot  Instanz des in der Applikation verwendeten Roboters, für dessen Achsen der Bremsentest durchgeführt werden soll.
<i>ex</i>	Typ: CommandInvalidException  Ausnahme, die auftritt, wenn der Bremsentest aufgrund eines Fehlers abgebrochen wird. Es wird empfohlen, die Ausnahme innerhalb des catch-Blocks so zu behandeln, dass ein abgebrochener Bremsentest für eine einzelne Bremse nicht zum Abbruch der gesamten Bremsentest-Applikation führt.

**8.3.5 Bremsentest auswerten****Beschreibung**

Wenn der Bremsentest beendet ist, kann das Ergebnis ausgewertet werden. Hierfür muss der Rückgabewert des Bewegungskommandos, über den der Bremsentest ausgeführt wird, einer Variablen vom Typ IMotionContainer zugewiesen werden.

Um den Bremsentest auszuwerten, wird die IMotionContainer-Instanz des zugehörigen Bewegungsbefehls an die statische Methode evaluateResult(...) übergeben. Die Methode gehört zur Klasse BrakeTest und liefert ein Objekt vom Typ BrakeTestResult zurück. Von diesem Objekt können unterschiedliche Informationen zum ausgeführten Bremsentest abgefragt werden.

**Syntax**

```
IMotionContainer brakeTestMotionContainer =
    robot.moveToMoveAsync(brakeTest);
BrakeTestResult result =
    BrakeTest.evaluateResult(brakeTestMotionContainer);
```

### Erläuterung der Syntax

Element	Beschreibung
<i>brakeTest Motion Container</i>	Typ: IMotionContainer  Variable für den Rückgabewert des Bewegungskommandos move(...) oder moveAsync(...), mit dem der Bremsentest ausgeführt wird.
<i>result</i>	Typ: BrakeTestResult  Variable für den Rückgabewert von evaluateResult(...). Der Rückgabewert enthält das Ergebnis des Bremsentests und weitere Informationen zum Bremsentest, die über die Variable abgefragt werden können.

### Übersicht

Zur Auswertung des Bremsentests stehen folgende Methoden der Klasse BrakeTestResult zur Verfügung:

Methoden	Beschreibung
getAxis()	Rückgabetyp: int  Gibt den Index der Achse zurück, deren Bremse geprüft wurde. Der Index beginnt bei 0 (= Achse A1).
getBrakeIndex()	Rückgabetyp: int  Gibt den Index der getesteten Bremse des Motors zurück (beginnend bei 0). Bei einem Bremsentest für den LBR iiwa wird immer der Wert 0 zurückgegeben.
getFriction()	Rückgabetyp: double, Einheit: Nm  Gibt das bei der Ermittlungsfahrt bestimmte Reibmoment (abtriebsseitig) zurück.
getGravity()	Rückgabetyp: double, Einheit: Nm  Gibt das bei der Ermittlungsfahrt bestimmte Gravitationsmoment (abtriebsseitig) zurück.
getMaxBrake HoldingTorque()	Rückgabetyp: double, Einheit: Nm  Gibt das aus den Motordaten ermittelte Moment (abtriebsseitig) zurück, das die Bremse nicht überschreiten darf. (= Maximales Bremsenhaltemoment)
getMeasuredBrake HoldingTorque()	Rückgabetyp: double, Einheit: Nm  Gibt das beim Bremsentest gemessene Haltemoment (abtriebsseitig) zurück. Dieser Wert wird mit dem zu testenden Haltemoment verglichen.
getMinBrake HoldingTorque()	Rückgabetyp: double, Einheit: Nm  Gibt das aus den Motordaten ermittelte mindestens zu erreichende Moment der Bremse (abtriebsseitig) zurück. (= Minimales Bremsenhaltemoment)
getMotor HoldingTorque()	Rückgabetyp: double, Einheit: Nm  Gibt das aus den Motordaten ermittelte Motorhaltemoment (abtriebsseitig) zurück.
getMotorIndex()	Rückgabetyp: int  Gibt den Index des getesteten Motors des Antriebs zurück (beginnend bei 0). Bei einem Bremsentest für den LBR iiwa wird immer der Wert 0 zurückgegeben.
getMotor MaximalTorque()	Rückgabetyp: double, Einheit: Nm  Gibt das aus den Motordaten ermittelte Motormaximalmoment (abtriebsseitig) zurück.

Methode	Beschreibung
getState()	Rückgabetyp: Enum vom Typ BrakeState Gibt das Ergebnis des Bremsentests zurück. (>>> 8.3.5.1 "Ergebnis des Bremsentests abfragen" Seite 131)
getTestedTorque()	Rückgabetyp: double, Einheit: Nm Gibt das zu testende Haltemoment zurück, mit dem das beim Bremsen-test aufgebrachte und gemessene Haltemoment (abtriebsseitig) vergli-chen wird.
getTimestamp()	Rückgabetyp: java.util.Date Gibt den Zeitpunkt zurück, zu dem der Bremsentest gestartet wurde.

### 8.3.5.1 Ergebnis des Bremsentests abfragen

<b>Beschreibung</b>	Das Testergebnis wird über die BrakeTestResult-Methode getState() abgefragt. Zurückgegeben wird ein Enum vom Typ BrakeState, dessen Werte die möglichen Testergebnisse beschreiben.  Die möglichen Testergebnisse sind bestimmten Log-Level zugeordnet. Der zum Testergebnis gehörige Log-Level kann mit getLogLevel() abgefragt werden.
<b>Syntax</b>	<pre>BrakeTestResult result = ...; BrakeState state = result.getState(); LogLevel logLevel = state.getLogLevel();</pre>

Erläuterung der Syntax	Element	Beschreibung
	<i>result</i>	Typ: BrakeTestResult  Variable für den Rückgabewert der statischen Methode evaluateResult(...), die die Klasse BrakeTest zur Auswer-tung des Bremsentests anbietet. Der Rückgabewert enthält das Ergebnis des Bremsentests und weitere Informationen zum Bremsentest, die über die Variable abgefragt werden können.
	<i>state</i>	Typ: Enum vom Typ BrakeState  Variable für den Rückgabewert von getState(). Der Rück-gabewert enthält das Testergebnis. (>>> "BrakeState" Seite 131)
	<i>logLevel</i>	Typ: Enum vom Typ LogLevel  Variable für den Rückgabewert von getLogLevel(). Der Rückgabewert enthält den Log-Level des Testergebnisses. <ul style="list-style-type: none"> <li>■ <b>LogLevel.Error:</b> Der Bremsentest konnte nicht ausge-führt werden oder ist fehlgeschlagen.</li> <li>■ <b>LogLevel.Info:</b> Der Bremsentest wurde erfolgreich ausgeführt.</li> <li>■ <b>LogLevel.Warning:</b> Das zu testende Haltemoment wurde erreicht, aber während der Ausführung des Bremsentests traten Probleme auf.</li> </ul>

<b>BrakeState</b>	Das Enum vom Typ BrakeState besitzt folgende Werte (mit Angabe des zugehörigen Log-Levels):
-------------------	---

Wert	Beschreibung
BrakeUntested	Der Bremsentest konnte aufgrund von Störungen nicht ausgeführt werden oder wurde während der Ausführung abgebrochen. Log-Level: LogLevel.Error
BrakeUnknown	Der Bremsentest konnte nicht ausgeführt werden, da nicht genügend Moment aufgebaut werden konnte (z. B. durch zu hohe Reibung). Log-Level: LogLevel.Error
BrakeError	Der Bremsentest ist fehlgeschlagen. Das gemessene Haltemoment unterschreitet das zu testende Haltemoment. Die Bremse ist defekt. Log-Level: LogLevel.Error
BrakeWarning	Das gemessene Haltemoment liegt weniger als 5 % über dem zu testenden Haltemoment. Die Bremse hat die Verschleißgrenze erreicht und wird in absehbarer Zeit als defekt erkannt werden. Log-Level: LogLevel.Warning
BrakeMax Unknown	Das zu testende Haltemoment wurde zwar erreicht, aber die Bremse konnte nicht gegen das maximale Bremsenhaltemoment getestet werden. Log-Level: LogLevel.Warning
BrakeExcessive	Das gemessene Haltemoment ist größer als das maximale Bremsenhaltemoment. Das Anhalten mit der Bremse kann zu einer Beschädigung der Maschine führen. Log-Level: LogLevel.Warning
BrakeReady	Das gemessene Haltemoment liegt mehr als 5 % über dem zu testenden Haltemoment. Die Bremse ist voll funktionsfähig. Log-Level: LogLevel.Info

**Beispiel**

Für Achse A2 wird ein Bremsentest durchgeführt. Wird der Bremsentest abgebrochen, wird dies über ein entsprechendes Ausgangssignal gemeldet. Wird der Bremsentest vollständig ausgeführt, wird eine Meldung mit dem gemessenen Haltemoment ausgegeben und das Testergebnis abgefragt. Abhängig davon, ob das gemessene Haltemoment zu niedrig ist, innerhalb des Toleranzbereichs oder im idealen Bereich liegt, wird ebenfalls je ein Ausgang gesetzt.

```

private LBR exampleLBR_iowa;
private BrakeTestIOGroup brakeTestIOS;
...
public void run() {
    ...

    try {
        int indexA2 = 1;
        BrakeTest exampleBrakeTest = new BrakeTest(indexA2);

        IMotionContainer exampleBrakeTestMotionContainer =
            exampleLBR_iowa.move(exampleBrakeTest);

        BrakeTestResult resultA2 = BrakeTest.evaluateResult(
            exampleBrakeTestMotionContainer);
    }
}

```

```

        double measuredTorque =
            resultA2.getMeasuredBrakeHoldingTorque();

        getLogger().info("Measured torque for A2: " + measuredTorque);

        BrakeState state = resultA2.getState();
        if(state == BrakeState.BrakeError)
            brakeTestIOs.setA2_BrakeError(true);
        else if(state == BrakeState.BrakeWarning)
            brakeTestIOs.setA2_BrakeWarning(true);
        else if(state == BrakeState.BrakeReady)
            brakeTestIOs.setA2_BrakeOK(true);

    } catch (CommandInvalidException ex) {
        brakeTestIOs.setBrakeTest_Aborted(true);
        ex.printStackTrace();
    }
}

```

## 8.4 Bremsentest durchführen

### HINWEIS

Wenn eine Bremse als defekt erkannt wird und die Antriebe abgeschaltet werden, kann der Roboter zusammensacken.

#### Beschreibung

Wird die Vorlage für die Bremsentest-Applikation unverändert übernommen, werden zunächst die Drehmomente, die während einer Bewegungsfolge gemessen werden, achsspezifisch ausgewertet, und das dabei aufgetretene maximale absolute Moment für jede Achse ermittelt. Das Ergebnis der Auswertung wird auf der smartHMI ausgegeben. Danach werden die Bremsen, beginnend bei Achse A1, nacheinander getestet. Das Ergebnis des Bremsentests, wird für jede Achse einzeln auf der smartHMI ausgegeben.

#### Voraussetzung

- Es befinden sich keine Personen oder Gegenstände im Bewegungsreich des Roboters.
- Programmablaufart **Continuous** (Standard-Betrieb)
- Der Roboter hat Betriebstemperatur.
- Bremsentest-Applikation anwählen und starten.

#### Vorgehensweise



Wird die Bremsentest-Applikation während des Tests einer Bremse pausiert (z. B. durch Drücken der Start-Taste am smartPAD oder durch eine Stopp-Anforderung), wird der Bremsentest der Achse abgebrochen.

Wird die Bremsentest-Applikation fortgesetzt, wird der zuvor abgebrochene Bremsentest für die betroffene Achse wiederholt. Falls die Achse sich nicht mehr an der Postion befindet, von der aus der abgebrochene Bremsentest gestartet wurde, muss sie durch Drücken der Start-Taste rückpositioniert werden, bevor die Applikation fortgesetzt werden kann.

#### 8.4.1 Ergebnis der Auswertung der maximalen absoluten Momente (Anzeige)

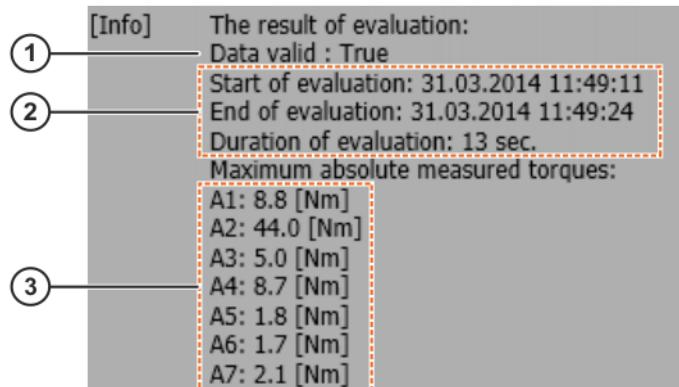


Abb. 8-3: Ergebnis einer Auswertung der maximalen absoluten Momente

Pos.	Beschreibung
1	Gültigkeit Zeigt an, ob die ermittelten Daten gültig sind. Die Daten sind gültig, wenn bei der Befehlsverarbeitung keine Fehler aufgetreten sind.
2	Zeitangaben Startzeitpunkt, Endzeitpunkt und Gesamtdauer der Auswertung.
3	Ermittelte Daten Für jede Achse wird das maximale absolute Drehmoment angezeigt, das bei der Auswertung ermittelt wurde.

#### 8.4.2 Ergebnis des Bremsentests (Anzeige)

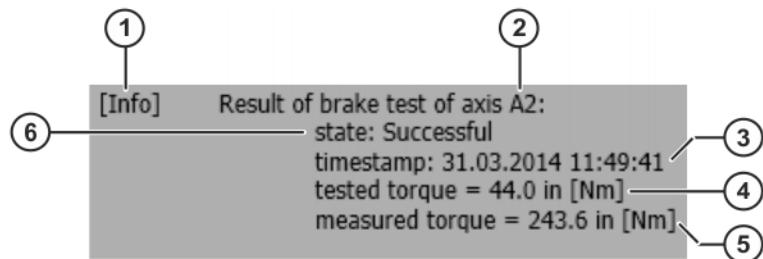


Abb. 8-4: Ergebnis eines Bremsentests für Achse A2

Pos.	Beschreibung
1	Log-Level Abhängig vom Ergebnis des Bremsentests wird die Meldung mit einem bestimmten Log-Level ausgegeben. <ul style="list-style-type: none"><li>■ <b>Info:</b> Der Bremsentest wurde erfolgreich ausgeführt.</li><li>■ <b>Warning:</b> Das zu testende Haltemoment wurde erreicht, aber während der Ausführung des Bremsentests traten Probleme auf (siehe Beschreibungen der möglichen Testergebnisse unter Pos. 6).</li><li>■ <b>Error:</b> Der Bremsentest konnte nicht ausgeführt werden oder ist fehlgeschlagen.</li></ul>
2	Getestete Achse

Pos.	Beschreibung
3	Zeitstempel Zeitpunkt, zu dem der Bremsentest für die Achse gestartet wurde
4	Zu testendes Haltemoment
5	Gemessenes Haltemoment
6	Ergebnis des Bremsentests <ul style="list-style-type: none"> <li>■ <b>Untested:</b> Der Bremsentest konnte aufgrund von Störungen nicht ausgeführt werden oder wurde während der Ausführung abgebrochen.</li> <li>■ <b>Unknown:</b> Der Bremsentest konnte nicht ausgeführt werden, da nicht genügend Moment aufgebaut werden konnte (z. B. durch zu hohe Reibung).</li> <li>■ <b>Failed:</b> Der Bremsentest ist fehlgeschlagen. Das gemessene Haltemoment unterschreitet das zu testende Haltemoment. Die Bremse ist defekt.</li> <li>■ <b>Warning:</b> Das gemessene Haltemoment liegt weniger als 5 % über dem zu testenden Haltemoment. Die Bremse hat die Verschleißgrenze erreicht und wird in absehbarer Zeit als defekt erkannt werden.</li> <li>■ <b>Maximum unknown:</b> Das zu testende Haltemoment wurde zwar erreicht, aber die Bremse konnte nicht gegen das maximale Bremsenhaltemoment getestet werden.</li> <li>■ <b>Excessive:</b> Das gemessene Haltemoment ist größer als das maximale Bremsenhaltemoment. Das Anhalten mit der Bremse kann zu einer Beschädigung der Maschine führen.</li> <li>■ <b>Successfull:</b> Das gemessene Haltemoment liegt mehr als 5 % über dem zu testenden Haltemoment. Die Bremse ist voll funktionsfähig.</li> </ul>

 **WARNING**

Ist die Funktionsfähigkeit einer Bremse nicht gewährleistet, kann der Roboter zusammensacken. Wird beim Bremsentest mindestens einer Achse des LBR iiwa festgestellt, dass die Bremse das gewünschte Haltemoment nicht aufbringen kann, ist der Roboter sofort aus dem Verkehr zu ziehen.



## 9 Projektverwaltung

### 9.1 Sunrise-Projekte – Übersicht

In einem Sunrise-Projekt sind alle Daten zusammengefasst, die zum Betrieb einer Station erforderlich sind. Zu einem Sunrise-Projekt gehören:

- Stationskonfiguration  
Die Stationskonfiguration beschreibt die statischen Eigenschaften der Station. Dazu gehören z. B. Hard- und Software-Komponenten.
- Applikationen  
Applikationen enthalten den Quellcode zur Ausführung einer Aufgabe für die Station. Sie werden mit KUKA Sunrise.Workbench in Java programmiert und auf der Steuerung ausgeführt. Ein Sunrise-Projekt kann beliebig viele Applikationen enthalten.
- Laufzeitdaten  
Laufzeitdaten sind alle Daten, die von den Applikationen zur Laufzeit verwendet werden. Dazu gehören z. B. Zielpunkte für Bewegungen, Werkzeugdaten und Prozessparameter.
- Sicherheitskonfiguration  
Die Sicherheitskonfiguration enthält die konfigurierten Sicherheitsfunktionen.
- E/A-Konfiguration (optional)  
Die E/A-Konfiguration enthält die in WorkVisual verschalteten Ein-/Ausgänge der verwendeten Feldbusse. Die Ein-/Ausgänge können in der Applikation verwendet werden.

Sunrise-Projekte werden mit KUKA Sunrise.Workbench erstellt und verwaltet.

(>>> 5.3 "Sunrise-Projekt mit Vorlage erstellen" Seite 51)

Es kann sich immer nur 1 Sunrise-Projekt auf der Robotersteuerung befinden. Dieses wird per Projekt-Synchronisation von Sunrise.Workbench auf die Robotersteuerung übertragen.

(>>> 9.4 "Synchronisation von Projekten" Seite 155)

### 9.2 Frame-Verwaltung

#### Übersicht

Frames sind Koordinaten-Transformationen, die die Lage von Raumpunkten oder Objekten in einer Station beschreiben. Die Koordinaten-Transformationen sind hierarchisch in einer Baumstruktur angeordnet. Jeder Frame besitzt in dieser Hierarchie einen übergeordneten Eltern-Frame, mit dem er durch die Transformation verknüpft ist.

Wurzelement oder Ursprung der Transformation ist das Welt-Koordinatensystem, das defaultmäßig im Roboterfuß liegt. D. h. alle Frames beziehen sich direkt oder indirekt auf das Welt-Koordinatensystem.

Eine Transformation beschreibt die relative Lage von 2 Koordinatensystemen zueinander, d. h. wie ein Frame bezogen auf sein Eltern-Frame verschoben und orientiert ist.

Die Lage eines Frames bezogen auf sein Eltern-Frame, ist durch folgende Transformationsdaten definiert:

- X, Y, Z: Verschiebung des Ursprungs entlang der Achsen des Eltern-Frames
- A, B, C: Verdrehung der Achswinkel des Eltern-Frames

Drehwinkel der Frames:

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

### 9.2.1 Neuen Frame anlegen

#### Beschreibung

In Sunrise.Workbench angelegte Frames sind projektspezifisch und können in jeder Roboter-Applikation des Projekts verwendet werden.

Nach der Synchronisation des Projekts stehen die Frames auf der smartHMI zur Verfügung. Auf der smartHMI können die Frames geteacht werden, um die Position der Frames im Raum zu bestimmen. Geteachte Frames können manuell angefahren werden.

(>>> 6.13 "Frames teachen und manuell anfahren" Seite 85)

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Applikationsdaten** auf den gewünschten Eltern-Frame rechtsklicken und im Kontextmenü **Neuen Frame anlegen** wählen. Der neue Frame wird angelegt und im Frame-Baum als Kindelement des Eltern-Frames eingefügt.
3. Das System vergibt automatisch einen Frame-Namen. Es wird empfohlen, den Namen in der Sicht **Eigenschaften** zu ändern.

Ein beschreibender Frame-Name erleichtert die Programmierung und die Orientierung im Programm. Innerhalb einer Hierarchieebene müssen die Frame-Namen eindeutig sein und dürfen nicht mehrfach vergeben werden.

#### Beispiel

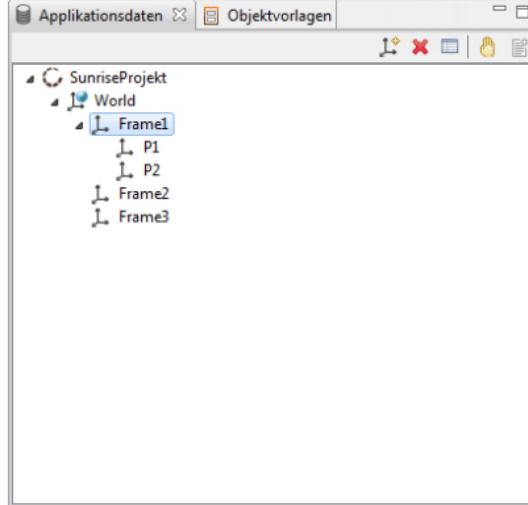


Abb. 9-1: Applikationsdaten – Frames

Frame1, 2 und 3 sind Kindelemente von World und liegen auf derselben Hierarchieebene. P1 und P2 sind Kindelemente von Frame1 und liegen eine Ebene darunter.

### 9.2.2 Frame als Basis kennzeichnen

#### Beschreibung

Frames können in der Sicht **Applikationsdaten** als Basis gekennzeichnet werden.

Nur so gekennzeichnete Frames können nach der Synchronisation des Projekts auf der smartHMI als Basis für das Handverfahren ausgewählt und vermessen werden.

(>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78)

(>>> 7.2.2 "Basis vermessen: 3-Punkt-Methode " Seite 110)

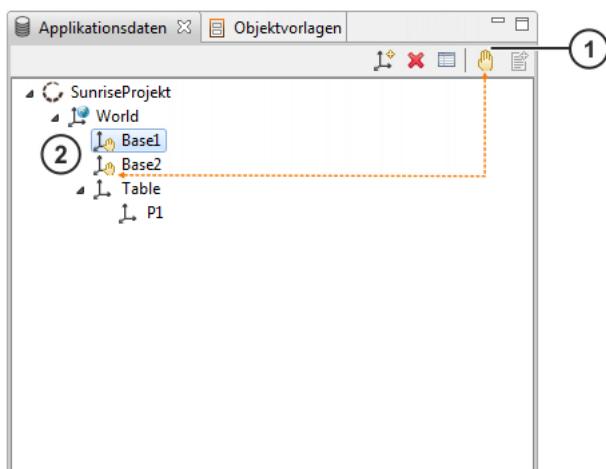


Es wird empfohlen, bei diesen Frames auf eine aussagekräftige Namensgebung zu achten, um dem Bediener am smartPAD die Auswahl der Basis für das Handverfahren zu erleichtern.

### Vorgehensweise

- Auf den gewünschten Frame rechtsklicken und im Kontextmenü **Basis** wählen.  
Alternative:  
Den Frame markieren und auf das Hand-Symbol **Basis** klicken.  
Der Frame wird mit einem Hand-Symbol gekennzeichnet.

### Beispiel



**Abb. 9-2: Frame als Basis kennzeichnen**

- 1 Hand-Symbol **Basis**
- 2 Base1 und Base2 sind als Basis gekennzeichnet

### 9.2.3 Frames verschieben

#### Beschreibung

Ein Frame kann in der Sicht **Applikationsdaten** verschoben und einem neuen Eltern-Frame zugeordnet werden. Folgende Punkte sind dabei zu beachten:

- Die untergeordneten Frames werden automatisch mitverschoben.
- Die absolute Position der verschobenen Frames im Raum bleibt erhalten. Die relative Transformation der Frames zum neuen Eltern-Frame wird angepasst.
- Frames können nicht unterhalb eines ihrer Kindelemente eingefügt werden.
- Die Namen der direkten Kindelemente eines Frames müssen eindeutig sein.



Wird ein Frame verschoben, ändert sich sein Pfad. Da Frames über diesen Pfad im Quellcode von Applikationen verwendet werden, muss die Pfadangabe in den Applikationen entsprechend korrigiert werden.

- |                       |  |
|-----------------------|--|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"> <li>1. Auf den gewünschten Frame klicken und die linke Maustaste gedrückt halten.</li> <li>2. Mit der Maus den Frame auf neuen Eltern-Frame ziehen.</li> <li>3. Wenn der gewünschte neue Eltern-Frame markiert ist, die Maustaste loslassen.</li> </ol> |
|-----------------------|--|

#### 9.2.4 Frames löschen

- |                     |  |
|---------------------|--|
| <b>Beschreibung</b> | Frames können in der Sicht <b>Applikationsdaten</b> aus dem Frame-Baum entfernt werden. Besitzt ein Frame Kind-Elemente, stehen dafür folgende Optionen zur Verfügung: |
|---------------------|--|

- **Elternframe ändern:** Nur der ausgewählte Frame wird gelöscht. Die untergeordneten Frames bleiben erhalten, werden eine Ebene nach oben verschoben und einem neuen Eltern-Frame zugeordnet.  
Die absolute Position der verschobenen Frames im Raum bleibt erhalten. Die relative Transformation der Frames zum neuen Eltern-Frame wird angepasst.



Wird ein Frame verschoben, ändert sich sein Pfad. Da Frames über diesen Pfad im Quellcode von Applikationen verwendet werden, muss die Pfadangabe in den Applikationen entsprechend korrigiert werden.

- **Eltern- und Kindframes löschen:** Löscht den ausgewählten Frame und alle untergeordneten Frames.

- |                       |   |
|-----------------------|---|
| <b>Vorgehensweise</b> | <ol style="list-style-type: none"> <li>1. Auf den zu löschenen Frame rechtsklicken und im Kontextmenü <b>Löschen</b> wählen. Ein Frame ohne Kindelemente wird sofort gelöscht.</li> <li>2. Wenn der Frame Kindelemente besitzt, wird abgefragt, ob diese ebenfalls gelöscht werden sollen. Die gewünschte Option auswählen.</li> <li>3. Nur bei Option <b>Elternframe ändern:</b> Tritt durch das Verschieben der Kind-Elemente ein Namenskonflikt auf, wird eine Hinweismeldung angezeigt und der Löschvorgang abgebrochen.</li> </ol> <p><b>Abhilfe:</b> Einen der beiden betroffenen Frames umbenennen und den Löschvorgang wiederholen.</p> |
|-----------------------|---|

#### 9.2.5 Eigenschaften eines Frames anzeigen und ändern



Die Lage und Orientierung eines Frames wird in der Regel beim Teachen mit dem Roboter festgelegt. Es ist jedoch möglich, die Positionswerte eines Frames manuell einzugeben oder sie nachträglich zu ändern.

Folgende Punkte sind dabei unbedingt zu beachten:

- Eine Änderung der Transformationsdaten verschiebt nicht nur den aktuellen Frame, sondern alle ihm untergeordneten Kindelemente und wirkt sich auf alle Applikationen aus, in denen diese Frames verwendet werden.
- Die eingelernten Werte von Status, Turn und Redundanzwinkel werden beibehalten. Ein Anfahren des Frames oder seiner Kindelemente ist unter Umständen nicht mehr möglich.
- Nach einer Änderung der Transformationsdaten müssen alle Programme, in denen der Frame Verwendung findet, in der Betriebsart Manuell Reduzierte Geschwindigkeit (T1) getestet werden.

- |                     |   |
|---------------------|---|
| <b>Beschreibung</b> | Frames, die als Basis markiert sind, können mit dem Roboter vermessen werden. |
|---------------------|---|

(>>> 7.2.2 "Basis vermessen: 3-Punkt-Methode " Seite 110)

Wenn die Daten einer vermessenen Basis per Synchronisation in Sunrise.Workbench übernommen werden, ändern sich die Transformationsdaten des Frames entsprechend der Vermessung. Die Transformationsdaten der Kindelemente des Frames ändern sich durch die Vermessung nicht, d. h. es ändert sich nur die Lage des Frames in Bezug auf das Welt-Koordinatensystem. Die Redundanzinformationen bleiben ebenfalls unverändert.

#### Vorgehensweise

1. Frame in der Sicht **Applikationsdaten** markieren. Die Eigenschaften des Frames werden in der Sicht **Eigenschaften** angezeigt.
2. Bei Bedarf über den Button  (**Kategorien anzeigen**) die Eigenschaften nach Kategorien geordnet anzeigen.
3. Im Feld **Wert** der Eigenschaft den neuen Wert eingeben und mit der Eingabe-Taste bestätigen.



Bei physikalischen Größen kann der Wert mit Einheit eingegeben werden. Ist diese mit der voreingestellten Einheit kompatibel, wird der Wert entsprechend umgerechnet (z. B. cm in mm oder ° in rad). Wird keine Einheit eingegeben, wird die voreingestellte Einheit verwendet.

#### Übersicht

Die nach Kategorien geordneten Eigenschaften enthalten folgende Informationen:

Kategorie	Beschreibung
<b>Allgemein</b>	Allgemeine Informationen <ul style="list-style-type: none"> <li>■ <b>Name:</b> Name des Frames</li> <li>■ <b>Kommentar:</b> Optional</li> <li>■ <b>Projekt:</b> Zugehöriges Projekt</li> <li>■ <b>Letzte Änderung:</b> Datum und Uhrzeit der letzten Änderung</li> </ul>
<b>Transformation</b>	Transformationsdaten <ul style="list-style-type: none"> <li>■ <b>X, Y, Z:</b> Verschiebung des Frames bezogen auf seinen Eltern-Frame</li> <li>■ <b>A, B, C:</b> Verdrehung des Frames bezogen auf seinen Eltern-Frame</li> </ul>
<b>Redundanz</b>	Redundanzinformationen <ul style="list-style-type: none"> <li>■ <b>E1:</b> Wert des Redundanzwinkels</li> <li>■ <b>Status</b></li> <li>■ <b>Turn</b></li> </ul>

Kategorie	Beschreibung
<b>Teach Informationen</b>	<p>Informationen zum geteachten Frame</p> <ul style="list-style-type: none"> <li>■ <b>Gerät:</b> Verwendeter Roboter</li> <li>■ <b>Werkzeug:</b> Verwendetes Werkzeug</li> <li>■ <b>TCP:</b> Frame-Pfad zum verwendeten TCP</li> <li>■ <b>X, Y, Z:</b> Verschiebung des TCP bezogen auf den Ursprungs-Frame des Werkzeugs</li> <li>■ <b>A, B, C:</b> Verdrehung des TCP bezogen auf den Ursprungs-Frame des Werkzeugs</li> </ul>
<b>Vermessung</b>	<p>Informationen zur Basisvermessung</p> <ul style="list-style-type: none"> <li>■ <b>Vermessungsmethode:</b> Verwendete Methode</li> <li>■ <b>Letzte Änderung:</b> Datum und Uhrzeit der letzten Änderung</li> </ul> <p><b>Hinweis:</b> Werden die Transformationsdaten einer vermessenen Basis editiert oder der Frame neu geteacht, werden die Informationen zur Vermessung gelöscht.</p>

### 9.2.6 Frame in Bewegungsanweisung einfügen

- Beschreibung** Ein in den Applikationsdaten angelegter Frame kann als Zielpunkt in eine Bewegungsanweisung eingefügt werden.
- Vorgehensweise**
1. Bewegungsanweisung programmieren, z. B. `robot.move(ptp(...))`.
  2. In der Sicht **Applikationsdaten** auf den Frame klicken, der als Zielpunkt verwendet werden soll, und die linke Maustaste gedrückt halten.
  3. Mit der Maus den Frame in den Editoren-Bereich ziehen und dort so positionieren, dass der Mauszeiger zwischen den Klammern der Bewegung steht.
  4. Maustaste loslassen. Der Frame wird als Zielpunkt der Bewegung eingefügt.

**Beispiel**

```
robot.move(ptp(getApplicationContext().getFrame("/P2/Target")));
```

Die Methode **getApplicationData().getFrame()** zeigt an, dass ein in den Applikationsdaten angelegter Frame eingefügt wurde. Zielpunkt der Bewegung ist der Frame **Target**.

Als Übergabeparameter erhält die Methode den Pfad des Frames im Frame-Baum. Der Frame **Target** ist ein Kindelement von **P2**.

## 9.3 Objektverwaltung

Werkzeuge und Werkstücke werden in Sunrise.Workbench angelegt und verwaltet. Sie gehören zu den Laufzeitdaten eines Projekts.

- Werkzeuge** Eigenschaften:
- Werkzeuge werden am Roboterflansch montiert.
  - Werkzeuge können in der Roboter-Applikation als bewegbare Objekte verwendet werden.
  - Die Werkzeug-Lastdaten wirken sich auf die Roboterbewegungen aus.
  - Werkzeuge können beliebig viele Arbeitspunkte (TCPs) besitzen, die als Frames definiert werden.
- Werkstücke** Eigenschaften:

- Werkstücke können unterschiedlichste Objekte sein, die im Verlauf einer Roboter-Applikation verwendet, bearbeitet oder bewegt werden.
- Werkstücke können an Werkzeuge oder andere Werkstücke gekoppelt werden.
- Werkstücke können in der Roboter-Applikation als bewegbare Objekte verwendet werden.
- Die Werkstück-Lastdaten wirken sich auf die Roboterbewegungen aus, z. B. wenn ein Greifer das Werkstück greift.
- Werkstücke können beliebig viele Frames besitzen, die relevante Punkte kennzeichnen, z. B. Punkte, an denen ein Greifer das Werkstück greift.

### 9.3.1 Geometrischer Aufbau von Werkzeugen

Jedes Werkzeug besitzt einen Ursprungs-Frame (Root). Der Ursprung des Werkzeugs ist defaultmäßig so definiert, dass er bei der Montage des Werkzeugs am Roboterflansch in Lage und Orientierung mit dem Flanschmittelpunkt übereinstimmt. Der Ursprungs-Frame ist immer vorhanden und muss nicht eigens angelegt werden.

Ein Werkzeug kann beliebig viele Arbeitspunkte (TCPs) besitzen, die relativ zum Ursprungs-Frame des Werkstücks (Root) oder zu einem seiner Kindelemente definiert werden.

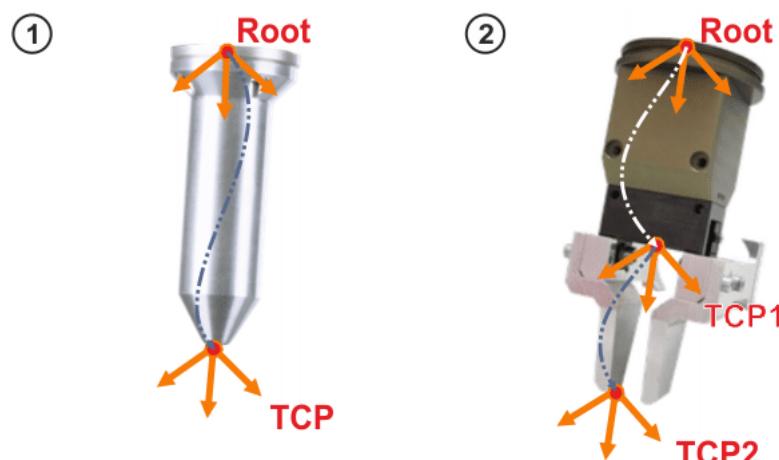


Abb. 9-3: Beispiele für TCPs von Werkzeugen

1 Handführspitze mit 1 TCP

2 Greifer mit 2 TCPs

**i** Die Transformation der Frames ist statisch. Für aktive Werkzeuge, z. B. Greifer, bedeutet das, dass sich der TCP nicht an die aktuelle Position der Backen oder Finger anpasst.

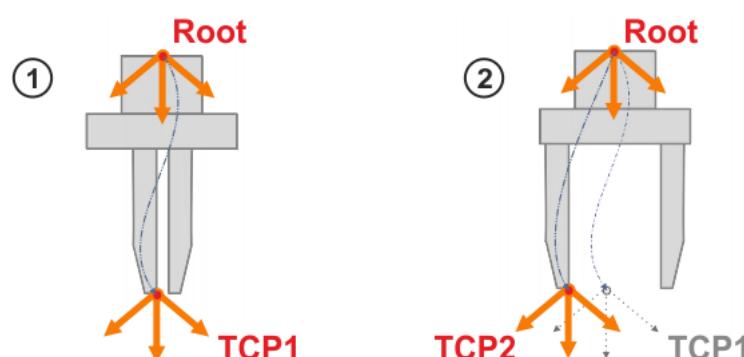


Abb. 9-4: Statischer TCP an einem Greifer

1 Greifer geschlossen

2 Greifer offen

### 9.3.2 Geometrischer Aufbau von Werkstücken

Jedes Werkstück besitzt einen Ursprungs-Frame (Root). Der Ursprungs-Frame ist immer vorhanden und muss nicht eigens angelegt werden.

Ein Werkstück kann beliebig viele Frames besitzen, die relativ zum Ursprungs-Frame des Werkstücks (Root) oder zu einem seiner Kindelemente definiert werden.

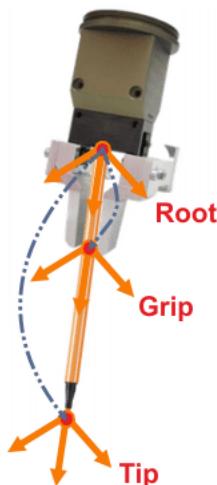


Abb. 9-5: Beispiele für Frames von Werkstücken

### 9.3.3 Werkzeug oder Werkstück anlegen

#### Beschreibung

In Sunrise.Workbench angelegte Werkzeuge und Werkstücke sind projektspezifisch und können in jeder Roboter-Applikation des Projekts verwendet werden.

Die angelegten Werkzeuge können nach der Synchronisation des Projekts auf der smartHMI in den Handverfahroptionen ausgewählt werden.

(>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78)

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Objektvorlagen** die Liste mit den Objektvorlagen öffnen.
3. Um ein Werkzeug anzulegen, auf den Objekttyp **Werkzeuge** rechtsklicken und im Kontextmenü **Neues Werkzeug anlegen** wählen. Die Objekt-Vorlage für das Werkzeug wird angelegt.
4. Um ein Werkstück anzulegen, auf den Objekttyp **Werkstücke** rechtsklicken und im Kontextmenü **Neues Werkstück anlegen** wählen. Die Objekt-Vorlage für das Werkstück wird angelegt.
5. Das System vergibt automatisch einen Objekt-Namen. Es wird empfohlen, den Namen in der Sicht **Eigenschaften** zu ändern.  
Die Objekt-Namen müssen eindeutig sein. Ein beschreibender Name erleichtert die Programmierung und die Orientierung im Programm.
6. In der Sicht **Eigenschaften** die Lastdaten eingeben.  
(>>> 9.3.6 "Lastdaten" Seite 147)

### 9.3.4 Frame für Werkzeug oder Werkstück anlegen

#### Beschreibung

Jeder Frame, der für ein Werkzeug oder Werkstück angelegt wurde, kann in der Roboter-Applikation als Bezugspunkt für Bewegungen programmiert werden.

Die Frames eines Werkzeugs können nach der Synchronisation des Projekts auf der smartHMI als TCP für das kartesische Handverfahren ausgewählt werden.

(>>> 6.8.1 "Fenster Handverfahroptionen" Seite 78)

Die Frames eines Werkzeugs (TCPs) können bezogen auf das Flansch-Koordinatensystem mit dem Roboter vermessen werden.

(>>> 7.2.1 "Werkzeug vermessen" Seite 104)

Wenn die Daten eines vermessenen Werkzeugs per Synchronisation in Sunrise.Workbench übernommen werden, ändern sich die Transformationsdaten des Frames entsprechend der Vermessung.



Die Werkzeugdaten des TCPs, mit dem eine kartesische Bewegung ausgeführt wird, beeinflussen die Robotergeschwindigkeit. Bei falsch eingetragenen Werkzeugdaten kann es zu unerwartet hohen kartesischen Geschwindigkeiten am montierten Werkzeug kommen. Eine Überschreitung der Geschwindigkeit von 250 mm/s in der Betriebsart T1 ist möglich.

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Objektvorlagen** die Liste mit den Objektvorlagen öffnen.
3. Auf die Objekt-Vorlage rechtsklicken und im Kontextmenü **Neuen Frame anlegen** wählen. Der Frame wird angelegt.  
In der obersten Hierarchieebene ist der Eltern-Frame des angelegten Frames der Ursprungs-Frame des Objekts.
4. Um einen neuen Frame unter einem bereits vorhandenen Frame des Objekts einzufügen, auf diesen Eltern-Frame rechtsklicken und im Kontextmenü **Neuen Frame anlegen** wählen. Der Frame wird angelegt.
5. Das System vergibt automatisch einen Frame-Namen. Es wird empfohlen, den Namen in der Sicht **Eigenschaften** zu ändern.  
Ein beschreibender Frame-Name erleichtert die Programmierung und die Orientierung im Programm. Innerhalb einer Hierarchieebene müssen die Frame-Namen eindeutig sein und dürfen nicht mehrfach vergeben werden.
6. In der Sicht **Eigenschaften** die Transformationsdaten des Frames bezogen auf seinen Eltern-Frame eingeben:
  - Felder **X, Y, Z**: Verschiebung des Frames entlang der Achsen des Eltern-Frames
  - Felder **A, B, C**: Orientierung des Frames bezogen auf den Eltern-Frame

#### Eigenschaften

Die nach Kategorien geordneten Eigenschaften enthalten folgende Informationen:

Kategorie	Beschreibung
Allgemein	<p>Allgemeine Informationen</p> <ul style="list-style-type: none"> <li>■ <b>Name:</b> Name des Frames</li> <li>■ <b>Kommentar:</b> Optional</li> <li>■ <b>Sicherheitsgerichtet:</b> Nur relevant bei sicherheitsgerichtetem Werkzeug           <ul style="list-style-type: none"> <li>■ <b>Ja:</b> Frame ist sicherheitsgerichteter Frame</li> <li>■ <b>Nein:</b> Frame ist kein sicherheitsgerichteter Frame</li> </ul> </li> </ul> <p>(&gt;&gt;&gt; 9.3.7 "Sicherheitsgerichtetes Werkzeug" Seite 148)</p>
Geometrie	<p>Radius der Kugel am sicherheitsgerichteten Frame</p> <ul style="list-style-type: none"> <li>■ <b>Radius</b></li> </ul>
Transformation	<p>Transformationsdaten</p> <ul style="list-style-type: none"> <li>■ <b>X, Y, Z:</b> Verschiebung des Frames bezogen auf seinen Eltern-Frame</li> <li>■ <b>A, B, C:</b> Verdrehung des Frames bezogen auf seinen Eltern-Frame</li> </ul>
Vermessung	<p>Informationen zur Werkzeugvermessung</p> <ul style="list-style-type: none"> <li>■ <b>Vermessungsmethode:</b> Verwendete Methode</li> <li>■ <b>Berechnungsfehler:</b> Translatorischer Berechnungsfehler, der die Qualität der Vermessung angibt (Einheit: mm)</li> <li>■ <b>Letzte Änderung:</b> Datum und Uhrzeit der letzten Änderung</li> </ul> <p><b>Hinweis:</b> Werden die Transformationsdaten eines vermessenen Werkzeugs editiert, werden die Informationen zur Vermessung gelöscht.</p>

### 9.3.5 Standard-Frame für Bewegungen festlegen

<b>Beschreibung</b>	<p>Besitzt ein Werkzeug oder Werkstück einen Frame, mit dem ein Großteil der Bewegungen ausgeführt werden soll, kann dieser Frame als Standard-Frame für Bewegungen festgelegt werden.</p> <p>Wenn für ein Werkzeug oder Werkstück ein geeigneter Standard-Frame für Bewegungen festgelegt ist, vereinfacht dies die Bewegungsprogrammierung.</p> <p>(&gt;&gt;&gt; 15.11.4 "Werkzeuge und Werkstücke bewegen" Seite 313)</p> <p>Wird kein Standard-Frame festgelegt, wird automatisch der Ursprungs-Frame des Werkzeugs oder Werkstücks als Standard-Frame für Bewegungen verwendet.</p>
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. Projekt im <b>Paket-Explorer</b> markieren.</li> <li>2. In der Sicht <b>Objektvorlagen</b> den Objekttyp <b>Werkzeuge</b> oder <b>Werkstücke</b> auswählen.</li> <li>3. Das gewünschte Werkzeug oder Werkstück auswählen.</li> <li>4. Auf den gewünschten Frame rechtsklicken und im Kontextmenü <b>Standard-Frame für Bewegungen</b> wählen.</li> </ol> <p>Alternative:</p> <p>Den Frame markieren und auf das Symbol <b>Standard-Frame für Bewegungen</b> klicken.</p>

Der Frame wird als Standard-Frame für Bewegungen gekennzeichnet.

## Beispiel

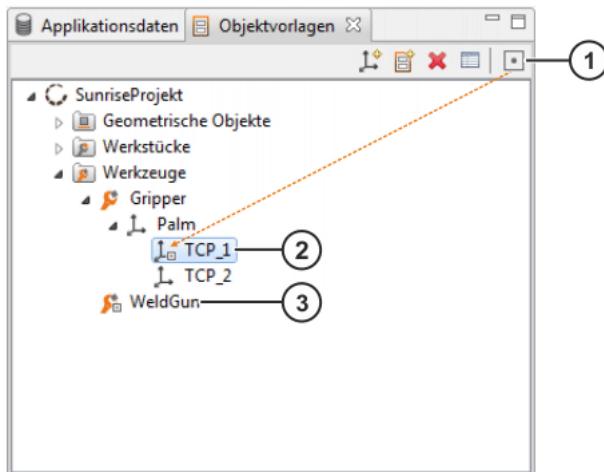


Abb. 9-6: Standard-Frame für Bewegungen

- 1 Symbol **Standard-Frame für Bewegungen**
- 2 Standard-Frame des Werkzeugs **Gripper**: TCP\_1
- 3 Standard-Frame des Werkzeugs **WeldGun**: Ursprungs-Frame

### 9.3.6 Lastdaten

Lastdaten sind alle am Roboterflansch montierten oder damit verbundene Lasten. Sie bilden eine zusätzlich am Roboter montierte Masse, die vom Roboter mitbewegt werden muss.

Die Lastdaten von Werkzeugen und Werkstücken müssen beim Erstellen der zugehörigen Objekt-Vorlagen angegeben werden. Sind mehrere Werkzeuge und Werkstücke mit dem Roboter verbunden, wird aus den einzelnen Lastdaten automatisch die resultierende Gesamtlast berechnet.

Die Lastdaten fließen in die Berechnung der Bahnen und Beschleunigungen ein. Korrekte Lastdaten sind eine wichtige Voraussetzung für die optimale Funktion der Regelung und tragen zur Optimierung der Taktzeiten bei.



**WARNUNG** Roboter nicht mit falschen Lastdaten oder ungeeigneten Lasten betreiben. Wenn dies nicht beachtet wird, können schwere Verletzungen oder erheblicher Sachschaden die Folge sein. Beispielsweise weil das Abbremsen des Roboters wegen falscher Lastdaten zu lange dauert.

#### Quellen

Lastdaten können folgenden Quellen entnommen werden:

- Herstellerangaben
- Manuelle Berechnung
- CAD-Programme
- Die Lastdaten von Werkzeugen können automatisch ermittelt werden.  
(>>> 7.3 "Werkzeug-Lastdaten ermitteln" Seite 112)

#### 9.3.6.1 Lastdaten eingeben

##### Vorgehensweise

1. Sunrise-Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Objektvorlagen** die Liste mit den Objektvorlagen öffnen.
3. Das gewünschte Werkstück oder Werkzeug markieren.

4. In der Sicht **Eigenschaften** die Lastdaten eingeben:
  - **Masse**: Masse des Objekts (Werkzeug oder Werkstück)
  - Felder **MS X**, **MS Y**, **MS Z**: Lage des Massen-Schwerpunkts relativ zum Ursprungs-Frame des Objekts
  - Felder **MS A**, **MS B**, **MS C**: Orientierung der Haupt-Trägheitsachsen relativ zum Ursprungs-Frame des Objekts. Die Haupt-Trägheitsachsen verlaufen durch den Massen-Schwerpunkt.
  - Felder **jX**, **jY**, **jZ**: Haupt-Trägheitsmomente

**Beispiel**

Haupt-Trägheitsmoment **jX**:

**jX** ist die Trägheit um die X-Achse der Haupt-Trägheitsachsen. Dieses ist durch **MS A**, **MS B** und **MS C** relativ zum Ursprungs-Frame des Objekts verdreht und im Massen-Schwerpunkt verschoben.

**jY** und **jZ** sind analog die Haupt-Trägheitsmomente um die Y- und Z-Achse.

### 9.3.7 Sicherheitsgerichtetes Werkzeug

**Beschreibung**

In einem Sunrise-Projekt kann maximal 1 sicherheitsgerichtetes Werkzeug festgelegt werden, das mit bis zu 6 konfigurierbaren Kugeln modelliert wird. Ein sicherheitsgerichtetes Werkzeug ist defaultmäßig an alle im Projekt verfügbaren Kinematiken gekoppelt.

Die Eigenschaften des sicherheitsgerichteten Werkzeugs sind für folgende konfigurierbaren Sicherheitsfunktionen relevant:

- Überwachung kartesischer Räume  
Die Kugeln können gegen die Grenzen aktiver kartesischer Überwachungsräume überwacht werden.  
(>>> 13.8.9 "Überwachungsräume" Seite 220)
- Überwachung der translatorischen kartesischen Geschwindigkeit  
Die Geschwindigkeit der Kugelmittelpunkte wird überwacht.  
(>>> 13.8.8 "Geschwindigkeitsüberwachungen" Seite 213)
- Kollisionserkennung und TCP-Kraftüberwachung  
Nur korrekt angegebene Lastdaten garantieren die Genauigkeit dieser Überwachungen. Die Lastdaten des sicherheitsgerichteten Werkzeugs, insbesondere die Werkzeugmasse und der Massen-Schwerpunkt sind zu konfigurieren. Bei Werkzeugen mit vergleichsweise hohen Trägheitsmomenten ( $> 0,1 \text{ kg}\cdot\text{m}^2$ ) sind darüber hinaus auch diese Daten anzugeben, um die Genauigkeit dieser Überwachungen zu gewährleisten.

Das sicherheitsgerichtete Werkzeug wird per Synchronisation auf die Robotersteuerung übertragen und nach einem Neustart der Robotersteuerung aktiviert. D. h. es ist für die Sicherheitssteuerung permanent aktiv, unabhängig davon, welches Werkzeug in der Applikation verwendet wird oder in den Handverfahroptionen eingestellt ist.

**Sicherheitsgerichtete Frames**

Ein sicherheitsgerichtetes Werkzeug kann wie jedes Werkzeug beliebig viele Frames besitzen. Um die Überwachungskugeln zu konfigurieren, müssen geeignete Frames als sicherheitsgerichtete Frames festgelegt werden. Der Kugelmittelpunkt liegt definitionsgemäß im Ursprung des sicherheitsgerichteten Frames. Der Kugelradius wird in den Frame-Eigenschaften definiert.

Werden Werkstücke verwendet, die bei der sicherheitsgerichteten kartesischen Raum- oder Geschwindigkeitsüberwachung berücksichtigt werden sollen, beispielsweise aufgrund der Ausmaße dieser Werkstücke, müssen die Kugeln des sicherheitsgerichteten Werkzeugs entsprechend konfiguriert werden.

Sicherheitsgerichtete Frames sind außerdem für folgende konfigurierbaren werkzeugbezogenen Sicherheitsüberwachungen relevant:

- Überwachung der Werkzeugorientierung (nur für Roboter verfügbar)  
Einer der sicherheitsgerichteten Frames kann als Werkzeugsorientierungs-Frame definiert werden. Die Orientierung dieses Frames kann über die AMF *Werkzeugorientierung* sicherheitsgerichtet überwacht werden.  
(>>> 13.8.10 "Überwachung der Werkzeugorientierung" Seite 228)
- Richtungsabhängige Überwachung der kartesischen Geschwindigkeit (für Roboter und mobile Plattformen verfügbar)  
Einer der sicherheitsgerichteten Frames kann als Überwachungspunkt für die werkzeugbezogene Geschwindigkeitsüberwachung definiert werden. Dazu kann ein zweiter Frame als Orientierung für die Überwachung definiert werden. Dieser Orientierungs-Frame legt die Orientierung des Koordinatensystems fest, in dem die Geschwindigkeit des Überwachungspunkts beschrieben wird. In der werkzeugbezogenen Geschwindigkeitsüberwachung kann eine Komponente dieser Geschwindigkeit überwacht werden.  
(>>> 13.8.8.3 "Richtungsabhängige Überwachung der kartesischen Geschwindigkeit" Seite 215)

### Beispiel

Für einen sicherheitsgerichteten Greifer sind 3 Überwachungskugeln konfiguriert.

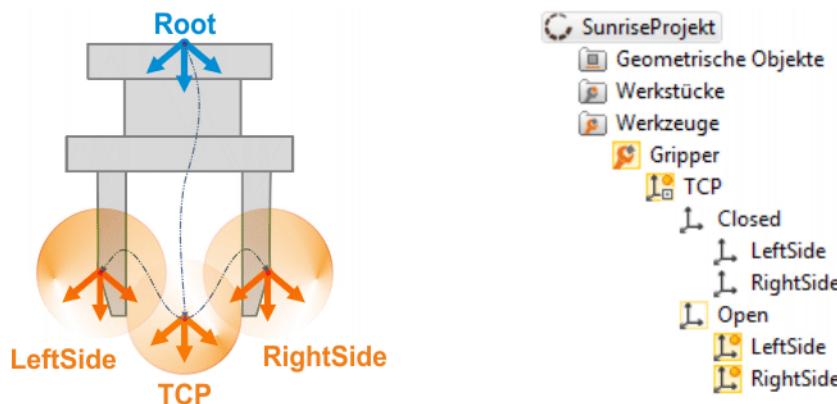


Abb. 9-7: Sicherheitsgerichteter Greifer

#### 9.3.7.1 Sicherheitsgerichtetes Werkzeug definieren

##### Voraussetzung

- Werkzeug und zugehörige Frames sind angelegt.
- Bei Verwendung der AMFs *Kollisionserkennung* und *TCP-Kraftüberwachung*: Die korrekten Lastdaten des Werkzeugs, insbesondere die Werkzeugmasse und der Massen-Schwerpunkt, sind bekannt.

##### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Objektvorlagen** das Werkzeug markieren, das sicherheitsgerichtet sein soll.
3. In der Sicht **Eigenschaften** die Eigenschaften des sicherheitsgerichteten Werkzeug festlegen.
  - Falls noch nicht festgelegt, die Lastdaten des Werkzeugs eingeben. Werkzeuge mit Lastdaten außerhalb des vorgegebenen Wertebereichs können nicht als sicherheitsgerichtete Werkzeuge verwendet werden.  
(>>> "Lastdaten" Seite 150)
  - Die Eigenschaft **Sicherheitsgerichtet** auf **Ja** setzen.

Das Werkzeug-Icon in der Sicht **Objektvorlagen** wird gelb hinterlegt.

Die Eigenschaften des Werkzeugs werden um die Sicherheitsparameter erweitert, die für die Konfiguration werkzeugbezogener Überwachungen benötigt werden.

(>>> "Sicherheitsparameter" Seite 151)

4. In der Sicht **Objektvorlagen** den Frame markieren, der sicherheitsgerichtet sein soll.

5. In der Sicht **Eigenschaften** die Eigenschaften des sicherheitsgerichteten Frames festlegen.

- Den Radius der Überwachungskugel eingeben.

- Falls noch nicht festgelegt, die Transformationsdaten des Frames bezogen auf seinen Eltern-Frame eingeben.

Frames mit Transformationsdaten außerhalb des vorgegebenen Wertebereichs können nicht als sicherheitsgerichtete Frames verwendet werden.

(>>> "Frames" Seite 151)

- Die Eigenschaft **Sicherheitsgerichtet** auf **Ja** setzen.

Das Frame-Icon in der Sicht **Objektvorlagen** wird gelb hinterlegt und mit einem Kugel-Symbol markiert.

6. Schritt 4 bis 5 wiederholen, um weitere sicherheitsgerichtete Frames zu definieren.

7. Bei Bedarf die Sicherheitsparameter einstellen, die für werkzeugbezogene Sicherheitsüberwachungen benötigt werden:

- a. In der Sicht **Objektvorlagen** das sicherheitsgerichtete Werkzeug markieren.
- b. In der Sicht **Eigenschaften** den gewünschten Sicherheitsparametern einen sicherheitsgerichteten Frame zuweisen.

(>>> "Sicherheitsparameter" Seite 151)

Das Frame-Icon in der Sicht **Objektvorlagen** wird um einen Pfeil erweitert.



Alternatives Vorgehen, um ein Werkzeug oder einen Frame als sicherheitsgerichtet zu markieren:

- In der Sicht **Objektvorlagen** auf das Werkzeug oder den Frame rechtsklicken und im Kontextmenü **Sicherheitsgerichtet** wählen.



Alternatives Vorgehen, um sicherheitsgerichtete Frames als Sicherheitsparameter für werkzeugbezogene Sicherheitsüberwachungen festzulegen:

- In der Sicht **Objektvorlagen** auf den gewünschten sicherheitsgerichteten Frame rechtsklicken und im Kontextmenü den gewünschten Sicherheitsparameter wählen.
- In der Sicht **Objektvorlagen** den sicherheitsgerichteten Frame markieren und in der Sicht **Eigenschaften** den gewünschten Sicherheitsparameter auf **Ja** setzen.

## Lastdaten

Lastdaten des sicherheitsgerichteten Werkzeugs:

<b>Parameter</b>	<b>Beschreibung</b>
<b>Masse</b>	Masse des sicherheitsgerichteten Werkzeugs ■ <b>≤2 000 kg</b>
<b>MS X, MS Y, MS Z</b>	Lage des Massen-Schwerpunkts relativ zum Ursprungs-Frame des sicherheitsgerichteten Werkzeugs ■ <b>-10 000 mm ... +10 000 mm</b>
<b>MS A, MS B, MS C</b>	Orientierung der Haupt-Trägheitsachsen relativ zum Ursprungs-Frame des sicherheitsgerichteten Werkzeugs ■ <b>Beliebig</b>
<b>jX, jY, jZ</b>	Massen-Trägheitsmomente des sicherheitsgerichteten Werkzeugs ■ <b>0 kg·m<sup>2</sup>... 1 000 kg·m<sup>2</sup></b>



Um unnötigen Verifikationsaufwand zu vermeiden, ein Werkzeug erst dann als sicherheitsgerichtet markieren, wenn die Lastdaten korrekt eingegeben oder ermittelt und in Sunrise.Workbench übertragen wurden.

Weitere Informationen zu den Lastdaten sind hier zu finden:  
(>>> 9.3.6 "Lastdaten" Seite 147)

## Frames

Eigenschaften sicherheitsgerichteter Frames:

<b>Parameter</b>	<b>Beschreibung</b>
<b>Radius</b>	Radius der Kugel am sicherheitsgerichteten Frame ■ <b>25 mm ... 10 000 mm</b>
<b>X, Y, Z</b>	Verschiebung des sicherheitgerichteten Frames entlang der Achsen des Eltern-Frames ■ <b>-10 000 mm ... +10 000 mm</b>
<b>A, B, C</b>	Orientierung des sicherheitgerichteten Frames bezogen auf den Eltern-Frame ■ <b>Beliebig</b>

## Sicherheitsparameter

Sicherheitsparameter des sicherheitsgerichteten Werkzeugs:

Parameter	Beschreibung
<b>Werkzeugorientierungs-Frame</b>	<p>Sicherheitsgerichteter Frame, dessen Orientierung mithilfe der AMF <i>Werkzeugorientierung</i> überwacht werden kann.</p> <p>Wird kein Werkzeugorientierungs-Frame eingestellt, wird der Ursprungs-Frame des Werkzeugs (= Flansch-Koordinatensystem des Roboters) als Werkzeugorientierungs-Frame verwendet.</p>
<b>Punkt für die werkzeugbezogene Geschwindigkeit</b>	<p>Sicherheitsgerichteter Frame, der einen Punkt am Werkzeug definiert, an dem die kartesische Geschwindigkeit in einer bestimmten Richtung mithilfe der AMF <i>Werkzeugbezogene Geschwindigkeitskomponente</i> überwacht werden kann.</p> <p>Wird kein Punkt für die werkzeugbezogene Geschwindigkeit definiert, wird der Ursprungs-Frame des Werkzeugs (= Flansch-Koordinatensystem des Roboters) verwendet, d. h. die Geschwindigkeit am Ursprung des Flansch-Koordinatensystems wird überwacht.</p> <p><b>Hinweis:</b> Bei einer mobilen Plattform entspricht der Ursprungs-Frame des Werkzeugs dem Koordinatensystem im Mittelpunkt der Plattform.</p>
<b>Orientierung für die werkzeugbezogene Geschwindigkeit</b>	<p>Sicherheitsgerichteter Frame, dessen Orientierung bestimmt, in welche Richtungen die kartesische Geschwindigkeit mithilfe der AMF <i>Werkzeugbezogene Geschwindigkeitskomponente</i> überwacht werden kann.</p> <p>Wird keine Orientierung für die werkzeugbezogene Geschwindigkeit definiert, wird der Ursprungs-Frame des Werkzeugs (= Flansch-Koordinatensystem des Roboters) verwendet, d. h. die Orientierung des Flansch-Koordinatensystems bestimmt die Überwachungsrichtung.</p> <p><b>Hinweis:</b> Bei einer mobilen Plattform entspricht der Ursprungs-Frame des Werkzeugs dem Koordinatensystem im Mittelpunkt der Plattform.</p>

### 9.3.8 Sicherheitsgerichtete Werkstücke

#### Beschreibung

Vom Roboter aufgenommene Lasten, z. B. ein gegriffenes Werkstück, üben eine zusätzliche Kraft auf den Roboter aus und haben Einfluss auf die von den Gelenkmomenten-Sensoren gemessenen Momente. Deswegen müssen die Lastdaten von aufgenommenen Werkstücken bei der sicheren Überwachung von Kollisionen und Kräften berücksichtigt werden. Hierzu ist es erforderlich, diese Werkstücke als sicherheitsgerichtete Werkstücke zu konfigurieren.

Für ein Sunrise-Projekt können maximal 8 sicherheitsgerichtete Werkstücke konfiguriert werden.

Die Eigenschaften sicherheitsgerichteter Werkstücke sind für folgende konfigurierbaren Sicherheitsfunktionen relevant:

- Kollisionserkennung

Die Lastdaten des aktiven sicherheitsgerichteten Werkstücks werden bei der Berechnung des externen Moments berücksichtigt.

(>>> 13.8.13.2 "Kollisionserkennung" Seite 234)

- TCP-Kraftüberwachung

Die Lastdaten des schwersten sicherheitsgerichteten Werkstücks werden bei der Ermittlung der unterhalb des Flansches angreifenden kartesischen Kraft berücksichtigt.

(>>> 13.8.13.3 "TCP-Kraftüberwachung" Seite 235)

Während eines Prozesses kann es durch Aufnehmen und Ablegen unterschiedlicher Werkstücke zu Lastwechseln kommen. Bei der TCP-Kraftüberwachung verwendet die Sicherheitssteuerung automatisch die Lastdaten des schwersten sicherheitsgerichteten Werkstücks. Bei der Kollisionserkennung muss der Benutzer der Sicherheitssteuerung explizit mitteilen, welches sicherheitsgerichtete Werkstück aktuell aktiv ist.

(>>> 15.11.6 "Kommandieren von Lastwechseln an Sicherheitssteuerung" Seite 317)

Ist ein sicherheitsgerichtetes Werkstück aktiviert, werden seine Lastdaten von der Sicherheitssteuerung dauerhaft berücksichtigt. Soll dieses Werkstück deaktiviert oder ein anderes sicherheitsgerichtetes Werkstück aktiviert werden, muss dies explizit kommandiert werden. Nach einem Neustart der Robotersteuerung ist kein sicherheitsgerichtetes Werkstück aktiviert.

Das Aktivieren von sicherheitsgerichteten Werkstücken erfolgt nicht sicherheitsgerichtet im Quellcode von Roboter-Applikationen und Hintergrund-Tasks. Daher ist es im Fehlerfall möglich, dass Kollisionserkennung und TCP-Kraftüberwachung Lastdaten verwenden, die von der tatsächlichen Werkstücklast abweichen. Diese Abweichungen werden als externes Moment oder externe TCP-Kraft fehlinterpretiert. Die maximale Abweichung entspricht bei geringen Geschwindigkeiten und Beschleunigungen der Gewichtskraft des schwersten Werkstücks, das in der Applikation aufgenommen werden kann.



Bei Verwendung der AMFs *Kollisionserkennung* und *TCP-Kraftüberwachung* wird empfohlen, alle Werkstücke, die vom Roboter aufgenommen werden, während eine der Überwachungen aktiv ist, als sicherheitsgerichtete Werkstücke zu konfigurieren.



Bei Verwendung der AMF *TCP-Kraftüberwachung* ist es notwendig, das schwerste Werkstück, das vom Roboter aufgenommen wird, während die Überwachung aktiv ist, als sicherheitsgerichtetes Werkstück zu konfigurieren. Eine fehlerhafte Konfiguration des schwersten Werkstücks kann zu einem Verlust der Sicherheitsintegrität der TCP-Kraftüberwachung führen.

Wie sich die Lastdaten eines Werkstücks bei der Kollisionsüberwachung auswirken, ist abhängig davon, wie das Werkstück aufgenommen wird. Die Sicherheitssteuerung setzt bei einem sicherheitsgerichteten Werkstück voraus, dass der Ursprungs-Frame des sicherheitsgerichteten Werkstücks mit dem Standard-Frame für Bewegungen des sicherheitsgerichteten Werkzeugs identisch ist.

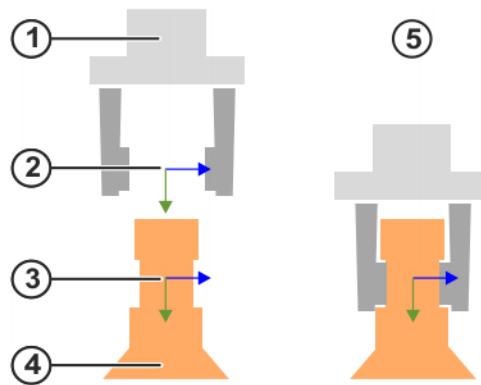


Abb. 9-8: Konfiguration sicherheitsgerichtetes Werkstück

Pos.	Beschreibung
1	Sicherheitsgerichtetes Werkzeug
2	Standard-Frame für Bewegungen des sicherheitsgerichteten Werkzeugs  Frame des sicherheitsgerichteten Werkzeugs, an dem das sicherheitsgerichtete Werkstück aufgenommen werden muss. Dieser Frame muss kein sicherheitsgerichteter Frame sein.
3	Ursprungs-Frame des sicherheitsgerichteten Werkstücks  Frame des sicherheitsgerichteten Werkstücks, an dem das sicherheitsgerichtete Werkzeug das Werkstück aufnehmen muss.
4	Sicherheitsgerichtetes Werkstück
5	Zustand nach dem Aktivieren des sicherheitsgerichteten Werkstücks  Der Ursprungs-Frame des sicherheitsgerichteten Werkstücks ist mit dem Standard-Frame für Bewegungen des sicherheitsgerichteten Werkzeugs identisch.

### 9.3.8.1 Sicherheitsgerichtete Werkstücke definieren

#### Voraussetzung

- Werkstück ist angelegt.
- Bei Verwendung der AMFs *Kollisionserkennung* und *TCP-Kraftüberwachung*: Die korrekten Lastdaten des Werkstücks, insbesondere die Werkstückmasse und der Massen-Schwerpunkt, sind bekannt.

#### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. In der Sicht **Objektvorlagen** das Werkstück markieren, das sicherheitsgerichtet sein soll.
3. In der Sicht **Eigenschaften** die Eigenschaften des sicherheitsgerichteten Werkstücks festlegen.
  - Falls noch nicht festgelegt, die Lastdaten des Werkstücks eingeben. Werkstücke mit Lastdaten außerhalb des vorgegebenen Wertebereichs können nicht als sicherheitsgerichtete Werkstücke verwendet werden.  
(>>> "Lastdaten" Seite 155)
  - Die Eigenschaft **Sicherheitsgerichtet** auf **Ja** setzen. Das Werkzeug-Icon in der Sicht **Objektvorlagen** wird gelb hinterlegt.

Nach der Synchronisation des Projekts kann das sicherheitsgerichtete Werkstück in der Applikation aktiviert werden.

 Alternatives Vorgehen, um ein Werkstück als sicherheitsgerichtet zu markieren:

- In der Sicht **Objektvorlagen** auf das Werkstück rechtsklicken und im Kontextmenü **Sicherheitsgerichtet** wählen.

## Lastdaten

Lastdaten des sicherheitsgerichteten Werkstücks:

Parameter	Beschreibung
<b>Masse</b>	Masse des sicherheitsgerichteten Werkstücks <ul style="list-style-type: none"> <li>■ <b>0,001 kg ... 2 000 kg</b></li> </ul>
<i>MS X, MS Y, MS Z</i>	Lage des Massen-Schwerpunkts relativ zum Ursprungs-Frame des sicherheitsgerichteten Werkstücks <ul style="list-style-type: none"> <li>■ <b>-10 000 mm ... +10 000 mm</b></li> </ul>
<i>MS A, MS B, MS C</i>	Orientierung der Haupt-Trägheitsachsen relativ zum Ursprungs-Frame des sicherheitsgerichteten Werkstücks (Nicht relevant für sicherheitsgerichtete Überwachung) <ul style="list-style-type: none"> <li>■ Beliebig</li> </ul>
<i>jX, jY, jZ</i>	Massen-Trägheitsmomente des sicherheitsgerichteten Werkstücks (Nicht relevant für sicherheitsgerichtete Überwachung) <ul style="list-style-type: none"> <li>■ Beliebig</li> </ul>

Weitere Informationen zu den Lastdaten sind hier zu finden:

(>>> 9.3.6 "Lastdaten" Seite 147)

## 9.4 Synchronisation von Projekten

### Übersicht

Bei der Synchronisation von Projekten werden die Projektdaten zwischen Sunrise.Workbench und Robotersteuerung übertragen. Dabei werden die Projekte miteinander verglichen. Liegen unterschiedliche Projekte oder Versionsunterschiede vor, kann der Benutzer auswählen, in welche Richtung die Projektdaten übertragen werden sollen. Folgende Fälle werden unterschieden:

- Projekt ist nur in Sunrise.Workbench vorhanden.
- Projekt auf der Robotersteuerung wird durch ein anderes Projekt ersetzt.
- Unterschiedliche Versionen des Projekts liegen vor:
  - Bei Änderung der Projektdaten nur in Sunrise.Workbench
  - Bei Änderung der Projektdaten nur auf der Robotersteuerung
  - Bei Änderung der Projektdaten auf beiden Seiten

### 9.4.1 Projekt auf Robotersteuerung übertragen

#### Beschreibung

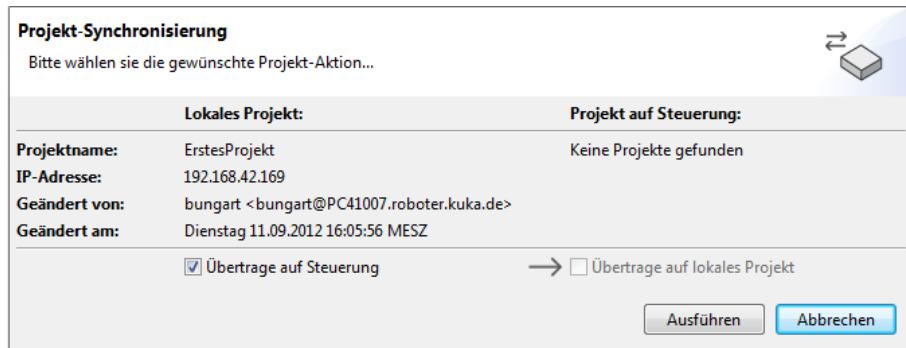
Die hier beschriebene Vorgehensweise gilt für den Fall, dass sich auf der Robotersteuerung noch kein Projekt oder ein anderes als das zu übertragende Projekt befindet.

#### Voraussetzung

- Netzwerk-Verbindung zur Robotersteuerung
  - Das zu übertragende Projekt enthält mindestens eine Applikation.
  - Die System Software ist installiert.
- (>>> 10.2 "System Software installieren" Seite 160)

- Die installierte System Software ist mit der Stationskonfiguration des zu übertragenden Projekts kompatibel.

<b>Vorgehensweise</b>	<p>Die hier beschriebene Vorgehensweise gilt für den Fall, dass sich auf der Robotersteuerung noch kein Projekt oder ein anderes als das zu übertragende Projekt befindet.</p> <ol style="list-style-type: none"> <li>1. Das zu übertragende Projekt im <b>Paket-Explorer</b> markieren.</li> <li>2. In der Symbolleiste auf den Button <b>Projekt synchronisieren</b> klicken. Es wird abgefragt, ob bereits Projektdaten auf der Steuerung vorhanden sind. Wenn die Abfrage fehlschlägt, wird die Fehlerursache in einer Meldung angezeigt.</li> <li>3. Bei erfolgreicher Abfrage öffnet sich das Fenster <b>Projekt-Synchronisierung</b>.</li> </ol>
-----------------------	---



**Abb. 9-9: Projekt auf Steuerung übertragen**

4. Auf **Ausführen** klicken. Das Projekt wird auf die Robotersteuerung übertragen.  
Der Fortschritt wird sowohl in Sunrise.Workbench als auch am smartPAD in einem Dialog angezeigt. Wenn die Übertragung abgeschlossen ist, wird der Dialog automatisch geschlossen.
5. Wenn die Übertragung fehlschlägt, wird sowohl in Sunrise.Workbench als auch am smartPAD ein entsprechender Dialog angezeigt. In Sunrise.Workbench wird zusätzlich die Fehlerursache angezeigt.  
Dialog in Sunrise.Workbench und am smartPAD mit **OK** bestätigen.
6. Ist die Übertragung erfolgreich, wird bei einer Änderung der Sicherheitskonfiguration oder der E/A-Konfiguration ein Neustart der Robotersteuerung angefordert.  
Die Aufforderung zum Neustart mit **OK** bestätigen. Die Robotersteuerung wird neu gestartet.
7. Bei einer Änderung der Sicherheitskonfiguration diese auf der Robotersteuerung aktivieren.  
**(>>> 13.7 "Sicherheitskonfiguration auf Robotersteuerung aktivieren"**  
Seite 205)

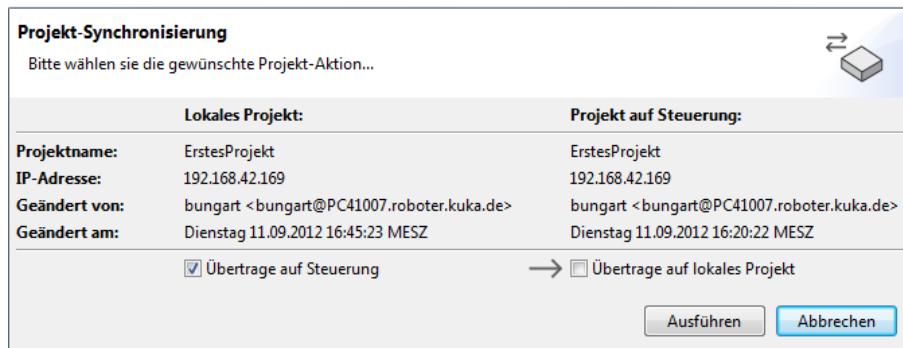
Das übertragene Projekt ist nun auf der Steuerung aktiv. Es stehen alle angelegten Projektdaten zur Verfügung.

#### 9.4.2 Projekt auf Robotersteuerung oder in Sunrise.Workbench aktualisieren

<b>Beschreibung</b>	Die hier beschriebene Vorgehensweise gilt für den Fall, dass auf beiden Seiten das gleiche Projekt vorliegt, jedoch in unterschiedlichen Versionen.
<b>Voraussetzung</b>	<ul style="list-style-type: none"> <li>■ Netzwerk-Verbindung zur Robotersteuerung</li> <li>■ Nur wenn ein Projekt auf die Robotersteuerung übertragen wird: Auf der Robotersteuerung läuft keine Applikation.</li> </ul>

**Vorgehensweise**

1. Das zu übertragende Projekt im **Paket-Explorer** markieren.
  2. In der Symbolleiste auf den Button **Projekt synchronisieren** klicken. Es wird abgefragt, ob bereits Projektdaten auf der Steuerung vorhanden sind. Wenn die Abfrage fehlschlägt, wird die Fehlerursache in einer Meldung angezeigt.
  3. Ist das Projekt in Sunrise.Workbench mit dem Projekt auf der Steuerung identisch, ist keine Synchronisation nötig. Der Vorgang wird automatisch abgebrochen.
- Bei erfolgreicher Abfrage öffnet sich das Fenster **Projekt-Synchronisierung**.



**Abb. 9-10: Projekt aktualisieren**

Es werden Informationen zu beiden Projekten angezeigt. Die Richtung, in die synchronisiert wird, ist defaultmäßig so gesetzt, dass die aktuellere Projektversion übertragen wird.

Wurden auf beiden Seiten Änderungen der Projektdaten vorgenommen, wird dies als Konflikt erkannt und ein Warnhinweis angezeigt. Die Richtung, in die synchronisiert wird, kann eingestellt werden:

- Häkchen bei **Übertrage auf Steuerung** gesetzt: Projekt wird von Sunrise.Workbench auf die Robotersteuerung übertragen.
  - Häkchen bei **Übertrage auf lokales Projekt** gesetzt: Projekt wird von der Robotersteuerung in Sunrise.Workbench übertragen.
4. Bei Bedarf die Richtung, in die synchronisiert werden soll, ändern und auf **Ausführen** klicken. Das Projekt wird übertragen. Bei Übertragung der älteren Version wird ein Warnhinweis angezeigt.  
Der Fortschritt wird sowohl in Sunrise.Workbench als auch am smartPAD in einem Dialog angezeigt. Wenn die Übertragung abgeschlossen ist, wird der Dialog automatisch geschlossen.
  5. Wenn die Übertragung fehlschlägt, wird sowohl in Sunrise.Workbench als auch am smartPAD ein entsprechender Dialog angezeigt. In Sunrise.Workbench wird zusätzlich die Fehlerursache angezeigt.  
Dialog in Sunrise.Workbench und am smartPAD mit **OK** bestätigen.
  6. Nur bei erfolgreicher Übertragung auf Robotersteuerung:
    - Bei einer Änderung der Sicherheitskonfiguration oder der E/A-Konfiguration wird ein Neustart der Robotersteuerung angefordert.  
Die Aufforderung zum Neustart mit **OK** bestätigen. Die Robotersteuerung wird neu gestartet.
    - Bei einer Änderung der Sicherheitskonfiguration diese auf der Robotersteuerung aktivieren.  
(>>> 13.7 "Sicherheitskonfiguration auf Robotersteuerung aktivieren"  
Seite 205)

## 9.5 Projekt von Robotersteuerung laden

**Beschreibung** Es ist möglich ein Projekt, das sich nicht im Arbeitsbereich von Sunrise.Workbench befindet, von der Robotersteuerung zu laden.

**Voraussetzung**

- Netzwerk-Verbindung zur Robotersteuerung
- Im Arbeitsbereich existiert kein Projekt mit dem gleichem Namen wie das zu ladende Projekt.

**Vorgehensweise**

1. Menüfolge **Datei > Neu > Sunrise Projekt** wählen. Der Assistent für die Projekterstellung öffnet sich.
2. Im Feld **Steuerungs-IP-Adresse** die IP-Adresse der Robotersteuerung eingeben, von der das Projekt geladen werden soll.



Die IP-Adresse der Robotersteuerung kann auf der smartHMI angezeigt werden. ([>>> 6.16.6 "IP-Adresse und Software-Version anzeigen"](#) Seite 101)

3. Den Radio-Button **Projekt von Steuerung laden** aktivieren.
4. Auf **Weiter >** klicken. Es wird geprüft, ob ein Projekt auf der Steuerung vorhanden ist.
5. Ist ein Projekt auf der Steuerung vorhanden und im Arbeitsbereich existiert kein Projekt mit identischem Namen, wird eine Zusammenfassung mit Informationen zum Projekt angezeigt.  
Auf **Fertigstellen** klicken. Das Projekt wird im Arbeitsbereich erstellt und dann im **Paket-Explorer** angezeigt.

## 10 Stationskonfiguration und Installation

### 10.1 Stationskonfiguration öffnen

- Vorgehensweise**
- Im **Paket-Explorer** auf die Datei **StationSetup.cat** doppelklicken.
- Die Datei enthält die Stationskonfiguration des Projekts. Diese kann über folgende Registerkarten bearbeitet und installiert werden:
- |                      |   |
|----------------------|---|
| <b>Topologie</b>     | Die Registerkarte <b>Topologie</b> zeigt die Hardware-Komponenten der Station an. Die Topologie kann neu aufgebaut oder geändert werden.  |
| <b>Software</b>      | Die Registerkarte <b>Software</b> zeigt die Software-Komponenten der Station an. Die zu installierenden Komponenten sowie ihre Version können ausgewählt werden. Die auswählbaren Komponenten sind abhängig von der erstellten Topologie. |
| <b>Konfiguration</b> | Die Registerkarte <b>Konfiguration</b> zeigt die Konfiguration der Robotersteuerung an. Die Konfiguration kann geändert werden.   |
| <b>Installation</b>  | In der Registerkarte <b>Installation</b> wird die System Software auf der Robotersteuerung installiert.   |

#### 10.1.1 Parameter für Vermessung konfigurieren

- Vorgehensweise**
1. Stationskonfiguration öffnen und Registerkarte **Konfiguration** wählen.
  2. Im Katalogelement **SmartHMI** unter **Vermessung** die Parameter wie gewünscht konfigurieren.
  3. Stationskonfiguration speichern. Es wird abgefragt, ob die Änderungen in das Projekt übernommen werden sollen. Auf **Speichern und übernehmen** klicken.

Übersicht	Parameter	Beschreibung
	<b>Minimaler Messpunktabstand (Werkzeug) [mm]</b>	Mindestabstand, den 2 Messpunkte bei der Werkzeugvermessung (XYZ 4-Punkt- und ABC 2-Punkt-Methode) einhalten müssen ■ <b>0 ... 200</b> Default: 8
	<b>Maximaler Berechnungsfehler [mm]</b>	Maximaler translatorischer Berechnungsfehler bei der Werkzeugvermessung, bis zu dem die Qualität der Vermessung als ausreichend angesehen wird ■ <b>0 ... 200</b> Default: 50
	<b>Minimaler Messpunktabstand (Basis) [mm]</b>	Mindestabstand, den 2 Messpunkte bei der Basisvermessung einhalten müssen ■ <b>0 ... 200</b> Default: 50
	<b>Minimaler Winkel in °</b>	Mindestwinkel zwischen den Geraden, die durch die 3 Messpunkte bei der Basisvermessung (3-Punkt-Methode) definiert sind, und der eingehalten werden muss ■ <b>0 ... 360</b> Default: 2.5

## 10.2 System Software installieren

### Voraussetzung

- Stationskonfiguration ist abgeschlossen.
- Netzwerk-Verbindung zur Robotersteuerung

### Vorgehensweise

1. Registerkarte **Installation** wählen.
2. Im Fenster **Installationsereignisse** werden defaultmäßig die Warnungen und Fehler angezeigt, die während der Installation auftreten: Das Häkchen bei **Nur Warnungen und Fehler anzeigen**. ist gesetzt.  
Um alle bei der Installation auftretenden Ereignisse anzuzeigen, das Häkchen bei **Nur Warnungen und Fehler anzeigen**. entfernen.
3. Auf **Installieren** klicken. Die Installation wird vorbereitet und das Fenster **Installation** öffnet sich. Das Feld **Konfigurierte IP** ist farblich markiert:
  - Grün markiert: Die Netzwerk-Verbindung zur Robotersteuerung wurde hergestellt, eine Installation ist möglich.
  - Rot markiert: Die Netzwerk-Verbindung zur Robotersteuerung konnte nicht hergestellt werden. Ursachen können sein, dass das Netzwerkkabel nicht korrekt angeschlossen ist oder die konfigurierte nicht mit der tatsächlichen IP-Adresse der Robotersteuerung übereinstimmt.



**Abb. 10-1: Robotersteuerung nicht erreichbar**

4. Nur wenn das Feld **Konfigurierte IP** rot markiert ist:
  - Stimmt die konfigurierte mit der tatsächlichen IP-Adresse der Robotersteuerung überein, besteht keine Netzwerk-Verbindung zur Robotersteuerung. Netzwerk-Verbindung herstellen.
  - Besitzt die Robotersteuerung eine andere als die konfigurierte IP-Adresse, ist die aktuelle IP-Adresse der Robotersteuerung im Feld **Tatsächliche IP** einzutragen. Dazu in das Feld doppelklicken.
5. Auf **OK** klicken, um die Installation fortzusetzen.
6. Nur relevant, wenn die IP-Adresse im Feld **Tatsächliche IP** geändert wurde: Wenn die rote Markierung unter **Konfigurierte IP** bestehen bleibt oder die Installation fehlschlägt, besteht keine Netzwerk-Verbindung zur Robotersteuerung mit dieser IP-Adresse.  
Netzwerk-Verbindung herstellen und Installationsprozess neu starten (zurück zu Schritt 3).
7. Meldung mit der Aufforderung zum Neustart mit **OK** bestätigen. Die Robotersteuerung wird neu gestartet und die Installation wird abgeschlossen.



Bei der Installation wird die Sicherheitskonfiguration auf die Robotersteuerung übertragen, ist aber noch nicht aktiv. Um den Roboter verfahren zu können, muss die Sicherheitskonfiguration über die smartHMI aktiviert werden. ([>>> 13.7 "Sicherheitskonfiguration auf Robotersteuerung aktivieren" Seite 205](#))



Der in der Stationskonfiguration des Sunrise-Projekts ausgewählte Robotertyp sowie der eingestellte Medien-Flansch werden bei der Installation in den Konfigurationsdaten auf der Robotersteuerung gespeichert. Stimmen diese Angaben nicht mit den entsprechenden Daten des elektronischen Typenschildes des an der Robotersteuerung angeschlossenen Roboters überein, kann der Roboter nicht verfahren werden.

## **Neuinstallation**

In folgenden Fällen ist eine Neuinstallation der System Software erforderlich:

- Änderung der Stationskonfiguration in der Registerkarte **Topologie**
- Änderung der Stationskonfiguration in der Registerkarte **Software**

Beispiele:

- Installation zusätzlicher Software-Pakete
- Software-Update
- Inkompatible Versionsänderungen bereits vorhandener Software-Pakete

Inkompatible Versionsänderungen können vorliegen, wenn ein Projekt nicht mit der Version von Sunrise.Workbench geöffnet wird, mit der es erstellt wurde.

(>>> 15.6 "Versionsinformationen RoboticsAPI" Seite 294)

- Änderung der Stationskonfiguration in der Registerkarte **Konfiguration**

### **10.2.1 Sicherheitskonfiguration in neue Software-Version konvertieren**

#### **Beschreibung**

Wenn eine neue Software-Version von Sunrise.Workbench installiert wird, kann ein Sunrise-Projekt, das mit einer früheren Software-Version erstellt wurde, in den Arbeitsbereich geladen und weiter verwendet werden.

Beim Laden des Sunrise-Projekts ändert sich die Stationskonfiguration. Durch Speichern der Stationskonfiguration wird die zugehörige Sicherheitskonfiguration in die neue Version überführt.

#### **Voraussetzung**

- Sunrise-Projekt ist archiviert oder in einem beliebigen Verzeichnis gesichert.
- Neue Version von Sunrise.Workbench ist installiert.

#### **Vorgehensweise**

1. Sunrise-Projekt in den Arbeitsbereich laden.
2. Stationskonfiguration des Projekts öffnen und auf **Speichern** klicken.
3. Es wird abgefragt, ob die Änderungen in das Projekt übernommen werden sollen. Auf **Speichern und übernehmen** klicken.
4. Die Sicherheitskonfiguration wird aktualisiert und ihre Parameter werden konvertiert. Wenn der Vorgang abgeschlossen ist, wird dies in einer Meldung angezeigt. Mit **OK** bestätigen.
5. Um das aktualisierte Projekt auf der Robotersteuerung nutzen zu können, sind weitere Schritte erforderlich:
  - a. System Software installieren.
  - b. Projekt synchronisieren.
  - c. Sicherheitskonfiguration neu aktivieren.
  - d. Sicherheitsabnahme durchführen.

## **10.3 Sprachpaket installieren**

#### **Beschreibung**

Die Bedienoberfläche smartHMI steht aktuell in folgenden Sprachen zur Verfügung:

Deutsch	Italienisch
Englisch	Spanisch
Französisch	

Sprachen, die erst nach Auslieferung einer Software zur Verfügung stehen, können bei Bedarf nachträglich installiert werden.

- Voraussetzung**
- Sprachpaket ist in Sunrise.Workbench verfügbar.  
(>>> 4.4 "Sprachpaket in Sunrise.Workbench installieren" Seite 46)

- Vorgehensweise**
1. Stationskonfiguration öffnen und Registerkarte **Software** wählen.
  2. Das Software-Paket **SmartHMI LanguagePack** für die Installation auswählen:
    - In der Spalte **Installieren** das Häkchen setzen.
  3. Stationskonfiguration speichern. Es wird abgefragt, ob die Änderungen in das Projekt übernommen werden sollen. Auf **Speichern und übernehmen** klicken.
  4. System Software neu auf Robotersteuerung installieren. Nach dem Neustart der Robotersteuerung stehen die neu installierten Sprachen auf der smartHMI zur Auswahl.

## 10.4 VirensScanner installieren oder updaten

- Beschreibung**
- Die Installation besteht aus 2 Teilen:
- Installation des Virensackers in KUKA Sunrise.Workbench  
Der Virensacker ist eine Option, die nachträglich in Sunrise.Workbench installiert werden muss. Erst nach der Installation steht der Virensacker im Software-Katalog von Sunrise.Workbench zur Verfügung (Eintrag **Ikarus AntiVirus**).
  - Installation der System Software auf der Robotersteuerung  
Nach der Installation der System Software ist der Virensacker auf der Robotersteuerung aktiv. Auf der smartHMI steht dann eine Kachel zur Verfügung, mit der die Ansicht **Virensacker** geöffnet werden kann.  
(>>> 6.16.8 "Meldungen des Virensackers anzeigen" Seite 101)



Es wird empfohlen, vor einem Update des Virensackers zu überprüfen, welche Version aktuell auf der Robotersteuerung installiert ist.  
Keinen Downgrade ausführen.

- Voraussetzung**
- Datenträger mit der zu installierenden Software:
    - Datei **com.kuka.ikarus.repository-[Version].ZIP**
  - Bei einem Update auf der Robotersteuerung: Netzwerk-Verbindung ohne Internet-Zugang oder mit einer aktiven Firewall  
Während des Updates ist der Virensacker für eine kurze Zeit nicht aktiv.
- Vorgehensweise**
1. Menüfolge **Hilfe > Neue Software installieren ...** wählen. Das Fenster **Installieren** öffnet sich.
  2. Rechts vom Feld **Work with** auf **Hinzufügen** klicken. Das Fenster **Repository hinzufügen** öffnet sich.
  3. Auf **Archiv ...** klicken und zur Installationsdatei navigieren:  
Datei **com.kuka.ikarus.repository-[Version].ZIP**
  4. Die Datei markieren und mit **Öffnen** bestätigen. Das Feld **Position** zeigt nun den Pfad zu der Datei an.
  5. Auf **OK** klicken.

- Im Fenster **Installieren** zeigt das Feld **Work with** nun den Pfad zu der Datei an.
  - Das Fenster zeigt nun außerdem die Checkbox **KUKA.Ikarus T3** an.
6. Bei der Checkbox **KUKA.Ikarus T3** ein Häkchen setzen.
  7. Die weiteren Einstellungen im Fenster **Installieren** belassen wie sie sind und auf **Weiter >** klicken.
  8. Die Übersicht **Details zur Installation** wird angezeigt. Die Einstellungen belassen wie sie sind und auf **Weiter >** klicken.
  9. Eine Lizenzvereinbarung wird angezeigt. Damit der Virenschanner installiert werden kann, muss die Vereinbarung akzeptiert werden. Dann auf **Fertigstellen** klicken.  
Der Installationsvorgang beginnt.
  10. Eine Sicherheitswarnung bezüglich unsignierten Inhalts wird angezeigt. Mit **OK** bestätigen.
  11. Sicherheitsabfrage mit **Ok** bestätigen.
  12. Eine Meldung zeigt an, dass Sunrise.Workbench neu gestartet werden muss, um die Änderungen zu übernehmen. Auf **Jetzt neu starten** klicken.
  13. Sunrise.Workbench startet neu. Danach ist die Installation in Sunrise.Workbench abgeschlossen.
  14. System Software neu auf Robotersteuerung installieren. Nach einem Neustart der Robotersteuerung ist der Virenschanner auf der Robotersteuerung aktiv.



## 11 Buskonfiguration

### 11.1 Konfiguration und E/A-Verschaltung in WorkVisual – Übersicht

Schritt	Beschreibung
1	<p>Optionspaket <b>Sunrise</b> in WorkVisual installieren.</p> <p>Das Optionspaket liegt als KOP-Datei vor und wird mit Sunrise.Workbench ausgeliefert. Die Datei <b>Sunrise.kop</b> befindet sich auf dem Datenträger mit der zu installierenden Software (bei Auslieferung zu finden im Verzeichnis <b>WorkVisual AddOn</b>).</p> <p><b>Hinweis:</b> Es ist immer das Optionspaket zu verwenden, das mit Sunrise.Workbench ausgeliefert wird. Wenn eine alte Version von Sunrise.Workbench deinstalliert und eine neue Version installiert wird, muss auch das Optionspaket in WorkVisual getauscht werden.</p>
2	<p>Nur notwendig, wenn ein Medien-Flansch Touch oder FSoE-Slaves verwendet werden:</p> <ol style="list-style-type: none"> <li>1. Die Verzeichnisse <b>Config</b> und <b>SafeDevDescr</b> kopieren (befinden sich auf dem Datenträger mit der zu installierenden Software; bei Auslieferung zu finden im Verzeichnis <b>WorkVisual AddOn/FSoE_Patch</b>).</li> <li>2. Die Verzeichnisse <b>Config</b> und <b>SafeDevDescr</b> in das Installationsverzeichnis von WorkVisual kopieren (überschreibt die vorhandenen Verzeichnisse).</li> </ol>
3	<p>WorkVisual beenden und in Sunrise.Workbench eine neue E/A-Konfiguration anlegen oder eine bereits bestehende E/A-Konfiguration öffnen. WorkVisual wird dadurch automatisch gestartet und das zur E/A-Konfiguration gehörige WorkVisual-Projekt geöffnet.</p> <p>(&gt;&gt;&gt; 11.3 "Neue E/A-Konfiguration anlegen" Seite 166)</p> <p>(&gt;&gt;&gt; 11.4 "Bestehende E/A-Konfiguration öffnen" Seite 166)</p>
4	<p>Nur notwendig, wenn Geräte verwendet werden, für die noch keine Gerätebeschreibungs-Dateien importiert wurden:</p> <ol style="list-style-type: none"> <li>1. WorkVisual-Projekt schließen.</li> <li>2. Benötigte Gerätebeschreibungs-Dateien importieren.</li> <li>3. WorkVisual-Projekt wieder öffnen.</li> </ol>
5	<p>Feldbus aufbauen.</p> <p>(&gt;&gt;&gt; 11.2 "Übersicht Feldbusse" Seite 166)</p>
6	<p>Sunrise E/As anlegen und verschalten.</p> <p>(&gt;&gt;&gt; 11.5 "Sunrise E/As anlegen" Seite 167)</p> <p>(&gt;&gt;&gt; 11.6.3 "Sunrise E/As verschalten" Seite 174)</p>
7	<p>E/A-Konfiguration in Sunrise-Projekt exportieren.</p> <p>(&gt;&gt;&gt; 11.7 "E/A-Konfiguration in Sunrise-Projekt exportieren" Seite 174)</p>
8	<p>E/A-Konfiguration auf die Robotersteuerung übertragen (Projekt synchronisieren) und Robotersteuerung neu starten.</p> <p>(&gt;&gt;&gt; 9.4 "Synchronisation von Projekten" Seite 155)</p>



Informationen zur Installation und Verwaltung von Optionspaketen sind in der Dokumentation **WorkVisual** zu finden.



Informationen zum Import von Gerätebeschreibungs-Dateien und allgemein zum Aufbau von Feldbussen sind in der Dokumentation **WorkVisual** zu finden.

## 11.2 Übersicht Feldbusse

Folgende Feldbusse werden von Sunrise unterstützt und können mit WorkVisual konfiguriert werden:

Feldbus	Beschreibung
PROFINET	Feldbus, der auf Ethernet basiert. Der Datenaustausch erfolgt in einem Client-Server-Verhältnis. PROFINET wird auf der Robotersteuerung installiert.
PROFIBUS	Universeller Feldbus, der die Kommunikation von Geräten verschiedener Hersteller ohne besondere Schnittstellenanpassungen ermöglicht. Der Datenaustausch erfolgt in einem Master-Slave-Verhältnis.
EtherCAT	Feldbus, der auf Ethernet basiert und für Echtzeitanforderungen geeignet ist.



Für die Konfiguration eines Feldbusses wird die Dokumentation zu diesem Feldbus benötigt.



Wenn die Robotersteuerung als PROFINET-Master oder -Device verwendet wird, können Hardware-Probleme dazu führen, dass Busteilnehmer nicht erreicht werden können. In diesem Fall wird der Einsatz eines Diagnosewerkzeugs, beispielsweise WorkVisual, Step 7 oder Wireshark empfohlen.



Beim Anschließen zusätzlicher EtherCAT-Geräte an einem Medienflansch mit EtherCAT-Ausgang, z. B. Medien-Flansch IO pneumatisch, ist zu beachten, dass der Umfang der am Bus zur Verfügung stehenden Signale begrenzt ist.  
Ist die Anzahl angeschlossener Teilnehmer zu groß, kann dies zu einer Überlastung des Busses und zu einem Verlust der Kommunikation führen. Der Roboter kann dann unter Umständen nicht mehr bewegt werden.

## 11.3 Neue E/A-Konfiguration anlegen

### Voraussetzung

- Sunrise-Projekt ohne E/A-Konfiguration

### Vorgehensweise

1. Projekt im **Paket-Explorer** markieren.
2. Menüfolge **Datei > Neu > E/A Konfiguration** wählen.

WorkVisual wird gestartet und das zur E/A-Konfiguration gehörige WorkVisual-Projekt geöffnet. Im Sunrise-Projekt wird die Datei **IOConfiguration.wvs** eingefügt, über die das zugehörige WorkVisual-Projekt aufgerufen werden kann.

## 11.4 Bestehende E/A-Konfiguration öffnen

### Voraussetzung

- Sunrise-Projekt mit E/A-Konfiguration

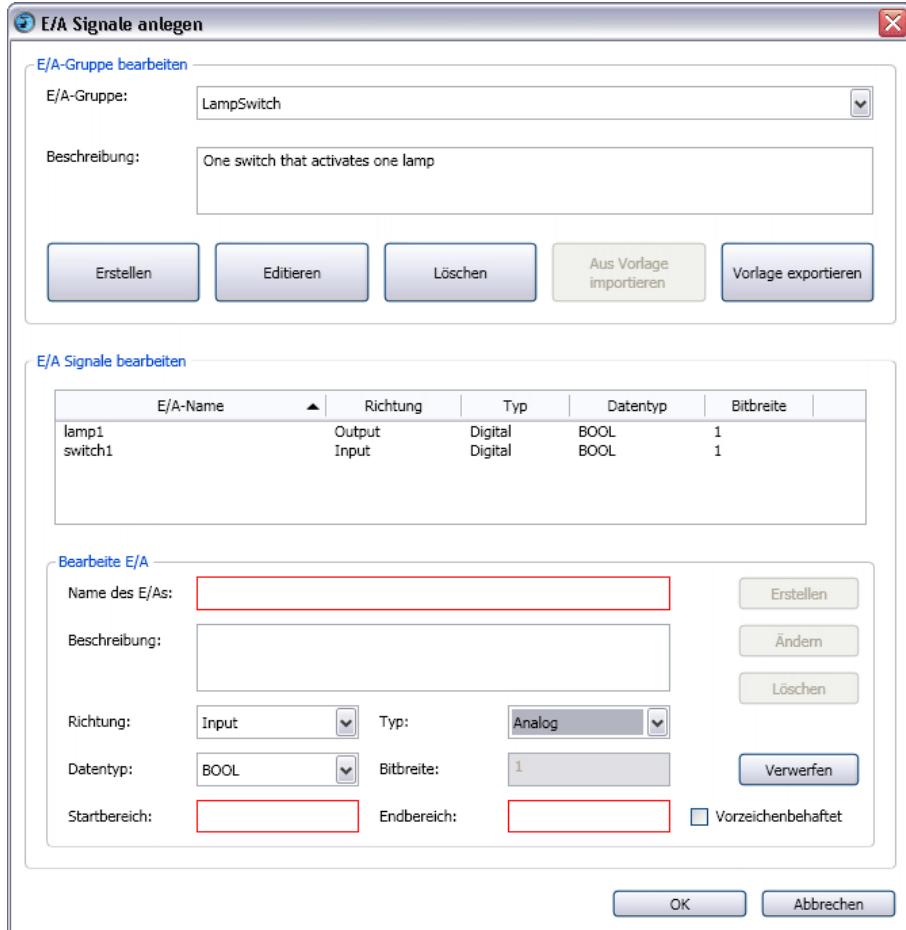
- Vorgehensweise**
1. Auf die Datei **IOConfiguration.wvs** doppelklicken. WorkVisual wird gestartet und das zur E/A-Konfiguration gehörige WorkVisual-Projekt geöffnet.
  2. Im Fenster **Projektstruktur** in der Registerkarte **Geräte** auf die inaktive Robotersteuerung rechtsklicken.
  3. Im Kontextmenü **Aktive Steuerung setzen** wählen. Das Fenster **EA-Verschaltung** öffnet sich. Die Sunrise E/As können bearbeitet werden.

## 11.5 Sunrise E/As anlegen

- Voraussetzung**
- Aufbau des Feldbusses ist abgeschlossen.
  - Robotersteuerung ist aktiv gesetzt.
- Vorgehensweise**
1. Im Fenster **EA-Verschaltung** oben rechts in der Registerkarte **Feldbusse** ein Ein- oder Ausgangsmodul des konfigurierten Busses markieren.  
(>>> 11.6.1 "Fenster EA-Verschaltung" Seite 172)
  2. Im Fenster **EA-Verschaltung** oben links die Registerkarte **Sunrise E/As** auswählen.
  3. Im Fenster **EA-Verschaltung** unten links auf den Button **Erzeuge Signale am Provider** klicken. Das Fenster **E/A Signale anlegen** öffnet sich.  
(>>> 11.5.1 "Fenster "E/A Signale anlegen"" Seite 168)
  4. E/A-Gruppe und Ein-/Ausgänge der Gruppe erstellen.  
(>>> 11.5.2 "E/A-Gruppe und Ein-/Ausgänge der Gruppe erstellen" Seite 169)
  5. Auf **OK** klicken. Die Sunrise E/As werden gespeichert. Das Fenster **E/A Signale anlegen** schließt sich.
- Die erstellte E/A-Gruppe wird im Fenster **EA-Verschaltung** in der Registerkarte **Sunrise E/As** angezeigt. Die Signale können jetzt verschaltet werden.  
(>>> 11.6.3 "Sunrise E/As verschalten" Seite 174)

### 11.5.1 Fenster "E/A Signale anlegen"

#### Übersicht



**Abb. 11-1: Fenster E/A Signale anlegen**

Das Fenster zum Anlegen und Bearbeiten der Sunrise E/As und Sunrise E/A-Gruppen besteht aus folgenden Bereichen:

Bereich	Beschreibung
<b>E/A-Gruppe bearbeiten</b>	In diesem Bereich werden E/A-Gruppen erstellt und bearbeitet. Es besteht die Möglichkeit, E/A-Gruppen als Vorlage zu speichern sowie bereits erstellte Vorlagen zu importieren.
<b>E/A Signale bearbeiten</b>	In diesem Bereich werden die Ein-/Ausgangssignale einer E/A-Gruppe angezeigt.
<b>Bearbeite E/A</b>	In diesem Bereich werden die Ein-/Ausgänge einer E/A-Gruppe erstellt und bearbeitet.

**i** Eingabefelder sind rot umrandet, wenn zwingend Werte eingegeben werden müssen oder wenn falsche Werte eingegeben worden sind. Es wird ein Hilfetext angezeigt, wenn man den Mauszeiger über das Feld bewegt.

#### Signaleigen-schaften

Im Bereich **Bearbeite E/A** können neue Signale angelegt und ihre Eigenschaften definiert werden:

Eigenschaft	Beschreibung
<b>Name des E/As</b>	Name des Ein- oder Ausgangs eingeben.
<b>Beschreibung</b>	Eine Beschreibung für den Ein- oder Ausgang eingeben (optional).

Eigenschaft	Beschreibung
Richtung	Festlegen, ob das Signal ein Ein- oder Ausgang ist. ■ <b>Input, Output</b>
Typ	Festlegen, ob das Signal ein analoges oder digitales Signal ist. ■ <b>Analog, Digital</b>
Datentyp	Datentyp des Signals auswählen.  In WorkVisual stehen insgesamt 15 verschiedene Datentypen zur Auswahl. Diese Datentypen werden für die Verwendung mit Java auf folgende Datentypen abgebildet: ■ integer, long, double, boolean
Bitbreite	Anzahl der Bits eingeben, aus denen das Signal besteht. Beim Datentyp Bool ist die Bitbreite immer 1.  <b>Hinweis:</b> Der Wert muss eine positive Ganzzahl sein, die die maximal erlaubte Länge des ausgewählten Datentyps nicht überschreitet.
Startbereich	Nur relevant für analoge Ein-/Ausgänge!
Endbereich	Start- und Endbereich des analogen Wertes eingeben und bei Bedarf das Häkchen bei <b>Vorzeichenbehaftet</b> setzen.
Vorzeichenbehaftet	 <b>Hinweis:</b> Diese Werte findet man in der Regel im Datenblatt des Feldbus-Moduls. Der Startbereich muss kleiner als der Endbereich sein. Es können auch Dezimalwerte angegeben werden.

### 11.5.2 E/A-Gruppe und Ein-/Ausgänge der Gruppe erstellen

**Voraussetzung** ■ Das Fenster **E/A Signale anlegen** ist geöffnet.

**Vorgehensweise** 1. Im Bereich **E/A-Gruppe bearbeiten** auf **Erstellen** klicken.  
Das Fenster **E/A-Gruppe erstellen** öffnet sich.



Abb. 11-2: E/A-Gruppe erstellen

2. Einen Namen für die E/A-Gruppe vergeben.
3. Eine Beschreibung für die E/A-Gruppe eingeben (optional).



Es wird empfohlen, immer eine Beschreibung einzugeben. Diese Beschreibung wird später als Hilfetext in der Roboter-Applikation und auf der smartHMI angezeigt.

4. Auf **Erstellen** klicken. Die E/A-Gruppe wird erstellt und im Auswahl-Menü **E/A-Gruppe:** angezeigt.
5. Im Bereich **Bearbeite E/A** einen Namen für den Ein- oder Ausgang der Gruppe eingeben und die Signaleigenschaften definieren.  
(>>> "Signaleigenschaften" Seite 168)
6. Im Bereich **Bearbeite E/A** auf **Erstellen** klicken. Das Ein- oder Ausgangssignal wird erstellt und im Bereich **E/A Signale bearbeiten** angezeigt.

7. Schritt 5 und 6 wiederholen, um weitere Ein-/Ausgänge der Gruppe zu definieren.

### 11.5.3 E/A-Gruppe editieren

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.
- Die Ein-/Ausgänge der Gruppe sind nicht verschaltet.

**Vorgehensweise**

1. Im Auswahl-Menü **E/A-Gruppe**: die gewünschte E/A-Gruppe auswählen.
2. Auf **Editieren** klicken. Das Fenster **E/A-Gruppe umbenennen** öffnet sich.
3. Den Namen der E/A-Gruppe und die zugehörige Beschreibung (optional) ändern. Mit **Übernehmen** bestätigen.

### 11.5.4 E/A-Gruppe löschen

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.
- Die Ein-/Ausgänge der Gruppe sind nicht verschaltet.

**Vorgehensweise**

1. Im Auswahl-Menü **E/A-Gruppe**: die gewünschte E/A-Gruppe auswählen.
2. Auf **Löschen** klicken. Wenn für die E/A-Gruppe bereits Signale angelegt wurden, wird eine Sicherheitsabfrage angezeigt.
3. Sicherheitsabfrage mit **Ja** beantworten. Die E/A-Gruppe wird gelöscht.

### 11.5.5 Ein-/Ausgang einer Gruppe ändern

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.
- Die zu ändernden Signale sind nicht verschaltet.

**Vorgehensweise**

1. Im Auswahl-Menü **E/A-Gruppe**: die E/A-Gruppe des Signals auswählen.
2. Im Bereich **E/A Signale bearbeiten** den gewünschten Ein- oder Ausgang markieren.
3. Im Bereich **Bearbeite E/A** die Signaleigenschaften bearbeiten wie benötigt.  
(>>> "Signaleigenschaften" Seite 168)



Mit einem Klick auf die Schaltfläche **Verwerfen** werden alle Änderungen verworfen.

4. Auf **Ändern** klicken. Die Änderungen werden übernommen.

### 11.5.6 Ein-/Ausgang einer Gruppe löschen

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.
- Die zu löschen Signale sind nicht verschaltet.

**Vorgehensweise**

1. Im Auswahl-Menü **E/A-Gruppe**: die E/A-Gruppe des Signals auswählen.
2. Im Bereich **E/A Signale bearbeiten** den gewünschten Ein- oder Ausgang markieren.
3. Auf **Löschen** klicken.

### 11.5.7 E/A-Gruppe als Vorlage exportieren

**Beschreibung**

E/A-Gruppen können als Vorlage gespeichert werden. Die Vorlage enthält alle Ein-/Ausgänge, die zur gespeicherten E/A-Gruppe gehören. Auf diese Weise

können einmal erstellte E/A-Gruppen wiederverwendet werden. Die Verschaltung der Ein- und Ausgänge wird dabei nicht übernommen.

Nach dem Exportieren der Vorlage stehen die in WorkVisual erstellten Vorlagen in Sunrise.Workbench im Ordner **IOTemplates** des Projekts zur Verfügung.

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.

**Vorgehensweise**

1. Im Bereich **E/A-Gruppe bearbeiten** die E/A-Gruppe auswählen, die als Vorlage exportiert werden soll.
2. Auf **Vorlage exportieren** klicken. Das Fenster **E/A-Gruppe als Vorlage exportieren** öffnet sich.
3. Einen Namen für die Vorlage vergeben.



Wenn bereits eine Vorlage mit dem gleichen Namen im Sunrise-Projekt existiert, wird diese beim Export überschrieben.

4. Eine Beschreibung für die Vorlage eingeben (optional).
5. Auf **Exportieren** klicken. Die Vorlage wird gespeichert.

### 11.5.8 E/A-Gruppe aus Vorlage importieren

**Voraussetzung**

- Das Fenster **E/A Signale anlegen** ist geöffnet.
- In Sunrise.Workbench steht mindestens eine E/A-Gruppe als Vorlage im Sunrise-Projekt zur Verfügung.

**Vorgehensweise**

1. Im Bereich **E/A-Gruppe bearbeiten** auf **Aus Vorlage importieren** klicken. Das Fenster **Importiere E/A-Gruppe aus Vorlage** öffnet sich.
2. In der Auswahlliste **Verwendete Vorlage** die zu importierende Vorlage auswählen.
3. Im Feld **Name der E/A-Gruppe** einen Namen für die zu erstellende E/A-Gruppe eingeben.
4. Auf **Importieren** klicken. Eine gemäß der Vorlage konfigurierte E/A-Gruppe wird importiert und kann bearbeitet werden.

## 11.6 Bus verschalten

### 11.6.1 Fenster EA-Verschaltung

#### Übersicht

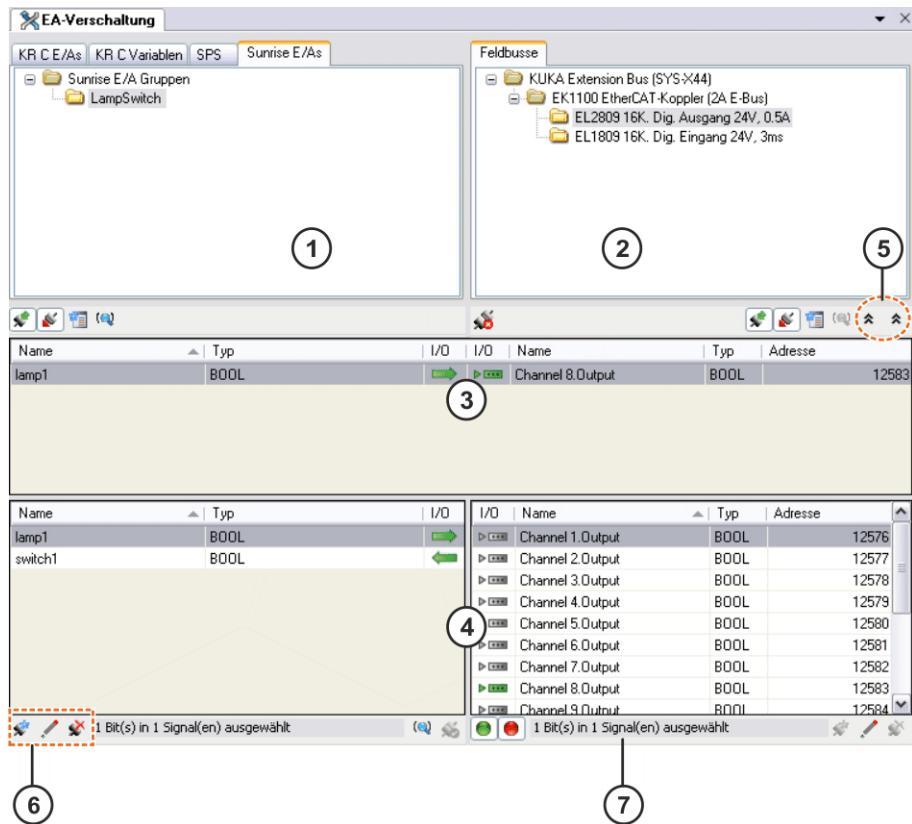


Abb. 11-3: Fenster EA-Verschaltung

Pos.	Beschreibung
1	Anzeige der Sunrise E/A Gruppen Die Signale der hier markierten E/A-Gruppe werden in den unteren Übersichten angezeigt.
2	Anzeige der Ein-/Ausgänge der Busmodule Die Signale des hier markierten Moduls werden in den unteren Übersichten angezeigt.
3	Verbindungsübersicht Anzeige der verschalteten Signale. Dies sind die Signale der unter <b>Sunrise E/As</b> markierten E/A-Gruppe, die mit dem unter <b>Feldbusse</b> markierten Busmodul verschalten sind.
4	Signalübersicht Hier können die Signale verschaltet werden. (>>> 11.6.3 "Sunrise E/As verschalten" Seite 174)

Pos.	Beschreibung
5	Mithilfe der Pfeil-Buttons können Verbindungs- und Signalübersicht getrennt voneinander zusammen- und wieder aufgeklappt werden. <ul style="list-style-type: none"> <li>■ <b>Verbindungsübersicht zusammenklappen</b> (Pfeilsymbol links zeigt nach oben)</li> <li>■ <b>Verbindungsübersicht aufklappen</b> (Pfeilsymbol links zeigt nach unten)</li> <li>■ <b>Signalübersicht zusammenklappen</b> (Pfeilsymbol rechts zeigt nach oben)</li> <li>■ <b>Signalübersicht aufklappen</b> (Pfeilsymbol rechts zeigt nach unten)</li> </ul>
6	Buttons zum Erstellen und Bearbeiten der Sunrise E/As
7	Anzeige, wie viele Bits die markierten Signale enthalten.



Für die E/A-Verschaltung in Sunrise sind nur die Registerkarten **Sunrise E/As** und **Feldbusse** relevant.

### 11.6.2 Buttons im Fenster EA-Verschaltung

Einige dieser Buttons sind mehrmals vorhanden. Sie beziehen sich dann immer auf die Seite des Fensters **EA-Verschaltung**, auf der sie stehen.

#### Bearbeiten

Button	Name / Beschreibung
	<b>Erzeuge Signale am Provider</b> Öffnet das Fenster <b>E/A Signale anlegen</b> . (>>> 11.5.1 "Fenster "E/A Signale anlegen"" Seite 168) Der Button ist nur aktiv, wenn in der Registerkarte <b>Feldbusse</b> ein Ein- oder Ausgangsmodul und in der Signalübersicht ein Signal der E/A-Gruppe markiert ist.
	<b>Editiere Signale am Provider</b> Öffnet das Fenster <b>E/A Signale bearbeiten</b> . Der Button ist nur aktiv, wenn in der Registerkarte <b>Sunrise E/As</b> eine E/A-Gruppe und in der Signalübersicht ein Signal der E/A-Gruppe markiert ist.
	<b>Lösche Signale am Provider</b> Löscht alle markierten Ein-/Ausgänge. Sind alle Ein-/Ausgänge einer Gruppe markiert, wird die E/A-Gruppe ebenfalls gelöscht. Der Button ist nur aktiv, wenn in der Registerkarte <b>Sunrise E/As</b> eine E/A-Gruppe und in der Signalübersicht ein Signal der E/A-Gruppe markiert ist.

**Verschalten**

<b>Button</b>	<b>Name / Beschreibung</b>
	<b>Trennen</b> Trennt markierte verschaltete Signale. Es können mehrere Verbindungen markiert werden und auf einmal getrennt werden.
	<b>Verbinden</b> Verschaltet die Signale miteinander, die in der Signalübersicht markiert sind.

**11.6.3 Sunrise E/As verschalten****Beschreibung**

Mit dieser Vorgehensweise werden die Sunrise E/As mit den Ein-/Ausgängen des Feldbus-Moduls verschaltet. Es können nur Eingänge mit Eingängen und Ausgänge mit Ausgängen des gleichen Datentyps verschaltet werden. Möglich ist beispielsweise BOOL mit BOOL oder INT mit INT, nicht möglich ist BOOL mit INT oder BYTE.

**Voraussetzung**

- Robotersteuerung ist aktiv gesetzt.

**Vorgehensweise**

1. In der linken Hälfte des Fensters in der Registerkarte **Sunrise E/As** die E/A-Gruppe markieren, von der E/As verschaltet werden sollen.  
Die Signale der Gruppe werden im unteren Bereich des Fensters **EA-Verschaltung** angezeigt.
2. In der rechten Hälfte des Fensters in der Registerkarte **Feldbusse** das gewünschte Ein- oder Ausgangsmodul markieren.  
Die Signale des markierten Feldbus-Moduls werden im unteren Bereich des Fensters **EA-Verschaltung** angezeigt.
3. Das Signal der Gruppe per Drag&Drop auf den Ein- oder Ausgang des Moduls ziehen. (Oder auch umgekehrt den Ein- oder Ausgang des Geräts auf das Signal der Gruppe ziehen.)  
Die Signale sind jetzt verschaltet. Verschaltete Signale sind durch grüne Pfeile gekennzeichnet.

**Alternative Vorgehensweise beim Verschalten:**

- Die zu verschaltenden Signale markieren und auf den Button **Verbinden** klicken.

**11.7 E/A-Konfiguration in Sunrise-Projekt exportieren****Beschreibung**

Beim Exportieren einer E/A-Konfiguration aus WorkVisual wird im zugehörigen Sunrise-Projekt für jede E/A-Gruppe eine eigene Java-Klasse erstellt. Jede dieser Java-Klassen enthält die für die Programmierung benötigten Methoden, um lesend auf die Ein-/Ausgänge einer E/A-Gruppe und schreibend auf die Ausgänge einer E/A-Gruppe zuzugreifen.

Die Klassen und Methoden werden im Quellenordner des Sunrise-Projekts im Java-Paket **com.kuka.generated.ioAccess** abgelegt.



Der Quellcode der Java-Klassen des Pakets **com.kuka.generated.ioAccess** darf nicht manuell geändert werden. Um die Funktionalität einer E/A-Gruppe zu erweitern, können von den erzeugten Klassen weitere Klassen abgeleitet oder Objekte dieser Klassen weiterverwendet werden, z. B. als Felder eigener Klassen (Aggregieren).

Die Struktur des Sunrise-Projekts nach dem Exportieren einer E/A-Konfiguration ist hier beschrieben:

(>>> 15.12 "Ein-/Ausgänge" Seite 319)

**Voraussetzung**

- Robotersteuerung ist aktiv gesetzt.
- In Sunrise.Workbench ist die automatische Änderungserkennung aktiviert.  
(>>> 5.10 "Automatische Änderungserkennung aktivieren" Seite 61)

**Vorgehensweise**

1. Menüfolge **Datei > Import / Export** wählen. Der Assistent für den Import/Export von Dateien öffnet sich.
2. **E/A Konfiguration in Sunrise Workbench Projekt exportieren** wählen.
3. Auf **Weiter>** und dann auf **Fertigstellen** klicken. Die Konfiguration wird in das Sunrise-Projekt exportiert.
4. Es wird angezeigt, ob der Export erfolgreich abgeschlossen wurde. Wenn Sunrise E/As nicht verschaltet wurden, wird dies ebenfalls angezeigt.



Es ist nicht zwingend erforderlich, alle angelegten Sunrise E/As zu verschalten.

Den Assistanten mit **Schließen** beenden.

5. WorkVisual über **Datei > Beenden** schließen.



## 12 Externe Steuerung

### 12.1 Externe Steuerung konfigurieren

**Beschreibung** Wenn die Prozesse der Station im Automatikbetrieb von einer übergeordneten Steuerung gesteuert werden sollen, muss das Sunrise-Projekt, das sich auf der Robotersteuerung befindet, für die externe Steuerung konfiguriert sein.

Jedes Sunrise-Projekt, das für die externe Steuerung konfiguriert ist, verwendet eine Default-Applikation, die bei einem Wechsel in die Betriebsart Automatik automatisch angewählt wird. Die Default-Applikation des extern gesteuerten Projekts kann im Automatikbetrieb nicht wieder abgewählt werden.

Die Default-Applikation kann nicht über die Start-Taste am smartPAD gestartet werden, sondern nur über einen externen Eingang.

Robotersteuerung und übergeordnete Steuerung kommunizieren über vordefinierte Ein-/Ausgangssignale:

- Die übergeordnete Steuerung kann die Default-Applikation über die Ein-Ausgangssignale starten, pausieren und fortsetzen.
- Die Ausgangssignale können dazu verwendet werden, Informationen über den Zustand von Default-Applikation und Station an die übergeordnete Steuerung zu liefern.

**Übersicht** Um die externe Steuerung nutzen zu können, sind folgende Schritte erforderlich:

Schritt	Beschreibung
1	Ein-/Ausgänge für die Kommunikation mit der übergeordneten Steuerung in WorkVisual konfigurieren und verschalten. (>>> 12.1.1 "Externe Steuerung Eingänge" Seite 178) (>>> 12.1.2 "Externe Steuerung Ausgänge" Seite 178)
2	E/A-Konfiguration von WorkVisual in Sunrise.Workbench exportieren.
3	Default-Applikation für die externe Steuerung des Sunrise-Projekts in Sunrise.Workbench erstellen.
4	Externe Steuerung des Sunrise-Projekts in Sunrise.Workbench konfigurieren. (>>> 12.1.4 "Externe Steuerung in den Projekt-Eigenschaften konfigurieren" Seite 180)
5	Sunrise-Projekt per Synchronisation auf die Robotersteuerung übertragen.



Die physikalischen Ein-/Ausgänge, die für die Kommunikation mit der übergeordneten Steuerung verwendet werden, dürfen nicht mehrfach verschaltet werden.

**Voraussetzung** Um eine Applikation extern starten zu können, müssen folgende Voraussetzungen erfüllt sein:

- Roboter ist justiert (alle Achsen).
- Sunrise-Projekt ist für die externe Steuerung konfiguriert.
- Betriebsart AUT
- Eingang App\_Start ist konfiguriert.
- Falls konfiguriert: Eingang App\_Enable liefert HIGH-Pegel (TRUE).

- Fahr freigabe ist vorhanden.

### 12.1.1 Externe Steuerung Eingänge

<b>App_Start</b>	Der Eingang App_Start ist für ein extern gesteuertes Projekt zwingend erforderlich.  Über eine steigende Flanke des Eingangssignals (Wechsel von FALSE auf TRUE) wird die Default-Applikation im Automatikbetrieb von der übergeordneten Steuerung gestartet und fortgesetzt.
<b>App_Enable</b>	Der Eingang App_Enable ist optional konfigurierbar.  Über einem LOW-Pegel an diesem Eingang kann die Default-Applikation im Automatikbetrieb von der übergeordneten Steuerung pausiert werden.
<b>Systemverhalten</b>	Der Eingang App_Enable besitzt eine höhere Priorität als der Eingang App_Start. Ist der Eingang App_Enable konfiguriert, kann die Default-Applikation nur gestartet werden, wenn an App_Enable ein High-Pegel anliegt.

<b>App_Start</b>	<b>App_Enable</b>	<b>Zustand Applikation</b>	<b>Reaktion</b>
FALSE --> TRUE	FALSE	<b>Angewählt</b>	Keine
FALSE --> TRUE	FALSE	<b>Bewegung pausiert</b>	Keine
FALSE --> TRUE	TRUE	<b>Angewählt</b>	Applikation wird gestartet.
FALSE --> TRUE	TRUE	<b>Bewegung pausiert</b>	Applikation wird fortgesetzt. Bei Verlassen der Bahn: Roboter wird rückpositioniert. Applikation danach pausiert.
Beliebig	TRUE --> FALSE	<b>Ausführung</b>	Applikation wird pausiert.
Beliebig	TRUE --> FALSE	<b>Rückpositionierung</b>	Applikation wird pausiert.

### 12.1.2 Externe Steuerung Ausgänge

Die Konfiguration dieser Ausgänge ist optional.

<b>AutExt_Active</b>	Ein High-Pegel an diesem Ausgang meldet der übergeordneten Steuerung, dass die Betriebsart Automatik aktiv ist und das Projekt auf der Robotersteuerung extern gesteuert werden kann.
<b>AutExt_AppRead</b> <b>yToStart</b>	Ein High-Pegel an diesem Ausgang meldet der übergeordneten Steuerung, dass die Default-Applikation startbereit ist (Zustand <b>Angewählt</b> oder <b>Bewegung pausiert</b> ).
<b>DefaultApp_Error</b>	Ein High-Pegel an diesem Ausgang meldet der übergeordneten Steuerung, dass beim Ausführen der Default-Applikation ein Fehler aufgetreten ist (Zustand <b>Fehler</b> ).
<b>Station_Error</b>	Ein High-Pegel an diesem Ausgang meldet der übergeordneten Steuerung, dass sich die Station im Fehlerzustand befindet. Der Fehlerzustand liegt vor, wenn eine der folgenden Bedingungen zutrifft: <ul style="list-style-type: none"> <li>■ Fahr freigabe nicht vorhanden.</li> <li>■ Antriebsfehler oder Busfehler liegt vor.</li> </ul>

- Mindestens eine Roboterachse ist nicht justiert und die Betriebsart ist nicht T1.

**HINWEIS** Es ist unzulässig, Ausgänge, die Systemzustände an die übergeordnete Steuerung melden, in einer Roboter-Applikation zu setzen. Wenn dies nicht beachtet wird, kann es zu einem Fehlverhalten der übergeordneten Steuerung und Sachschaden kommen.

### 12.1.3 Signallaufpläne

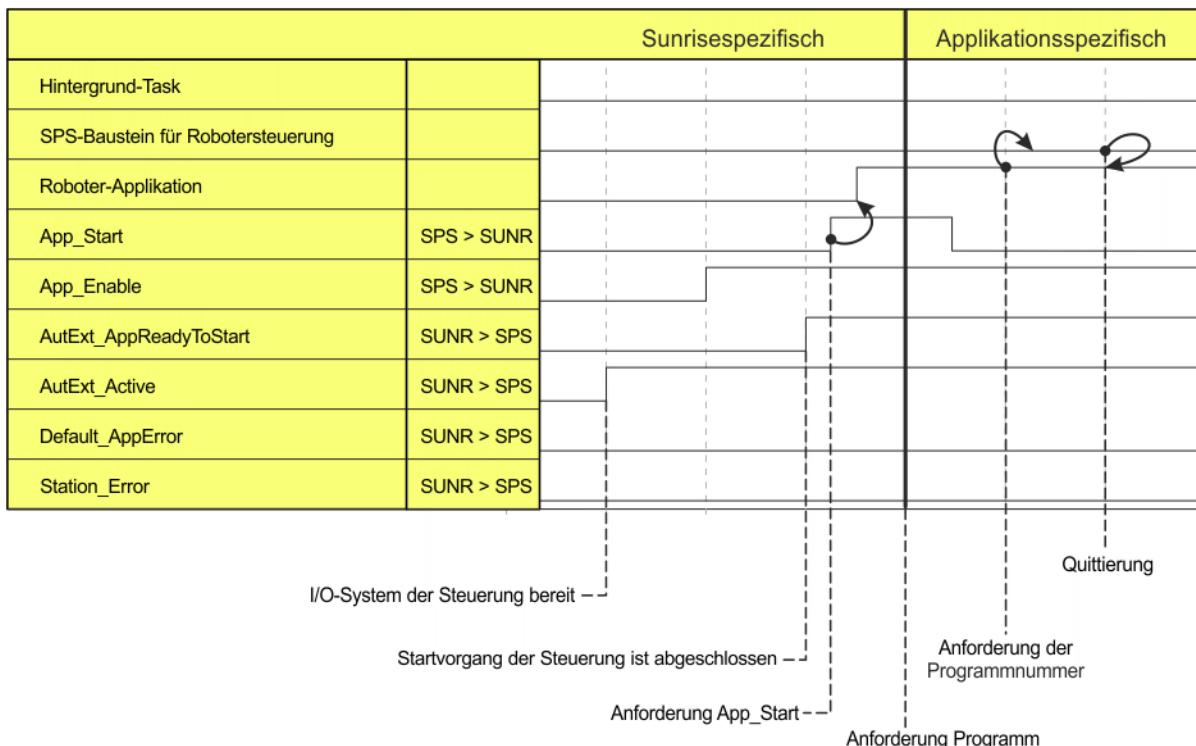


Abb. 12-1: Automatischer Anlagenanlauf und Normalbetrieb

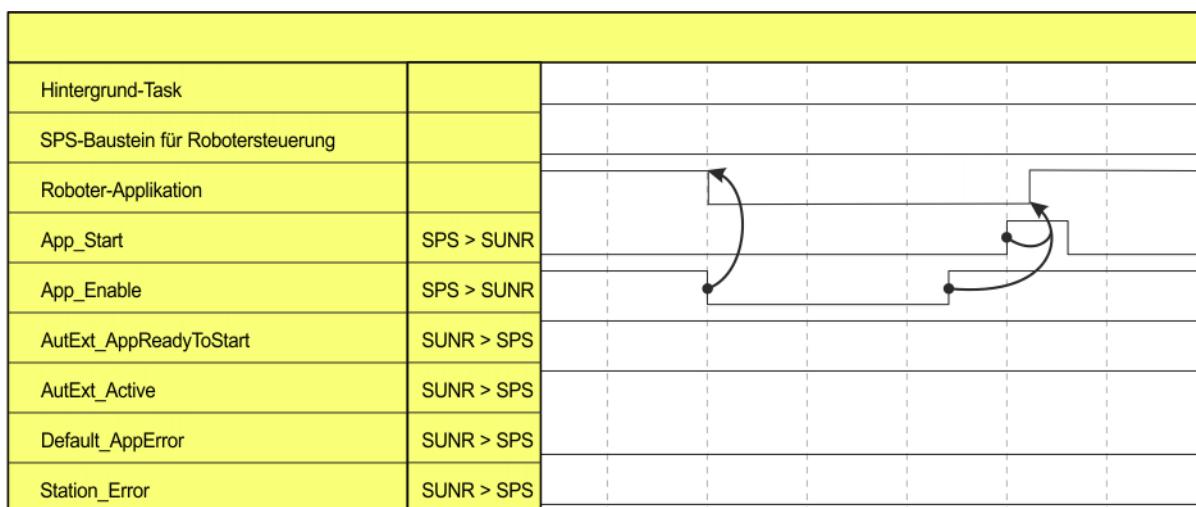


Abb. 12-2: Wiederauflauf nach Anwenderhalt

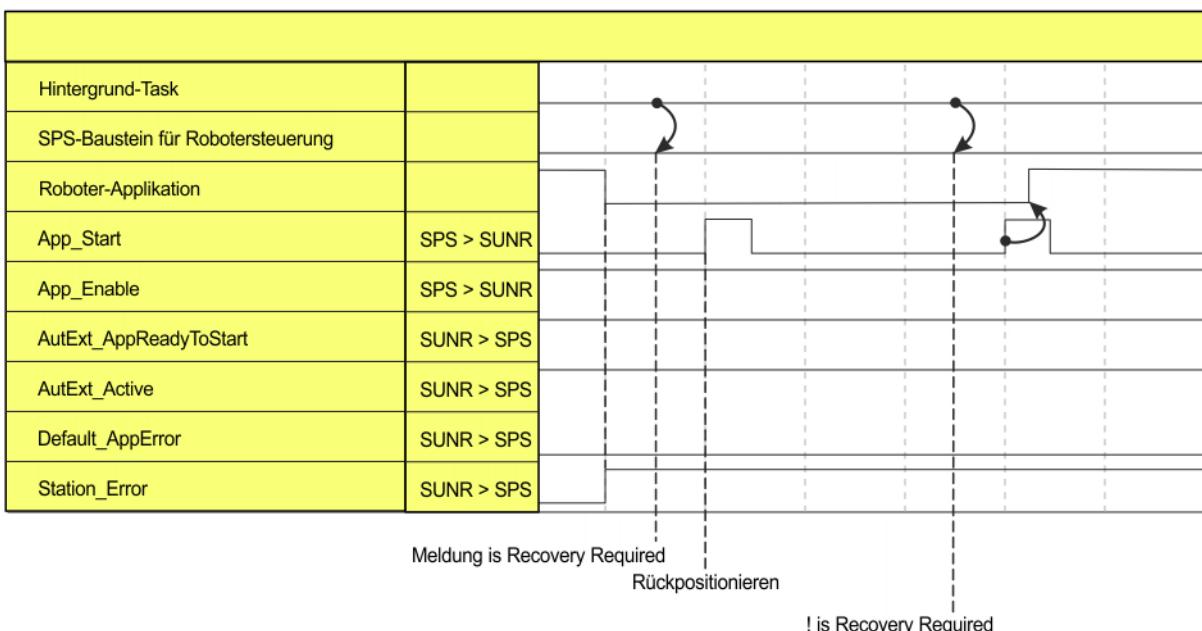


Abb. 12-3: Wiederanlauf nach externem NOT-HALT

#### 12.1.4 Externe Steuerung in den Projekt-Eigenschaften konfigurieren

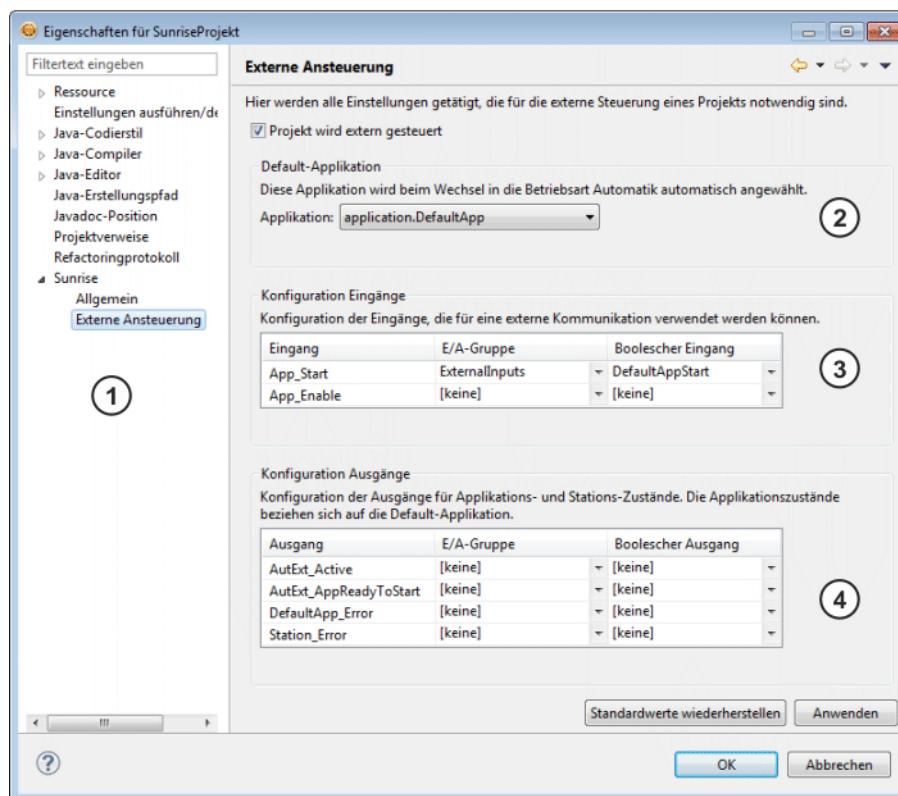
##### Vorgehensweise

1. Im **Paket-Explorer** auf das gewünschte Sunrise-Projekt rechtsklicken und im Kontextmenü **Sunrise > Projekteinstellungen anpassen** wählen.  
Das Fenster **Eigenschaften für Projekt** öffnet sich.
2. In der Verzeichnisstruktur im linken Bereich des Fensters **Sunrise > Externe Ansteuerung** wählen.
3. Im rechten Bereich des Fensters die Einstellungen für die externe Steuerung des Projekts vornehmen.
  - Bei **Projekt wird extern gesteuert** Häkchen setzen.
  - Im Bereich **Default-Applikation** die gewünschte Default-Applikation auswählen.
  - Im Bereich **Konfiguration Eingänge** die Eingänge für die externe Kommunikation konfigurieren.
    - Spalte **E/A-Gruppe**: E/A-Gruppe auswählen, die den gewünschten Eingang enthält.
    - Spalte **Boolescher Eingang**: Eingang der E/A-Gruppe auswählen.
  - Optional: Im Bereich **Konfiguration Ausgänge** die Meldeausgänge für das Projekt konfigurieren.
    - Spalte **E/A-Gruppe**: E/A-Gruppe auswählen, die den gewünschten Ausgang enthält.
    - Spalte **Boolescher Ausgang**: Ausgang der E/A-Gruppe auswählen.
4. Auf **Anwenden** klicken, um die Einstellungen zu übernehmen und Fenster mit **OK** schließen.



Die Default-Applikation ist im **Paket-Explorer** mit einem Symbol gekennzeichnet:  
Wird die Default-Applikation umbenannt, wird das Symbol nicht mehr angezeigt, und sie muss erneut als Default-Applikation markiert werden.  
(>>> 12.2 "Roboter-Applikation als Default-Applikation markieren" Seite 181)

## Beschreibung



**Abb. 12-4: Minimal-Konfiguration: Externe Steuerung eines Projekts**

Pos.	Beschreibung
1	Verzeichnisstruktur Projekt-Eigenschaften
2	Bereich <b>Default-Applikation</b> Es stehen alle Roboter-Applikationen des Projekts zur Auswahl.
3	Bereich <b>Konfiguration Eingänge</b> Es stehen alle E/A-Gruppen der E/A-Konfiguration des Projekts und die zugehörigen Eingänge zur Auswahl. Der Eingang App_Start ist für die externe Steuerung eines Projekts zwingend erforderlich.
4	Bereich <b>Konfiguration Ausgänge</b> Es stehen alle E/A-Gruppen der E/A-Konfiguration des Projekts und die zugehörigen Ausgänge zur Auswahl.

## 12.2 Roboter-Applikation als Default-Applikation markieren

### Beschreibung

Für jedes Sunrise-Projekt kann eine Default-Applikation festgelegt werden, die bei einem Neustart der Robotersteuerung und bei einer Synchronisation des Projekts automatisch angewählt wird.

Bei einem extern gesteuerten Projekt ist es zwingend erforderlich, eine Default-Applikation festzulegen. Diese wird bei einem Wechsel in die Betriebsart Automatik automatisch angewählt.

### Vorgehensweise

- Im **Paket-Explorer** auf die gewünschte Roboter-Applikation rechtsklicken und im Kontextmenü **Sunrise > Als Default-Applikation markieren** wählen.
- Die Roboter-Applikation wird im **Paket-Explorer** als Default-Applikation gekennzeichnet und in den Projekt-Eigenschaften automatisch als Default-Applikation eingestellt.

**Beispiel****Abb. 12-5: Default-Applikation MainApp.java****12.3 Meldeausgänge für nicht extern gesteuertes Projekt festlegen****Beschreibung**

Die für die Kommunikation mit der übergeordneten Steuerung vordefinierten Ausgangssignale können auch in nicht extern gesteuerten Projekten verwendet werden, um Applikations- und Stationszustände zu melden.

Die Applikationszustände beziehen sich immer auf die Default-Applikation des Projekts.

**Voraussetzung**

- Die E/A-Konfiguration des Projekts enthält die in WorkVisual konfigurierten und verschalteten Ausgänge.  
(>>> 12.1.2 "Externe Steuerung Ausgänge" Seite 178)

**Vorgehensweise**

1. Im **Paket-Explorer** auf das gewünschte Sunrise-Projekt rechtsklicken und im Kontextmenü **Sunrise > Projekteinstellungen anpassen** wählen.  
Das Fenster **Eigenschaften für Projekt** öffnet sich.
2. In der Verzeichnisstruktur im linken Bereich des Fensters **Sunrise > Allgemein** wählen.
3. Im rechten Bereich des Fensters die allgemeinen Einstellungen für das Projekt vornehmen.
  - Wenn Applikationszustände gemeldet werden sollen: Im Bereich **Default-Applikation** die gewünschte Default-Applikation auswählen.
  - Im Bereich **Konfiguration Ausgänge** die Meldeausgänge für das Projekt konfigurieren.
    - Spalte **E/A-Gruppe**: E/A-Gruppe auswählen, die den gewünschten Ausgang enthält.
    - Spalte **Boolescher Ausgang**: Ausgang der E/A-Gruppe auswählen.
4. Auf **Anwenden** klicken, um die Einstellungen zu übernehmen und Fenster mit **OK** schließen.

## 13 Sicherheitskonfiguration

### 13.1 Übersicht Sicherheitskonfiguration

Mit der Sicherheitskonfiguration werden die sicherheitsgerichteten Funktionen festgelegt, um den Industrieroboter sicherheitsgerecht in die Anlage zu integrieren. Sicherheitsgerichtete Funktionen dienen dazu, den Menschen bei der Arbeit mit dem Roboter zu schützen.

Die Sicherheitskonfiguration ist fester Bestandteil eines Sunrise-Projekts und wird in Tabellenform verwaltet. Die einzelnen Sicherheitsfunktionen werden in KUKA Sunrise.Workbench anwendungsspezifisch zusammengestellt. Die Sicherheitskonfiguration wird anschließend mit dem Projekt auf die Steuerung übertragen und dort aktiviert.



**WARNUNG** Durch eine falsche Sicherheitskonfiguration können schwerer Sachschaden und Verletzungen mit Todesfolge entstehen. Wenn eine neue oder geänderte Sicherheitskonfiguration aktiviert wird, muss der Sicherheitsinbetriebnehmer durch Tests überprüfen, dass die konfigurierten Sicherheitsparameter korrekt übernommen wurden und die Sicherheitsfunktionen der Konfiguration voll funktionsfähig sind (Sicherheitsabnahme).



Die Konfiguration der Sicherheitsfunktionen, die Aktivierung und Deaktivierung der Sicherheitskonfiguration sowie die Sicherheitsabnahme darf nur von einem geschulten Sicherheitsinbetriebnehmer durchgeführt werden. Der Sicherheitsinbetriebnehmer trägt die Verantwortung, dass die Sicherheitskonfiguration nur auf Robotern aktiviert wird, für die sie vorgesehen ist.

Die Sicherheitskonfiguration wird von KUKA Sunrise.Workbench zur Zeit nicht auf Plausibilität geprüft.



Bei einer unvollständigen Inbetriebnahme der Anlage sind zusätzliche risikomindernde Ersatzmaßnahmen zu ergreifen und zu dokumentieren, z. B. Anbringen eines Schutzauns oder Warnschildes, Verriegelung des Hauptschalters etc. Eine unvollständige Inbetriebnahme liegt beispielsweise vor, wenn noch nicht alle notwendigen Sicherheitsüberwachungen implementiert wurden oder die Sicherheitsfunktionen noch nicht auf ihre sichere Funktion getestet wurden.



Durch den Anlagenintegrator ist zu verifizieren, dass die Sicherheitskonfiguration Risiken beim kollaborierenden Betrieb (MRK) ausreichend reduziert. Es wird empfohlen, diese Verifikation konform zu den Informationen und Hinweisen für den Betrieb von kollaborierenden Robotern nach ISO/TS 15066 durchzuführen.



Im Rahmen des ESM-Mechanismus (Event-Driven Safety Monitoring) werden in der Sicherheitskonfiguration Zustände mit unterschiedlichen Sicherheitseinstellungen definiert, zwischen denen in der Applikation umgeschaltet werden kann. Da die Umschaltung zwischen diesen Zuständen durch nicht sicherheitsgerichtete Signale erfolgt, müssen alle konfigurierten Zustände konsistent sein. Das heißt, jeder Zustand muss unabhängig vom Zeitpunkt und Ort seiner Aktivierung (also unabhängig vom aktuellen Prozessschritt) ein ausreichendes Maß an Sicherheit herstellen.

## 13.2 Sicherheitskonzept

### Übersicht

Die Sicherheitskonfiguration muss alle Sicherheitsfunktionen realisieren, die zum Betrieb des Industrieroboters erforderlich sind. Eine Sicherheitsfunktion überwacht das gesamte System anhand bestimmter Kriterien. Diese werden durch einzelne Überwachungsfunktionen, sog. AMFs (Atomic Monitoring Functions) beschrieben. Für die Konfiguration einer Sicherheitsfunktion können mehrere AMFs zu komplexen Sicherheitsüberwachungen verknüpft werden. Die Sicherheitsfunktion legt außerdem eine geeignete Reaktion fest, die im Fehlerfall ausgelöst wird.

Beispiel: In einem bestimmten Bereich im Arbeitsraum des Roboters darf die Geschwindigkeit am TCP 500 mm/s nicht übersteigen (Überwachungsfunktionen Raumüberwachung und Geschwindigkeitsüberwachung). Andernfalls muss der Roboter sofort anhalten (Reaktion im Fehlerfall).

### PSM und ESM

Das Sunrise Sicherheitskonzept bietet 2 unterschiedliche Überwachungsmechanismen:

■ Permanente sicherheitsgerichtete Überwachung

Die Sicherheitsfunktionen des PSM-Mechanismus (Permanent Safety Monitoring) sind ständig aktiv. Eine Deaktivierung einzelner Sicherheitsfunktionen ist nur durch eine Änderung der Sicherheitskonfiguration möglich.

Der PSM-Mechanismus dient zur ständigen Überwachung des Systems. Hier werden grundlegende Sicherheitseinstellungen realisiert, die unabhängig vom ausgeführten Prozessschritt sind. Dazu gehören beispielsweise NOT-HALT-Funktionen, die Zustimmung am smartPAD, die Festlegung eines Zellenbereichs oder betriebsartenabhängige Sicherheitsfunktionen.

■ Ereignisabhängige sicherheitsgerichtete Überwachung

Der ESM-Mechanismus (Event-driven Safety Monitoring) definiert sichere Zustände, zwischen denen in der Applikation umgeschaltet werden kann. Ein sicherer ESM-Zustand enthält die im jeweiligen Prozessschritt benötigten Sicherheitsfunktionen.

Da die Umschaltung durch nicht sicherheitsgerichtete Signale erfolgt, ist bei der Festlegung des Zustands zu beachten, dass dieser unabhängig von Ort und Zeitpunkt seiner Aktivierung immer ein ausreichendes Maß an Sicherheit gewährleisten muss.

Der ESM-Mechanismus ermöglicht eine prozessabhängige Anpassung bestimmter Sicherheitseinstellungen. Dies ist besonders für Anwendungen im Bereich Mensch-Roboter-Kooperation interessant, da hier häufig situationsabhängig unterschiedliche Sicherheitseinstellungen benötigt werden. Über einen ESM-Zustand können für jeden Prozessschritt die nötigen Parameter individuell festgelegt werden, beispielsweise die zulässige Geschwindigkeit, Kollisionswerte oder räumliche Grenzen.

### AMF

Die kleinste Einheit einer Sicherheitsüberwachung wird als Atomic Monitoring Function (AMF) bezeichnet.

Jede AMF liefert eine elementare sicherheitsrelevante Information, beispielsweise ob ein sicherer Eingang gesetzt oder ob die Betriebsart Automatik gewählt ist.

Atomic Monitoring Functions können 2 verschiedene Zustände annehmen und verhalten sich LOW-aktiv. D. h. wird eine Überwachung verletzt, wechselt der Zustand von "1" auf "0".

- Zustand "0": Die AMF ist verletzt.
- Zustand "1": Die AMF ist nicht verletzt.

Die AMF **NOT-HALT smartPAD** ist beispielsweise verletzt, wenn die NOT-HALT-Einrichtung am Bediengerät gedrückt ist.

## Sicherheitsfunktion

Ein sicherer ESM-Zustand wird aus bis zu 20 Sicherheitsfunktionen definiert. Die Sicherheitsfunktionen des ESM-Mechanismus verwenden jeweils genau eine AMF. Ist diese AMF verletzt, gilt die Sicherheitsfunktion und damit der gesamte ESM-Zustand als verletzt.

Für Sicherheitsfunktionen des PSM-Mechanismus werden bis zu 3 AMFs logisch miteinander verknüpft. So können komplexe Sicherheitsüberwachungen realisiert werden. Sind alle AMFs einer Sicherheitsfunktion des PSM-Mechanismus verletzt, gilt die gesamte Sicherheitsfunktion als verletzt.

## Sicherheitsschnittstellen

Es stehen verschiedene Sicherheitsschnittstellen zur Verfügung, um sicherheitsgerichtete Signale zwischen übergeordneter Steuerung und Robotersteuerung auszutauschen. Über die sicheren Eingänge dieser Schnittstellen können Sicherheitseinrichtungen, beispielsweise externe NOT-HALT-Geräte oder Schutztüren, angeschlossen und die zugehörigen Eingangssignale ausgewertet werden. Die sicheren Ausgänge dieser Schnittstellen können verwendet werden, um eine Verletzung von Sicherheitsfunktionen zu melden.

- Ethernet-Sicherheitsschnittstellen (nur Slave-Funktion verfügbar)
  - PROFINET/PROFIsafe
  - EtherCAT/FSoE
- Diskrete Sicherheitsschnittstellen
  - CIB\_SR/X11



Der PROFINET-Bus ist in WorkVisual konfigurierbar. Weitere Informationen zum konkreten Aufbau des Feldbusses sind in der zugehörigen Feldbus-Dokumentation zu finden.



Weitere Informationen zur Schnittstelle X11 sind in der Betriebsanleitung der Robotersteuerung **KUKA Sunrise Cabinet** zu finden.

## Reaktionen

Für jede Sicherheitsfunktion wird eine geeignete Reaktion festgelegt, die im Fehlerfall erfolgen und das System in einen sicheren Zustand versetzen muss.

Folgende Reaktionen sind konfigurierbar:

- Sicherheitshalt 0 wird ausgelöst.



Es wird empfohlen, einen Sicherheitshalt 0 nur dann zu konfigurieren, wenn ein sofortiges Abschalten der Antriebe und Einfallen der Bremsen als Reaktion benötigt wird.

- Sicherheitshalt 1 wird ausgelöst.
- Sicherheitshalt 1 (bahntreu) wird ausgelöst.

Dies ist die empfohlene Stopp-Reaktion. Sie bietet die geringste Auswirkung auf den Prozess, da eine Applikation fortgesetzt werden kann, ohne dass der Roboter rückpositioniert werden muss.



**VORSICHT** Im Fall von Quetschsituationen können Sicherheitshalt 1 und Sicherheitshalt 1 (bahntreu) aufgrund des gegebenen Anhaltens auf einem geplanten Bremspfad zu höheren Quetschkräften führen. Daher wird empfohlen, bei Sicherheitsüberwachungen zur Erkennung von Quetschsituationen (z. B. mittels der AMF *Kollisionserkennung*, *TCP-Kraftüberwachung*) den Sicherheitshalt 0 zu verwenden.

- Sicherer Ausgang wird auf "0" (LOW-Pegel) gesetzt.



Das Setzen eines sicheren Ausgangs ist als Reaktion nur für Sicherheitsfunktionen des PSM-Mechanismus konfigurierbar. Für Sicherheitsfunktionen des ESM-Mechanismus kann sie nicht konfiguriert werden.

Die Reaktionen können für beliebig viele Sicherheitsfunktionen verwendet werden. Eine Reaktion wird ausgelöst, sobald eine der Sicherheitsfunktionen, die diese Reaktion verwenden, verletzt ist. Damit ist es z. B. möglich, über einen sicheren Ausgang eine übergeordnete Steuerung bei Auftreten bestimmter Fehler zu informieren.

Der PSM-Mechanismus ermöglicht es, bei Verletzung einer bestimmten Kombination von AMFs mehrere verschiedene Reaktionen auszulösen. So kann z. B. ein Sicherheitshalt ausgelöst und zusätzlich ein sicherer Ausgang gesetzt werden. Um dies zu realisieren, müssen 2 Sicherheitsfunktionen mit identischer AMF-Kombination konfiguriert werden.

Unterscheiden sich 2 Sicherheitsfunktionen nur in der Art des konfigurierten Stopps, wird bei Verletzung die schärfere Stopp-Reaktion ausgelöst. D. h. es wird diejenige Stopp-Reaktion ausgelöst, die zu einem früheren sicherheitsgerichteten Abschalten der Antriebe führt. Verwenden mehrere Sicherheitsfunktionen das gleiche Ausgangssignal als Reaktion, wird dieses auf "0" gesetzt, sobald eine der Sicherheitsfunktionen verletzt ist.

#### Zeitverhalten

Alle sicherheitsgerichteten Ausgänge verwenden LOW als sicheren Zustand.

Ist eine Sicherheitsfunktion verletzt, die einen sicheren Ausgang als Reaktion verwendet, wird dieser Ausgang sofort auf LOW gesetzt.

Wird der Verletzungszustand aufgehoben, wird der Ausgang erst dann wieder auf HIGH gesetzt, wenn folgende Bedingungen erfüllt sind:

- Die Sicherheitsfunktion ist mindestens 24 ms lang nicht verletzt. Es wird immer zeitverzögert auf die Aufhebung des Verletzungszustandes reagiert.
- Wenn eine Ethernet-Sicherheitsschnittstelle verwendet wird:  
Der Ausgang hat zuvor mindestens 500 ms lang LOW-Pegel. Liegt der LOW-Pegel noch nicht so lange an, wird mit dem Pegelwechsel auf HIGH gewartet bis die 500 ms erreicht sind.
- Wenn die diskrete Sicherheitsschnittstelle verwendet wird:  
Der Ausgang hat zuvor mindestens 200 ms lang LOW-Pegel. Liegt der LOW-Pegel noch nicht so lange an, wird mit dem Pegelwechsel auf HIGH gewartet bis die 200 ms erreicht sind.



Bei Verwendung von Sicherheitsfunktionen mit einem sicheren Ausgang als Reaktion muss beachtet werden, dass Verbindungsfehler (d. h. Kommunikationsfehler) an sicheren Ein- oder Ausgängen von der Sicherheitsteuerung automatisch quittiert werden, wenn die Verbindung wiederhergestellt ist. Entsprechend kann der Pegel des sicheren Ausgangs nach Wiederherstellung der Verbindung von LOW nach HIGH wechseln. Aus diesem Grund muss der Sicherheitsinbetriebnehmer sicherstellen, dass es zu keinem automatischen Wiederanlauf von Peripheriegeräten kommen kann.

### 13.3 Permanent Safety Monitoring

Die Sicherheitsfunktionen des PSM-Mechanismus (Permanent Safety Monitoring) sind permanent aktiv und sorgen für eine ständige Überwachung des gesamten Systems anhand der von diesen Funktionen festgelegten Kriterien.

Für eine Sicherheitsfunktion des PSM-Mechanismus können bis zu 3 AMFs (Atomic Monitoring Functions) miteinander verknüpft werden. Die gesamte Si-

cherheitsfunktion gilt erst dann als verletzt, wenn alle verwendeten AMFs verletzt sind. Die Sicherheitsfunktion legt außerdem eine Reaktion fest. Diese wird ausgelöst, wenn die gesamte Sicherheitsfunktion verletzt ist.

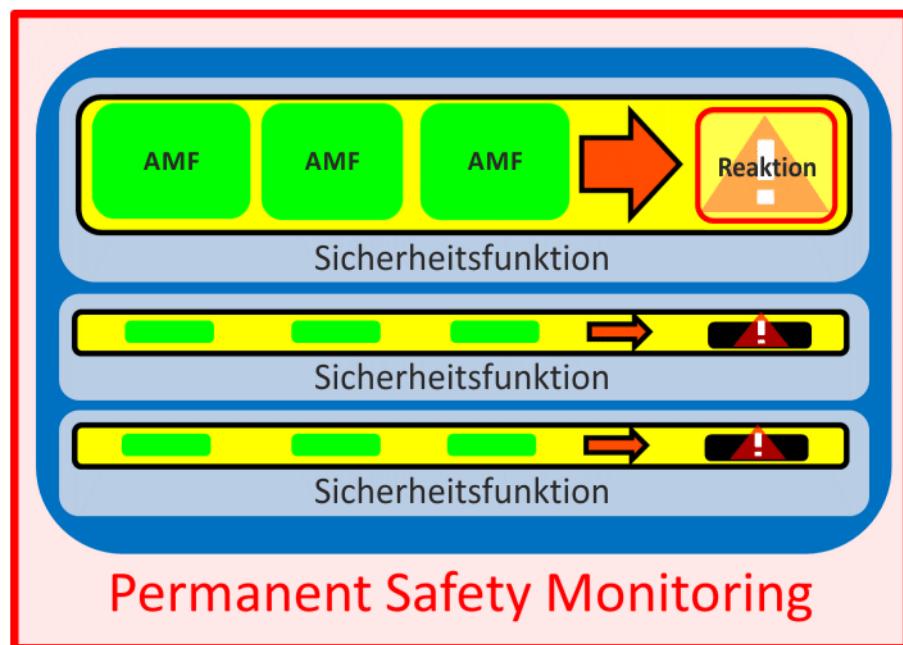


Abb. 13-1: Sicherheitsfunktionen des PSM-Mechanismus

#### Kategorien

Für die Diagnose im Fehlerfall wird jeder Sicherheitsfunktion des PSM-Mechanismus eine Kategorie zugeordnet. Abhängig von der Kategorie werden Fehler am smartPAD angezeigt und in der LOG-Datei gespeichert. Daher empfiehlt es sich diese sinnvoll zu wählen.

Folgende Kategorien stehen zur Verfügung:

Kategorie	Empfohlene Verwendung
<b>Keine</b>	Für Sicherheitsfunktionen, die keiner konkreten Kategorie zugeordnet werden können
<b>Ausgang</b>	Für Sicherheitsfunktionen, die das Setzen eines sicheren Ausgangs als Reaktion verwenden  Bei dieser Kategorie wird im Verletzungsfall keine Diagnoseinformation abgesetzt.
<b>Zustimmung</b>	Für Sicherheitsfunktionen, die einen Zustimmungsschalter auswerten  Bei dieser Kategorie wird im Verletzungsfall keine Diagnoseinformation abgesetzt, da es sich im Fall der Zustimmung um einen normalen Betriebszustand und nicht um einen Fehlerzustand handelt.
<b>NOT-HALT lokal</b>	Für Sicherheitsfunktionen, die einen NOT-HALT auswerten, der von der NOT-HALT-Einrichtung am smartPAD ausgelöst wird
<b>NOT-HALT extern</b>	Für Sicherheitsfunktionen, die einen NOT-HALT auswerten, der von einer externen NOT-HALT-Einrichtung ausgelöst wird
<b>Bedienerenschutz</b>	Für Sicherheitsfunktionen, die das Signal für den Bedienerenschutz auswerten
<b>Sicherer Betriebshalt</b>	Für Sicherheitsfunktionen, die den Stillstand des Roboters überwachen
<b>Kollisionserkennung</b>	Für Sicherheitsfunktionen, die zur Kollisionserkennung oder Kraftüberwachung verwendet werden

Kategorie	Empfohlene Verwendung
<b>Sicherheitshalt</b>	Für Sicherheitsfunktionen, die einen Sicherheitshalt als Reaktion verwenden und keiner anderen Kategorie zugeordnet werden können. Beispiel: externer Sicherheitshalt
<b>Geschwindigkeitsüberwachung</b>	Für Sicherheitsfunktionen, die zur Überwachung einer achsspezifischen oder kartesischen Geschwindigkeit verwendet werden
<b>Raumüberwachung</b>	Für Sicherheitsfunktionen, die zur Überwachung eines achsspezifischen oder kartesischen Raums verwendet werden

### 13.4 Event-driven Safety Monitoring

Der ESM-Mechanismus (Event-driven Safety Monitoring) ermöglicht es, situationsabhängig zwischen unterschiedlichen sicheren ESM-Zuständen umzuschalten.

Insgesamt können bis zu 10 sichere Zustände definiert werden. Die Umschaltung zwischen den Zuständen erfolgt über die Roboter-Applikation oder über einen Hintergrund-Task.

(>>> 13.6.4.8 "Umschalten zwischen ESM-Zuständen" Seite 204)

Ein sicherer ESM-Zustand wird mit bis zu 20 Sicherheitsfunktionen definiert, die in jeder Situation ein ausreichendes Maß an Sicherheit gewährleistet müssen. Durch die Umschaltung in einen ESM-Zustand im Programm wird dieser ESM-Zustand aktiv. Solange der ESM-Zustand aktiv ist, werden alle zugehörigen Sicherheitsfunktionen zusätzlich zu den permanent aktiven Sicherheitsfunktionen überwacht.

Die Verwendung ESM-Mechanismus ist optional. Der ESM-Mechanismus ist deaktiviert, wenn kein ESM-Zustand in der Sicherheitskonfiguration definiert ist.

Bei Verwendung des ESM-Mechanismus ist immer genau ein sicherer Zustand aktiv. Ein Abschalten über die Applikation ist nicht möglich.

Die Sicherheitsfunktionen eines ESM-Zustands enthalten jeweils eine einzelne AMF, der eine geeignete Stopp-Reaktion zugeordnet wird.

Sobald eine Sicherheitsfunktion des aktiven ESM-Zustands verletzt ist, wird ein Stopp ausgelöst. Die Art der Stopp-Reaktion richtet sich nach der schärfsten Stopp-Reaktion aller verletzten Sicherheitsfunktionen in allen ESM-Zuständen (sowohl dem aktiven als auch den inaktiven). D. h. es wird diejenige Stopp-Reaktion ausgelöst, die zu dem frühesten sicherheitsgerichteten Abschalten der Antriebe führt.

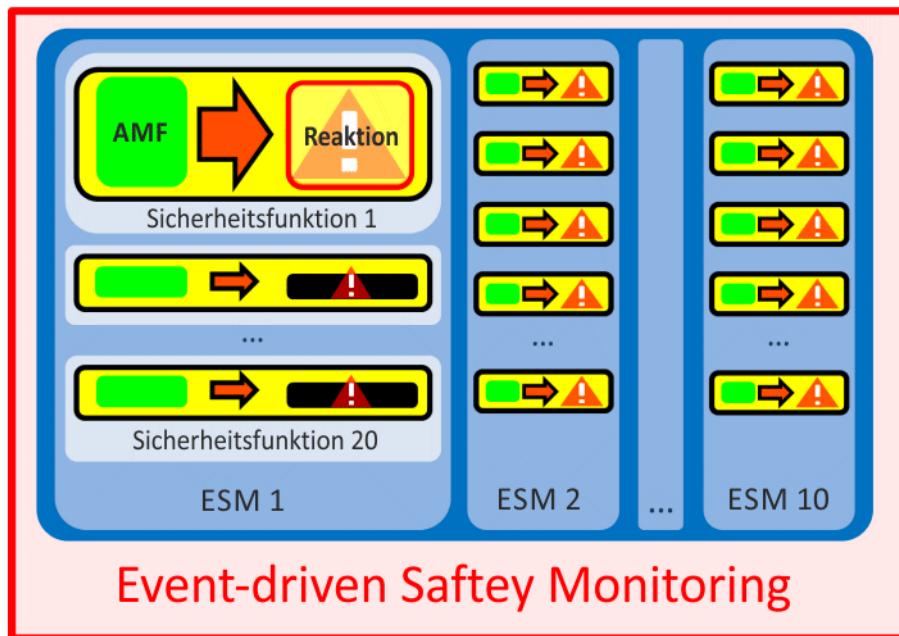


Abb. 13-2: Sicherheitsfunktionen des ESM-Mechanismus

### 13.5 Übersicht Atomic Monitoring Functions

Die kleinste Einheit einer Sicherheitsfunktion wird als Atomic Monitoring Function (AMF) bezeichnet, beispielsweise die Auswertung der Zustimmung am smartPAD oder die Überwachung der Geschwindigkeit einer Achse.

Atomic Monitoring Functions werden in 3 Kategorien unterteilt:

- Standard-AMFs
- Parametrierbare AMFs
- Extended AMFs

#### 13.5.1 Standard Atomic Monitoring Functions

**Beschreibung** Standard Atomic Monitoring Functions liefern Informationen über Systemkomponenten oder Systemzustände, beispielsweise die Sicherheitseinrichtungen am smartPAD oder die aktive Betriebsart. Standard-AMFs können in beliebig vielen Sicherheitsfunktionen verwendet werden.

**Übersicht** AMFs zur Auswertung der Sicherheitseinrichtungen am smartPAD:

AMF	Aufgabe
NOT-HALT smartPAD	Überwacht die NOT-HALT-Einrichtung am smartPAD
Zustimmung smartPAD inaktiv	Prüft, ob die Zustimmung am smartPAD nicht erteilt ist
Zustimmung Panik smartPAD aktiv	Prüft, ob ein Zustimmungsschalter am smartPAD durchgedrückt (Panikstellung) ist.

(>> 13.8.1 "Auswertung der Sicherheitseinrichtungen am KUKA smartPAD" Seite 206)

AMFs zur Auswertung der Betriebsart:

AMF	Aufgabe
Betriebsart Test	Prüft, ob eine Testbetriebsart aktiv ist (T1, T2, KRF)
Betriebsart Automatik	Prüft, ob die Betriebsart Automatik aktiv ist (AUT)

AMF	Aufgabe
<i>Betriebsart mit reduzierter Geschwindigkeit</i>	Prüft, ob eine Betriebsart mit reduzierter Geschwindigkeit aktiv ist (T1, KRF)  <b>Hinweis:</b> Bei einer mobilen Plattform ist die Geschwindigkeit in der Betriebsart T1 und KRF nicht reduziert.
<i>Betriebsart mit hoher Geschwindigkeit</i>	Prüft, ob eine Betriebsart mit programmierte Geschwindigkeit aktiv ist (T2, AUT)

(>>> 13.8.2 "Auswertung der Betriebsart" Seite 207)

AMF zur Auswertung der Fahr freigabe:

AMF	Aufgabe
<i>Fahr freigabe</i>	Überwacht die Fahr freigabe, d. h. sie prüft, ob ein Sicherheits- holt aktiv ist.

(>>> 13.8.3 "Auswertung der Fahr freigabe" Seite 207)

### 13.5.2 Parametrierbare Atomic Monitoring Functions

**Beschreibung** Parametrierbare Atomic Monitoring Functions besitzen im Gegensatz zu Standard-AMFs zusätzlich einen oder mehrere Parameter. Diese können abhängig davon, für welche Werte die AMF verletzt sein soll, konfiguriert werden. Beispielsweise Überwachungsgrenzen.

Für parametrierbare AMFs steht eine fest definierte Anzahl an Instanzen zur Verfügung. Die Anzahl gibt an, wie viele unterschiedlich parametrierte Versionen der AMF konfiguriert und verwendet werden können. Jede Instanz kann unterschiedliche Parameterwerte besitzen. Die Instanz einer AMF darf mehrfach in der Tabelle, in der die Sicherheitsfunktionen konfiguriert werden, verwendet werden.

**Übersicht** AMF zur Auswertung von sicheren Eingängen:

AMF	Aufgabe
<i>Eingangssignal</i>	Überwacht einen sicheren Eingang  (>>> 13.8.4 "Überwachung von sicheren Eingängen" Seite 208)

AMFs zur Auswertung der Zustimmung am Handführgerät:

AMF	Aufgabe
<i>Zustimmung Handführgerät inaktiv</i>	Prüft, ob die Zustimmung am Handführgerät nicht erteilt ist
<i>Zustimmung Handführgerät aktiv</i>	Prüft, ob die Zustimmung am Handführgerät erteilt ist  Die AMF dient zur Aktivierung weiterer Überwachungen während des Handführens mit Zustimmeinrichtung.

(>>> 13.8.5 "Handführen mit Zustimmeinrichtung und Geschwindigkeitsüberwachung" Seite 209)

AMFs zur Auswertung der Referenzierungsstatus:

<b>AMF</b>	<b>Aufgabe</b>
<i>Positionsreferenzierung</i>	Überwacht den Referenzierungsstatus der Positionswerte der Achsen einer Kinematik  (>>> 13.8.6 "Auswertung der Positionsreferenzierung" Seite 212)
<i>Momentenreferenzierung</i>	Überwacht den Referenzierungsstatus der Gelenkmomenten-Sensoren der Achsen einer Kinematik  (>>> 13.8.7 "Auswertung der Momentenreferenzierung" Seite 212)

AMFs zur Geschwindigkeitsüberwachung:

<b>AMF</b>	<b>Aufgabe</b>
<i>Achsgeschwindigkeitsüberwachung</i>	Überwacht die Geschwindigkeit einer der Achsen einer Kinematik  (>>> 13.8.8.1 "Achsspezifische Geschwindigkeitsüberwachung definieren" Seite 213)
<i>Kartesische Geschwindigkeitsüberwachung</i>	Überwacht die translatorische kartesische Geschwindigkeit an definierten Punkten einer Kinematik  (>>> 13.8.8.2 "Kartesische Geschwindigkeitsüberwachung definieren" Seite 214)
<i>Werkzeugbezogene Geschwindigkeitskomponente</i>	Prüft, ob die kartesische translatorische Geschwindigkeit in einer bestimmten Richtung unter der konfigurierten Grenze liegt.  (>>> 13.8.8.3 "Richtungsabhängige Überwachung der kartesischen Geschwindigkeit" Seite 215)

AMFs zur Raumüberwachung:

<b>AMF</b>	<b>Aufgabe</b>
<i>Kartesische Arbeitsraumüberwachung</i>	Prüft, ob sich ein Teil der überwachten Struktur einer Kinematik außerhalb eines erlaubten Arbeitsbereichs befindet  (>>> 13.8.9.1 "Kartesische Arbeitsräume definieren" Seite 222)
<i>Kartesische Schutzraumüberwachung</i>	Prüft, ob sich ein Teil der überwachten Struktur einer Kinematik innerhalb eines nicht erlaubten Schutzraums befindet  (>>> 13.8.9.2 "Kartesische Schutzräume definieren" Seite 224)
<i>Achsbereichsüberwachung</i>	Überwacht die Position einer der Achsen einer Kinematik  (>>> 13.8.9.3 "Achsspezifische Überwachungsräume definieren" Seite 227)

AMF zur Überwachung der Werkzeugorientierung:

<b>AMF</b>	<b>Aufgabe</b>
<i>Werkzeugorientierung</i>	Prüft, ob die Orientierung des Werkzeugs einer Kinematik außerhalb eines zulässigen Bereichs liegt  (>>> 13.8.10 "Überwachung der Werkzeugorientierung" Seite 228)

AMFs zur sicheren Überwachung von Kräften und Momenten (MRK):

AMF	Aufgabe
Achsmomentenüberwachung	Überwacht das Drehmoment einer der Achsen einer Kinematik (>>> 13.8.13.1 "Achsmomentenüberwachung" Seite 233)
Kollisionserkennung	Überwacht das externe Moment der Achsen einer Kinematik (>>> 13.8.13.2 "Kollisionserkennung" Seite 234)
TCP-Kraftüberwachung	Überwacht die externe Kraft, die am TCP des Werkzeugs oder am Roboterflansch einer Kinematik wirkt (>>> 13.8.13.3 "TCP-Kraftüberwachung" Seite 235)

### 13.5.3 Extended Atomic Monitoring Functions

**Beschreibung** Extended Atomic Monitoring Functions unterscheiden sich von den Standard-AMFs und parametrierbaren AMFs dadurch, dass Überwachungsparameter erst im Betrieb festgelegt werden. Die Parameter werden zum Zeitpunkt der Aktivierung gesetzt. Zum Beispiel werden bei der AMF *Stillstandsüberwachung aller Achsen* die Achswinkel zum Zeitpunkt der Aktivierung als Referenzwinkel für die Überwachung gesetzt. Eine Extended AMF wird aktiviert, wenn alle anderen verwendeten AMFs der Sicherheitsfunktion verletzt sind. Solange mindestens eine der anderen AMFs nicht verletzt ist, ist die Extended AMF nicht aktiv und wird nicht ausgewertet.



Extended AMFs werden erst einen Takt nach ihrer Aktivierung ausgewertet. Hieraus kann sich eine Verlängerung der Reaktionszeit von bis zu 12 ms ergeben.

Für Extended AMFs stehen mehrere Instanzen zur Verfügung. Auch bei nicht parametrierbaren Extended AMFs wird empfohlen, jede Instanz nur einmal in der Sicherheitskonfiguration zu verwenden.



Extended AMFs stehen für die Sicherheitsfunktionen des ESM-Mechanismus nicht zur Verfügung.

#### Übersicht

AMF zur Stillstandsüberwachung:

AMF	Aufgabe
<i>Stillstandsüberwachung aller Achsen</i>	Überwacht den Stillstand aller Achsen einer Kinematik. (>>> 13.8.11 "Stillstandsüberwachung (Sicherer Betriebshalt)" Seite 231)

AMF zum Schalten einer Zeitverzögerung:

AMF	Aufgabe
<i>Zeitverzögerung</i>	Verzögert das Auslösen der Reaktion einer Sicherheitsfunktion um eine definierte Zeit. (>>> 13.8.12 "Einschaltverzögerung für Sicherheitsfunktionen" Seite 232)

### 13.5.4 Verfügbarkeit der AMFs in Abhängigkeit von der Kinematik

**Beschreibung** Einige von der System Software zur Verfügung gestellten Sicherheitsüberwachungen (AMFs) sind kinematispezifisch, d. h. bei der Konfiguration dieser AMFs muss die zu überwachende Kinematik ausgewählt werden. (Parameter *Überwachte Kinematik* mit den Werten **Erste Kinematik ... Vierte Kinematik**)

Bei Verwendung kinematikspezifischer AMFs in der Sicherheitskonfiguration ist die zu überwachende Kinematik wie folgt anzugeben:

- **Erste Kinematik:** Wenn der LBR iiwa überwacht werden soll
  - **Zweite Kinematik:** Wenn die mobile Plattform überwacht werden soll
- Dritte und vierte Kinematik sind zur Zeit keiner Kinematik zugeordnet.

## Übersicht

Es sind nicht alle kinematikspezifischen AMFs für die mobile Plattform nutzbar, da die notwendigen sicherheitsgerichteten Sensorinformationen nicht verfügbar sind. Ist eine AMF für die überwachte Kinematik nicht nutzbar, wird sie bei der Auswertung immer als verletzt gewertet.

AMF	LBR iiwa	KMP
Positionsreferenzierung	✓	✗
Momentenreferenzierung	✓	✗
Achsgeschwindigkeitsüberwachung	✓	✓
Kartesische Geschwindigkeitsüberwachung	✓	✓
Werkzeugbezogene Geschwindigkeitskomponente	✓	✓
Kartesische Arbeitsraumüberwachung	✓	✗
Kartesische Schutzraumüberwachung	✓	✗
Achsbereichsüberwachung	✓	✗
Werkzeugorientierung	✓	✗
Achsmomentenüberwachung	✓	✗
Kollisionserkennung	✓	✗
TCP-Kraftüberwachung	✓	✗
Stillstandsüberwachung aller Achsen	✓	✗

## 13.6 Sicherheitskonfiguration mit KUKA Sunrise.Workbench

Die Sicherheitskonfiguration ist fester Bestandteil eines Sunrise-Projekts. Sie wird in Tabellenform verwaltet.

Die Sicherheitskonfiguration definiert die Sicherheitsfunktionen des PSM-Mechanismus und die sicheren Zustände des ESM-Mechanismus.

Beim Erstellen eines neuen Sunrise-Projekts wird automatisch eine Standard-Sicherheitskonfiguration erzeugt.

(>>> 5.3 "Sunrise-Projekt mit Vorlage erstellen" Seite 51)

Die Standard-Sicherheitskonfiguration enthält von KUKA vordefinierte, permanent aktive Sicherheitsfunktionen. Der ESM-Mechanismus besitzt keine vorkonfigurierten Zustände und ist daher deaktiviert.



Weitere Informationen zur Standard-Sicherheitskonfiguration sind im Kapitel "Sicherheit" zu finden.

In KUKA Sunrise.Workbench wird die Sicherheitskonfiguration angezeigt, bearbeitet und über die Installation der System Software oder die Projektsynchronisation auf die Steuerung übertragen. Über die smartHMI kann die Sicherheitskonfiguration aktiviert und deaktiviert werden.

### 13.6.1 Übersicht Sicherheitskonfiguration ändern und auf Steuerung aktivieren

Schritt	Beschreibung
1	Sicherheitskonfiguration öffnen. (>>> 13.6.2 "Sicherheitskonfiguration öffnen" Seite 195)
2	Sicherheitsfunktionen in der Tabelle Anwender PSM bearbeiten oder neue Sicherheitsfunktionen erstellen. (>>> 13.6.3 "Sicherheitsfunktionen des PSM-Mechanismus konfigurieren" Seite 197)
3	Falls erforderlich, ereignisabhängige Überwachungen konfigurieren. Hierzu sichere ESM-Zustände erzeugen und zugehörige Sicherheitsfunktionen konfigurieren. Vorhandene ESM-Zustände können durch Anpassen bereits konfigurierter oder durch Hinzufügen neuer Sicherheitsfunktionen geändert werden. (>>> 13.6.4 "Sichere Zustände des ESM-Mechanismus konfigurieren" Seite 200)
4	Sicherheitskonfiguration speichern.
5	Bei Verwendung des ESM-Mechanismus Die benötigte Umschaltung zwischen den sicheren Zuständen in Roboter-Applikationen und Hintergrund-Tasks programmieren. (>>> 13.6.4.8 "Umschalten zwischen ESM-Zuständen" Seite 204)
6	Bei Verwendung positionsbasierter AMFs (>>> "Positionsisierte AMFs" Seite 246) Falls erforderlich, die von KUKA vorbereitete Applikation für die Positions- und Momentenreferenzierung des LBR iiwa oder eine eigene Referenzfahrt-Applikation erstellen. (>>> 13.10.1 "Positionsreferenzierung" Seite 240)
7	Bei Verwendung achsmomentbasierter AMFs (>>> "Achsmomentbasierte AMFs" Seite 247) Die von KUKA vorbereitete Applikation für die Positions- und Momentenreferenzierung des LBR iiwa erstellen und das sicherheitsgerichtete Werkzeug in die Applikation einbinden. Weitere Anpassungen in der Applikation können erforderlich sein. (>>> 13.10.2 "Momentenreferenzierung" Seite 241)
8	Projekt mit der Sicherheitskonfiguration auf die Robotersteuerung übertragen. <ul style="list-style-type: none"> <li>■ Über die Installation der System Software oder über die Projekt-Synchronisation</li> </ul>
9	Robotersteuerung neu starten, um die geänderte Sicherheitskonfiguration zu übernehmen.
10	Sicherheitskonfiguration auf der Robotersteuerung aktivieren. (>>> 13.7 "Sicherheitskonfiguration auf Robotersteuerung aktivieren" Seite 205)
11	Bei Verwendung positionsbasierter AMFs (>>> "Positionsisierte AMFs" Seite 246) Positionsreferenzierung durchführen.

Schritt	Beschreibung
12	Bei Verwendung achsmomentbasierter AMFs ( <a href="#">&gt;&gt;&gt; "Achsmomentbasierte AMFs"</a> Seite 247) Momentenreferenzierung durchführen.
13	Sicherheitsfunktionen der aktiven Sicherheitskonfiguration testen. ( <a href="#">&gt;&gt;&gt; 13.11 "Übersicht Sicherheitsabnahme"</a> Seite 243)

### 13.6.2 Sicherheitskonfiguration öffnen

- Vorgehensweise**
- Im Sunrise-Projekt auf die Datei **SafetyConfiguration.sconf** doppelklicken.
- Beschreibung**
- Die Sicherheitskonfiguration enthält mehrere Tabellen.
- **KUKA PSM**  
Die Tabelle enthält die von KUKA vorgeschriebenen Sicherheitsfunktionen. Diese können weder geändert noch deaktiviert werden.  
Die Tabelle dient dazu, das Systemverhalten zu dokumentieren und ergibt im Zusammenspiel mit der Tabelle **Anwender PSM** eine vollständige Beschreibung der ständig aktiven Sicherheitsfunktionen.
  - **Anwender PSM**  
In dieser Tabelle werden die anwenderspezifischen Sicherheitsfunktionen konfiguriert. Sie enthält von KUKA vorkonfigurierte Sicherheitsfunktionen. Diese können deaktiviert, geändert oder gelöscht werden.
  - **ESM**  
Für jeden ESM-Zustand wird eine eigene Tabelle angelegt. Diese enthält die Sicherheitsfunktionen des Zustands. Die Standardkonfiguration enthält keine vorkonfigurierten ESM-Zustände.

#### 13.6.2.1 Auswertung der Sicherheitskonfiguration

Bei der Auswertung der Sicherheitskonfiguration werden die Tabellen **Anwender PSM** und **KUKA PSM** immer parallel überprüft. Es ist möglich, dass die beiden Tabellen eine identische Sicherheitsfunktion mit unterschiedlichen Reaktionen enthalten. Sind dabei unterschiedliche Stopp-Reaktionen konfiguriert, wird bei einer Verletzung die schärfere Stopp-Reaktion ausgelöst. D. h. es wird diejenige Stopp-Reaktion ausgelöst, die zu einem früheren sicherheitsgerichteten Abschalten der Antriebe führt.

Wird der ESM-Mechanismus verwendet, werden zusätzlich alle Sicherheitsfunktionen des aktuell aktiven ESM-Zustands überwacht.

### 13.6.2.2 Übersicht Bedienoberfläche Sicherheitskonfiguration

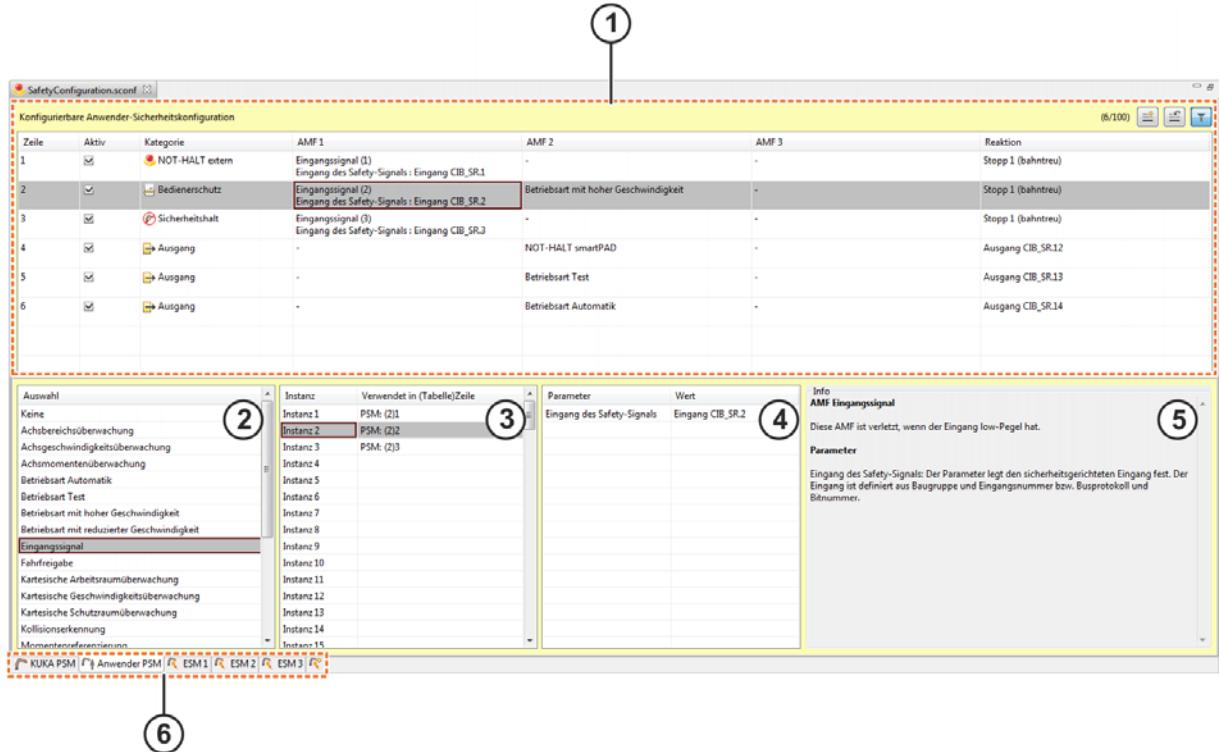
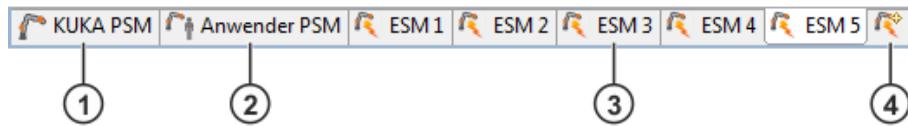


Abb. 13-3: Bedienoberfläche Sicherheitskonfiguration

Pos.	Beschreibung
1	Ausgewählte Tabelle Enthält die konfigurierten Sicherheitsfunktionen der ausgewählten PSM-Tabelle oder des ausgewählten ESM-Zustands.
2	Auswahl-Tabelle In Abhängigkeit von der Zelle, die in der ausgewählten Tabellenzeile markiert ist, kann hier die Kategorie, AMF oder Reaktion einer Sicherheitsfunktion ausgewählt werden.
3	Instanz-Tabelle Hier werden die Instanzen der in der Auswahl-Tabelle ausgewählten AMF angezeigt, sowie die Tabellenzeilen, in denen sie verwendet werden.
4	Parameter-Tabelle Hier werden die Parameterwerte der AMF-Instanz angezeigt, die in der Instanz-Tabelle markiert ist. Die Werte können geändert werden.
5	Informationsanzeige Anzeige der Beschreibung der ausgewählten Kategorie, AMF oder Reaktion
6	Tabellenverzeichnis Hier können die gewünschten Tabellen ausgewählt und neue ESM-Zustände hinzugefügt werden.

Über das Tabellenverzeichnis im unteren Bereich des Editors wird die Tabelle ausgewählt, die angezeigt und bearbeitet werden soll.



**Abb. 13-4: Tabellenverzeichnis Sicherheitskonfiguration**

Pos.	Beschreibung
1	Registerkarte <i>KUKA PSM</i> Öffnet die PSM-Tabelle <i>KUKA PSM</i> . Eine Bearbeitung der Tabelle ist nicht möglich.
2	Registerkarte <i>Anwender PSM</i> Öffnet die PSM-Tabelle <i>Anwender PSM</i> . Die Tabelle kann bearbeitet werden.
3	Registerkarte für einen ESM-Zustand Öffnet den ESM-Zustand. Der ESM-Zustand kann bearbeitet werden.
4	Schaltfläche <b>Neuen ESM-Zustand hinzufügen</b> Fügt einen neuen ESM-Zustand hinzu. Der neue Zustand wird automatisch geöffnet und kann bearbeitet werden.

### 13.6.3 Sicherheitsfunktionen des PSM-Mechanismus konfigurieren

Der PSM-Mechanismus definiert Sicherheitsfunktionen, die permanent aktiv sind.

Die Sicherheitsfunktionen werden in Tabellenform dargestellt. Jede Zeile der Tabelle enthält eine Sicherheitsfunktion.

In der PSM-Tabelle **Anwender PSM** werden neue Sicherheitsfunktionen hinzugefügt und bestehende Einstellungen angepasst. Dabei können die Kategorie, die verwendeten Atomic Monitoring Functions (AMFs), die Parametrierung der AMF-Instanzen und die Reaktion geändert werden. Einzelne Sicherheitsfunktionen können aktiv oder inaktiv gesetzt werden.

#### 13.6.3.1 PSM-Tabelle *Anwender PSM* öffnen

##### Vorgehensweise

1. Sicherheitskonfiguration öffnen.
2. Registerkarte *Anwender PSM* im Tabellenverzeichnis auswählen. Die PSM-Tabelle *Anwender PSM* wird angezeigt und kann bearbeitet werden.

The diagram illustrates the PSM configuration table with numbered callouts:

- 1**: Column header "Aktiv" (Active).
- 2**: Column header "Kategorie" (Category).
- 3**: Column headers "AMF 1", "AMF 2", and "AMF 3".
- 4**: Cell "(6/100)" indicating the number of configured functions.
- 5**: Cell containing three icons: a copy symbol, a delete symbol, and a refresh symbol.
- 6**: Cell containing a small trash can icon.
- 7**: Callout pointing to the currently selected row (row 2), which is highlighted with a gray background.

Konfigurierbare Anwender-Sicherheitskonfiguration						
Zeile	Aktiv	Kategorie	AMF 1	AMF 2	AMF 3	Reaktion
1	<input checked="" type="checkbox"/>	NOT-HALT extern	Eingangssignal (1) Eingang des Safety-Signals : EingangCIB_SR.1	-	-	Stopp 1 (bahntreu)
2	<input checked="" type="checkbox"/>	Bedienerschutz	Eingangssignal (2) Eingang des Safety-Signals : EingangCIB_SR.2	Betriebsart mit hoher Geschwindigkeit	-	Stopp 1 (bahntreu)
3	<input checked="" type="checkbox"/>	Sicherheitshalt	Eingangssignal (3) Eingang des Safety-Signals : EingangCIB_SR.3	-	-	Stopp 1 (bahntreu)
4	<input checked="" type="checkbox"/>	Ausgang	-	NOT-HALT smartPAD	-	Ausgang CIB_SR.12
5	<input checked="" type="checkbox"/>	Ausgang	-	Betriebsart Test	-	Ausgang CIB_SR.13
6	<input checked="" type="checkbox"/>	Ausgang	-	Betriebsart Automatik	-	Ausgang CIB_SR.14

**Abb. 13-5: PSM-Tabelle Anwender PSM**

Pos.	Beschreibung
1	<b>Spalte Aktiv</b> Legt fest, ob die Sicherheitsfunktion aktiv ist. Deaktivierte Sicherheitsfunktionen werden nicht überwacht. <ul style="list-style-type: none"> <li>■ Hækchen gesetzt: Sicherheitsfunktion ist aktiv.</li> <li>■ Hækchen nicht gesetzt: Sicherheitsfunktion ist deaktiviert.</li> </ul>
2	<b>Spalte Kategorie</b> Legt die Kategorie der Sicherheitsfunktion fest. Die Kategorie wird im Fehlerfall auf der smartHMI als Fehlerursache angegeben.
3	<b>Spalten AMF 1, AMF 2, AMF 3</b> Legen die einzelnen AMFs der Sicherheitsfunktion fest. Es können bis zu 3 AMFs verwendet werden. Die Sicherheitsfunktion ist verletzt, wenn alle verwendeten AMFs verletzt sind.
4	<b>Spalte Reaktion</b> Legt die Reaktion der Sicherheitsfunktion fest. Sie wird ausgelöst, wenn die Sicherheitsfunktion verletzt ist.
5	<b>Anzeige der Anzahl der aktuell konfigurierten Sicherheitsfunktionen</b> Insgesamt stehen 100 Zeilen für die Konfiguration von anwenderspezifischen Sicherheitsüberwachungen zur Verfügung.
6	<b>Schaltflächen zur Bearbeitung der Tabelle</b>
7	<b>Ausgewählte Zeile</b> Die Zeile, die die aktuell ausgewählte Sicherheitsfunktion enthält, wird grau hinterlegt.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
	<b>Neue Zeile</b> Fügt eine neue Zeile in die Tabelle ein (nur möglich wenn Default-Zeilen ausgeblendet sind). Die neue Zeile besitzt die Standardkonfiguration und wird automatisch aktiviert.
	<b>Zeile zurücksetzen</b> Setzt die Konfiguration der markierten Zeile auf die Standardkonfiguration zurück. Die Sicherheitsfunktion wird deaktiviert.
	<b>Leere Zeilen einblenden / Leere Zeilen ausblenden</b> Alle nicht konfigurierten leeren Zeilen sind deaktiviert und mit einer Standardkonfiguration vorbelegt. <ul style="list-style-type: none"> <li>■ <i>Kategorie:</i> Keine</li> <li>■ <i>AMF 1, AMF 2, AMF 3:</i> Keine</li> <li>■ <i>Reaktion:</i> Stopp 1</li> </ul> Die leeren Zeilen können ein- oder ausgeblendet werden. Defaultmäßig sind die leeren Zeilen ausgeblendet.

### 13.6.3.2 Sicherheitsfunktionen für PSM-Mechanismus erstellen

- Voraussetzung** ■ PSM-Tabelle *Anwender PSM* ist geöffnet.
- Vorgehensweise**
- Nicht konfigurierte leere Zeilen sind eingeblendet:**
    1. Eine leere Zeile in der Tabelle auswählen.
    2. Die Kategorie, die verwendeten AMFs und die Reaktion der Sicherheitsfunktion in den zugehörigen Spalten einstellen.
    3. In der Spalte *Aktiv* das Häkchen setzen, wenn die Zeile aktiviert werden soll.
  - Nicht konfigurierte leere Zeilen sind ausgeblendet:**
    1. Auf **Neue Zeile** klicken. Eine vorkonfigurierte Zeile wird in die Tabelle eingefügt. Die Zeile ist automatisch aktiv gesetzt (Häkchen in Spalte *Aktiv*).
    2. Die Kategorie, die verwendeten AMFs und die Reaktion der Sicherheitsfunktion in den zugehörigen Spalten einstellen.

### 13.6.3.3 Sicherheitsfunktion des PSM-Mechanismus löschen

- Voraussetzung** ■ PSM-Tabelle *Anwender PSM* ist geöffnet.
- Vorgehensweise**
1. Zeile mit der zu löschenen Sicherheitsfunktion in der Tabelle markieren.
  2. Auf **Zeile zurücksetzen** klicken. Die Sicherheitsfunktion wird deaktiviert und erhält die Standardkonfiguration (Kategorie: Keine, AMF: Keine, Reaktion: Stopp 1).

### 13.6.3.4 Bestehende Sicherheitsfunktionen des PSM-Mechanismus bearbeiten

- Voraussetzung** ■ PSM-Tabelle *Anwender PSM* ist geöffnet.
- Vorgehensweise**
- Kategorie ändern:**
    1. Die Spalte *Kategorie* in der gewünschten Zeile markieren. In der Auswahl-Tabelle werden die zur Verfügung stehenden Kategorien angezeigt.

2. Gewünschte Kategorie in der Auswahl-Tabelle markieren. Die Kategorie wird für die Sicherheitsfunktion übernommen.

#### Verwendete AMF ändern:

1. Die Spalte *AMF 1*, *AMF 2* oder *AMF 3* in der gewünschten Zeile markieren. In der Auswahl-Tabelle werden die zur Verfügung stehenden AMFs angezeigt.
2. Gewünschte AMF in der Auswahl-Tabelle markieren. Die AMF wird für die Sicherheitsfunktion übernommen.
3. Bei mehrfach instanzierten AMFs: Gewünschte Instanz in der Instanz-Tabelle auswählen. Die Instanz wird für die Sicherheitsfunktion übernommen.
4. Bei parametrierbaren AMFs: In der Parameter-Tabelle die Parameter der AMF in der Spalte **Wert** einstellen und die Einstellungen mit der Eingabe-Taste übernehmen.

#### Reaktion ändern:

1. Die Spalte *Reaktion* in der gewünschten Zeile markieren. In der Auswahl-Tabelle werden die zur Verfügung stehenden Reaktionen angezeigt.
2. Gewünschte Reaktion in der Auswahl-Tabelle markieren. Die Reaktion wird für die Sicherheitsfunktion übernommen.
3. Wenn die Reaktion **Ausgang** ausgewählt wurde: In der Parameter-Tabelle das Ausgangs-Bit wählen, dessen Signal bei einer Verletzung der Sicherheitsfunktion auf LOW-Pegel gesetzt werden soll. Die Einstellung mit der Eingabe-Taste übernehmen.

#### Sicherheitsfunktion aktivieren / deaktivieren:

- In der gewünschten Zeile auf die Spalte *Aktiv* klicken. Das Häkchen wird gesetzt / entfernt.



Nur die aktivierte Sicherheitsfunktionen stehen nach dem Übertragen und Aktivieren der Sicherheitskonfiguration auf der Robotersteuerung zur Verfügung.

### 13.6.4 Sichere Zustände des ESM-Mechanismus konfigurieren

Über den ESM-Mechanismus werden unterschiedliche Sicherheitseinstellungen durch konfigurierbare sichere Zustände festgelegt. Es können bis zu 10 sichere Zustände angelegt werden. Die Zustände werden fortlaufend von 1 bis 10 durchnummeriert und sind damit eindeutig identifizierbar.

Ein sicherer Zustand wird in einer Tabelle mit bis zu 20 Sicherheitsfunktionen definiert. Diese Sicherheitsfunktionen legen die Sicherheitseinstellungen fest, die für den Zustand gelten müssen.

Ein sicherer Zustand wird in einer Tabelle dargestellt. Jede Zeile der Tabelle enthält eine Sicherheitsfunktion.

Die Verwendung des ESM-Mechanismus ist optional. Der ESM-Mechanismus ist aktiviert, wenn mindestens ein ESM-Zustand konfiguriert ist. Sind keine ESM-Zustände konfiguriert, ist der Mechanismus deaktiviert.

Ist der ESM-Mechanismus aktiv, ist immer genau ein sicherer Zustand gültig, dessen Sicherheitsfunktionen zusätzlich zu den permanent aktiven Sicherheitsfunktionen überwacht werden. Zwischen den konfigurierten sicheren Zuständen kann situationsabhängig umgeschaltet werden. Die Umschaltung erfolgt über die Roboter-Applikation oder über einen Hintergrund-Task.

(>>> 13.6.4.8 "Umschalten zwischen ESM-Zuständen" Seite 204)

Ein ESM-Zustand ist so lange aktiv, bis die Umschaltung in einen anderen ESM-Zustand kommandiert wird.

Nach dem Hochfahren der Steuerung ist automatisch der konfigurierte ESM-Zustand mit der niedrigsten Nummer aktiv.

#### 13.6.4.1 Neuen ESM-Zustand hinzufügen

Für den ESM-Mechanismus können bis zu 10 sichere Zustände erstellt werden. Ist diese Anzahl erreicht, wird die Registerkarte zum Hinzufügen neuer Zustände ausgeblendet.

##### Vorgehensweise

1. Sicherheitskonfiguration öffnen.
2. Registerkarte **Neuer ESM Zustand hinzufügen** im Tabellenverzeichnis wählen. Ein neuer ESM-Zustand wird erstellt.

Der neue ESM-Zustand erhält die kleinste nicht verwendete Zustandsnummer. Er besitzt eine aktive Sicherheitsfunktion mit Standardkonfiguration. Im Tabellenverzeichnis wird eine neue Registerkarte für den Zustand hinzugefügt. Die Tabelle des Zustands wird automatisch geöffnet und kann bearbeitet werden.

#### 13.6.4.2 Tabelle eines ESM-Zustands öffnen

##### Voraussetzung

- Der ESM-Mechanismus ist aktiviert.

##### Vorgehensweise

1. Sicherheitskonfiguration öffnen.
2. Registerkarte des gewünschten ESM-Zustands im Tabellenverzeichnis auswählen. Die Tabelle des ESM-Zustands wird angezeigt und kann bearbeitet werden.

Zeilenummer	Aktiv	AMF	Reaktion
1	<input checked="" type="checkbox"/>	Kollisionserkennung (1) Maximales externes Moment : 15 Nm	Stop 1
2	<input checked="" type="checkbox"/>	Kartesische Geschwindigkeitsüberwachung (2) Maximale Geschwindigkeit : 250 mm/s	Stop 1

Abb. 13-6: Tabelle eines ESM-Zustands

Pos.	Beschreibung
1	<p>Spalte <i>Aktiv</i></p> <p>Legt fest, ob die Sicherheitsfunktion aktiv ist. Deaktivierte Sicherheitsfunktionen werden nicht überwacht.</p> <ul style="list-style-type: none"> <li>■ Häkchen gesetzt: Sicherheitsfunktion ist aktiv.</li> <li>■ Häkchen nicht gesetzt: Sicherheitsfunktion ist deaktiviert.</li> </ul> <p>Die Sicherheitsfunktion in der ersten Zeile der Tabelle ist immer aktiv. Sie kann nicht deaktiviert werden (gekennzeichnet durch Schloss-Symbol).</p>
2	<p>Spalte <i>AMF</i></p> <p>Legt die AMF der Sicherheitsfunktion fest. Für Sicherheitsfunktionen von ESM-Zuständen wird nur eine AMF verwendet. Ist diese AMF verletzt, ist die Sicherheitsfunktion und damit der gesamte Zustand verletzt.</p>
3	<p>Spalte <i>Reaktion</i></p> <p>Legt die Reaktion der Sicherheitsfunktion fest. Sie wird ausgelöst, wenn die Sicherheitsfunktion verletzt ist.</p>
4	<p>Anzeige der Anzahl der aktuell konfigurierten Sicherheitsfunktionen</p> <p>Insgesamt stehen 20 Zeilen für die Konfiguration von Sicherheitsüberwachungen eines ESM-Zustands zur Verfügung.</p>
5	Schaltflächen zur Bearbeitung der Tabelle
6	Ausgewählte Zeile  Die Zeile, die die aktuell ausgewählte Sicherheitsfunktion enthält, wird grau hinterlegt.

Folgende Schaltflächen stehen zur Verfügung:

Schaltfläche	Beschreibung
	<p><b>Zustand löschen</b></p> <p>Löscht den gesamten Zustand. Der Löschevorgang muss über einen Dialog bestätigt werden.</p>
	<p><b>Neue Zeile</b></p> <p>Fügt eine neue Zeile in die Tabelle ein (nur möglich wenn Default-Zeilen ausgeblendet sind). Die neue Zeile besitzt die Standardkonfiguration und wird automatisch aktiviert.</p>
	<p><b>Zeile zurücksetzen</b></p> <p>Setzt die Konfiguration der markierten Zeile auf die Standardkonfiguration zurück. Die Sicherheitsfunktion wird deaktiviert (Ausnahme: Die erste Zeile der Tabelle ist immer aktiv).</p>
	<p><b>Leere Zeilen einblenden / Leere Zeilen ausblenden</b></p> <p>Alle nicht konfigurierten leeren Zeilen sind deaktiviert und mit einer Standardkonfiguration vorbelegt.</p> <ul style="list-style-type: none"> <li>■ <i>AMF</i>: Keine</li> <li>■ <i>Reaktion</i>: Stopp 1</li> </ul> <p>Die leeren Zeilen können ein- oder ausgeblendet werden. Defaultmäßig sind die leeren Zeilen ausgeblendet.</p>

### 13.6.4.3 ESM-Zustand löschen

- Vorgehensweise**
1. Sicherheitskonfiguration öffnen.
  2. Registerkarte des zu löschenenden ESM-Zustands im Tabellenverzeichnis auswählen.
  3. Auf **Zustand löschen** klicken.
  4. Sicherheitsabfrage mit **Ja** beantworten. Der Zustand wird entfernt.
- Ein ESM-Zustand wird nach dem Speichern und Schließen der Sicherheitskonfiguration automatisch entfernt, wenn er folgende Einstellungen besitzt:
- Alle Zeilen besitzen Standardkonfiguration (AMF: Keine, Reaktion: Stopp 1).
  - Erste Zeile ist aktiviert, alle anderen Zeilen sind deaktiviert.

### 13.6.4.4 Sicherheitsfunktion für ESM-Zustand erstellen

- Voraussetzung**
- Die Tabelle des gewünschten ESM-Zustands ist geöffnet.
- Vorgehensweise**
- Nicht konfigurierte leere Zeilen sind eingeblendet:**
1. Eine leere Zeile in der Tabelle auswählen.
  2. Die verwendete AMF und die Reaktion der Sicherheitsfunktion in den zugehörigen Spalten einstellen.
  3. In der Spalte *Aktiv* das Häkchen setzen, wenn die Zeile aktiviert werden soll.
- Nicht konfigurierte leere Zeilen sind ausgeblendet:**
1. Auf **Neue Zeile** klicken. Eine vorkonfigurierte Zeile wird in die Tabelle eingefügt. Die Zeile ist automatisch aktiv gesetzt (Häkchen in Spalte *Aktiv*).
  2. Die verwendete AMF und die Reaktion der Sicherheitsfunktion in den zugehörigen Spalten einstellen.

### 13.6.4.5 Sicherheitsfunktion eines ESM-Zustands löschen

- Voraussetzung**
- Die Tabelle des gewünschten ESM-Zustands ist geöffnet.
- Vorgehensweise**
1. Zeile mit der zu löschenen Sicherheitsfunktion in der Tabelle markieren.
  2. Auf **Zeile zurücksetzen** klicken. Die Sicherheitsfunktion wird deaktiviert und erhält die Standardkonfiguration (AMF: Keine, Reaktion: Stopp 1).

### 13.6.4.6 Bestehende Sicherheitsfunktion eines ESM-Zustands bearbeiten

- Voraussetzung**
- Die Tabelle des gewünschten ESM-Zustands ist geöffnet.
- Vorgehensweise**
- Verwendete AMF ändern:**
1. Die Spalte **AMF** in der gewünschten Zeile markieren. In der Auswahl-Tabelle werden die zur Verfügung stehenden AMFs angezeigt.
  2. Gewünschte AMF in der Auswahl-Tabelle markieren. Die AMF wird für die Sicherheitsfunktion übernommen.
  3. Bei mehrfach instanzierten AMFs: Gewünschte Instanz in der Instanz-Tabelle auswählen. Die Instanz wird für die Sicherheitsfunktion übernommen.
  4. Bei parametrierbaren AMFs: In der Parameter-Tabelle die Parameter der AMF in der Spalte **Wert** einstellen und die Einstellungen mit der Eingabe-Taste übernehmen.
- Reaktion ändern:**

1. Die Spalte **Reaktion** in der gewünschten Zeile markieren. In der Auswahl-Tabelle werden die zur Verfügung stehenden Reaktionen angezeigt.
2. Gewünschte Reaktion in der Auswahl-Tabelle markieren. Die Reaktion wird für die Sicherheitsfunktion übernommen.

**Sicherheitsfunktion aktivieren / deaktivieren:**

- In der gewünschten Zeile auf die Spalte **Aktiv** klicken. Das Häkchen wird gesetzt / entfernt.

#### 13.6.4.7 ESM-Mechanismus deaktivieren

Die Verwendung des ESM-Mechanismus ist optional. Er kann deaktiviert werden.

- Vorgehensweise** ■ Alle ESM-Zustände löschen.

#### 13.6.4.8 Umschalten zwischen ESM-Zuständen

Mit der Methode `setESMState(...)` kann ein ESM-Zustand aktiviert und zwischen den verschiedenen ESM-Zuständen umgeschalten werden. Die Methode gehört zur Klasse LBR und kann in Roboter-Applikationen oder Hintergrund-Tasks verwendet werden.

**Syntax** `lbr.setESMState(state);`

**Erläuterung der Syntax**

Element	Beschreibung
<code>lbr</code>	Typ: LBR Name des Roboters, für den der ESM-Zustand aktiviert wird
<code>state</code>	Typ: String Nummer des ESM-Zustands, der aktiviert wird ■ <b>1 ... 10</b> Wird ein nicht konfigurierter ESM-Zustand angegeben, stoppt der Roboter mit einem Sicherheitshalt 1.

**Beispiel**

In einer Applikation soll der LBR iiwa von Hand geführt werden. Dafür wird eine geeignete Startposition angefahren. Zum Anfahren der Startposition muss ESM-Zustand 3 aktiviert werden. ESM-Zustand 3 gewährleistet eine sensible Kollisionserkennung und überwacht die kartesische Geschwindigkeit.

Nach dem Erreichen des Startpunktes soll das Handführen beginnen. Zum Handführen muss ESM-Zustand 8 aktiviert werden. ESM-Zustand 8 fordert die Zustimmung am Handführgerät, lässt aber eine höhere kartesische Geschwindigkeit zu als ESM-Zustand 3.

```
private LBR lbr;
...
public void run() {
    ...
    lbr.setESMState("3");
    lbr.move(lin(getFrame("Start")).setCartVelocity(300));

    lbr.setESMState("8");
    lbr.move(handGuiding());
```

...  
}

## 13.7 Sicherheitskonfiguration auf Robotersteuerung aktivieren

### Beschreibung

Eine Sicherheitskonfiguration ist erst wirksam, wenn sie nach der Übertragung auf die Robotersteuerung über die smartHMI aktiviert wird. Ist keine Sicherheitskonfiguration aktiv, kann der Roboter nicht verfahren werden.

Bei der Aktivierung wird der Sicherheitskonfiguration eine eindeutige ID (= Prüfsumme der Sicherheitskonfiguration) zugewiesen. Diese wird unter der Ebene **Safety** im Bereich **Aktivierung** angezeigt. Mit dieser ID kann der Sicherheitsinbetriebnehmer die auf der Robotersteuerung aktivierte Sicherheitskonfiguration eindeutig identifizieren.

Bei Installation der System Software ist ein Neustart der Robotersteuerung erforderlich. Wird mit der Installation eine geänderte Sicherheitskonfiguration übertragen, muss diese nach dem Neustart aktiviert werden.

Werden Änderungen einer aktiven Sicherheitskonfiguration per Projekt-Synchronisation auf die Robotersteuerung übertragen, ist das Verhalten des Systems abhängig davon, ob die Synchronisation mit einem Neustart der Robotersteuerung abgeschlossen wurde:

- Synchronisation mit Neustart: Die alte Sicherheitskonfiguration ist nicht mehr und die neue noch nicht aktiv. Der Roboter kann nicht mehr verfahren werden.
- Synchronisation ohne Neustart: Die alte Sicherheitskonfiguration bleibt noch so lange aktiv bis die Robotersteuerung neu gestartet wird. Bis dahin kann der Roboter verfahren werden.



Für den Fall, dass die neu übertragene Sicherheitskonfiguration nicht aktiviert werden soll, kann die alte Sicherheitskonfiguration wiederhergestellt werden.

(>>> 13.7.2 "Sicherheitskonfiguration wiederherstellen" Seite 206)

Zur Aktivierung ist die Eingabe eines Passworts notwendig. Das Default-Passwort lautet "argus".



Das Passwort für die Aktivierung der Sicherheitskonfiguration muss vor der Inbetriebnahme geändert werden. Dieses Passwort darf nur geschulten Sicherheitsinbetriebnehmern bekanntgegeben werden, die autorisiert sind, die Sicherheitskonfiguration zu aktivieren.

(>>> 13.7.3 "Passwort für die Aktivierung der Sicherheitskonfiguration ändern" Seite 206)

### Vorgehensweise

1. In der Stationssicht **Safety** > **Aktivierung** wählen.
2. Passwort eingeben und auf **Aktivieren** drücken.
3. Nur notwendig, wenn nach der Installation der System Software oder Projekt-Synchronisation kein Neustart ausgeführt wurde: Robotersteuerung neu starten.

## 13.7.1 Sicherheitskonfiguration deaktivieren

### Beschreibung

Eine aktivierte Sicherheitskonfiguration kann wieder deaktiviert werden.

Es ist vom Sicherheitsinbetriebnehmer zu prüfen, dass die Deaktivierung erfolgreich war. Ist die Sicherheitskonfiguration deaktiviert, kann der Roboter nicht mehr verfahren werden. Dazu wird eine Meldung in der Stationssicht angezeigt (Kachel **Safety**).

- Vorgehensweise**
1. In der Stationssicht **Safety > Aktivierung** wählen.
  2. Passwort eingeben und auf **Deaktivieren** drücken.

### 13.7.2 Sicherheitskonfiguration wiederherstellen

- Beschreibung**
- Diese Funktion steht nur zur Verfügung, wenn eine neue Sicherheitskonfiguration auf die Robotersteuerung übertragen, aber nicht aktiviert wurde. Mit dieser Vorgehensweise kann die zuletzt aktivierte Sicherheitskonfiguration wiederhergestellt werden.

- Vorgehensweise**
1. In der Stationssicht **Safety > Aktivierung** wählen.
  2. Passwort eingeben und auf **Zurücksetzen** drücken.
  3. Nur notwendig, wenn nach der Projekt-Synchronisation kein Neustart ausgeführt wurde: Robotersteuerung neu starten.

### 13.7.3 Passwort für die Aktivierung der Sicherheitskonfiguration ändern

- Vorgehensweise**
1. In der Stationssicht **Safety > Neues Passwort** wählen.
  2. Im Feld **Passwort** das alte Passwort eingeben.
  3. In den Feldern darunter das neue Passwort 2-mal eingeben.  
Die Eingaben werden aus Sicherheitsgründen verschlüsselt angezeigt.  
Die Groß- und Kleinschreibung wird berücksichtigt.
  4. **Ändern** drücken. Das neue Passwort ist sofort gültig.



Wenn das Passwort vergessen wird, muss die Speicherkarte in der Robotersteuerung vom KUKA Service getauscht werden. Auf der Basis des Default-Passworts muss dann das Passwort für die Aktivierung der Sicherheitskonfiguration erneut geändert werden.

## 13.8 Verwendung und Parametrierung der Atomic Monitoring Functions

Im Folgenden werden Verwendung, Funktionsweise und Parametrierung der einzelnen AMFs beschrieben. Die Beschreibung wird durch Konfigurationsbeispiele ergänzt.

### 13.8.1 Auswertung der Sicherheitseinrichtungen am KUKA smartPAD

Das KUKA smartPAD verfügt über eine NOT-HALT-Einrichtung und eine Zustimmeinrichtung. Die zugehörigen sicherheitsgerichteten Funktionen sind in der KUKA PSM-Tabelle vorkonfiguriert und nicht veränderbar.

Weitere Sicherheitsfunktionen, die die Sicherheitseinrichtungen am smartPAD auswerten, sind konfigurierbar. Folgende AMFs stehen hierfür zur Verfügung:

AMF	Beschreibung
NOT-HALT smartPAD	Die AMF ist verletzt, wenn das NOT-HALT-Gerät am smartPAD gedrückt ist. Standard-AMF

<b>AMF</b>	<b>Beschreibung</b>
<i>Zustimmung smartPAD inaktiv</i>	Die AMF ist verletzt, wenn keine Zustimmung am smartPAD erteilt ist (kein Zustimmungsschalter am smartPAD gedrückt oder Zustimmungsschalter durchgedrückt). Standard-AMF
<i>Zustimmung Panik smartPAD aktiv</i>	Die AMF ist verletzt, wenn ein Zustimmungsschalter am smartPAD durchgedrückt ist (Panikstellung). Standard-AMF

### 13.8.2 Auswertung der Betriebsart

Die eingestellte Betriebsart hat starke Auswirkungen auf das Verhalten des Industrieroboters und bestimmt, welche Sicherheitsvorkehrungen erforderlich sind.

Um eine Sicherheitsfunktion zu konfigurieren, die die eingestellte Betriebsart auswertet, stehen folgende AMFs zur Verfügung:

<b>AMF</b>	<b>Beschreibung</b>
<b>Betriebsart Test</b>	Diese AMF ist verletzt, wenn eine Test-Betriebsart aktiv ist (T1, T2, KRF). Standard-AMF
<b>Betriebsart Automatik</b>	Diese AMF ist verletzt, wenn eine Automatik-Betriebsart aktiv ist (AUT). Standard-AMF
<b>Betriebsart mit reduzierter Geschwindigkeit</b>	Diese AMF ist verletzt, wenn eine Betriebsart aktiv ist, bei der die Geschwindigkeit auf maximal 250mm/s reduziert ist (T1, KRF). Standard-AMF
<b>Betriebsart mit hoher Geschwindigkeit</b>	Diese AMF ist verletzt, wenn eine Betriebsart aktiv ist, bei der der Roboter mit programmierter Geschwindigkeit verfährt (T2, AUT). Standard-AMF

### 13.8.3 Auswertung der Fahrfreigabe

Ohne Fahrfreigabe kann der Roboter nicht verfahren werden. Die Fahrfreigabe kann aus unterschiedlichen Gründen entzogen sein, beispielsweise wenn im Testbetrieb die Zustimmung nicht erteilt ist oder der NOT-HALT am smartPAD gedrückt ist.

Die AMF für die Fahrfreigabe funktioniert wie ein Sammelsignal für alle konfigurierten Stopp-Bedingungen. Sie kann insbesondere zum Abschalten von Peripheriegeräten verwendet werden. Daher sollte bei Sicherheitsfunktionen, die die Auswertung der Fahrfreigabe enthalten, ein sicherer Ausgang als Reaktion konfiguriert sein. Wird als Reaktion ein Sicherheitshalt eingestellt, kann der Roboter nicht verfahren werden.

AMF	Beschreibung
Fahr freigabe	<p>Diese AMF ist verletzt, wenn die Fahr freigabe aufgrund einer Stopp-Anforderung nicht erteilt ist.</p> <p><b>Hinweis:</b> Diese AMF kann nur mit einem Ausgang als Reaktion sinnvoll verwendet werden.</p> <p>Standard-AMF</p>

**Beispiel**Werkzeug abschalten (Kategorie: **Ausgang**)

Ein Werkzeug (z. B. ein Laser), das an einem Ausgang angeschlossen ist, soll bei Zurücknahme der Fahr freigabe abgeschalten werden. Die Abschaltung soll nur bei verletztem Bedienerschutz erfolgen.

AMF1	AMF2	AMF3	Reaktion
Eingangssignal (Bedienerschutz)	-	Fahr freigabe	Ausgang (Werkzeug)

**13.8.4 Überwachung von sicheren Eingängen****Beschreibung**

Als sichere Eingänge können die Eingänge der diskreten Sicherheitsschnittstelle verwendet werden oder die sicheren Eingänge der Ethernet-Sicherheitsschnittstelle, sofern sie in WorkVisual konfiguriert sind.

(>>> "Sicherheitsschnittstellen" Seite 185)

An den sicheren Eingängen können Sicherheitseinrichtungen, beispielsweise externe NOT-HALT-Geräte oder Schutztüren, angeschlossen werden. Um das zugehörige Eingangssignal auszuwerten, wird die AMF *Eingangssignal* verwendet.



Wenn ein Roboter mit einem Medien-Flansch Touch verwendet wird, können die sicheren Eingänge, an denen Zustimmungs- und Panikschaalter des Medien-Flansches angeschlossen sind, in der AMF verwendet werden.

AMF	Beschreibung
<i>Eingangssignal</i>	<p>Die AMF ist verletzt, wenn der verwendete sichere Eingang LOW-Pegel (Zustand "0") hat.</p> <p>Parametrierbare AMF mit 100 verfügbaren Instanzen</p>

Parameter	Beschreibung
<i>Eingang des Safety-Signals</i>	Sicherer Eingang, der überwacht werden soll

**Beispiel 1**Bedienerschutz (Kategorie: *Bedienerschutz*)

An einem sicheren Eingang wird eine Schutztür angeschlossen. Wenn die Schutztür in der Betriebsart Automatik oder T2 geöffnet wird, soll ein Sicherheitshalt 1 (bahntreu) ausgelöst werden.

AMF1	AMF2	AMF3	Reaktion
<i>Eingangssignal</i>	<i>Betriebsart mit hoher Geschwindigkeit</i>	-	<i>Stopp 1 (bahntreu)</i>

**Beispiel 2**Externer NOT-HALT (Kategorie: *NOT-HALT extern*)

An einem sicheren Eingang wird eine externe NOT-HALT-Einrichtung angegeschlossen. Wenn das externe NOT-HALT-Gerät gedrückt wird, soll ein Sicherheitshalt 1 (bahntreu) ausgelöst werden.

<b>AMF1</b>	<b>AMF2</b>	<b>AMF3</b>	<b>Reaktion</b>
<i>Eingangssignal</i>	-	-	<i>Stopp 1 (bahntreu)</i>

### 13.8.5 Handführen mit Zustimmeinrichtung und Geschwindigkeitsüberwachung

Ein Anwendungsfall im Rahmen der Mensch-Roboter-Kooperation ist das Führen des Roboters von Hand, um z. B. die Punkte einer Bahn zu teachen. Dafür ist ein Handführgerät mit Zustimmeinrichtung erforderlich.

Beim Handführen in der Betriebsart T1 ist die Geschwindigkeit nicht auf 250 mm/s begrenzt. Deswegen muss während des Handführens die Geschwindigkeit sicher überwacht werden. Die während des Handführens maximal zulässige Geschwindigkeit muss durch eine Risikobeurteilung festgelegt werden.

(>>> 13.8.5.3 "Geschwindigkeitsüberwachung während des Handführens"  
Seite 211)



Wenn der Roboter von Hand geführt wird, muss eine NOT-HALT-Einrichtung installiert sein, die der Bediener stets erreichen kann.

#### 13.8.5.1 Überwachung von Zustimmeinrichtungen an Handführgeräten

##### Beschreibung

Die AMF *Zustimmung Handführgerät inaktiv* dient zur Auswertung 3-stufiger Zustimmeinrichtungen. Es können bis zu 3 Zustimmungsschalter und bis zu 3 Panikschalter konfiguriert werden. Es können auch 3-stufige Zustimmeinrichtungen mit nur einem Ausgang ausgewertet werden, die das Paniksignal intern verarbeiten.

Die AMF erfüllt folgende normative Anforderungen und Maßnahmen gegen vorhersehbare Fehlanwendungen:

- Die Zustimmung wird nicht erteilt, wenn der Zustimmungsschalter nach dem Durchdrücken wieder bis zur Mittelstellung losgelassen wird.
- Die Zustimmung wird bei einer Stopp-Anforderung entzogen. Um die Zustimmung danach wieder zu erteilen, muss der Zustimmungsschalter losgelassen und und erneut gedrückt werden.
- Die Zustimmung wird erst 100 ms nach dem Drücken des Zustimmungsschalters erteilt.

Bei Verwendung mehrerer Zustimmungsschalter gilt:

- Werden alle 3 Zustimmungsschalter einer Zustimmeinrichtung gleichzeitig in Mittelstellung gehalten, wird ein Sicherheitshalt 1 ausgelöst.
- Es ist möglich, 2 Zustimmungsschalter einer Zustimmeinrichtung bis zu 15 Sekunden gleichzeitig in Mittelstellung zu halten. Dies erlaubt das Umgreifen von einem Zustimmungsschalter auf einen anderen. Wenn die Zustimmungsschalter länger als 15 Sekunden gleichzeitig in Mittelstellung gehalten werden, löst dies einen Sicherheitshalt 1 aus.
- Werden die Zustimmungsschalter verschiedener Zustimmeinrichtungen gleichzeitig betätigt, z. B. ein Zustimmungsschalter am smartPAD und ein Zustimmungsschalter am Handführgerät, wird ein Sicherheitshalt 1 (bahntreu) ausgelöst.

AMF	Beschreibung
Zustimmung Hand-führgerät inaktiv	<p>Parametrierbare AMF mit einer verfügbaren Instanz</p> <p>Die AMF ist verletzt, wenn alle sicheren Eingänge, an denen ein Zustimmungsschalter angeschlossen ist, LOW-Pegel (Zustand "0") haben oder wenn mindestens einer der sicheren Eingänge, an denen ein Panikschalter angeschlossen ist, LOW-Pegel (Zustand "0") hat.</p>



Wenn ein Roboter mit einem Medien-Flansch Touch verwendet wird, können die sicheren Eingänge, an denen Zustimmungs- und Panikschalter des Medien-Flansches angeschlossen sind, in der AMF verwendet werden.

Parameter	Beschreibung
Zustimmtaster 1 verwendet	Gibt an, ob der Zustimmungsschalter an einem sicheren Eingang angeschlossen ist
Zustimmtaster 2 verwendet	<ul style="list-style-type: none"> <li>■ <b>true</b>: Ein Eingang ist angeschlossen.</li> <li>■ <b>false</b>: Es ist kein Eingang angeschlossen.</li> </ul>
Zustimmtaster 3 verwendet	Default: <b>false</b>
Zustimmtaster 1 Eingangssignal	Sicherer Eingang, an dem der Zustimmungsschalter angeschlossen ist
Zustimmtaster 2 Eingangssignal	Als sichere Eingänge können die Eingänge der diskreten Sicherheitsschnittstelle verwendet werden oder die sicheren Eingänge der Ethernet-Sicherheitsschnittstelle, sofern in WorkVisual konfiguriert.
Zustimmtaster 3 Eingangssignal	(>>> "Sicherheitsschnittstellen" Seite 185)
Paniktaster 1 verwendet	Gibt an, ob der Panikschalter an einem sicheren Eingang angeschlossen ist
Paniktaster 2 verwendet	<ul style="list-style-type: none"> <li>■ <b>true</b>: Ein Eingang ist angeschlossen.</li> <li>■ <b>false</b>: Es ist kein Eingang angeschlossen.</li> </ul>
Paniktaster 3 verwendet	Default: <b>false</b>
Paniktaster 1 Eingangssignal	Sicherer Eingang, an dem der Panikschalter angeschlossen ist
Paniktaster 2 Eingangssignal	Als sichere Eingänge können die Eingänge der diskreten Sicherheitsschnittstelle verwendet werden oder die sicheren Eingänge der Ethernet-Sicherheitsschnittstelle, sofern in WorkVisual konfiguriert.
Paniktaster 3 Eingangssignal	(>>> "Sicherheitsschnittstellen" Seite 185)

### Beispiel

#### Handführen mit Zustimmung (Kategorie: *Zustimmung*)

Ein mit einem Handführgerät ausgestatteter Roboter soll in einem bestimmten Bereich von Hand geführt werden, um die Punkte einer Bahn zu teachen. Der Bereich für das Handführen wird durch einen kartesischen Schutzraum definiert. In diesem Schutzraum darf eine kartesische Geschwindigkeit von 250 mm/s nicht überschritten werden, es sei denn über das Handführgerät wurde die Zustimmung erteilt. Andernfalls wird ein Sicherheitshalt 1 (bahntreu) ausgelöst.

AMF1	AMF2	AMF3	Reaktion
Zustimmung Hand-führgerät inaktiv	Kartesische Schutzraumüberwachung	Kartesische Geschwindigkeitsüberwachung	Stopp 1 (bahntreu)

### 13.8.5.2 Überwachungen während des Handführens

Beschreibung	Die AMF <i>Zustimmung Handführgerät aktiv</i> ermöglicht es, Sicherheitsfunktionen zu realisieren, die während des Handführens mit Zustimmeinrichtung weitere Überwachungen aktivieren, z. B die kartesische Geschwindigkeitsüberwachung.
--------------	---

AMF	Beschreibung
<i>Zustimmung Handführgerät aktiv</i>	<p>Standard-AMF</p> <p>Die AMF ist verletzt, wenn die Zustimmung für das Handführen erteilt ist.</p> <p>Die AMF <i>Zustimmung Handführgerät aktiv</i> stellt den inversen Zustand der AMF <i>Zustimmung Handführgerät inaktiv</i> dar:</p> <ul style="list-style-type: none"> <li>■ Die AMF <i>Zustimmung Handführgerät aktiv</i> ist verletzt, wenn die AMF <i>Zustimmung Handführgerät inaktiv</i> nicht verletzt ist.</li> <li>■ Die AMF <i>Zustimmung Handführgerät aktiv</i> ist nicht verletzt, solange die AMF <i>Zustimmung Handführgerät inaktiv</i> verletzt ist.</li> </ul> <p>Die AMF <i>Zustimmung Handführgerät aktiv</i> berücksichtigt die Zustimmeeinrichtung, die für die AMF <i>Zustimmung Handführgerät inaktiv</i> konfiguriert ist.</p>

Beispiel	Raum- und Geschwindigkeitsüberwachung während des Handführens mit Zustimmeinrichtung (Kategorie: Raumüberwachung, Geschwindigkeitsüberwachung)
	<p>Während des Handführens eines Roboters mit Zustimmeinrichtung darf ein definierter Arbeitsraum nicht verlassen werden. Außerdem soll sich der Roboter während des Handführens mit maximal 600 mm/s bewegen können. Wird bei aktiver Zustimmung der Arbeitsraum verlassen oder wird die Geschwindigkeitsgrenze überschritten, soll ein Sicherheitshalt 1 (bahntreu) ausgeführt werden.</p>

AMF1	AMF2	AMF3	Reaktion
<i>Zustimmung Handführgerät aktiv</i>	<i>Kartesische Arbeitsraumüberwachung</i>	-	<i>Stopp 1 (bahntreu)</i>
<i>Zustimmung Handführgerät aktiv</i>	<i>Kartesische Geschwindigkeitsüberwachung</i>	-	<i>Stopp 1 (bahntreu)</i>

### 13.8.5.3 Geschwindigkeitsüberwachung während des Handführens

Für das Handführen des Roboters muss eine maximal zulässige Geschwindigkeit festgelegt werden, die während des Handführens nicht überschritten werden darf. Der Wert dieser Geschwindigkeit muss durch eine Risikobeurteilung festgelegt werden.

In Zeile 3 der PSM-Tabelle *KUKA PSM* ist eine Sicherheitsfunktion zur sicheren Geschwindigkeitsüberwachung während des Handführens konfiguriert. Die Sicherheitsfunktion berücksichtigt die Zustimmeinrichtung, die für die AMF *Zustimmung Handführgerät inaktiv* konfiguriert ist. Ist die Zustimmung erteilt, wird bei Überschreiten einer Geschwindigkeitsgrenze ein Sicherheitshalt 1 (bahntreu) ausgeführt.

Die verwendete Instanz der AMF *Kartesische Geschwindigkeitsüberwachung* besitzt folgende voreingestellten Parameterwerte:

- *Überwachte Kinematik*: Erste Kinematik
- *Maximale Geschwindigkeit [mm/s]*: 250

Die Parameterwerte können geändert werden. ([>>> 13.8.8.2 "Kartesische Geschwindigkeitsüberwachung definieren"](#) Seite 214)

### 13.8.6 Auswertung der Positionsreferenzierung

#### Beschreibung

Bei der Positionsreferenzierung wird überprüft, ob die gespeicherte Nullstellung des Motors einer Achse (= gespeicherte Justageposition) mit der tatsächlichen mechanischen Nullstellung dieser Achse übereinstimmt.

Solange noch keine Positionsreferenzierung durchgeführt wurde, ist die Sicherheitsintegrität der darauf basierenden Sicherheitsfunktionen eingeschränkt. Dazu gehören sicher überwachte kartesische und achsspezifische Roboterpositionen, kartesische Geschwindigkeiten, Kräfte und Kollisionen.

Mit der AMF *Positionsreferenzierung* kann überprüft werden, ob die Positions-werte aller Achsen referenziert sind.

AMF	Beschreibung
Positionsreferenzierung	Parametrierbare AMF mit 4 verfügbaren Instanzen  Die AMF ist verletzt, wenn die Position mindestens einer Achse der überwachten Kinematik nicht referenziert ist, oder wenn die Positionsreferenzierung mindestens einer Achse fehlgeschlagen ist.

Parameter	Beschreibung
Überwachte Kinematik	Kinematik, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>

#### Beispiel

Status der Positionsreferenzierung überwachen (Kategorie: *Sicherheitshalt*)

Um eine Gefährdung bei einer nicht durchgeführten oder fehlgeschlagenen Positionsreferenzierung zu verhindern, darf ein Roboter mit nicht referenzierten Achsen nur mit einer reduzierten Geschwindigkeit von maximal 250 mm/s verfahren werden.

Um dies zu garantieren, wird der Referenzierungsstatus aller Achsen in den Betriebsarten mit hoher Geschwindigkeit (T2 und AUT) überwacht und ein Sicherheitshalt 1 (bahntreu) ausgelöst, sobald die Position mindestens einer Achse nicht erfolgreich referenziert ist.

AMF1	AMF2	AMF3	Reaktion
Betriebsart mit hoher Geschwindigkeit	-	Positionsreferenzierung	Stopp 1 (bahntreu)

### 13.8.7 Auswertung der Momentenreferenzierung

#### Beschreibung

Bei der Referenzierung der Gelenkmomenten-Sensoren wird überprüft, ob das erwartete externe Drehmoment, das für eine Achse aufgrund des Robotermodells und der angegebenen Lastdaten berechnet werden kann, mit dem basierend auf dem Messwert des Gelenkmomenten-Sensors ermittelten Wert übereinstimmt. Übersteigt die Differenz zwischen diesen Werten einen bestimmten Toleranzwert, ist die Referenzierung der Momentensensoren fehlgeschlagen.

Solange die Momentenreferenzierung nicht erfolgreich durchgeführt wurde, ist die Sicherheitsintegrität der darauf basierenden Sicherheitsfunktionen einge-

schränkt. Dazu gehören Achsmomenten- und TCP-Kraftüberwachung sowie die Kollisionserkennung.

Mit der AMF *Momentenreferenzierung* kann überprüft werden, ob die Gelenkmomenten-Sensoren aller Achsen referenziert sind.

<b>AMF</b>	<b>Beschreibung</b>
<i>Momentenreferenzierung</i>	<p>Parametrierbare AMF mit 4 verfügbaren Instanzen</p> <p>Die AMF ist verletzt, wenn der Gelenkmomenten-Sensor von mindestens einer Achse der überwachten Kinematik nicht referenziert ist, oder wenn die Referenzierung von mindestens einem Gelenkmomenten-Sensor fehlgeschlagen ist.</p>

<b>Parameter</b>	<b>Beschreibung</b>
<i>Überwachte Kinematik</i>	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>

- Beispiel** Status der Momentenreferenzierung überwachen (Kategorie: *Sicherheitshalt*)
- Für eine Station ist eine sichere Kollisionserkennung konfiguriert. Wird ein Moment von mehr als 20 Nm in mindestens einer Achse des Roboters erkannt, wird ein Sicherheitshalt 0 ausgelöst. Da die Sicherheitsintegrität dieser Funktion nur bei erfolgreich referenzierten Gelenkmomenten-Sensoren gewährleistet ist, muss gleichzeitig der Referenzierungsstatus der Sensoren überwacht werden. Sobald mindestens ein Gelenkmomenten-Sensor nicht oder fehlerhaft referenziert wurde, soll in Betriebsarten mit hoher Geschwindigkeit (T2 und AUT) ein Sicherheitshalt 1 (bahntreu) ausgelöst werden.

<b>AMF1</b>	<b>AMF2</b>	<b>AMF3</b>	<b>Reaktion</b>
<i>Kollisionserkennung</i>	-	-	<i>Stopp 0</i>
<i>Betriebsart mit hoher Geschwindigkeit</i>	-	<i>Momentenreferenzierung</i>	<i>Stopp 1 (bahntreu)</i>

### 13.8.8 Geschwindigkeitsüberwachungen

Eine bewegte Kinematik stellt immer eine Gefahr für Menschen dar, die sich in ihrer Nähe befinden. Zum Schutz von Personen kann es nötig sein, dass eine definierte Geschwindigkeit nicht überschritten wird, um beispielsweise rechtzeitiges Ausweichen zu ermöglichen. Dafür muss die Geschwindigkeit dauerhaft überwacht werden.

Es können die achsspezifischen Geschwindigkeiten und die kartesische Geschwindigkeit einer Kinematik überwacht werden.

#### 13.8.8.1 Achsspezifische Geschwindigkeitsüberwachung definieren

Um eine achsspezifische Geschwindigkeitsüberwachung zu definieren, wird die AMF *Achsgeschwindigkeitsüberwachung* verwendet.

AMF	Beschreibung
Achsgeschwindigkeitsüberwachung	<p>Parametrierbare AMF mit 16 verfügbaren Instanzen</p> <p>Die AMF ist verletzt, wenn der Betrag der Geschwindigkeit der überwachten Achse der überwachten Kinematik die definierte Geschwindigkeitsgrenze überschreitet.</p>

Parameter	Beschreibung
Überwachte Kinematik	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Mobile Plattform</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Überwachte Achse	<p>Achse der Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Achse1 ... Achse16</b></li> </ul> <p>Bei einem LBR iiwa werden <b>Achse1 ... Achse7</b> verwendet.</p> <p>Bei einer mobilen Plattform sind die Achsen wie folgt zugeordnet:</p> <ul style="list-style-type: none"> <li>■ <b>Achse1:</b> Antrieb vorne links</li> <li>■ <b>Achse2:</b> Antrieb vorne rechts</li> <li>■ <b>Achse3:</b> Antrieb hinten links</li> <li>■ <b>Achse4:</b> Antrieb hinten rechts</li> </ul>
Maximale Geschwindigkeit [%/s]	<p>Maximal zulässige Geschwindigkeit, mit der sich die überwachte Achse in positiver und negativer Drehrichtung bewegen darf</p> <ul style="list-style-type: none"> <li>■ <b>1 ... 500 °/s</b></li> </ul>

### 13.8.8.2 Kartesische Geschwindigkeitsüberwachung definieren

#### Beschreibung

Um eine kartesische Geschwindigkeitsüberwachung zu definieren, wird die AMF *Kartesische Geschwindigkeitsüberwachung* verwendet.

- Bei einem Roboter wird die translatorische kartesische Geschwindigkeit an allen Achs-Mittelpunkten sowie am Roboterflansch überwacht.

Wenn ein sicherheitsgerichtetes Werkzeug auf der Robotersteuerung aktiv ist, wird zusätzlich die Geschwindigkeit an den Mittelpunkten der Kugeln überwacht, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist.

(>>> 9.3.7 "Sicherheitsgerichtetes Werkzeug" Seite 148)



Es wird nicht die gesamte Struktur von Roboter und Werkzeug gegen die Verletzung einer Geschwindigkeitsgrenze überwacht, sondern nur die Überwachungskugeln. Insbesondere bei ausladenden Werkzeugen und Werkstücken sind die Überwachungskugeln des sicherheitsgerichteten Werkzeugs so zu planen und zu konfigurieren, dass die Sicherheitsintegrität der Geschwindigkeitsüberwachung gewährleistet ist.

- Bei einer mobilen Plattform wird die translatorische kartesische Geschwindigkeit an 4 definierten Punkten in der Nähe der Eckpunkte der Plattform überwacht. Es werden nur Geschwindigkeitsanteile innerhalb der Ebene der Plattform berücksichtigt.

 Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) ist zu berücksichtigen, dass die kartesische Relativgeschwindigkeit der Überwachungskugeln zur Trägerkinematik überwacht wird und nicht die Absolutgeschwindigkeit der Überwachungskugeln im Raum.

AMF	Beschreibung
Kartesische Geschwindigkeitsüberwachung	<p>Parametrierbare AMF mit 30 verfügbaren Instanzen</p> <p>Die AMF ist verletzt, wenn die translatorische kartesische Geschwindigkeit an mindestens einem Punkt der überwachten Kinematik die definierte Grenze überschreitet.</p> <p>Zusätzlich ist die AMF in folgenden Fällen verletzt:</p> <ul style="list-style-type: none"> <li>■ Eine Achse ist nicht justiert.</li> <li>■ Die Referenzierung einer justierten Achse ist fehlgeschlagen.</li> </ul> <p><b>Hinweis:</b> Bei einer Verletzung der AMF durch Justageverlust, kann der Roboter nur durch einen Wechsel in die Betriebsart KRF wieder verfahren und justiert werden.</p>

Parameter	Beschreibung
Überwachte Kinematik	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Mobile Plattform</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Maximale Geschwindigkeit [mm/s]	<p>Maximal zulässige kartesische Geschwindigkeit, die an keinem der überwachten Punkte überschritten werden darf</p> <ul style="list-style-type: none"> <li>■ <b>1 ... 10 000 mm/s</b></li> </ul>

**Beispiel** Kategorie: *Geschwindigkeitsüberwachung*

Wenn ein kartesischer Arbeitsraum verletzt wird, darf die kartesische Geschwindigkeit des Roboters maximal 300 mm/s betragen. Wird diese Geschwindigkeit überschritten, soll ein Sicherheitshalt 1 ausgelöst werden.

AMF1	AMF2	AMF3	Reaktion
Kartesische Arbeitsraumüberwachung	-	Kartesische Geschwindigkeitsüberwachung	Stopp 1

### 13.8.8.3 Richtungsabhängige Überwachung der kartesischen Geschwindigkeit

**Beschreibung** Mit der AMF *Werkzeugbezogene Geschwindigkeitskomponente* wird geprüft, ob die kartesische translatorische Geschwindigkeit in einer bestimmten Richtung unter einem konfigurierbaren Grenzwert liegt.

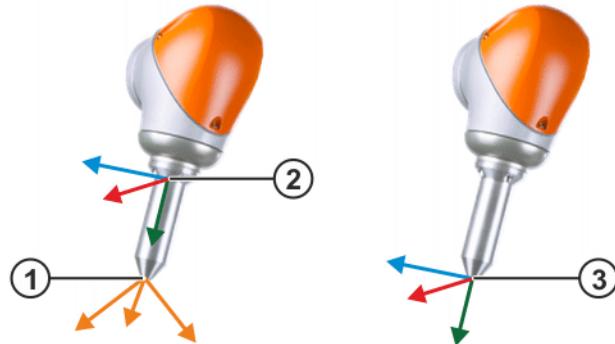
Mit der AMF kann beispielsweise sichergestellt werden, dass an einem spitzen Werkzeug die Geschwindigkeit in Stoßrichtung nicht zu hoch wird. Außerdem kann die AMF zur Überwachung der Bewegungsrichtung genutzt werden.

Bei der Konfiguration der AMF muss die überwachte Kinematik angegeben werden. Die AMF kann für Roboter (**Erste Kinematik**) und mobile Plattformen (**Zweite Kinematik**) angewendet werden.

Die AMF überwacht die Geschwindigkeit an einem Bezugs-Frame am sicherheitsgerichteten Werkzeug. Position und Orientierung des Bezugs-Frames werden in den Eigenschaften des Werkzeugs durch sicherheitsgerichtete Frames festgelegt.

Folgende Sicherheitsparameter stehen hierfür in den Eigenschaften des sicherheitsgerichteten Werkzeugs zur Verfügung:

- **Punkt für die werkzeugbezogene Geschwindigkeit:** Der hier eingestellte sicherheitsgerichtete Frame bestimmt die Position des Bezugs-Frames.
- **Orientierung für die werkzeugbezogene Geschwindigkeit:** Der hier eingestellte sicherheitsgerichtete Frame bestimmt die Orientierung des Bezugs-Frames.

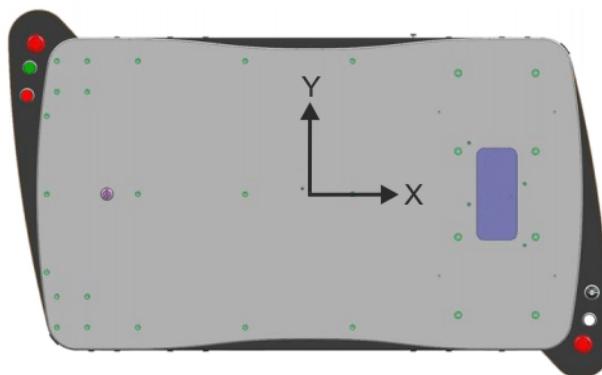


**Abb. 13-7: Bezugs-Frame für werkzeugbezogene Geschwindigkeit**

- 1 Punkt für die werkzeugbezogene Geschwindigkeit
- 2 Orientierung für die werkzeugbezogene Geschwindigkeit
- 3 Position und Orientierung des Bezugs-Frames für die werkzeugbezogene Geschwindigkeit (Kombination aus 1 und 2)

(>>> 9.3.7.1 "Sicherheitsgerichtetes Werkzeug definieren" Seite 149)

Ist die überwachte Kinematik eine mobile Plattform wird für die Festlegung des Bezugs-Frames angenommen, dass das sicherheitsgerichtete Werkzeug am Mittelpunkt der Plattform befestigt ist.



**Abb. 13-8: Bezugs-Frame im Mittelpunkt der mobilen Plattform**

Ist kein sicherheitsgerichtetes Werkzeug aktiv, wird abhängig von der überwachten Kinematik folgender Bezugs-Frame verwendet:

- **Roboter (Erste Kinematik):** Frame im Roboterflansch-Mittelpunkt
- **Mobile Plattform (Zweite Kinematik):** Frame im Mittelpunkt der Plattform

Überwacht wird der Anteil des Geschwindigkeitsvektors in einer bestimmten Richtung des Bezugs-Frames. Bei der Konfiguration der AMF wird diese Richtung als Komponente des Geschwindigkeitsvektors im Koordinatensystem des Bezugs-Frames angegeben. Dabei kann eine der insgesamt 6 Kompo-

nenten des Koordinatensystems (X-, Y- und Z-Komponente, jeweils in positiver und negativer Richtung) gewählt werden.

Außerdem wird die maximale Geschwindigkeit festgelegt, die die gewählte Komponente des Geschwindigkeitsvektors nicht überschreiten darf.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) ist zu berücksichtigen, dass die Geschwindigkeit des überwachten Punkts relativ zur Trägerkinematik überwacht wird und nicht die absolute Geschwindigkeit im Raum.

<b>AMF</b>	<b>Beschreibung</b>
<i>Werkzeugbezogene Geschwindigkeitskomponente</i>	<p>Parametrierbare AMF mit 30 verfügbaren Instanzen</p> <p>Die AMF ist verletzt, wenn die konfigurierte Komponente des Geschwindigkeitsvektors im Koordinatensystem des Bezugs-Frames der überwachten Kinematik den festgelegten Maximalwert überschreitet.</p> <p>Bei einem LBR iiwa ist die AMF zusätzlich in folgenden Fällen verletzt:</p> <ul style="list-style-type: none"> <li>■ Eine Achse ist nicht justiert.</li> <li>■ Die Referenzierung einer justierten Achse ist fehlgeschlagen.</li> </ul>

<b>Parameter</b>	<b>Beschreibung</b>
<i>Überwachte Kinematik</i>	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Mobile Plattform</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
<i>Maximale Geschwindigkeit [mm/s]</i>	<p>Maximale kartesische Geschwindigkeit für die überwachte Komponente des Geschwindigkeitsvektors</p> <ul style="list-style-type: none"> <li>■ <b>1 ... 10000 mm/s</b></li> </ul> <p><b>Hinweis:</b> Bei der Wahl der maximalen Geschwindigkeit ist zu beachten, dass insbesondere bei hochdynamischen Bewegungen aufgrund von Überschwingen geringe Geschwindigkeiten entgegen der kommandierten Bewegungsrichtung auftreten können. Daher sollte die Maximalgeschwindigkeit nicht zu niedrig gewählt werden.</p>
<i>Komponente des Geschwindigkeitsvektors</i>	<p>Komponente des Geschwindigkeitsvektors, die überwacht wird (Richtung der Überwachung)</p> <ul style="list-style-type: none"> <li>■ <b>X positiv oder X negativ</b></li> <li>■ <b>Y positiv oder Y negativ</b></li> <li>■ <b>Z positiv oder Z negativ</b></li> </ul>

### Beispiel 1

Ein spitzes Werkzeug darf in Stoßrichtung mit maximal 25 mm/s bewegt werden. Dafür wird das Werkzeug als sicherheitsgerichtet markiert und die Werkzeugspitze als sicherheitsgerichteter Frame angelegt. Dieser Frame wird verwendet, um die Position und die Orientierung des Bezugs-Frames in den Eigenschaften des sicherheitsgerichteten Werkzeugs festzulegen.

Eine Instanz der AMF *Werkzeugbezogene Geschwindigkeitskomponente* wird so konfiguriert, dass die positive Z-Komponente des Geschwindigkeitsvektors im Koordinatensystem der Werkzeugspitze den Wert von 25 mm/s nicht übersteigen darf. Dafür werden für die verwendete Instanz folgende Parameter eingestellt:

- **Überwachte Kinematik:** Erste Kinematik
- **Maximale Geschwindigkeit [mm/s]:** 25

- Komponente des Geschwindigkeitsvektors: Z positiv

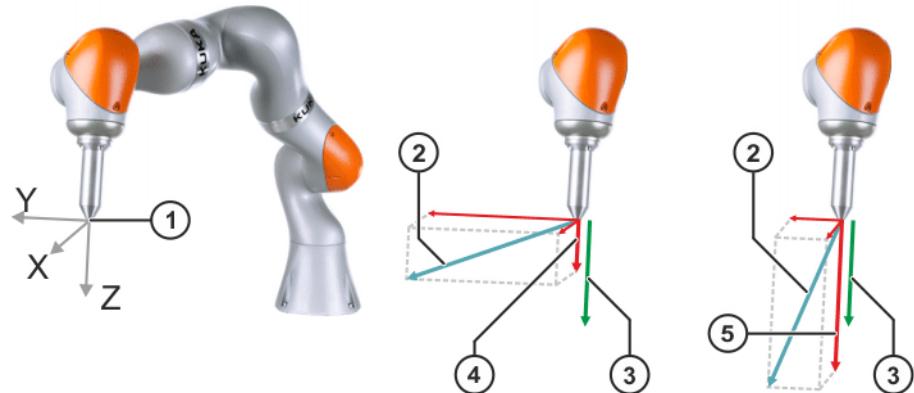


Abb. 13-9: Geschwindigkeitsüberwachung in Werkzeug-Stoßrichtung

Pos.	Beschreibung
1	Bezugs-Frame für die werkzeugbezogene Geschwindigkeitskomponente
2	Geschwindigkeitsvektor der kartesischen translatorischen Geschwindigkeit
3	Maximal zulässige Geschwindigkeit für die positive Z-Komponente des Geschwindigkeitsvektors
4	Positive Z-Komponente des Geschwindigkeitsvektors Die Geschwindigkeit liegt unter der zulässigen Maximalgeschwindigkeit; die AMF ist nicht verletzt.
5	Positive Z-Komponente des Geschwindigkeitsvektors Die Geschwindigkeit liegt über der zulässigen Maximalgeschwindigkeit; die AMF ist verletzt.

Die mit der AMF konfigurierte Sicherheitsfunktion überwacht die positive Z-Komponente des Geschwindigkeitsvektors. Wird in der Betriebsart Automatik die Maximalgeschwindigkeit von 25 mm/s von der überwachten Komponente überschritten, soll ein Sicherheitshalt 1 (bahntreu) ausgeführt werden.

Kategorie: Geschwindigkeitsüberwachung

AMF1	AMF2	AMF3	Reaktion
Betriebsart Automatik	Werkzeugbezogene Geschwindigkeitskomponente (1) Erste Kinematik	-	Stopp 1 (bahntreu)

### Beispiel 2

Um die Abmessungen des Schutzraums einer mobilen Plattform nach hinten und auf beiden Seiten möglichst klein halten zu können, muss die Bewegungsrichtung der Plattform derart überwacht werden, dass nur Vorwärtsbewegungen mit hoher Geschwindigkeit durchgeführt werden können.

Konfiguration:

- Es werden 3 Instanzen der AMF Werkzeugbezogene Geschwindigkeitskomponente benötigt. Bezugs-Frame ist jeweils der Mittelpunkt der Plattform.
- Die Bewegung nach links und rechts soll mit maximal 50 mm/s durchgeführt werden. Dafür werden die positive und negative Y-Komponente des Geschwindigkeitsvektors auf 50 mm/s begrenzt.

- Die Rückwärtsbewegung soll mit maximal 20 mm/s durchgeführt werden. Dafür wird die negative X-Komponente des Geschwindigkeitsvektors auf 20 mm/s begrenzt.

Parametrierung der konfigurierten Instanzen:

- Instanz 1:
  - Überwachte Kinematik: Zweite Kinematik
  - Maximale Geschwindigkeit [mm/s]: 50
  - Komponente des Geschwindigkeitsvektors: Y positiv
- Instanz 2:
  - Überwachte Kinematik: Zweite Kinematik
  - Maximale Geschwindigkeit [mm/s]: 50
  - Komponente des Geschwindigkeitsvektors: Y negativ
- Instanz 3:
  - Überwachte Kinematik: Zweite Kinematik
  - Maximale Geschwindigkeit [mm/s]: 20
  - Komponente des Geschwindigkeitsvektors: X negativ

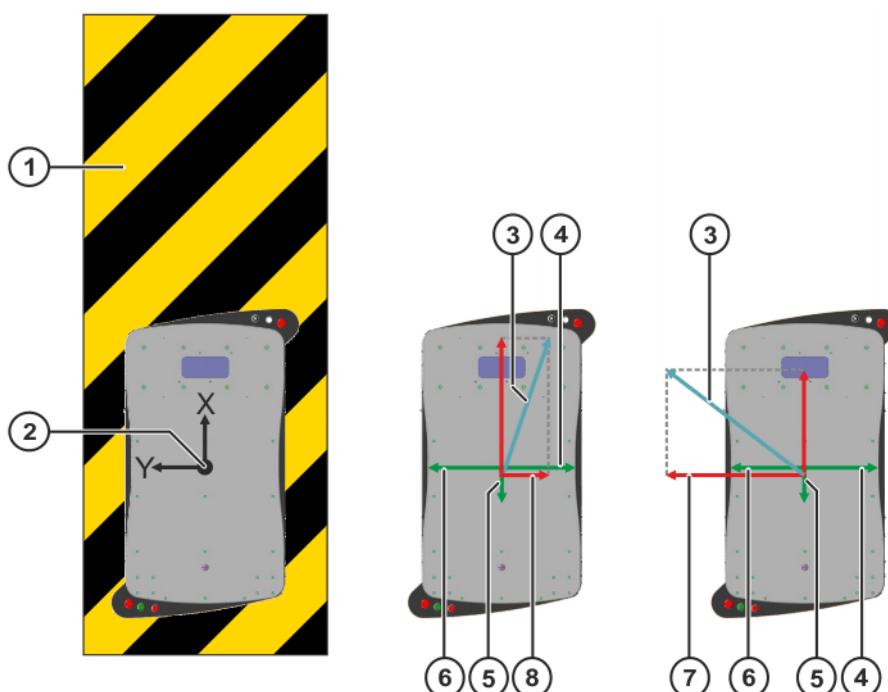


Abb. 13-10: Schutzraum-Begrenzung durch Geschwindigkeitsüberwachung in Bewegungsrichtung

Pos.	Beschreibung
1	Ungefähr Abmessungen des gewünschten Schutzraums
2	Bezugs-Frame für die werkzeugbezogene Geschwindigkeitskomponente
3	Geschwindigkeitsvektor der kartesischen translatorischen Geschwindigkeit
4	Maximal zulässige Geschwindigkeit für die negative Y-Komponente des Geschwindigkeitsvektors
5	Maximal zulässige Geschwindigkeit für die negative X-Komponente des Geschwindigkeitsvektors
6	Maximal zulässige Geschwindigkeit für die positive Y-Komponente des Geschwindigkeitsvektors

Pos.	Beschreibung
7	Positive Y-Komponente des Geschwindigkeitsvektors Die Geschwindigkeit liegt über der zulässigen Maximalgeschwindigkeit von Instanz 1; die AMF ist verletzt.
8	Negative Y-Komponente des Geschwindigkeitsvektors Die Geschwindigkeit liegt unter der zulässigen Maximalgeschwindigkeit; keine der 3 Instanzen der AMF ist verletzt.

Es werden 3 Sicherheitsfunktionen konfiguriert, die jeweils eine der 3 Instanzen verwenden. Wird in der Betriebsart Automatik die jeweils konfigurierte Maximalgeschwindigkeit in mindestens einer der 3 überwachten Komponenten überschritten, soll ein Sicherheitshalt 1 (bahntreu) ausgeführt werden.

Kategorie: *Geschwindigkeitsüberwachung*

AMF1	AMF2	AMF3	Reaktion
<i>Betriebsart Automatik</i>	<i>Werkzeugbezogene Geschwindigkeitskomponente (1)</i> <i>Zweite Kinematik</i>	-	<i>Stopp 1 (bahntreu)</i>
<i>Betriebsart Automatik</i>	<i>Werkzeugbezogene Geschwindigkeitskomponente (2)</i> <i>Zweite Kinematik</i>	-	<i>Stopp 1 (bahntreu)</i>
<i>Betriebsart Automatik</i>	<i>Werkzeugbezogene Geschwindigkeitskomponente (3)</i> <i>Zweite Kinematik</i>	-	<i>Stopp 1 (bahntreu)</i>

### 13.8.9 Überwachungsräume

#### Beschreibung

Das Umfeld des Roboters kann in Bereiche eingeteilt werden, in denen er sich zur Ausführung der Applikation aufhalten muss, und Bereiche, in denen er sich nicht oder nur unter bestimmten Bedingungen befinden soll. Es muss dann ständig geprüft werden, ob sich Teile der Roboterstruktur inner- oder außerhalb eines solchen Überwachungsraums befinden.

Ein Überwachungsraum kann als kartesischer Quader oder über einzelne Achsbereiche definiert werden.

Ein kartesischer Überwachungsraum kann als Arbeitsraum, in dem sich der Roboter aufhalten muss, oder als Schutzraum, in den er nicht eindringen darf, konfiguriert werden.

Über die Verknüpfung mit anderen Atomic Monitoring Functions können weitere Bedingungen festgelegt werden, die bei Verletzung eines Überwachungsraums erfüllt sein müssen. So kann ein Überwachungsraum beispielsweise über einen sicheren Eingang aktivierbar sein oder nur im Automatikbetrieb gelten.



Wenn der Roboter einen sicher überwachten Raum verletzt hat und von der Sicherheitssteuerung gestoppt wird, kann er in der Betriebsart KRF aus dem verletzten Bereich herausfahren werden.  
(>> 6.9 "Betriebsart KRF – Roboter kontrolliert freifahren" Seite 82)

#### Kugeln am Roboter

Um ausgewählte Punkte am Roboter sind Kugeln modelliert, die den Roboter einhüllen und sich mit dem Roboter bewegen. Diese Kugeln sind fest vorgegeben und werden defaultmäßig gegen die Grenzen aktiverter kartesischer Überwachungsräume überwacht.

Die Zentren und Radien der überwachten Kugeln sind in den Maschinendaten des Roboters festgelegt. Für jede Roboterachse, für den Roboterfuß und den Roboterflansch ist je eine Kugel definiert. Das Kugelzentrum liegt jeweils auf dem Achs-, Roboterfuß- oder Roboterflansch-Mittelpunkt.

Die Abmessungen der überwachten Kugeln variieren nach Robotertyp und verwendetem Medien-Flansch:

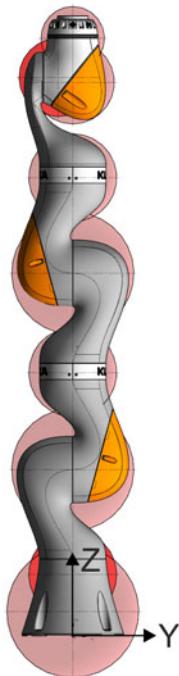
- $r$  = Kugelradius
- $z, y$  = Kugelmittelpunkt bezogen auf das Roboterfuß-Koordinatensystem

#### **Variante 1: LBR iiwa 7 R800 mit Medien-Flansch Touch**

	Fuß	A1	A2	A3	A4	A5	A6	A7	Flansch
r [mm]	135	90	125	90	125	90	80	85	65
z [mm]	50	90	340	538	740	935	1140	1130	1240
y [mm]							-30		

#### **Variante 2: LBR iiwa 7 R800 mit Medien-Flansch (alle Varianten außer Medien-Flansch Touch)**

	Fuß	A1	A2	A3	A4	A5	A6	A7	Flansch
r [mm]	135	90	125	90	125	90	80	85	65
z [mm]	50	90	340	538	740	935	1140	1130	1220
y [mm]							-30		



**Abb. 13-11: Kugeln am LBR iiwa 7 R800 (Variante 2)**

#### **Variante 3: LBR iiwa 14 R820 mit Medien-Flansch Touch**

	Fuß	A1	A2	A3	A4	A5	A6	A7	Flansch
r [mm]	150	100	140	90	131	90	80	85	65
z [mm]	50	160	360	580	780	980	1180	1170	1280
y [mm]							-30		

#### **Variante 4: LBR iiwa 14 R820 mit Medien-Flansch (alle Varianten außer Medien-Flansch Touch)**

	Fuß	A1	A2	A3	A4	A5	A6	A7	Flansch
r [mm]	150	100	140	90	131	90	80	85	65
z [mm]	50	160	360	580	780	980	1180	1170	1260
y [mm]							-30		

### Kugeln am Werkzeug

Wenn ein sicherheitsgerichtetes Werkzeug auf der Robotersteuerung aktiv ist, werden neben den Kugeln am Roboter defaultmäßig auch die Kugeln überwacht, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist.

(>>> 9.3.7 "Sicherheitsgerichtetes Werkzeug" Seite 148)



Es wird nicht die gesamte Struktur von Roboter und Werkzeug gegen die Verletzung eines Raumes überwacht, sondern nur die Überwachungskugeln. Insbesondere bei ausladenden Werkzeugen und Werkstücken sind die Überwachungskugeln des sicherheitsgerichteten Werkzeugs so zu planen und zu konfigurieren, dass die Sicherheitsintegrität von Arbeits- und Schutzraumüberwachung gewährleistet ist.

### Auswahl Überwachungskugeln

Es ist nicht in jedem Anwendungsfall notwendig oder sinnvoll, alle Roboter- und Werkzeugkugeln in die kartesische Raumüberwachung einzubeziehen.

**Beispiel:** Wenn das Eindringen eines Werkzeugs in einen Schutzraum dazu dienen soll, weitere Überwachungen zu aktivieren, sind nur die Werkzeugkugeln zu überwachen.

Die zu überwachende Struktur kann bei der Konfiguration kartesischer Überwachungsräume ausgewählt werden:

- Roboter und Werkzeug (Default)
- Nur Werkzeug
- Nur Roboter

### Anhalteweg

Wenn der Roboter durch eine Überwachung gestoppt wird, benötigt er einen Anhalteweg bis zum Stillstand.

Der Anhalteweg ist im Wesentlichen von folgenden Faktoren abhängig:

- Robotertyp
- Geschwindigkeit des Roboters
- Position der Roboterachsen
- Traglast



**WARNING** Der Anhalteweg des Roboters hängt im Wesentlichen von der Dynamik des Robotertyps ab. Je nach Robotertyp beschleunigt der Roboter im Fehlerfall innerhalb der Reaktionszeit der Überwachungsfunktionen unterschiedlich stark. Dies beeinflusst den tatsächlichen Anhalteweg.

Im Rahmen der Sicherheitsbetrachtung muss dieser Aspekt vom Systemintegrator bei der Parametrierung der Überwachungsfunktionen berücksichtigt werden.

### 13.8.9.1 Kartesische Arbeitsräume definieren

#### Beschreibung

Ein kartesischer Arbeitsraum wird als Quader definiert, dessen Lage und Orientierung im Raum relativ zum Welt-Koordinatensystem festgelegt wird.

Die Überwachungskugeln werden gegen die Grenzen aktiverter kartesischer Arbeitsräume überwacht und müssen sich innerhalb dieser Arbeitsräume bewegen.

Um einen kartesischen Arbeitsraum zu definieren, wird die AMF *Kartesische Arbeitsraumüberwachung* verwendet. Die AMF ist verletzt, sobald sich eine der überwachten Kugeln nicht mehr vollständig innerhalb des definierten Arbeitsraums befindet.

Zusätzlich ist die AMF in folgenden Fällen verletzt:

- Eine Achse ist nicht justiert.
- Die Referenzierung einer justierten Achse ist fehlgeschlagen.

Von dieser parametrierbaren AMF sind 8 Instanzen verfügbar, d. h. es können maximal 8 kartesische Arbeitsräume konfiguriert werden.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) ist zu berücksichtigen, dass die die Lage und Orientierung des Überwachungsraums auf das Welt-Koordinatensystem bezogen ist und damit relativ zur Ausrichtung der Basis der überwachten Kinematik definiert ist. Daher wird bei einer Positions- oder Neigungsänderung der Trägerkinematik der Überwachungsraum mitbewegt.

Parameter	Beschreibung
<i>Überwachte Kinematik</i>	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
<i>Überwachte Struktur</i>	<p>Struktur, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <i>Roboter und Werkzeug:</i> Die Kugeln am Roboter und die Kugeln, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist, werden überwacht. (Default)</li> <li>■ <i>Roboter:</i> Die Kugeln am Roboter werden überwacht.</li> <li>■ <i>Werkzeug:</i> Die Kugeln, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist, werden überwacht.</li> </ul> <p><b>Hinweis:</b> Wenn kein sicherheitsgerichtetes Werkzeug konfiguriert ist und das Werkzeug als zu überwachende Struktur ausgewählt ist, wird die Kugel am Roboterflansch überwacht. (<a href="#">&gt;&gt;&gt; "Kugeln am Roboter"</a> Seite 220)</p>

Eine Ecke des Quaders wird relativ zum Welt-Koordinatensystem definiert. Sie ist der Ursprung des Arbeitsraums und wird durch folgende Parameter definiert:

Parameter	Beschreibung
<b>X, Y, Z [mm]</b>	<p>Verschiebung des Ursprungs des Arbeitsraums entlang der X-, Y- und Z-Achse des Welt-Koordinatensystems</p> <ul style="list-style-type: none"> <li>■ <b>-100 000 mm ... +100 000 mm</b></li> </ul>
<b>A, B, C [°]</b>	<p>Orientierung des Ursprungs des Arbeitsraums um die Achsen des Welt-Koordinatensystems, angegeben durch die Rotationswinkel A, B, C</p> <ul style="list-style-type: none"> <li>■ <b>0° ... 359°</b></li> </ul>

Ausgehend vom so festgelegten Ursprung wird die Größe des Arbeitsraums entlang der Koordinatenachsen bestimmt:

Parameter	Beschreibung
<b>Länge [mm]</b>	Länge des Arbeitsraums (= Strecke entlang der positiven X-Achse des Ursprungs) ■ <b>0 mm ... 100 000 mm</b>
<b>Breite [mm]</b>	Breite des Arbeitsraums (= Strecke entlang der positiven Y-Achse des Ursprungs) ■ <b>0 mm ... 100 000 mm</b>
<b>Höhe [mm]</b>	Höhe des Arbeitsraums (= Strecke entlang der positiven Z-Achse des Ursprungs) ■ <b>0 mm ... 100 000 mm</b>



Erst wenn sich nach Verletzung eines kartesischen Arbeitsraumes alle überwachten Kugeln wieder innerhalb der Arbeitsraumgrenzen befinden und einen Mindestabstand von 10 mm zu diesen haben, wird die Verletzung aufgehoben.

### Beispiel

Die Abbildung zeigt ein Beispiel für einen kartesischen Arbeitsraum. Dessen Ursprung ist bezogen auf das Welt-Koordinatensystem in negativer X- und Y-Richtung verschoben.

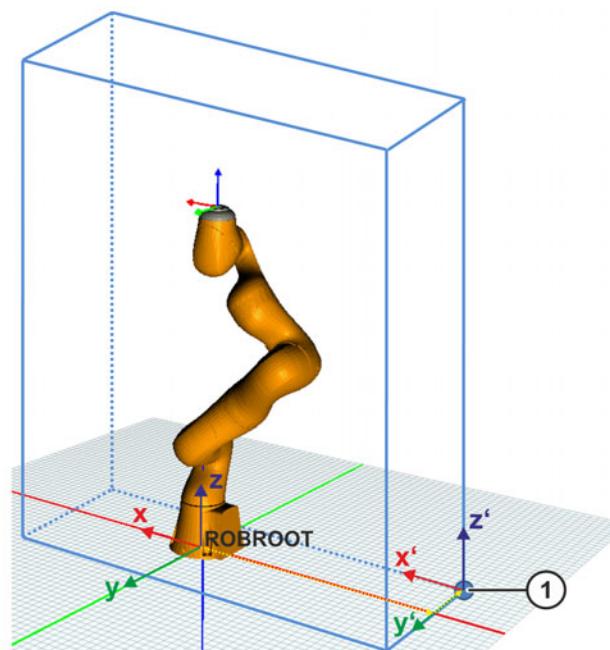


Abb. 13-12: Beispiel eines kartesischen Arbeitsraums

1 Ursprung des Arbeitsraums

#### 13.8.9.2 Kartesische Schutzräume definieren

##### Beschreibung

Ein kartesischer Schutzraum wird als Quader definiert, dessen Lage und Orientierung im Raum relativ zum Welt-Koordinatensystem festgelegt wird.

Die Überwachungskugeln werden gegen die Grenzen aktiverter kartesischer Schutzräume überwacht und müssen sich außerhalb dieser Schutzräume bewegen.

Um einen kartesischen Schutzraum zu definieren, wird die AMF *Kartesische Schutzraumüberwachung* verwendet. Die AMF ist verletzt, sobald sich eine

der überwachten Kugeln nicht mehr vollständig außerhalb des definierten Schutzraums befindet.

Zusätzlich ist die AMF in folgenden Fällen verletzt:

- Eine Achse ist nicht justiert.
- Die Referenzierung einer justierten Achse ist fehlgeschlagen.

Von dieser parametierbaren AMF sind 8 Instanzen verfügbar, d. h. es können maximal 8 kartesische Schutzzräume konfiguriert werden.

Wenn ein sehr schmaler Schutzraum konfiguriert ist, kann es vorkommen, dass der Roboter in den Schutzraum fährt und ihn wieder verlässt, ohne dass die Raumverletzung erkannt wird. Mögliche Ursache: Aufgrund einer sehr hohen Werkzeug-Geschwindigkeit ist der Schutzraum nur in einem sehr kurzen Zeitintervall verletzt.

Angenommen es sind folgende Minimalwerte konfiguriert:

- Radius Werkzeugkugeln: 25 mm
- Dicke Schutzraum: 0 mm

In diesem Fall wären Werkzeug-Geschwindigkeiten von mehr als 4 m/s erforderlich, damit der Schutzraum unerkannt durchfahren wird.

Folgende Maßnahmen werden empfohlen, um zu verhindern, dass Schutzzräume unerkannt durchfahren werden:

- Kartesische Geschwindigkeitsüberwachung von 4 m/s konfigurieren.
- ODER: Bei der Konfiguration des Schutzraums ausreichend hohe Werte für Länge, Breite und Höhe des Schutzraums wählen.
- ODER: Bei der Konfiguration der Werkzeugkugeln ausreichend hohe Werte für den Radius wählen.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) ist zu berücksichtigen, dass die die Lage und Orientierung des Überrachungsraums auf das Welt-Koordinatensystem bezogen ist und damit relativ zur Ausrichtung der Basis der überwachten Kinematik definiert ist. Daher wird bei einer Positions- oder Neigungsänderung der Trägerkinematik der Überwachungsraum mitbewegt.

Parameter	Beschreibung
Überwachte Kinematik	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Überwachte Struktur	<p>Struktur, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <i>Roboter und Werkzeug:</i> Die Kugeln am Roboter und die Kugeln, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist, werden überwacht. (Default)</li> <li>■ <i>Roboter:</i> Die Kugeln am Roboter werden überwacht.</li> <li>■ <i>Werkzeug:</i> Die Kugeln, mit denen das sicherheitsgerichtete Werkzeug konfiguriert ist, werden überwacht.</li> </ul> <p><b>Hinweis:</b> Wenn kein sicherheitsgerichtetes Werkzeug konfiguriert ist und das Werkzeug als zu überwachende Struktur ausgewählt ist, wird die Kugel am Roboterflansch überwacht. (<a href="#">&gt;&gt;&gt; "Kugeln am Roboter"</a> Seite 220)</p>

Eine Ecke des Quaders wird relativ zum Welt-Koordinatensystem definiert. Sie ist der Ursprung des Schutzraums und wird durch folgende Parameter definiert:

Parameter	Beschreibung
X, Y, Z [mm]	Verschiebung des Ursprungs des Schutzraums entlang der X-, Y- und Z-Achse des Welt-Koordinatensystems. ■ -100 000 mm ... +100 000 mm
A, B, C [°]	Orientierung des Ursprungs des Schutzraums um die Achsen des Welt-Koordinatensystems, angegeben durch die Rotationswinkel A, B, C ■ 0° ... 359°

Ausgehend vom so festgelegten Ursprung wird die Größe des Schutzraums entlang der Koordinatenachsen bestimmt:

Parameter	Beschreibung
Länge [mm]	Länge des Schutzraums (= Strecke entlang der positiven X-Achse des Ursprungs) ■ 0 mm ... 100 000 mm
Breite [mm]	Breite des Schutzraums (= Strecke entlang der positiven Y-Achse des Ursprungs) ■ 0 mm ... 100 000 mm
Höhe [mm]	Höhe des Schutzraums (= Strecke entlang der positiven Z-Achse des Ursprungs) ■ 0 mm ... 100 000 mm



Erst wenn sich nach Verletzung eines kartesischen Schutzraumes alle überwachten Kugeln wieder außerhalb der Schutzraumgrenzen befinden und einen Mindestabstand von 10 mm zu diesen haben, wird die Verletzung aufgehoben.

### Beispiel

Die Abbildung zeigt ein Beispiel für einen kartesischen Schutzraum. Dessen Ursprung ist bezogen auf das Welt-Koordinatensystems in negativer Y- und positiver X-Richtung verschoben.

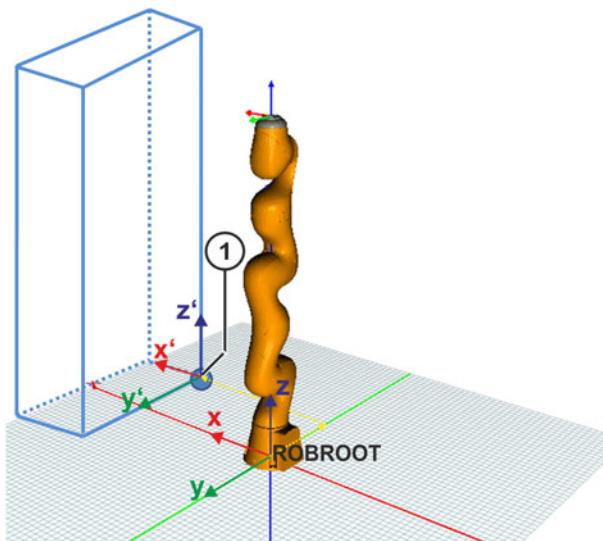


Abb. 13-13: Beispiel eines kartesischen Schutzraums

1 Ursprung des Schutzraums

### 13.8.9.3 Achsspezifische Überwachungsräume definieren

#### Beschreibung

Die Achsgrenzen können für jede Achse einzeln festgelegt und sicher überwacht werden. Der Achswinkel muss innerhalb des definierten Achsbereichs liegen.

Um einen achsspezifischen Überwachungsraum zu definieren, wird die AMF **Achsbereichsüberwachung** verwendet. Die AMF ist verletzt, wenn sich eine Achse außerhalb des definierten Achsbereiches befindet.

Zusätzlich ist die AMF in folgenden Fällen verletzt:

- Eine Achse ist nicht justiert.
- Die Referenzierung einer justierten Achse ist fehlgeschlagen.

Von dieser parametrierbaren AMF sind 16 Instanzen verfügbar, d. h. es können maximal 16 achsspezifische Überwachungsräume konfiguriert werden.

Parameter	Beschreibung
Überwachte Kinematik	Kinematik, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Überwachte Achse	Achse, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Achse1 ... Achse16</b></li> </ul> <p><b>Hinweis:</b> Bei einem LBR iiwa werden <b>Achse1 ... Achse7</b> verwendet.</p>
Untere Grenze [°]	Untere Grenze des erlaubten Achsbereichs, in dem sich die überwachte Achse bewegen darf <ul style="list-style-type: none"> <li>■ <b>-180° ... +180°</b></li> </ul>
Obere Grenze [°]	Obere Grenze des erlaubten Achsbereichs, in dem sich die überwachte Achse bewegen darf <ul style="list-style-type: none"> <li>■ <b>-180° ... +180°</b></li> </ul>

**i** Der erlaubte Achsbereich verläuft in positiver Drehrichtung der Achse von der unteren zur oberen Grenze.  
Soll die Achsposition bei  $\pm 180^\circ$  im zulässigen Winkelbereich liegen, muss die untere Grenze größer sein als die obere Grenze.

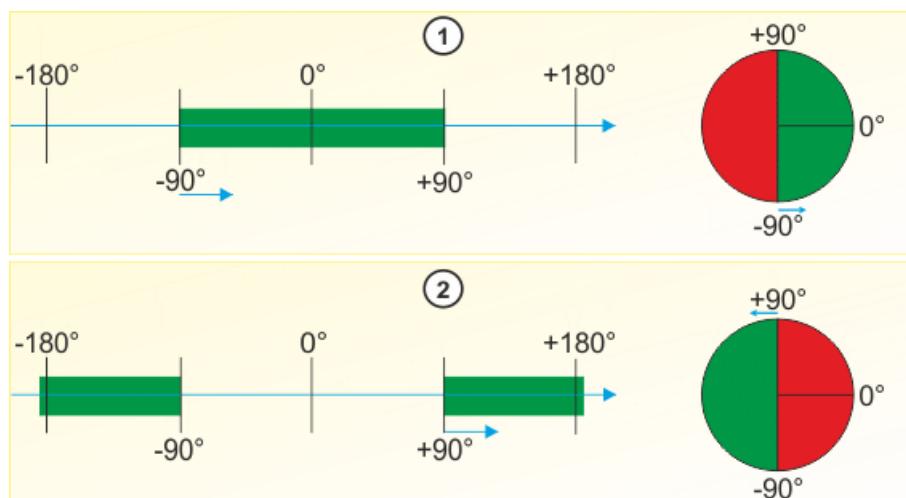


Abb. 13-14: Beispiele für achsspezifische Arbeitsräume

- 1 Untere Grenze: -90°; Obere Grenze: +90°
- 2 Untere Grenze: +90°; Obere Grenze: -90°



Für den Personenschutz ist nur die Stellung der Achse relevant. Darauf werden auch bei Achsen, die sich mehr als 360° drehen können, die Positionen auf den Achsbereich **-180° ... +180°** umgerechnet.

## Beispiel

Die Achsen A1, A2 und A4 sollen überwacht werden, so dass der Roboter nur in einem eingeschränkten Bereich verfahren kann. Die Überwachung wird über einen sicheren Eingang aktiviert. Der erlaubte Bereich jeder Achse wird durch eine obere und eine untere Grenze festgelegt und in der zugehörigen Grafik in der PSM-Tabelle grün dargestellt.

Sobald einer der überwachten Achsbereiche verletzt wird, soll ein Sicherheitshalt 1 (bahntreu) ausgelöst werden. Um dies zu erreichen, muss für jede Achse eine eigene Tabellenzeile verwendet werden.

Konfigurierbare Anwender-Sicherheitskonfiguration						
Zeile	Aktiv	Kategorie	AMF 1	AMF 2	AMF 3	Reaktion
8	<input checked="" type="checkbox"/>	Raumüberwachung	Eingangssignal (4) Eingang des Safety-Signals : EingangCIB_SR.4	Achsbereichsüberwachung (1) Überwachte Achse : Achse 1	 -	Stopp 1 (bahntreu)
9	<input checked="" type="checkbox"/>	Raumüberwachung	Eingangssignal (4) Eingang des Safety-Signals : EingangCIB_SR.4	Achsbereichsüberwachung (2) Überwachte Achse : Achse 2	 -	Stopp 1 (bahntreu)
10	<input checked="" type="checkbox"/>	Raumüberwachung	Eingangssignal (4) Eingang des Safety-Signals : EingangCIB_SR.4	Achsbereichsüberwachung (3) Überwachte Achse : Achse 4	 -	Stopp 1 (bahntreu)

Abb. 13-15: PSM-Tabelle – Gleichzeitige Überwachung von 3 Achsen

### 13.8.10 Überwachung der Werkzeugorientierung

Mit der AMF *Werkzeugorientierung* kann die Orientierung des sicherheitsgerichteten Werkzeugs überwacht werden. Dabei wird geprüft, ob eine bestimmte Achse des Werkzeugorientierungs-Frames innerhalb eines zulässigen Richtungsbereichs liegt.

Diese Funktion kann beispielsweise verwendet werden, um in MRK-Anwendungen zu verhindern, dass gefährliche Bereiche des montierten Werkzeugs, wie scharfe Kanten, in Richtung des Menschen zeigen.

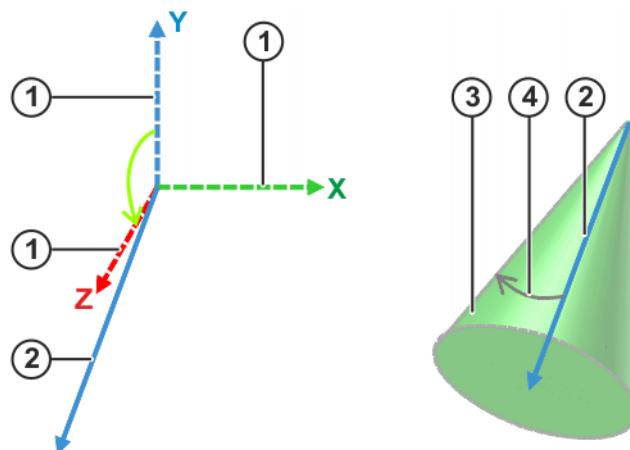
Überwacht wird die Orientierung der Z-Achse des Werkzeugorientierungs-Frames des sicherheitsgerichteten Werkzeugs. Ist kein sicherheitsgerichtetes Werkzeug konfiguriert, wird die Z-Achse des Flansch-Koordinatensystems überwacht.

(>>> 9.3.7.1 "Sicherheitsgerichtetes Werkzeug definieren" Seite 149)

Der zulässige Bereich für den Orientierungswinkel wird durch einen Referenzvektor mit fester Orientierung bezüglich des Welt-Koordinatensystems und einen zulässigen Abweichungswinkel von diesem Referenzvektor definiert.

Der Referenzvektor wird durch Drehung des Einheitsvektors der Z-Achse des Welt-Koordinatensystems um die 3 Euler-Winkel A, B und C bezüglich des Welt-Koordinatensystems definiert. Um den Referenzvektor wird ein Überwachungskegel aufgespannt. Die Öffnung des Kegels wird durch einen einstellbaren Abweichungswinkel definiert. Der Abweichungswinkel legt den erlaubten Winkel zwischen Werkzeugorientierung und Referenzvektor fest. Die Werte der Winkel des Referenzvektors sowie der Abweichungswinkel werden bei der Parametrierung der AMF festgelegt.

Der Überwachungskegel definiert den zulässigen Bereich für die Werkzeugorientierung.



**Abb. 13-16: Überwachungskegel für Werkzeugorientierung**

Pos.	Beschreibung
1	Achsen des Welt-Koordinatensystems
2	Referenzvektor Der Referenzvektor legt eine feste Orientierung bezüglich des Welt-Koordinatensystems fest.
3	Überwachungskegel Legt den zulässigen Bereich für die Werkzeugorientierung fest.
4	Abweichungswinkel Der Abweichungswinkel bestimmt die Öffnung des Überwachungskegels.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform) ist zu berücksichtigen, dass die Orientierung des Referenzvektors auf das Welt-Koordinatensystem bezogen ist und damit relativ zur Ausrichtung der Basis der überwachten Kinematik definiert ist. Daher wird bei einer Neigungsänderung der Trägerkinematik (z. B. aufgrund des Fahrens über eine Rampe) die Referenzorientierung mitbewegt.

AMF	Beschreibung
Werkzeugorientierung	Parametrierbare AMF mit 8 verfügbaren Instanzen  Die AMF ist verletzt, wenn der Winkel zwischen Referenzvektor und Z-Achse des Werkzeugorientierungs-Frames größer als der konfigurierte Abweichungswinkel ist.  Zusätzlich ist die AMF in folgenden Fällen verletzt: <ul style="list-style-type: none"><li>■ Eine Achse ist nicht justiert.</li><li>■ Die Positionsreferenzierung einer justierten Achse ist fehlgeschlagen.</li></ul>

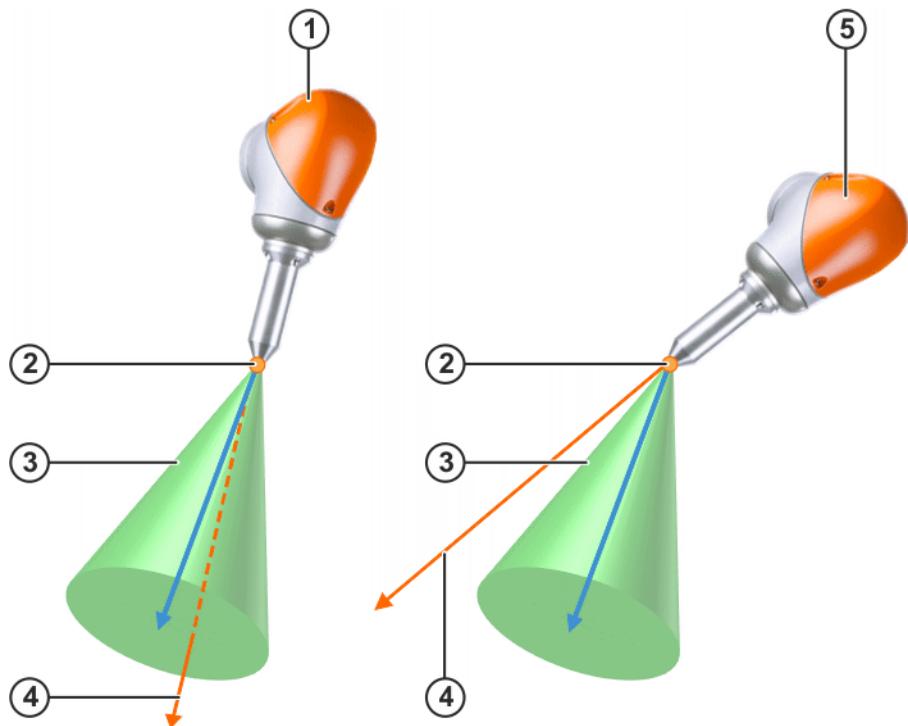


Abb. 13-17: Werkzeugorientierung (nicht verletzt und verletzt)

Pos.	Beschreibung
1	Roboter verletzt die AMF Werkzeugorientierung nicht. Die Z-Achse des Werkzeugorientierungs-Frames liegt innerhalb des durch den Überwachungskegel definierten Bereichs.
2	Ursprung des Werkzeugorientierungs-Frames
3	Überwachungskegel
4	Z-Achse des Werkzeugorientierungs-Frames
5	Roboter verletzt die AMF Werkzeugorientierung. Die Z-Achse des Werkzeugorientierungs-Frames liegt außerhalb des durch den Überwachungskegel definierten Bereichs.

Parameter	Beschreibung
Überwachte Kinematik	Kinematik, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
A [°]	Rotation des Referenzvektors um die Z-Achse des Welt-Koordinaten-systems <ul style="list-style-type: none"> <li>■ <b>0° ... 359°</b></li> </ul>
B [°]	Rotation des Referenzvektors um die Y-Achse des Welt-Koordinaten-systems <ul style="list-style-type: none"> <li>■ <b>0° ... 359°</b></li> </ul>

Parameter	Beschreibung
$C [^{\circ}]$	Rotation des Referenzvektors um die X-Achse des Welt-Koordinaten-systems ■ <b>0° ... 359°</b>
<i>Arbeitsbereich [^{\circ}]</i>	Arbeitsbereich der Werkzeugorientierung Legt den maximal zulässigen Abweichungswinkel zwischen Referenzvektor und der Z-Achse des Werkzeugorientierungs-Frames fest. ■ <b>1° ... 179°</b>

### 13.8.11 Stillstandsüberwachung (Sicherer Betriebshalt)

<b>Beschreibung</b>	Soll sich der Roboter unter bestimmten Bedingungen nicht bewegen, aber in Regelung bleiben, muss der Stillstand aller Achsen sicher überwacht werden. Hierfür wird die AMF <i>Stillstandsüberwachung aller Achsen</i> verwendet.  Bei dieser AMF handelt es sich um eine Extended AMF, d. h. die Überwachung beginnt erst dann, wenn alle anderen AMFs der Sicherheitsfunktion verletzt sind.
---------------------	---



Extended AMFs stehen für die Sicherheitsfunktionen des ESM-Mechanismus nicht zur Verfügung.

Stillstand ist definiert als Beibehalten der Achspositionen. Zu Beginn der Stillstandsüberwachung werden die Achspositionen gespeichert und solange die Überwachung aktiv ist mit den aktuellen Gelenkwerten verglichen.

Da die Stillstandsüberwachung in einem engen Toleranzbereich überwacht, kann die Überwachung auch dann verletzt sein, wenn die Bewegung des Roboters durch äußere Krafteinwirkung, beispielsweise ein Anstoßen des Roboters, verursacht wird.

AMF	Beschreibung
<i>Stillstandsüberwachung aller Achsen</i>	Parametrierbare Extended AMF mit 8 verfügbaren Instanzen  Die AMF ist verletzt, sobald der Gelenkwert einer Achse außerhalb eines Toleranzbereichs von +/- 0.1° um den bei Aktivierung der Stillstandsüberwachung gespeicherten Wert befindet oder sich eine der Achsen mit betragsmäßig mehr als 1 °/s bewegt.

Parameter	Beschreibung
<i>Überwachte Kinematik</i>	Kinematik, die überwacht werden soll ■ <b>Erste Kinematik:</b> Roboter ■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion ■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion ■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion

#### Beispiel

##### Kategorie: Sicherer Betriebshalt

Wenn sich der Roboter außerhalb seines Arbeitsbereichs befindet, soll sichergestellt werden, dass er sich nicht mehr bewegt, sobald sich Personen in Reichweite befinden. Der Arbeitsbereich ist durch einen kartesischen Arbeitsraum konfiguriert. An einem sicheren Eingang ist ein Sensor angeschlossen, durch den gefährdete Personen erkannt werden. Sind sowohl der Arbeitsraum als auch das Eingangssignal verletzt, wird die Stillstandsüberwachung aktiv.

AMF1	AMF2	AMF3	Reaktion
Eingangssignal	Kartesische Arbeitsraumüberwachung	Stillstandsüberwachung aller Achsen	Stopp 1

### 13.8.12 Einschaltverzögerung für Sicherheitsfunktionen

#### Beschreibung

Mithilfe der AMF *Zeitverzögerung* kann das Auslösen der Reaktion einer Sicherheitsfunktion um eine definierte Zeit verzögert werden.

Bei dieser AMF handelt es sich um eine Extended AMF, d. h. die Verzögerungszeit beginnt erst abzulaufen, wenn alle anderen AMFs der Sicherheitsfunktion verletzt sind.



Extended AMFs stehen für die Sicherheitsfunktionen des ESM-Mechanismus nicht zur Verfügung.

AMF	Beschreibung
Zeitverzögerung	Parametrierbare Extended AMF mit 16 verfügbaren Instanzen.  Die AMF ist verletzt, wenn die eingestellte Zeit abgelaufen ist.



Wenn dieselbe Instanz der AMF für mehrere Sicherheitsfunktionen verwendet wird, läuft die Verzögerungszeit ab der ersten Aktivierung.

Parameter	Beschreibung
Verzögerungszeit	Zeit, um die das Auslösen der Reaktion einer Sicherheitsfunktion verzögert wird.  ■ <b>12 ms ... 24 h</b>  Die Zeit kann in Millisekunden (ms), Sekunden (s), Minuten (min) und Stunden (h) eingegeben werden. Jede Verzögerung ist ein Vielfaches von 12 ms, d. h. es wird zum nächsten Vielfachen aufgerundet.

#### Beispiel

Kategorie: *Sicherheitshalt*

Ein Roboter mit nicht referenzierten Achsen soll für einen begrenzte Zeit im Automatikbetrieb verfahren werden können. Nach Ablauf dieser Zeit, beispielsweise nach 2 Stunden, soll ein Sicherheitshalt 1 (bahntreu) ausgelöst werden.

AMF1	AMF2	AMF3	Reaktion
Betriebsart Automatik	Positionsreferenzierung	Zeitverzögerung	Stopp 1 (bahntreu)

### 13.8.13 Überwachung von Kräften und Momenten

Der LBR iiwa ist mit Positions- und Gelenkmomenten-Sensoren in allen Achsen ausgestattet. Dadurch können von außen wirkende Kräfte und Momente gemessen und darauf reagiert werden.



Bei Verwendung der AMF *Kollisionserkennung*, *TCP-Kraftüberwachung* und *Achsmomentenüberwachung* ist zu berücksichtigen, dass aufgrund der Anhaltewege des Roboters die Interaktionskräfte weiter steigen können, wenn die AMF verletzt und ein Sicherheitshalt ausgelöst wird. Aus diesem Grund wird empfohlen, den kollaborierenden Betrieb (MRK) nur mit reduzierten Geschwindigkeiten durchzuführen, d. h. diese AMFs mit der AMF *Kartesische Geschwindigkeitsüberwachung*, *Achs geschwindigkeitsüberwachung* oder *Werkzeugbezogene Geschwindigkeitskomponente* zu kombinieren.



Bei Verwendung der AMF *Kollisionserkennung* und *TCP-Kraftüberwachung* ist zu berücksichtigen, dass externe Kräfte auf die Roboterstruktur mit geringem Abstand zu den Roboterachsen unter Umständen nur geringe externe Achsmomente in den Roboterachsen verursachen. Dies kann insbesondere in Quetschsituationen ein Sicherheitsrisiko beim kollaborierenden Betrieb darstellen. Quetschsituationen können sich aufgrund von Klemmsituationen innerhalb der Roboterstruktur oder des Roboters mit der Umgebung ergeben. Daher wird empfohlen, mögliche kritische Quetschsituationen durch eine geeignete Gestaltung der Roboterzelle und/oder eine zusätzliche Verwendung der AMFs *Kartesische Arbeitsraumüberwachung*, *Kartesische Schutzraumüberwachung*, *Achsbereichsüberwachung* oder *Werkzeugorientierung* zu vermeiden.



Bei Verwendung der AMF *Kollisionserkennung* und *TCP-Kraftüberwachung* muss der Betreiber sicherstellen, dass sich aufgenommene Werkstücke nicht unbeabsichtigt lösen und herabfallen, während eine der Überwachungen aktiv ist.

### 13.8.13.1 Achsmomentenüberwachung

Mit der Achsmomentenüberwachung können die Momente einzelner Achsen begrenzt und überwacht werden.



Kommt es aufgrund einer Einklemmsituation zu einer dauerhaften Überschreitung des erlaubten Achsmoments, kann der Roboter durch einen Wechsel in die Betriebsart KRF freigefahren werden.

AMF	Beschreibung
Achsmomentenüberwachung	Parametrierbare AMF mit 16 verfügbaren Instanzen  Die AMF ist verletzt, wenn das Drehmoment der überwachten Achse die definierte Momentenbegrenzung über- oder unterschreitet.

Parameter	Beschreibung
Überwachte Kinematik	Kinematik, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Überwachte Achse	Achse, die überwacht werden soll <ul style="list-style-type: none"> <li>■ <b>Achse1 ... Achse16</b></li> </ul> <p><b>Hinweis:</b> Bei einem LBR iiwa werden <b>Achse1 ... Achse7</b> verwendet.</p>

Parameter	Beschreibung
<i>Minimales Moment [Nm]</i>	Minimal zulässiges Drehmoment der angegebenen Achse ■ -500 ... 500 Nm
<i>Maximales Moment [Nm]</i>	Maximal zulässiges Drehmoment der angegebenen Achse ■ -500 ... 500 Nm

### 13.8.13.2 Kollisionserkennung

Bei der Kollisionserkennung werden die externen Achsmomente gegen einen definierbaren Grenzwert überwacht.

Das externe Achsmoment ist definiert als der Anteil des Drehmoments einer Achse, der aufgrund der auftretenden Kräfte und Momente bei der Interaktion des Roboters mit der Umgebung erzeugt wird. Es wird nicht direkt gemessen, sondern mithilfe des dynamischen Robotermodells ermittelt. Die Genauigkeit der ermittelten Werte ist abhängig von der Dynamik der Roboterbewegung und der Dynamik der Interaktionskräfte des Roboters mit der Umgebung.

Folgende Punkte sind bei Einsatz der Kollisionserkennung zu beachten:

- Eine erfolgreich durchgeführte Positions- und Momentenreferenzierung wird vorausgesetzt.
- Die Lastdaten des sicherheitsgerichteten Werkzeugs (sofern konfiguriert) werden berücksichtigt.
- Ist das sicherheitsgerichtete Werkzeug konfiguriert, muss dieses am Roboterflansch montiert sein.
- Die Lastdaten sicherheitsgerichteter Werkstücke (sofern konfiguriert) werden nur berücksichtigt, wenn der Sicherheitssteuerung das aktuell aktive sicherheitsgerichtete Werkstück bekanntgegeben wird.  
(>>> 15.11.6 "Kommandieren von Lastwechseln an Sicherheitssteuerung" Seite 317)



In der AMF Kollisionserkennung werden mögliche Fehler beim Aktivieren des sicherheitsgerichteten Werkstücks nicht automatisch berücksichtigt.

Bei der Konfiguration der Kollisionserkennung müssen deshalb möglichst geringe Werte für das maximal zulässige externe Moment eingestellt werden, damit Abweichungen in den Lastdaten als Kollision interpretiert werden und dadurch eine Verletzung der AMF bewirken.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) muss während der Verwendung der AMF sichergestellt sein, dass sich die Trägerkinematik nicht bewegt. Solange die Roboterbasis der überwachten Kinematik beschleunigt wird, ist die Sicherheitsintegrität der AMF nicht gewährleistet.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) muss während der Verwendung der AMF sichergestellt sein, dass die Montagerichtung der überwachten Kinematik nicht von der konfigurierten Montagerichtung abweicht (z. B. aufgrund einer Schiefstellung der mobilen Plattform). Andernfalls ist die Sicherheitsintegrität der AMF nicht gewährleistet.

AMF	Beschreibung
<i>Kollisionserkennung</i>	Parametrierbare AMF mit 8 verfügbaren Instanzen  Die AMF ist verletzt, wenn das externe Moment mindestens einer Achse den konfigurierten Grenzwert überschreitet.

Parameter	Beschreibung
Überwachte Kinematik	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Maximales externes Moment [Nm]	<p>Maximal zulässiges externes Moment</p> <ul style="list-style-type: none"> <li>■ <b>0 ... 30 Nm</b></li> </ul>

### 13.8.13.3 TCP-Kraftüberwachung

**Beschreibung** Bei der TCP-Kraftüberwachung wird die externe Kraft, die am TCP des sicherheitsgerichteten Werkzeugs wirkt, gegen einen definierbaren Grenzwert überwacht. Wird kein Werkzeug eingesetzt, wird die externe Kraft überwacht, die am Roboterflansch-Mittelpunkt wirkt.

Die externe Kraft am TCP wird nicht direkt gemessen, sondern mithilfe des dynamischen Robotermodells ermittelt. Die Genauigkeit der ermittelten externen Kraft ist unter anderem abhängig von der Dynamik der Roboterbewegung und der Dynamik der tatsächlich wirkenden Kraft.

Folgende Punkte sind bei Einsatz der TCP-Kraftüberwachung zu beachten:

- Eine erfolgreich durchgeführte Positions- und Momentenreferenzierung wird vorausgesetzt.
- Die Lastdaten des sicherheitsgerichteten Werkzeugs (sofern konfiguriert) werden berücksichtigt.
- Ist das sicherheitsgerichtete Werkzeug konfiguriert, muss dieses am Roboterflansch montiert sein.
- Die Lastdaten des schwersten sicherheitsgerichteten Werkstücks (sofern konfiguriert) werden berücksichtigt.



In der AMF *TCP-Kraftüberwachung* werden mögliche Fehler beim Aktivieren eines sicherheitsgerichteten Werkstücks automatisch berücksichtigt.

Bei der Konfiguration der TCP-Kraftüberwachung muss deshalb für die maximal zulässige externe Kraft am TCP ein Wert eingestellt werden, der größer ist als die Gewichtskraft des schwersten aufzunehmenden Werkstücks.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) muss während der Verwendung der AMF sichergestellt sein, dass sich die Trägerkinematik nicht bewegt. Solange die Roboterbasis der überwachten Kinematik beschleunigt wird, ist die Sicherheitsintegrität der AMF nicht gewährleistet.



Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) muss während der Verwendung der AMF sichergestellt sein, dass die Montagerichtung der überwachten Kinematik nicht von der konfigurierten Montagerichtung abweicht (z. B. aufgrund einer Schiefstellung der mobilen Plattform). Andernfalls ist die Sicherheitsintegrität der AMF nicht gewährleistet.

AMF	Beschreibung
TCP-Kraftüberwachung	<p>Parametrierbare AMF mit 8 verfügbaren Instanzen</p> <p>Die AMF ist verletzt, wenn die externe Kraft, die am TCP des sicherheitsgerichteten Werkzeugs (oder Roboterflansch-Mittelpunkt) wirkt, den konfigurierten Grenzwert überschreitet.</p>

Parameter	Beschreibung
Überwachte Kinematik	<p>Kinematik, die überwacht werden soll</p> <ul style="list-style-type: none"> <li>■ <b>Erste Kinematik:</b> Roboter</li> <li>■ <b>Zweite Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Dritte Kinematik:</b> Zur Zeit ohne Funktion</li> <li>■ <b>Vierte Kinematik:</b> Zur Zeit ohne Funktion</li> </ul>
Maximale TCP-Kraft [N]	<p>Maximal zulässige externe Kraft am TCP</p> <ul style="list-style-type: none"> <li>■ <b>50 ... 1 000 N</b></li> </ul>

<b>Genauigkeit der Krafterfassung</b>	<p>Die Genauigkeit der der TCP-Krafterfassung ist abhängig von der Roboterpose. Die Sicherheitssteuerung erkennt unzulässige Posen und setzt die AMF <i>TCP-Kraftüberwachung</i> mit einer entsprechenden Diagnosemeldung auf verletzt.</p> <p>Unzulässig sind Posen, in denen TCP-Kräfte möglich sind, die einen geringen Abstand zu allen Roboterachsen aufweisen. Dies gilt für singuläre Posen und Posen in der Nähe von Singularitäten. Zu vermeiden sind insbesondere Achswinkel in der Nähe von 0° in den Achsen A2, A4 und A6:</p> <ul style="list-style-type: none"> <li>■ Achse A2: -35° bis 35°</li> <li>■ Achse A4: -55° bis 55°</li> <li>■ Achse A6: -20° bis 20°</li> </ul> <p>■ Externe Kräfte auf die Roboterstruktur verringern die Genauigkeit der TCP-Krafterfassung. Die Sicherheitssteuerung kann in vielen Fällen das Einwirken von externen Kräften auf die Roboterstruktur selbstständig erkennen. In diesem Fall ist die AMF <i>TCP-Kraftüberwachung</i> verletzt.</p>
---------------------------------------	---



Es kann nicht garantiert werden, dass die Sicherheitssteuerung immer selbstständig erkennt, dass externe Kräfte auf die Roboterstruktur wirken. Der Anwender muss sicherstellen, dass die externen Kräfte ausschließlich auf den TCP wirken, um die Sicherheitsintegrität der AMF *TCP-Kraftüberwachung* zu gewährleisten.

### 13.9 Beispiel einer Sicherheitskonfiguration



Dieses Beispiel dient ausschließlich zur Veranschaulichung der Sicherheitskonfiguration mit KUKA Sunrise.Workbench.



Grundlage für die Sicherheitskonfiguration einer Anlage ist immer eine eigene Risikoanalyse. Die unten gezeigte Sicherheitskonfiguration dient als Beispiel und erhebt keinen Anspruch auf Vollständigkeit.

#### 13.9.1 Aufgabe

Der LBR iiwa wird in einer Applikation in Kooperation mit einem Menschen eingesetzt. Das am Roboter montierte Werkzeug ist als sicherheitsgerichtetes Werkzeug eingestellt.

Der Bediener legt in regelmäßigen Abständen an einer Werkstückaufnahme ein Werkstück bereit. Eine Aufgabe des Roboters ist es, die Anwesenheit des Werkstücks zu prüfen. Dafür bewegt er sich, von einer Startposition ausgehend, mit einer Transferbewegung durch den für den Menschen zugängigen Bereich (Kollaborationsbereich). Ziel dieser Transferbewegung ist eine Vorposition, die sich 20 cm über dem bereitgestellten Werkstück befindet. Anschließend fährt er auf das Werkstück zu.

Diese Absenkbewegung ist mit einer Kraftabbruchbedingung (Prozessschwellwert: 20 N) parametriert. Der Roboter stellt anhand der aktuellen Position nach Erreichen des Prozessschwellwertes fest, ob das Werkstück anwesend ist oder nicht. Das Werkstück ist nicht anwesend, wenn er bis auf die Werkstückaufnahme fahren kann. Nach der Prüfung fährt der Roboter über die Vorposition aus dem Kollaborationsbereich heraus.

### **13.9.2 Anforderung**

Im Rahmen einer Risikobeurteilung werden für den beschriebenen Prozess folgende Sicherheitsfunktionen angefordert:

1. Der Roboter muss über einen externen NOT-HALT in Reichweite des Bedieners stillgesetzt werden können.
2. Der Roboter darf einen festgelegten Arbeitsraum nicht verlassen. Der Kollaborationsbereich ist ein Teil des Arbeitsraums.
3. Bei der Transferbewegung zwischen Startposition und Vorposition kann es zu ungewollten Kollisionen mit dem Bediener kommen. Der Bereich ist jedoch so gestaltet, dass dabei keine Klemmung des Menschen auftreten kann. Daher wurde die erlaubte maximale Geschwindigkeit des Roboters in diesem Bereich auf 500 mm/s festgelegt.
4. Kollisionen müssen bei der Transferbewegung sicher erkannt werden und zum Stillstand des Roboters führen, wenn durch die Kollisionen ein Moment von 15 Nm in mindestens einer Achse überschritten wird.
5. Bei Bewegungen zwischen Vorposition und Werkstückaufnahme kann es zu Klemmungen von Hand und Arm des Bedieners kommen. Damit der Bediener auf eine Roboterbewegung angemessen reagieren kann und die Bremswege ausreichend klein sind, darf die Geschwindigkeit des Roboters 100 mm/s nicht überschreiten.
6. Darüber hinaus muss der Roboter während Bewegungen zwischen Vorposition und Werkstückaufnahme bei Klemmkräften von mehr als 50 N stillgesetzt werden. Der Kraftwert, ab dem die Absenkbewegung im Prozess abgebrochen wird, liegt mit 20 N ausreichend weit unter dieser Grenze.

### **13.9.3 Lösungsvorschlag für Aufgabenstellung**

Um die Anforderungen zu realisieren, müssen permanente und umschaltbare Sicherheitsüberwachungen konfiguriert werden:

- Permanente Überwachung der externen NOT-HALT-Einrichtung und des Arbeitsraums
- ESM-Zustand für die Transferbewegung zwischen Start- und Vorposition
- ESM-Zustand für die Bewegung zwischen Vorposition und Werkstückaufnahme

Um einen stabilen und flüssigen Prozessablauf zu gewährleisten, muss die Applikation so gestaltet werden, dass die für die Sicherheitsfunktionen festgelegten Grenzwerte (z. B. für Geschwindigkeit und Arbeitsraum) eingehalten werden.

Die Roboter-Applikation, die den beschriebenen Prozess realisiert, ist hier nicht aufgeführt.

### Permanente Sicherheitsüberwachung

Während des gesamten Betriebs muss die NOT-HALT-Funktion aktiv sein und der Roboter darf den Arbeitsraum nicht verlassen. Entsprechende Sicherheitsfunktionen werden in der Anwender PSM-Tabelle konfiguriert.

Konfigurierbare Anwender-Sicherheitskonfiguration							(6/100)
Zeile	Aktiv	Kategorie	AMF 1	AMF 2	AMF 3	Reaktion	
1	<input checked="" type="checkbox"/>	NOT-HALT extern	Eingangssignal (1) Eingang des Safety-Signals : EingangCIB_SR.1	-	-	Stopp 1 (bahntreu)	
2	<input checked="" type="checkbox"/>	Raumüberwachung	Kartesische Arbeitsraumüberwachung (1)	-	-	Stopp 1 (bahntreu)	

Abb. 13-18: Permanente Sicherheitsüberwachung

Zeile	Beschreibung
1	<p>NOT-HALT Extern</p> <p>Realisiert Anforderung 1</p> <p>Ein externer NOT-HALT ist an einem sicheren Eingang angegeschlossen. Betätigt der Bediener den NOT-HALT, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.</p>
2	<p>Kartesische Arbeitsraumüberwachung 2</p> <p>Realisiert Anforderung 2</p> <p>Der Arbeitsraum wird durch einen sicher überwachten kartesischen Arbeitsraum repräsentiert. Verlässt der Roboter den konfigurierten Bereich, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.</p>

### ESM-Zustand für Transferbewegung

Für die Transferbewegung zwischen Start- und Vorposition durch den Kollaborationsbereich wird ein ESM-Zustand definiert. Dieser wird in der Applikation vor dem Beginn der Transferbewegung aktiviert.

Während der Transferbewegung müssen Geschwindigkeitsüberwachung und Kollisionserkennung aktiv sein, um die Gefährdung beim Zusammenstoß von Mensch und Roboter auf ein ausreichendes Maß zu reduzieren.

Um eine Klemmung jederzeit auszuschließen, wird zusätzlich ein Schutzraum definiert, der den Roboter stillsetzt, sobald zwischen Roboter oder Werkzeug und der Werkstückaufnahme ein Abstand von 15 cm unterschritten wird.

Zustand für das Event-Driven Safety Monitoring (ESM)				(3/20)
Zeile	Aktiv	AMF	Reaktion	
1	<input checked="" type="checkbox"/>	Kartesische Geschwindigkeitsüberwachung (1) Maximale Geschwindigkeit : 500 mm/s	Stopp 1 (bahntreu)	
2	<input checked="" type="checkbox"/>	Kollisionserkennung (1) Maximales externes Moment : 15 Nm	Stopp 1 (bahntreu)	
3	<input checked="" type="checkbox"/>	Kartesische Schutzraumüberwachung (1)	Stopp 1 (bahntreu)	

Abb. 13-19: ESM-Zustand für Transferbewegung

Zeile	Beschreibung
1	Kartesische Geschwindigkeitsüberwachung Realisiert Anforderung 3  Wird eine kartesische Geschwindigkeit von 500 mm/s überschritten, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.
2	Kollisionserkennung Realisiert Anforderung 4  Bei einer Kollision, die ein externes Moment von mehr als 15 Nm in mindestens einer Achse des Roboters verursacht, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.
3	Schutzraumüberwachung Realisiert Sicherheit des Zustands unabhängig von Zeit und Ort der Aktivierung  Der sicher überwachte Schutzraum bildet den Bereich oberhalb der Werkstückaufnahme ab. Sobald der Roboter oder das sicher überwachte Werkzeug in diesen Bereich eintritt, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.

**ESM-Zustand für Werkstückaufnahme**

Für die Bewegungen zwischen Vorposition und Werkstückaufnahme wird ein eigener ESM-Zustand definiert. Dieser wird in der Applikation vor Beginn der Absenkbewegung aktiviert.

Bei der Bewegung müssen Geschwindigkeitsüberwachung und Kraftüberwachung aktiv sein, um die Gefährdung bei einer Klemmung von Hand oder Unterarm des Bedieners auf ein ausreichendes Maß zu reduzieren.

Der Zustand muss unabhängig von Zeitpunkt und Ort der Aktivierung ein ausreichendes Maß an Sicherheit gewährleisten. Aufgrund der geringen zulässigen Geschwindigkeit und der aktiven Kraftüberwachung ist dies ohne weitere Maßnahmen erfüllt.

Zustand für das Event-Driven Safety Monitoring (ESM)

(2/20)

Zeile	Aktiv	AMF	Reaktion
1	<input checked="" type="checkbox"/>	Kartesische Geschwindigkeitsüberwachung (2) Maximale Geschwindigkeit : 100 mm/s	Stopp 1 (bahntreu)
2	<input checked="" type="checkbox"/>	TCP-Kraftüberwachung (1) Maximale TCP-Kraft : 50 N	Stopp 0

Abb. 13-20: ESM-Zustand für Werkstückaufnahme

Zeile	Beschreibung
1	Kartesische Geschwindigkeitsüberwachung Realisiert Anforderung 5  Wird eine kartesische Geschwindigkeit von 100 mm/s überschritten, wird ein Sicherheitshalt 1 (bahntreu) ausgeführt.
2	Kraftüberwachung Realisiert Anforderung 6  Bei einem Kontakt, durch den eine Kraft von mehr als 50 N am TCP des Roboters verursacht wird, wird ein Sicherheitshalt 0 ausgeführt.

## 13.10 Positions- und Momentenreferenzierung

### 13.10.1 Positionsreferenzierung

**Beschreibung** Bei der Positionsreferenzierung wird überprüft, ob die gespeicherte Nullstellung des Motors einer Achse (= gespeicherte Justageposition) mit der tatsächlichen mechanischen Nullstellung dieser Achse übereinstimmt.

Die Referenzierung wird kontinuierlich vom System durchgeführt, wenn eine Achse mit weniger als 30 °/s verfährt. Die Referenzierung erfolgt, wenn der Justagesensor die mechanische Nullstellung der Achse in einem engen Bereich um die gespeicherte Nullstellung des Motors erkennt. Die Referenzierung schlägt fehl, wenn der Justagesensor die mechanische Nullstellung der Achse im Bereich der gespeicherten Nullstellung des Motors nicht erkennt, oder wenn sie an einer unerwarteten Stelle erkannt wird.

Solange noch keine Positionsreferenzierung durchgeführt wurde, ist die Sicherheitsintegrität der darauf basierenden Sicherheitsfunktionen eingeschränkt. Dazu gehören sicher überwachte kartesische und achsspezifische Roboterpositionen, kartesische Geschwindigkeiten, Kräfte und Kollisionen.

Ist die Positionsreferenzierung mindestens einer Achse fehlgeschlagen, sind alle AMFs, die auf sicheren Achspositionen basieren, verletzt. ([>>> "Positionsbasierte AMFs" Seite 246](#))

**Anforderung** Die Position einer Achse ist nach folgenden Ereignissen nicht referenziert:

- Robotersteuerung startet neu.
- Achse wird neu justiert.
- Momentenreferenzierung der Achse schlägt fehl.
- Maximaldrehmoment des Gelenkmomenten-Sensors der Achse wird überschritten.



Nach diesen Ereignissen sind die auf sicheren Positionen basierenden Sicherheitsfunktionen nicht verletzt. Der Roboter kann verfahren werden, aber die Sicherheitsintegrität der Sicherheitsfunktionen ist nicht gegeben.

Die positionsbasierten Sicherheitsfunktionen werden nach diesen Ereignissen erst dann verletzt, wenn die Positionsreferenzierung einer Achse fehlschlägt. Vor dem Ausführen sicherheitskritischer Anwendungen muss die Referenzierung erfolgreich ausgeführt worden sein.

Der Status der Positionsreferenzierung kann in der Sicherheitskonfiguration als AMF verwendet werden. ([>>> 13.8.6 "Auswertung der Positionsreferenzierung" Seite 212](#))

**Voraussetzung**

Die Position einer Achse ist referenziert, wenn über die gespeicherte Nullposition des Motors gefahren und dabei in einem Bereich von 0° +/- 0.5° die Nullstellung der Achse durch den Justagesensor erkannt wird. Voraussetzungen hierfür sind:

- Die Geschwindigkeit, mit der über die Nullposition gefahren wird, ist < 30 °/s.
- Ein achsspezifischer Bereich, vor und nach der Nullposition, wird mindestens durchfahren. Die Verfahrrichtung ist nicht relevant.

Der achsspezifische Verfahrbereich ist roboterspezifisch:

Robotervariante	A1	A2	A3	A4	A5	A6	A7
LBR iiwa 7 R800	±10.5°	±10.5°	±10.5°	±10.5°	±10.5°	±14°	±14°
LBR iiwa 14 R820	±9.5°	±9.5°	±10.5°	±10.5°	±10.5°	±14°	±14°

**Durchführung**

Die Positionsreferenzierung aller Achsen wird kontinuierlich vom System durchgeführt, wenn die oben genannten Bedingungen erfüllt sind. Die Positionsreferenzierung kann auf folgende Arten gezielt vorgenommen werden:

- Automatisch während des Programmablaufs, wenn eine Achse mit weniger als 30 °/s über die Nullposition fährt.
- Per Handverfahren jede Achse einzeln über die Nullposition fahren.
- Die von KUKA für die Referenzierung vorbereitete Applikation ausführen. Von der Kerzenstellung aus werden die Achsen über die Nullposition gefahren.

Sunrise.Workbench stellt eine vorbereitete Applikation für die Positions- und Momentenreferenzierung des LBR iiwa zur Verfügung. Mit dieser Applikation werden Positions- und Momentenreferenzierung gleichzeitig durchgeführt.

(>>> 13.10.3 "Applikation für die Positions- und Momentenreferenzierung erstellen" Seite 243)



Falls es nicht möglich ist, von der Kerzenstellung aus zu referenzieren, ist es notwendig, eine eigene Applikation für die Positionsreferenzierung zu erstellen und auszuführen.

### 13.10.2 Momentenreferenzierung

**Beschreibung**

Der LBR iiwa besitzt in jeder Achse einen Gelenkmomenten-Sensor, über den das aktuell auf die Achse wirkende Drehmoment sicher ermittelt wird. Basierend auf diesen Daten werden beispielsweise extern wirkende Momente oder kartesische Kräfte berechnet und überwacht.

Bei der Referenzierung der Gelenkmomenten-Sensoren wird überprüft, ob das erwartete externe Drehmoment, das für eine Achse aufgrund des Robotermodells und der angegebenen Lastdaten berechnet werden kann, mit dem basierend auf dem Messwert des Gelenkmomenten-Sensors ermittelten Wert übereinstimmt. Übersteigt die Differenz zwischen diesen Werten einen bestimmten Toleranzwert, ist die Referenzierung der Momentensensoren fehlgeschlagen.

Solange die Momentenreferenzierung nicht erfolgreich durchgeführt wurde, ist die Sicherheitsintegrität der darauf basierenden Sicherheitsfunktionen eingeschränkt. Dazu gehören Achsmomenten- und TCP-Kraftüberwachung sowie die Kollisionserkennung.

Ist die Momentenreferenzierung mindestens einer Achse fehlgeschlagen, sind alle AMFs, die auf sicheren Momentenwerten basieren, verletzt.

(>>> "Achsmomentbasierte AMFs" Seite 247)

**Anforderung**

Der Gelenkmomenten-Sensor einer Achse ist nach folgenden Ereignissen nicht referenziert:

- Robotersteuerung startet neu.
- Positionsreferenzierung der Achse schlägt fehl.
- Maximaldrehmoment des Gelenkmomenten-Sensors der Achse wird überschritten.



Nach diesen Ereignissen sind die auf sicheren Momenten basierenden Sicherheitsfunktionen nicht verletzt. Der Roboter kann verfahren werden, aber die Sicherheitsintegrität der Sicherheitsfunktionen ist nicht gegeben.

Die momentenbasierten Sicherheitsfunktionen werden nach diesen Ereignissen erst dann verletzt, wenn die Momentenreferenzierung einer Achse fehlschlägt. Vor dem Ausführen sicherheitskritischer Anwendungen muss die Referenzierung erfolgreich ausgeführt worden sein.

Der Status der Momentenreferenzierung kann in der Sicherheitskonfiguration als AMF verwendet werden. ([>>> 13.8.7 "Auswertung der Momentenreferenzierung" Seite 212](#))

## Durchführung

Sunrise.Workbench stellt eine vorbereitete Applikation für die Positions- und Momentenreferenzierung des LBR iiwa zur Verfügung. Mit dieser Applikation werden Positions- und Momentenreferenzierung gleichzeitig durchgeführt.

([>>> 13.10.3 "Applikation für die Positions- und Momentenreferenzierung erstellen" Seite 243](#))

Für die Momentenreferenzierung müssen insgesamt 10 gemessene Gelenkmomentwerte pro Achse vorliegen. Hierfür sind in der Applikation 5 Messposes definiert, die jeweils mit positiver und negativer Achsdrehung angefahren werden. Sind die Posen nicht anfahrbar, müssen sie in der Applikation angepasst werden.



Bei der Momentenreferenzierung muss jede der Messposes nacheinander mit positiver und negativer Achsdrehung angefahren werden, bevor die nächste Messpose angefahren wird. Andernfalls ist die Sicherheitsintegrität der Referenzierung der Gelenkmomenten-Sensoren nicht gegeben.



Vor dem Durchführen der Momentenreferenzierung muss der Anwender sicherstellen, dass die Lastdaten des am Roboter befestigten Werkzeugs mit den für das sicherheitsgerichtete Werkzeug angegebenen Lastdaten übereinstimmen und die Lastdaten des aufgenommenen Werkstücks (sofern vorhanden) mit den Lastdaten des aktivierten sicherheitsgerichteten Werkstücks. Andernfalls ist die Sicherheitsintegrität der Referenzierung der Gelenkmomenten-Sensoren nicht gegeben.  
Insbesondere dürfen keine Zusatzlasten am Roboter befestigt sein, z. B. an der Roboterstruktur befestigte Lasten oder von der Sicherheitssteuerung nicht berücksichtigte aufgenommene Werkstücke.



Während der Momentenreferenzierung dürfen keine externen Kräfte auf den Roboter sowie auf das am Flansch montierte Werkzeug und aufgenommene Werkstück (sofern vorhanden) wirken. Dies muss der Anwender beim Durchführen der Referenzierung sicherstellen. Andernfalls ist die Sicherheitsintegrität der Referenzierung der Gelenkmomenten-Sensoren nicht gegeben.

Die Sicherheitssteuerung wertet für alle 10 Messwerte das externe Moment aus und ermittelt den Mittelwert des externen Moments jeder Achse. Ist dieser Mittelwert unterhalb einer bestimmten Toleranz, ist die Referenzierung erfolgreich. Andernfalls ist die Referenzierung fehlgeschlagen.

**!** Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit) muss der Anwender sicherstellen, dass die Trägerkinematik während der Momentenreferenzierung nicht bewegt wird und die Montagerichtung der zu referenzierenden Kinematik nicht von der konfigurierten Montagerichtung abweicht (z. B. aufgrund einer Schiefstellung der mobilen Plattform). Andernfalls ist die Sicherheitsintegrität der Referenzierung der Gelenkmomenten-Sensoren nicht gegeben.

### 13.10.3 Applikation für die Positions- und Momentenreferenzierung erstellen

#### Beschreibung

Folgende Punkte sind zu beachten, wenn die Applikation für die Momentenreferenzierung wegen nicht anfahrbarer Messposen geändert werden muss:

- Die Gelenkmomentwerte werden gemessen, während der Roboter steht. Zwischen dem Erreichen einer Messpose und der Messung muss eine Wartezeit von mindestens 2,5 Sekunden eingehalten werden, in der der Roboter nicht verfährt. Bei zu geringen Wartezeiten kann die Referenzierungsgenauigkeit aufgrund von Schwingungen der Roboterstruktur reduziert werden.
- Mit der Methode sendSafetyCommand() wird die Messung gestartet.
- Zwischen 2 aufeinanderfolgenden Messungen dürfen maximal 15 s liegen.

#### Vorgehensweise

1. Sunrise-Projekt im **Paket-Explorer** markieren.
2. Menüfolge **Datei > Neu > Andere...** wählen.
3. Im Ordner **Sunrise** die Option **Applikation für Positions- und GMS-Referenzierung des LBR iiwa** markieren und auf **Fertigstellen** klicken. Die Applikation **PositionAndGMSReferencing.java** wird im Quellenordner des Projekts erstellt und im Editoren-Bereich von Sunrise.Workbench geöffnet.
4. Wenn Messposen aufgrund der Anlagenkonfiguration nicht anfahrbar sind, diese in der Applikation anpassen.
5. Projekt synchronisieren, um die Applikation auf die Robotersteuerung zu übertragen.

## 13.11 Übersicht Sicherheitsabnahme

Die Anlage darf erst nach einer erfolgreichen Sicherheitsabnahme betrieben werden. Für eine erfolgreiche Sicherheitsabnahme müssen die Punkte in den Checklisten vollständig vom Sicherheitsinbetriebnehmer abgearbeitet und schriftlich bestätigt werden.

**!** Die abgearbeiteten und schriftlich bestätigten Checklisten sind als Nachweis aufzubewahren.

Die Sicherheitsabnahme muss in folgenden Fällen durchgeführt werden:

- Nach der Erst- oder Wiederinbetriebnahme des Industrieroboters
- Nach einer Änderung des Industrieroboters
- Nach einer Änderung in der Sicherheitskonfiguration
- Nach einem Software-Update, z. B. der System Software

Die Sicherheitsabnahme nach einem Software-Update ist nur notwendig, wenn sich durch das Update die ID der Sicherheitskonfiguration (= Prüfsumme) geändert hat.

Auf Basis der Risikoanalyse ermittelt der Systemintegrator die erforderlichen Sicherheitsfunktionen. Die korrekte Funktion der Sicherheitsfunktionen ist nach der Aktivierung der Sicherheitskonfiguration auf der Robotersteuerung zu testen.



Wenn es bei einem Test erforderlich ist, sich im Gefahrenbereich aufzuhalten, ist der Test in der Betriebsart T1 durchzuführen.

Anhand der folgenden Checklisten ist zu überprüfen, ob die konfigurierten Sicherheitsparameter korrekt übernommen wurden.

Die Checklisten sind in der hier angegebenen Reihenfolge abzuarbeiten:

1. Checkliste für grundlegende Prüfung der Sicherheitskonfiguration  
(>>> 13.11.1 "Checkliste Sicherheitsfunktionen Allgemein" Seite 244)
2. Checklisten zum Prüfen des sicherheitsgerichteten Werkzeugs  
(>>> 13.11.2 "Checklisten Sicherheitsgerichtetes Werkzeug" Seite 247)
3. Checkliste zum Prüfen sicherheitsgerichteter Werkstücke  
(>>> 13.11.3 "Checkliste Sicherheitsgerichtete Werkstücke" Seite 248)
4. Checkliste zum Prüfen der verwendeten Zeilen in der PSM-Tabelle *KUKA PSM* und in der PSM-Tabelle *Anwender PSM*  
(>>> 13.11.4 "Checkliste Verwendete Zeilen in den PSM-Tabellen" Seite 249)
5. Checklisten zum Prüfen der verwendeten und nicht verwendeten ESM-Zustände  
(>>> 13.11.5 "Checklisten ESM-Zustände" Seite 250)
6. Checklisten zum Prüfen der verwendeten AMFs  
(>>> 13.11.6 "Checklisten Verwendete AMFs" Seite 252)

Es kann ein Bericht zur aktuellen Sicherheitskonfiguration erstellt werden.

(>>> 13.11.7 "Bericht zur Sicherheitskonfiguration erstellen" Seite 259)

### 13.11.1 Checkliste Sicherheitsfunktionen Allgemein

#### Checkliste

- Seriennummer des Roboters: \_\_\_\_\_
- ID der Sicherheitskonfiguration: \_\_\_\_\_
- Name des Sicherheitsinbetriebnehmers: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Bedienerschutz: Sind alle Bedienerschutz-Einrichtungen konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		
2	Bedienerschutz: Es wird ein Stopp ausgelöst, wenn bei geöffnetem Bedienerschutz die Betriebsart AUT oder T2 aktiv ist.		
3	Bedienerschutz: Eine manuelle Rückstellfunktion ist vorhanden und funktionsfähig.		
4	Zustimmung Handführgerät: Ist die Zustimmleinrichtung des Handführgeräts konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		
5	NOT-HALT lokal: Sind alle lokalen NOT-HALT-Einrichtungen konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		
6	NOT-HALT extern: Sind alle externen NOT-HALT-Einrichtungen konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		

Nr.	Tätigkeit	Ja	Nicht relevant
7	NOT-HALT lokal und extern: Sind der lokale und externe NOT-HALT jeweils als einzige AMF in einer Zeile der PSM-Tabelle konfiguriert?		
8	Sicherheitshalt: Sind alle Sicherheitshalt-Einrichtungen konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		
9	Sicherer Betriebshalt: Sind alle Einrichtungen für den sicheren Betriebshalt konfiguriert, richtig angeschlossen und auf Funktionsfähigkeit getestet?		
10	Bei Verwendung positionsbasierter AMFs: Ist die eingeschränkte Sicherheitsintegrität der positionsbasierten AMFs bei fehlender Positionsreferenzierung berücksichtigt?  (>>> "Positionsbasierte AMFs" Seite 246)  <b>Hinweis:</b> Die Einleitung des sicheren Zustands bei fehlender Positionsreferenzierung kann mit Verwendung der AMF <i>Positionsreferenzierung</i> konfiguriert werden.		
11	Bei Verwendung positionsbasierter AMFs: Wurde die Positionsreferenzierung erfolgreich durchgeführt?		
12	Geschwindigkeitsüberwachung: Wurden alle erforderlichen Geschwindigkeitsüberwachungen konfiguriert und getestet?		
13	Handführen: Ist die Konfiguration so gestaltet, dass bei Handführen in jeder Betriebsart eine angemessene Geschwindigkeitsüberwachung aktiv ist?		
14	Raumüberwachung: Wurden alle erforderlichen Raumüberwachungen konfiguriert und getestet?		
15	Kartesische Raumüberwachungen: Wurde berücksichtigt, dass nicht die gesamte Struktur des Roboters, Werkzeugs und Werkstücks sondern lediglich die Überwachungskugeln an Roboter und Werkzeug gegen die Raumverletzung geprüft werden?		
16	Kollisionserkennung: Wurden alle erforderlichen MRK-Funktionalitäten konfiguriert?		
17	Kollisionserkennung: Ist die Konfiguration so gestaltet, dass bei aktiver Kollisionserkennung immer auch eine Geschwindigkeitsüberwachung aktiv ist?		
18	Kollisionserkennung: Ist die Konfiguration so gestaltet, dass bei aktiver TCP-Kraftüberwachung immer auch eine Geschwindigkeitsüberwachung aktiv ist?		
19	Kollisionserkennung: Ist bei allen Sicherheitsüberwachungen zur Erkennung von Quetschsituationen ein Sicherheitshalt 0 konfiguriert?		
20	Bei Verwendung achsmomentbasierter AMFs: Ist die eingeschränkte Sicherheitsintegrität der achsmomentbasierten AMFs bei fehlender Positionsreferenzierung und/oder Momentenreferenzierung berücksichtigt?  (>>> "Achsmomentbasierte AMFs" Seite 247)  <b>Hinweis:</b> Die Einleitung des sicheren Zustands bei fehlender Positions- und/oder Momentenreferenzierung kann mit Verwendung der AMF <i>Positionsreferenzierung</i> und der AMF <i>Momentenreferenzierung</i> konfiguriert werden.		

Nr.	Tätigkeit	Ja	Nicht relevant
21	<p>Wurde bei der Konfiguration aller Zeilen der PSM-Tabelle und ESM-Zustände berücksichtigt, dass der sichere Zustand der AMFs der Zustand "verletzt" (Zustand "0") ist?</p> <p><b>Hinweis:</b> Eine AMF geht im Fehlerfall in den sicheren Zustand.</p>		
22	PSM-Konfiguration: Wurde bei der Konfiguration von Ausgangssignalen als Sicherheitsreaktion berücksichtigt, dass ein Ausgang im sicheren Zustand LOW-Pegel (Zustand "0") hat?		
23	ESM-Konfiguration: Sind alle ESM-Zustände in sich konsistent, d. h. reduziert jeder einzelne ESM-Zustand alle Gefährdungen ausreichend?		
24	Wurden die Momentenreferenzierung und die Positionsreferenzierung erfolgreich durchgeführt?		
25	<p>Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit):</p> <p>Wurde berücksichtigt, dass bei der AMF <i>Kartesische Arbeitsraumüberwachung / Kartesische Schutzraumüberwachung</i> der Überwachungsraum relativ zur Basis der überwachten Kinematik definiert ist, d. h. mit der Trägerkinematik mitbewegt wird?</p>		
26	<p>Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit):</p> <p>Wurde berücksichtigt, dass bei der AMF <i>Kartesische Geschwindigkeitsüberwachung</i> nicht die Absolutgeschwindigkeit, sondern die Relativgeschwindigkeit der überwachten Kinematik zur Trägerkinematik überwacht wird?</p>		
27	<p>Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit):</p> <p>Wurde berücksichtigt, dass bei der AMF <i>Werkzeugbezogene Geschwindigkeitskomponente</i> nicht die absolute Geschwindigkeit, sondern die Geschwindigkeit der überwachten Kinematik relativ zur Trägerkinematik überwacht wird?</p>		
28	<p>Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform):</p> <p>Wurde berücksichtigt, dass bei der AMF <i>Werkzeugorientierung</i> die Referenzorientierung relativ zur Trägerkinematik definiert ist, d. h. mit der Trägerkinematik mitbewegt wird?</p>		
29	<p>Bei Befestigung der überwachten Kinematik auf einer Trägerkinematik (z. B. mobile Plattform, Lineareinheit):</p> <p>Wurde berücksichtigt, dass die Sicherheitsintegrität der AMFs <i>Kollisionserkennung</i> und <i>TCP-Kraftüberwachung</i> nur gewährleistet ist, solange die Trägerkinematik im Stillstand ist?</p>		

Ort, Datum	
Unterschrift	

Der Unterzeichner bestätigt mit seiner Unterschrift die korrekte und vollständige Durchführung der Sicherheitsabnahme.

**Positionsba-  
sierte AMFs**

Die Sicherheitsintegrität positionsbasierter AMFs ist nur dann uneingeschränkt gegeben, wenn die Positionsreferenzierung erfolgreich durchgeführt worden ist. (Bei einer mobilen Plattform ist keine Referenzierung erforderlich.)

<b>AMF</b>	<b>Positionsreferenzierung</b>	<b>Momentenreferenzierung</b>
<i>Stillstandsüberwachung aller Achsen</i>	✓	✗
<i>Achsbereichsüberwachung</i>	✓	✗
<i>Kartesische Geschwindigkeitsüberwachung</i>	✓	✗
<i>Werkzeugbezogene Geschwindigkeitskomponente</i>	✓	✗
<i>Kartesische Arbeitsraumüberwachung</i>	✓	✗
<i>Kartesische Schutzraumüberwachung</i>	✓	✗
<i>Werkzeugorientierung</i>	✓	✗

**Achsmomentbasierte AMFs** Die Sicherheitsintegrität achsmomentbasierter AMFs ist nur dann uneingeschränkt gegeben, wenn Positions- und/oder Momentenreferenzierung erfolgreich durchgeführt worden sind. (Bei einer mobilen Plattform ist keine Referenzierung erforderlich.)

<b>AMF</b>	<b>Positionsreferenzierung</b>	<b>Momentenreferenzierung</b>
<i>Achsmomentenüberwachung</i>	✗	✓
<i>Kollisionserkennung</i>	✓	✓
<i>TCP-Kraftüberwachung</i>	✓	✓

## 13.11.2 Checklisten Sicherheitsgerichtetes Werkzeug

### 13.11.2.1 Geometriedaten des sicherheitsgerichteten Werkzeugs

- Beschreibung** Wenn eine der folgenden AMFs in der Sicherheitskonfiguration verwendet wird, muss überprüft werden, ob die geometrischen Werkzeugdaten korrekt angegeben wurden:
- *Kartesische Geschwindigkeitsüberwachung*
  - *Kartesische Arbeitsraumüberwachung*  
Nur bei Räumen, bei denen die Überwachungskugeln am Werkzeug als zu überwachende Struktur konfiguriert sind.
  - *Kartesische Schutzraumüberwachung*  
Nur bei Räumen, bei denen die Überwachungskugeln am Werkzeug als zu überwachende Struktur konfiguriert sind.
- Die geometrischen Werkzeugdaten können getestet werden, indem einer der konfigurierten Überwachungsräume mit jeder Werkzeugkugel bewusst verletzt wird und die Reaktion geprüft wird.
- Wenn keine Raumüberwachungen verwendet werden, ist nur die Lage der Kugelmittelpunkte relevant. Die konfigurierte kartesische Geschwindigkeitsgrenze kann getestet werden, indem sie für jede Werkzeugkugel bewusst überschritten wird und die Reaktion geprüft wird.
- Voraussetzung**
- Die Positionsreferenzierung wurde erfolgreich durchgeführt.

**Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		
2	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		
3	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		
4	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		
5	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		
6	Werkzeugkugel (Frame-Name): _____ Radius und Lage der Werkzeugkugel sind korrekt angegeben und überprüft?		

**13.11.2.2 Lastdaten des sicherheitsgerichteten Werkzeugs****Beschreibung**

Wenn eine der folgenden AMFs in der Sicherheitskonfiguration verwendet wird, muss überprüft werden, ob die Lastdaten des sicherheitsgerichteten Werkzeugs korrekt angegeben wurden.

- *Kollisionserkennung*
- *TCP-Kraftüberwachung*

Es wird empfohlen, die Lastdaten durch eine Momentenreferenzierung an mehreren geeigneten Posen zu überprüfen. Geeignet sind beispielsweise Posen mit ähnlichen Achswinkeln in horizontaler Strecklage mit folgenden Eigenschaften:

- Die Achsen A2, A4 und A6 sind belastet.
- Die Posen unterscheiden sich im Achswert von A7 um 90°.

Bei korrekten Lastdaten muss die Momentenreferenzierung erfolgreich sein.

**Voraussetzung**

- Positions- und Momentenreferenzierung wurden erfolgreich durchgeführt.
- Wenn zur Überprüfung der Lastdaten ein sicherheitsgerichtetes Werkstück vom Werkzeug aufgenommen wird: Das korrekte sicherheitsgerichtete Werkstück ist aktiv.

**Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Lastdaten des Werkzeugs sind korrekt angegeben und überprüft?		

**13.11.3 Checkliste Sicherheitsgerichtete Werkstücke****Beschreibung**

Wenn eine der folgenden AMFs in der Sicherheitskonfiguration verwendet wird, muss überprüft werden, ob die Lastdaten der sicherheitsgerichteten Werkstücke korrekt angegeben wurden.

- *Kollisionserkennung*
- *TCP-Kraftüberwachung*

Es wird empfohlen, die Lastdaten durch eine Momentenreferenzierung an mehreren geeigneten Posen zu überprüfen. Geeignet sind beispielsweise Posen mit ähnlichen Achswinkeln in horizontaler Strecklage mit folgenden Eigenschaften:

- Die Achsen A2, A4 und A6 sind belastet.
- Die Posen unterscheiden sich im Achswert von A7 um 90°.

Bei korrekten Lastdaten muss die Momentenreferenzierung erfolgreich sein.

#### **Voraussetzung**

- Die Positions- und Momentenreferenzierung wurden erfolgreich durchgeführt.
- Das korrekte sicherheitsgerichtete Werkstück ist aktiv.

#### **Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
2	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
3	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
4	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
5	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
6	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
7	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		
8	Name des Werkstücks: _____ Lastdaten des Werkstücks sind korrekt angegeben und überprüft?		

#### **13.11.4 Checkliste Verwendete Zeilen in den PSM-Tabellen**

##### **Beschreibung**

Es ist für jede Zeile in der PSM-Tabelle *KUKA PSM* und in der PSM-Tabelle *Anwender PSM* zu prüfen, ob die erwartete Reaktion ausgelöst wird. Wenn die Reaktion das Abschalten eines Ausgangs ist, ist durch den Test zusätzlich sicherzustellen, dass der Ausgang korrekt angeschlossen wurde.

Eine Zeile der PSM-Tabelle kann getestet werden, indem jeweils 2 AMFs verletzt sind. Dann kann gezielt die verbleibende AMF einzeln getestet werden. Wenn weniger als 3 AMFs in einer Zeile verwendet werden, gelten die unbeliebten Spalten als verletzte AMFs.

(>>> 13.11.6 "Checklisten Verwendete AMFs" Seite 252)



Die Punkte in der Checkliste müssen für jede Zeile der PSM-Tabelle abgearbeitet und separat dokumentiert werden.

### Checkliste

■ Zeilen-Nr.: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	AMF 1 wurde erfolgreich getestet. Voraussetzung: AMF 2 und AMF 3 sind verletzt. AMF 1: _____		
2	AMF 2 wurde erfolgreich getestet. Voraussetzung: AMF 1 und AMF 3 sind verletzt. AMF 2: _____		
3	AMF 3 wurde erfolgreich getestet. Voraussetzung: AMF 1 und AMF 2 sind verletzt. AMF 3: _____		

### 13.11.5 Checklisten ESM-Zustände

#### 13.11.5.1 Verwendete ESM-Zustände

##### Beschreibung

Es ist für jede Zeile eines ESM-Zustands zu prüfen, ob die erwartete Reaktion ausgelöst wird, wenn die konfigurierte AMF verletzt ist.

(>>> 13.11.6 "Checklisten Verwendete AMFs" Seite 252)



Die Punkte in der Checkliste müssen für jeden verwendeten ESM-Zustand abgearbeitet und separat dokumentiert werden.

### Checkliste

■ ESM-Zustand: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	AMF Zeile 1 wurde erfolgreich getestet. AMF Zeile 1: _____		
2	AMF Zeile 2 wurde erfolgreich getestet. AMF Zeile 2: _____		
3	AMF Zeile 3 wurde erfolgreich getestet. AMF Zeile 3: _____		
4	AMF Zeile 4: wurde erfolgreich getestet. AMF Zeile 4: _____		
5	AMF Zeile 5 wurde erfolgreich getestet. AMF Zeile 5: _____		
6	AMF Zeile 6 wurde erfolgreich getestet. AMF Zeile 6: _____		
7	AMF Zeile 7 wurde erfolgreich getestet. AMF Zeile 7: _____		
8	AMF Zeile 8 wurde erfolgreich getestet. AMF Zeile 8: _____		

Nr.	Tätigkeit	Ja	Nicht relevant
9	AMF Zeile 9 wurde erfolgreich getestet. AMF Zeile 9: _____		
10	AMF Zeile 10 wurde erfolgreich getestet. AMF Zeile 10: _____		
11	AMF Zeile 11 wurde erfolgreich getestet. AMF Zeile 11: _____		
12	AMF Zeile 12 wurde erfolgreich getestet. AMF Zeile 12: _____		
13	AMF Zeile 13 wurde erfolgreich getestet. AMF Zeile 13: _____		
14	AMF Zeile 14 wurde erfolgreich getestet. AMF Zeile 14: _____		
15	AMF Zeile 15 wurde erfolgreich getestet. AMF Zeile 15: _____		
16	AMF Zeile 16 wurde erfolgreich getestet. AMF Zeile 16: _____		
17	AMF Zeile 17 wurde erfolgreich getestet. AMF Zeile 17: _____		
18	AMF Zeile 18 wurde erfolgreich getestet. AMF Zeile 18: _____		
19	AMF Zeile 19 wurde erfolgreich getestet. AMF Zeile 19: _____		
20	AMF Zeile 20 wurde erfolgreich getestet. AMF Zeile 20: _____		

### 13.11.5.2 Nicht verwendete ESM-Zustände

**Beschreibung** Für alle nicht verwendeten ESM-Zustände muss getestet werden, ob bei der Anwahl des ESM-Zustands ein Sicherheitshalt ausgelöst wird.

#### Checkliste

Nr.	Tätigkeit	Ja	Nicht relevant
1	Anwahl des nicht verwendeten ESM-Zustands 1 wurde erfolgreich getestet.		
2	Anwahl des nicht verwendeten ESM-Zustands 2 wurde erfolgreich getestet.		
3	Anwahl des nicht verwendeten ESM-Zustands 3 wurde erfolgreich getestet.		
4	Anwahl des nicht verwendeten ESM-Zustands 4 wurde erfolgreich getestet.		
5	Anwahl des nicht verwendeten ESM-Zustands 5 wurde erfolgreich getestet.		
6	Anwahl des nicht verwendeten ESM-Zustands 6 wurde erfolgreich getestet.		
7	Anwahl des nicht verwendeten ESM-Zustands 7 wurde erfolgreich getestet.		

Nr.	Tätigkeit	Ja	Nicht relevant
8	Anwahl des nicht verwendeten ESM-Zustands 8 wurde erfolgreich getestet.		
9	Anwahl des nicht verwendeten ESM-Zustands 9 wurde erfolgreich getestet.		
10	Anwahl des nicht verwendeten ESM-Zustands 10 wurde erfolgreich getestet.		

### 13.11.6 Checklisten Verwendete AMFs

Eine AMF, die in mehreren Zeilen der PSM-Tabelle verwendet wird, muss in jeder Zeile einzeln getestet werden.

#### 13.11.6.1 AMF NOT-HALT smartPAD

##### Checkliste

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird beim Drücken des NOT-HALTs am smartPAD ausgelöst.		

#### 13.11.6.2 AMF Zustimmung smartPAD inaktiv

##### Checkliste

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird beim Loslassen eines Zustimmungsschalters am smartPAD ausgelöst.		

#### 13.11.6.3 AMF Zustimmung Panik smartPAD aktiv

##### Checkliste

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird beim Durchdrücken eines Zustimmungsschalters am smartPAD ausgelöst.		

#### 13.11.6.4 AMF Zustimmung Handführgerät inaktiv

##### Checkliste

- Verwendeter Eingang Zustimmungsschalter 1: \_\_\_\_\_
- Verwendeter Eingang Zustimmungsschalter 2: \_\_\_\_\_
- Verwendeter Eingang Zustimmungsschalter 3: \_\_\_\_\_
- Verwendeter Eingang Panikschalter 1: \_\_\_\_\_
- Verwendeter Eingang Panikschalter 2: \_\_\_\_\_
- Verwendeter Eingang Panikschalter 2: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird beim Loslassen eines Zustimmungsschalters am Handführgerät ausgelöst.		
2	Die konfigurierte Reaktion wird beim Durchdrücken eines Zustimmungsschalters am Handführgerät ausgelöst.		

**13.11.6.5AMF Zustimmung Handführgerät aktiv****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird beim Drücken eines Zustimmungsschalters am Handführgerät ausgelöst.		

**13.11.6.6AMF Betriebsart Test****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird in T1 ausgelöst.		
2	Die konfigurierte Reaktion wird in T2 ausgelöst.		
3	Die konfigurierte Reaktion wird in KRF ausgelöst.		

**13.11.6.7AMF Betriebsart Automatik****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird in AUT ausgelöst.		

**13.11.6.8AMF Betriebsart mit reduzierter Geschwindigkeit****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird in T1 ausgelöst.		
2	Die konfigurierte Reaktion wird in KRF ausgelöst.		

**13.11.6.9AMF Betriebsart mit hoher Geschwindigkeit****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird in T2 ausgelöst.		
2	Die konfigurierte Reaktion wird in AUT ausgelöst.		

**13.11.6.10AMF Fahrfreigabe****Checkliste**

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn beispielsweise der NOT-HALT am smartPAD gedrückt wird.		

**13.11.6.11AMF Eingangssignal****Checkliste**

- Verwendeter Eingang: \_\_\_\_\_
- Instanz des verwendeten Eingangs: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn der Eingang LOW-Pegel (Zustand "0") hat.		

### 13.11.6.12AMF Stillstandsüberwachung aller Achsen

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn eine Achse der überwachten Kinematik bewegt wird.		

### 13.11.6.13AMF Achsmomentenüberwachung

**Beschreibung**

Die AMF kann getestet werden, indem am smartPAD die aktuell gemessenen Achsmomente angezeigt werden und die überwachte Achse dann durch Schwerkraft oder manuell belastet wird.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Überwachte Achse: \_\_\_\_\_
- Maximal zulässiges Achsmoment: \_\_\_\_\_
- Minimal zulässiges Achsmoment: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn das maximal zulässige Achsmoment überschritten wird.		
2	Die konfigurierte Reaktion wird ausgelöst, wenn das minimal zulässige Achsmoment unterschritten wird.		

### 13.11.6.14AMF Achsgeschwindigkeitsüberwachung

**Beschreibung**

Um die AMF zu testen, ist die überwachte Achse mit einer Geschwindigkeit von ca. 10 % über der konfigurierten Geschwindigkeitsgrenze zu bewegen.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Überwachte Achse: \_\_\_\_\_
- Maximal zulässige Achsgeschwindigkeit: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn die maximal zulässige Achsgeschwindigkeit überschritten wird.		

### 13.11.6.15AMF Positionsreferenzierung



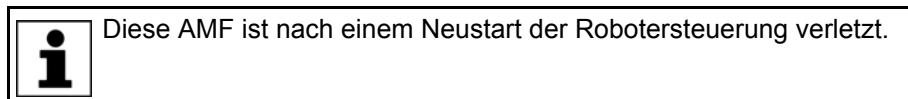
Diese AMF ist nach einem Neustart der Robotersteuerung verletzt.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn eine oder mehrere Achsen der überwachten Kinematik nicht referenziert sind.		

### **13.11.6.16AMF Momentenreferenzierung**



#### **Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn eine oder mehrere Achsen der überwachten Kinematik nicht referenziert sind.		

### **13.11.6.17AMF Achsbereichsüberwachung**

#### **Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Überwachte Achse: \_\_\_\_\_
- Untere Grenze des erlaubten Achsbereichs: \_\_\_\_\_
- Obere Grenze des erlaubten Achsbereichs: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn die untere Grenze des erlaubten Achsbereichs unterschritten wird.		
2	Die konfigurierte Reaktion wird ausgelöst, wenn die obere Grenze des erlaubten Achsbereichs überschritten wird.		

### **13.11.6.18AMF Kartesische Geschwindigkeitsüberwachung**

#### **Beschreibung**

Um die AMF zu testen, ist ein überwachter Punkt der überwachten Kinematik mit einer kartesischen Geschwindigkeit von ca. 10 % über der konfigurierten Geschwindigkeitsgrenze zu bewegen.

#### **Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Maximal zulässige kartesische Geschwindigkeit: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn die Maximal zulässige kartesische Geschwindigkeit eines überwachten Punktes überschritten wird.		

### **13.11.6.19AMF Kartesische Arbeitsraumüberwachung / Kartesische Schutzraumüberwachung**

#### **Beschreibung**

Im ersten Schritt ist zu testen, ob die Orientierung des Überwachungsraums korrekt konfiguriert ist. Dazu sind 2 aneinander grenzende Raumflächen an jeweils mindestens 3 unterschiedlichen Punkten zu verletzen.

Im zweiten Schritt ist zu testen, ob die Ausdehnung des Überwachungsraums korrekt konfiguriert ist. Dazu sind die anderen Raumflächen an jeweils 1 Punkt zu verletzen. Insgesamt müssen mindestens 10 Punkte angefahren werden.

Im dritten Schritt ist zu testen, ob die zu überwachende Struktur korrekt konfiguriert ist. Dazu ist die Raumüberwachung sowohl mit den Überwachungskugeln am Roboter als auch am Werkzeug zu verletzen, sofern beide Strukturen

überwacht werden sollen, oder nur mit den Überwachungskugeln am Roboter oder am Werkzeug.

**Checkliste**

- Art des Überwachungsraums: \_\_\_\_\_
- Instanz des Überwachungsraums: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Überwachte Struktur: \_\_\_\_\_
- Verschiebung des Ursprungs des Überwachungsraums:
  - X: \_\_\_\_\_ mm
  - Y: \_\_\_\_\_ mm
  - Z: \_\_\_\_\_ mm
- Orientierung des Ursprungs des Überwachungsraums:
  - A: \_\_\_\_\_ °
  - B: \_\_\_\_\_ °
  - C: \_\_\_\_\_ °
- Länge des Überwachungsraums: \_\_\_\_\_ mm
- Breite des Überwachungsraums: \_\_\_\_\_ mm

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die korrekte Konfiguration des Überwachungsraums wurde wie oben beschrieben getestet und die konfigurierte Reaktion wird ausgelöst, wenn der Überwachungsraum verletzt ist.		
2	Die konfigurierte Reaktion wird ausgelöst, wenn die Raumüberwachung von den Überwachungskugeln am Roboter verletzt wird.		
3	Die konfigurierte Reaktion wird ausgelöst, wenn die Raumüberwachung von den Überwachungskugeln am Werkzeug verletzt wird.		

**13.11.6.20AMF Kollisionserkennung****Beschreibung**

Die AMF kann getestet werden, indem am smartPAD die aktuell gemessenen externen Achsmomente angezeigt werden und dann einzelne Achsen belastet werden.

**Voraussetzung**

- Die Momentenreferenzierung wurde erfolgreich durchgeführt.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Maximal zulässiges externes Achsmoment: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn das externe Moment einer oder mehrere Achsen der überwachten Kinematik das maximal zulässige externe Moment überschreitet.		

**13.11.6.21AMF TCP-Kraftüberwachung****Beschreibung**

Um die AMF zu testen, ist ein geeignetes Messmittel erforderlich, z. B. eine Federwaage.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Maximal zulässige externe TCP-Kraft: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn die externe Kraft, die auf den TCP wirkt, die maximal zulässige Kraft überschreitet.		

### 13.11.6.22AMF Zeitverzögerung

#### Checkliste

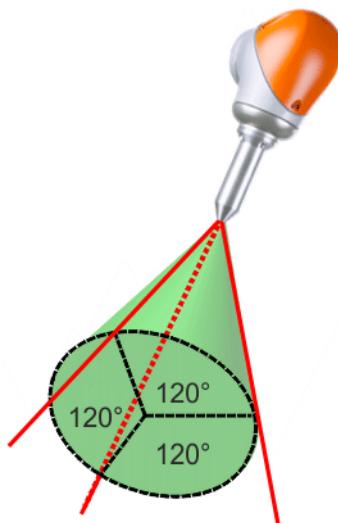
- Instanz der Zeitverzögerung: \_\_\_\_\_
- Verzögerungszeit: \_\_\_\_\_

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird nach der konfigurierten Zeit ausgelöst.		

### 13.11.6.23AMF Werkzeugorientierung

#### Beschreibung

Zum Überprüfen der AMF ist der zulässige Orientierungskegel an 3 um ca. 120° versetzte zueinander liegenden Geraden zu verletzen. Hiermit wird sichergestellt, dass der zulässige Orientierungswinkel, die Orientierung des Referenzvektors und die Werkzeugorientierung korrekt konfiguriert sind.



**Abb. 13-21: Lage der Geraden am Überwachungskegel**

Über 3 um 120° zueinander versetzte Geraden, die am Rand des Überwachungskegels liegen, werden die Orientierungswinkel der Z-Achse des Werkzeugorientierungs-Frames festgelegt, die eingenommen werden müssen, um die AMF *Werkzeugorientierung* zu testen. Die AMF muss bei Überschreiten aller 3 Werkzeugorientierungs-Winkel verletzt sein.

#### Vorgehensweise

Die Vorgehensweise beschreibt beispielhaft, wie die korrekte Konfiguration des Überwachungskegels überprüft werden kann.

1. Z-Achse des Werkzeugorientierungs-Frames gemäß Referenzvektor bezüglich des Welt-Koordinatensystems ausrichten.
2. Zulässigen Abweichungswinkel durch Kippen des Werkzeugorientierungs-Frames in B oder C überschreiten.  
Die konfigurierte Reaktion muss ausgelöst werden.
3. Z-Achse des Werkzeugorientierungs-Frames gemäß Referenzvektor bezüglich des Welt-Koordinatensystems ausrichten.

Wurde eine Stopp-Reaktion konfiguriert, ist zum Verfahren des Roboters ein Umschalten in Betriebsart KRF erforderlich.

4. Werkzeugorientierungs-Frame um 120° in A drehen.
5. Zulässigen Abweichungswinkel durch Kippen des Werkzeugorientierungs-Frames in B oder C überschreiten.  
Die konfigurierte Reaktion muss ausgelöst werden.
6. Z-Achse des Werkzeugorientierungs-Frames gemäß Referenzvektor bezüglich des Welt-Koordinatensystems ausrichten.  
Wurde eine Stopp-Reaktion konfiguriert, ist zum Verfahren des Roboters ein Umschalten in Betriebsart KRF erforderlich.
7. Werkzeugorientierungs-Frame um 120° in A drehen.
8. Zulässigen Abweichungswinkel durch Kippen des Werkzeugorientierungs-Frames in B oder C überschreiten.  
Die konfigurierte Reaktion muss ausgelöst werden.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Orientierung des Referenzvektors bezüglich des Welt-Koordinatensystems:
  - A: \_\_\_\_\_ °
  - B: \_\_\_\_\_ °
  - C: \_\_\_\_\_ °
- Zulässiger Arbeitsbereich (Abweichungswinkel): \_\_\_\_\_ °

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die korrekte Konfiguration des Überwachungskegels wurde überprüft und die konfigurierte Reaktion wird bei Überschreitung des zulässigen Winkels für alle 3 Geraden ausgelöst.		

**13.11.6.24AMF Werkzeugbezogene Geschwindigkeitskomponente****Beschreibung**

Für den Test ist eine Bewegung mit dem konfigurierten Punkt für die werkzeugbezogene Geschwindigkeitskomponente zu programmieren. Es ist darauf zu achten, dass die Testbewegungen eine Umorientierung des Werkzeugs beinhalten, um die korrekte Konfiguration des überwachten Punkts zu überprüfen.

Der Test ist 2-mal durchzuführen:

- Mit einer Geschwindigkeit, die knapp über der maximal zulässigen Geschwindigkeit liegt.
- Mit einer Geschwindigkeit, die knapp unter der maximal zulässigen Geschwindigkeit liegt.

Dadurch soll sichergestellt werden, dass die Geschwindigkeitsgrenze nur durch den konfigurierten überwachten Punkt verletzt wird.

**Checkliste**

- Instanz der Überwachung: \_\_\_\_\_
- Überwachte Kinematik: \_\_\_\_\_
- Überwachte Komponente des Geschwindigkeitsvektors:  
\_\_\_\_\_
- Maximal zulässige kartesische Geschwindigkeit der überwachten Komponente: \_\_\_\_\_
- Orientierung der überwachten Komponente des Geschwindigkeitsvektors bezogen auf das Flansch-Koordinatensystem:
  - A: \_\_\_\_\_ rad
  - B: \_\_\_\_\_ rad
  - C: \_\_\_\_\_ rad

- Überwachter Punkt bezogen auf das Flansch-Koordinatensystem:
  - X: \_\_\_\_\_ mm
  - Y: \_\_\_\_\_ mm
  - Z: \_\_\_\_\_ mm

Nr.	Tätigkeit	Ja	Nicht relevant
1	Die konfigurierte Reaktion wird ausgelöst, wenn die Bewegung mit einer Geschwindigkeit durchgeführt wird, die über der maximal zulässigen Geschwindigkeit liegt.		
2	Die konfigurierte Reaktion wird nicht ausgelöst, wenn die Bewegung mit einer Geschwindigkeit durchgeführt wird, die unter der maximal zulässigen Geschwindigkeit liegt.		

### 13.11.7 Bericht zur Sicherheitskonfiguration erstellen

**Beschreibung** Es kann ein Bericht zur aktuellen Sicherheitskonfiguration erstellt und im Editor angezeigt werden. Der Bericht ist editierbar und kann zur Dokumentation gedruckt werden.

Der Bericht zur Sicherheitskonfiguration stellt passend zur Sicherheitskonfiguration folgende Checklisten bereit:

- Checkliste zum Prüfen der verwendeten Zeilen der Anwender PSM-Tabelle
- Checklisten zum Prüfen der verwendeten und nicht verwendeten ESM-Zustände
- Checklisten zum Prüfen der verwendeten AMFs



Die vom Bericht zur Sicherheitskonfiguration bereitgestellten Checklisten sind nicht ausreichend für eine vollständige Sicherheitsabnahme. Folgende weitere Checklisten sind für eine vollständige Sicherheitsabnahme zu prüfen:

- Checkliste für grundlegende Prüfung der Sicherheitskonfiguration
- Checklisten zum Prüfen des sicherheitsgerichteten Werkzeugs
- Checkliste zum Prüfen der sicherheitsgerichteten Werkstücke

Der Bericht zur Sicherheitskonfiguration enthält folgende Informationen, um die Sicherheitskonfiguration eindeutig zuordnen zu können:

- Name des Sunrise-Projekts, zu dem die Sicherheitskonfiguration gehört
- Verwendete Safety-Version
- Safety-ID (Prüfsumme der Sicherheitskonfiguration)  
Die Safety-ID muss mit der ID der Sicherheitskonfiguration übereinstimmen, die auf der Robotersteuerung aktiviert ist und überprüft werden soll.
- Datum und Uhrzeit der letzten Änderung der Sicherheitskonfiguration

**Vorgehensweise** ■ Im **Paket-Explorer** auf das gewünschte Projekt rechtsklicken und im Kontextmenü **Sunrise > Sicherheitskonfigurationsbericht erstellen** wählen.

Der Bericht zur aktuellen Sicherheitskonfiguration wird erstellt und im Editoren-Bereich geöffnet.



## 14 Grundlagen der Bewegungsprogrammierung

In diesem Kapitel werden die theoretischen Grundlagen der Bewegungsprogrammierung beschrieben.

Die Programmierung von Bewegungen in KUKA Sunrise.Workbench wird in folgendem Kapitel beschrieben: ([>>> 15 "Programmierung" Seite 283](#))

### 14.1 Bewegungsarten Übersicht



Der Startpunkt einer Bewegung ist immer der Zielpunkt der vorhergehenden Bewegung.

Folgende Bewegungsarten können als Einzelbewegung programmiert werden:

- Point-to-Point-Bewegung (PTP)  
([>>> 14.2 "Bewegungsart PTP" Seite 261](#))
- Linear-Bewegung (LIN)  
([>>> 14.3 "Bewegungsart LIN" Seite 262](#))
- Circular-Bewegung (CIRC)  
([>>> 14.4 "Bewegungsart CIRC" Seite 262](#))
- Handführ-Bewegung mit Handführgerät  
([>>> 14.7 "Bewegungsart Handführen" Seite 269](#))

Folgende Bewegungsarten können als Segmente eines CP-Spline-Blocks programmiert werden:

- Linear-Bewegung (LIN)
- Circular-Bewegung (CIRC)
- Polynominal-Bewegung (SPL)

Folgende Bewegungsarten können als Segmente eines JP-Spline-Blocks programmiert werden:

- Point-to-Point-Bewegung (PTP)  
([>>> 14.6 "Bewegungsart Spline" Seite 263](#))

Folgende Bewegungen werden unter dem Begriff "CP-Bewegungen" ("Continuous Path") zusammengefasst:

- LIN, CIRC, SPL, CP-Spline-Blöcke

Folgende Bewegungen werden unter dem Begriff "JP-Bewegungen" ("Joint Path") zusammengefasst:

- PTP, JP-Spline-Blöcke

### 14.2 Bewegungsart PTP

Der Roboter führt den TCP entlang der schnellsten Bahn zum Zielpunkt. Die schnellste Bahn ist in der Regel nicht die kürzeste Bahn im Raum und somit keine Gerade. Da sich die Roboterachsen gleichzeitig und rotatorisch bewegen, können geschwungene Bahnen schneller ausgeführt werden als gerade Bahnen.

Der PTP ist eine schnelle Zustellbewegung. Der exakte Verlauf der Bewegung ist nicht vorhersehbar, aber solange die Rahmenbedingungen nicht verändert werden, immer gleich.

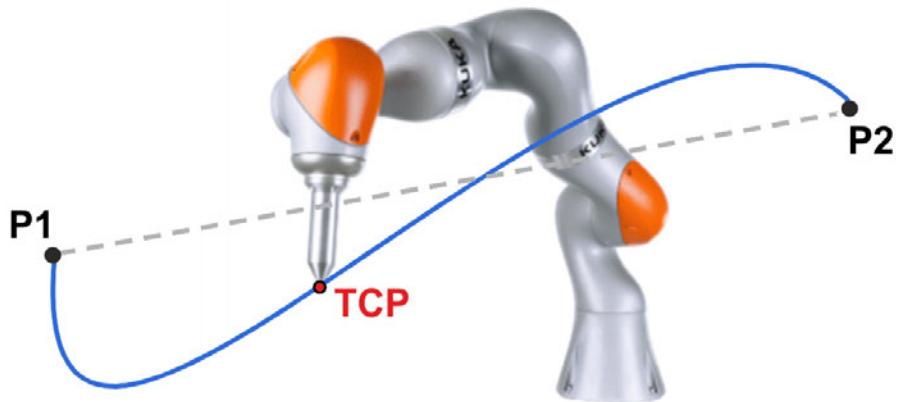


Abb. 14-1: PTP-Bewegung

#### 14.3 Bewegungsart LIN

Der Roboter führt den TCP mit der definierten Geschwindigkeit entlang einer Geraden im Raum zum Zielpunkt.

Bei der LIN-Bewegung wird die Roboterkonfiguration der Zielpose nicht berücksichtigt.

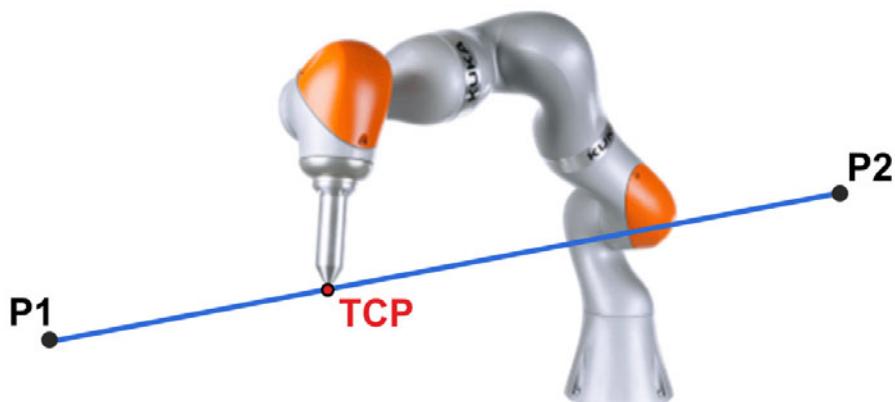
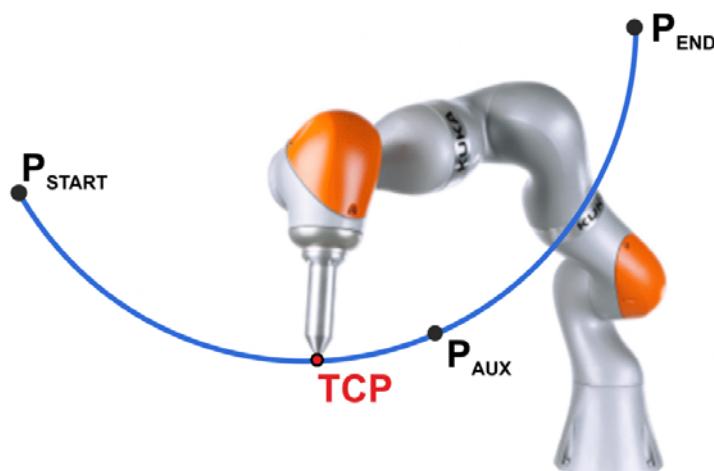


Abb. 14-2: LIN-Bewegung

#### 14.4 Bewegungsart CIRC

Der Roboter führt den TCP mit der definierten Geschwindigkeit entlang einer Kreisbahn zum Zielpunkt. Die Kreisbahn ist definiert durch Startpunkt, Hilfspunkt und Zielpunkt.

Bei einer CIRC-Bewegung wird die Roboterkonfiguration der Zielpose nicht berücksichtigt.



**Abb. 14-3: CIRC-Bewegung**

#### 14.5 Bewegungsart SPL

Mithilfe der Bewegungsart SPL werden kurvenförmige Bahnen generiert. SPL-Bewegungen werden immer in Spline-Blöcken zusammengefasst. Es entstehen Bahnen, die glatt durch die Zielpunkte der SPL-Bewegung verlaufen.

Bei der SPL-Bewegung wird die Roboterkonfiguration der Zielpose nicht berücksichtigt.



Erst durch das Zusammenschalten von 2 oder mehr SPL-Segmenten entstehen geschwungene Linien. Wird ein einzelnes SPL-Segment ausgeführt, verhält sich dieses wie ein LIN-Befehl.

#### 14.6 Bewegungsart Spline

Der Spline ist eine Bewegungsart, die besonders für komplexe geschwungene Bahnen geeignet ist. Mit einer Spline-Bewegung kann der Roboter diese komplexen Bahnen in einer kontinuierlichen Bewegung abfahren. Solche Bahnen können grundsätzlich auch mit überschliffenen LIN- und CIRC-Bewegungen erzeugt werden, der Spline hat jedoch Vorteile.

Splines werden in Spline-Blöcken programmiert. Mit einem Spline-Block fasst man mehrere Einzelbewegungen zu einer Gesamtbewegung zusammen. Der Spline-Block wird von der Robotersteuerung als 1 Bewegungssatz geplant und ausgeführt.

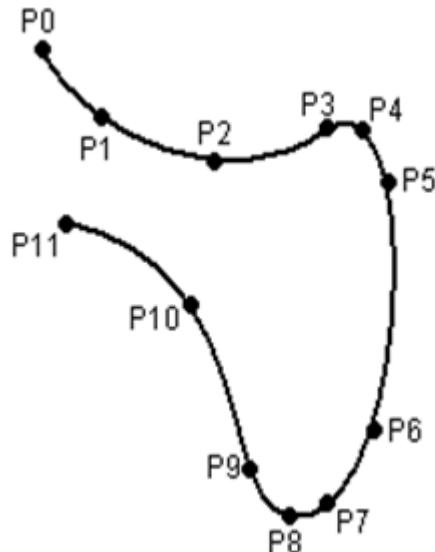
Die Bewegungen, die in einem Spline-Block enthalten sind, heißen Spline-Segmente.

- Ein CP-Spline-Block kann SPL-, LIN- und CIRC-Segmente enthalten.
- Ein JP-Spline-Block kann PTP-Segmente enthalten.

Bei einer kartesischen Spline-Bewegung wird die Roboterkonfiguration der Zielpose nicht berücksichtigt.

Die Konfiguration der Zielpose eines Spline-Segments ist abhängig von der Roboterkonfiguration am Start des Spline-Segments.

### Bahnverlauf Spline-Block



**Abb. 14-4: Geschwungene Bahn mit Spline-Block**

- Die Bahn wird definiert über Punkte, die auf der Bahn liegen. Diese Punkte sind die Zielpunkte der einzelnen Spline-Segmente.
  - Alle Punkte werden ohne Genauhalt durchfahren.  
Ausnahme: Die Geschwindigkeit wird auf 0 abgesenkt.  
(>>> 14.6.1 "Geschwindigkeitsprofil bei Spline-Bewegungen" Seite 264)
  - Liegen alle Punkte auf einer Ebene, liegt die Bahn auf dieser Ebene.
  - Liegen alle Punkte auf einer Geraden, ist die Bahn eine Gerade.
- In wenigen Fällen kommt es zu einer Geschwindigkeitsreduzierung.  
(>>> 14.6.1 "Geschwindigkeitsprofil bei Spline-Bewegungen" Seite 264)
- Der Bahnverlauf ist immer gleich, unabhängig von Override, Geschwindigkeit oder Beschleunigung.
- Kreise und enge Radien werden mit hoher Präzision gefahren.

#### 14.6.1 Geschwindigkeitsprofil bei Spline-Bewegungen

Die Robotersteuerung berücksichtigt bereits bei der Planung die physikalischen Grenzen des Roboters. Der Roboter bewegt sich im Rahmen der programmierten Geschwindigkeit so schnell wie möglich, d. h. so wie es seine physikalischen Grenzen erlauben.

Die Bahn verläuft immer gleich, unabhängig von Override, Geschwindigkeit oder Beschleunigung.

Nur Dynamikeffekte, die beispielsweise durch hohe Werkzeuglasten oder durch den Montagewinkel des Roboters entstehen, können bei unterschiedlichen Geschwindigkeiten geringfügige Bahnabweichungen hervorrufen.

#### Absenkung der Geschwindigkeit

Beim Spline wird in folgenden Fällen die programmierte Geschwindigkeit unterschritten:

- Ausgeprägte Ecken, z. B. aufgrund starker Richtungsänderung
- Große Umorientierungen
- In der Nähe von Singularitäten

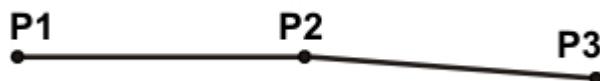
Eine Absenkung der Geschwindigkeit aufgrund von großen Umorientierungen kann man bei Spline-Segmenten vermeiden, indem man die Orientierungsführung `SplineOrientationType.Ignore` programmiert.

(>>> 14.9 "Orientierungsführung LIN, CIRC, SPL" Seite 272)

### Absenkung der Geschwindigkeit auf 0

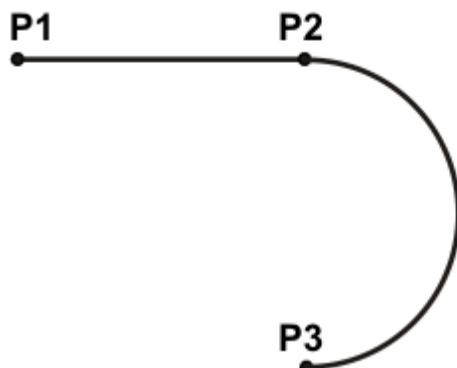
Beim Spline wird in folgenden Fällen ein Genauhalt ausgeführt:

- Aufeinanderfolgende Spline-Segmente mit gleichen Zielpunkten
- Aufeinanderfolgenden LIN- und/oder CIRC-Segmenten. Ursache: Unstetiger Verlauf der Geschwindigkeitsrichtung.



**Abb. 14-5: Genauhalt in P2**

Bei LIN-CIRC-Übergängen wird die Geschwindigkeit auch dann 0, wenn die Gerade tangential in den Kreis übergeht. Ursache dafür ist, dass im Übergangspunkt zwischen Gerade (Krümmung gleich 0) und Kreis (Krümmung ungleich 0) der Krümmungsverlauf unstetig ist.



**Abb. 14-6: Genauhalt in P2**

Ausnahmen:

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



**Abb. 14-7: P2 wird ohne Genauhalt durchfahren.**

- Bei einem CIRC-CIRC-Übergang wird die Geschwindigkeit nicht reduziert, wenn beide Kreise den gleichen Mittelpunkt und den gleichen Radius haben, und wenn sich die Orientierungen gleichmäßig ändern. Da die erforderliche Genauigkeit durch das Teachen der Ziel- und Hilfspunkte nur schwer zu erreichen ist, wird empfohlen, die Kreispunkte zu berechnen.

## 14.6.2 Änderungen an Spline-Blöcken

### Beschreibung

- Änderung der Punktposition:

Wenn ein Punkt innerhalb eines Spline-Blocks verschoben wird, ändert sich die Bahn maximal in den 2 Segmenten vor diesem Punkt und in den 2 Segmenten danach.

Kleine Punktverschiebungen ergeben in der Regel kleine Bahnänderungen. Wenn jedoch sehr lange und sehr kurze Segmente aufeinanderfolgen, können kleine Änderungen sehr große Auswirkungen haben.

- Änderung des Segmenttyps:

Wenn ein SPL-Segment in ein LIN-Segment geändert wird oder umgekehrt, ändert sich die Bahn im vorhergehenden Segment und im folgenden Segment.

### Beispiel 1

#### Ursprüngliche Bahn:

```
Spline mySpline = new Spline(
    spl(getApplicationData().getFrame("/P1")),
    spl(getApplicationData().getFrame("/P2")),
    spl(getApplicationData().getFrame("/P3")),
    spl(getApplicationData().getFrame("/P4")),
    circ(getApplicationData().getFrame("/P5")),
    getApplicationData().getFrame("/P6")),
    spl(getApplicationData().getFrame("/P7")),
    lin(getApplicationData().getFrame("/P8"))
);
...
robot.move(ptpgetApplicationData().getFrame("/P0"));
robot.move(mySpline);
```

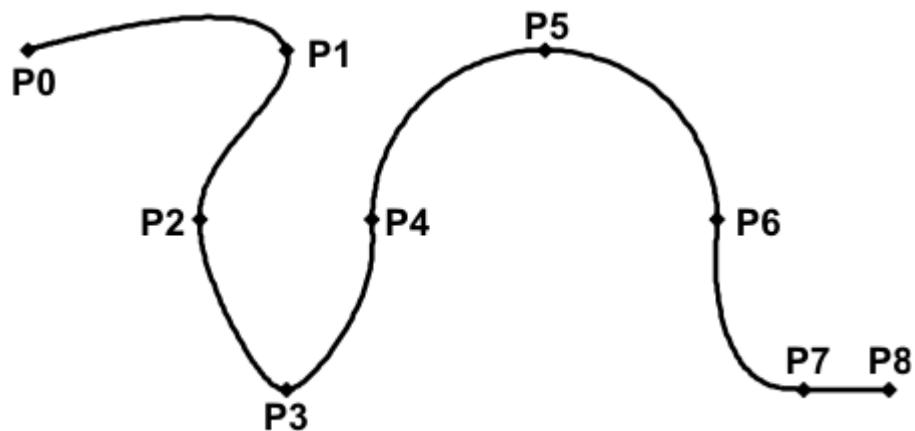
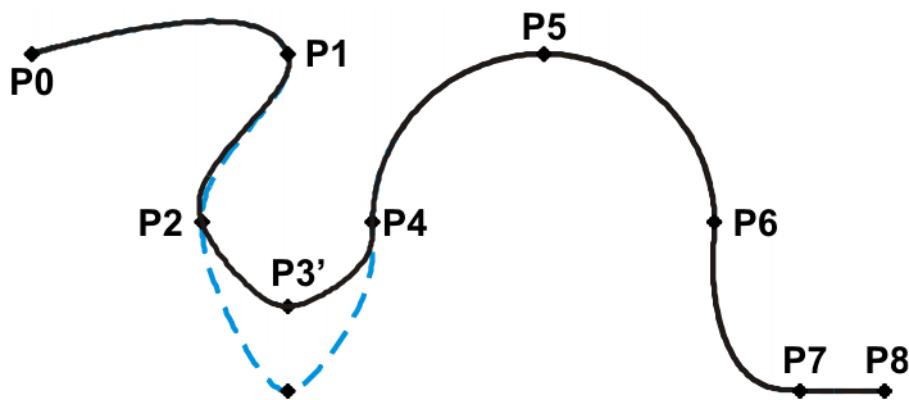


Abb. 14-8: Ursprüngliche Bahn

#### Gegenüber der ursprünglichen Bahn wird ein Punkt verschoben:

P3 wird verschoben. Dadurch ändert sich die Bahn in den Segmenten P1 - P2, P2 - P3 und P3 - P4. Das Segment P4 - P5 ändert sich in diesem Fall nicht, da es zu einem CIRC-Segment gehört und dadurch eine Kreisbahn festgelegt ist.

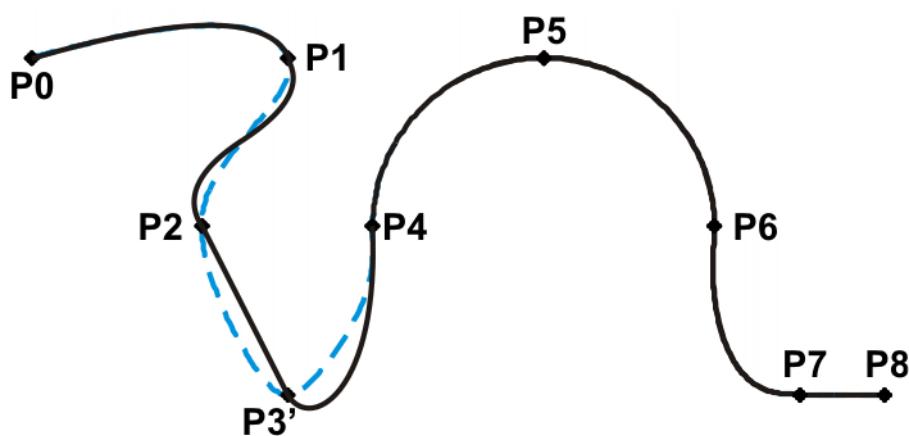


**Abb. 14-9: Punkt wurde verschoben**

**Gegenüber der ursprünglichen Bahn wird der Typ eines Segments geändert:**

Bei der ursprünglichen Bahn wird der Segmenttyp von P2 - P3 von SPL in LIN geändert. Die Bahn ändert sich in den Segmenten P1 - P2, P2 - P3 und P3 - P4.

```
Spline mySpline = new Spline(
    splgetApplicationData().getFrame("/P1")),
    splgetApplicationData().getFrame("/P2")),
    lin(getApplicationData().getFrame("/P3")),
    splgetApplicationData().getFrame("/P4")),
    circ(getApplicationData().getFrame("/P5")),
    getApplicationData().getFrame("/P6")),
    splgetApplicationData().getFrame("/P7")),
    lin(getApplicationData().getFrame("/P8"))
);
...
robot.move(ptp(getApplicationData().getFrame("/P0")));
robot.move(mySpline);
```



**Abb. 14-10: Segmenttyp wurde geändert**

## Beispiel 2

### Ursprüngliche Bahn:

```
Spline mySpline = new Spline(
    splgetApplicationData().getFrame("/P2")),
    splgetApplicationData().getFrame("/P3")),
```

```

spl(getApplicationData().getFrame("/P4")),
spl(getApplicationData().getFrame("/P5")),
);
...
robot.move(mySpline);

```



**Abb. 14-11: Ursprüngliche Bahn**

Folgende Frame-Koordinaten wurden gelehrt:

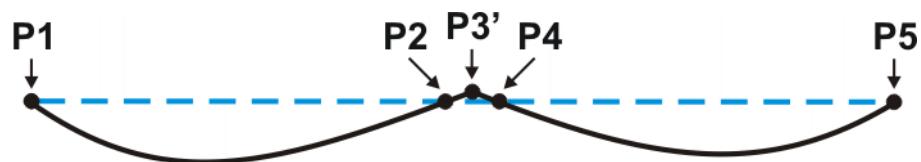
Frame	X	Y	Z
P2	100.0	0.0	0.0
P3	102.0	0.0	0.0
P4	104.0	0.0	0.0
P5	204.0	0.0	0.0

**Gegenüber der ursprünglichen Bahn wird ein Punkt verschoben:**

P3 wird geringfügig in Y-Richtung verschoben. Dadurch ändert sich die Bahn in allen dargestellten Segmenten.

Frame	X	Y	Z
P3	102.0	1.0	0.0

Da P2 - P3 und P3 - P4 sehr kurze Segmente und P1 - P2 und P4 - P5 lange Segmente sind, bewirkt die kleine Verschiebung ein starke Änderung der Bahn.



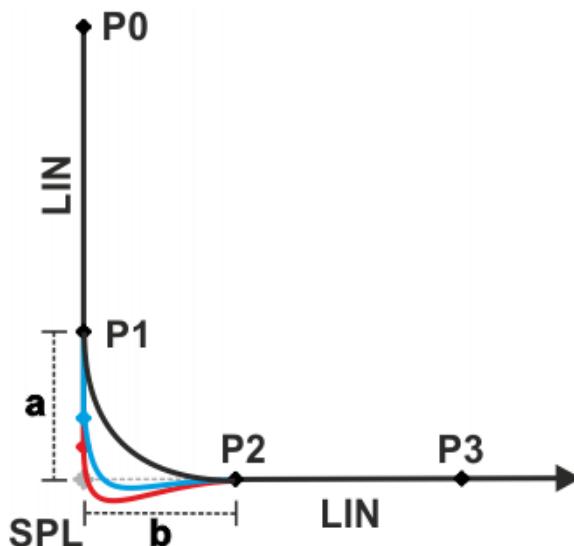
**Abb. 14-12: Punkt wurde verschoben**

Abhilfe:

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

### 14.6.3 LIN-SPL-LIN-Übergang

Bei einer Segmentfolge LIN-SPL-LIN ist es in der Regel erwünscht, dass das SPL-Segment innerhalb des kleineren Winkels zwischen den beiden Geraden verläuft. Abhängig von Start- und Zielpunkt des SPL-Segments kann die Bahn jedoch auch außerhalb verlaufen.



**Abb. 14-13: LIN-SPL-LIN**

Die Bahn verläuft innerhalb des kleineren Winkels, wenn folgende Voraussetzungen erfüllt sind:

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

## 14.7 Bewegungsart Handführen

**Beschreibung** Der Roboter lässt sich mithilfe eines Handführgerätes von Hand führen. Das Handführgerät ist eine Einrichtung, die zum Handführen des Roboters benötigt wird und mit einer Zustimmmeinrichtung ausgestattet ist.

Der Handführmodus kann mit dem Bewegungsbefehl `handGuiding()` in der Applikation eingeschaltet werden. Das Handführen beginnt an der Istposition, die vor dem Einschalten zuletzt erreicht wurde.

(>>> 15.10 "Handführen programmieren" Seite 304)

Im Handführmodus reagiert der Roboter nachgiebig auf äußere Kräfte und kann von Hand zu beliebigen Punkten im kartesischen Raum geführt werden. Beim Schalten in den Handführmodus werden die Impedanz-Parameter automatisch gesetzt. Es ist nicht möglich, die Impedanz-Parameter für das Handführen zu ändern.

**Voraussetzung**

- Handführgerät mit Zustimmungsschalter
- ESM-Zustand für Handführ-Bewegung ist konfiguriert.

Der ESM-Zustand enthält die AMF *Zustimmung Handführgerät inaktiv*, die prüft, ob die Zustimmung am Handführgerät nicht erteilt ist.

(>>> 13.8.5 "Handführen mit Zustimmmeinrichtung und Geschwindigkeitsüberwachung" Seite 209)

Wenn der Zustimmungsschalter am Handführgerät gedrückt und in Mittelstellung gehalten wird, kann der Roboter von Hand geführt werden. Wird der Zustimmungsschalter durchgedrückt oder losgelassen, wird die Zustimmung zum Handführen entzogen und der Roboter bleibt an seiner aktuellen Position stehen.

- ESM-Zustand für alle Bewegungen außer Handführ-Bewegung ist konfiguriert.  
Der ESM-Zustand enthält keine AMF *Zustimmung Handführgerät inaktiv*. Die Zustimmung am Handführgerät wird nicht ausgewertet.
- Betriebsart Automatik  
Die Sicherheitskonfiguration muss MRK-konform sein.



Beim Handführen können falsch gewählte Parameter (z. B. fehlerhafte Lastdaten, falsches Werkzeug), fehlerhafte Informationen (z. B. von defekten Momentensensoren) oder aufgeschaltete Zusatzkräfte als äußere Kräfte interpretiert werden und zu unvorhersehbaren Bewegungen des Roboters führen.



Wird die Zustimmung zum Handführen erteilt, bevor der Handführmodus in der Applikation eingeschaltet wird, ist der Handführmodus sofort nach dem Einschalten aktiv. D. h. die Bewegungsausführung wird beim Einschalten nicht pausiert, so dass der Übergang zwischen Applikations- und Handführmodus fließend ist.

Voraussetzung für dieses Verhalten: Die Applikationsgeschwindigkeit liegt unter der maximal zulässigen Geschwindigkeit, die für das Handführen konfiguriert ist.

(>>> 13.8.5.3 "Geschwindigkeitsüberwachung während des Handführens" Seite 211)

Wird die Applikation schneller abgefahren, wird die Applikation vor dem Schalten in den Handführmodus pausiert. (Dann Zustimmungsschalter loslassen, Start-Taste drücken und warten bis die Applikation erneut pausiert wird.)

## 14.8 Überschleifen

Überschleifen bedeutet: Der Zielpunkt einer programmierten Bewegung wird nicht genau angefahren, damit eine kontinuierliche Roboterbewegung möglich ist. Bei der Bewegungsprogrammierung stehen verschiedene Parameter zur Verfügung, um das Überschleifen zu beeinflussen.

Der Punkt, an dem die ursprüngliche Bahn verlassen wird und der Überschleifbogen beginnt, wird als Überschleipunkt bezeichnet.



Um Bewegungen ohne Genauhalt zu überschleifen, müssen diese asynchron ausgeführt oder in einem MotionBatch zusammengefasst werden.

(>>> 15.7.1 "Aufbau eines Bewegungsbefehls (move/moveAsync)" Seite 294)

(>>> 15.7.6 "MotionBatch" Seite 298)



Beim Überschleifen synchron ausgeführter Bewegungen wird am Beginn des Überschleifbogens ein Genauhalt ausgeführt. Dies gilt, bei synchroner Ausführung, auch für die letzte Bewegung innerhalb eines MotionBatch.

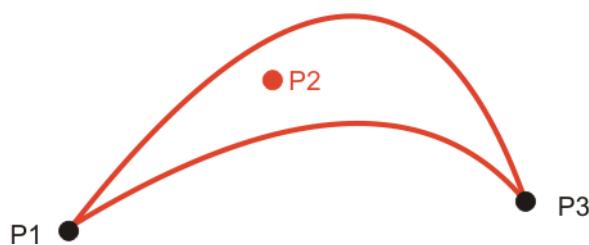
### PTP-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt stattdessen eine Bahn, mit der er den Zielpunkt ohne Genauhalt passieren kann. Damit verläuft die Bahn am Punkt vorbei und nicht mehr durch diesen hindurch.

Bei der Programmierung wird die relative Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn im Achsraum ab-

weichen darf. Eine relative Distanz von 100 % entspricht der gesamten Bahn von Start- bis Zielpunkt der Bewegung.

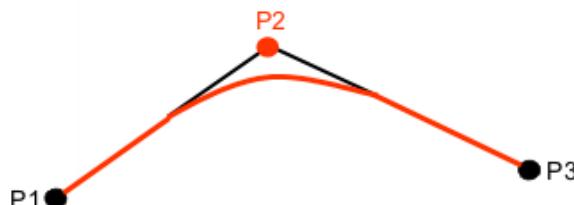
Die Überschleifikontur, die der TCP fährt, ist im kartesischen Raum nicht zwangsläufig die kürzere Bahn. Daher kann der überschliffene Punkt auch innerhalb des Überschleifbogens liegen.



**Abb. 14-14: PTP-Bewegung, P2 ist überschliffen**

#### LIN-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt eine kürzere Bahn. Beim Programmieren der Bewegung wird die Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn abweichen darf.

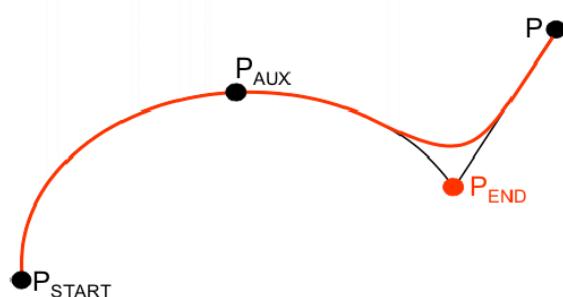


**Abb. 14-15: LIN-Bewegung, P2 ist überschliffen**

#### CIRC-Bewegung

Der TCP verlässt die Bahn, auf der er den Zielpunkt genau anfahren würde, und fährt eine kürzere Bahn. Beim Programmieren der Bewegung wird die Distanz zum Zielpunkt festgelegt, bei der der TCP frühestens von seiner ursprünglichen Bahn abweichen darf.

Es kann vorkommen, dass der Hilfspunkt in den Überschleifbereich fällt und nicht genau durchfahren wird. Dies ist abhängig von der Position des Hilfspunktes und den programmierten Überschleifparametern.



**Abb. 14-16: CIRC-Bewegung, P<sub>END</sub> ist überschliffen**

Alle Spline-Blöcke und alle Einzelbewegungen können miteinander überschließen werden. Es ist gleichgültig, ob es sich um CP- oder JP-Spline-Blöcke handelt oder um welche Einzelbewegung es sich handelt.

Der Überschleifbogen entspricht vom Bewegungstyp her immer der zweiten Bewegung. Beim PTP-LIN-Überschleifen z. B. ist der Überschleifbogen vom Typ CP.

Wird ein Spline-Block überschliffen, wird das gesamte letzte Segment überschliffen. Besteht der Spline-Block nur aus einem Segment, wird maximal das halbe Segment überschliffen (gilt auch für PTP, LIN und CIRC).

#### **Überschleifen nicht möglich wegen Zeit:**

Wenn Überschleifen wegen einer zeitlichen Verzögerung der Bewegungskommandierung nicht möglich ist, wartet der Roboter am Beginn des Überschleifbogens. Der Roboter fährt weiter, sobald der nächste Satz geplant werden konnte. Danach fährt der Roboter den Überschleifbogen. Das Überschleifen ist also genaugenommen möglich, es verzögert sich nur.

#### **Kein Überschleifen im Step-Betrieb:**

Im Step-Betrieb wird auch bei überschliffenen Bewegungen der Zielpunkt genau angefahren.

Beim Überschleifen von Spline-Block zu Spline-Block ist als Folge dieses Genauhalts die Bahn im letzten Segment des ersten Blocks und im ersten Segment des zweiten Blocks anders als im Standard-Betrieb.

Bei allen anderen Segmenten in den beiden Spline-Blöcken ist die Bahn in beiden Programmablaufarten gleich.

Überschliffene Bewegungen, die bereits vor Aktivierung des Step-Betriebs asynchron an die Robotersteuerung geschickt wurden und dort auf Ausführung warten, stoppen am Überschleipunkt. Für diese Bewegungen wird beim Fortsetzen noch der Überschleifbogen gefahren.

## **14.9 Orientierungsführung LIN, CIRC, SPL**

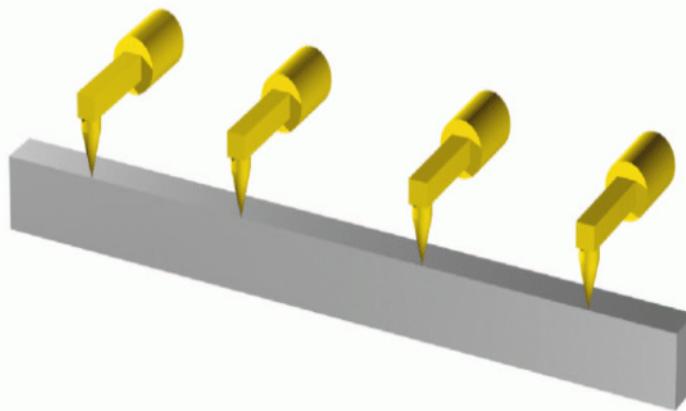
### **Beschreibung**

Der TCP kann am Start- und am Zielpunkt einer Bewegung unterschiedliche Orientierungen haben. Bei der Bewegungsprogrammierung kann festgelegt werden, wie mit den unterschiedlichen Orientierungen umgegangen werden soll.

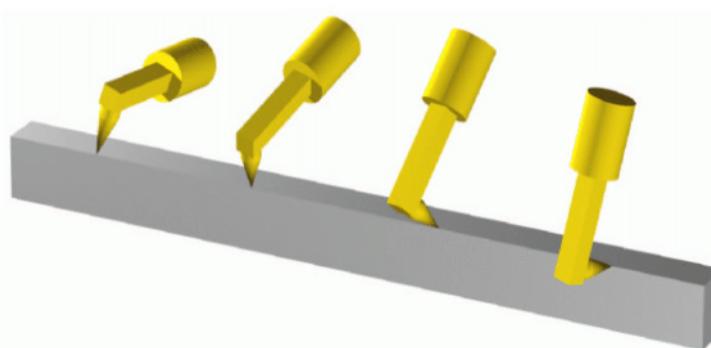
Die Orientierungsführung wird über die Methode `setOrientationType(...)` als Bewegungsparameter gesetzt. Die Orientierungsführung ist ein Wert vom Typ `Enum SplineOrientationType`.

Orientierungsführung	Beschreibung
Constant	<p>Die Orientierung des TCP bleibt während der Bewegung konstant.</p> <p>Die Orientierung des Startpunkts wird beibehalten. Die Orientierung des Zielpunkts wird nicht berücksichtigt.</p>
Ignore	<p>Die Orientierung des TCP ändert sich während der Bewegung.</p> <p>Diese Option steht nur für einzelne Spline-Segmente zur Verfügung, nicht für den gesamten Spline-Block oder Einzelbewegungen. Die Steuerung berechnet die Orientierungsführung auf Grundlage der Orientierung der umgebenden Stützpunkte, sofern deren Orientierung nicht ebenfalls ignoriert wird.</p> <p>Ignore wird verwendet, wenn für ein Spline-Segment keine bestimmte Orientierung erforderlich ist.</p> <p>(&gt;&gt;&gt; "Ignore" Seite 273)</p> <p><b>Hinweis:</b> Bei Ignore wird die Orientierung des Zielpunkts nicht berücksichtigt. Wenn die Einhaltung der geteachten Orientierung am Zielpunkt notwendig ist, um beispielsweise Kollisionen zu vermeiden, darf Ignore nicht verwendet werden.</p>

Orientierungsführung	Beschreibung
OriJoint	<p>Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich. Dies geschieht durch lineare Überführung (achsspezifisches Verfahren) der Handachsenwinkel.</p> <p><b>Hinweis:</b> OriJoint dann verwenden, wenn der Roboter mit VariableOrientation in eine Handachsen-Singularität gerät. Die Orientierung des TCP ändert sich während der Bewegung kontinuierlich, jedoch nicht ganz gleichmäßig. OriJoint ist deshalb nicht geeignet, wenn ein bestimmter Verlauf der Orientierung exakt gehalten muss, z. B. beim Laserschweißen.</p>
VariableOrientation	<p>Die Orientierung des TCP wird während der Bewegung kontinuierlich von der Orientierung des Startpunkts in die Orientierung des Zielpunkts überführt.</p> <p>Wenn die Orientierungsführung nicht gesetzt wird, gilt defaultmäßig diese Orientierungsführung.</p>



**Abb. 14-17: Konstante Orientierung (Constant)**



**Abb. 14-18: Variable Orientierung (VariableOrientation oder OriJoint)**

**CIRC-Bewegung** Für CIRC-Bewegungen kann festgelegt werden, ob die Orientierungsführung raumbezogen oder bahnbezogen sein soll.

(>>> 14.9.1 "CIRC – Bezugssystem der Orientierungsführung" Seite 274)

Bei CIRC-Bewegungen berücksichtigt die Robotersteuerung nur die Orientierung des Zielpunkts. Es kann festgelegt werden, ob und inwieweit die Orientierung des Hilfspunkts berücksichtigt wird. Außerdem kann das Orientierungsverhalten am Zielpunkt festgelegt werden.

**Ignore** Der Orientierungstyp SplineOrientationType.Ignore wird verwendet, wenn an einem Punkt keine bestimmte Orientierung erforderlich ist. Die Robotersteue-

rung ignoriert die geteachte oder programmierte Orientierung des Punktes. Stattdessen errechnet sie auf Grundlage der Orientierungen der umgebenden Punkte die optimale Orientierung für diesen Punkt. Dies verringert die Taktzeit.

### Beispiel:

```
robot.move(P0);
Spline path6 = new Spline(
    spl(P1),
    spl(P2),
    spl(P3).setOrientationType(SplineOrientationType.Ignore),
    spl(P4).setOrientationType(SplineOrientationType.Ignore),
    spl(P5),
    spl(P6)
),
...
robot.move(path6);
```

Die geteachte oder programmierte Orientierung von P3 und P4 wird ignoriert.

Für folgende Spline-Segmente ist SplineOrientationType.Ignore nicht erlaubt:

- Das erste und letzte Segment in einem Spline-Block
- CIRC-Segmente mit OrientationReferenceSystem.Path
- Segmente, auf die ein CIRC-Segment folgt mit OrientationReferenceSystem.Path
- Segmente, auf die ein Segment folgt mit SplineOrientationType.Constant
- Aufeinanderfolgende Segmente in einem Spline-Block, die den gleichen Zielpunkt besitzen

## 14.9.1 CIRC – Bezugssystem der Orientierungsführung

### Beschreibung

Für CIRC-Bewegungen kann festgelegt werden, ob die Orientierungsführung raumbezogen oder bahnbezogen sein soll.

Das Bezugssystem der Orientierungsführung wird über die Methode setOrientationReferenceSystem(...) als Bewegungsparameter gesetzt. Die Orientierungsführung ist ein Wert vom Typ Enum OrientationReferenceSystem.

Das Bezugssystem der Orientierungsführung kann nur vor dem Orientierungstyp angegeben werden.

Bezugssystem	Beschreibung
Base	Basisbezogene Orientierungsführung während der Kreisbewegung
Path	Bahnbezogene Orientierungsführung während der Kreisbewegung

### Beispiel

Bahnbezogene Kreisbewegung mit konstanter Orientierung:

```
robot.move(circ(P6, P7)
.setOrientationReferenceSystem(OrientationReferenceSystem.Path)
.setOrientationType(SplineOrientationType.Constant));
```

### Einschränkung

Für folgende Spline-Segmente ist OrientationReferenceSystem.Path nicht erlaubt:

- CIRC-Segmente mit SplineOrientationType.Ignore
- CIRC-Segmente, denen ein Segment mit SplineOrientationType.Ignore vorausgeht

## 14.9.2 CIRC – Kombinationen von Bezugssystem und Typ der Orientierungsführung

**i** Wenn das Bezugssystem der Orientierungsführung mit SplineOrientationType.OriJoint kombiniert wird, hat das Bezugssystem keinen Einfluss auf die Orientierungsführung.

### Bahnbezogene Kreisbewegung mit konstanter Orientierung:

- OrientationReferenceSystem.Path
- SplineOrientationType.Constant

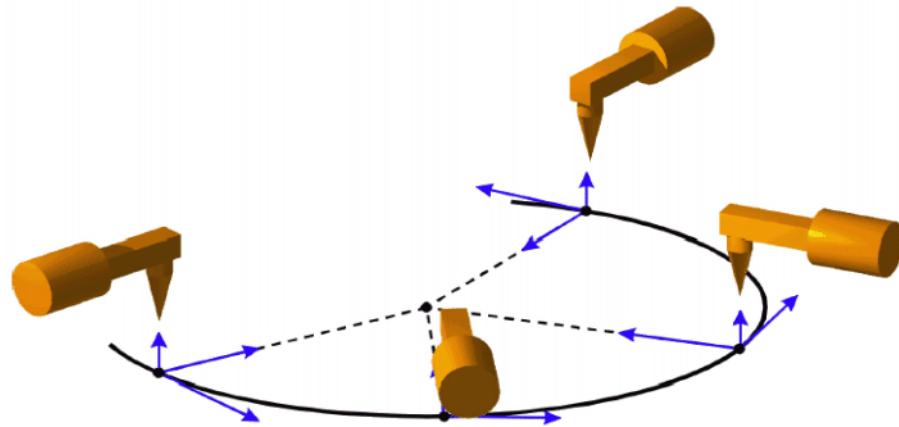


Abb. 14-19: Konstante Orientierung, bahnbezogen

### Bahnbezogene Kreisbewegung mit variabler Orientierung:

- OrientationReferenceSystem.Path
- SplineOrientationType.VariableOrientation

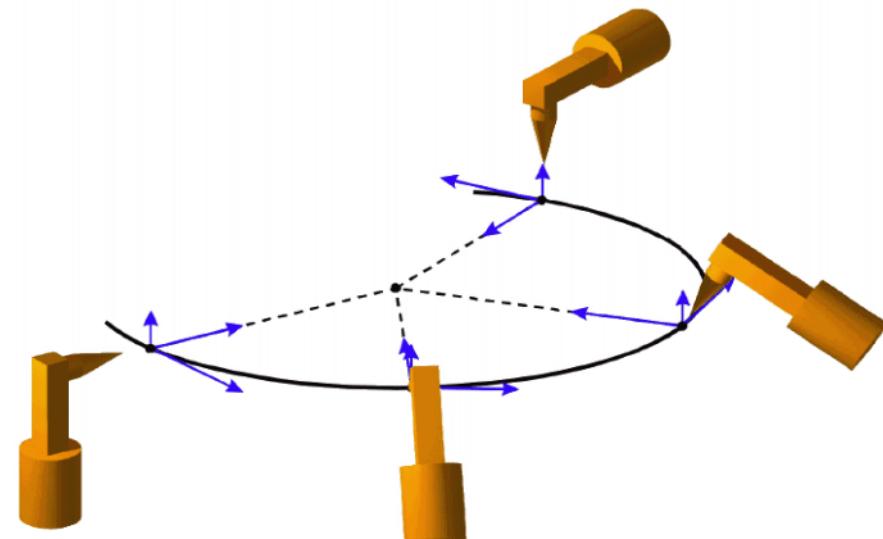


Abb. 14-20: Variable Orientierung, bahnbezogen

### Basisbezogene Kreisbewegung mit konstanter Orientierung:

- OrientationReferenceSystem.Base
- SplineOrientationType.Constant

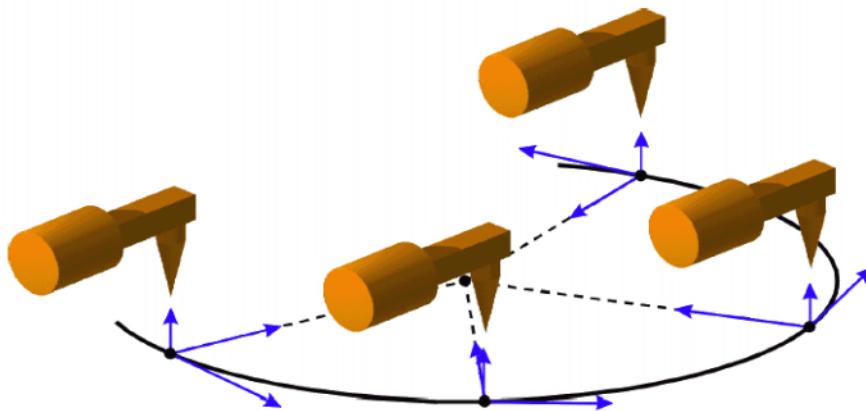


Abb. 14-21: Konstante Orientierung, basisbezogen

**Basisbezogene Kreisbewegung mit variabler Orientierung:**

- OrientationReferenceSystem.Base
- SplineOrientationType.VariableOrientation

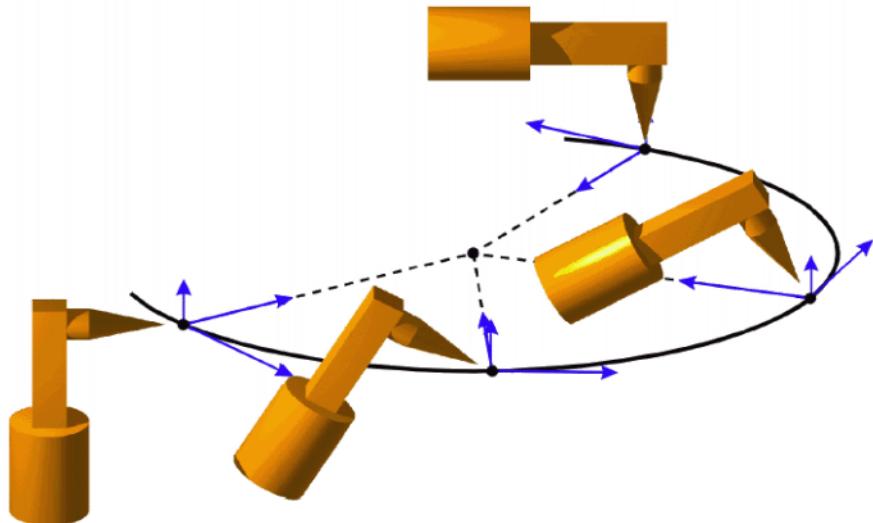


Abb. 14-22: Variable Orientierung, basisbezogen

## 14.10 Redundanzinformationen

Für eine gegebene Achsposition eines Roboters ist der resultierende Punkt im kartesischen Raum, an dem sich der TCP befindet, eindeutig definiert. Umgekehrt kann jedoch ausgehend von der kartesischen Position X, Y, Z und Orientierung A, B, C des TCP die Achsstellung des Roboters nicht eindeutig bestimmt werden. Ein kartesischer Punkt kann mit mehreren Achskonfigurationen erreicht werden. Um eine eindeutige Konfiguration bestimmen zu können, ist die Angabe des Status-Parameters erforderlich.

Die Mehrdeutigkeit der Achsstellungen für einen gegebenen kartesischen Punkt liegt bereits bei Robotern mit 6 Achsen vor. Der KUKA LBR iiwa kann durch seine zusätzliche 7. Achse eine gegebene Position und Orientierung mit theoretisch beliebig vielen Achsposen erreichen. Zur eindeutigen Bestimmung der Achspose ist bei einem LBR iiwa neben dem Status auch die Angabe des Redundanzwinkels erforderlich.

Der Parameter Turn wird für Achsen benötigt, die den Winkel  $\pm 180^\circ$  überfahren können. Er dient bei PTP-Bewegungen dazu, die Drehrichtung der Achsen eindeutig zu definieren. Auf CP-Bewegungen hat der Turn keinen Einfluss.

Status, Turn und Redundanzwinkel werden beim Teachen eines Frames gespeichert. Sie werden als Felder des Datentyps AbstractFrame verwaltet.

## Programmierung

Der Status eines Frames wird nur bei PTP-Bewegungen zu diesem Frame berücksichtigt. Bei CP-Bewegungen wird der Status verwendet, der durch die Achskonfiguration bei Bewegungsbeginn gegeben ist.

Um eine unvorhersehbare Bewegung zu Beginn einer Applikation zu vermeiden und eine eindeutige Achskonfiguration zu definieren, wird empfohlen, die erste Bewegung in einer Applikation mit einer der folgenden Anweisungen zu programmieren. Die Achskonfiguration sollte nicht in der Nähe einer singulären Achsstellung liegen.

- PTP-Bewegung an eine vorgegebene Achskonfiguration mit Angabe aller Achswerte:

```
ptp(double a1, double a2, double a3, double a4, double  
     a5, double a6, double a7)
```

- PTP-Bewegung an eine vorgegebene Achskonfiguration:

```
ptp(JointPosition joints)
```

- PTP-Bewegung an einen geteachten Frame (Typ: AbstractFrame):

```
ptp(getApplicationContext().getFrame(String frameName));
```

### 14.10.1 Redundanzwinkel

Der KUKA LBR iiwa ist durch seine 7. Achse in der Lage, einen Punkt im Raum mit einer theoretisch unbegrenzten Anzahl unterschiedlicher Achskonfigurationen zu erreichen. Über den Redundanzwinkel wird eine eindeutige Pose definiert.

Bei einem LBR iiwa enthält der Redundanzwinkel den Wert der 3. Achse.

Für alle Bewegungen gilt:

- Der Redundanzwinkel des Ziel-Frames wird berücksichtigt, wenn beim Teachen des Frames der Roboter verwendet wurde, der auch den Bewegungsbefehl ausführt. Insbesondere muss der in der Stationskonfiguration definierte Robotername mit dem in den Frame-Eigenschaften angegebenen Gerät übereinstimmen.
- Stimmen die Roboter nicht überein oder werden berechnete Frames verwendet, wird der durch die Achskonfiguration bei Bewegungsbeginn gegebene Redundanzwinkel beibehalten.

### 14.10.2 Status

Die Status-Angabe verhindert Mehrdeutigkeiten bei der Achsstellung. Der Status wird durch eine Binärzahl mit 3 Bit beschrieben.

#### Bit 0

Bit 0 gibt die Position des Handwurzelpunktes (Schnittpunkt der Achsen A5, A6, A7) bezogen auf die X-Achse des Koordinatensystems von Achse A1 an. Die Ausrichtung des A1-Koordinatensystems ist mit dem Roboterfuß-Koordinatensystem identisch, wenn die Achse A1 auf  $0^\circ$  steht. Es wird mit der Achse A1 mitbewegt.

Position	Wert
Überkopfbereich Der Roboter befindet sich im Überkopfbereich, wenn der x-Wert der Position des Handwurzelpunktes, bezogen auf das A1-Koordinatensystem, negativ ist.	Bit 0 = 1
Grundbereich Der Roboter befindet sich im Grundbereich, wenn der x-Wert der Position des Handwurzelpunktes, bezogen auf das A1-Koordinatensystem, positiv ist.	Bit 0 = 0

**Bit 1**

Bei einem LBR iiwa gibt Bit 1 die Position von Achse A4 an.

Position	Wert
A4 < 0°	Bit 1 = 1
A4 ≥ 0°	Bit 1 = 0

**Bit 2**

Bei einem LBR iiwa gibt Bit 2 die Position von Achse A6 an.

Position	Wert
A6 ≤ 0°	Bit 2 = 1
A6 > 0°	Bit 2 = 0

Für PTP-Bewegungen gilt:

- Der Status des Ziel-Frames wird berücksichtigt, wenn beim Teachen des Frames der Roboter verwendet wurde, der auch den Bewegungsbefehl ausführt. Insbesondere muss der in der Stationskonfiguration definierte Roboternname mit dem in den Frame-Eigenschaften angegebenen Gerät übereinstimmen.
- Stimmen die Roboter nicht überein oder werden berechnete Frames verwendet, wird der durch die Achskonfiguration bei Bewegungsbeginn gegebene Status beibehalten.

Für CP-Bewegungen gilt:

- Der Status des Ziel-Frames wird nicht berücksichtigt. Es wird der durch die Achskonfiguration bei Bewegungsbeginn gegebene Status beibehalten.
- **Ausnahme:** Eine Status-Änderung ist möglich, wenn der Ziel-Frame mit der Orientierungsführung SplineOrientationType.OriJoint angefahren wird. Auch in diesem Fall wird der Status des Ziel-Frames nicht berücksichtigt. Der Status bei Bewegungsende wird durch die Bahnplanung bestimmt, die den kürzesten Weg zur Zielposition wählt.

**14.10.3 Turn**

Die Turn-Angabe ermöglicht es, auch Achswinkel, die größer +180° oder kleiner -180° sind, ohne besondere Verfahrstrategie (z. B. Zwischenpunkte) anzufahren. Der Turn wird durch eine Binärzahl mit 7 Bit angegeben.

Die einzelnen Bits bestimmen bei rotatorischen Achsen das Vorzeichen des Achswertes folgendermaßen:

Bit = 0: Winkel ≥ 0°

Bit = 1: Winkel < 0°

Wert	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A7 $\geq 0^\circ$	A6 $\geq 0^\circ$	A5 $\geq 0^\circ$	A4 $\geq 0^\circ$	A3 $\geq 0^\circ$	A2 $\geq 0^\circ$	A1 $\geq 0^\circ$
1	A7 $< 0^\circ$	A6 $< 0^\circ$	A5 $< 0^\circ$	A4 $< 0^\circ$	A3 $< 0^\circ$	A2 $< 0^\circ$	A1 $< 0^\circ$

Bei einem LBR iiwa wird der Turn nicht berücksichtigt, da keine seiner Achsen über  $\pm 180^\circ$  drehen kann.

## 14.11 Singularitäten

Es ist möglich, dass aufgrund der Achsstellung kartesische Bewegungen des Roboters eingeschränkt werden. Grund dafür ist, dass durch die Kombination von Achsstellungen des gesamten Roboters keine Bewegungen von den Antrieben auf den Flansch (oder ein Objekt am Flansch, z. B. ein Werkzeug) in mindestens einer kartesischen Richtung übertragen werden können. In diesem Fall oder wenn kleinste kartesische Änderungen sehr große Achswinkeländerungen benötigen, spricht man von singulären Stellungen.



Es wird empfohlen, den Roboter in der Nähe von Singularitäten möglichst langsam bewegen.

### 14.11.1 Kinematische Singularitäten

Die Flexibilität durch die Redundanz eines 7-Achs-Roboters führt dazu, dass im Gegensatz zum 6-Achs-Roboter 2 oder mehr kinematische Bedingungen (z. B. Strecklage, 2 Drehachsen liegen aufeinander) gleichzeitig aktiv werden müssen, um eine singuläre Stellung zu erreichen. Es gibt 4 unterschiedliche Stellungen des Roboters, bei denen die Bewegung des Flansches in einer kartesischen Richtung nicht mehr möglich ist. Dabei ist jeweils nur die Stellung von einer oder 2 Achsen wichtig. Die anderen Achsen können eine beliebige Stellung einnehmen.

#### A4-Singularität

Diese kinematische Singularität ist mit  $A4 = 0^\circ$  gegeben. Sie wird als Streckposition bezeichnet.

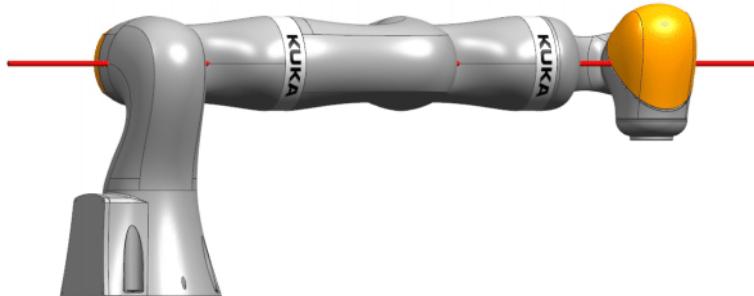


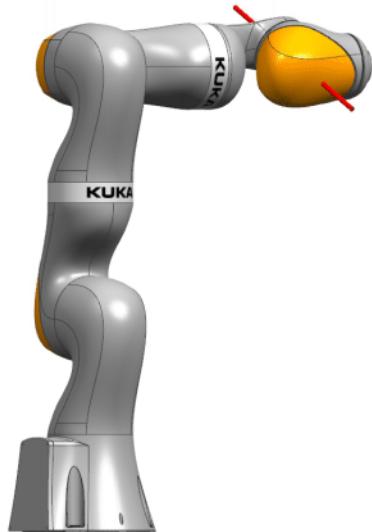
Abb. 14-23: Streckposition  $A4 = 0^\circ$

Die Bewegung ist in Richtung Roboterfuß blockiert oder parallel zur Achse A3 oder A5. Eine zusätzliche kinematische Bedingung für diese Singularität ist das Erreichen der Arbeitsraumgrenze. Sie ist durch  $A4 = 0^\circ$  automatisch erfüllt.

Bei gestrecktem Roboterarm geht ein Bewegungs-Freiheitsgrad für den Handwurzelpunkt verloren (er kann nicht mehr entlang der Achse des Roboterarms bewegt werden). Die Stellung der Achsen A3 und A5 ist nicht mehr auflösbar.

#### A4/A6-Singularität

Diese kinematische Singularität ist mit  $A4 = 90^\circ$  und  $A6 = 0^\circ$  gegeben.

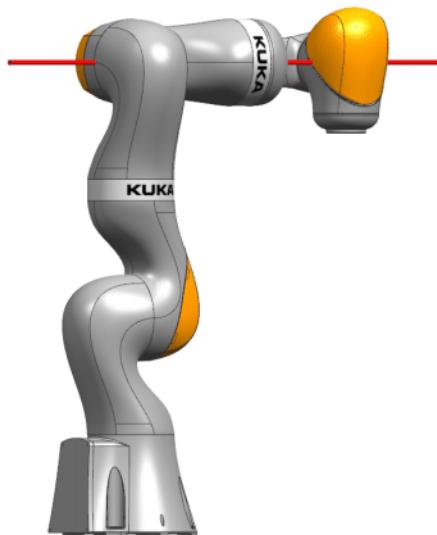


**Abb. 14-24:  $A4 = 90^\circ$  und  $A6 = 0^\circ$**

Die Bewegung parallel zur Achse A6 oder A2 ist blockiert.

#### A2/A3-Singulärheit

Diese kinematische Singularität ist mit  $A2 = 0^\circ$  und  $A3 = \pm 90^\circ (\pi/2)$  gegeben.



**Abb. 14-25:  $A2 = 0^\circ$  und  $A3 = \pm 90^\circ (\pi/2)$**

Die Bewegung ist in Richtung Roboter blockiert oder parallel zur Achse A2 oder A5.

#### A5/A6-Singulärheit

Diese kinematische Singularität ist mit  $A5 = \pm 90^\circ (\pi/2)$  und  $A6 = 0^\circ$  gegeben.



**Abb. 14-26:  $A_5 = \pm 90^\circ (\pi/2)$  und  $A_6 = 0^\circ$**

Die Bewegung parallel zur Achse A6 ist blockiert.

#### 14.11.2 Systembedingte Singularitäten

Dem LBR ist es durch seine redundante Konstruktion mit der 7. Achse möglich, dass sich der Roboterarm bewegt, ohne dass sich der Flansch bewegt. Bei dieser so genannten Nullraum-Bewegung bewegen sich alle Achsen außer A4, die "Ellenbogen-Achse". Zusätzlich zur normalen Redundanz ist es unter bestimmten Voraussetzungen möglich, dass sich nur Teilketten des Roboters bewegen und nicht alle Achsen.

Die Roboterstellungen, die unter diese Kategorie fallen, haben alle gemeinsam, dass bei kleinen kartesischen Änderungen sehr große Achswinkel-Änderungen hervorgerufen werden. Sie sind den Singularitäten sehr ähnlich, die von 6-Achs-Robotern bekannt sind, da auch beim LBR eine Aufspaltung in Positions- und Orientierungsteil des Handwurzelpunktes vorgenommen wird.

<b>Handachsen-Singularität</b>	Mit der Handachsen-Singularität bezeichnet man die Achsstellung $A_6 = 0^\circ$ . Damit ist die Stellung der Achsen A5 und A7 nicht mehr auflösbar und es gibt unendlich viele Möglichkeiten diese beiden Achsen zu stellen, um dieselbe Position am Flansch zu erzeugen.
<b>A1-Singularität</b>	Steht der Handwurzelpunkt direkt über A1, so kann man nach obiger Definition keinen Referenzwert für den Redundanzkreis angeben, da hier für $A_3 = 0^\circ$ ein beliebiger A1-Wert zulässig ist.  Jede Achsstellung von A1 kann durch eine Kombination von A5, A6 und A7 ausgeglichen werden, so dass die Flanschposition unverändert bleibt.
<b>A2-Singularität</b>	Bei gestreckter "Schulter" ist die Stellung der Achsen A1 und A3 nicht mehr nach obigem Muster auflösbar.
<b>A2/A4-Singularität</b>	Fallen A1 und A7 aufeinander, ist die Stellung der Achsen A1 und A7 nicht mehr nach obigem Muster auflösbar.



Systembedingte Singularitäten lassen sich in den meisten Fällen durch eine geeignete Ellenbogen-Stellung vermeiden.



# 15 Programmierung

## 15.1 Java-Editor

### 15.1.1 Roboter-Applikation im Java-Editor öffnen

**Beschreibung** Im Java-Editor können mehrere Dateien gleichzeitig geöffnet sein. Bei Bedarf können sie neben- oder untereinander angezeigt werden. Dies ermöglicht es z. B. Inhalte bequem zu vergleichen.

**Voraussetzung** ■ Roboter-Applikation ist angelegt.  
(>>> 5.4 "Neue Roboter-Applikation erstellen" Seite 54)

**Vorgehensweise**

1. Im **Paket-Explorer** auf eine Java-Datei doppelklicken.  
Oder: Die Datei markieren und Menüfolge **Navigieren > Öffnen** wählen.  
Oder: Auf die Datei rechtsklicken und im Kontextmenü **Öffnen** oder **Öffnen mit > Java-Editor** wählen.
2. Um die Datei zu schließen: Rechts oben auf der zugehörigen Registerkarte auf das "X" klicken.

### 15.1.2 Aufbau einer Roboter-Applikation

```

RobotApplication.java
package application;

import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;

public class RobotApplication extends RoboticsAPIApplication {
    private Controller kuka_Sunrise_Cabinet_1;
    private LBR lbr_iwa_7_R800_1;

    public void initialize() {
        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        lbr_iwa_7_R800_1 = (LBR) getRobot(kuka_Sunrise_Cabinet_1,
                                             "LBR_iwa_7_R800_1");
    }

    public void run() {
        lbr_iwa_7_R800_1.move(ptpHome());
    }

    /**
     * Auto-generated method stub. Do not modify the contents of this method.
     */
    public static void main(String[] args) {
        RobotApplication app = new RobotApplication();
        app.runApplication();
    }
}

```

Abb. 15-1: Aufbau einer Roboter-Applikation

Pos.	Beschreibung
1	Diese Zeile enthält den Namen des Pakets, in dem die Applikation liegt.
2	Der Bereich <b>import</b> enthält die importierten Klassen, die zur Programmierung der Applikation benötigt werden. <b>Hinweis:</b> Durch Klicken auf das "+"-Symbol öffnet man den Bereich und kann sich die importierten Klassen anzeigen lassen.
3	Kopfzeile der Applikation (enthält den Klassennamen der Applikation) (>>> "Kopfzeile" Seite 284)

Pos.	Beschreibung
4	Deklarations-Bereich  Hier werden die Datenfelder der Klassen deklariert, die in der Applikation benötigt werden. Beim Erstellen der Applikation wird automatisch je eine Instanz folgender Klassen angelegt:  ■ <b>Controller:</b> Verwendete Robotersteuerung ■ <b>LBR:</b> Verwendeter Roboter
5	Methode <b>initialize()</b>  In dieser Methode werden die im Deklarations-Bereich angelegten Datenfelder mit Anfangswerten belegt.
6	Methode <b>run()</b>  In dieser Methode wird der Roboter programmiert. Beim Erstellen der Applikation wird automatisch eine Bewegungsanweisung eingefügt, die den Roboter in die HOME-Position bringt.  (>>> 15.16 "HOME-Position" Seite 331)
7	Methode <b>main()</b>  Diese Methode ist notwendig, damit die Applikation ausgeführt werden kann. Sie darf nicht verändert werden.

**Kopfzeile**

Bei einer Roboter-Applikation handelt es sich um die Sonderform einer Java-Klasse:

```
public class RobotApplication extends RoboticsAPIApplication
```

Element	Beschreibung
<b>public</b>	Das Schlüsselwort <b>public</b> kennzeichnet eine Klasse, die öffentlich sichtbar ist. Öffentliche Klassen können paketübergreifend verwendet werden.
<b>class</b>	Das Schlüsselwort <b>class</b> kennzeichnet eine Java-Klasse. Der Name der Klasse wird aus dem Namen der Applikation gebildet.
<b>extends</b>	Die Applikation ist der Klasse <b>RoboticsAPIApplication</b> untergeordnet.

**15.1.3 Bearbeitungsfunktionen****15.1.3.1 Variablen umbenennen**

**Beschreibung** Ein Variablenname kann mit einer einzigen Aktion an sämtlichen Stellen, an denen er vorkommt, geändert werden.

**Vorgehensweise**

1. An einer beliebigen Stelle die gewünschte Variable markieren.
2. Rechtsklicken und im Kontextmenü **Refactoring > Umbenennen...** wählen.
3. Die Variable wird blau umrandet und ist jetzt bearbeitbar. Den Namen ändern und mit Eingabe-Taste bestätigen.

**15.1.3.2 Auto-Vervollständigen**

**Beschreibung** Im Java-Editor steht eine Auto-Vervollständigen-Funktionalität zur Verfügung. Beim Eintippen von Code kann man sich eine Vervollständigen-Liste mit Einträgen anzeigen lassen, die zu den bereits eingetippten Zeichen passen. Die-

se Einträge sind nach der Häufigkeit ihrer Verwendung priorisiert, d. h. die Auswahl passt sich ständig an das Benutzerverhalten an.

Bei Bedarf kann ein Eintrag aus der Vervollständigen-Liste in den Programmcode übernommen werden. Dies erübrigt es z. B., die komplexe Syntax von Methoden immer wieder tippen zu müssen. Nur die variablen Elemente der Syntax müssen dann noch manuell eingegeben werden.

#### Vorgehensweise

1. Beginnen den Code zu tippen.



Beim Eingeben eines Punktoperators an einem Datenfeld oder Enum wird die Vervollständigen-Liste automatisch angezeigt. Die Liste enthält folgende Einträge:

- Verfügbare Methoden der zugehörigen Klasse (nur bei Datenfeldern)
- Verfügbare Konstanten der zugehörigen Klasse

2. STRG + Leertaste drücken. Die Vervollständigen-Liste mit den zur Verfügung stehenden Einträgen wird angezeigt.



Wenn die Liste aus nur einem passenden Eintrag besteht, wird dieser nach Drücken von STRG + Leertaste automatisch in den Programmcode übernommen.

3. Den passenden Eintrag in der Liste markieren und die Eingabe-Taste drücken. Der Eintrag wird in den Programmcode übernommen.

Ist ein Eintrag markiert, werden die Javadoc-Informationen zu diesem Eintrag automatisch angezeigt.

(>>> 15.1.4 "Javadoc-Informationen anzeigen" Seite 287)

4. Bei Bedarf Syntax vervollständigen.

#### Navigieren und filtern

Es gibt verschiedene Möglichkeiten, um in der Vervollständigen-Liste zu navigieren und die zur Verfügung stehenden Einträge zu filtern:

- Mit den Pfeiltasten der Tastatur von einem Eintrag zum nächsten (nach oben oder unten)
- Scrollen
- Den eingetippten Code um weitere Zeichen ergänzen. Die Liste wird gefiltert und nur noch die Einträge angezeigt, zu denen die Zeichen passen.
- STRG + Leertaste drücken. Es werden ausschließlich die verfügbaren Schablonen-Vorschläge angezeigt.

### 15.1.3.3 Schablonen – Schnelleingabe für Java-Anweisungen

#### Beschreibung

Für gebräuchliche Java-Anweisungen, z. B. eine for-Schleife, stehen im Java-Editor Schablonen für eine Schnelleingabe zur Verfügung.

#### Vorgehensweise

1. Beginnen den Code zu tippen.
2. STRG + Leertaste drücken. Eine Liste mit den Schablonen-Vorschlägen, die zu den bereits eingegebenen Zeichen passen, wird angezeigt.
3. Die Anweisung mit der Eingabe-Taste übernehmen. Oder auf eine andere Anweisung doppelklicken.
4. Syntax vervollständigen.

#### Alternative Vorgehensweise

Schablonen in der Sicht **Schablonen** auswählen:

1. Menüfolge **Fenster > Sicht anzeigen > Andere...** wählen. Das Fenster **Sicht anzeigen** öffnet sich.
2. Im Ordner **Allgemein** den Eintrag **Schablonen** markieren. Mit **OK** bestätigen. Die Schablonen-Sicht öffnet sich.

3. Cursor im Java-Editor in die Zeile setzen, in die der vorgefertigte Code eingefügt werden soll.
4. Unter **Java-Anweisungen** auf die gewünschte Schablone in der Schablonen-Sicht doppelklicken. Der Code wird in den Editor eingefügt.
5. Syntax vervollständigen.

#### 15.1.3.4 Benutzerspezifische Schablonen erstellen

<b>Beschreibung</b>	Der Benutzer kann seine eigenen Schablonen erstellen, z. B. Schablonen für Bewegungssätze mit bestimmten Bewegungsparametern, die häufig beim Programmieren verwendet werden.
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. In der Schablonen-Sicht den Kontext markieren, in den die Schablone eingefügt werden soll.</li> <li>2. Auf den Kontext rechtsklicken und im Kontextmenü <b>Neu...</b> wählen. Oder: Auf das Symbol <b>Eine neue Schablone erstellen</b> klicken. Das Fenster <b>Neue Schablone</b> öffnet sich.</li> <li>3. Im Feld <b>Name</b> einen Namen für die Schablone eintragen.</li> <li>4. Im Feld <b>Beschreibung</b> eine Beschreibung eintragen (optional).</li> <li>5. Im Feld <b>Muster</b> den gewünschten Code eintragen.</li> <li>6. Schablonen-Eigenschaften mit <b>OK</b> bestätigen. Die Schablone wird erstellt und in der Schablonen-Sicht eingefügt.</li> </ol>

#### 15.1.3.5 Methoden extrahieren

<b>Beschreibung</b>	Teile des Programmcodes können in der Roboter-Applikation ausgelagert und als eigene Methode zur Verfügung gestellt werden. Dies ist insbesondere bei wiederkehrenden Aufgaben sinnvoll, da dadurch die Übersichtlichkeit innerhalb der Roboter-Applikation erhöht wird.
<b>Vorgehensweise</b>	<ol style="list-style-type: none"> <li>1. Gewünschten Programmcode markieren.</li> <li>2. In den Editoren-Bereich rechtsklicken.</li> <li>3. Im Kontextmenü <b>Refactoring &gt; Methode extrahieren...</b> wählen. Das Fenster <b>Methode extrahieren</b> öffnet sich.</li> <li>4. Unter <b>Methodenname</b> einen eindeutigen Methodennamen eingeben und den gewünschten <b>Änderungswert für Zugriff</b> auswählen. Mit <b>OK</b> bestätigen. Der markierte Programmcode wird entfernt und die neue Methode erzeugt. Die neue Methode wird an der Stelle aufgerufen, an der zuvor der Code entfernt wurde.</li> </ol>
<b>Änderungswert für Zugriff</b>	Diese Option legt fest, welche Klassen die ausgelagerte Methode aufrufen können.

Option	Beschreibung
<b>Privat (private)</b>	Methode kann nur von der zugehörigen Klasse selbst aufgerufen werden.
<b>Standard (default)</b>	Folgende Klassen können die Methode aufrufen: <ul style="list-style-type: none"> <li>■ Die zugehörige Klasse</li> <li>■ Die inneren Klassen der zugehörigen Klasse</li> <li>■ Alle Klassen des Pakets, in dem sich die zugehörige Klasse befindet</li> </ul>

Option	Beschreibung
<b>Geschützt (protected)</b>	Folgende Klassen können die Methode aufrufen: <ul style="list-style-type: none"> <li>■ Die zugehörige Klasse</li> <li>■ Die Unterklassen der zugehörigen Klasse (Vererbung)</li> <li>■ Die inneren Klassen der zugehörigen Klasse</li> <li>■ Alle Klassen des Pakets, in dem sich die zugehörige Klasse befindet</li> </ul>
<b>Öffentlich (public)</b>	Alle Klassen können die Methode aufrufen. Unabhängig von der Beziehung zur zugehörigen Klasse und unabhängig von der Paketzugehörigkeit.

#### 15.1.4 Javadoc-Informationen anzeigen

**Beschreibung** Javadoc ist eine aus speziellen Java-Kommentaren generierte Dokumentation. In Javadoc sind die Funktionalität und Verwendung von Klassen, Methoden und Bibliotheken beschrieben.

Die Javadoc-Informationen können bei der Programmierung angezeigt werden. Die Informationen sind nur in englischer Sprache verfügbar.

Die verschiedenen Anzeigemöglichkeiten sind am Beispiel der Klasse LBR beschrieben.

**Vorgehensweise Javadoc-Informationen beim Auto-Vervollständigen anzeigen:**

1. Beim Auto-Vervollständigen (STRG + Leertaste) einen Eintrag in der Vervollständigen-Liste markieren. Die zugehörigen Javadoc-Informationen werden in einem zusätzlichen Fenster im Editoren-Bereich angezeigt.

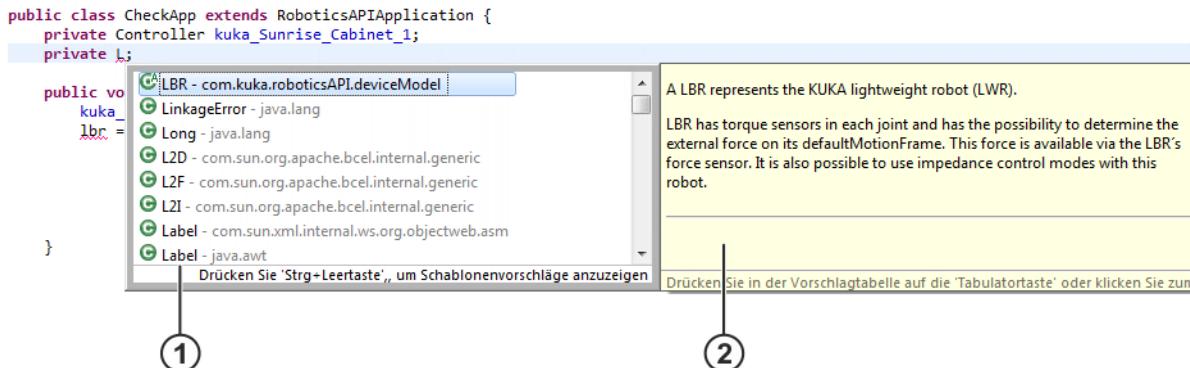


Abb. 15-2: Javadoc-Informationen beim Auto-Vervollständigen anzeigen

- 1 Vervollständigen-Liste
  - 2 Javadoc-Informationen
2. Um das Fenster im Editoren-Bereich zu fixieren, die Tabulatortaste drücken oder in das Fenster klicken.  
Das Fixieren des Fensters ermöglicht das Navigieren in der Javadoc-Beschreibung, z. B. durch Scrollen.

**Javadoc-Informationen mithilfe des Mauszeigers anzeigen:**

- Im Programmcode den Mauszeiger auf den gewünschten Element-Namen bewegen. Die zugehörigen Javadoc-Informationen werden automatisch in einem Fenster im Editoren-Bereich angezeigt.

Folgende Elemente reagieren auf den Mauszeiger:

- Methoden

- Klassen (Datentypen, nicht selbst definierte Datenfelder)
- Schnittstellen
- Enums

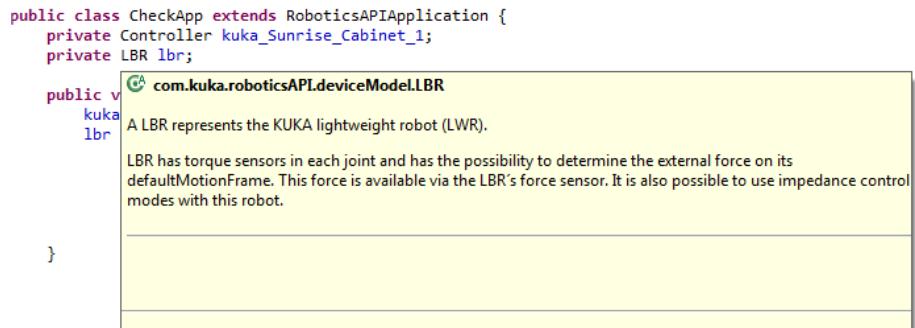


Abb. 15-3: Javadoc-Informationen über Mauszeiger anzeigen

- Von hier aus stehen weitere Anzeigeeoptionen zur Verfügung:
  - Um in der Javadoc-Beschreibung navigieren zu können, z. B. durch Scrollen, den Mauszeiger in das Fenster bewegen.  
Das Fenster ist nicht fixiert. Wird der Mauszeiger aus dem Fenster bewegt, schließt sich das Fenster.
  - Um das Fenster im Editoren-Bereich zu fixieren, die F2-Taste drücken oder in das Fenster klicken.  
In dem fixierten Fenster kann ebenfalls in der Javadoc-Beschreibung navigiert werden.
  - Um die Javadoc-Informationen zusätzlich in der Sicht **Javadoc** anzeigen, das ausgewählte Element mit einem Linksklick markieren.  
Wenn das Fenster nicht im Editoren-Bereich fixiert ist, wird es geschlossen.

## Navigieren

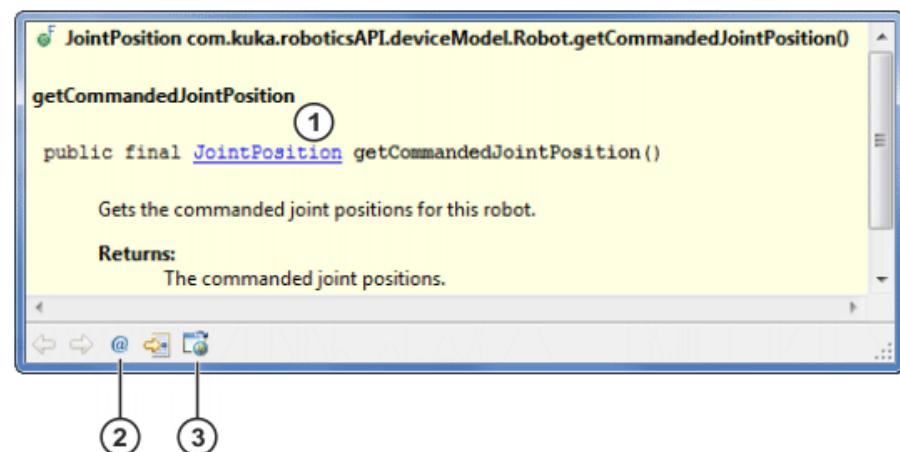


Abb. 15-4: Navigieren in Javadoc-Beschreibung

<b>Pos.</b>	<b>Beschreibung</b>
1	<p>Verlinkte Klasse</p> <p>Durch einen Linksklick auf die verlinkte Klasse, können die vollständigen Javadoc-Informationen dieser Klasse im Javadoc-Browser angezeigt werden.</p> <p><b>Hinweis:</b> Wird der entsprechende Link in der Sicht <b>Javadoc</b> gewählt, werden die vollständigen Javadoc-Informationen in der Sicht selbst angezeigt.</p>
2	<p>Button <b>In Javadoc-Ansicht anzeigen</b></p> <p>Das Fenster im Editoren-Bereich schließt sich und die Javadoc-Informationen werden in der Sicht <b>Javadoc</b> angezeigt.</p>
3	<p>Button <b>Open Attached Javadoc Browser</b></p> <p>Das Fenster im Editoren-Bereich schließt sich und die vollständigen Javadoc-Informationen der zugehörigen Klasse werden im Javadoc-Browser angezeigt.</p>



Es gibt eine weitere Möglichkeit sich die vollständigen Javadoc-Informationen zu einem bestimmten Element im Javadoc-Browser anzeigen zu lassen: Gewünschtes Element im Programmcode markieren und SHIFT + F2 drücken.

#### 15.1.4.1 Aufbau des Javadoc-Browsers

Der Aufbau des Javadoc-Browsers wird in verkürzter Form am Beispiel der Klasse LBR beschrieben.

**Overview Package Class Tree Deprecated Index Help**

**PREV CLASS NEXT CLASS**

SUMMARY: NESTED | FIELD | CONSTR | METHOD

com.kuka.roboticsAPI.deviceModel

**Class LBR**

java.lang.Object

- └ com.kuka.common.TaggableObject
- └ com.kuka.roboticsAPI.geometricModel.SceneGraphNode
- └ com.kuka.roboticsAPI.geometricModel.SpatialObject
- └ com.kuka.roboticsAPI.geometricModel.PhysicalObject
- └ com.kuka.roboticsAPI.deviceModel.Device
- └ com.kuka.roboticsAPI.deviceModel.Robot
- └ com.kuka.roboticsAPI.deviceModel.LBR

**All Implemented Interfaces:**

com.kuka.common.ITaggable, IOperationModeProvider

**Direct Known Subclasses:**

SunriseLBR

**public abstract class LBR**  
extends Robot

A LBR represents the KUKA lightweight robot (LWR).

LBR has torque sensors in each joint and has the possibility to determine the external force on its defaultMotionFrame. This force is available via the LBR's force sensor. It is also possible to use impedance control modes with this robot.

### Field Summary

protected	<code>gmsSensorLimits</code> double[] torque sensor limits.
-----------	--

**Fields inherited from class com.kuka.roboticsAPI.deviceModel.Device**

hardwareVersion

**Constructor Summary**

`LBR(Controller controller, java.lang.String name)`  
Creates a new LBR instance with the given controller.

**Method Summary**

boolean	<code>checkTorqueSensor(JointEnum joint)</code> Checks if the torque sensor of the given axis works properly.
---------	--

**Field Detail**

`_gmsSensorLimits`  
protected double[] \_gmsSensorLimits  
torque sensor limits.

**Constructor Detail**

**LBR**

`public LBR(Controller controller,  
              java.lang.String name)`

Creates a new LBR instance with the given controller.

**Parameters:**  
controller - The controller to which this device belongs.  
name - The name of the device.

**Method Detail**

**initializeJointCount**

`protected int initializeJointCount()`

Abb. 15-5: Aufbau des Javadoc-Browsers

<b>Pos.</b>	<b>Beschreibung</b>
1	Navigation
2	<p>Klassenhierarchie            (&gt;&gt;&gt; Abb. 15-6 )</p> <p>Hier werden die Vererbungsbeziehungen der Klasse dargestellt.</p>
3	<p>Beschreibung der Klasse</p> <p>Hier wird die Aufgabe der Klasse und ihre Funktionalität beschrieben. Es ist üblich, dass auf Besonderheiten bei der Verwendung der Klasse hingewiesen wird. Es können auch kurze Beispiele zur Verwendung der Klasse enthalten sein.</p> <p>Am Ende der Beschreibung wird üblicherweise angegeben, ab welcher Bibliotheks-Version die Klasse verfügbar ist. Zusätzlich kann eine Auflistung mit Verweisen zu weiteren Klassen oder Methoden enthalten sein, die ebenfalls von Interesse sind.</p>
4	<p>Übersichten</p> <ul style="list-style-type: none"> <li>■ <b>Field Summary</b>            Übersicht der Datenfelder, die zur Klasse gehören            Darunter sind die Datenfelder aufgelistet, die von einer Elternklasse geerbt wurden.</li> <li>■ <b>Constructor Summary</b>            Übersicht der Konstruktoren, die zur Klasse gehören</li> <li>■ <b>Method Summary</b>            Übersicht der Methoden, die zur Klasse gehören            Darunter sind die Methoden aufgelistet, die von einer Elternklasse geerbt wurden.</li> </ul> <p>Sofern bei der Erstellung von Javadoc angegeben, enthalten die Übersichten Kurzbeschreibungen zu den Datenfeldern, Konstruktoren und Methoden der Klasse. Datenfelder und Methoden, die vererbt wurden, sind lediglich aufgelistet.</p> <p>Ausführliche Beschreibungen zu den Datenfeldern, Konstruktoren und Methoden befinden sich im Bereich Details. Durch einen Klick auf den jeweiligen Namen, gelangt man direkt zur ausführlichen Beschreibung.</p>
5	<p>Details</p> <ul style="list-style-type: none"> <li>■ <b>Field Detail</b>            Ausführliche Beschreibung der Datenfelder, die zur Klasse gehören</li> <li>■ <b>Constructor Detail</b>            Ausführliche Beschreibung der Konstruktoren, die zur Klasse gehören</li> <li>■ <b>Method Detail</b>            Ausführliche Beschreibung der Methoden, die zur Klasse gehören</li> </ul> <p>Die ausführliche Beschreibung kann beispielsweise bei Methoden eine Auflistung und Beschreibung der übergebenen Parameter sowie des Rückgabewertes enthalten. Sofern vorhanden, werden hier auch die Ausnahmen (Exceptions) genannt, die beim Ausführen einer Methode oder eines Konstruktors auftreten können.</p>

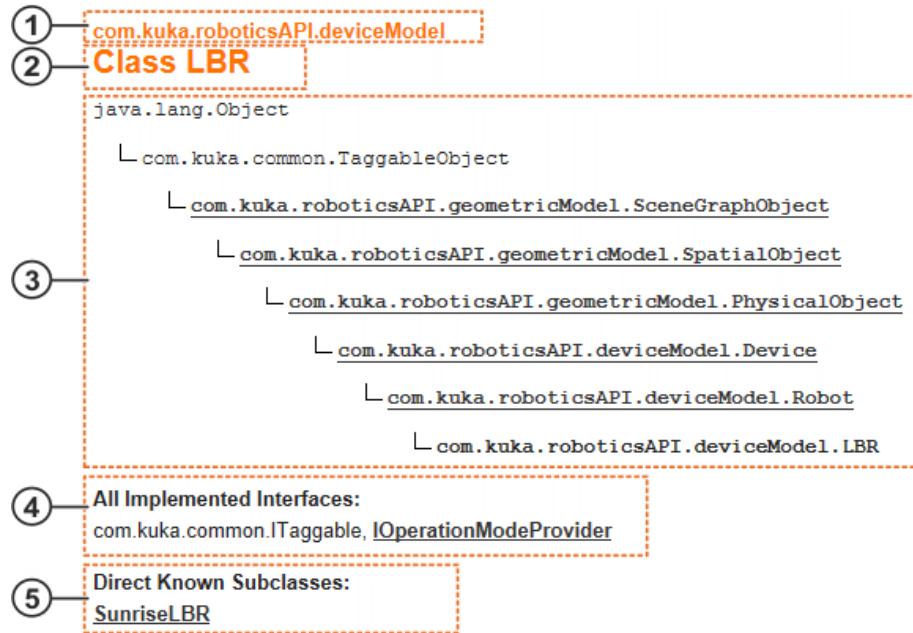


Abb. 15-6: Klassenhierarchie

Pos.	Beschreibung
1	Name des Pakets, zu dem die Klasse gehört
2	Name der Klasse
3	Klassenhierarchie (Abstammung der Klasse)
4	Aufzählung der Schnittstellen, die die Klasse implementiert
5	Aufzählung der Unterklassen, die von der Klasse abgeleitet sind

## 15.2 Zeichen und Schriftarten

In den Syntaxbeschreibungen werden folgende Zeichen und Schriftarten verwendet:

Syntax-Element	Darstellung
Java-Code	<ul style="list-style-type: none"> <li>■ Schriftart Courier</li> <li>■ Groß-/Kleinschreibung</li> </ul> Beispiele: <code>private; new; linRel; Tool</code>
Elemente, die durch programmspezifische Angaben ersetzt werden müssen	<ul style="list-style-type: none"> <li>■ Kursiv</li> <li>■ Groß-/Kleinschreibung</li> </ul> Beispiele: <i>Zielpunkt</i> ; <i>Name</i> ; <i>mode</i>
Optionale Elemente	<ul style="list-style-type: none"> <li>■ In spitzen Klammern</li> </ul> Beispiel: <code>&lt;.setVelocity (Wert) &gt;</code>
Elemente, die sich gegenseitig ausschließen	<ul style="list-style-type: none"> <li>■ Getrennt durch das Zeichen "   "</li> </ul> Beispiel: <code>++   --</code>

## 15.3 Datentypen

### Übersicht

In Java gibt es 2 Arten von Datentypen:

- Primitive Datentypen
- Komplexe Datentypen

Komplexe Datentypen werden in Java durch Klassen definiert.

### Übersicht wichtiger Datentypen:

Datentyp	Beschreibung
int	Ganzzahl ■ $-2^{31}-1 \dots +2^{31}-1$ Beispiele: -1; 32; 8000
double	Gleitkommazahl mit doppelter Genauigkeit ■ $-1.7E+308 \dots +1.7E+308$ Beispiele: 1.25; -98.76; 123.456
boolean	Logischer Zustand ■ true ■ false
char	Character (1 Zeichen) ■ ASCII-Zeichen Beispiele: 'A'; '1'; 'q'
String	Zeichenkette ■ ASCII-Zeichen Beispiele: "KUKA"; "tool"



Im Java-Editor werden die Namen der primitiven Datentypen in violetter Schriftfarbe angezeigt.

## 15.4 Variablen

**Beschreibung** Bevor eine Variable im Programm verwendet werden kann, muss sie deklariert werden, d. h. Datentyp und Bezeichner müssen festgelegt werden. Eine Variable kann z. B. in der run()-Methode einer Applikation deklariert werden.

**Syntax** *Datentyp Name;*

**Erläuterung der Syntax**

Element	Beschreibung
Datentyp	Datentyp der Variablen
Name	Name der Variablen

**Beispiele**

```
int counter;
double value;
boolean isObjectPlaced;
```

## 15.5 Netzwerk-Kommunikation über UDP und TCP/IP

Auf der Robotersteuerung sind bestimmte Ports zur Kommunikation mit externen Geräten über UDP oder TCP/IP freigegeben.

Folgende Port-Nummern (Client- oder Server-Socket) können in einer Roboter-Applikation verwendet werden:

- 30 000 bis 30 010

## 15.6 Versionsinformationen RoboticsAPI

Die RoboticsAPI ist die Programmierschnittstelle für alle roboterspezifischen Befehle. Sie besitzt eine 4-stellige Versionsnummer. Diese gibt bei einem Versions-Vergleich Auskunft über Änderungen der Schnittstelle.

### 15.6.1 RoboticsAPI-Version anzeigen

Die RoboticsAPI-Versionsnummer kann über die Datei StationSetup.cat (Stationskonfiguration) abgefragt werden.

<b>Vorgehensweise</b>	<ol style="list-style-type: none"><li>1. Projekt im <b>Paket-Explorer</b> öffnen.</li><li>2. Stationskonfiguration öffnen.</li><li>3. Registerkarte <b>Software</b> wählen.</li><li>4. Bei <b>Einzelne Bibliotheken anzeigen</b> das Häkchen setzen.</li></ol>
<b>Beschreibung</b>	<p>Die RoboticsAPI-Versionsnummer wird in der Zeile <b>BasicRoboticsJavaLib</b> angezeigt:</p> <ul style="list-style-type: none"><li>■ Spalte <b>Aktuell installierte Version</b>: Version, die zuletzt auf der Steuerung installiert wurde</li><li>■ Spalte <b>Versionsauswahl</b>: Version, die aktuell im Sunrise-Projekt verwendet wird</li></ul>

### 15.6.2 Aufbau RoboticsAPI-Versionsnummer

Die RoboticsAPI-Versionsnummer ist im Format XX.YY.ZZ.Build-Nummer aufgebaut.

- XX: Release-Version
- YY: Inkompatible Änderungen zwischen den Versionen  
In Applikationen, die unter Verwendung der älteren API-Version erstellt wurden, kann es bei Umstellung auf eine aktuellere Version zu Fehlern kommen.
- ZZ: Kompatible Änderungen zwischen den Versionen  
Applikationen, die unter Verwendung der älteren API-Version erstellt wurden, können auch mit der aktuellen API-Version genutzt werden.
- Build-Nummer: Wird automatisch vergeben.

## 15.7 Bewegungsprogrammierung: PTP, LIN, CIRC

### 15.7.1 Aufbau eines Bewegungsbefehls (move/moveAsync)

<b>Beschreibung</b>	<p>In Sunrise können Bewegungsbefehle auf alle bewegbaren Objekte einer Station angewendet werden. Ein bewegbares Objekt kann z. B. der Roboter sein, aber auch ein Werkzeug, das mit dem Roboterflansch verbunden ist oder ein Werkstück, das von einem Werkzeug (z. B. einem Greifer) gehalten wird.</p> <p>Bewegungsbefehle können synchron und asynchron ausgeführt werden. Dazu stehen die Methoden move(...) und moveAsync(...) zur Verfügung:</p>
	<ul style="list-style-type: none"><li>■ move(...) für die synchrone Ausführung Synchron bedeutet, dass die Bewegungsbefehle schrittweise an die Echtzeitsteuerung gesendet und abgearbeitet werden. Die weitere Programmausführung ist solange unterbrochen bis die Bewegung ausgeführt wurde. Erst dann wird der nächste Befehl gesendet.</li></ul>

- moveAsync(...) für die asynchrone Ausführung  
Asynchron bedeutet, dass direkt nach dem Senden des Bewegungsbefehls die nächste Programmzeile ausgeführt wird. Die asynchrone Ausführung von Bewegungen ist z. B. notwendig, um Bewegungen zu überschleifen.

Wie die verschiedenen Bewegungsarten programmiert werden, wird beispielhaft am Objekt Roboter gezeigt.

Die Bewegungsprogrammierung für Werkzeuge und Werkstücke ist hier beschrieben: (>>> 15.11.4 "Werkzeuge und Werkstücke bewegen" Seite 313)



Bei der Programmierung können Werte mit einer höheren Genauigkeit angegeben werden, als der Roboter erreichen kann. Beispielsweise sind Positionsangaben im Nanometer-Bereich möglich, aber in dieser Genauigkeit nicht erreichbar.

## Syntax

Bewegung synchron ausführen:

*Objekt.move (Bewegung) ;*

Bewegung asynchron ausführen:

*Objekt.moveAsync (Bewegung) ;*

## Erläuterung der Syntax

Element	Beschreibung
<i>Objekt</i>	Objekt der Station, das bewegt wird  Hier wird der in der Applikation deklarierte und initialisierte Variablenname des Objekts angegeben.
<i>Bewegung</i>	Bewegung, die ausgeführt wird  Die auszuführende Bewegung ist durch folgende Elemente definiert: <ul style="list-style-type: none"> <li>■ Bewegungsart oder -block: <b>ptp</b>, <b>lin</b>, <b>circ</b>, <b>spl</b> oder <b>spline</b>, <b>splineJP</b>, <b>batch</b></li> <li>■ Zielposition</li> <li>■ Weitere optionale Bewegungsparameter</li> </ul>

## 15.7.2 PTP

### Beschreibung

Führt eine Punkt-zu-Punkt-Bewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind absolut.

Um den Zielpunkt zu programmieren, gibt es folgende Möglichkeiten:

- Frame aus den Applikationsdaten in Bewegungsanweisung einfügen.
- Frame im Programm anlegen und in Bewegungsanweisung verwenden.



Die Redundanzinformationen zum Zielpunkt – Status, Turn und Redundanzwinkel – müssen korrekt angegeben werden. Der Zielpunkt kann sonst nicht korrekt angefahren werden.

- Achswinkel der Achsen A1 ... A7 angeben. Es müssen immer alle Achswerte angegeben werden.

## Syntax

PTP-Bewegung mit Frame-Angabe:

```
ptp(getApplicationContext().getFrame("Zielpunkt") <. Bewegungsparameter>)
```

PTP-Bewegung mit Angabe der Achswinkel:

`ptp(A1, A2, ... A7) <.Bewegungsparameter>)`

#### Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Pfad des Frames im Frame-Baum oder Variablenname des Frames (wenn im Programm angelegt)
A1 ... A7	Achswinkel der Achsen A1 ... A7 (Typ: double; Einheit: rad)
Bewegungs-parameter	Weitere Parameter der Bewegung, z. B. Geschwindigkeit oder Beschleunigung

#### Beispiele

PTP-Bewegung zum Frame "StartPos":

```
robot.move(ptp(getApplicationContext().getFrame("/StartPos")));
```

PTP-Bewegung in die Kerzenstellung:

```
robot.move(ptp(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
```

PTP-Bewegung zum Frame "StartPos" mit Angabe der Relativgeschwindigkeit:

```
robot.move(ptp(getApplicationContext().getFrame("/StartPos"))
.setJointVelocityRel(0.25));
```

### 15.7.3 LIN

#### Beschreibung

Führt eine Linearbewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind kartesisch absolut.

Um den Zielpunkt zu programmieren, gibt es folgende Möglichkeiten:

- Frame aus den Applikationsdaten in Bewegungsanweisung einfügen.
- Frame im Programm anlegen und in Bewegungsanweisung verwenden.

#### Syntax

```
lin(getApplicationContext().getFrame("Zielpunkt") <.Bewegungs-parameter>)
```

#### Erläuterung der Syntax

Element	Beschreibung
Zielpunkt	Pfad des Frames im Frame-Baum oder Variablenname des Frames (wenn im Programm angelegt)  Die Redundanzinformationen zum Zielpunkt – Status und Turn – werden bei LIN-Bewegungen (ebenso wie bei CIRC-Bewegungen) ignoriert. Nur der Redundanzwinkel wird berücksichtigt.
Bewegungs-parameter	Weitere Parameter der Bewegung, z. B. Geschwindigkeit oder Beschleunigung

#### Beispiele

LIN-Bewegung zum Frame "/Table/P1":

```
robot.move(lin(getApplicationContext().getFrame("/Table/P1")));
```

LIN-Bewegung mit Angabe der kartesischen Geschwindigkeit:

```
robot.move(lin(getApplicationContext().getFrame("/Table/P1"))
.setCartVelocity(150.0));
```

## 15.7.4 CIRC

### Beschreibung

Führt eine Kreisbewegung aus. Damit die Steuerung die Kreisbewegung berechnen kann, müssen ein Hilfspunkt und ein Zielpunkt angegeben werden. Die Koordinaten von Hilfs- und Zielpunkt sind kartesisch und absolut.

Um den Hilfs- und Zielpunkt zu programmieren, gibt es folgende Möglichkeiten:

- Frame aus den Applikationsdaten in Bewegungsanweisung einfügen.
- Frame im Programm anlegen und in Bewegungsanweisung verwenden.

### Syntax

```
circ(getApplicationData().getFrame("Hilfspunkt"),
getApplicationData().getFrame("Zielpunkt")
<.Bewegungsparameter>)
```

### Erläuterung der Syntax

Element	Beschreibung
Hilfspunkt	Pfad des Frames im Frame-Baum oder Variablenname des Frames (wenn im Programm angelegt)  Die Redundanzinformationen zum Hilfspunkt – Status, Turn und Redundanzwinkel – werden ignoriert.
Zielpunkt	Pfad des Frames im Frame-Baum oder Variablenname des Frames (wenn im Programm angelegt)  Die Redundanzinformationen zum Zielpunkt – Status und Turn – werden bei CIRC-Bewegungen (ebenso wie bei LIN-Bewegungen) ignoriert. Nur der Redundanzwinkel wird berücksichtigt.
Bewegungsparameter	Weitere Parameter der Bewegung, z. B. Geschwindigkeit oder Beschleunigung

### Beispiele

CIRC-Bewegung zum Ziel-Frame "/Table/P4" über den Hilfs-Frame "/Table/P3":

```
robot.move(circ(getApplicationData().getFrame("/Table/P3"),
getApplicationData().getFrame("/Table/P4"));
```

CIRC-Bewegung mit Angabe der absoluten Beschleunigung:

```
robot.move(circ(getApplicationData().getFrame("/Table/P3"),
getApplicationData().getFrame("/Table/P4")).setCartAcceleration(25));
```

## 15.7.5 LIN REL

### Beschreibung

Führt eine Linearbewegung zum Zielpunkt aus. Die Koordinaten des Zielpunkts sind relativ zur Zielposition der Vorgängerbewegung, es sei denn, diese Vorgängerbewegung wird durch eine Abbruchbedingung abgebrochen. Dann sind die Koordinaten des Zielpunkts relativ zur Abbruch-Position.

Bei einer Relativbewegung wird der Zielpunkt defaultmäßig im Koordinatensystem des bewegten Frames verschoben. Optional kann ein anderes Referenz-Koordinatensystem angegeben werden, in dem die Relativbewegung ausgeführt wird. Die Koordinaten des Zielpunkts beziehen sich dann auf dieses Referenz-Koordinatensystem. Dies kann beispielsweise ein in den Applikationsdaten angelegter Frame sein oder eine vermessene Basis.

Um den Zielpunkt zu programmieren, gibt es folgende Möglichkeiten:

- Die kartesischen Offset-Werte einzeln angeben.
- Eine Frame-Transformation vom Typ Transformation verwenden. Die Frame-Transformation hat den Vorteil, dass die Rotation auch in Grad angegeben werden kann.

**Syntax**

LinRel-Bewegung mit Offset-Werten:

```
linRel(x, y, z<, a, b, c>
<, Referenzsystem>)
```

LinRel-Bewegung mit Frame-Transformation:

```
linRel(Transformation.ofDeg|ofRad(x, y, z, a, b, c)
<, Referenzsystem>)
```

**Erläuterung der Syntax**

Element	Beschreibung
x, y, z	Verschiebung in X-, Y-, Z-Richtung (Typ: double; Einheit: mm)
a, b, c	Rotation um die Z-, Y-, X-Achse (Typ: double) Die Einheit ist abhängig von der verwendeten Methode: <ul style="list-style-type: none"> <li>■ Offset-Werte und Transformation.ofRad: Rad</li> <li>■ Transformation.ofDeg: Grad</li> </ul>
Referenzsystem	Typ: AbstractFrame Referenz-Koordinatensystem, in dem die Bewegung ausgeführt wird

**Beispiele**

Der bewegte Frame ist der TCP eines Greifers. Dieser TCP bewegt sich von der aktuellen Position aus um 100 mm in X-Richtung und um 200 mm in negative Z-Richtung im Werkzeug-Koordinatensystem. Die Orientierung des TCP ändert sich nicht.

```
gripper.getFrame("/TCP2").move(linRel(100, 0, -200));
```

Der Roboter bewegt sich von der aktuellen Position aus um je 10 mm im Koordinatensystem des Frames P1. Zusätzlich dreht sich der Roboter 30° um die Z- und Y-Achse des Koordinatensystems des Frames P1.

```
robot.move(linRel(Transformation.ofDeg(10, 10, 10, 30, 30, 0),
getApplicationData().getFrame("/P1")));
```

**15.7.6 MotionBatch****Beschreibung**

Mehrere Einzelbewegungen können in einem MotionBatch zusammengefasst und so zeitgleich an die Robotersteuerung übermittelt werden. Dadurch können Bewegungen innerhalb des MotionBatch überschliffen werden.

Die Bewegungsparameter, z. B. Geschwindigkeit, Beschleunigung, Orientierungsführung etc., können für den gesamten Batch oder bewegungsweise programmiert werden.



Für den gesamten Batch können nur achsspezifische Bewegungsparameter – setJoint...Rel(...) – angegeben werden. Kartesische Bewegungsparameter – setCart(......) – müssen im Einzelsatz angegeben werden.

Die beiden Formen können gemischt auftreten, z. B. um einer einzelnen Bewegung einen anderen Parameterwert zuzuweisen als dem Batch.



Der Einzelsatz-Parameter überschreibt den Batch-Parameter. Dies gilt auch dann, wenn für den Batch ein niedrigerer Parameterwert angegeben wird als für den Einzelsatz.

**Syntax**

Objekt.move (batch (

```

Bewegung,
Bewegung,
...
Bewegung,
Bewegung
) <. Bewegungsparameter> );

```

## Erläuterung der Syntax

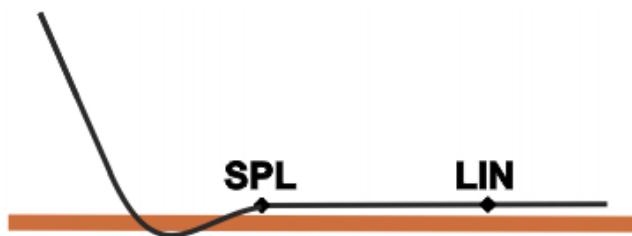
Element	Beschreibung
Objekt	Objekt der Station, das bewegt wird
Bewegung	Bewegung, mit oder ohne Bewegungsparameter <ul style="list-style-type: none"> <li>■ <b>ptp, lin, circ oder spline</b></li> </ul>
Bewegungsparameter	Bewegungsparameter, die am Ende des Batches programmiert werden, gelten für den gesamten Batch.  Nur achsspezifische Bewegungsparameter programmierbar!

## 15.8 Bewegungsprogrammierung: Spline

### 15.8.1 Programmiertipps für Spline-Bewegungen

- Die Anzahl der Spline-Segmente in einem Spline-Block ist nur durch den verfügbaren Speicher begrenzt.
- Die Planung einer Spline-Bewegung mit vielen kleinen Segmenten und kleinen Punktabständen kann sehr lange dauern. Um überlange Planungszeiten zu vermeiden:
  - Maximal 500 Segmente pro Spline-Block programmieren.
  - Möglichst Punktabstände > 5 mm programmieren.
- Spline-Bewegungen (mit vielen Segmenten) können über ein Array von Spline-Segmenten programmiert werden.
- Ein Spline-Block soll nur 1 Prozess umfassen (z. B. 1 Klebenahrt). Mehrere Prozesse in einem Spline-Block machen das Programm unübersichtlich und erschweren Änderungen.
- Wenn durch das Werkstück Geraden und Kreisabschnitte vorgegeben sind, LIN- und CIRC-Segmente verwenden. (Ausnahme: Für sehr kurze Geraden SPL-Segmente verwenden.) Ansonsten SPL-Segmente verwenden, besonders bei kurzen Punktabständen.
- Vorgehensweise bei der Festlegung der Bahn:
  - a. Zunächst wenige charakteristische Punkte teachen oder berechnen. Beispiel: Punkte, an denen die Krümmung umschlägt.
  - b. Die Bahn testen. An den Stellen, an denen die Genauigkeit noch nicht ausreicht, weitere SPL-Punkte einfügen.
- Aufeinanderfolgende LIN- und/oder CIRC-Segmente vermeiden, da die Geschwindigkeit hierdurch häufig auf 0 reduziert wird. Zur Vermeidung:
  - Zwischen LIN- und CIRC-Segmenten SPL-Segmente programmieren. Die Länge der SPL-Segmente muss mindestens > 0,5 mm sein. Abhängig vom konkreten Bahnverlauf können auch deutlich größere SPL-Segmente erforderlich sein.
  - Ein LIN-Segment durch mehrere SPL-Segmente ersetzen, die auf einer Geraden liegen. Damit wird auch die Bahn eine Gerade.
- Aufeinanderfolgende Punkte mit gleichen kartesischen Koordinaten vermeiden, da die Geschwindigkeit hierdurch auf 0 reduziert wird.

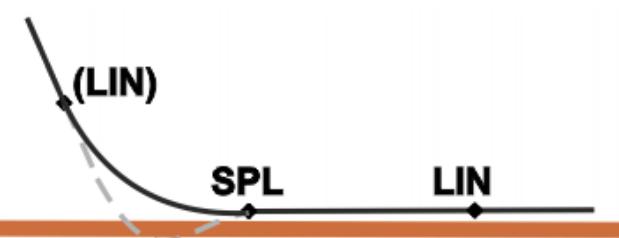
- Wenn der Roboter Punkte abfährt, die auf einer Arbeitsfläche liegen, kann es beim Anfahren des ersten Punkts zu einer Kollision mit der Arbeitsfläche kommen.



**Abb. 15-7: Kollision mit Arbeitsfläche**

Eine Kollision kann durch Einfügen eines LIN-Segments vor der Arbeitsfläche vermieden werden. Die Empfehlungen für den LIN-SPL-LIN-Übergang beachten.

(>>> 14.6.3 "LIN-SPL-LIN-Übergang" Seite 268)



**Abb. 15-8: Kollision mit Arbeitsfläche vermeiden**

- SPL-Segmente möglichst nicht verwenden, wenn nah an der Arbeitsraumgrenze verfahren wird. Es kann vorkommen, dass mit SPL die Arbeitsraumgrenze überschritten wird, obwohl der Roboter den Ziel-Frame in einer anderen Bewegungsart oder beim Handverfahren erreichen kann.

### 15.8.2 CP-Spline-Block anlegen

#### Beschreibung

Mit einem CP-Spline-Block kann man mehrere SPL-, LIN- und/oder CIRC-Segmente zu einer Gesamtbewegung zusammenfassen.

Ein Spline-Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.

Die Bewegungsparameter, z. B. Geschwindigkeit, Beschleunigung, Orientierungsführung etc., können für den gesamten Spline-Block oder segmentweise programmiert werden. Die beiden Formen können gemischt auftreten, z. B. um einem einzelnen Segment einen anderen Parameterwert zuzuweisen als dem Block.



Der Einzelsatz-Parameter überschreibt den Block-Parameter. Dies gilt auch dann, wenn für den Block ein niedrigerer Parameterwert angegeben wird als für den Einzelsatz.

#### Syntax

```
Spline Name = new Spline (
    Segment,
    Segment,
    ...
    Segment,
    Segment
)
```

) < .Bewegungsparameter> ;

### Erläuterung der Syntax

Element	Beschreibung
<i>Name</i>	Name des Spline-Blocks
<i>Segment</i>	Bewegung, mit oder ohne Bewegungsparameter ■ <b>spl, lin oder circ</b>
<i>Bewegungs-parameter</i>	Bewegungsparameter, die am Ende des Spline-Blocks programmiert werden, gelten für den gesamten Spline-Block.

### Beispiel

```
Spline mySpline = new Spline(
    spl(getApplicationContext().getFrame("/P1")),
    circ(getApplicationContext().getFrame("/P2"),
        getApplicationContext().getFrame("/P3")),
    spl(getApplicationContext().getFrame("/P4")).setCartVelocity(150),
    lin(getApplicationContext().getFrame("/P5"))
).setCartVelocity(250);
```

### 15.8.3 JP-Spline-Block anlegen

#### Beschreibung

Mit einem JP-Spline-Block kann man mehrere PTP-Segmente zu einer Gesamtbewegung zusammenfassen.

Ein Spline-Block darf keine sonstigen Anweisungen, z. B. Variablenzuweisungen oder Logikanweisungen, enthalten.

Die Bewegungsparameter, z. B. Geschwindigkeit, Beschleunigung etc., können für den gesamten Spline-Block oder segmentweise programmiert werden. Die beiden Formen können gemischt auftreten, z. B. um einem einzelnen Segment einen anderen Parameterwert zuzuweisen als dem Block.



Der Einzelsatz-Parameter überschreibt den Block-Parameter. Dies gilt auch dann, wenn für den Block ein niedrigerer Parameterwert angegeben wird als für den Einzelsatz.

#### Syntax

```
SplineJP Name = new SplineJP(
    Segment,
    Segment,
    ...
    Segment,
    Segment
) < .Bewegungsparameter> ;
```

### Erläuterung der Syntax

Element	Beschreibung
<i>Name</i>	Name des Spline-Blocks
<i>Segment</i>	PTP-Bewegung, mit oder ohne Bewegungsparameter
<i>Bewegungs-parameter</i>	Bewegungsparameter, die am Ende des Spline-Blocks programmiert werden, gelten für den gesamten Spline-Block.

### Beispiel

```
SplineJP mySpline = new SplineJP(
    ptp(getApplicationContext().getFrame("/P1")),
    ptp(getApplicationContext().getFrame("/P2"))
).setJointVelocityRel(0.75);
```

#### 15.8.4 Spline in Bewegungsanweisung verwenden

**Beschreibung** Die in einem Spline-Block programmierte Spline-Bewegung wird als Bewegungsart in der Bewegungsanweisung verwendet.

**Syntax** `Objekt.move (Spline-Name) ;`

**Erläuterung der Syntax**

Element	Beschreibung
<i>Objekt</i>	Objekt der Station, das bewegt wird
<i>Spline-Name</i>	Name des Spline-Blocks

**Beispiel**

```
robot.move (mySpline) ;
```

### 15.9 Bewegungsparameter

Die benötigten Bewegungsparameter können in beliebiger Reihenfolge an die Bewegungsanweisung angefügt werden. Hierfür werden der Punktoperator und set-Methoden verwendet.

#### Übersicht

Methode	Beschreibung
<code>setCartVelocity(...)</code>	<p>Absolute kartesische Geschwindigkeit (Typ: double; Einheit: mm/s)</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Der Wert gibt die maximale kartesische Geschwindigkeit an, die der Roboter bei der Bewegung fahren darf. Aufgrund anderer Bahnplanungsbegrenzungen kann es sein, dass diese maximale Geschwindigkeit nicht erreicht wird und die tatsächliche Geschwindigkeit niedriger ist.</p> <p>Wenn keine Geschwindigkeit angegeben wird, wird die Bewegung mit der schnellstmöglichen Geschwindigkeit ausgeführt.</p> <p><b>Hinweis:</b> Dieser Parameter kann bei PTP-Bewegungen nicht gesetzt werden.</p>
<code>setJointVelocity-Rel(...)</code>	<p>Achsspezifische Relativgeschwindigkeit (Typ: double; Einheit: %)</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul> <p>Bezieht sich auf den Maximalwert der Achsgeschwindigkeit in den Maschinendaten.</p> <p>(&gt;&gt;&gt; 15.9.1 "Achsspezifische Bewegungsparameter programmieren" Seite 304)</p>
<code>setCartAcceleration(...)</code>	<p>Absolute kartesische Beschleunigung (Typ: double; Einheit: mm/s<sup>2</sup>)</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Wenn keine Beschleunigung angegeben wird, wird die Bewegung mit der schnellstmöglichen Beschleunigung ausgeführt.</p> <p><b>Hinweis:</b> Dieser Parameter kann bei PTP-Bewegungen nicht gesetzt werden.</p>
<code>setJointAcceleration-Rel(...)</code>	<p>Achsspezifische relative Beschleunigung (Typ: double; Einheit: %)</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul> <p>Bezieht sich auf den Maximalwert der Achsbeschleunigung in den Maschinendaten.</p> <p>(&gt;&gt;&gt; 15.9.1 "Achsspezifische Bewegungsparameter programmieren" Seite 304)</p>

<b>Methode</b>	<b>Beschreibung</b>
setCartJerk(...)	<p>Absoluter kartesischer Ruck (Typ: double; Einheit: mm/s<sup>3</sup>)</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Wenn kein Ruck angegeben wird, wird die Bewegung mit der schnellstmöglichen Beschleunigungsänderung ausgeführt.</p> <p><b>Hinweis:</b> Dieser Parameter kann bei PTP-Bewegungen nicht gesetzt werden.</p>
setJointJerkRel(...)	<p>Achsspezifischer relativer Ruck (Typ: double; Einheit: %)</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul> <p>Bezieht sich auf den Maximalwert der achsspezifischen Beschleunigungsänderung in den Maschinendaten.</p> <p>(&gt;&gt;&gt; 15.9.1 "Achsspezifische Bewegungsparameter programmieren" Seite 304)</p>
setBlendingRel(...)	<p>Relative Überschleifdistanz (Typ: double)</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul> <p>Die relative Überschleifdistanz ist die Distanz vor dem Zielpunkt der Bewegung, bei der das Überschleifen frühestens beginnt. Wird "0.0" eingestellt, ist der Überschleifparameter ohne Auswirkung.</p> <p>Die Maximaldistanz (= 1.0) entspricht immer der Länge der Einzelbewegung oder bei Splines der Länge des letzten Segments. Für Bewegungen, die nicht innerhalb eines Splines kommandiert werden, steht nur der Bereich zwischen 0 % und 50 % für das Überschleifen zur Verfügung. Wird in diesem Fall ein Wert größer 50 % parametriert, beginnt das Überschleifen dennoch erst bei 50 % der Satzlänge.</p>
setBlendingCart(...)	<p>Absolute Überschleifdistanz (Typ: double; Einheit: mm)</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Die absolute Überschleifdistanz ist die Entfernung zum Zielpunkt der Bewegung, bei der das Überschleifen frühestens beginnt. Wird "0.0" eingestellt, ist der Überschleifparameter ohne Auswirkung.</p>
setBlendingOri(...)	<p>Orientierungsparameter für das Überschleifen (Typ: double; Einheit: rad)</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Das Überschleifen beginnt frühestens, wenn die absolute Differenz des dominierenden Orientierungswinkels zur Zielorientierung den hier eingestellten Wert unterschreitet. Wird "0.0" eingestellt, ist der Überschleifparameter ohne Auswirkung.</p>
setOrientationType(...)	<p>Orientierungsführung (Typ: Enum)</p> <ul style="list-style-type: none"> <li>■ <b>Constant</b></li> <li>■ <b>Ignore</b></li> <li>■ <b>OriJoint</b></li> <li>■ <b>VariableOrientation</b> (Default)</li> </ul> <p>(&gt;&gt;&gt; 14.9 "Orientierungsführung LIN, CIRC, SPL" Seite 272)</p>
setOrientationReferenceSystem(...)	<p>Nur relevant für CIRC-Bewegungen: Bezugssystem der Orientierungsführung (Typ: Enum)</p> <ul style="list-style-type: none"> <li>■ <b>Base</b></li> <li>■ <b>Path</b></li> </ul> <p>(&gt;&gt;&gt; 14.9.1 "CIRC – Bezugssystem der Orientierungsführung" Seite 274)</p>

### 15.9.1 Achsspezifische Bewegungsparameter programmieren

#### Beschreibung

Folgende achsspezifischen Bewegungsparameter können programmiert werden:

- Relativgeschwindigkeit setJointVelocityRel(...)
- Relative Beschleunigung setJointAccelerationRel(...)
- Relativer Ruck setJointJerkRel(...)

Es gibt verschiedene Möglichkeiten diese achsspezifischen Relativwerte anzugeben. Einen gültigen Wert für alle Achsen, verschiedene Werte für jede einzelne Achse oder einen Wert für eine einzelne Achse.

Diese Möglichkeiten werden am Beispiel der Relativgeschwindigkeit beschrieben:

- `setJointVelocityRel (Wert)`

Wenn ein Wert vom Typ double übergeben wird, gilt die Relativgeschwindigkeit für alle Achsen.

- `setJointVelocityRel (Array-Variable)`

Um jeder Achse eine eigene Relativgeschwindigkeit zuzuweisen, wird ein double-Array mit den jeweiligen Achswerten übergeben. In einem Array können beginnend mit Achse A1 die Achswerte von bis zu 12 Achsen definiert werden.

- `setJointVelocityRel (Achse, Wert)`

Um die Relativgeschwindigkeit einer einzelnen Achse anzugeben, wird diese Achse als Enum vom Typ JointEnum übergeben. Dieser Enum enthält 12 Achsen (JointEnum.J1 ... JointEnum.J12).

#### Beispiele

Alle Achsen bewegen sich mit 50 % der Maximalgeschwindigkeit:

```
robot.move(ptpgetApplicationData().getFrame("/P1"))
.setJointVelocityRel(0.5);
```

Die Achse A5 bewegt sich mit 50 %, alle anderen Achsen bewegen sich mit 20 % der Maximalgeschwindigkeit:

```
double[] velRelJoints = {0.2, 0.2, 0.2, 0.2, 0.5, 0.2, 0.2};
robot.move(ptpgetApplicationData().getFrame("/P1"))
.setJointVelocityRel(velRelJoints);
```

Die Achse A4 bewegt sich mit 50 % der Maximalgeschwindigkeit, alle anderen Achsen bewegen sich mit Maximalgeschwindigkeit:

```
robot.move(ptpgetApplicationData().getFrame("/P1"))
.setJointVelocityRel(JointEnum.J4, 0.5);
```

### 15.10 Handführen programmieren

#### Beschreibung

Der Roboter lässt sich mithilfe eines Handführgerätes von Hand führen. Der Handführmodus kann mit dem Bewegungsbefehl handGuiding() in der Applikation eingeschaltet werden. Das Handführen beginnt an der Istposition, die vor dem Einschalten zuletzt erreicht wurde.

Wenn der Handführmodus in der Applikation verwendet wird, müssen mindestens 2 ESM-Zustände konfiguriert sein:

- ESM-Zustand für Handführ-Bewegung

Der ESM-Zustand enthält die AMF *Zustimmung Handführgerät inaktiv*, die prüft ob die Zustimmung am Handführgerät nicht erteilt ist.

(>>> 13.8.5 "Handführen mit Zustimmmeinrichtung und Geschwindigkeitsüberwachung" Seite 209)

Es wird empfohlen, für die AMF *Zustimmung Handführgerät inaktiv* einen Sicherheitshalt 1 (bahntreu) als Stopp-Reaktion zu konfigurieren. Nach einem bahntreuen Stopp kann die Applikation direkt durch Drücken der Start-Taste fortgesetzt werden.

Wird für die AMF *Zustimmung Handführgerät inaktiv* eine nicht bahntreue Stopp-Reaktion konfiguriert, muss der Roboter nach dem Handführen erst rückpositioniert werden, bevor die Applikation fortgesetzt werden kann.



In einer Risikobetrachtung muss beurteilt werden, ob es zulässig ist, für den ESM-Zustand, der den Zustimmungsschalter am Handführgerät überwacht, eine bahntreue Stopp-Reaktion zu konfigurieren.

- ESM-Zustand für alle Bewegungen außer Handführ-Bewegung

Der ESM-Zustand enthält keine AMF *Zustimmung Handführgerät inaktiv*. Die Zustimmung am Handführgerät wird nicht ausgewertet.

In der Applikation sind in der Regel Bewegungen vor und nach dem Handführen notwendig. Es wird empfohlen, diese Bewegungen jeweils mithilfe eines ESM-Zustands zu überwachen, der die Zustimmung am Handführgerät nicht auswertet, und erst direkt vor dem Schalten in den Handführmodus in den ESM-Zustand für die Handführ-Bewegung zu wechseln. Wird dies in der Applikation so umgesetzt, ist das Verhalten wie folgt:

- Wird die Zustimmung zum Handführen erteilt, bevor der Handführmodus in der Applikation eingeschaltet wird, ist der Handführmodus sofort nach dem Einschalten aktiv. D. h. die Applikation wird beim Einschalten nicht pausiert, so dass der Übergang zwischen Applikations- und Handführmodus fließend ist.

Voraussetzung für dieses Verhalten: Die Applikationsgeschwindigkeit liegt unter der maximal zulässigen Geschwindigkeit, die für das Handführen konfiguriert ist.

(>>> 13.8.5.3 "Geschwindigkeitsüberwachung während des Handführens" Seite 211)

Wird die Applikation schneller abgefahren, wird die Applikation vor dem Schalten in den Handführmodus pausiert. (Dann Zustimmungsschalter loslassen, Start-Taste drücken und warten bis die Applikation erneut pausiert wird.)

- Wird die Zustimmung zum Handführen erst erteilt, wenn der Handführmodus in der Applikation bereits eingeschaltet ist, muss die Start-Taste gedrückt werden, um den Roboter von Hand führen zu können. Die Applikationspause ermöglicht es dem Bediener, die Hand zum Handführgerät zu bewegen.
- Der Handführmodus ist beendet, wenn die Zustimmung zum Handführen entzogen wird, z. B. durch Loslassen des Zustimmungsschalters. Die Applikation wird pausiert und kann nur durch Drücken der Start-Taste fortgesetzt werden. Die Applikationspause ermöglicht es dem Bediener, die Hand vom Handführgerät wegzubewegen.

**VORSICHT**

Wenn sich die Applikation beim Schalten in den Handführmodus in einem ESM-Zustand befindet, der keine AMF Zustimmung Handführgerät inaktiv enthält, kann der Roboter in einem Fall trotzdem von Hand geführt werden: Der Zustimmungsschalter am Handführgerät wird gedrückt und es ist eine AMF Zustimmung Handführgerät inaktiv in einem beliebigen anderen ESM-Zustand oder der PSM-Tabelle konfiguriert.

Diese Konstellation ist unbedingt zu vermeiden, da in so einem Fall die Applikation nach Beenden des Handführens nicht pausiert wird, wenn der Zustimmungsschalter am Handführgerät losgelassen wird. Stattdessen wird die Applikation ohne weitere Bedienhandlung fortgesetzt. Schließen an das Handführen weitere Bewegungen an, werden diese direkt ausgeführt, so dass der Bediener die Hand noch am Handführgerät hat und sich im Bewegungsbereich des Roboters befindet.



Das Umschalten zwischen ESM-Zuständen erfolgt durch nicht sicherheitsgerichtete Signale. Bei der Festlegung eines ESM-Zustands ist deswegen zu beachten, dass dieser unabhängig von Ort und Zeitpunkt seiner Aktivierung immer ein ausreichendes Maß an Sicherheit gewährleistet. (>>> 13.2 "Sicherheitskonzept" Seite 184)

**Vorbereitung**

Der Bewegungsbefehl handGuiding() gehört zur Klasse MMCMotions. Die Klasse muss manuell in den Import-Bereich der Roboter-Applikation eingefügt werden. Folgende Zeile ist zu programmieren:

```
import static com.kuka.roboticsAPI.motionModel.MMCMotions.*;
```

**Syntax**

```
Objekt.move(handGuiding());
```

**Erläuterung der Syntax**

Element	Beschreibung
Objekt	Objekt der Station, das bewegt wird

**Beispiel**

```
1 robot.setESMState("1");
2 robot.move(ptpgetApplicationData().getFrame("/P1"));
3 robot.setESMState("2");
4 robot.move(handGuiding());
5 robot.setESMState("1");
6 robot.move(ptpgetApplicationData().getFrame("/P2"));
```

Zeile	Beschreibung
1	ESM-Zustand 1 wird für den Roboter aktiviert. ESM-Zustand 1 überwacht in diesem Beispiel den Bedienerschutz.
2	Frame "/P1" wird mit einer PTP-Bewegung angefahren.
3	ESM-Zustand 2 wird für den Roboter aktiviert. ESM-Zustand 2 überwacht den Zustimmungsschalter am Handführgerät. Wenn noch keine Zustimmung über den Schalter erteilt ist, wird die konfigurierte Stopp-Reaktion ausgelöst und die Applikation pausiert.

Zeile	Beschreibung
4	<p>Der Handführmodus wird eingeschaltet.</p> <p>Sobald der Zustimmungsschalter am Handführgerät gedrückt und in Mittelstellung gehalten wird, kann der Roboter von Hand geführt werden.</p> <p>Wenn die Zustimmung zum Handführen entzogen wird, z. B. durch Loslassen des Zustimmungsschalters, ist der Handführmodus beendet. Die für ESM-Zustand 2 konfigurierte Stopp-Reaktion wird ausgelöst und die Bewegungsausführung pausiert.</p>
5	<p>ESM-Zustand 1 wird für den Roboter aktiviert. ESM-Zustand 1 überwacht in diesem Beispiel den Bedienerschutz.</p> <p>Die Bewegungsausführung ist immer noch pausiert. Um die Applikation fortzusetzen, muss die Start-Taste gedrückt werden.</p>
6	Frame "/P2" wird mit einer PTP-Bewegung angefahren.

### 15.10.1 Achsspezifische Grenzen für das Handführen

**Beschreibung** Der Achsbereich jeder Achse ist durch Software-Endschalter begrenzt. Für das Handführen können in der Roboter-Applikation zusätzlich benutzerspezifische Achsgrenzen festgelegt werden.

Durch Festlegen einer unteren und oberen Achsgrenze erhält man einen erlaubten Achsbereich, in dem freies Handführen möglich ist, sowie 2 nicht erlaubte Achsbereiche, die zwischen unterer und oberer Achsgrenze und dem jeweiligen Software-Endschalter liegen.

Wird eine benutzerdefinierte Achsgrenze beim Handführen erreicht, wird ein virtuelles Feder-Dämpfer-System aufgespannt. Dieses setzt jeder weiteren Bewegung in Richtung Endschalter einen Widerstand entgegen, der umso größer wird, je mehr sich eine Achse dem Endschalter nähert.

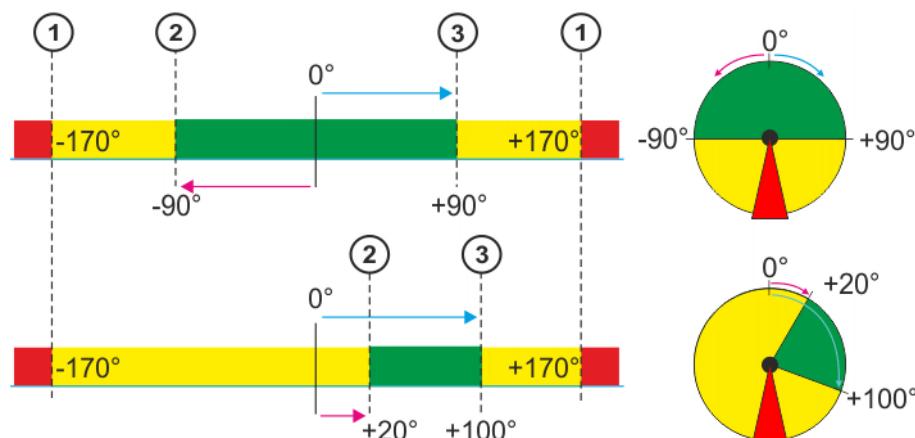


Abb. 15-9: Achsspezifische Grenzen für das Handführen (Beispiele)

- 1 Position Software-Endschalter
- 2 Untere Grenze des Achsbereichs
- 3 Obere Grenze des Achsbereichs

Die Grenzen für das Handführen müssen für jede Achse einzeln angegeben und aktiviert werden.

Defaultmäßig gilt folgendes Verhalten:

- Bei Erreichen einer Achsgrenze wirken alle an der Bewegung beteiligten Achsen einer weiteren Bewegung in Richtung Endschalter entgegen.  
Es kann definiert werden, dass nur die Achse, die die Grenze erreicht hat, einer weiteren Bewegung in Richtung Endschalter entgegenwirkt.
- Wenn zu Beginn des Handführrens bereits eine Achsgrenze überschritten ist, muss die betroffene Achse manuell aus dem nicht erlaubten Bereich herausbewegt werden.  
Es kann definiert werden, dass sich die Achse automatisch aus dem nicht erlaubten Bereich herausbewegen soll.

## Übersicht

Die benötigten Bewegungsparameter können in beliebiger Reihenfolge an den Bewegungsbefehl handGuiding() angefügt werden. Hierfür werden der Punktoperator und set-Methoden verwendet.

Methoden	Beschreibung
setAxisLimitsEnabled(...)	<p>Aktivieren der benutzerspezifischen Achsgrenzen für das Handführen (Typ: boolean[])</p> <ul style="list-style-type: none"> <li>■ <b>true</b>: Achsgrenze aktiv</li> <li>■ <b>false</b>: Achsgrenze nicht aktiv</li> </ul> <p><b>Hinweis:</b> Diese Methode bezieht sich auf die Grenzen, die der Benutzer mit den Methoden setAxisLimitsMax(...) und setAxisLimitsMin(...) setzen kann. Die äußersten Achsgrenzen des Roboters (Software-Endschalter) werden immer überwacht.</p>
setAxisLimitsMax(...)	Obere Achsgrenzen (Typ: double[]; Einheit: rad)
setAxisLimitsMin(...)	Untere Achsgrenzen (Typ: double[]; Einheit: rad)
	<b>Hinweis:</b> Die untere Achsgrenze muss immer kleiner sein als die zugehörige obere Achsgrenze.
setAxisLimitViolationFreezesAll(...)	<p>Verhalten bei Erreichen einer Achsgrenze (Typ: boolean)</p> <ul style="list-style-type: none"> <li>■ <b>true</b>: Bei Erreichen einer Achsgrenze wirken alle an der Bewegung beteiligten Achsen einer weiteren Bewegung in Richtung Endschalter entgegen.</li> <li>■ <b>false</b>: Bei Erreichen einer Achsgrenze wirkt nur die betroffene Achse einer weiteren Bewegung in Richtung Endschalter entgegen.</li> </ul> <p>Default: <b>true</b></p> <p>Wird dieser Wert nicht gesetzt, gilt automatisch der Default-Wert.</p>
setPermanentPullOnViolationAtStart(...)	<p>Verhalten, wenn zu Beginn des Handführrens bereits eine Achsgrenze überschritten ist (Typ: boolean)</p> <ul style="list-style-type: none"> <li>■ <b>true</b>: Wenn die Zustimmung zum Handführen erteilt wird, bewegt sich die Achse automatisch aus dem nicht erlaubten Bereich heraus. Bei Erreichen des erlaubten Bereichs wird die Bewegung automatisch gestoppt.</li> <li>■ <b>false</b>: Wenn die Zustimmung zum Handführen erteilt wird, bewegt sich die Achse nicht. Sie muss manuell aus dem nicht erlaubten Bereich herausbewegt werden.</li> </ul> <p>Default: <b>false</b></p> <p>Wird dieser Wert nicht gesetzt, gilt automatisch der Default-Wert.</p>

## Beispiel

```
private LBR robot;
private HandGuidingMotion motion;
...
motion = handGuiding()
```

```

.setAxisLimitsMax(+1.407, +0.872, +0.087, -0.785, +0.087,
+1.571, +0.087)
.setAxisLimitsMin(-1.407, +0.175, -0.087, -1.571, -0.087,
-1.571, -0.087)
.setAxisLimitsEnabled(false, true, false, true, false,
true, false)
.setAxisLimitViolationFreezesAll(true)
.setPermanentPullOnViolationAtStart(true);

robot.move(motion);

```

## 15.11 Werkzeuge und Werkstücke im Programm verwenden

Eine Applikation stellt ein programmiertes Abbild einer realen Station dar und muss damit auch alle bewegbaren Objekte und feststehenden geometrischen Objekte der Station enthalten. Bewegbare Objekte einer Station sind z. B. Roboter, Werkzeuge und Werkstücke. Feststehende Objekte sind z. B. Ablagetische oder Förderbänder.

Robotersteuerung und Roboter werden beim Erstellen der Applikation automatisch deklariert und initialisiert. Werkzeuge und Werkstücke, die in der Applikation verwendet werden, müssen vom Benutzer deklariert und initialisiert werden.

Werkzeuge und Werkstücke mit Lastdaten und geometrischen Daten werden in der Sicht **Objektvorlagen** angelegt und verwaltet.

(>>> 9.3 "Objektverwaltung" Seite 142)

### Datentypen

Die Datentypen für die Objekte einer Station sind in der RoboticsAPI vordefiniert:

Datentyp	Objekt
Controller	Robotersteuerung
LBR	Leichtbauroboter
Tool	Werkzeug
Workpiece	Werkstück
GeometricObject	Feststehende geometrische Objekte

### 15.11.1 Werkzeuge und Werkstücke deklarieren

#### Syntax

private *Datentyp* *Objekt-Name*;

#### Erläuterung der Syntax

Element	Beschreibung
private	Das Schlüsselwort kennzeichnet lokal gültige Variablen. Lokal gültig bedeutet, dass das Datenfeld nur von der zugehörigen Klasse verwendet werden kann.
<i>Datentyp</i>	Objekttyp
<i>Objekt-Name</i>	Name der Objekt-Variablen

#### Beispiel

In der Applikation werden 2 Werkzeuge (Greifer, Handführspitze) und 1 Werkstück (Stift) verwendet.

```

private Tool gripper;
private Tool guidingTool;
private Workpiece pen;

```

### 15.11.2 Werkzeuge und Werkstücke initialisieren

<b>Beschreibung</b>	Hier wird beschrieben, wie Werkzeuge und Werkstücke initialisiert werden, die in den Objekt-Vorlagen des zugehörigen Projekts angelegt wurden.						
<b>Syntax</b>	<i>Objekt-Name</i> = getApplicationData().createFromTemplate("Objekt-Vorlage");						
<b>Erläuterung der Syntax</b>	<table border="1"> <thead> <tr> <th>Element</th><th>Beschreibung</th></tr> </thead> <tbody> <tr> <td><i>Objekt-Name</i></td><td>Name der Objekt-Variablen</td></tr> <tr> <td><i>Objekt-Vorlage</i></td><td>Name der Objekt-Vorlage, wie in der Sicht <b>Objektvorlagen</b> angegeben</td></tr> </tbody> </table>	Element	Beschreibung	<i>Objekt-Name</i>	Name der Objekt-Variablen	<i>Objekt-Vorlage</i>	Name der Objekt-Vorlage, wie in der Sicht <b>Objektvorlagen</b> angegeben
Element	Beschreibung						
<i>Objekt-Name</i>	Name der Objekt-Variablen						
<i>Objekt-Vorlage</i>	Name der Objekt-Vorlage, wie in der Sicht <b>Objektvorlagen</b> angegeben						

**Beispiel** Folgende Werkzeuge und Werkstücke wurden in den Objekt-Vorlagen angelegt:

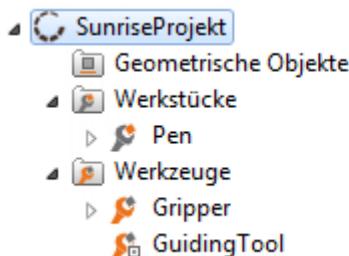


Abb. 15-10: Objekt-Vorlagen

```
private Tool gripper;
private Tool guidingTool;
private Workpiece pen;

public void initialize() {
    ...
    gripper = getApplicationData().createFromTemplate("Gripper");
    guidingTool = getApplicationData().createFromTemplate("GuidingTool");
    pen = getApplicationData().createFromTemplate("Pen");
    ...
}
```

### 15.11.3 Werkzeuge und Werkstücke mit Roboter verbinden

Damit Werkzeuge und Werkstücke als bewegbare Objekte in Bewegungsanweisungen verwendet werden können, müssen sie in der Applikation über die Methode attachTo(...) mit dem Roboter verbunden werden.

- Werkzeuge sind direkt oder indirekt mit dem Roboterflansch verbunden.
- Werkstücke sind indirekt über ein Werkzeug oder andere Werkstücke mit dem Roboter verbunden.

Sobald ein Werkzeug oder Werkstück über die Methode attachTo(...) mit dem Roboter verbunden ist, werden die Lastdaten von der Robotersteuerung berücksichtigt. Außerdem können alle Frames des verbundenen Objekts für die Bewegungsprogrammierung verwendet werden.

(>>> 9.3.6 "Lastdaten" Seite 147)

### 15.11.3.1 Werkzeug mit Roboterflansch verbinden

**Beschreibung** Der Ursprungs-Frame eines Werkzeugs wird über die Methode attachTo(...) mit dem Flansch eines in der Applikation verwendeten Roboters verbunden. Auf den Roboterflansch wird über die Methode getFlange() zugegriffen.

**Syntax** Werkzeug.attachTo(Roboter.getFlange());

**Erläuterung der Syntax**

Element	Beschreibung
Werkzeug	Name der Werkzeug-Variablen
Roboter	Name des Roboters

**Beispiel** Eine Handführspitze wird mit dem Roboterflansch verbunden.

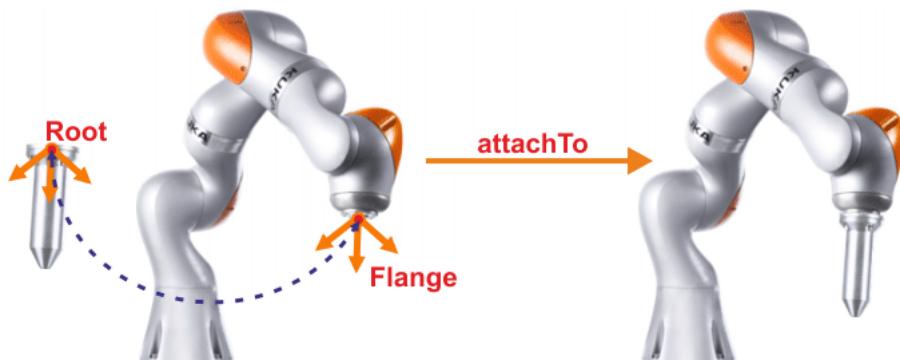


Abb. 15-11: Handführspitze mit Flansch verbinden

```
private LBR robot;
private Tool guidingTool;
...
public void run() {
    ...
    guidingTool.attachTo(robot.getFlange());
    ...
}
```

### 15.11.3.2 Werkstück mit anderen Objekten verbinden

**Beschreibung** Defaultmäßig wird der Ursprungs-Frame des Werkstücks verwendet, um sich mit dem Frame eines anderen Objekts zu verbinden.

Es kann aber auch jeder andere Frame, der für ein Werkstück angelegt wurde, als Bezugspunkt für die Verbindung mit einem anderen Objekt verwendet werden.

Frames für Werkzeuge und Werkstücke werden in der Sicht **Objektvorlagen** angelegt. Um einen Frame im Programm zu verwenden, wird das Werkzeug- oder Werkstück-Objekt mit der Methode getFrame(...) abgefragt. Diese erhält als Eingangsparameter den Pfad des Frames als String.

(>>> 9.3.4 "Frame für Werkzeug oder Werkstück anlegen" Seite 145)

**Syntax** Ursprungs-Frame für Verbindung verwenden:

Werkstück.attachTo(Objekt.getFrame("Ziel-Frame"));

Anderen Bezugs-Frame für Verbindung verwenden:

Werkstück.getFrame("Bezugs-Frame").attachTo(Objekt.getFrame("Ziel-Frame"));

### Erläuterung der Syntax

Element	Beschreibung
Werkstück	Name der Werkstück-Variablen
Bezugs-Frame	Bezugs-Frame des Werkstücks, der für die Verbindung mit dem anderen Objekt verwendet wird
Ziel-Frame	Frame des Objekts, mit dem der Bezugs-Frame des Werkstücks verbunden wird



Nach dem Attach stimmen der Bezugs-Frame des Werkstücks und der Ziel-Frame des mit ihm verbundenen Objekts überein.

### Beispiel 1

Ein Stift wird über seinen Ursprungs-Frame mit dem Greifer-Frame verbunden.

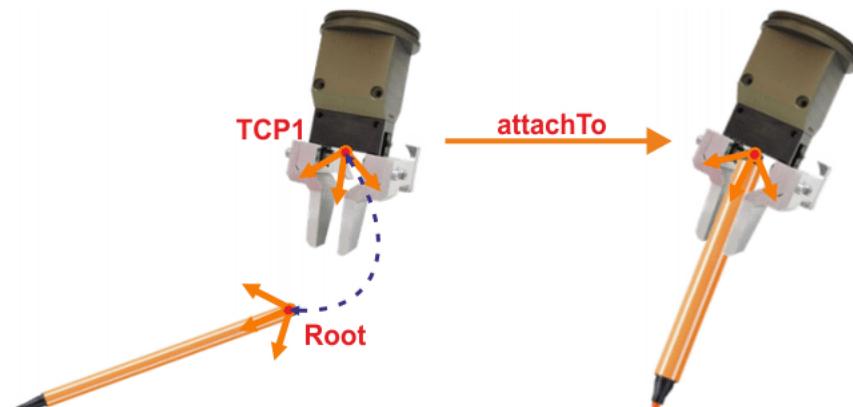


Abb. 15-12: Stift in Greifer (Verbindung über Ursprungs-Frame)

```
private LBR robot;
private Tool gripper;
private Workpiece pen;
...
public void run() {
    ...
    pen.attachTo(gripper.getFrame("/TCP1"));
    ...
}
```

### Beispiel 2

An der Greiferspitze ist ein 2. Frame definiert. Wenn dieser verwendet werden soll, um den Stift zu greifen, ist eine Verbindung über den Ursprungs-Frame des Stiftes nicht möglich. Für diesen Fall wurde ein Greipunkt am Stift angelegt. Dieser wird als Bezugs-Frame für die Verbindung mit dem Greifer angegeben.

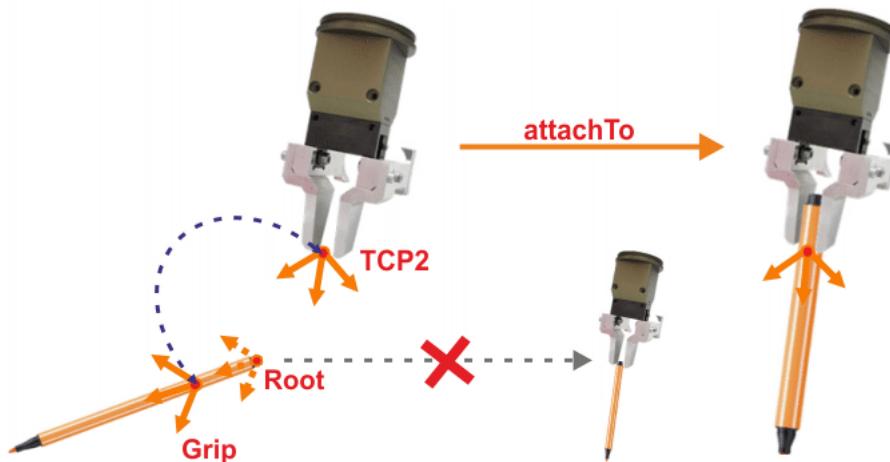


Abb. 15-13: Stift in Greifer (Verbindung über Grip-Frame)

```

private LBR robot;
private Tool gripper;
private Workpiece pen;
...
public void run() {
    ...
    pen.getFrame("/Grip").attachTo(gripper.getFrame("/TCP2"));
    ...
}

```

### 15.11.3 Verbindungen lösen

**Beschreibung** Wird ein Werkzeug demontiert oder ein Werkstück abgelegt, muss die Verbindung auch in der Applikation gelöst werden. Hierfür wird die Methode `detach()` verwendet.

**Syntax** `Objekt.detach();`

**Erläuterung der Syntax**

Element	Beschreibung
<code>Objekt</code>	Name der Objekt-Variablen

**Beispiel** Die Verbindung mit der Handführspitze wird gelöst.

```
guidingTool.detach();
```

### 15.11.4 Werkzeuge und Werkstücke bewegen

**Beschreibung** Jedes bewegbare Objekt einer Station kann mit `move(...)` und `moveAsync(...)` bewegt werden. Der Bezugspunkt der Bewegung ist abhängig vom Objekttyp:

- Wird ein Roboter bewegt, ist der Bezugspunkt immer der Roboterflansch-Mittelpunkt.
- Wird ein Werkzeug oder Werkstück bewegt, ist der Bezugspunkt defaultmäßig der Standard-Frame für Bewegungen, der für dieses Objekt in der Sicht **Objektvorlagen** festgelegt wurde.

(>>> 9.3.5 "Standard-Frame für Bewegungen festlegen" Seite 146)

In diesem Fall wird das Werkzeug oder Werkstück über den in der Applikation deklarierten Variablennamen direkt mit dem Bewegungsbefehl verknüpft.

- Es kann aber auch jeder andere Frame, der für ein Werkzeug oder Werkstück angelegt wurde, als Bezugspunkt der Bewegung programmiert werden.

In diesem Fall muss mithilfe der Methode `getFrame(...)` der Pfad zu dem Frame des Objekts angegeben werden, der für die Bewegung verwendet wird (ausgehend vom Ursprungs-Frame des Objekts).

#### Syntax

Standard-Frame des Objekts für Bewegung verwenden:

```
Objekt.move (Bewegung) ;
```

Anderen Frame des Objekts für Bewegung verwenden:

```
Objekt.getFrame ("Bewegter Frame") .move (Bewegung) ;
```

#### Erläuterung der Syntax

Element	Beschreibung
<i>Objekt</i>	Objekt der Station, das bewegt wird Hier wird der in der Applikation deklarierte und initialisierte Variablenname des Objekts angegeben.
<i>Bewegter Frame</i>	Pfad zum Frame des Objekts, der für die Bewegung verwendet wird
<i>Bewegung</i>	Bewegung, die ausgeführt wird

#### Beispiele

Die PTP-Bewegung zum Punkt P1 wird mit dem Standard-Frame des Greifers ausgeführt.

```
gripper.attachTo(robot.getFlange());
gripper.move(ptpgetApplicationData().getFrame("/P1"));
```

Die PTP-Bewegung zum Punkt P1 wird mit einem anderen Frame als dem Standard-Frame des Greifers ausgeführt, hier TCP1:

```
gripper.attachTo(robot.getFlange());
gripper.getFrame("/TCP1").move(ptpgetApplicationData().getFrame("/P1"));
```

Ein Stift wird gegriffen. Die nächste Bewegung ist eine PTP-Bewegung zum Punkt P20. Dieser Punkt wird mit dem Standard-Frame des Werkstücks Stift ausgeführt.

```
gripper.attachTo(robot.getFlange());
...
pen.attachTo(gripper.getFrame("/TCP1"));
pen.move(ptpgetApplicationData().getFrame("/P20"));
```

#### 15.11.5 Eigene Objektklassen definieren

##### Beschreibung

In den Objektvorlagen angelegte Werkzeuge, Werkstücke und feststehende geometrische Objekte basieren auf den Klassen `Tool`, `Workpiece` und `GeometricalObject`. Spezifische Eigenschaften oder Funktionalitäten, die Werkzeuge, Werkstücke und feststehende geometrische Objekte in der Regel besitzen, werden von diesen Basisklassen nicht berücksichtigt. Beispielsweise Funktionen zum Öffnen und Schließen bei einem Greifer.

Solche spezifischen Objekteigenschaften und Funktionalitäten können in eigenen Objektklassen definiert werden. Damit die benutzerdefinierten Objektklassen genauso wie die Basisklassen in Applikationen genutzt werden können, sind folgende Schritte erforderlich:

Schritt	Beschreibung
1	<p>Neue Objektklasse von geeigneter Basisklasse ableiten:</p> <ul style="list-style-type: none"> <li>■ Basisklasse für Werkzeuge: com.kuka.roboticsAPI.geometricModel.Tool</li> <li>■ Basisklasse für Werkstücke: com.kuka.roboticsAPI.geometricModel.Workpiece</li> <li>■ Basisklasse für feststehende geometrische Objekte: com.kuka.roboticsAPI.geometricModel.GeometricObject</li> </ul> <p><b>Hinweis:</b> Der Konstruktor der erstellten Objektklasse muss folgende Eigenschaften besitzen:</p> <ul style="list-style-type: none"> <li>■ Sichtbarkeit <b>public</b></li> <li>■ Übergabeparameter vom Typ String (Name der in der Applikation verwendeten Objektvorlage wird übergeben)</li> </ul>
2	Objekteigenschaften und Funktionalitäten in der neuen Objektklasse definieren.
3	<p>In den Objektvorlagen den gewünschten Objekten die neue Objektklasse zuweisen. Hierzu in der Sicht <b>Eigenschaften</b> unter <b>Klasse der Vorlage</b> den vollständigen Bezeichner der Objektklasse eintragen.</p> <p><b>Hinweis:</b> Objektvorlagen, die eine von einer Basisklasse abgeleitete Objektklasse verwenden, werden in einer Applikation wie diese über den Befehl <code>getApplicationData().createFromTemplate(...)</code> initialisiert.</p>

**Vorgehensweise****Neue Objektklasse von Basisklasse ableiten:**

1. Gewünschtes Sunrise-Projekt im **Paket-Explorer** markieren.
2. Menüfolge **Datei > Neu > Klasse** wählen. Das Fenster **Neue Java-Klasse** öffnet sich.
3. Im Feld **Paket**: einen Namen für das Java-Paket eingeben, in dem die neue Klasse erstellt werden soll, z. B. **tools**.
4. Im Feld **Name**: einen Namen für die neue Klasse eingeben.
5. Rechts vom Feld **Superklasse**: auf **Durchsuchen...** klicken. Das Fenster **Superklassenauswahl** öffnet sich.
6. Im Feld **Typ auswählen** den Namen der Basisklasse eingeben, z. B. **Tool** und die Auswahl mit **OK** bestätigen. Der Name der Basisklasse wird nun im Feld **Superklasse**: angezeigt.
7. Bei der Checkbox **Konstruktoren aus Superklasse** das Häkchen setzen.
8. Auf **Fertigstellen** klicken. Das Java-Paket mit der neu erstellten Klasse wird im Quellenordner des Sunrise-Projekts eingefügt und im Editoren-Bereich geöffnet.
9. Die benötigten Felder und Methoden können jetzt definiert werden.

**Beispiel****Schritt 1:**

Für einen Greifer wird die Objektklasse Gripper mit der oben beschriebenen Vorgehensweise erstellt. Die Klasse Gripper leitet von der Basisklasse Tool ab und ergänzt die Basisklasse um die Funktionen zum Öffnen und Schließen des Greifers.

```

1 package tools;
2 import com.kuka.roboticsAPI.geometricModel.Tool;
3 public class Gripper extends Tool {
4     public Gripper(String name) {
5         super(name);

```

```

6      // TODO Automatisch generierter Konstruktorstub
7  }
8 /**
9 * Opens the gripper
10 */
11 public void openGripper() {
12     ...
13 }
14 /**
15 * Closes the gripper
16 */
17 public void closeGripper() {
18     ...
19 }
20
21 public Gripper(String name, LoadData load) {
22     super(name, load);
23     // TODO Automatisch generierter Konstruktorstub
24 }
25 }
```

Zeile	Beschreibung
1	Name des Java-Pakets, in dem die Klasse Gripper enthalten ist
4 ... 19	Standardkonstruktor der Klasse Gripper zur Definition der Greiferfunktionen
11 ... 13	Methode openGripper() zum Öffnen des Greifers
17 ... 19	Methode closeGripper() zum Schließen des Greifers
21 ... 24	Erweiterter Konstruktor der Klasse Gripper (wird nicht benötigt)

### Schritt 2:

Für den Greifer ist eine Objektvorlage mit dem Namen "ExampleGripper" angelegt. Der Objektvorlage wird die Objektklasse Gripper zugewiesen:

- Eintrag in der Sicht **Eigenschaften** unter **Klasse der Vorlage**: tools.Gripper  
Der Name des Java-Pakets, in dem die Klasse Gripper enthalten ist, muss mit angegeben werden.

### Schritt 3:

Die Objektklasse Gripper und die zugehörigen Funktionen können in der Applikation verwendet werden.

```

1 public class ExampleApplication extends RoboticsAPIApplication {
2     private Controller kuka_Sunrise_Cabinet;
3     private LBR lbr_iwa;
4     private Gripper gripper;
5
6     public void initialize() {
7         ...
8         gripper = createFromTemplate("ExampleGripper");
9         gripper.attachTo(lbr_iwa.getFlange());
10    }
11
12    public void run() {
13        ...
14        gripper.openGripper();
15        gripper.move(
```

```

17         lingetApplicationData().getFrame("GripPos")));
18         gripper.closeGripper();
19         ...
20     }

```

<b>Zeile</b>	<b>Beschreibung</b>
4	Ein Werkzeug vom Typ Gripper wird deklariert. Das Werkzeug verfügt über die mit der Objektklasse Gripper definierten Greiferfunktionen.
8, 9	Das Werkzeug wird mit der Objektvorlage mit dem Namen "ExampleGripper" initialisiert und mit dem Roboterflansch verbunden.
15 ... 18	Die für die Klasse Gripper definierten Greiferfunktionen werden verwendet, um einen Greifprozess zu programmieren (Greifer öffnen, Greifposition anfahren, Greifer schließen).

### 15.11.6 Kommandieren von Lastwechseln an Sicherheitssteuerung

#### Beschreibung

Die Lastdaten eines Werkstücks werden von der Sicherheitssteuerung für die Berechnung der externen Momente benötigt. Die Sicherheitssteuerung kann ausschließlich die Lastdaten sicherheitsgerichteter Werkstücke verarbeiten.

(>>> 9.3.8 "Sicherheitsgerichtete Werkstücke" Seite 152)

Während eines Prozesses kann es durch Aufnehmen und Ablegen unterschiedlicher Werkstücke zu Lastwechseln kommen. Bei der Kollisionserkennung muss der Benutzer der Sicherheitssteuerung über die Methode `setSafetyWorkpiece(...)` mitteilen, welches sicherheitsgerichtete Werkstück aktuell aktiv ist. Dazu wird dieses Werkstück als Eingangsparameter übergeben.



Es muss immer ein sicherheitsgerichtetes Werkstück an `setSafetyWorkpiece(...)` übergeben werden. Wird ein nicht sicherheitsgerichtetes Werkstück übergeben, tritt ein Ausnahmefehler auf.

Die Methode `setSafetyWorkpiece(...)` gehört zur Klasse LBR und kann sowohl in Roboter-Applikationen als auch in Hintergrund-Tasks verwendet werden. Voraussetzung für die Übergabe eines sicherheitsgerichteten Werkstücks an die Methode ist, dass eine Instanz des Werkstücks aus den Objekt-Vorlagen erzeugt worden ist.

(>>> 15.11.2 "Werkzeuge und Werkstücke initialisieren" Seite 310)

Um ein aktives sicherheitsgerichtetes Werkstück zu deaktivieren und der Sicherheitssteuerung mitzuteilen, dass kein sicherheitsgerichtetes Werkstück mehr gegriffen ist, wird der Methode `setSafetyWorkpiece(...)` der Wert `null` übergeben.

Durch `setSafetyWorkpiece(...)` wird der Lastwechsel für die Sicherheitssteuerung kommandiert. Sollen die Werkstück-Lastdaten auch im nicht sicherheitsgerichteten Teil der Robotersteuerung berücksichtigt werden, muss der Lastwechsel zusätzlich mit den entsprechenden Befehlen programmiert werden.

(>>> 15.11.3.2 "Werkstück mit anderen Objekten verbinden" Seite 311)

#### Syntax

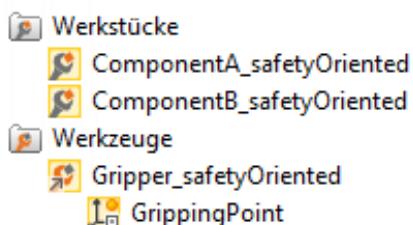
`lbr.setSafetyWorkpiece(Werkstück);`

## Erläuterung der Syntax

Element	Beschreibung
<i>lbr</i>	Typ: LBR Name des Roboters, für den der Lastwechsel programmiert wird
<i>Werkstück</i>	Typ: Workpiece Sicherheitsgerichtetes Werkstück, dessen Lastdaten an die Sicherheitssteuerung übergeben werden sollen Wenn kein sicherheitsgerichtetes Werkstück mehr berücksichtigt werden soll, muss <code>null</code> übergeben werden.

### Beispiel

Ein sicherheitsgerichtetes Werkzeug und 2 sicherheitsgerichtete Werkstücke werden in den Objekt-Vorlagen angelegt.



**Abb. 15-14: Werkstücke und Werkzeug (Objekt-Vorlagen)**

Das Werkzeug erhält den Frame "GrippingPoint", der als Greifpunkt für Werkstücke dient und als Standard-Frame für Bewegungen markiert ist.

In der Applikation wird das Werkstück "ComponentA\_safetyOriented" aufgenommen und abgelegt. Anschließend wird das Werkstück "ComponentB\_safetyOriented" aufgenommen. Alle 3 Lastwechsel sollen sowohl im sicherheitsgerichteten als auch im nicht sicherheitsgerichteten Teil der Robotersteuerung berücksichtigt werden.

```
public class ChangeOfLoadExample extends RoboticsAPIApplication {
    ...
    // safety-oriented tool and workpieces
    private Tool gripper;
    private Workpiece componentA, componentB;

    public void initialize() {
        ...
        // initialize safety-oriented components
        gripper = getApplicationData().
            createFromTemplate("Gripper_safetyOriented");
        componentA = getApplicationData().
            createFromTemplate("ComponentA_safetyOriented");
        componentB = getApplicationData().
            createFromTemplate("ComponentB_safetyOriented");

        // attach gripper to robot flange
        gripper.attachTo(lbr_iwia.getFlange());
    }

    public void run() {
        ...
        // after pick-up, attach workpiece to set load data for
        // motion control
        componentA.attachTo(gripper.getDefaultMotionFrame());
        // set load data for safety controller
    }
}
```

```

lbr_iowa.setSafetyWorkpiece(componentA);
...
// after putting it down, detach workpiece to no longer
// consider its load for motion control
gripper.detach();
// workpiece is no longer considered for safety
// controller
lbr_iowa.setSafetyWorkpiece(null);
...
// pick-up of second workpiece
componentB.attachTo(gripper.getDefaultMotionFrame());
lbr_iowa.setSafetyWorkpiece(componentB);
...
}
...
}

```

## 15.12 Ein-/Ausgänge

Beim Exportieren einer E/A-Konfiguration aus WorkVisual wird im zugehörigen Sunrise-Projekt für jede E/A-Gruppe eine eigene Java-Klasse erstellt. Jede dieser Java-Klassen enthält die für die Programmierung benötigten Methoden, um lesend auf die Ein-/Ausgänge einer E/A-Gruppe und schreibend auf die Ausgänge einer E/A-Gruppe zuzugreifen.



Der Quellcode der Java-Klassen des Pakets **com.kuka.generated.ioAccess** darf nicht manuell geändert werden. Um die Funktionalität einer E/A-Gruppe zu erweitern, können von den erzeugten Klassen weitere Klassen abgeleitet oder Objekte dieser Klassen weiterverwendet werden, z. B. als Felder eigener Klassen (Aggregieren).

Um die Ein-/Ausgänge einer E/A-Gruppe in der Applikation verwenden zu können, muss der Benutzer ein Datenfeld vom Typ der E/A-Gruppe anlegen und initialisieren.

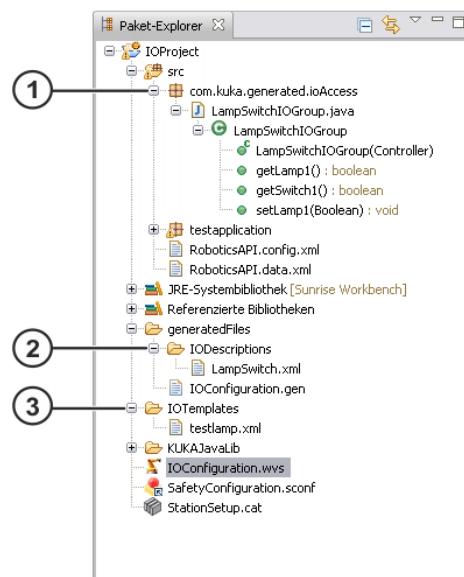


Abb. 15-15: Projektstruktur nach Export E/A-Konfiguration

Pos.	Beschreibung
1	<p>Java-Paket <b>com.kuka.generated.ioAccess</b></p> <p>In dem Paket werden die für eine E/A-Gruppe erstellte Klasse sowie die zugehörigen Methoden abgelegt.</p> <p>Die Java-Klasse <i>NameIOGroup.java</i> (hier: <i>LampSwitchIOLroup.java</i>) enthält folgende Elemente:</p> <ul style="list-style-type: none"> <li>■ Klassenname der E/A-Gruppe: <i>NameIOGroup</i></li> <li>■ Konstruktor, um der E/A-Gruppe die Robotersteuerung zuzuweisen: <i>NameIOGroup (Controller)</i></li> <li>■ Get- und set-Methode für jeden konfigurierten Ausgang: <i>getAusgang ()</i>, <i>setAusgang (Wert)</i></li> <li>■ Get-Methode für jeden konfigurierten Eingang: <i>getEingang ()</i></li> </ul>
2	<p>Ordner <b>generatedFiles &gt; IODescriptions</b></p> <p>Die Daten einer E/A-Gruppe werden in einer XML-Datei gespeichert. Die XML-Datei kann angezeigt, darf aber nicht bearbeitet werden.</p>
3	<p>Ordner <b>IOTemplates</b></p> <p>Die Daten einer als Vorlage gespeicherten E/A-Gruppe sind in einer XML-Datei gespeichert. Die XML-Datei kann angezeigt, darf aber nicht bearbeitet werden.</p> <p>Eine Vorlage kann in ein anderes Sunrise-Projekt kopiert werden, um sie dort zur Verfügung zu stellen. Die Vorlage kann dann in WorkVisual importiert, bearbeitet und neu exportiert werden.</p> <p>(&gt;&gt;&gt; 11.5.8 "E/A-Gruppe aus Vorlage importieren" Seite 171)</p> <p>(&gt;&gt;&gt; 11.5.7 "E/A-Gruppe als Vorlage exportieren" Seite 170)</p>



Der Ordner **generatedFiles** wird vom System genutzt und darf nicht für die Ablage eigener Dateien verwendet werden.

### 15.12.1 Datenfeld für E/A-Gruppe anlegen

#### Beschreibung

Durch die Deklaration des Datenfeldes vom Typ der E/A-Gruppe wird das Java-Paket com.kuka.generated.ioAccess mit den Klassen und Methoden der E/A-Gruppe automatisch importiert.

#### Syntax

`private Klassename Feldname;`

#### Erläuterung der Syntax

Element	Beschreibung
<code>private</code>	Das Schlüsselwort <code>private</code> kennzeichnet Datenfelder, die nur von der umgebenden Klasse verwendet werden können.
<code>Klassename</code>	<p>Typ des Datenfeldes</p> <p>Klassename der E/A-Gruppe:</p> <ul style="list-style-type: none"> <li>■ <i>NameIOGroup</i></li> </ul> <p><i>Name</i> = Name der E/A-Gruppe, wie in WorkVisual definiert</p>
<code>Feldname</code>	Name des Datenfeldes, der bei der Programmierung verwendet wird

#### Beispiel

Für die E/A-Gruppe "SwitchLamp" wird das Datenfeld "switchLamp" angelegt.

```
public class RobotApplication extends RoboticsAPIApplication {
    ...
    private Controller controller;
    private SwitchLampIOGroup switchLamp;
    ...
}
```

### 15.12.2 Datenfeld für E/A-Gruppe initialisieren

**Beschreibung** Bei der Initialisierung des Datenfeldes wird über den Konstruktor der Klasse die Robotersteuerung angegeben, an der die Ein-/Ausgänge der Gruppe über den Feldbus angebunden sind.

**Syntax** *Feldname* = new *Klassenname*(*Controller*) ;

**Erläuterung der Syntax**

Element	Beschreibung
<i>Feldname</i>	Name des Datenfeldes
new	Operator, mit dem eine neue Instanz der Klasse <i>Klassenname</i> erzeugt wird
<i>Klassenname</i>	Klassenname der E/A-Gruppe: ■ <i>NameIOGroup</i> <i>Name</i> = Name der E/A-Gruppe, wie in WorkVisual definiert
<i>Controller</i>	Name des Objekts, das der Robotersteuerung zugeordnet ist  Die Zuordnung erfolgt in der Regel in der Methode initialize() einer Roboter-Applikation. ( <a href="#">&gt;&gt;&gt; 15.1.2 "Aufbau einer Roboter-Applikation"</a> Seite 283)

**Beispiel** Das Datenfeld "switchLamp" wird initialisiert.

```
public void initialize() {
    ...
    switchLamp = new SwitchLampIOGroup(controller);
    ...
}
```

### 15.12.3 Ein-/Ausgänge lesen

**Beschreibung** Um den Zustand eines Ein-/Ausgangs abzufragen, wird die get-Methode des Ein-/Ausgangs verwendet.

**Syntax** *Feldname*.get*Ein-/Ausgang*() ;

**Erläuterung der Syntax**

Element	Beschreibung
<i>Feldname</i>	Name des Datenfeldes
<i>Ein-/Ausgang</i>	Name des Ein-/Ausgangs (wie in WorkVisual definiert)

**Beispiel** Der Zustand des Schalters am Eingang "Switch1" und der Zustand der Lampe am Ausgang "Lamp1" wird abgefragt.

```
public void run() {
    ...
    switchLamp.getLamp1();
    switchLamp.getSwitch1();
    ...
}
```

### 15.12.4 Ausgänge setzen

**⚠️ WARNING**

Ausgänge werden in bestimmten Fällen geschaltet, obwohl eine sicherheitsgerichtete Stopp-Anforderung anliegt (z. B. bei gedrücktem NOT-HALT oder verletzter Raumüberwachung). Dadurch kann es zu unerwarteten Bewegungen der angeschlossenen Peripherie kommen (z. B. Öffnen eines Greifers).

Beispielsweise können folgende Fälle auftreten:

- Hintergrund-Task schaltet Ausgang.
- Über Benutzertaste aufgerufene Funktion schaltet Ausgang.
- Roboter-Applikationen laufen bei einer Stopp-Anforderung bis zum nächsten synchronen Bewegungsbefehl weiter. Code, der bis dahin zur Ausführung kommt, schaltet Ausgang.

Das beschriebene Verhalten kann auch erwünscht sein, es darf von ihm jedoch keinerzeit eine Gefahr für Mensch und Maschine ausgehen. Dies muss durch den Anlagenbauer sichergestellt werden, z. B. indem Ausgänge mit Gefährdungspotential stromlos geschalten werden.

**HINWEIS**

Es ist unzulässig, Ausgänge, die Systemzustände an die übergeordnete Steuerung melden, in einer Roboter-Applikation zu setzen. Wenn dies nicht beachtet wird, kann es zu einem Feherverhalten der übergeordneten Steuerung und Sachschaden kommen.

#### Beschreibung

Um den Wert eines Ausgangs zu ändern, wird die `set`-Methode des Ausgangs verwendet.



Für Eingänge stehen keine `set`-Methoden zur Verfügung. Sie können nur gelesen werden.

#### Syntax

`Feldname.setAusgang (Wert);`

#### Erläuterung der Syntax

Element	Beschreibung
<code>Feldname</code>	Name des Datenfeldes
<code>Ausgang</code>	Name des Ausgangs (wie in WorkVisual definiert)
<code>Wert</code>	Wert des Ausgangs. Der zu übergebende Datentyp des Wertes ist abhängig vom Typ des Ausgangs.

#### Beispiel

Die Lampe am Ausgang "Lamp1" wird eingeschalten und nach 2000 ms wieder ausgeschalten.

```
public void run() {
    ...
    switchLamp.setLamp1(true);
    ThreadUtil.milliSleep(2000);
    switchLamp.setLamp1(false);
    ...
}
```

### 15.13 Achsmomente abfragen

#### Beschreibung

In jeder Achse des KUKA LBR iiwa befindet sich ein Gelenkmomenten-Sensor, der das auf die Achse wirkende Moment misst. Über die Methode `getMeasuredTorque()` der Klasse LBR können die gemessenen Momentenwerte in der Applikation abgefragt und ausgewertet werden.

Häufig interessieren nicht die reinen Messwerte, sondern nur die von außen wirkenden Momente ohne den Anteil, der durch die Gewichtskraft der Roboterruktur und Masseträgheiten bei Bewegung entsteht. Diese Werte werden als externe Momente bezeichnet. Auf die externen Momente kann über die LBR-Methode `getExternalTorque()` zugegriffen werden.

Beide Befehle liefern ein Objekt vom Typ `TorqueSensorData` zurück, das die Momentensensor-Daten aller Achsen enthält. Von diesem Objekt können dann wahlweise mit `getTorqueValues()` alle Werte als Array oder mit `getSingleTorqueValue(...)` ein einzelner Achswert abgefragt werden.



Bei der Abfrage der Daten der Momentensensoren mittels Java liegt kein Echtzeitverhalten vor. Das bedeutet, dass die im Programm vom System gelieferten Daten bereits einige Millisekunden früher erfasst wurden.

## Syntax

Gemessene Sensordaten abfragen:

```
TorqueSensorData measuredData = lbr.getMeasuredTorque();
```

Extern wirkende Momentendaten abfragen:

```
TorqueSensorData externalData = lbr.getExternalTorque();
```

Momentenwerte aller Achsen von den Sensordaten abfragen:

```
double[] allValues = measuredData|externalData.getTorqueValues();
```

Momentenwert einer bestimmten Achse von den Sensordaten abfragen:

```
double singleValue =
measuredData|externalData.getSingleTorqueValues(joint);
```

## Erläuterung der Syntax

Element	Beschreibung
<code>measuredData</code>	Typ: <code>TorqueSensorData</code> Variable für den Rückgabewert von <code>getMeasuredTorque()</code> . Der Rückgabewert enthält die gemessenen Sensordaten.
<code>externalData</code>	Typ: <code>TorqueSensorData</code> Variable für den Rückgabewert von <code>getExternalTorque()</code> . Der Rückgabewert enthält die extern wirkenden Drehmomente.
<code>lbr</code>	Typ: LBR Name des Roboters, von dem die Sensordaten abgefragt werden
<code>allValues</code>	Typ: <code>double[]</code> ; Einheit: Nm Array mit allen Momentenwerten, die von den Sensordaten abgefragt werden
<code>singleValue</code>	Typ: <code>double</code> ; Einheit: Nm Momentenwert der Achse, der von den Sensordaten abgefragt wird
<code>joint</code>	Typ: Enum vom Typ <code>JointEnum</code> Achse, deren Momentenwert abgefragt werden soll ■ <b>JointEnum.J1 ... JointEnum.J12</b> : Achse A1 ... A12

## Beispiel

Zu einem bestimmten Prozess-Schritt werden die gemessenen und extern wirkenden Momente in allen Achsen abgefragt und in einem Array für die spätere Auswertung gespeichert. Das gemessene Moment in Achse A2 wird ausgelesen und auf der smartHMI ausgegeben.

```

TorqueSensorData measuredData = lbr.getMeasuredTorque();
TorqueSensorData externalData = lbr.getExternalTorque();

double[] measuredTorques = measuredData.getTorqueValues();
double[] externalTorques = externalData.getTorqueValues();

double torqueA2 = measuredData.getSingleTorqueValue(JointEnum.J2);
getLogger().info("Currently measured torque for joint 2 [Nm]:" + torqueA2);

```

## 15.14 Auslesen kartesischer Kräfte und Momente

Die externen kartesischen Kräfte und Momente, die aktuell auf den Roboterflansch, TCP eines Werkzeugs oder beliebigen Punkt eines gegriffenen Werkstücks wirken, können ausgelesen werden.

Folgende Punkte sind dabei zu beachten:

- Die Drehmomente der Achsen werden von den Momentensensoren gemessen.
- Aus den gemessenen Drehmomenten werden die kartesischen Kräfte und Momente berechnet.
- Die Zuverlässigkeit der berechneten Werte kann in extremen Posen, z. B. Strecklagen oder Singularitäten, stark abnehmen.
- In der RoboticsAPI stehen Methoden zur Verfügung, um die Qualität und Gültigkeit der berechneten Werte zu prüfen.



Bei einer Änderung der Lastdaten, z. B. durch den attachTo-Befehl, kann die Abfrage erst nach dem Senden eines Bewegungsbefehls an die Steuerung durchgeführt werden. Hierfür ist eine Nullraum-Bewegung oder das Senden des Bewegungsbefehls positionHold(...) ausreichend.



Die kartesischen Kräfte und Momente werden auf Basis der Messwerte der Gelenkmomenten-Sensoren geschätzt. Für die Berechnung ist die Angabe eines Kraftangriffspunktes nötig. Die ermittelten Kräfte und Momente für den angegebenen Angriffspunkt sind physikalisch nur dann aussagekräftig, wenn an keinen weiteren Punkten der Roboterstruktur externe Kräfte wirken.

### 15.14.1 Berechnete Kraft-Momenten-Daten abfragen

#### Beschreibung

Um die externen kartesischen Kräfte und Momente, die aktuell auf einen bestimmten Punkt wirken, abzufragen wird die Methode getExternalForce-Torque(...) der Klasse LBR verwendet.

Als Überabeparameter erhält die Methode einen Frame. Der übergebene Frame ist der Bezugs-Frame für die Berechnung der Kräfte und Momente, z. B. die Spitze eines Messtasters. Für die Position, die durch den Frame beschrieben wird, berechnet die Methode die extern anliegenden Kräfte und Momente.

Für eine physikalisch aussagekräftige Berechnung muss der übergebene Frame einen Punkt beschreiben, der mechanisch starr mit dem Flansch verbunden ist. In der Frame-Struktur muss der angegebene Frame ebenfalls statisch mit dem Roboterflansch-Frame verbunden sein.

Optional kann der Methode ein zweiter Frame als Parameter übergeben werden. Dieser Frame gibt die Orientierung eines Koordinatensystems an, in dem die Kräfte und Momente dargestellt werden.

**Syntax**

```
ForceSensorData data = lbr.getExternalForceTorque(
    measureFrame<, orientationFrame>);
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>data</i>	Typ: ForceSensorData  Variable für den Rückgabewert von getExternalForceTorque(...). Der Rückgabewert enthält die berechneten Kraft-Momenten-Daten.
<i>lbr</i>	Typ: LBR  Name des Roboters
<i>measure Frame</i>	Typ: AbstractFrame  Bezugs-Frame, für den die aktuell wirkenden Kräfte und Momente berechnet werden
<i>orientation Frame</i>	Typ: AbstractFrame  Optional: Orientierung des Frames, in dem die Kräfte und Momente dargestellt werden

**Beispiele**

Abfrage der Kraft-Momenten-Daten am Roboterflansch:

```
ForceSensorData data =
    robot.getExternalForceTorque(robot.getFlange());
```

Abfrage der Kraft-Momenten-Daten am Roboterflansch, bezogen auf die Orientierung des Welt-Koordinatensystems:

```
ForceSensorData data =
    robot.getExternalForceTorque(robot.getFlange(),
        World.Current.getRootFrame());
```

### 15.14.2 Einzelne Kraft-Momenten-Werte abfragen

**Beschreibung**

Die mit getExternalForceTorque() ausgelesenen Kraft-Momenten-Daten können mithilfe der Methoden getForce() und getTorque(...) der Klasse ForceSensorData getrennt voneinander abgefragt werden.

Das Ergebnis dieser Abfragen ist jeweils ein Vektor. Mit den get-Methoden der Klasse Vector können die Werte für jeden Freiheitsgrad einzeln abgefragt werden.

(>>> 15.14.4 "Einzelwerte von einem Vektor abfragen" Seite 327)

**Syntax**

Kraftvektor abfragen:

```
Vector force = data.getForce();
```

Momentenvektor abfragen:

```
Vector torque = data.getTorque();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>force</i>	Typ: Vector (com.kuka.roboticsAPI.geometricModel.math)  Vektor mit den kartesischen Kräften, die in X-, Y- und Z-Richtung wirken (Einheit: N)

Element	Beschreibung
<i>torque</i>	Typ: Vector (com.kuka.roboticsAPI.geometricModel.math) Vektor mit den kartesischen Momenten, die um die X-, Y- und Z-Achse wirken (Einheit: Nm)
<i>data</i>	Typ: ForceSensorData Variable für den Rückgabewert von getExternalForceTorque(...). Der Rückgabewert enthält die berechneten Kraft-Momenten-Daten.

**Beispiel**

Abfrage der kartesischen Kraft, die aktuell in X-Richtung am Roboterflansch-Frame wirkt:

```
ForceSensorData data =
robot.getExternalForceTorque(robot.getFlange());

Vector force = data.getForce();
double forceInX = force.getX();
```

**15.14.3 Zuverlässigkeit der berechneten Kraft-Momenten-Werte prüfen****Beschreibung**

Die berechneten Kraft-Momenten-Werte können in ungünstigen Stellungen des Roboters von den tatsächlich anliegenden Kräften und Momenten abweichen. Besonders in der Nähe von Singularitäten unterliegen einige der berechneten Werte einer starken Unsicherheit und können ungültig sein. Das gilt je nach Achsstellung nur für einen Teil der berechneten Werte.

Die Qualität und die Gültigkeit der berechneten Werte kann bewertet und im Programm abgefragt werden. Hierzu stehen folgende Methoden der Klasse ForceSensorData zur Verfügung:

- **getForceInaccuracy(), getTorqueInaccuracy()**

Die Ungenauigkeiten der berechneten Kraftwerte und der berechneten Momentenwerte können abgefragt werden. Das Ergebnis dieser Abfragen ist jeweils ein Vektor. Mit den get-Methoden der Klasse Vector können die Werte für jeden Freiheitsgrad einzeln abgefragt werden.

Abhängig von der Achsstellung kann die Qualität der berechneten Werte für die einzelnen Freiheitsgrade unterschiedlich sein. Durch die Abfrage von Einzelwerten können diejenigen Freiheitsgrade bestimmt werden, für die die Berechnung der Kräfte und Momente in der aktuellen Pose gültige Werte liefert.

(>>> 15.14.4 "Einzelwerte von einem Vektor abfragen" Seite 327)

- **isForceValid(...), isTorqueValid(...)**

Es kann abgefragt werden, ob die berechneten Kraft- und Momentenwerte gültig sind. Den Methoden wird jeweils ein Grenzwert für die maximal zulässige Ungenauigkeit bis zu dem die berechneten Werte noch gültig sind als Parameter übergeben.

**Syntax**

Ungenauigkeit der berechneten Kraft- und Momentenwerte abfragen:

```
Vector force = data.getForceInaccuracy();
Vector torque = data.getTorqueInaccuracy();
```

Gültigkeit der Kraft- und Momentenwerte abfragen:

```
boolean valid = data.isForceValid(tolerance);
boolean valid = data.isTorqueValid(tolerance);
```

## Erläuterung der Syntax

Element	Beschreibung
<i>force</i>	Typ: Vector (com.kuka.roboticsAPI.geometricModel.math) Vektor mit den Werten für die Ungenauigkeit, mit der die kartesischen Kräfte, die in X-, Y- und Z-Richtung wirken, berechnet wurden (Einheit: N)
<i>torque</i>	Typ: Vector (com.kuka.roboticsAPI.geometricModel.math) Vektor mit den Werten für die Ungenauigkeit, mit der die kartesischen Momente, die um die X-, Y- und Z-Achse wirken, berechnet wurden (Einheit: Nm)
<i>data</i>	Typ: ForceSensorData Variable für den Rückgabewert der Methode getExternalForceTorque(...). Der Rückgabewert enthält die berechneten Kraft-Momenten-Daten.
<i>tolerance</i>	Typ: double; Einheit: N oder Nm Grenzwert für die maximal zulässige Ungenauigkeit bis zu dem die berechneten Kraft- oder Momentenwerte noch gültig sind
<i>valid</i>	Typ: boolean Variable für den Rückgabewert von isForceValid(...) oder isTorqueValid(...) <ul style="list-style-type: none"> <li>■ <b>true:</b> Der Ungenauigkeitswert ist in allen kartesischen Richtungen kleiner als der mit <i>tolerance</i> definierte Grenzwert oder gleich groß.</li> <li>■ <b>false:</b> Der Ungenauigkeitswert überschreitet in einer oder mehreren kartesischen Richtungen den Wert von <i>tolerance</i>.</li> </ul>

## Beispiel

Ein bestimmter Anweisungsblock soll nur dann ausgeführt werden, wenn die externen kartesischen Kräfte, die aktuell entlang der Achsen des Flansch-Koordinatensystems wirken, mit einer Genauigkeit von 20 N oder besser berechnet wurden.

```
ForceSensorData data =
robot.getExternalForceTorque(robot.getFlange());

if (data.isForceValid(20)) {
    //do something
}
```

### 15.14.4 Einzelwerte von einem Vektor abfragen

Methoden, die Daten von einem Frame abfragen, liefern in der Regel ein Objekt der Klasse Vector (Paket: com.kuka.roboticsAPI.geometricModel.math) zurück. Die Komponenten des Vektors können einzeln abgefragt werden.

## Übersicht

Folgende Methoden der Klasse Vector stehen zur Verfügung:

Methode	Beschreibung
getX()	Rückgabetyp: double Abfrage der x-Komponente des Vektors
getY()	Rückgabetyp: double Abfrage der y-Komponente des Vektors

Methode	Beschreibung
getZ()	Rückgabetyp: double Abfrage der z-Komponente des Vektors
get( <i>index</i> )	Rückgabetyp: double Abfrage der durch den Parameter <i>index</i> festgelegten Komponente Werte von <i>index</i> (Typ: int): <ul style="list-style-type: none"> <li>■ 0: x-Komponente des Vektors</li> <li>■ 1: y-Komponente des Vektors</li> <li>■ 2: z-Komponente des Vektors</li> </ul>

## 15.15 Abfrage der Roboterposition

Die achsspezifische und kartesische Roboterposition kann in der Applikation abgefragt werden. Es kann jeweils die Istposition und die Sollposition abgefragt werden.

### Übersicht

Folgende Methoden der Klasse Robot stehen zur Verfügung:

Methode	Beschreibung
getCommandedCartesianPosition(...)	Rückgabetyp: Frame Abfrage der kartesischen Sollposition
getCommandedJointPosition()	Rückgabetyp: JointPosition Abfrage der achsspezifischen Sollposition
getCurrentCartesianPosition(...)	Rückgabetyp: Frame Abfrage der kartesischen Istposition
getCurrentJointPosition()	Rückgabetyp: JointPosition Abfrage der achsspezifischen Istposition
getPositionInformation(...)	Rückgabetyp: PositionInformation Abfrage der kartesischen Positionsinformationen Der Rückgabewert enthält folgende Informationen: <ul style="list-style-type: none"> <li>■ achsspezifische Istposition</li> <li>■ achsspezifische Sollposition</li> <li>■ kartesische Istposition</li> <li>■ kartesische Sollposition</li> <li>■ kartesische Soll-Ist-Differenz (rotatorisch)</li> <li>■ kartesische Soll-Ist-Differenz (translatorisch)</li> </ul>

### 15.15.1 Achsspezifische Ist- oder Sollposition abfragen

#### Beschreibung

Für die Abfrage der achsspezifischen Ist- oder Sollposition des Roboters wird die Position der Roboterachsen zunächst in einer Variablen vom Typ JointPosition gespeichert.

Von dieser Variablen können dann die Achswerte einzeln abgefragt werden. Die gewünschte Achse wird entweder über ihren Index oder den entsprechenden JointEnum-Wert angegeben.

#### Syntax

Achsspezifische Istposition abfragen:

```
JointPosition position = robot.getCurrentJointPosition();
```

Achsspezifische Sollposition abfragen:

```
JointPosition position = robot.getCommandedJointPosition();
```

Achswerte einzeln abfragen:

```
double value = position.get(axis);
```

### Erläuterung der Syntax

Element	Beschreibung
<i>position</i>	Typ: JointPosition  Variable für den Rückgabewert. Der Rückgabewert enthält die abgefragten Achspositionen.
<i>robot</i>	Typ: Robot  Name des Roboters, von dem die Achspositionen abgefragt werden
<i>value</i>	Typ: double; Einheit: rad  Achswinkel der abgefragten Achse
<i>axis</i>	Typ: int  1. Möglichkeit: Index der Achse angeben, deren Achswert abgefragt wird  ■ <b>0 ... 11</b> : Achse A1 ... A12  2. Möglichkeit: JointEnum-Wert der Achse angeben, deren Achswert abgefragt wird  ■ <b>JointEnum.J1 ... JointEnum.J12</b> : Achse A1 ... A12

### Beispiel

Zunächst wird die achsspezifische Istposition des Roboters und dann der Achswert der Achse A3 über den Index der Achse abgefragt. Der Winkel der Achse A3 wird in Grad auf der smartHMI ausgegeben.

```
JointPosition actPos = lbr.getCurrentJointPosition();
double a3 = actPos.get(2);
getLogger().info(Math.toDegrees(a3));
```

## 15.15.2 Kartesische Ist- oder Sollposition abfragen

### Beschreibung

Die kartesische Ist- oder Sollposition des Roboterflansches sowie jedes anderen darunterliegenden Frames kann abgefragt werden. D. h. jeder Frame eines Objekts, der über den attachTo-Befehl mit dem Roboterflansch verbunden ist, z. B. der TCP eines Werkzeugs oder der Frame eines gegriffenen Werkstücks.

Das Ergebnis der Abfrage, d. h. die kartesische Position, bezieht sich defaultmäßig auf das Welt-Koordinatensystem. Optional kann ein anderes Referenz-Koordinatensystem angegeben werden, relativ zu dem die kartesische Position abgefragt wird. Dies kann beispielsweise ein in den Applikationsdaten angelegter Frame sein oder eine vermessene Basis.

Das Ergebnis der Abfrage wird in einer Variablen vom Typ Frame gespeichert und enthält alle notwendigen Redundanzinformationen (Redundanzwinkel, Status und Turn). Von dieser Variablen können dann mithilfe der typspezifischen get-Methoden die Position (X, Y, Z) und Orientierung (A, B, C) des Frames abgefragt werden.

### Syntax

Kartesische Istposition abfragen:

```
Frame position = robot.getCurrentCartesianPosition(
frameOnFlange<, referenceFrame>);
```

Kartesische Sollposition abfragen:

```
Frame position = robot.getCommandedCartesianPosition(
frameOnFlange<, referenceFrame>);
```

#### Erläuterung der Syntax

Element	Beschreibung
position	Typ: Frame  Variable für den Rückgabewert. Der Rückgabewert enthält die abgefragte kartesische Position.
robot	Typ: Robot  Name des Roboters, von dem die kartesische Position abgefragt wird
frameOnFlange	Typ: ObjectFrame  Roboterflansch oder ein dem Flansch hierarchisch untergeordneter Frame, dessen kartesische Position abgefragt wird
referenceFrame	Typ: AbstractFrame  Referenz-Koordinatensystem, relativ zu dem die kartesische Position abgefragt wird. Wird kein Referenz-Koordinatensystem angegeben, bezieht sich die kartesische Position auf das Welt-Koordinatensystem.

#### Beispiele

Kartesische Istposition des Roboterflansches bezogen auf das Welt-Koordinatensystem:

```
Frame cmdPos = lbr.getCurrentCartesianPosition(lbr.getFlange());
```

Kartesische Istposition des TCP eines Werkzeug bezogen auf eine Basis:

```
tool.attachTo(lbr.getFlange());
...
Frame cmdPos = lbr.getCurrentCartesianPosition(tool.getFrame("/TCP"),
getApplicationData().getFrame("/Base"));
```

#### 15.15.3 Kartesische Soll-Ist-Differenz abfragen

##### Beschreibung

Die kartesische Soll-Ist-Differenz (= Differenz zwischen programmierte und gemessener Position) kann mit der Methode getPositionInformation(...) abgefragt werden.

Das Ergebnis der Abfrage wird in einer Variablen vom Typ PositionInformation gespeichert. Von dieser Variablen können die translatorischen und rotatorischen Soll-Ist-Differenzen getrennt voneinander abgefragt werden.

##### Syntax

Positionsinformationen abfragen:

```
PositionInformation info = robot.getPositionInformation(
frameOnFlange<, referenceFrame>);
```

Translatorische Soll-Ist-Differenz abfragen:

```
Vector translatoryDiff = info.getTranslationOffset();
```

Rotatorische Soll-Ist-Differenz abfragen:

```
Rotation rotatoryDiff = info.getRotationOffset();
```



Die im PositionInfomation-Objekt gespeicherte kartesische Ist- und Sollposition kann mit den bereits bekannten Methoden getCurrentCartesianPosition(...) und getCommandedCartesianPosition(...) ausgelesen werden.

## Erläuterung der Syntax

Element	Beschreibung
<i>info</i>	Typ: PositionInformation  Variable für den Rückgabewert. Der Rückgabewert enthält die abgefragten Positionsinformationen.
<i>robot</i>	Typ: Robot  Name des Roboters, von dem die Positionsinformationen abgefragt werden
<i>frameOnFlange</i>	Typ: ObjectFrame  Roboterflansch oder ein dem Flansch hierarchisch untergeordneter Frame, dessen Positionsinformationen abgefragt werden
<i>referenceFrame</i>	Typ: AbstractFrame  Referenz-Koordinatensystem, relativ zu dem die Positionsinformationen abgefragt werden. Wird kein Referenz-Koordinatensystem angegeben, beziehen sich die Positionsinformationen auf das Welt-Koordinatensystem.
<i>translatoryDiff</i>	Typ: Vector (com.kuka.roboticsAPI.geometricModel.math)  Translatorische Soll-Ist-Differenz in X-, Y-, Z-Richtung (Typ: double; Einheit: mm)  Mit den get-Methoden der Klasse Vector können die Offset-Werte für jeden Freiheitsgrad einzeln abgefragt werden.  (>>> 15.14.4 "Einzelwerte von einem Vektor abfragen" Seite 327)
<i>rotatoryDiff</i>	Typ: Rotation (com.kuka.roboticsAPI.geometricModel.math)  Soll-Ist-Differenz der Achswinkel A, B, C (Typ: double; Einheit: rad)  Mit den get-Methoden der Klasse Rotation - getAlphaRad(), getBetaRad, getGammaRad() - können die Offset-Werte für jeden Freiheitsgrad einzeln abgefragt werden.

## Beispiel

Auslesen der translatorischen Soll-Ist-Differenz in X-Richtung und der Soll-Ist-Differenz des Achswinkels C.

```
tool.attachTo(lbr.getFlange());
...
PositionInformation posInf =
lbr.getPositionInformation(tool.getFrame("/TCP"),
getApplicationData().getFrame("/Base"));

Vector transDiff = posInf.getTranslationOffset();
Rotation rotDiff = posInf.getRotationOffset();

double transOffsetInX = transDiff.getX();
double rotOffsetofC = rotDiff.getGammaRad();
```

## 15.16 HOME-Position

Die HOME-Position ist eine applikationsspezifische Position des Roboters. Sie kann bei der Initialisierung für eine Applikation neu gesetzt werden.

Die HOME-Position ist defaultmäßig mit folgenden Werten angelegt:

Achse	A1	A2	A3	A4	A5	A6	A7
Pos.	0°	0°	0°	0°	0°	0°	0°

### 15.16.1 HOME-Position ändern

#### Beschreibung

Die HOME-Position in einer Applikation kann mit `setHomePosition(...)` geändert werden. Die Methode gehört zur Klasse Robot.

Eine HOME-Position muss folgende Bedingungen erfüllen:

- Günstige Ausgangsposition für den Programmablauf
- Günstige Stillstandsposition. Beispielsweise darf der Roboter im Stillstand kein Hindernis darstellen.

Die neue HOME-Position kann als achsspezifische oder als kartesische Position (Frame) übergeben werden. Sie gilt nur in der Applikation, in der sie geändert wurde. Andere Applikationen verwenden weiterhin die HOME-Position mit den Default-Werten.

#### Syntax

```
robot.setHomePosition (home);
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>robot</code>	Typ: Robot  Name des Roboters, auf den sich die neue HOME-Position bezieht
<code>home</code>	Typ: JointPosition; Einheit: rad  1. Möglichkeit: Achspositionen des Roboters in der neuen HOME-Position übergeben.  Typ: AbstractFrame  2. Möglichkeit: Einen Frame als neue HOME-Position übergeben.  <b>Hinweis:</b> Der Frame muss alle Redundanzinformationen enthalten, damit die Achspositionen des Roboters in der HOME-Position eindeutig sind. Dies ist beispielsweise bei einem geteachten Frame der Fall.

#### Beispiele

Achsspezifische Position als HOME-Position übergeben:

```
private LBR lbr;
...
JointPosition newHome = new JointPosition(0.0, 0.0, 0.0,
Math.toRadians(90), 0.0, 0.0, 0.0);
lbr.setHomePosition(newHome);
```

Geteachten Frame als HOME-Position übergeben und mit `ptpHome()` anfahren:

```
private LBR lbr;
...
ObjectFrame newHome = getApplicationData().getFrame("/Homepos");
lbr.setHomePosition(newHome);
lbr.moveAsync(ptpHome());
```

### 15.17 Systemzustände abfragen

Es können verschiedene Systemzustände vom Roboter abgefragt und in der Applikation verarbeitet werden. Die Abfrage von Systemzuständen ist in erster

Linie bei Einsatz einer übergeordneten Steuerung erforderlich, damit diese auf Zustandsänderungen reagieren kann.

### 15.17.1 HOME-Position abfragen

**Beschreibung** Um die HOME-Position abzufragen, stehen folgende Methoden der Klasse Robot zur Verfügung:

- `getHomePosition()`  
Abfrage der HOME-Position, die aktuell für den Roboter definiert ist
- `isInHome()`  
Abfrage, ob sich der Roboter aktuell in der HOME-Position befindet

**Syntax** HOME-Position abfragen:

```
JointPosition homePos = robot.getHomePosition();
```

Prüfen, ob sich der Roboter in der HOME-Position befindet:

```
boolean result = robot.isInHome();
```

**Erläuterung der Syntax**

Element	Beschreibung
<code>homePos</code>	Typ: JointPosition  Variable für den Rückgabewert von <code>getHomePosition()</code> . Der Rückgabewert enthält die Achswinkel der abgefragten HOME-Position.
<code>robot</code>	Typ: Robot  Name des Roboters, von dem die HOME-Position abgefragt wird
<code>result</code>	Typ: boolean  Variable für den Rückgabewert von <code>isInHome()</code> . Der Rückgabewert ist <b>true</b> , wenn der Roboter in der HOME-Position steht.

**Beispiel**

Solange der Roboter noch nicht in der HOME-Position steht, soll ein bestimmter Anweisungsblock ausgeführt werden.

```
private LBR lbr;
...
while(! lbr.isInHome()) {
    //do something
}
```

### 15.17.2 Justagezustand abfragen

**Beschreibung** Um den Justagezustand abzufragen, steht die Methode `isMastered()` zur Verfügung. Die Methode gehört zur Klasse Robot.

**Syntax**

```
boolean result = robot.isMastered();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>robot</i>	Typ: Robot Name des Roboters, dessen Justagezustand abgefragt wird
<i>result</i>	Typ: boolean Variable für den Rückgabewert <ul style="list-style-type: none"> <li>■ <b>true:</b> Alle Achsen sind justiert.</li> <li>■ <b>false:</b> Eine oder mehrere Achsen sind dejustiert.</li> </ul>

**15.17.3 Fahrbereitschaft abfragen**
**Beschreibung**

Um abzufragen, ob der Roboter fahrbereit ist, steht die Methode `isReadyToMove()` zur Verfügung. Die Methode gehört zur Klasse `Robot`.

**Syntax**

```
boolean result = robot.isReadyToMove();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>robot</i>	Typ: Robot Name des Roboters, dessen Fahrbereitschaft abgefragt wird
<i>result</i>	Typ: boolean Variable für den Rückgabewert <ul style="list-style-type: none"> <li>■ <b>true:</b> Roboter ist fahrbereit.</li> <li>■ <b>false:</b> Ein Sicherheitshalt ist aktiv oder die Roboterantriebe sind im Fehlerzustand.</li> </ul>



Wenn der Rückgabewert **true** ist, bedeutet dies nicht zwangsläufig, dass die Bremsen geöffnet sind und der Roboter sich in aktiver Regelung befindet.

**15.17.3.1 Auf Änderung des Fahrbereitschaft-Signals reagieren**
**Beschreibung**

In der RoboticsAPI steht ein Benachrichtigungsdienst der Klasse `Controller` zur Verfügung, der eine Änderung des Fahrbereitschaft-Signals meldet. Um sich für den Dienst zu registrieren, übergibt man dem `Controller`-Attribut in der Roboter-Applikation ein `IControllerStateListener`-Objekt. Hierzu wird die Methode `addControllerListener(...)` verwendet.

Bei jeder Änderung des Fahrbereitschaft-Signals wird die Methode `onIsReadyToMoveChanged(...)` aufgerufen. Im Methodenrumpf von `onIsReadyToMoveChanged(...)` kann programmiert werden, wie auf die Änderung reagiert werden soll.

**Syntax**

```
kuka_Sunrise_Cabinet.addControllerListener(new
IControllerStateListener() {
    ...
    @Override
    public void onIsReadyToMoveChanged(Device device,
        boolean isReadyToMove) {
        // Reaktion auf Änderung
    }
})
```

...

} ) ;

## Erläuterung der Syntax

Element	Beschreibung
<code>kuka_Sunrise _Cabinet</code>	Typ: Controller  Controller-Attribut der Roboter-Applikation (= Name der Robotersteuerung in der Applikation)

### 15.17.4 Roboteraktivität abfragen

#### Beschreibung

Ein Roboter ist aktiv, wenn ein Bewegungskommando aktiv ist. Dies betrifft sowohl Bewegungsbefehle aus der Applikation als auch Handverfahrkommandos.

Um abzufragen, ob der Roboter aktiv ist, steht die Methode `hasActiveMotionCommand()` zur Verfügung. Die Methode gehört zur Klasse `Robot`.

#### Syntax

```
boolean result = robot.hasActiveMotionCommand();
```

## Erläuterung der Syntax

Element	Beschreibung
<code>robot</code>	Typ: Robot  Name des Roboters, dessen Aktivität abgefragt wird
<code>result</code>	Typ: boolean  Variable für den Rückgabewert <ul style="list-style-type: none"> <li>■ <b>true</b>: Ein Bewegungskommando ist aktiv.</li> <li>■ <b>false</b>: Es ist kein Bewegungskommando aktiv.</li> </ul>



Die Abfrage liefert keine Informationen darüber, ob sich der Roboter aktuell bewegt oder nicht.

Wenn der Rückgabewert **false** ist, bedeutet dies nicht zwangsläufig, dass der Roboter stillsteht.

Beispielsweise kann es vorkommen, dass die Roboteraktivität direkt nach einem synchronen Bewegungsbefehl mit Abbruchbedingung abgefragt wird. Tritt die Abbruchbedingung ein, liefert die Abfrage den Wert **false**, wenn der Roboter abgebremst wird und sich bewegt.

Wenn der Rückgabewert **true** ist, bedeutet dies nicht zwangsläufig, dass der Roboter sich bewegt. Beispielsweise liefert die Abfrage den Wert **true**, wenn ein positionsgergelter Roboter den Bewegungsbefehl `positionHold(...)` ausführt und stillsteht.

### 15.17.5 Sicherheitssignale abfragen und auswerten

#### Übersicht

In einer Roboter-Applikation kann der Zustand der folgenden Sicherheitssignale abgefragt und ausgewertet werden.

- Betriebsart
- NOT-HALT lokal
- NOT-HALT extern
- Bedienerschutz
- Stopp-Anforderung (Sicherheitshalt)

#### Voraussetzung

Um einen NOT-HALT oder Bedienerschutz auswerten zu können:

- In der Sicherheitskonfiguration ist in der zugehörigen Zeile der PSM-Tabelle die passende Kategorie ausgewählt.

- Kategorie **NOT-HALT lokal** für lokalen NOT-HALT
- Kategorie **NOT-HALT extern** für externen NOT-HALT
- Kategorie **Bedienerschutz** für Bedienerschutz
- Die konfigurierte Reaktion ist ein Sicherheitshalt (kein Ausgang).

### 15.17.5.1 Zustand der Sicherheitssignale abfragen

#### Beschreibung

Der aktuelle Zustand der verschiedenen Sicherheitssignale wird zunächst mit der Methode `getSafetyState()` abgefragt und in einem Objekt vom Typ `ISafetyState` zusammengefasst.

Von diesem Objekt kann dann mithilfe spezifischer Methoden der Zustand einzelner Sicherheitssignale abgefragt werden. Die möglichen Signalzustände sind Enums des jeweiligen Rückgabetyps.

#### Syntax

```
ISafetyState currentState = robot.getSafetyState();
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>currentState</code>	Typ: <code>ISafetyState</code>  Variable für den Rückgabewert. Der Rückgabewert enthält den Zustand der Sicherheitssignale zum Zeitpunkt der Abfrage.
<code>robot</code>	Typ: LBR  Name des Roboters, von dem der Zustand der Sicherheitssignale abgefragt wird

#### Übersicht

Um die Sicherheitssignale einzeln von `ISafetyState` abzufragen, stehen folgende Methoden zur Verfügung:

Methode	Beschreibung
<code>getOperationMode()</code>	Rückgabetyp: <code>OperationMode</code> (Paket <code>com.kuka.roboticsAPI.deviceModel</code> )  Abfrage der aktuell eingestellten Betriebsart  Enum-Werte des Rückgabetyps: <ul style="list-style-type: none"> <li>■ <b>T1, T2, AUT</b></li> <li>■ <b>KRF</b>: Betriebsart KRF</li> </ul>
<code>getEmergencyStopInt()</code>	Rückgabetyp: <code>EmergencyStop</code>  Abfrage, ob ein lokaler NOT-HALT aktiv ist  Enum-Werte des Rückgabetyps: <ul style="list-style-type: none"> <li>■ <b>ACTIVE</b>: Lokaler NOT-HALT ist aktiv.</li> <li>■ <b>INACTIVE</b>: Lokaler NOT-HALT ist nicht aktiv.</li> <li>■ <b>NOT_CONFIGURED</b>: Nicht relevant, da immer ein lokaler NOT-HALT konfiguriert ist.</li> </ul>
<code>getEmergencyStopEx()</code>	Rückgabetyp: <code>EmergencyStop</code>  Abfrage, ob ein externer NOT-HALT aktiv ist  Enum-Werte des Rückgabetyps: <ul style="list-style-type: none"> <li>■ <b>ACTIVE</b>: Externer NOT-HALT ist aktiv.</li> <li>■ <b>INACTIVE</b>: Externer NOT-HALT ist nicht aktiv.</li> <li>■ <b>NOT_CONFIGURED</b>: Es ist kein externer NOT-HALT konfiguriert.</li> </ul>

Methode	Beschreibung
getOperatorSafetyState()	Rückgabetypr: OperatorSafety Abfrage des Bedienerschutz-Signals Enum-Werte des Rückgabetyps: <ul style="list-style-type: none"> <li>■ <b>OPERATOR_SAFETY_OPEN</b>: Bedienerschutz ist verletzt (z. B. Schutztür geöffnet).</li> <li>■ <b>OPERATOR_SAFETY_CLOSED</b>: Bedienerschutz ist nicht verletzt.</li> <li>■ <b>NOT_CONFIGURED</b>: Es ist kein Bedienerschutz konfiguriert.</li> </ul>
getSafetyStopSignal()	Rückgabetypr: SafetyStopType Abfrage, ob ein Sicherheitshalt aktiv ist Enum-Werte des Rückgabetyps: <ul style="list-style-type: none"> <li>■ <b>NOSTOP</b>: Es ist kein Sicherheitshalt aktiv.</li> <li>■ <b>STOP0</b>: Ein Sicherheitshalt 0 oder ein Sicherheitshalt 1 ist aktiv.</li> <li>■ <b>STOP1</b>: Ein Sicherheitshalt 1 (bahntreu) ist aktiv.</li> <li>■ <b>STOP2</b>: Dieser Wert wird aktuell nicht zurückgegeben.</li> </ul>

**Beispiel**

Es wird abgefragt, ob ein Sicherheitshalt aktiv ist. Ist dies der Fall, wird anschließend der Bedienerschutz geprüft. Wenn dieser verletzt ist, wird eine Meldung auf der smartHMI ausgegeben.

```
ISafetyState safetyState = robot.getSafetyState();

SafetyStopType safetyStop = safetyState.getSafetyStopSignal();

if(safetyStop != SafetyStopType.NOSTOP) {
    OperatorSafety operatorSafety =
    safetyState.getOperatorSafetyState();
    if (operatorSafety == OperatorSafety.OPERATOR_SAFETY_OPEN)
        getLogger().warn("The safety gate is open!");
}
```

### 15.17.5.2 Auf Zustandsänderung von Sicherheitssignalen reagieren

**Beschreibung**

In der RoboticsAPI steht ein Benachrichtigungsdienst der Klasse Controller zur Verfügung, der Zustandsänderungen von Sicherheitssignalen meldet. Dieser Dienst ermöglicht es dem Benutzer, direkt auf die Änderung eines Signalzustands zu reagieren.

Um sich für den Dienst zu registrieren, übergibt man dem Controller-Attribut in der Roboter-Applikation ein ISunriseControllerStateListener-Objekt. Hierzu wird die Methode addControllerListener(...) verwendet.

Bei jeder Zustandsänderung eines Sicherheitssignals wird die Methode onSafetyStateChanged(...) aufgerufen. Im Methodenrumpf von onSafetyStateChanged(...) kann programmiert werden, wie auf die Zustandsänderung reagiert werden soll.

**Syntax**

```
kuka_Sunrise_Cabinet.addControllerListener(new
ISunriseControllerStateListener() {
    ...
    @Override
```

```
public void onSafetyStateChanged(Device device,
SunriseSafetyState safetyState) {
    // Reaktion auf Zustandsänderung
}
});
```

**Erläuterung der Syntax**

Element	Beschreibung
<code>kuka_Sunrise_Cabinet</code>	Typ: Controller Controller-Attribut der Roboter-Applikation (= Name der Robotersteuerung in der Applikation)

**Beispiel**

Wenn sich der Zustand eines Sicherheitssignals ändert, wird über die Methode `onSafetyStateChanged(...)` der Bedienerschutz geprüft. Wenn dieser verletzt ist, wird eine Meldung auf der smartHMI ausgegeben.

```
kuka_Sunrise_Cabinet.addControllerListener(new
ISunriseControllerStateListener() {
    ...
    @Override
    public void onSafetyStateChanged(Device device,
        SunriseSafetyState safetyState) {
        OperatorSafety operatorSafety =
        safetyState.getOperatorSafetyState();
        if (operatorSafety == OperatorSafety.OPERATOR_SAFETY_OPEN)
            getLogger().warn("The saftey gate is open!");
    }
});
```

## 15.18 Programmablaufart ändern und abfragen

**Beschreibung**

Über die Methoden `setExecutionMode(...)` und `getExecutionMode()` des SunriseExecutionService kann die Programmablaufart geändert und abgefragt werden. Der SunriseExecutionService selbst wird vom Controller abgefragt.

**Vorbereitung**

1. Variable vom Typ SunriseExecutionService anlegen.
2. SunriseExecutionService mithilfe der Methode `getExecutionService()` vom Controller abfragen und in der Variablen speichern.

**Syntax**

Programmablaufart ändern:

```
service.setExecutionMode(ExecutionMode.newMode);
```

Aktuelle Programmablaufart abfragen:

```
currentMode = service.getExecutionMode();
```

**Erläuterung der Syntax**

Element	Beschreibung
<code>service</code>	Typ: SunriseExecutionService Variable für den Rückgabewert (enthält den vom Controller abgefragten SunriseExecutionService)

Element	Beschreibung
<i>newMode</i>	<p>Typ: Enum vom Typ ExecutionMode</p> <p>Neue Programmablaufart</p> <ul style="list-style-type: none"> <li>■ <b>ExecutionMode.Step:</b> Step-Betrieb (Programmablauf mit Stopp nach jedem Bewegungsbefehl)</li> <li>■ <b>ExecutionMode.Continuous:</b> Standard-Betrieb (kontinuierlicher Programmablauf ohne Stopps)</li> </ul>
<i>currentMode</i>	<p>Typ: ExecutionMode</p> <p>Variable für den Rückgabewert (enthält die vom SunriseExecutionService abgefragte Programmablaufart)</p>

**Beispiel**

Der SunriseExecutionService wird vom Controller abgefragt und in der Variable "serv" gespeichert.

```
private SunriseExecutionService serv;
...
public void initialize() {
    controller = getController("Controller_LBR5");
    robot = (LBR) getRobot(controller, "LBR5_7kg");
    serv = (SunriseExecutionService) controller.getExecutionService();
    ...
}
```

Erst wird in den Step-Betrieb und dann wieder in den Standard-Betrieb geschalten.

```
public void run() {
    ...
    serv.setExecutionMode(ExecutionMode.Step);
    ...
    serv.setExecutionMode(ExecutionMode.Continuous);
    ...
}
```

Die aktuelle Programmablaufart wird abgefragt.

```
public void run() {
    ...
    ExecutionMode currentMode;
    currentMode = serv.getExecutionMode();
    ...
}
```

## 15.19 Override ändern und abfragen

Die Schnittstelle IApplicationOverrideControl stellt Methoden zur Verfügung, mit denen der aktuelle Override in der Applikation abgefragt oder geändert werden kann. Dazu muss im ersten Schritt mithilfe der Methode getApplicationControl() auf die Schnittstelle IApplicationControl zugegriffen werden.

Folgende Override-Typen werden unterschieden:

- Manueller Override: Vom Bediener über das smartPAD manuell einstellbarer Override  
(>>> 6.14.3 "Manuellen Override einstellen" Seite 93)
- Applikations-Override: Von der Applikation gesetzter programmierte Override
- Effektiver Override: Produkt aus manuellem und Applikations-Override  
Effektiver Override = Manueller Override · Applikations-Override

**Übersicht**

Methoden zur Abfrage des aktuellen Overrides:

Methode	Beschreibung
getApplicationOverride()	Rückgabetyp: double Abfrage des Applikations-Overrides
getManualOverride()	Rückgabetyp: double Abfrage des manuellen Overrides
getEffectiveOverride()	Rückgabetyp: double Abfrage des effektiven Overrides

Methoden zum Ändern des Overrides:

Methode	Beschreibung
setApplicationOverride(...)	Setzt den Applikations-Override auf den angegebenen Wert (Typ: double) <b>■ 0 ... 1</b>
clipApplicationOverride(...)	Reduziert den Applikations-Override auf den angegebenen Wert (Typ: double) <b>■ 0 ... 1</b> Wird ein Wert angegeben, der größer ist als der aktuell für den Applikations-Override programmierte Wert, wird die Anweisung clipApplicationOverride(...) ignoriert.
clipManualOverride(...)	Reduziert den manuellen Override auf den angegebenen Wert (Typ: double) <b>■ 0 ... 1</b> Wird ein Wert angegeben, der größer ist als der aktuell eingesetzte manuelle Override, wird die Anweisung clipManualOverride(...) ignoriert.

**Beispiel**

```
getApplicationControl().setApplicationOverride(0.5);
...
double actualOverride =
getApplicationControl().getEffectiveOverride();
```

**15.19.1 Auf Override-Änderung reagieren****Beschreibung**

Eine Applikation kann sich informieren lassen, wenn sich ein Override ändert. Hierfür muss ein Listener vom Typ IApplicationOverrideListener definiert und registriert werden.

Bei jeder Änderung eines Overrides wird die Methode overrideChanged(...) aufgerufen. Im Methodenkörper von overrideChanged(...) kann programmiert werden, wie auf die Änderung reagiert werden soll.

**Syntax**

Listener definieren:

```
IApplicationOverrideListener overrideListener =
new IApplicationOverrideListener() {
    @Override
    public void overrideChanged(double effectiveOverride,
        double manualOverride, double applicationOverride) {
        // Reaktion auf Override-Änderung
    }
};
```

```

} ;

Listener registrieren:
getApplicationControl() .
addOverrideListener (overrideListener) ;

Listener abmelden:
getApplicationControl() .
removeOverrideListener (overrideListener) ;

```

### Erläuterung der Syntax

Element	Beschreibung
<i>override Listener</i>	Typ: IApplicationOverrideListener Name des Listeners

## 15.20 Bedingungen

Häufig sollen in Applikationen Werte überwacht und bei Über- oder Unterschreiten definierbarer Grenzen bestimmte Reaktionen ausgelöst werden. Mögliche Quellen für diese Werte sind beispielsweise die Sensoren des Roboters oder konfigurierte Eingänge. Auch der Fortschritt einer Bewegung kann überwacht werden. Mögliche Reaktionen sind der Abbruch einer laufenden Bewegung oder die Ausführung einer Behandlungsroutine.

### 15.20.1 Bedingungen in der RoboticsAPI

**Beschreibung** Eine Bedingung kann 2 Zustände besitzen: Sie ist erfüllt (Zustand TRUE) oder nicht erfüllt (Zustand FALSE). Zur Definition einer Bedingung wird ein Ausdruck formuliert. In diesem Ausdruck werden z. B. vom System bereitgestellte Messungen mit einem zulässigen Grenzwert verglichen. Das Ergebnis der Auswertung des Ausdrucks definiert den Zustand der Bedingung.

Da unterschiedliche Systemdaten zur Formulierung von Bedingungen verwendet werden können, gibt es unterschiedliche Arten von Bedingungen. Je-der dieser Bedingungstypen wird in der RoboticsAPI als eigene Klasse zur Verfügung gestellt. Sie gehören zum Paket com.kuka.roboticsAPI.condition-Model und implementieren die Schnittstelle ICondition.

**Übersicht** Folgende Bedingungstypen stehen zur Verfügung:

Datentyp	Beschreibung
JointTorqueCondition	Die Achsmomenten-Bedingung ist erfüllt, wenn das in einer Achse gemessene Moment außerhalb eines definierten Wertebereichs liegt. (>>> 15.20.3 "Achsmomenten-Bedingung" Seite 343)
ForceCondition	Die Kraftbedingung ist erfüllt, wenn die kartesische Kraft, die auf einen Frame unterhalb des Roboterflansches wirkt (z. B. auf den TCP), einen definierten Betrag überschreitet. (>>> 15.20.4 "Kraftbedingung" Seite 344)
ForceComponentCondition	Die Kraft-Komponenten-Bedingung ist erfüllt, wenn die kartesische Kraft, die entlang einer Achse eines Frames unterhalb des Roboterflansches wirkt (z. B. entlang einer Achse des TCP), außerhalb eines definierten Bereichs liegt. (>>> 15.20.5 "Kraft-Komponenten-Bedingung" Seite 349)

Datentyp	Beschreibung
CartesianTorqueCondition	Die Bedingung für das kartesische Moment ist erfüllt, wenn das kartesische Moment, das um die Achse eines Frames unterhalb des Roboterflansches wirkt (z. B. um die Achse des TCP), einen definierten Betrag überschreitet. (>>> 15.20.6 "Bedingung für kartesisches Moment" Seite 351)
TorqueComponentCondition	Die Moment-Komponenten-Bedingung ist erfüllt, wenn das kartesische Moment, das um eine Achse eines Frames unterhalb des Roboterflansches wirkt (z. B. um eine Achse des TCP), außerhalb eines definierten Bereichs liegt. (>>> 15.20.7 "Moment-Komponenten-Bedingung" Seite 355)
MotionPathCondition	Die bahnbezogene Bedingung ist erfüllt, wenn eine definierte Distanz auf der geplanten Bahn, ausgehend vom Start- oder Zielpunkt der Bewegung, erreicht wurde. Zusätzlich kann eine zeitliche Verschiebung definiert werden, die erfüllt sein muss. (>>> 15.20.8 "Bahnbezogene Bedingung" Seite 356)
BooleanIOCondition	Die Bedingung für boolesche Signale ist erfüllt, wenn ein boolescher digitaler Eingang einen bestimmten Zustand besitzt. (>>> 15.20.9 "Bedingung für boolesche Signale" Seite 359)
IORangeCondition	Die Bedingung für den Wertebereich eines Signals ist erfüllt, wenn der Wert eines analogen oder digitalen Eingangs innerhalb eines definierten Bereichs liegt. (>>> 15.20.10 "Bedingung für Wertebereich eines Signals" Seite 359)

- Anwendungsge-  
biete**
- Abbruch von Bewegungen  
Eine Bewegung wird abgebrochen, sobald ein bestimmtes Ereignis eintritt. Das Ereignis tritt ein, wenn die Bedingung bereits vor Bewegungsbeginn den Zustand TRUE hat oder während der Bewegung in den Zustand TRUE wechselt.  
(>>> 15.21 "Abbruchbedingungen für Bewegungsbefehle" Seite 360)
  - Bahnbezogene Schaltaktionen (Trigger)  
Eine Aktion wird ausgelöst, sobald ein bestimmtes Ereignis eintritt. Das Ereignis tritt ein, wenn die Bedingung bereits vor Bewegungsbeginn den Zustand TRUE hat oder während der Bewegung in den Zustand TRUE wechselt.  
(>>> 15.22 "Bahnbezogene Schaltaktionen (Trigger)" Seite 364)
  - Überwachen von Prozessen (Monitoring)  
Der Zustand einer Bedingung wird mithilfe eines Listeners zyklisch überprüft. Ändert sich der Zustand der Bedingung, kann darauf reagiert werden.  
(>>> 15.23 "Überwachen von Prozessen (Monitoring)" Seite 368)
  - Blockierendes Warten auf Bedingung  
Eine Applikation wird so lange angehalten bis eine bestimmte Bedingung erfüllt ist oder eine bestimmte Wartezeit abgelaufen ist.  
(>>> 15.24 "Blockierendes Warten auf Bedingung" Seite 372)

## 15.20.2 Komplexe Bedingungen

Bedingungen können logisch miteinander verknüpft werden, so dass die Festlegung komplexer Bedingungen möglich ist. Die dafür benötigten logischen Operatoren stehen als ICondition-Methoden zur Verfügung. Das aufrufende

ICondition-Objekt wird dabei mit einer oder mehreren Bedingungen verknüpft, indem diese als Parameter übergeben werden.

Die Verknüpfungsbefehle können mehrmals hintereinander aufgerufen und so Klammerung und Verschachtelung von Operationen realisiert werden. Die Auswertung ist dabei abhängig von der Reihenfolge des Aufrufs.

## Operatoren

Operator	Beschreibung/Syntax
NOT	Invertierung des aufrufenden ICondition-Objekts <code>ICondition invert();</code>
XOR	Entweder-Oder-Verknüpfung des aufrufenden ICondition-Objekts mit einer weiteren Bedingung <code>ICondition xor(ICondition other);</code> <i>other</i> : weitere Bedingung
AND	Und-Verknüpfung des aufrufenden ICondition-Objekts mit einer oder mehreren zusätzlichen Bedingungen <code>ICondition and(ICondition other1, ICondition other2, ...);</code> <i>other1, other2, ...</i> : weitere Bedingungen
OR	Oder-Verknüpfung des aufrufenden ICondition-Objekts mit einer oder mehreren zusätzlichen Bedingungen <code>ICondition or(ICondition other1, ICondition other2, ...);</code> <i>other1, other2, ...</i> : weitere Bedingungen

## Beispiel

```
JointTorqueCondition condA = ...;
JointTorqueCondition condB = ...;
JointTorqueCondition condC = ...;
JointTorqueCondition condD = ...;

ICondition combi1, combi2, combi3, combi4;

// NOT A
combi1 = condA.invert();

// A AND B AND C
combi2 = condA.and(condB, condC);

// (A OR B) AND C
combi3 = condA.or(condB).and(condC);

// (A OR B) AND (C OR D)
combi4 = condA.or(condB).and(condC.or(condD));
```

### 15.20.3 Achsmomenten-Bedingung

#### Beschreibung

Mit der Achsmomenten-Bedingung kann geprüft werden, ob das in einer Achse ermittelte externe Moment außerhalb eines definierten Wertebereichs liegt.

(>>> 15.13 "Achsmomente abfragen" Seite 322)



Bei der Programmierung müssen die Lastdaten korrekt angegeben werden. Nur dann ist die Achsmomenten-Bedingung sinnvoll anwendbar.

**Konstruktorsyntax**

```
JointTorqueCondition(JointEnum joint, double minTorque,
double maxTorque)
```

Element	Beschreibung
<i>joint</i>	Achse, deren Momentenwert geprüft wird ■ <b>JointEnum.J1 ... JointEnum.J12:</b> Achse A1 ... A12
<i>minTorque</i>	Unterer Grenzwert für das Achsmoment (Einheit: Nm) Die Bedingung ist erfüllt, wenn das Achsmoment kleiner oder gleich <i>minTorque</i> ist.
<i>maxTorque</i>	Oberer Grenzwert für das Achsmoment (Einheit: Nm) Die Bedingung ist erfüllt, wenn das Achsmoment größer oder gleich <i>maxTorque</i> ist.

Bei der Festlegung des unteren und oberen Grenzwerts für das Achsmoment muss gelten:  $minTorque \leq maxTorque$ .

**Beispiel**

```
JointTorqueCondition torqueCondJ3 =
new JointTorqueCondition(JointEnum.J3, -2.5, 4.0);
```

Die Bedingung ist erfüllt, wenn in Achse A3 ein Momentenwert  $\leq -2.5$  Nm oder  $\geq 4.0$  Nm gemessen wird.

#### 15.20.4 Kraftbedingung

**Beschreibung**

Mit der Kraftbedingung kann geprüft werden, ob eine auf einen Frame unterhalb des Roboterflansches wirkende kartesische Kraft einen definierten Grenzwert überschreitet.

Beispielsweise kann auf die Kraft reagiert werden, die entsteht, wenn der Roboter mit einem am Flansch montierten Werkzeug auf eine Oberfläche drückt. Für die Kraftbedingung werden die Projektionen des Kraftvektors betrachtet, der auf einen Frame unterhalb des Flansches wirkt. Die Position dieses Frames wird durch den Kraft-Angriffspunkt (hier die Werkzeugspitze) definiert. Die Orientierung des Frames soll der Orientierung der Oberfläche entsprechen.

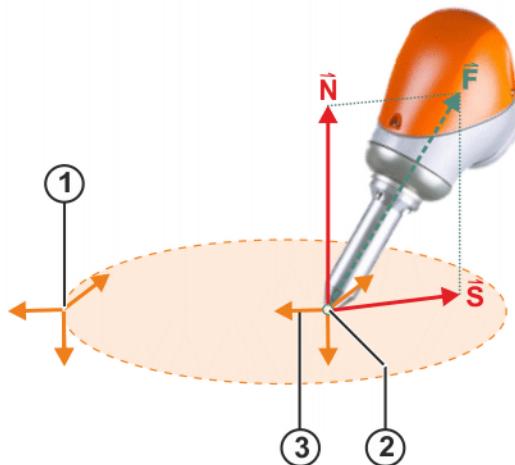


Abb. 15-16: Kraftvektoren

- 1 Frame, der die Orientierung des Bezugs-Frames vorgibt (hier: Orientierung der Oberfläche)

- 2 Kraft-Angriffspunkt, hier die Spitze des Werkzeugs
- 3 Bezugs-Frame unterhalb des Flansches, auf den der Kraftvektor projiziert wird. Die Position des Frames entspricht dem Kraft-Angriffspunkt. Die Orientierung entspricht der Orientierung der Oberfläche.

Folgende Kraftvektoren sind relevant:

- **Normalkraft N:**  
Projektion der aufgebrachten Kraft auf die Oberflächennormale (= Vektor, der senkrecht auf der Oberfläche steht). Daraus resultiert der Teil der aufgebrachten Kraft, die senkrecht auf die Oberfläche wirkt. Über die Normalkraft wird beispielsweise Druck ausgeübt, um ein Bauteil einzupassen.
- **Scherkraft S:**  
Projektion der aufgebrachten Kraft auf die Oberfläche. Daraus resultiert der Teil der aufgebrachten Kraft, die parallel zur Oberfläche wirkt. Die Scherkraft entsteht z. B. durch Reibung.



Bei der Programmierung müssen die Lastdaten korrekt angegeben werden. Nur dann ist die Kraftbedingung sinnvoll anwendbar.



In der Nähe singulärer Stellungen kann die Kraftschätzung keine sinnvollen Werte liefern. Es wird empfohlen, die Kraftbedingung für solche Achskonfigurationen nicht einzusetzen. Alternativ kann die Achsmomenten-Bedingung genutzt werden oder die Achsstellung mittels der Redundanz so angepasst werden, dass keine singuläre Pose mehr vorliegt.

## Methoden

Kraftbedingungen sind vom Datentyp ForceCondition. ForceCondition enthält folgende statischen Methoden zur Programmierung von Bedingungen:

- `createSpatialForceCondition(...)`: Bedingung für kartesische Kraft aus allen Richtungen
- `createNormalForceCondition(...)`: Bedingung für Normalkraft
- `createShearForceCondition(...)`: Bedingung für Scherkraft

Zur Formulierung der Bedingung wird ein Frame unterhalb des Flansch-Koordinatensystems (z. B. die Spitze eines Werkzeugs) als Bezugssystem definiert. Die Kräfte, die bezogen auf diesen Frame wirken, werden ermittelt. Die Orientierung des Bezugssystems kann optional über einen Orientierungs-Frame definiert werden. Damit kann beispielsweise die Lage der Oberfläche, auf der die Kraft aufgebracht wird, definiert werden.

Durch die Angabe eines Grenzwertes wird definiert, ab welchem Betrag der Kraft die Bedingung erfüllt ist.

Die kartesische Kraft wird aus den Werten der Gelenkmomenten-Sensoren berechnet. Abhängig von der Achskonfiguration variiert die Zuverlässigkeit der ermittelten Kraftwerte. Soll die Güte der Kraftberechnung ebenfalls berücksichtigt werden, kann ein Wert für die maximal erlaubte Ungenauigkeit angegeben werden. Überschreitet die vom System ermittelte Ungenauigkeit diesen Wert, ist die Kraftbedingung ebenfalls erfüllt.

### 15.20.4.1 Bedingung für kartesische Kraft aus allen Richtungen

#### Beschreibung

Über die statische Methode `createSpatialForceCondition(...)` kann eine Bedingung definiert werden, die unabhängig davon gilt, aus welcher Richtung die kartesische Kraft auf einen Frame unterhalb des Flansches wirkt.

#### Syntax

```
ForceCondition.createSpatialForceCondition(
AbstractFrame measureFrame<, AbstractFrame orientationFrame>,
double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measureFrame</i>	Frame unterhalb des Roboterflansches, bezogen auf den die wirkende Kraft ermittelt wird. Über diesen Parameter wird die Position des Kraft-Angriffspunkts definiert.
<i>orientationFrame</i>	Optional. Über diesen Parameter wird die Orientierung des Bezugssystems festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.
<i>threshold</i>	Betrag der Kraft, die maximal auf das Bezugssystem wirken darf (Einheit: N). <b>■ ≥ 0.0</b> Die Bedingung ist erfüllt, wenn der Betrag der aus beliebiger Richtung auf das Bezugssystem wirkenden Kraft den hier angegebenen Wert überschreitet.
<i>tolerance</i>	Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: N). <b>■ &gt; 0.0</b> Default: 10.0 Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Kraftberechnung größer oder gleich dem hier angegebenen Wert ist. Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.

**Beispiel**

Sobald der Betrag der Kraft, die aus einer beliebigen Richtung auf den TCP eines Werkzeugs wirkt, 30 N übersteigt, ist die Bedingung erfüllt.

```
Tool gripper = ...;
gripper.attachTo(lbr.getFlange());
```

```
ForceCondition spatialForce_tcp =
ForceCondition.createSpatialForceCondition(
    gripper.getFrame("/TCP"),
    30.0
```

```
) ;
```

**15.20.4.2 Bedingung für Normalkraft****Beschreibung**

Über die statische Methode `createNormalForceCondition(...)` kann eine Bedingung für die Normalkraft definiert werden. Betrachtet wird der Anteil der aufgebrachten Kraft, der entlang einer definierbaren Achse eines Frames unterhalb des Flansches wirkt (z. B. entlang einer Achse des TCP). Diese Achse wird in der Regel so festgelegt, dass sie senkrecht auf der Oberfläche steht, auf der die Kraft aufgebracht wird (Oberflächennormale).

**Syntax**

```
ForceCondition.createNormalForceCondition(AbstractFrame
measureFrame<, AbstractFrame orientationFrame>, CoordinateAxis
direction, double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measureFrame</i>	Frame unterhalb des Roboterflansches, bezogen auf den die wirkende Kraft ermittelt wird. Über diesen Parameter wird die Position des Kraft-Angriffs-punkts definiert.
<i>orientationFrame</i>	Optional. Über diesen Parameter wird die Orientierung des Bezugssystems festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.
<i>direction</i>	Koordinatenachse des Bezugssystems. Der Kraftanteil, der entlang der hier angegebenen Achse wirkt, wird mit der Bedingung geprüft. <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>
<i>threshold</i>	Betrag der Kraft, die maximal entlang der Achse des Bezugssystems wirken darf (Einheit: N). <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Die Bedingung ist erfüllt, wenn der Betrag der Kraft den hier angegebenen Wert überschreitet.
<i>tolerance</i>	Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: N). <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> Default: 10.0 Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Kraftberechnung größer oder gleich dem hier angegebenen Wert ist. Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.

## Beispiel

Ein am Flansch montierter Greifer drückt auf eine Tischplatte. Es soll auf den Teil der am TCP des Greifers angreifenden Kraft reagiert werden, der senkrecht auf die Tischplatte wirkt. Das Bezugssystem wird daher so festgelegt, dass seine Z-Achse entlang der Oberflächennormalen der Tischplatte verläuft.

Sobald die Normalkraft einen Betrag von 45 N übersteigt, ist die Bedingung erfüllt. Außerdem soll die Bedingung als erfüllt gelten, wenn die Ungenauigkeit der ermittelten Daten einen Wert von 8 überschreitet.

```
Tool gripper = ...;
gripper.attachTo(lbr.getFlange());

SpatialObject table = ...;

ForceCondition normalForce_z =
ForceCondition.createNormalForceCondition(
    gripper.getFrame("/TCP"),
    table.getFrame("/Tabletop"),
    CoordinateAxis.Z,
    45.0,
    8.0
);
```

### 15.20.4.3 Bedingung für Scherkraft

**Beschreibung** Über die statische Methode `createShearForceCondition(...)` kann eine Bedingung für die Scherkraft definiert werden. Betrachtet wird der Anteil der aufgebrachten Kraft, der parallel zu einer Ebene wirkt. Die Lage der Ebene wird festgelegt, indem man die Achse angibt, die senkrecht auf ihr steht.

**Syntax**

```
ForceCondition.createShearForceCondition(AbstractFrame
measureFrame<, AbstractFrame orientationFrame>, CoordinateAxis
normalDirection, double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measure Frame</i>	Frame unterhalb des Roboterflansches, bezogen auf den die wirkende Kraft ermittelt wird. Über diesen Parameter wird die Position des Kraft-Angriffspunkts definiert.
<i>orientation Frame</i>	Optional. Über diesen Parameter wird die Orientierung des Bezugssystems festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.
<i>normal Direction</i>	Koordinatenachse des Bezugssystems. Die hier angegebene Achse definiert die Oberflächennormale einer Ebene. Der Kraftanteil, der parallel zu dieser Ebene wirkt, wird geprüft. <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>
<i>threshold</i>	Betrag der Kraft, die maximal parallel zu der durch ihre Oberflächennormale definierte Ebene des Bezugssystems wirken darf (Einheit: N). <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Die Bedingung ist erfüllt, wenn der Betrag der Kraft den hier angegebenen Wert überschreitet.
<i>tolerance</i>	Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: N). <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> Default: 10.0 Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Kraftberechnung größer oder gleich dem hier angegebenen Wert ist. Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.

**Beispiel**

Ein am Flansch montierter Greifer drückt auf eine Tischplatte. Es soll die Kraft am TCP des Greifers mit der Orientierung der Tischplatte ermittelt werden. Dabei wird die Scherkraft betrachtet, die parallel zur XY-Ebene des durch den TCP und die Lage des Tisches definierten Messpunktes wirkt.

Um die XY-Ebene zu definieren, muss die Achse, die senkrecht zu dieser Ebene steht als Parameter angegeben werden. Dies ist die Z-Achse.

Sobald die Scherkraft einen Betrag von 25 N übersteigt, ist die Bedingung erfüllt. Außerdem soll die Bedingung als erfüllt gelten, wenn die Ungenauigkeit der ermittelten Daten einen Wert von 5 überschreitet.

```

Tool gripper = ...;
gripper.attachTo(lbr.getFlange());

SpatialObject table = ...;

ForceCondition shearForce_xyPlane =
ForceCondition.createShearForceCondition(
    gripper.getFrame("/TCP"),
    table.getFrame("/Tabletop"),
    CoordinateAxis.Z,
    25.0,
    5.0

);

```

### 15.20.5 Kraft-Komponenten-Bedingung

#### Beschreibung

Mit der Kraft-Komponenten-Bedingung kann geprüft werden, ob die kartesische Kraft, die in X-, Y- oder Z-Richtung auf einen Frame unterhalb des Roboterflansches wirkt (z. B. auf den TCP), außerhalb eines definierten Bereichs liegt.



Bei der Programmierung müssen die Lastdaten korrekt angegeben werden. Nur dann ist die Kraft-Komponenten-Bedingung sinnvoll anwendbar.



In der Nähe singulärer Stellungen kann die Kraftschätzung keine sinnvollen Werte liefern. Es wird empfohlen, die Kraft-Komponenten-Bedingung für solche Achskonfigurationen nicht einzusetzen. Alternativ kann die Achsmomenten-Bedingung genutzt werden oder die Achsstellung mittels der Redundanz so angepasst werden, dass keine Singularität mehr vorliegt.

Die Kraft-Komponenten-Bedingung gehört zur Klasse ForceComponentCondition. Für die Kraft-Komponenten-Bedingung wird ein Frame unterhalb des Flansch-Koordinatensystems als Bezugssystem definiert. An diesem Frame, z. B. an der Spitze eines Werkzeugs, wird die Kraft ermittelt. Die Orientierung des Bezugssystems kann optional über einen Orientierungs-Frame definiert werden.

Die Richtung, aus der die Kraft überprüft wird, wird mit einer der Koordinatenachsen des Bezugssystems festgelegt. Die Kraft-Komponenten-Bedingung ist erfüllt, wenn die kartesische Kraft entlang der festgelegten Koordinatenachse des Bezugssystems außerhalb eines definierbaren Wertebereichs liegt.

Die kartesische Kraft wird aus den Werten der Gelenkmomenten-Sensoren berechnet. Abhängig von der Achskonfiguration variiert die Zuverlässigkeit der ermittelten Kraftwerte. Soll die Güte der Kraftberechnung ebenfalls berücksichtigt werden, kann ein Wert für die maximal erlaubte Ungenauigkeit angegeben werden. Überschreitet die vom System ermittelte Ungenauigkeit diesen Wert, ist die Kraft-Komponenten-Bedingung ebenfalls erfüllt.

#### Konstruktor-Syntax

Die Klasse ForceComponentCondition besitzt mehrere Konstruktoren, die sich in der Anzahl der Eingangsparameter unterscheiden:

```

ForceComponentCondition(AbstractFrame measureFrame
<, AbstractFrame orientationFrame>, CoordinateAxis coordinateAxis,
double min, double max<, double tolerance>)

```

Element	Beschreibung
<i>measureFrame</i>	Frame unterhalb des Roboterflansches, bezogen auf den die wirkende Kraft ermittelt wird. Über diesen Parameter wird die Position des Kraft-Angriffspunkts definiert.
<i>orientationFrame</i>	Optional. Über diesen Parameter wird die Orientierung des Bezugssystems festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.
<i>coordinateAxis</i>	Koordinatenachse des Frames, bezogen auf den die wirkende Kraft ermittelt wird. Legt die Richtung fest, aus der die wirkende Kraft geprüft wird. <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>
<i>min</i>	Untere Grenze des Wertebereichs für die entlang der Koordinatenachse des Bezugssystems wirkende Kraft (Einheit: N). Die Kraft-Komponenten-Bedingung ist erfüllt, wenn die Kraft den hier angegebenen Wert unterschreitet.
<i>max</i>	Obere Grenze des Wertebereichs für die entlang der Koordinatenachse des Bezugssystems wirkende Kraft (Einheit: N). Die Kraft-Komponenten-Bedingung ist erfüllt, wenn die Kraft den hier angegebenen Wert überschreitet. <b>Hinweis:</b> Der obere Grenzwert muss größer sein als untere Grenzwert: <i>max &gt; min</i> .
<i>tolerance</i>	Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte. <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b> Default: 10.0</li> </ul> Die Kraft-Komponenten-Bedingung ist erfüllt, wenn die Ungenauigkeit der Kraftberechnung größer oder gleich dem hier angegebenen Wert ist. Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.

**Beispiel**

Ein Fügeprozess wird idealerweise mit einer Kraft zwischen 20 N und 25 N ausgeführt. Es soll eine Kraft-Komponenten-Bedingung definiert werden, die erfüllt ist, wenn die Kraft, die am freien Ende eines geöffneten Werkstücks in Z-Richtung wirkt, zwischen 20 N und 25 N beträgt.

Dazu wird zunächst eine Kraft-Komponenten-Bedingung definiert, die in diesem Wertebereich den Status FALSE besitzt. Durch Invertierung wird anschließend das gewünschte Ergebnis realisiert.

```
Workpiece peg = ...;
peg.attachTo(...);

ForceComponentCondition assemblyForce_inverted = new
ForceComponentCondition(
    peg.getFrame("/Assembly"),
    CoordinateAxis.Z,
    20.0,
```

```

25.0
);

ForceComponentCondition assemblyForce = (ForceComponentCondition)
assemblyForce_inverted.invert();

```

### 15.20.6 Bedingung für kartesisches Moment

#### Beschreibung

Mit der Bedingung kann geprüft werden, ob ein auf einen Frame unterhalb des Roboterflansches wirkendes kartesisches Moment einen definierten Grenzwert überschreitet. Der Angriffspunkt des Moments wird durch einen Frame unterhalb des Flansch-Koordinatensystems angegeben.

Ein Anwendungsfall für diese Bedingung ist die Überwachung von Momenten, die bei einem Schraubprozess auftreten.

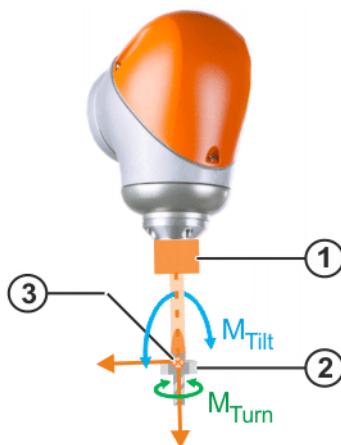


Abb. 15-17: Momentenvektoren bei Schraubprozess

- 1 Schraubwerkzeug
- 2 Schraube
- 3 Bezugs-Frame, hier die Spitze des Schraubwerkzeugs

Mit der Bedingung für das kartesische Moment können unterschiedliche Projektionen des Momentenvektors, die auf die Achsen des Bezugs-Frames wirken, geprüft werden:

- Drehmoment  $M_{Turn}$   
Das Drehmoment, das um eine Achse wirkt, entsteht durch Projektion des Momentenvektors auf diese Achse.
- Kippmoment  $M_{Tilt}$   
Das Kippmoment entsteht durch Projektion des Momentenvektors auf eine Ebene.

Das Drehmoment wird bei einem Schraubprozess um die Längsachse des Schraubwerkzeugs aufgebracht, um die Schraube einzudrehen. Wird die Bedingung für das Drehmoment verwendet, kann gewährleistet werden, dass zulässige Maximalwerte beim Schrauben nicht überschritten werden.

Das Kippmoment entsteht bei einem Schraubprozess durch unerwünschtes Kippen des Schraubwerkzeugs um die Längsachse nach vorne oder zur Seite. Wird die Bedingung für das Kippmoment konfiguriert, kann geprüft werden, ob das Kippmoment innerhalb eines akzeptablen Wertebereichs liegt.



Bei der Programmierung müssen die Lastdaten korrekt angegeben werden. Nur dann ist die Bedingung für das kartesische Moment sinnvoll anwendbar.



In der Nähe singulärer Stellungen kann die Kraftschätzung keine sinnvollen Werte liefern. Es wird empfohlen, die Bedingung für das kartesische Moment für solche Achskonfigurationen nicht einzusetzen. Alternativ kann die Achsmomenten-Bedingung genutzt werden oder die Achsstellung mittels der Redundanz so angepasst werden, dass keine Singularität mehr vorliegt.

## Methoden

Bedingungen für das kartesische Moment sind vom Datentyp `CartesianTorqueCondition`. `CartesianTorqueCondition` enthält folgende statischen Methoden zur Programmierung von Bedingungen:

- `createSpatialTorqueCondition(...)`: Bedingung für kartesisches Moment aus allen Richtungen
- `createTurningTorqueCondition(...)`: Bedingung für Drehmoment
- `createTiltingTorqueCondition(...)`: Bedingung für Kippmoment

Zur Formulierung der Bedingung wird ein Frame unterhalb des Flansch-Koordinatensystems als Bezugssystem definiert. An diesem Frame, z. B. der Spitze eines Schraubwerkzeugs, wird das Moment ermittelt. Die Orientierung des Bezugssystems kann optional über einen Orientierungs-Frame definiert werden. Damit kann beispielsweise die gewünschte Orientierung der Schraube angegeben werden.

Durch die Angabe eines Grenzwertes wird definiert, ab welchem Betrag des kartesischen Moments die Bedingung erfüllt ist.

Das kartesische Moment wird aus den Werten der Gelenkmomenten-Sensoren berechnet. Abhängig von der Achskonfiguration variiert die Zuverlässigkeit der ermittelten kartesischen Momente. Soll die Güte der Berechnung ebenfalls berücksichtigt werden, kann ein Wert für die maximal erlaubte Ungenauigkeit angegeben werden. Überschreitet die vom System ermittelte Ungenauigkeit diesen Wert, ist die Bedingung für das kartesische Moment ebenfalls erfüllt.

### 15.20.6.1 Bedingung für kartesisches Moment aus allen Richtungen

#### Beschreibung

Über die statische Methode `createSpatialTorqueCondition(...)` kann eine Bedingung definiert werden, die unabhängig davon gilt, aus welcher Richtung das kartesische Moment auf einen Frame unterhalb des Flansches wirkt.

#### Syntax

```
CartesianTorqueCondition.createSpatialTorqueCondition(
    AbstractFrame measureFrame<, AbstractFrame orientationFrame<,
    double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measureFrame</i>	Frame unterhalb des Roboterflansches, an dem das wirkende Moment ermittelt wird. Über diesen Parameter wird die Position des Angriffspunktes des Moments definiert.
<i>orientationFrame</i>	Optional. Über diesen Parameter wird die Orientierung des Bezugssystems festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.

Element	Beschreibung
<i>threshold</i>	<p>Betrag des Moments, das maximal auf das Bezugssystem wirken darf (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Die Bedingung ist erfüllt, wenn der Betrag des aus beliebiger Richtung auf das Bezugssystem wirkenden Moments den hier angegebenen Wert überschreitet.</p>
<i>tolerance</i>	<p>Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Default: 10.0</p> <p>Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Momentenberechnung größer oder gleich dem hier angegebenen Wert ist.</p> <p>Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.</p>

### 15.20.6.2 Bedingung für Drehmoment

#### Beschreibung

Über die statische Methode `createTurningTorqueCondition(...)` kann eine Bedingung für das Drehmoment definiert werden. Betrachtet wird der Anteil des angreifenden Gesamtmoments, der um eine definierbare Achse eines Frames unterhalb des Flansches wirkt (z. B. um eine Achse des TCP).

#### Syntax

```
CartesianTorqueCondition.createTurningTorqueCondition(AbstractFrame measureFrame<, AbstractFrame orientationFrame<, CoordinateAxis direction, double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measureFrame</i>	<p>Frame unterhalb des Roboterflansches, an dem das wirkende Moment ermittelt wird.</p> <p>Über diesen Parameter wird die Position des Angriffs punkts des Moments definiert.</p>
<i>orientationFrame</i>	<p>Optional. Über diesen Parameter wird die Orientierung des Frames, bezogen auf den das wirkende Moment ermittelt wird, festgelegt.</p> <p>Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.</p>
<i>direction</i>	<p>Koordinatenachse des Bezugssystems.</p> <p>Der Anteil des Gesamtmoments, der um die hier angegebene Achse des Bezugssystems wirkt, wird mit der Bedingung geprüft.</p> <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>

Element	Beschreibung
<i>threshold</i>	<p>Betrag des Drehmoments, das maximal an der Achse des Bezugssystems angreifen darf (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Die Bedingung ist erfüllt, wenn der Betrag des Drehmoments den hier angegebenen Wert überschreitet.</p>
<i>tolerance</i>	<p>Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Default: 10.0</p> <p>Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Momentenberechnung größer oder gleich dem hier angegebenen Wert ist.</p> <p>Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.</p>

### 15.20.6.3 Bedingung für Kippmoment

#### Beschreibung

Über die statische Methode `createTiltingTorqueCondition(...)` kann eine Bedingung für das Kippmoment definiert werden. Betrachtet wird der Anteil des Gesamtmoments, der an einer Ebene des Bezugssystems angreift. Die Lage der Ebene wird festgelegt, indem man die Achse angibt, die senkrecht auf ihr steht (Oberflächennormale).

#### Syntax

```
CartesianTorqueCondition.createTiltingTorqueCondition(AbstractFrame measureFrame<, AbstractFrame orientationFrame<, CoordinateAxis normalDirection, double threshold<, double tolerance>)
```

Element	Beschreibung
<i>measure Frame</i>	<p>Frame unterhalb des Roboterflansches, an dem das wirkende Moment ermittelt wird.</p> <p>Über diesen Parameter wird die Position des Angriffs punkts des Moments definiert.</p>
<i>orientation Frame</i>	<p>Optional. Über diesen Parameter wird die Orientierung des Frames, bezogen auf den das wirkende Moment ermittelt wird, festgelegt.</p> <p>Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.</p>
<i>normal Direction</i>	<p>Koordinatenachse des Bezugssystems.</p> <p>Die hier angegebene Achse definiert die Oberflächennormale einer Ebene. Der Anteil des Gesamtmoments, der an der Ebene angreift, wird geprüft.</p> <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>

Element	Beschreibung
<i>threshold</i>	<p>Betrag des Kippmoments, das maximal an der durch ihre Oberflächennormale definierte Ebene des Bezugssystems angreifen darf (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Die Bedingung ist erfüllt, wenn der Betrag des Drehmoments den hier angegebenen Wert überschreitet.</p>
<i>tolerance</i>	<p>Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: Nm).</p> <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b></li> </ul> <p>Default: 10.0</p> <p>Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Momentenberechnung größer oder gleich dem hier angegebenen Wert ist.</p> <p>Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.</p>

### 15.20.7 Moment-Komponenten-Bedingung

#### Beschreibung

Mit der Moment-Komponenten-Bedingung kann geprüft werden, ob das kartesische Moment, das um die X-, Y- oder Z-Achse eines Frames unterhalb des Roboterflansches wirkt (z. B. um eine Achse des TCP), außerhalb eines definierten Bereichs liegt. Sie dient zur Überwachung des kartesischen Moments in einer bestimmten Richtung und kann z. B. zur Überwachung von Schraubprozessen eingesetzt werden.



Bei der Programmierung müssen die Lastdaten korrekt angegeben werden. Nur dann ist die Moment-Komponenten-Bedingung sinnvoll anwendbar.



In der Nähe singulärer Stellungen kann die Kraftschätzung keine sinnvollen Werte liefern. Es wird empfohlen, die Moment-Komponenten-Bedingung für solche Achskonfigurationen nicht einzusetzen. Alternativ kann die Achsmomenten-Bedingung genutzt werden oder die Achsstellung mittels der Redundanz so angepasst werden, dass keine Singularität mehr vorliegt.

Die Moment-Komponenten-Bedingung wird durch die Klasse TorqueComponentCondition repräsentiert. Für die Moment-Komponenten-Bedingung wird ein Frame unterhalb des Flansch-Koordinatensystems als Bezugssystem definiert. An diesem Frame, z. B. an der Spitze eines Schraubwerkzeugs, wird das Moment ermittelt. Die Orientierung des Bezugssystems kann optional über einen Orientierungs-Frame definiert werden.

Die Richtung, in der das Moment überprüft wird, wird mit einer der Koordinatenachsen des Bezugssystems festgelegt. Die Moment-Komponenten-Bedingung ist erfüllt, wenn das kartesische Moment, das um die festgelegte Koordinatenachse des Bezugssystems wirkt, außerhalb eines definierbaren Wertebereichs liegt.

Das kartesische Moment wird aus den Werten der Gelenkmomenten-Sensoren berechnet. Abhängig von der Achskonfiguration variiert die Zuverlässigkeit der ermittelten kartesischen Momente. Soll die Güte der Berechnung ebenfalls berücksichtigt werden, kann ein Wert für die maximal erlaubte Ungenauigkeit angegeben werden. Überschreitet die vom System ermittelte Ungenauigkeit diesen Wert, ist die Moment-Komponenten-Bedingung ebenfalls erfüllt.

**Konstruktor-Syntax** Die Klasse TorqueComponentCondition besitzt mehrere Konstruktoren, die sich in der Anzahl der Eingangsparameter unterscheiden:

```
TorqueComponentCondition(AbstractFrame measureFrame
<, AbstractFrame orientationFrame>, CoordinateAxis component,
double min, double max<, double tolerance>)
```

Element	Beschreibung
<i>measureFrame</i>	Frame unterhalb des Roboterflansches, bezogen auf den das wirkende Moment ermittelt wird. Über diesen Parameter wird die Position des Angriffs-punkts des Moments definiert.
<i>orientationFrame</i>	Optional. Über diesen Parameter wird die Orientierung des Frames, bezogen auf den das wirkende Moment ermittelt wird, festgelegt. Wird der Parameter <i>orientationFrame</i> nicht angegeben, legt <i>measureFrame</i> die Orientierung des Bezugssystems fest.
<i>coordinateAxis</i>	Koordinatenachse des Frames, bezogen auf den das wirkende Moment ermittelt wird. Legt die Richtung fest, in der das wirkende Moment geprüft wird. <ul style="list-style-type: none"> <li>■ <b>CoordinateAxis.X</b></li> <li>■ <b>CoordinateAxis.Y</b></li> <li>■ <b>CoordinateAxis.Z</b></li> </ul>
<i>min</i>	Untere Grenze des Wertebereichs für das um die Koordinatenachse des Bezugssystems wirkende Moment (Einheit: Nm). Die Moment-Komponenten-Bedingung ist erfüllt, wenn das Moment den hier angegebenen Wert unterschreitet.
<i>max</i>	Obere Grenze des Wertebereichs für das um die Koordinatenachse des Bezugssystems wirkende Moment (Einheit: Nm). Die Moment-Komponenten-Bedingung ist erfüllt, wenn das Moment den hier angegebenen Wert überschreitet. <b>Hinweis:</b> Der obere Grenzwert muss größer sein als untere Grenzwert: <i>max &gt; min</i> .
<i>tolerance</i>	Optional. Maximal zulässige Ungenauigkeit der ermittelten Werte (Einheit: Nm). <ul style="list-style-type: none"> <li>■ <b>&gt; 0.0</b> Default: 10.0</li> </ul> Die Bedingung ist erfüllt, wenn die Ungenauigkeit der Momentenberechnung größer oder gleich dem hier angegebenen Wert ist. Wird der Parameter nicht angegeben, wird automatisch der Default-Wert verwendet.

### 15.20.8 Bahnbezogene Bedingung

#### Beschreibung

Bahnbezogene Bedingungen werden immer im Zusammenhang mit einem Bewegungsbefehl verwendet. Sie dienen als Abbruchkriterium oder als Trigger für bahnbezogene Schaltaktionen.

Die Bedingung definiert den Punkt auf der geplanten Bahn (Schaltpunkt), an dem eine Bewegung abgebrochen oder eine gewünschte Aktion ausgelöst werden soll. Ist der Schaltpunkt erreicht, gilt die Bedingung als erfüllt.



Der Bremsvorgang oder die definierte Aktion wird erst bei Erreichen des Schaltpunkts ausgelöst. Dadurch kommt der Roboter bei Verwendung der bahnbezogenen Bedingung als Abbruchbedingung nicht direkt auf dem Schaltpunkt, sondern hinter dem Schaltpunkt zum Stillstand.

Der Schaltpunkt kann durch eine örtliche und/oder zeitliche Verschiebung festgelegt werden. Die Verschiebung kann sich wahlweise auf den Startpunkt oder den Zielpunkt einer Bewegung beziehen.



Wird eine zeitliche Verschiebung definiert, hat eine Änderung des Overrides Einfluss auf den Schaltpunkt. Die mit einer bahnbezogenen Bedingung verknüpfte Aktion wird daher nur bei einem effektiven Override von 100 % und in den Betriebsarten T2 oder Automatik am definierten Schaltpunkt ausgelöst.

Bahnbezogene Bedingungen sind vom Datentyp MotionPathCondition.

## Konstruktor-

### Syntax

Die Klasse MotionPathCondition besitzt folgenden Konstruktor:

```
MotionPathCondition(ReferenceType reference, double distance,  
long delay)
```

## Statische Methoden

Ein MotionPathCondition-Objekt kann auch über eine der folgenden statischen Methoden erzeugt werden:

```
MotionPathCondition.createFromDelay(ReferenceType reference, long delay)
```

```
MotionPathCondition.createFromDistance(ReferenceType reference, double distance)
```

Element	Beschreibung
<i>reference</i>	Datentyp: com.kuka.roboticsAPI.conditionModel.ReferenceType  Bezugspunkt der Bedingung  ■ <b>ReferenceType.START</b> : Startpunkt ■ <b>ReferenceType.DEST</b> : Zielpunkt

Element	Beschreibung
<i>distance</i>	<p>Örtliche Verschiebung in Bezug auf den Bezugspunkt der Bedingung</p> <p>Bei CP-Bewegungen gibt <i>distance</i> den kartesischen Abstand zwischen Schaltpunkt und Bezugspunkt an (= Strecke auf der Bahn, die Schaltpunkt und Bezugspunkt verbindet), nicht die kürzeste Verbindung zwischen diesen Punkten. (Einheit: mm)</p> <p>Bei PTP-Bewegungen wird mit <i>distance</i> kein kartesischer Abstand angegeben, sondern ein Bahnpараметer ohne Einheit.</p> <ul style="list-style-type: none"> <li>■ Negativer Wert: Verschiebung entgegen der Bewegungsrichtung</li> <li>■ Positiver Wert: Verschiebung in Bewegungsrichtung</li> </ul> <p>(&gt;&gt;&gt; "Maximale Verschiebung" Seite 358)</p>
<i>delay</i>	<p>Zeitliche Verschiebung in Bezug auf den durch <i>distance</i> definierten Bahnpunkt. Oder wenn <i>distance</i> nicht definiert ist, auf den Bezugspunkt der Bedingung. (Einheit: ms)</p> <ul style="list-style-type: none"> <li>■ Negativer Wert: Verschiebung entgegen der Bewegungsrichtung</li> <li>■ Positiver Wert: Verschiebung in Bewegungsrichtung</li> </ul> <p>Phasen, in denen die Applikation pausiert ist, gehen nicht in die Zeitmessung ein.</p> <p>(&gt;&gt;&gt; "Maximale Verschiebung" Seite 358)</p>

## Maximale Verschiebung

Der Schaltpunkt kann nur bis zu bestimmten Grenzen verschoben werden. Die Grenzen gelten für die Gesamtverschiebung, die sich aus örtlicher und zeitlicher Verschiebung ergibt.

- Negative Verschiebung maximal bis zum Startpunkt der Bewegung
- Positive Verschiebung maximal bis zum Zielpunkt der Bewegung

Folgende Parametrierungen dürfen nicht verwendet werden, da sie zwangsläufig zu einer Verschiebung über die zulässigen Grenzen hinaus und zu einem Laufzeitfehler führen:

Wertekombination	Auswirkung
reference = ReferenceType.START <i>distance</i> < 0	Schaltpunkt liegt vor Bewegungsstart.
reference = ReferenceType.START <i>distance</i> = 0 <i>delay</i> < 0	
reference = ReferenceType.DEST <i>distance</i> > 0	Schaltpunkt liegt nach Bewegungsende.
reference = ReferenceType.DEST <i>distance</i> = 0 <i>delay</i> > 0	

Wurde eine gültige Wertekombination verwendet, kann es trotzdem vorkommen, dass der Schaltpunkt über die zulässigen Grenzen hinaus verschoben wird. In diesen Fällen ist das Verhalten wie folgt:

- Eine Bedingung, die vor Bewegungsstart erfüllt ist, löst am Startpunkt der Bewegung aus.
- Eine Bedingung, die nach Bewegungsende erfüllt ist, löst nie aus.

**Beispiel**

Für eine Klebeapplikation soll eine bahnbezogene Bedingung formuliert werden. Die Klebespur soll 5 cm vor dem Zielpunkt der Bewegung enden. Damit der Kleberfluss rechtzeitig beendet werden kann, muss die Bedingung 700 ms vor Erreichen dieser Distanz zum Ziel erfüllt sein.

```
MotionPathCondition glueStop = new
MotionPathCondition(ReferenceType.DEST, -50.0, -700);
```

**15.20.9 Bedingung für boolesche Signale****Beschreibung**

Mit der booleschen Signalbedingung können digitale 1-Bit-Eingänge geprüft werden. Die Bedingung ist erfüllt, wenn ein boolescher Eingang einen bestimmten Zustand besitzt.

Boolesche Signalbedingungen sind vom Datentyp BooleanIOPCondition.

**Konstruktor-Syntax**

```
BooleanIOPCondition(AbstractIO booleanSignal, boolean booleanIO-
Value)
```

Element	Beschreibung
boolean Signal	Boolesches Eingangssignal, das geprüft wird
boolean IOValue	Zustand des Eingangssignals, bei dem die Bedingung erfüllt ist <ul style="list-style-type: none"> <li>■ true, false</li> </ul>

**Beispiel**

Über einen Schalter wird ein boolesches digitales Eingangssignal geliefert. Um in einer Applikation auf das Signal reagieren zu können, soll eine boolesche Signalbedingung formuliert werden. Die Bedingung soll erfüllt sein, sobald durch Betätigen des Schalters ein HIGH-Pegel (Zustand TRUE) anliegt.

```
SwitchesIOPGroup switches = new SwitchesIOPGroup(...);
AbstractIO switch_1 = switches.getInput("Switch1");

BooleanIOPCondition switch1_active = new BooleanIOPCondition(switch_1,
true);
```

**15.20.10 Bedingung für Wertebereich eines Signals****Beschreibung**

Mit der Bedingung für den Wertebereich eines Signals kann der Wert eines digitalen oder analogen Eingangs geprüft werden. Die Bedingung ist erfüllt, wenn der Wert des Signals innerhalb eines definierten Bereichs liegt.

Bedingungen für Wertebereiche sind vom Datentyp IORangeCondition.

**Konstruktor-Syntax**

```
IORangeCondition(AbstractIO signal, Number minValue, Number
maxValue)
```

Element	Beschreibung
signal	Analoges oder digitales Signal, das geprüft wird

Element	Beschreibung
<i>minValue</i>	Untere Grenze des Wertebereichs, in dem die Bedingung erfüllt ist  Der Wert, den das Signal liefert, muss größer oder gleich <i>minValue</i> sein.
<i>maxValue</i>	Obere Grenze des Wertebereichs, in dem die Bedingung erfüllt ist  Der Wert, den das Signal liefert, muss kleiner oder gleich <i>maxValue</i> sein.

**Beispiel**

Über einen Temperatursensor wird ein analoges Eingangssignal geliefert, dessen Wert in einem Bereich zwischen 0 °C und 2000 °C liegen kann. Sobald eine Schwelle von 35 °C überschritten wird, soll eine Bedingung zur Überwachung des Sensorsignals erfüllt sein.

```
SensorIOGroup sensors = new SensorIOGroup(...);
AbstractIO temperatureSensor =
sensors.getInput("TemperatureSensor2");

IORangeCondition tempHigher35 = new
IORangeCondition(temperatureSensor, 35.0, 2000.0);
```

## 15.21 Abbruchbedingungen für Bewegungsbefehle

Für bestimmte Prozesse ist es nötig, dass eine geplante Bewegung nicht bis zum Ende ausgeführt wird, sondern bei Eintreten definierbarer Ereignisse abgebrochen wird. Bei Fügeprozessen muss der Roboter beispielsweise stoppen, wenn eine Kraftschwelle erreicht ist.

### 15.21.1 Abbruchbedingung festlegen

**Beschreibung**

Abbruchbedingungen sind Bedingungen, die zum Abbruch einer Bewegung führen sollen. Eine Abbruchbedingung ist erfüllt, wenn sie bereits vor Bewegungsbeginn den Zustand TRUE hat oder während der Bewegung in den Zustand TRUE wechselt.

Bedingungen sind als Objekte vom Typ ICondition definiert. Die verfügbaren Bedingungstypen gehören zum Paket com.kuka.roboticsAPI.conditionModel.

Eine Übersicht der verfügbaren Bedingungstypen ist hier zu finden:  
(>>> 15.20.1 "Bedingungen in der RoboticsAPI" Seite 341)

Um eine Abbruchbedingung für eine Bewegung festzulegen, wird dem Bewegungsbefehl mit der Motion-Methode breakWhen(...) ein Objekt des gewünschten Bedingungstyps übergeben.

breakWhen(...) kann bei der Programmierung eines Bewegungsbefehls mehrmals aufgerufen werden, um verschiedene Abbruchbedingungen für eine Bewegung festzulegen. Die einzelnen Abbruchbedingungen werden dann logisch ODER-verknüpft.

Folgende Punkte sind bei der Programmierung von Abbruchbedingungen zu beachten:

- Bei einem Spline-Block können Abbruchbedingungen nur für den gesamten Spline-Block programmiert werden. Abbruchbedingungen für einzelne Spline-Segmente sind nicht zulässig.
- Tritt eine Abbruchbedingung ein, die für eine Bewegung innerhalb eines MotionBatch definiert ist, wird diese abgebrochen und dann der nächste Bewegungsbefehl im Batch ausgeführt. Tritt eine für den gesamten Motion-

Batch definierte Abbruchbedingung ein, wird der gesamte MotionBatch abgebrochen.

- Eine Abbruchbedingung führt zum Abbruch der aktuell ausgeführten Bewegung. Folgebewegungen werden, sofern in der Applikation keine geeignete Reaktionsstrategie programmiert ist, unmittelbar nach der abgebrochenen Bewegung ausgeführt.
- Bei überschliffenen Bewegungen gehört der Überschleifbogen zur Bahn der Folgebewegung. Auf dem Überschleifbogen wirken daher nur die Abbruchbedingungen für die Folgebewegung.
- Tritt bei einer überschliffenen Bewegung die Abbruchbedingung kurz vor Erreichen des Überschleipunktes ein und würde der Roboter dadurch erst auf dem Überschleifbogen zum Stehen kommen, wird der Roboter bei Erreichen des Überschleifbogens wieder beschleunigt, um die Folgebewegung auszuführen.

## Syntax

```
motion.breakWhen( condition_1<, condition_2, ... >);
```

## Erläuterung der Syntax

Element	Beschreibung
<i>motion</i>	<p>Typ: Motion</p> <p>Bewegung, für die eine Abbruchbedingung festgelegt werden soll</p> <p>Beispiel:</p> <ul style="list-style-type: none"> <li>■ <code>ptpgetApplicationData().getFrame("/P1")</code></li> </ul>
<i>condition</i>	<p>Typ: ICondition</p> <p>Parametriertes ICondition-Objekt, das eine Abbruchbedingung beschreibt</p>

## Beispiel

Eine LIN-Bewegung wird abgebrochen, wenn das Moment in Achse A3 kleiner oder gleich -12 Nm ist oder größer oder gleich 0 Nm ist.

```
JointTorqueCondition cond_1 = new JointTorqueCondition(JointEnum.J3,
-12.0, 0.0);
robot.move(lin(getApplicationData().getFrame("/P10"))
.breakWhen(cond_1));
```

## 15.21.2 Abbruchbedingungen auswerten

### Beschreibung

Wurden für einen Bewegungsbefehl Abbruchbedingungen festgelegt, können verschiedene Informationen zum Abbruch einer Bewegung ausgelesen werden. Dafür wird der Bewegungsbefehl in einer IMotionContainer-Variable zwischengespeichert. Von dieser Variablen kann über die Methode `getFiredBreakConditionInfo()` ein Objekt vom Typ IFiredConditionInfo abgefragt werden, das die Informationen zum Abbruch der Bewegung enthält. Wenn während der Bewegung keine Abbruchbedingung eintritt, gibt `getFiredBreakConditionInfo()` null zurück.

### Syntax

```
IMotionContainer motionCmd = motion.breakWhen(...);
IFiredConditionInfo firedCondInfo =
motionCmd.getFiredBreakConditionInfo();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>motion</i>	Bewegungsanweisung Beispiel: ■ Ibr.move(ptp(getApplicationContext().getFrame("/P1")))
<i>motionCmd</i>	Typ: IMotionContainer Zwischenspeicher für den Bewegungsbefehl
<i>firedCondInfo</i>	Typ: IFiredConditionInfo Informationen zum Abbruch der Bewegung

**Übersicht**

Die Schnittstelle IFiredConditionInfo stellt folgende Methoden zur Verfügung:

Methode	Beschreibung
getFiredCondition()	Rückgabewert: ICondition Abfrage der Bedingung, die zum Abbruch einer Bewegung geführt hat
getPositionInfo()	Rückgabewert: PositionInformation Abfrage der Roboterposition zum Zeitpunkt des Auslösens der Abbruchbedingung
getStoppedMotion()	Rückgabewert: IMotion Abfrage, welches Segment eines Spline-Blocks oder welche Bewegung eines MotionBatch abgebrochen wurde

### 15.21.2.1 Abbruchbedingung abfragen

**Beschreibung**

Die Bedingung, die zum Abbruch einer Bewegung geführt hat, kann mit der Methode getFiredCondition() abgefragt werden. Der Rückgabewert ist vom Typ ICondition und kann mithilfe der Methode equals(...) mit den übergebenen Abbruchbedingungen verglichen werden.

Die Abfrage ist vor allem dann sinnvoll, wenn durch wiederholten Aufruf der Methode breakWhen(...) mehrere Abbruchbedingungen für eine Bewegung definiert wurden.

**Syntax**

```
ICondition firedCondition = firedCondInfo.getFiredCondition();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>firedCondition</i>	Typ: ICondition Variable für den Rückgabewert. Die Variable enthält die Bedingung, die zum Abbruch der Bewegung geführt hat.
<i>firedCondInfo</i>	Typ: IFiredConditionInfo Informationen zum Abbruch der Bewegung

**Beispiel**

Es werden die Abbruchbedingungen "cond1" und "cond2" erzeugt.

```
ICondition cond1;
ICondition cond2;
cond1 = new ...;
cond2 = new ...;
```

Die Abbruchbedingungen "cond1" und "cond2" werden einer LIN-Bewegung mit breakWhen(...) übergeben. Über die Variable "motionCmd" vom Typ IMotionContainer kann der Bewegungsbefehl ausgewertet werden.

```
IMotionContainer motionCmd =
lbr.move(lingetApplicationData().getFrame("P10")).breakWhen(cond1).breakWhen(cond2);
```

Von "motionCmd" wird die Informationen zum Abbruch der Bewegung abgefragt. Wenn die abgefragte Information ungleich `null` ist, wurde die Bewegung abgebrochen. Nur für diesen Fall wird abgefragt, welche Abbruchbedingung eingetreten ist.

```
IFiredConditionInfo firedInfo = motionCmd.getFiredConditionInfo();
if(firedInfo != null) {

    ICondition firedCond = firedInfo.getFiredCondition();
    if(firedCond.equals(cond1)) {
        ...
    }
    ...
}
```

### 15.21.2.2 Roboterposition zum Abbruchzeitpunkt abfragen

#### Beschreibung

Die Roboterposition zum Zeitpunkt des Auslösens der Abbruchbedingung kann über die Methode `getPositionInfo()` abgefragt werden.

Über den Rückgabewert vom Typ `PositionInformation` kann auf folgende Positionsinformationen zugegriffen werden:

- achsspezifische Istposition
- kartesische Istposition
- achsspezifische Sollposition
- kartesische Sollposition
- Soll-Ist-Differenz (translatorisch)
- Soll-Ist-Differenz (rotatorisch)

#### Syntax

```
PositionInformation firedPosInfo =
firedCondInfo.getPositionInfo();
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>firedPosInfo</code>	Typ: <code>PositionInformation</code> Variable für den Rückgabewert. Der Rückgabewert enthält die Positionsinformationen zum Zeitpunkt des Auslösens der Abbruchbedingung.
<code>firedCondInfo</code>	Typ: <code>IFiredConditionInfo</code> Informationen zum Abbruch der Bewegung

#### Beispiel

Die kartesische Istposition des Roboters zum Zeitpunkt des Auslösens der Abbruchbedingung wird über die Methode `getCurrentCartesianPosition()` abgefragt.

```
PositionInformation firedPosInfo = firedInfo.getPositionInfo();
Frame firedCurrPos = firedPosInfo.getCurrentCartesianPosition();
```

### 15.21.2.3 Abgebrochene Bewegung abfragen (Spline-Block, MotionBatch)

#### Beschreibung

Abbruchbedingungen können für einen gesamten Spline-Block oder MotionBatch definiert sein. Tritt eine Abbruchbedingung ein, wird der gesamte Spline-Block oder MotionBatch abgebrochen.

Über die Methode getStoppedMotion() kann abgefragt werden, welches Spline-Segment oder welche Bewegung eines MotionBatch abgebrochen wurde. Der Rückgabewert ist vom Typ IMotion.

### Syntax

```
IMotion stoppedMotion = firedCondInfo.getStoppedMotion();
```

### Erläuterung der Syntax

Element	Beschreibung
stoppedMotion	Typ: IMotion Variable für den Rückgabewert. Die Variable enthält die abgebrochene Bewegung.
firedCondInfo	Typ: IFiredConditionInfo Informationen zum Abbruch der Bewegung

### Beispiel

Abfrage am Beispiel eines Spline-Blocks:

```
ICondition stopCondition = new ...;
...
Spline splineMotion = new Spline(
    splgetApplicationData().getFrame("/P1"),
    circ(getApplicationData().getFrame("/P2"),
        getApplicationData().getFrame("/P3")),
    splgetApplicationData().getFrame("/P4")).setCartVelocity(150),
    lin(getApplicationData().getFrame("/P5"))
).setCartVelocity(250).breakWhen(stopCondition);

IMotionContainer splineCont = robot.move(splineMotion);

IFiredConditionInfo firedInfoSpline =
splineCont.getFiredConditionInfo();
if(firedInfoSpline != null){
    IMotion stoppedMotion = firedInfoSpline.getStoppedMotion();
    ...
}
```

## 15.22 Bahnbezogene Schaltaktionen (Trigger)

Ein Trigger ist ein Ereignis, das als Auslöser für benutzerdefinierte bahnbezogene Aktionen eingesetzt wird. Tritt beim Ausführen einer Bewegung ein bestimmtes Ereignis ein, wird die Aktion ausgelöst. Die Aktion wird parallel zur Roboterbewegung ausgeführt. Beispielsweise soll bei einer Zustellbewegung der Greifer rechtzeitig geöffnet werden, so dass er geöffnet ist, wenn eine freie Ablageposition für das transportierte Werkstück erreicht ist.

### 15.22.1 Trigger programmieren

#### Beschreibung

Ereignisse, die eine bahnbezogene Schaltaktion auslösen sollen, werden als Trigger bezeichnet. Ereignisse werden mithilfe von Bedingungen definiert. Ein Ereignis tritt ein, wenn die definierte Bedingung bereits vor Bewegungsbeginn den Zustand TRUE hat oder während der Bewegung in den Zustand TRUE wechselt.

Bedingungen sind als Objekte vom Typ ICondition definiert. Die verfügbaren Bedingungstypen gehören zum Paket com.kuka.roboticsAPI.conditionModel.

Eine Übersicht der verfügbaren Bedingungstypen ist hier zu finden:  
(>>> 15.20.1 "Bedingungen in der RoboticsAPI" Seite 341)

Um einen Trigger zu programmieren, wird einem Bewegungsbefehl mit der Motion-Methode triggerWhen(...) ein Objekt des gewünschten Bedin-

gungstyps und ein ITriggerAction-Objekt, das die auszuführende Aktion beschreibt, übergeben.

triggerWhen(...) kann bei der Programmierung eines Bewegungsbefehls mehrmals aufgerufen werden, um verschiedene Trigger für eine Bewegung festzulegen. Die Ausführung der zugehörigen Schaltaktionen ist nur abhängig davon, ob das auslösende Ereignis eintritt, und wird nicht durch die Reihenfolge des Aufrufs mit triggerWhen(...) beeinflusst.



Während eine Aktion ausgeführt wird, kann das auslösende Ereignis diese Aktion nicht erneut auslösen. Der Trigger ist erst wieder wirksam, wenn die Methode triggerWhen(...) beendet ist. Die Anzahl der während der Methodenausführung verpassten Ereignisse kann abgefragt werden.  
(>>> 15.22.3 "Trigger-Informationen auswerten" Seite 366)

## Syntax

```
motion.triggerWhen( condition, action );
```

## Erläuterung der Syntax

Element	Beschreibung
<i>motion</i>	Typ: Motion Bewegung, für die ein Trigger festgelegt werden soll Beispiel: ■ ptpgetApplicationData().getFrame("/P1"))
<i>condition</i>	Typ: ICondition Parametriertes ICondition-Objekt, das die Bedingung für den Trigger beschreibt
<i>action</i>	Typ: ITriggerAction ITriggerAction-Objekt, das die auszuführende Aktion beschreibt (>>> 15.22.2 "Bahnbezogene Schaltaktion programmieren" Seite 365)

## 15.22.2 Bahnbezogene Schaltaktion programmieren

### Beschreibung

Die Aktion, die bahnbezogen bei Eintreten eines Ereignisses ausgeführt werden soll, wird über ein ITriggerAction-Objekt festgelegt. ITriggerAction ist eine Schnittstelle aus dem Paket com.kuka.roboticsAPI.conditionModel. Diese Schnittstelle stellt aktuell keine Methoden zur Verfügung.

Für die Programmierung der Aktionen kann die von ITriggerAction abgeleitete Schnittstelle ICallbackAction verwendet werden. Die Schnittstelle besitzt die Methode onTriggerFired(...). Im Methodenkörper von onTriggerFired(...), kann die Aktion programmiert werden, die beim Auslösen des Triggers ausgeführt werden soll.

Ein ICallbackAction-Objekt kann in beliebig vielen Triggern eingesetzt werden.



Die Methode onTriggerFired(...) wird nicht in Echtzeit aufgerufen. Ein bestimmtes Zeitverhalten kann daher nicht garantiert werden. Es kann zu einer verzögerten Ausführung der Aktion kommen.

## Syntax

```
ICallbackAction action = new ICallbackAction() {  
    @Override
```

```
public void onTriggerFired(IFiredTriggerInfo
triggerInformation) {
    // Aktion, die ausgeführt wird
}
};
```

### Erläuterung der Syntax

Element	Beschreibung
action	Typ: ICallbackAction ICallbackAction-Objekt, das die Aktion beschreibt, die mit triggerWhen(...) übergeben wird
onTrigger Fired(...)	Methode, deren Ausführung durch den Trigger ausgelöst wird
triggerInformation	Typ: IFiredTriggerInfo Enthält Informationen über den auslösenden Trigger (>>> 15.22.3 "Trigger-Informationen auswerten" Seite 366)

### Beispiel

Während der Bewegung zum Punkt "P1" wird der Ausgang "DO1" immer dann umgeschalten, wenn der Eingang "DI1" TRUE ist.

```
//set trigger action
ICallbackAction toggleOut_1 = new ICallbackAction() {

    @Override
    public void onTriggerFired(IFiredTriggerInfo triggerInformation) {
        //toggle output state when trigger fired
        if(IOs.getDO1())
        {
            IOs.setDO1(false);
        }
        else
        {
            IOs.setDO1(true);
        }

    }
};

//set trigger condition
BooleanIOCondition buttonPressed = new BooleanIOCondition(IOs.getInput("DI1"), true);

//motion with trigger
robot.move(ptp(P1)).triggerWhen(buttonPressed, toggleOut_1));
robot.move(ptp(P2));
```

### 15.22.3 Trigger-Informationen auswerten

Beim Auslösen eines Triggers wird die Methode onTriggerFired(...) aufgerufen. Der Methode onTriggerFired(...) wird das Objekt triggerInformation vom Typ IFiredTriggerInfo übergeben, das verschiedene Informationen über den auslösenden Trigger enthält. Diese Trigger-Informationen können abgefragt werden.

### Übersicht

Folgende Methoden der Klasse IFiredTriggerInfo stehen zur Verfügung:

Methode	Beschreibung
getFiredCondition()	Rückgabetyp: ICondition Abfrage der Bedingung, die den Trigger ausgelöst hat
getMissedEvents()	Rückgabetyp: int Abfrage, wie oft das Ereignis, das den Trigger ausgelöst hat, noch eingetreten ist, während die vom Trigger ausgelöste Aktion ausgeführt wurde <b>Hinweis:</b> Während eine Aktion ausgeführt wird, kann das auslösende Ereignis diese Aktion nicht erneut auslösen.
getMotionContainer()	Rückgabetyp: IMotionContainer Abfrage des Bewegungsbefehls, bei dessen Ausführung der Trigger ausgelöst hat
getPositionInformation()	Rückgabetyp: PositionInformation Abfrage der Positionsinformationen zum Zeitpunkt des Auslösens des Triggers Der Rückgabewert enthält folgende Positionsinformationen: <ul style="list-style-type: none"><li>■ achsspezifische Istposition</li><li>■ kartesische Istposition</li><li>■ achsspezifische Sollposition</li><li>■ kartesische Sollposition</li><li>■ Soll-Ist-Differenz (translatorisch)</li><li>■ Soll-Ist-Differenz (rotatorisch)</li></ul>
getTriggerTime()	Rückgabetyp: java.util.Date Abfrage des Zeitpunkts, zu dem der Trigger ausgelöst hat

Um die mit getPositionInformation() erhaltenen Positionsinformationen abzufragen, stehen folgende Methoden der Klasse PositionInformation zur Verfügung:

Methode	Beschreibung
getCommandedCartesianPosition()	Rückgabetyp: Frame Abfrage der kartesischen Sollposition zum Trigger-Zeitpunkt
getCommandedJointPosition()	Rückgabetyp: JointPosition Abfrage der achsspezifischen Sollposition zum Trigger-Zeitpunkt
getCurrentCartesianPosition()	Rückgabetyp: Frame Abfrage der kartesischen Istposition zum Trigger-Zeitpunkt
getCurrentJointPosition()	Rückgabetyp: JointPosition Abfrage der achsspezifischen Istposition zum Trigger-Zeitpunkt

**Beispiel 1** Wenn der Trigger auslöst, werden der Auslösezeitpunkt und die auslösende Bedingung auf der smartHMI ausgegeben.

```
BooleanIOCondition in1 = new BooleanIOCondition(_input_1, true);

ICallbackAction ica = new ICallbackAction() {

    @Override
    public void onTriggerFired(IFiredTriggerInfo triggerInformation)
    {
```

```

        getLogger().info("TriggerTime: "+
triggerInformation.getTriggerTime().toString());
        getLogger().info("TriggerCondition: "+
triggerInformation.getFiredCondition().toString());
    }
};

robot.move(P1).triggerWhen(in1, ica));

```

**Beispiel 2**

Die achsspezifische und kartesische Roboterposition zum Trigger-Zeitpunkt werden abgefragt.

```

BooleanIOCondition in1 = new BooleanIOCondition(_input_1, true);

ICallbackAction ica = new ICallbackAction() {

    @Override
    public void onTriggerFired(IFiredTriggerInfo triggerInformation)
    {
        PositionInformation posInfo =
triggerInformation.getPositionInformation();
        posInfo.getCommandedCartesianPosition();
        posInfo.getCommandedJointPosition();
        posInfo.getCurrentCartesianPosition();
        posInfo.getCommandedJointPosition();
    }
};

robot.move(P1).triggerWhen(in1, ica));

```

## 15.23 Überwachen von Prozessen (Monitoring)

Monitoring bedeutet, einen Prozess mithilfe eines Listeners zu überwachen, um während einer laufenden Applikation auf bestimmte Ereignisse reagieren zu können.

Diese Ereignisse sind Zustandsänderungen definierter Bedingungen. Der Listener überwacht den Zustand der Bedingung. Ändert sich der Zustand der Bedingung, wird der Listener benachrichtigt und die festgelegte Behandlungsroutine als Reaktion ausgelöst.

Der Listener wird während des Ausführens einer Behandlungsroutine nicht benachrichtigt, wenn weitere Ereignisse auftreten. Ist die Behandlungsroutine abgeschlossen, werden diese Ereignisse nur dann an den Listener übermittelt und behandelt, wenn die passende Benachrichtigungsart definiert ist.

(>>> 15.23.3 "Listener für Benachrichtigung bei Zustandsänderung registrieren" Seite 370)

### 15.23.1 Listener für das Überwachen von Bedingungen

Zur Überwachung einer Bedingung stehen verschiedene Listener-Schnittstellen aus dem Paket com.kuka.roboticsAPI.conditionModel zur Verfügung. Die Listener unterscheiden sich typmäßig darin, dass sie jeweils bei einer bestimmten Zustandsänderung der überwachten Bedingung benachrichtigt werden.

Jeder Listener-Typ deklariert eine Methode, die ausgeführt wird, wenn der Listener benachrichtigt wird. Im Methodenrumpf dieser Methode wird die gewünschte Behandlungsroutine programmiert.

Datentyp	Beschreibung
IRisingEdgeListener	<p>Benachrichtigung, wenn die überwachte Bedingung erfüllt wird (steigende Flanke, Zustandsänderung FALSE &gt; TRUE).</p> <p>Methode für die Behandlungsroutine:</p> <ul style="list-style-type: none"> <li>■ onRisingEdge(...)</li> </ul>
IFallingEdgeListener	<p>Benachrichtigung, wenn die überwachte Bedingung nicht mehr erfüllt wird (fallende Flanke, Zustandsänderung TRUE &gt; FALSE).</p> <p>Methode für die Behandlungsroutine:</p> <ul style="list-style-type: none"> <li>■ onFallingEdge(...)</li> </ul>
IAnyEdgeListener	<p>Benachrichtigung bei jeder Zustandsänderung der Bedingung (steigende oder fallende Flanke, Zustandsänderung FALSE &gt; TRUE oder TRUE &gt; FALSE).</p> <p>Methode für die Behandlungsroutine:</p> <ul style="list-style-type: none"> <li>■ onAnyEdge(...)</li> </ul>

**Übersicht**

Um auf die Zustandsänderung einer Bedingung reagieren zu können, sind folgende Schritte bei der Programmierung erforderlich:

Schritt	Beschreibung
1	<p>Ein Listener-Objekt für die Überwachung der Bedingung erzeugen.</p> <p>(&gt;&gt;&gt; 15.23.2 "Listener-Objekt für Überwachung einer Bedingung erzeugen" Seite 369)</p>
2	Gewünschte Behandlungsroutine in der Listener-Methode programmieren.
3	<p>Den Listener für die Benachrichtigung bei einer Zustandsänderung der Bedingung registrieren.</p> <p>(&gt;&gt;&gt; 15.23.3 "Listener für Benachrichtigung bei Zustandsänderung registrieren" Seite 370)</p>
4	<p>Falls nicht bereits durch die für die Registrierung gewählte Methode erledigt, den Benachrichtigungsdienst für den Listener aktivieren.</p> <p>(&gt;&gt;&gt; 15.23.4 "Benachrichtigungsdienst für Listener aktivieren oder deaktivieren" Seite 372)</p>

**15.23.2 Listener-Objekt für Überwachung einer Bedingung erzeugen****Beschreibung**

Hier wird die Syntax eines Listener-Objekts am Beispiel des Listeners IAnyEdgeListener beschrieben. Die bei der Erzeugung des Objekts automatisch deklarierte Listener-Methode onAnyEdge(...) besitzt Eingangsparameter. Diese Eingangsparameter enthalten Informationen zum Ereignis, das die Ausführung der Methode ausgelöst hat, und können abgefragt und ausgewertet werden.

Die Listener-Objekte der anderen Listener-Typen werden auf die gleiche Weise erzeugt und sind analog aufgebaut.

**Syntax**

```
IAnyEdgeListener condListener = new IAnyEdgeListener() {
    @Override
    public void onAnyEdge(ConditionObserver conditionObserver, Date time, int missedEvents, boolean conditionValue)
    {
```

```
// Reaktion auf Zustandsänderung
}
};
```

### Erläuterung der Syntax

Element	Beschreibung
<i>condListener</i>	Typ: IAnyEdgeListener Name des Listener-Objekts
Eingangsparameter der Listener-Methode:	
<i>condition Observer</i>	Typ: ConditionObserver Objekt, das den Listener benachrichtigt hat
<i>time</i>	Typ: Date Datum und Uhrzeit, zu der der Listener benachrichtigt wurde
<i>missed Events</i>	Typ: int Anzahl der aufgetretenen Zustandsänderungen, die nicht behandelt wurden  Mögliche Ursachen für nicht behandelte Ereignisse: <ul style="list-style-type: none"><li>■ Benachrichtigungsdienst war deaktiviert als das auslösende Ereignis eingetreten ist.</li><li>■ Die Behandlungsroutine wurde gerade ausgeführt, als das auslösende Ereignis erneut eingetreten ist.</li></ul> Diese Ereignisse können bei Verwendung der Benachrichtigungsart <b>NotificationType.MissedEvents</b> behandelt werden. ( <b>&gt;&gt;&gt; "NotificationType"</b> Seite 371)
<i>condition Value</i>	Typ: boolean Nur vorhanden bei der Listener-Methode <code>onAnyEdge(...)</code> . Gibt die Flanke an, durch die die Methode aufgerufen wurde. <ul style="list-style-type: none"><li>■ <b>true</b>: steigende Flanke (Zustandsänderung FALSE &gt; TRUE)</li><li>■ <b>false</b>: fallende Flanke (Zustandsänderung TRUE &gt; FALSE)</li></ul>

### 15.23.3 Listener für Benachrichtigung bei Zustandsänderung registrieren

#### Beschreibung

Um einen Listener für die Benachrichtigung bei einer Zustandsänderung einer Bedingung zu registrieren, wird ein Objekt vom Typ ConditionObserver benötigt.

Für die Erzeugung eines Objekts vom Typ ConditionObserver muss zunächst der `ObserverManager` der Applikation über die Methode `getObserverManager()` abgefragt werden. Die Klasse `ObserverManager` stellt verschiedene Methoden bereit, um das benötigte Objekt zu erzeugen.

- `createAndEnableConditionObserver(...)`  
Der Benachrichtigungsdienst für den Listener ist sofort aktiv.
- `createConditionObserver(...)`  
Der Benachrichtigungsdienst für den Listener ist nicht sofort aktiv, sondern muss eigens aktiviert werden.  
(**>>> 15.23.4 "Benachrichtigungsdienst für Listener aktivieren oder deaktivieren"** Seite 372)

Die jeweils übergebenen Parameter sind für beide Methoden identisch.

### Syntax

```
ConditionObserver myObserver =
getObserverManager().createAndEnableConditionObserver
(condition, notificationType, listener)
```

### Erläuterung der Syntax

Element	Beschreibung
<i>myObserver</i>	Typ: ConditionObserver Objekt, das die definierte Bedingung überwacht
<i>condition</i>	Typ: ICondition Bedingung, die überwacht wird
<i>notification Type</i>	Typ: Enum vom Typ NotificationType Benachrichtigungsart Legt fest, bei welchen Ereignissen der Listener benachrichtigt werden soll, um die gewünschte Behandlungsroutine auszuführen. (>>> "NotificationType" Seite 371)
<i>listener</i>	Typ: IRisingEdgeListener, IFallingEdgeListener oder IAnyEdgeListener Listener-Objekt, das registriert wird

### NotificationType

Das Enum vom Typ NotificationType besitzt folgende Werte:

Wert	Beschreibung
EdgesOnly	Der Listener wird nur bei einem Flankenwechsel benachrichtigt (gemäß des verwendeten Listener-Typen).
OnEnable	Der Listener wird bei einem Flankenwechsel benachrichtigt (gemäß des verwendeten Listener-Typen). Zusätzlich wird der Zustand der überwachten Bedingung bei der Aktivierung des Listeners geprüft. Je nach Listener-Typ wird der Listener bei folgenden Ereignissen benachrichtigt: <ul style="list-style-type: none"><li>■ IRisingEdgeListener: Nur wenn die Bedingung bei Aktivierung erfüllt ist</li><li>■ IFallingEdgeListener: Nur wenn die Bedingung bei Aktivierung nicht erfüllt ist</li><li>■ IAnyEdgeListener: Wenn die Bedingung bei Aktivierung erfüllt ist oder nicht erfüllt ist</li></ul>
MissedEvents	Der Listener wird bei einem Flankenwechsel benachrichtigt (gemäß des verwendeten Listener-Typen). Zusätzlich wird der Listener nach dem Ausführen der Behandlungsroutine benachrichtigt, wenn auslösende Ereignisse verpasst wurden. Das bedeutet: Tritt der auslösende Flankenwechsel beim Ausführen der Behandlungsroutine erneut auf, wird auch der Listener erneut benachrichtigt und die Behandlungsroutine ein zweites Mal ausgeführt.
All	Kombination aus OnEnable und MissedEvents Der Listener wird bei allen Ereignissen benachrichtigt, die unter OnEnable und MissedEvents beschrieben sind.

#### 15.23.4 Benachrichtigungsdienst für Listener aktivieren oder deaktivieren

<b>Beschreibung</b>	Die Methoden, um den Benachrichtigungsdienst zu aktivieren oder deaktivieren, gehören zur Klasse ConditionObserver.				
	Den Benachrichtigungsdienst zu aktivieren ist nur erforderlich, wenn zur Registrierung des Listeners die Methode createConditionObserver(...) verwendet wurde.				
<b>Syntax</b>	<p>Benachrichtigungsdienst aktivieren:</p> <pre>myObserver.enable()</pre> <p>Benachrichtigungsdienst deaktivieren:</p> <pre>myObserver.disable()</pre>				
<b>Erläuterung der Syntax</b>	<table border="1"> <thead> <tr> <th>Element</th><th>Beschreibung</th></tr> </thead> <tbody> <tr> <td><i>myObserver</i></td><td>Typ: ConditionObserver Objekt, das die definierte Bedingung überwacht</td></tr> </tbody> </table>	Element	Beschreibung	<i>myObserver</i>	Typ: ConditionObserver Objekt, das die definierte Bedingung überwacht
Element	Beschreibung				
<i>myObserver</i>	Typ: ConditionObserver Objekt, das die definierte Bedingung überwacht				

#### 15.23.5 Monitoring Programmierbeispiel

Um eine Kraftbedingung zu überwachen, wird ein Listener vom Typ IRisingEdgeListener definiert. Sobald am TCP eine Kraft von 35 N überschritten wird, gilt dies als Kollision. Der Listener wird benachrichtigt und eine Warnleuchte eingeschalten.

Als Benachrichtigungsart ist **NotificationType.MissedEvents** definiert. Falls die zulässige Kraft am TCP wiederholt überschritten wird, während die Warnleuchte eingeschalten ist, wird der Listener zeitnah darüber benachrichtigt.

```
ForceCondition collision = ForceCondition
    .createSpatialForceCondition(tool.getDefaultMotionFrame(), 35);

IRisingEdgeListener collisionListener = new IRisingEdgeListener() {

    @Override
    public void onRisingEdge(ConditionObserver conditionObserver,
        Date time, int missedEvents) {

        signals.setWarningLED(true);

    }
};

ConditionObserver collisionObserver = getObserverManager()
    .createConditionObserver(collision, NotificationType.MissedEvents,
        collisionListener);
collisionObserver.enable();
```

#### 15.24 Blockierendes Warten auf Bedingung

<b>Beschreibung</b>	Mit <code>waitFor(...)</code> wird eine Applikation so lange angehalten bis eine bestimmte Bedingung erfüllt ist oder eine bestimmte Wartezeit abgelaufen ist. Danach wird die Applikation fortgesetzt.
	 Bei der Bearbeitung des Befehls können Latenzzeiten auftreten. Es kann nicht garantiert werden, dass die programmierte Wartezeit genau eingehalten wird.

`waitFor(...)` muss auf den `ObserverManager` der Applikation zugreifen. Dieser wird mit `getObserverManager()` aufgerufen.

Es werden alle Bedingungstypen unterstützt außer `MotionPathCondition`.

Eine Übersicht der verfügbaren Bedingungstypen ist hier zu finden:  
 (>>> 15.20.1 "Bedingungen in der RoboticsAPI" Seite 341)

## Syntax

Blockierendes Warten ohne zeitliche Begrenzung:

```
getObserverManager().waitFor(condition)
```

Blockierendes Warten mit zeitlicher Begrenzung:

```
boolean result = getObserverManager().waitFor(condition, timeout, timeUnit)
```

## Erläuterung der Syntax

Element	Beschreibung
<i>condition</i>	Typ: <code>ICondition</code>  Bedingung, auf die gewartet wird. Wenn beim Aufruf von <code>waitFor(...)</code> die Bedingung bereits erfüllt ist, wird die Applikation sofort fortgesetzt.
<i>timeout</i>	Typ: <code>long</code>  Maximale Wartezeit. Tritt die Bedingung in der definierten Wartezeit nicht ein, wird die Applikation auch ohne Eintreten der Bedingung fortgesetzt.
<i>timeUnit</i>	Typ: <code>Enum</code> vom Typ <code>TimeUnit</code>  Einheit der angegebenen Wartezeit.  Das <code>Enum TimeUnit</code> ist Bestandteil der Java-Standardbibliothek.
<i>result</i>	Typ: <code>boolean</code>  Variable für den Rückgabewert von <code>waitFor(...)</code> . Der Rückgabewert ist <code>true</code> , wenn die Bedingung innerhalb der angegebenen Wartezeit eintritt.  <b>Hinweis:</b> Wird keine Wartezeit definiert, liefert <code>waitFor(...)</code> keinen Rückgabewert.

## Beispiel

In der Applikation soll auf ein boolesches Eingangssignal gewartet werden. Dabei soll die Applikation maximal 30 Sekunden blockiert sein. Wird das Eingangssignal in dieser Zeit nicht geliefert, soll im Anschluss eine bestimmte Behandlungsroutine ausgeführt werden

```
SwitchIOGroup inputs = new SwitchIOGroup(kuka_Sunrise_Cabinet);
Input input = inputs.getInput ("Input");

BooleanIOCondition inputCondition = new BooleanIOCondition(input,
true);

boolean result = getObserverManager().waitFor(inputCondition, 30,
TimeUnit.SECONDS);

if(!result){
    //do something
}
else{
    //continue program
}
```

## 15.25 Daten aufzeichnen und auswerten

Während der Ausführung einer Applikation können bestimmte Daten, z. B. externe Kräfte und Momente, für eine spätere Auswertung aufgezeichnet werden. Für die Programmierung der Datenaufzeichnung steht die Klasse DataRecorder (Paket: com.kuka.roboticsAPI.sensorModel) zur Verfügung.

Die aufgezeichneten Daten werden in einer Datei gespeichert und auf der Robotersteuerung im Verzeichnis C:\KRC\Roboter\Log\DataRecorder abgelegt.

Der Dateiname wird mit dem zu erstellenden DataRecorder-Objekt festgelegt. Wenn bei der Aufzeichnung ein Fehler aufgetreten ist, beginnt der Dateiname mit "FaultyDataRecorder...".

Die Datei kann mit einem Texteditor geöffnet oder in eine Excel-Tabelle eingelesen werden.

### 15.25.1 Objekt für Datenaufzeichnung erstellen

#### Beschreibung

Für die Datenaufzeichnung muss zunächst ein Objekt vom Typ DataRecorder erzeugt und parametrisiert werden. Wird hierfür der Standardkonstruktor verwendet, sind folgende Default-Parameter gesetzt:

- Der Dateiname, unter dem die aufgezeichneten Daten gespeichert werden, wird automatisch erzeugt und enthält eine intern vom System vergebene ID: DataRecorder/*ID.log*
- Es ist keine Aufzeichnungsdauer festgelegt. Es werden so lange Daten aufgezeichnet bis der Puffer (zur Zeit 16 MB) voll ist oder die maximale Anzahl an Datensätzen (zur Zeit 30 000) erreicht ist.
- Die Aufzeichnungsrate, d. h. der zeitliche Mindestabstand zwischen 2 Aufzeichnungen, beträgt 1 ms.

#### Konstruktor-Syntax

Die Klasse DataRecorder besitzt folgende Konuktoren:

`DataRecorder()` (Standardkonstruktor)

`DataRecorder(String fileName, long timeout, TimeUnit timeUnit, int sampleRate)`

#### Erläuterung der Syntax

Element	Beschreibung
<i>fileName</i>	Dateiname (mit Endung), unter dem die aufgezeichneten Daten gespeichert werden Beispiel: "Aufzeichnung_1.log"
<i>timeout</i>	Aufzeichnungsdauer <ul style="list-style-type: none"> <li>■ -1: Es ist keine Aufzeichnungsdauer festgelegt.</li> <li>■ ≥ 1</li> </ul> Default: -1 Die Zeiteinheit wird mit <i>timeUnit</i> festgelegt.
<i>timeUnit</i>	Zeiteinheit für die Aufzeichnungsdauer Beispiel: TimeUnit.SECONDS Das Enum TimeUnit ist Bestandteil der Java-Standardbibliothek.
<i>sampleRate</i>	Aufzeichnungsrate (Einheit: ms) <ul style="list-style-type: none"> <li>■ ≥ 1</li> </ul> Default: 1



Die Klasse DataRecorder bietet set-Methoden an, mit denen die Parameterwerte, insbesondere bei Verwendung des Standardkonstruktors, angepasst werden können.

- `setFileName(...), setSampleRate(...), setTimeout(..., ...)`

In `setTimeout(..., ...)` wird mit dem ersten Parameter die Aufzeichnungsduer und mit dem zweiten Parameter die zugehörige Zeiteinheit festgelegt.

### Beispiel 1

Es sollen 5 s lang Daten im 100 ms-Takt aufgezeichnet und in die Datei Recording\_1.log geschrieben werden.

```
DataRecorder rec_1 = new DataRecorder("Recording_1.log", 5,
TimeUnit.SECONDS, 100);
```

### Beispiel 2

Das DataRecorder-Objekt wird mit dem Standardkonstruktor erzeugt. Damit ist nur festgelegt, dass im 1 ms-Takt für eine unbestimmte Zeit Daten aufgezeichnet werden. Die aufgezeichneten Daten sollen in die Datei Recording\_2.log geschrieben werden. Dazu wird der Dateiname mit der zugehörigen set-Methode festgelegt.

```
DataRecorder rec_2 = new DataRecorder();
rec_2.setFileName("Recording_2.log");
```

## 15.25.2 Daten für Aufzeichnung festlegen

Die Daten, die aufgezeichnet werden sollen, werden mithilfe von Punktoperator und zugehöriger add-Methode an das hierfür erstellte DataRecorder-Objekt angefügt. Die gleichzeitige Aufzeichnung verschiedener Daten ist möglich.

### Übersicht

Folgende add-Methoden der Klasse DataRecorder stehen zur Verfügung:

Methode	Beschreibung
<code>addInternalJointTorque(...)</code>	Rückgabetyp: DataRecorder Aufzeichnung der gemessenen Achsmomente des als Parameter übergebenen Roboters (Typ: Robot)
<code>addExternalJointTorque(...)</code>	Rückgabetyp: DataRecorder Aufzeichnung der externen Achsmomente (modellbereinigt) des als Parameter übergebenen Roboters (Typ: Robot)
<code>addCartesianForce(...)</code>	Rückgabetyp: DataRecorder Aufzeichnung der kartesischen Kräfte entlang der X-, Y- und Z-Achse des als Parameter übergebenen Frames (Einheit: N). Die Varianz der kartesischen Kräfte wird ebenfalls aufgezeichnet. Optional kann ein zweiter Frame als Parameter übergeben werden, um die Orientierung für die Kraftmessung festzulegen. Wird für die Orientierung kein eigener Frame angegeben, muss <code>null</code> übergeben werden.

Methode	Beschreibung
addCartesianTorque(...)	<p>Rückgabetyp: DataRecorder</p> <p>Aufzeichnung der kartesischen Momente um die X-, Y- und Z-Achse des als Parameter übergebenen Frames (Einheit: Nm). Die Varianz der kartesischen Momente wird ebenfalls aufgezeichnet.</p> <p>Optional kann ein zweiter Frame als Parameter übergeben werden, um die Orientierung für die Momentenmessung festzulegen. Wird für die Orientierung kein eigener Frame angegeben, muss <code>null</code> übergeben werden.</p>
	<p>Parameter:</p> <ul style="list-style-type: none"> <li>■ <code>AbstractFrame measureFrame</code> Mit dem Roboterflansch verbundener Frame, z. B. der TCP eines Werkzeugs. Definiert die Position des Messpunkts.</li> <li>■ <code>AbstractFrame orientationFrame</code> Definiert die Orientierung des Messpunkts.</li> </ul> <p><b>Hinweis:</b> Es müssen immer beide Parameter übergeben werden. Die Orientierung darf <code>null</code> sein.</p>
addCommandedJointPosition(...)	<p>Rückgabetyp: DataRecorder</p> <p>Aufzeichnung der achsspezifischen Sollposition des als Parameter übergebenen Roboters (Typ: Robot). Als zweiter Parameter muss die Einheit übergeben werden, in der die Achswinkel aufgezeichnet werden (Enum vom Typ: AngleUnit).</p>
addCurrentJointPosition(...)	<p>Rückgabetyp: DataRecorder</p> <p>Aufzeichnung der achsspezifischen Istposition des als Parameter übergebenen Roboters (Typ: Robot). Als zweiter Parameter muss die Einheit übergeben werden, in der die Achswinkel aufgezeichnet werden (Enum vom Typ: AngleUnit).</p>
	<p>Parameter:</p> <ul style="list-style-type: none"> <li>■ <code>Robot robot</code></li> <li>■ <code>AngleUnit angleUnit</code> <ul style="list-style-type: none"> <li>■ <b>AngleUnit.Degree:</b> Achswinkel in Grad</li> <li>■ <b>AngleUnit.Radian:</b> Achswinkel in Rad</li> </ul> </li> </ul>
addCommandedCartesianPositionXYZ(...)	<p>Rückgabetyp: DataRecorder</p> <p>Aufzeichnung der kartesischen Sollposition (translatorischer Teil)</p> <p>Messpunkt und Referenz-Koordinatensystem, relativ zu dem die Position aufgezeichnet wird, werden als Parameter übergeben.</p>

Methode	Beschreibung
addCurrentCartesianPositionXYZ(...)	<p>Rückgabetyp: DataRecorder</p> <p>Aufzeichnung der kartesischen Istposition (translatorischer Teil)</p> <p>Messpunkt und Referenz-Koordinatensystem, relativ zu dem die Position aufgezeichnet wird, werden als Parameter übergeben.</p>

**Beispiel**

Für einen LBR iiwa sollen folgende Daten mithilfe eines DataRecorder-Objekts aufgezeichnet werden:

- Achsmomente, die am Roboter gemessen werden
- Kraft am TCP eines am Roboter montierten Greifers mit der Orientierung eines Basis-Frames

```
private LBR lbr_iiwa;
private Tool gripper;
...
gripper.attachTo(lbr_iiwa.getFlange());

DataRecorder rec = new DataRecorder();
rec.addInternalJointTorque(lbr_iiwa);
rec.addCartesianForce(gripper.getFrame("TCP"),
getApplicationData().getFrame("/Base"));
```

**15.25.3 Datenaufzeichnung starten**

Die Datenaufzeichnung kann bewegungsunabhängig gestartet werden (an jeder Stelle der Applikation möglich), oder bewegungssynchron, ausgelöst über einen Trigger.

**bewegungsunabhängig**

Vor dem Start einer bewegungsunabhängigen Aufzeichnung muss das DataRecorder-Objekt über die Methode enable() aktiviert werden. Über die Methode startRecording() wird die Aufzeichnung gestartet.

Wenn die Aufzeichnung beendet ist, wird das DataRecorder-Objekt automatisch deaktiviert. Wenn mit demselben DataRecorder-Objekt noch einmal Daten aufgezeichnet werden sollen, muss das DataRecorder-Objekt erneut aktiviert werden.



Es können nicht mehrere DataRecorder-Objekte gleichzeitig aktiviert sein.

**synchron über Trigger**

Für einen Trigger muss eine Bedingung vom Typ ICondition formuliert werden, bei deren Erfüllung der Trigger auslöst, sowie eine Aktion, die bei Auslösen des Triggers ausgeführt wird.

(>> 15.22.1 "Trigger programmieren" Seite 364)

Die auszuführende Aktion ist der Start der Datenaufzeichnung. Hierfür muss ein Objekt vom Typ StartRecordingAction übergeben werden. Beim Erstellen

des Objekts muss das DataRecorder-Objekt angegeben werden, das für die Datenaufzeichnung genutzt wird.

#### Konstruktor-Syntax:

```
StartRecordingAction(DataRecorder recorder)
```

Anschließend werden das ICondition-Objekt und das StartRecordingAction-Objekt mit triggerWhen(...) an einen Bewegungsbefehl geknüpft.

#### Beispiel 1

Die Datenaufzeichnung soll gestartet werden, wenn der Roboter die Anfahrbewegung zu einer Vorposition ausgeführt hat. Das DataRecorder-Objekt wird aktiviert, bevor die Vorposition angefahren wird, um die zeitliche Verzögerung beim Starten der Aufzeichnung zu reduzieren.

```
private LBR lbr_iwa;
...
DataRecorder rec = new DataRecorder();
...
rec.enable();
...
lbr_iwa.move(lingetApplicationData().getFrame("/Pre-position")));
rec.startRecording();
```

#### Beispiel 2

Die Datenaufzeichnung soll 2 s nach dem Start einer Bewegung beginnen. Dafür wird ein MotionPathCondition-Objekt parametriert.

```
private LBR lbr_iwa;
...
DataRecorder rec = new DataRecorder();
...
StartRecordingAction startAct = new StartRecordingAction(rec);
MotionPathCondition startCond = new
MotionPathCondition(ReferenceType.START, 0.0, 2000);
lbr_iwa.move(lingetApplicationData().getFrame("/
Destination")).triggerWhen(startCond, startAct));
```

#### 15.25.4 Datenaufzeichnung beenden

Die Datenaufzeichnung kann bewegungsunabhängig beendet werden (an jeder Stelle der Applikation möglich), oder bewegungssynchron, ausgelöst über einen Trigger.

Außerdem wird die Aufzeichnung automatisch beendet, wenn die Applikation beendet ist oder die im verwendeten DataRecorder-Objekt festgelegte Aufzeichnungsdauer erreicht ist.

#### bewegungsunabhängig

Die Aufzeichnung kann jederzeit über die Methode stopRecording() beendet werden.

#### synchron über Trigger

Für einen Trigger muss eine Bedingung vom Typ ICondition formuliert werden, bei deren Erfüllung der Trigger auslöst, sowie eine Aktion, die bei Auslösen des Triggers ausgeführt wird.

(>>> 15.22.1 "Trigger programmieren" Seite 364)

Die auszuführende Aktion ist der Stopp der Datenaufzeichnung. Hierfür muss ein Objekt vom Typ StopRecordingAction übergeben werden. Beim Erstellen des Objekts muss das DataRecorder-Objekt angegeben werden, das für die Datenaufzeichnung genutzt wird.

#### Konstruktor-Syntax:

```
StopRecordingAction(DataRecorder recorder)
```

Anschließend werden das ICondition-Objekt und das StopRecordingAction-Objekt mit triggerWhen(...) an einen Bewegungsbefehl geknüpft.

### 15.25.5 Zustände vom DataRecorder-Objekt abfragen

#### Übersicht

Folgende Methoden der Klasse DataRecorder stehen zur Verfügung:

Methode	Beschreibung
isEnabled()	Rückgabetyp: boolean Es wird abgefragt, ob das DataRecorder-Objekt aktiviert ist (= true).
isRecording()	Rückgabetyp: boolean Es wird abgefragt, ob die Datenaufzeichnung läuft (= true).
isFileAvailable()	Rückgabetyp: boolean Es wird abgefragt, ob die Datei mit den aufgezeichneten Daten bereits auf der Robotersteuerung gespeichert und für die Auswertung verfügbar ist (= true).
awaitFileAvailable(...)	Rückgabetyp: boolean Blockiert die aufrufende Applikation oder Hintergrund-Task maximal so lange bis die definierte Blockierungszeit abgelaufen ist oder bis die Datei mit den aufgezeichneten Daten auf der Robotersteuerung gespeichert und für die Auswertung verfügbar ist (= true).  Die blockierende Anweisung liefert false zurück, wenn die Datei innerhalb der maximalen Blockierungszeit nicht verfügbar wird.  Syntax: <ul style="list-style-type: none"> <li>■ <code>awaitFileAvailable(long timeout, java.util.concurrent.TimeUnit timeUnit)</code></li> </ul> Parameter: <ul style="list-style-type: none"> <li>■ <code>timeout</code>: maximale Blockierungszeit</li> <li>■ <code>timeUnit</code>: Zeiteinheit für maximale Blockierungszeit</li> </ul>

### 15.25.6 Beispielprogramm für Datenaufzeichnung

Während eines Montageprozesses sollen die extern auf die Achsen eines LBR iiwa wirkenden Momente aufgezeichnet werden, außerdem die kartesischen Kräfte, die auf den TCP eines am Roboterflansch montierten Greifers wirken. Die Daten sollen alle 10 ms aufgezeichnet werden.

Die Aufzeichnung soll bewegungssynchron beginnen, wenn die aus beliebiger Richtung auf den TCP des Greifers wirkende Kraft 20 N übersteigt. Wenn der Montageprozess beendet ist, soll auch die Aufzeichnung beendet werden.

Im Anschluss soll die Datei ausgewertet werden, wenn sie nach maximal 5 s zur Verfügung steht.

```

private LBR lbr_iiwa;
private Tool gripper;
...
gripper.attachTo(lbr_iiwa.getFlange());
...
DataRecorder rec = new DataRecorder();
rec.setFileName("Recording.log");
rec.setSampleRate(10);

rec.addExternalJointTorque(lbr_iiwa);
rec.addCartesianForce(gripper.getFrame("/TCP"), null);

```

```

StartRecordingAction startAction = new StartRecordingAction(rec);
ForceCondition startCondition =
ForceCondition.createSpatialForceCondition(gripper.getFrame("/TCP"),
20.0);

lbr_iwa.move(ptpgetApplicationData().getFrame("/StartPosition")));
lbr_iwa.move(lingetApplicationData().getFrame("/")
MountingPosition")).triggerWhen(startCondition, startAction);
lbr_iwa.move(lingetApplicationData().getFrame("/DonePosition")));

rec.stopRecording();

if (rec.awaitFileEnable(5, TimeUnit.SECONDS)) {
// Evaluation of the file if available
}

```

## 15.26 Benutzertasten definieren

### Beschreibung

Die 4 Benutzertasten auf dem smartPAD können frei mit Funktionen belegt werden. Hierzu können im Quellcode von Roboter-Applikationen oder Hintergrund-Tasks verschiedene Benutzertasten-Leisten definiert werden.

Über die Benutzertasten-Leiste wird den Benutzertasten ihre jeweilige Funktion zugeordnet. Es müssen nicht alle Benutzertasten der Leiste mit einer Funktion belegt werden, mindestens jedoch eine. Zusätzlich wird dem Feld auf dem Seitenrahmen der smartHMI neben der Benutzertaste ein grafisches Element oder ein Textelement zugeordnet, um die Funktion einer Benutzertaste zu verdeutlichen.

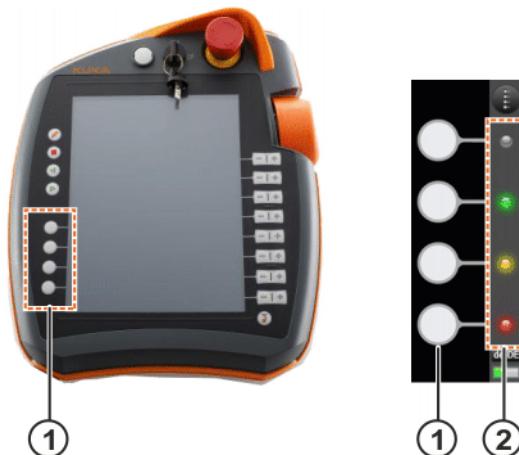


Abb. 15-18: Benutzertasten am smartPAD (Beispiel)

1 Benutzertasten

2 Leiste mit LED-Icons

Dem Bediener stehen alle Benutzertasten-Leisten zur Verfügung, die in der laufenden Roboter-Applikation oder im laufenden Hintergrund-Task definiert sind. Beispielsweise kann eine Benutzertasten-Leiste zur Ansteuerung eines Greifers angeboten werden, während über eine andere Leiste die Benutzertasten genutzt werden, um unterschiedliche Programmabschnitte anzuwählen.

Benutzertasten-Leisten sind verfügbar bis die Roboter-Applikation oder der Hintergrund-Task, der sie erzeugt hat, beendet ist.

### Übersicht

Um eine Benutzertasten-Leiste zu programmieren, sind folgende Schritte erforderlich:

Schritt	Beschreibung
1	Benutzertasten-Leiste erstellen. (>>> 15.26.1 "Benutzertasten-Leiste erstellen" Seite 381)
2	Benutzertasten zur Leiste hinzufügen (mindestens eine). (>>> 15.26.2 "Benutzertasten zur Leiste hinzufügen" Seite 382)
3	Funktion festlegen, die beim Betätigen der Benutzertaste ausgeführt werden soll. (>>> 15.26.3 "Funktion einer Benutzertaste festlegen" Seite 384)
4	Dem Feld auf dem Seitenrahmen der smartHMI neben der Benutzertaste jeweils mindestens ein grafisches Element oder ein Textelement zuweisen. (>>> 15.26.4 "Beschriftung und grafische Gestaltung der Benutzertasten-Leiste" Seite 386)
5	Für Benutzertasten, die Funktionen auslösen, die mit einem Risiko verbunden sind: Warnmeldung definieren, die beim Betätigen der Benutzertaste angezeigt wird, bevor die Funktion ausgelöst werden kann. (>>> 15.26.5 "Sicherheitskritische Benutzertasten kennzeichnen" Seite 389)
6	Benutzertasten-Leiste veröffentlichen. (>>> 15.26.6 "Benutzertasten-Leiste veröffentlichen" Seite 390)

### 15.26.1 Benutzertasten-Leiste erstellen

#### Beschreibung

Um eine Benutzertasten-Leiste zu erstellen, werden folgende Methoden benötigt:

- `getApplicationUI()`  
Mit dieser Methode kann aus einer Roboter-Applikation oder einem Hintergrund-Task heraus auf die Schnittstelle zur Bedienoberfläche smartHMI zugegriffen werden. Rückgabetyp: ITaskUI
- `createUserKeyBar(...)`  
Mit dieser Methode wird die Benutzertasten-Leiste erzeugt. Sie gehört zur Schnittstelle ITaskUI.

#### Syntax

```
IUserKeyBar keybar =
getApplicationUI().createUserKeyBar("name");
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>keybar</code>	Typ: IUserKeyBar  Name der mit <code>createUserKeyBar(...)</code> erzeugten Benutzertasten-Leiste
<code>name</code>	Typ: String  Name, unter dem die Benutzertasten-Leiste auf der smartHMI angezeigt wird (>>> Abb. 6-16 )  Die darstellbare Zeichenanzahl ist begrenzt: <ul style="list-style-type: none"><li>■ Maximal 12 bis 15 Zeichen empfohlen.</li></ul>

**Beispiel**

Es wird eine Benutzertasten-Leiste zur Ansteuerung eines Greifers erstellt.

```
IUserKeyBar gripperBar =  
getApplicationUI().createUserKeyBar("Gripper");
```

### 15.26.2 Benutzertasten zur Leiste hinzufügen

**Beschreibung**

Eine neu erzeugte Benutzertasten-Leiste besitzt zunächst keine Benutzertasten. Die Benutzertasten, die genutzt werden sollen, müssen zur Leiste hinzugefügt werden.

Die Schnittstelle IUserKeyBar stellt hierfür folgende Methoden zur Verfügung:

- **addUserKey(...)**

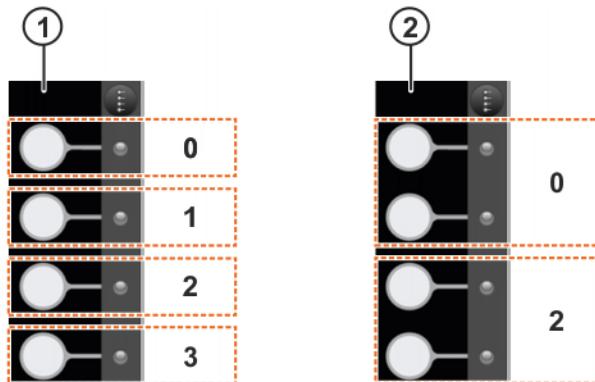
Fügt eine einzelne Benutzertaste zur Leiste hinzu.

- **addDoubleUserKey(...)**

Fasst 2 benachbarte Benutzertasten zu einer Doppeltaste zusammen und fügt sie zur Leiste hinzu. Die zugehörigen Felder auf dem Seitenrahmen der smartHMI werden ebenfalls zu einem gemeinsamen größeren Feld verbunden.

Beim Hinzufügen einer Benutzertaste zu einer Leiste wird die Funktion festgelegt, die beim Betätigen der Benutzertaste ausgeführt wird (z. B. Öffnen eines Greifers, Änderung eines Parameters, etc.). Je nach Programmierung kann sowohl das Drücken als auch das Loslassen der Benutzertaste als Betätigung gelten und mit einer Funktion verknüpft sein.

Eine Benutzertasten-Leiste muss mindestens eine Benutzertaste enthalten. Jeder Benutzertaste ist eine eindeutige Nummer zugeordnet. Diese Nummer wird beim Hinzufügen einer Benutzertaste übergeben.



**Abb. 15-19: Nummerierung der Benutzertasten**

1 Einzeltasten

2 Doppeltasten

**Syntax**

Einzeltaste hinzufügen:

```
IUserKey key = keybar.addUserKey(slot, listener, ignoreEvents);
```

Doppeltaste hinzufügen:

```
IUserKey doubleKey = keybar.addDoubleUserKey(slot, listener, ignoreEvents);
```

## Erläuterung der Syntax

Element	Beschreibung
<i>keybar</i>	Typ: IUserKeyBar Name der Benutzertasten-Leiste, zu der eine Benutztaste hinzugefügt wird
<i>key</i>	Typ: IUserKey Name der Einzeltaste, die zur Leiste hinzugefügt wird
<i>doubleKey</i>	Typ: IUserKey Name der Doppeltaste, die zur Leiste hinzugefügt wird
<i>slot</i>	Typ: int Nummer der Benutzertaste, die hinzugefügt wird Einzeltasten: <ul style="list-style-type: none"><li>■ <b>0 ... 3</b></li></ul> Doppeltasten: <ul style="list-style-type: none"><li>■ <b>0, 2</b></li></ul>
<i>listener</i>	Typ: IUserKeyListener Name des Listener-Objekts, mit dem die Funktion festgelegt wurde, die beim Betätigen der Benutzertaste ausgeführt wird  (>>> 15.26.3 "Funktion einer Benutzertaste festlegen" Seite 384)
<i>ignoreEvents</i>	Typ: boolean Legt fest, ob darauf reagiert wird, wenn die Benutzertaste während des Ausführens der Tastenfunktion erneut betätigt wird <ul style="list-style-type: none"><li>■ <b>true:</b> Wenn die Taste während des Ausführens der Funktion betätigt wird, hat dies keine Auswirkung.</li><li>■ <b>false:</b> Es wird gezählt, wie oft die Taste während des Ausführens der Funktion betätigt wird. Die Funktion wird entsprechend oft wiederholt.</li></ul>

## Beispiel

Den Benutzertasten werden folgende Funktionen zur Ansteuerung eines Greifers zugeordnet:

- Die oberste Benutzertaste wird zum Öffnen und die darunterliegende Benutzertaste zum Schließen des Greifers verwendet.
- Die beiden unteren Benutzertasten werden zu einer Doppeltaste zusammengefasst. Über sie soll die Geschwindigkeit des Greifers erhöht und verringert werden können.
- Die Funktionen zum Öffnen und Schließen des Greifers sollen erst dann erneut aufgerufen werden können, wenn die jeweilige Funktion beendet ist.

```
IUserKeyBar gripperBar =
getApplicationUI().createUserKeyBar("Gripper");

IUserKeyListener openGripperListener = ...;
IUserKeyListener closeGripperListener = ...;
IUserKeyListener gripperVelocityListener = ...;

IUserKey openKey = gripperBar.addUserKey(0,
    openGripperListener, true);
IUserKey closeKey = gripperBar.addUserKey(1,
    closeGripperListener, true);
```

```
IUserKey velocityKey = gripperBar.addDoubleUserKey(2,  
    gripperVelocityListener, false);
```

### 15.26.3 Funktion einer Benutzertaste festlegen

#### Beschreibung

Um festzulegen, welche Funktion beim Betätigen einer Benutzertaste ausgeführt werden soll, muss ein Listener-Objekt vom Typ IUserKeyListener erzeugt werden. Bei der Erzeugung des Objekts wird automatisch die Methode onKeyEvent(...) deklariert.

Die Listener-Methode onKeyEvent(...) wird bei folgenden Ereignissen aufgerufen:

- Der Benutzertaste wird gedrückt.
- Der Benutzertaste wird losgelassen.



Auch wenn unterschiedliche Listener verwendet werden, kann immer nur ein OnKeyEvent(...) ausgeführt werden. Löst der Bediener beispielsweise das OnKeyEvent(...) von Benutzertaste 2 aus während noch das OnKeyEvent(...) von Benutzertaste 1 ausgeführt wird, startet das zweite OnKeyEvent(...) erst, wenn das erste abgeschlossen ist.

#### Syntax

```
IUserKeyListener listener = new IUserKeyListener() {  
    @Override  
    public void onKeyEvent(IUserKey key, UserKeyEvent event) {  
        // Reaktion auf Ereignis  
    }  
};
```

#### Erläuterung der Syntax

Element	Beschreibung
<i>listener</i>	Typ: IUserKeyListener Name des Listener-Objekts
Eingangsparameter der Listener-Methode onKeyEvent(...):	

Element	Beschreibung
key	<p>Typ: IUserKey</p> <p>Benutzertaste, die betätigt wurde</p> <p>Über den Parameter kann direkt auf die Benutzertaste zugegriffen werden, um beispielsweise die zugehörige Beschriftung oder grafische Gestaltung zu ändern. Außerdem kann, insbesondere bei Verwendung der gleichen Reaktion für unterschiedliche Benutzertasten, bestimmt werden, welche Benutzertaste betätigt wurde.</p>
event	<p>Typ: Enum vom Typ UserKeyEvent</p> <p>Ereignis, das die Listener-Methode onKeyEvent(...) aufgerufen hat</p> <p>Enum-Werte für Einzeltasten:</p> <ul style="list-style-type: none"> <li>■ <b>UserKeyEvent.KeyDown:</b> Taste wurde gedrückt.</li> <li>■ <b>UserKeyEvent.KeyUp:</b> Taste wurde losgelassen.</li> </ul> <p>Enum-Werte für Doppeltasten:</p> <ul style="list-style-type: none"> <li>■ <b>UserKeyEvent.FirstKeyDown:</b> Die obere der beiden Tasten wurde gedrückt.</li> <li>■ <b>UserKeyEvent.SecondKeyDown:</b> Die untere der beiden Tasten wurde gedrückt.</li> <li>■ <b>UserKeyEvent.FirstKeyUp:</b> Die obere der beiden Tasten wurde losgelassen.</li> <li>■ <b>UserKeyEvent.SecondKeyUp:</b> Die untere der beiden Tasten wurde losgelassen.</li> </ul>

## Beispiel

Die Benutzertasten-Leiste zur Ansteuerung eines Greifers wird um eine Methode erweitert, durch die die Geschwindigkeit des Greifers angepasst werden kann. Hierfür werden die beiden unteren zu einer Doppeltaste zusammengefassten Benutzertasten verwendet.

Zum Einstellen der Geschwindigkeit wird das Attribut `velocity` deklariert, das die aktuelle Geschwindigkeit als Anteil der Maximalgeschwindigkeit angibt (Wertebereich: 0.1 ... 1.0). Beim Drücken der oberen Benutzertaste wird der Wert um 0.1 erhöht, beim Drücken der unteren Benutzertaste um 0.1 verringert.

```

final double velocity = 0.1;
...
IUserKeyBar gripperBar = ...;
...
IUserKeyListener gripperVelocityListener = new IUserKeyListener() {
    @Override
    public void onKeyEvent(IUserKey key, IUserKeyEvent event) {
        if(event == UserKeyEvent.FirstKeyDown && velocity <= 0.9){
            velocity = velocity + 0.1;
        }
        else if(event == UserKeyEvent.SecondKeyDown && velocity >= 0.2){
            velocity = velocity - 0.1;
        }
    }
};
...
IUserKey velocityKey = gripperBar.addDoubleUserKey(2,
    gripperVelocityListener, false);

```

## 15.26.4 Beschriftung und grafische Gestaltung der Benutzertasten-Leiste

### Beschreibung

Dem Feld auf dem Seitenrahmen der smartHMI neben der Benutzertaste muss jeweils mindestens ein grafisches Element oder ein Textelement zugewiesen werden. Als grafische Elemente stehen LED-Icons unterschiedlicher Farbe und Größe zur Verfügung. Diese Elemente können während der Laufzeit der Roboter-Applikation oder des Hintergrund-Tasks angepasst werden.

Um die einzelnen Elemente eindeutig platzieren zu können, wird das Feld neben der Benutzertaste in ein Raster mit 3x3 Plätzen eingeteilt. Dies gilt auch für Benutzertasten, die zu einer Doppeltaste zusammengefasst wurden, wobei sich das Raster in diesem Fall über beide Felder erstreckt.

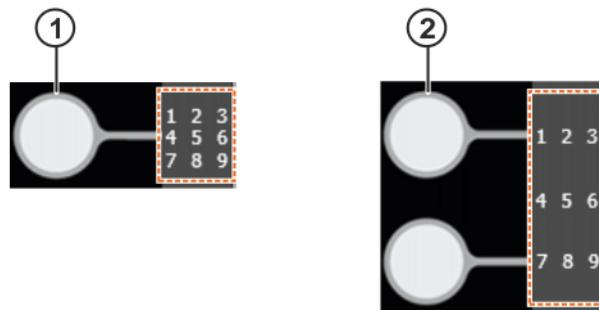


Abb. 15-20: Aufteilung des Rasters

1 Einzeltasten

2 Doppeltasten

Auf jeden Platz kann maximal ein Element gesetzt werden. Dieser Platz wird durch die Werte des Enums UserKeyAlignment festgelegt. Wird einem bereits belegten Platz ein neues Element zugewiesen, wird das vorhandene gelöscht.

### UserKey-Alignment

Platz-Nr.	Wert
1	UserKeyAlignment.TopLeft
2	UserKeyAlignment.TopMiddle
3	UserKeyAlignment.TopRight
4	UserKeyAlignment.MiddleLeft
5	UserKeyAlignment.Middle
6	UserKeyAlignment.MiddleRight
7	UserKeyAlignment.BottomLeft
8	UserKeyAlignment.BottomMiddle
9	UserKeyAlignment.BottomRight

### 15.26.4.1 Textelement zuweisen

#### Beschreibung

Jedem Rasterplatz kann ein Textelement zugewiesen werden. Hierfür wird die Methode setText(...) verwendet. Die Methode gehört zur Schnittstelle IUserKey.

#### Syntax

```
key.setText(position, "text");
```

## Erläuterung der Syntax

Element	Beschreibung
<i>key</i>	Typ: IUserKey Benutzertaste, der ein Textelement zugewiesen wird
<i>position</i>	Typ: Enum vom Typ UserKeyAlignment Position des Elements (Rasterplatz) (>>> "UserKeyAlignment" Seite 386)
<i>text</i>	Typ: String Text, der angezeigt werden soll  Häufig übersteigt bereits bei 2 oder wenig mehr Zeichen die Textlänge die Größe des Rasterplatzes. Der Anzeigebereich des Textes wird dann erweitert, aber es ist dennoch nur eine begrenzte Zeichenanzahl sinnvoll darstellbar. Dies ist abhängig von den Textelementen der benachbarten Rasterplätze sowie den verwendeten Zeichen.

## Beispiel

Die Benutzertasten-Leiste zur Ansteuerung eines Greifers wird erweitert. Neben jeder der Benutzertasten soll eine passende Beschriftung dauerhaft angezeigt werden.

- Beschriftung für die Benutzertasten zum Öffnen und Schließen des Greifers: OPEN und CLOSE
  - Beschriftung für die Benutzertasten zum Erhöhen und Verringern der Greifergeschwindigkeit: Pluszeichen und Minuszeichen
- Zusätzlich soll die aktuelle Geschwindigkeit angezeigt und bei jeder Änderung automatisch aktualisiert werden.

```
final double velocity = 0.1;
...
IUserKeyBar gripperBar = ...;
...
IUserKeyListener gripperVelocityListener = new IUserKeyListener() {
    @Override
    public void onKeyEvent(IUserKey key, IUserKeyEvent event){
        if(event == UserKeyEvent.FirstKeyDown && velocity <= 0.9){
            velocity = velocity + 0.1;
        }
        else if(event == UserKeyEvent.SecondKeyDown && velocity >= 0.2){
            velocity = velocity - 0.1;
        }
        // Folgende Zeile formatiert die Anzeige der Geschwindigkeit
        // Anzeige der ersten 3 Zeichen
        String value = String.valueOf(velocity).substring(0, 3);
        key.setText(UserKeyAlignment.Middle, value);
    }
};
IUserKey openKey = ...;
openKey.setText(UserKeyAlignment.TopLeft, "OPEN");

IUserKey closeKey = ...;
closeKey.setText(UserKeyAlignment.TopLeft, "CLOSE");

IUserKey velocityKey = ...;
velocityKey.setText(UserKeyAlignment.TopMiddle, "+");
velocitykey.setText(UserKeyAlignment.Middle,
```

```
Double.toString(velocity));
velocityKey.setText(UserKeyAlignment.BottomMiddle, "-");
```

#### 15.26.4.2 LED-Icon zuweisen

##### Beschreibung

Jedem Rasterplatz kann ein LED-Icon zugewiesen werden. Hierfür wird die Methode setLED(...) verwendet. Die Methode gehört zur Schnittstelle IUserKey.

##### Syntax

```
key.setLED(position, led, size);
```

##### Erläuterung der Syntax

Element	Beschreibung
key	Typ: IUserKey Benutzertaste, der ein grafisches Element zugewiesen wird
position	Typ: Enum vom Typ UserKeyAlignment Position des Elements (Rasterplatz) (>>> "UserKeyAlignment" Seite 386)
led	Typ: Enum vom Typ UserKeyLED Farbe des LED-Icons <ul style="list-style-type: none"> <li>■ <b>UserKeyLED.Grey:</b> Grau</li> <li>■ <b>UserKeyLED.Green:</b> Grün</li> <li>■ <b>UserKeyLED.Yellow:</b> Gelb</li> <li>■ <b>UserKeyLED.Red:</b> Rot</li> </ul>
size	Typ: Enum vom Typ UserKeyLEDSize Größe des LED-Icons <ul style="list-style-type: none"> <li>■ <b>UserKeyLEDSize.Small:</b> Klein</li> <li>■ <b>UserKeyLEDSize.Normal:</b> Groß</li> </ul>

##### Beispiel

Die Benutzertasten-Leiste zur Ansteuerung eines Greifers wird erweitert. Den Benutzertasten zum Öffnen und Schließen des Greifers soll jeweils ein LED-Icon kleiner Größe zugewiesen werden.

Solange sich der Greifer öffnet oder schließt, sollen die LED-Icon in grüner Farbe angezeigt werden. Steht der Greifer, sollen die LED-Icon in grauer Farbe angezeigt werden.

```
IUserKeyBar gripperBar =
getApplicationUI().createUserKeyBar("Gripper");

IUserKeyListener openGripperListener = new IUserKeyListener() {
@Override
public void onKeyEvent(IUserKey key, UserKeyEvent event) {
    key.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Green,
        UserKeyLEDSize.Small);
    openGripper(); // Methode zum Öffnen des Greifers
    key.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Grey,
        UserKeyLEDSize.Small);
}
};

IUserKeyListener openGripperListener = new IUserKeyListener() {
@Override
public void onKeyEvent(IUserKey key, UserKeyEvent event) {
    key.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Green,
```

```

        UserKeyLEDSIZE.Small);
closeGripper(); // Methode zum Schließen des Greifers
key.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Grey,
        UserKeyLEDSIZE.Small);
}

;

IUserKeyListener gripperVelocityListener = ...;
...
IUserKey openKey = ...;
openKey.setText...;
openKey.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Grey,
        UserKeyLEDSIZE.Small);

IUserKey closeKey = ...;
closeKey.setText...;
closeKey.setLED(UserKeyAlignment.BottomMiddle, UserKeyLED.Grey,
        UserKeyLEDSIZE.Small);

IUserKey velocityKey = ...

```

### 15.26.5 Sicherheitskritische Benutzertasten kennzeichnen

#### Beschreibung

Benutzertasten können Funktionen auslösen, die mit einem Risiko verbunden sind. Um zu verhindern, dass durch ein unbeabsichtigtes Betätigen solcher Benutzertasten Schaden entsteht, können sie mit einer Warnmeldung versehen und auf diese Weise als sicherheitskritisch gekennzeichnet werden. Hierfür wird die Methode `setCriticalText(...)` verwendet. Die Methode gehört zur Schnittstelle `IUserKey`.

Betätigt der Bediener eine als sicherheitskritisch gekennzeichnete Benutzertaste, wird die mit `setCriticalText(...)` definierte Meldung in einem Fenster mit dem Namen **Kritische Operation** auf der smartHMI angezeigt. Danach ist die Benutzertaste für ca. 5 s deaktiviert. Nach Ablauf dieser Zeit kann der Bediener die gewünschte Funktion auslösen, indem er die Benutzertaste innerhalb von 5 s erneut betätigt.

Wird die Benutzertaste in dieser Zeit nicht betätigt oder wird ein Bereich außerhalb des Fensters **Kritische Operation** berührt, wird das Fenster geschlossen und die Benutzertaste auf den vorherigen Zustand zurückgesetzt.

#### Syntax

```
key.setCriticalText("text");
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>key</code>	Typ: <code>IUserKey</code> Benutzertaste, die mit einer Warnmeldung versehen wird
<code>text</code>	Typ: <code>String</code> Meldungstext, der beim Betätigen der Benutzertaste angezeigt wird

#### Beispiel

Die Benutzertasten-Leiste zur Ansteuerung eines Greifers wird erweitert. Wird die Benutzertaste zum Öffnen des Greifers betätigt, soll eine Warnmeldung angezeigt werden. Der Bediener wird aufgefordert sicherzustellen, dass beim Öffnen des Greifers kein Schaden durch herausfallende Werkstücke entstehen kann.

```

IUserKeyBar gripperBar =
getApplicationUI().createUserKeyBar("Gripper");
...

```

```
IUserKey openKey = ...;
openKey.setText...;
openKey.setLED...;
openKey.setCriticalText("Greifer öffnet sich bei erneuter Betätigung.
Stellen Sie sicher, dass kein Schaden durch herausfallende Werkstücke
entsteht!");
```

### 15.26.6 Benutzertasten-Leiste veröffentlichen

#### Beschreibung

Wenn eine Benutzertasten-Leiste mit allen benötigten Benutzertasten und Funktionalitäten ausgestattet ist, muss sie über Methode publish() veröffentlicht werden. Erst danach steht sie dem Bediener am smartPAD zur Verfügung.

Wenn eine Benutzertasten-Leiste veröffentlicht ist, können zu einem späteren Zeitpunkt im Programmablauf keine weiteren Benutzertasten hinzugefügt werden. D. h. es ist nicht möglich, eine unbelegt gebliebene Benutzertaste nachträglich hinzuzufügen und mit einer Funktion zu versehen. Im Gegensatz dazu kann die Beschriftung oder Grafik, die auf der smartHMI neben der Benutztaste angezeigt wird, nachträglich geändert werden.

#### Syntax

```
keybar.publish();
```

#### Erläuterung der Syntax

Element	Beschreibung
keybar	Typ: IUserKeyBar Name der mit createUserKeyBar(...) erzeugten Benutzertasten-Leiste.

#### Beispiel

Die zur Ansteuerung eines Greifers erstellte Benutzertasten-Leiste wird veröffentlicht.

```
IUserKeyBar gripperBar =
getApplicationUI().createUserKeyBar("Gripper");
...
gripperBar.publish();
```

## 15.27 Meldungsprogrammierung

### 15.27.1 Benutzermeldungen programmieren

#### Beschreibung

Es können Hinweis-, Warn- und Fehlermeldungen programmiert werden, die beim Ausführen der Applikation auf der smartHMI ausgegeben und in die Log-Datei der Applikation geschrieben werden. Außerdem sind Meldungen programmierbar, die nicht auf der smartHMI ausgegeben, sondern nur in die Log-Datei geschrieben werden.



Es wird empfohlen, nur Meldungen auf der smartHMI auszugeben, die unbedingt notwendig sind. Eine zu intensive Nutzung der Meldungsausgabe kann sich negativ auf die Laufzeit der Applikation und die Bedienung der smartHMI auswirken.



Es wird empfohlen, nur die hier beschriebenen Befehle für die Meldungsausgabe zu verwenden und nicht andere Logging-Funktionalitäten, wie z. B. die Java-Befehle System.out.println(...) oder System.err.println(...) zu nutzen. Bei der Verwendung dieser Befehle kann nicht garantiert werden, dass die Meldung auf der smartHMI angezeigt wird.

#### Syntax

Hinweismeldung:

```

getLogger().info ("Meldungstext") ;

Warnmeldung:
getLogger().warn ("Meldungstext") ;

Fehlernmeldung:
getLogger().error ("Meldungstext") ;

Meldung, die nur in Log-Datei geschrieben wird:
getLogger().fine ("Meldungstext") ;

```

### Erläuterung der Syntax

Element	Beschreibung
Meldungstext	Text, der auf der smartHMI ausgegeben und/oder in Log-Datei geschrieben werden soll

### Beispiel

Nachdem der Roboter einen Zielpunkt erreicht hat, soll eine Hinweismeldung ausgegeben werden. Wurde die Bewegung durch eine Kollision beendet, wird stattdessen ein Warnhinweis ausgegeben.

```

IMotionContainer motion = lbr.move(lin(getFrame ("/P20"))
.breakWhen(collision));

if(motion.getFiredBreakConditionInfo() == null) {
    getLogger().info("End point reached.");
}
else{
    getLogger().warn("Motion canceled after collision!");
}

```

## 15.27.2 Benutzerdialoge programmieren

### Beschreibung

In einer Applikation können Benutzerdialoge programmiert werden. Diese Benutzerdialoge werden beim Ausführen der Applikation auf der smartHMI in einem Dialogfenster angezeigt und erfordern eine Reaktion des Benutzers.

Über die Methode `displayModalDialog(...)` können verschiedene Dialogtypen programmiert werden. Auf der smartHMI werden je nach Typ folgende Symbole angezeigt:

Symbol	Typ
	INFORMATION Dialog mit einem Hinweis, den der Benutzer zur Kenntnis nehmen muss
	QUESTION Dialog mit einer Frage, die der Benutzer beantworten muss
	WARNING Dialog mit einer Warnung, die der Benutzer zur Kenntnis nehmen muss
	ERROR Dialog mit einer Fehlernmeldung, die der Benutzer zur Kenntnis nehmen muss

Der Benutzer wählt die Antwort über einen Button, dessen Beschriftung der Programmierer definiert. Es können bis zu 12 Buttons definiert werden.

Die Applikation oder der Hintergrund-Task, aus dem der Dialog aufgerufen wurde, wird so lange angehalten bis der Benutzer reagiert hat. Der weitere Programmablauf kann davon abhängig gemacht werden, welchen Button der

Benutzer wählt. Die Methode `displayModalDialog(...)` liefert den Index des Buttons zurück, den der Benutzer auf der smartHMI wählt. Der Index beginnt bei "0" (= Index des ersten Buttons).

<b>Syntax</b>	<code>getApplicationUI().displayModalDialog (Dialogtyp, "Dialogtext", "Button_1" &lt;, ... "Button_12" &gt;)</code>
---------------	---

Erläuterung der Syntax	Element	Beschreibung
	<i>Dialogtyp</i>	Typ: Enum vom Typ <code>ApplicationDialogType</code> <ul style="list-style-type: none"> <li>■ INFORMATION: Dialog mit dem Symbol für einen Hinweis wird angezeigt.</li> <li>■ QUESTION: Dialog mit dem Symbol für eine Frage wird angezeigt.</li> <li>■ WARNING: Dialog mit dem Symbol für eine Warnung wird angezeigt.</li> <li>■ ERROR: Dialog mit dem Symbol für einen Fehler wird angezeigt.</li> </ul>
	<i>Dialogtext</i>	Typ: String Text, der auf der smartHMI im Dialogfenster angezeigt wird
	<i>Button_1</i> ... <i>Button_12</i>	Typ: String Beschriftung der Buttons 1 ... 12 (auf der smartHMI von links nach rechts zu sehen)

### Beispiel

Der folgende Benutzerdialog vom Typ QUESTION soll auf der smartHMI angezeigt werden:

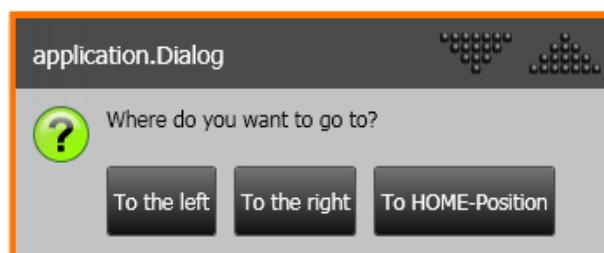


Abb. 15-21: Beispiel eines Benutzerdialogs

```
int direction = getApplicationUI().displayModalDialog(
    ApplicationDialogType.QUESTION, "Where do you want to go to?", "To
the left", "To the right", "To HOME-Position");

switch (direction) {
    case 0:
        lbr.move(ptp(getApplicationData().getFrame("/Left")));
        break;
    case 1:
        lbr.move(ptp(getApplicationData().getFrame("/Right")));
        break;
    case 2:
        lbr.move(ptpHome());
        break;
}
```

## 15.28 Programmablaufkontrolle

### 15.28.1 Applikation pausieren

#### Beschreibung

Eine Applikation kann über die Methode halt() pausiert werden.

Durch halt() wird die aktuell ausgeführte Bewegung pausiert und der Applikationszustand auf der smartHMI wechselt zu **Bewegung pausiert**.

halt() verursacht ein blockierendes Anhalten des aufrufenden Threads. Laufen parallel dazu weitere Threads, werden diese weiterhin ausgeführt. Die Applikationsausführung wird nur dann angehalten, wenn halt() im Applikations-Thread aufgerufen wird. Es wird daher empfohlen, halt() nicht in Behandlungsroutinen für bahnbezogene Schaltaktionen oder in Behandlungsroutinen zum Überwachen von Prozessen aufzurufen. Stattdessen wird empfohlen, in diesen Behandlungsroutinen die Methode pause() zu verwenden.

(>>> 15.28.2 "Bewegungsausführung pausieren" Seite 393)

Ein Fortsetzen der Bewegung und des pausierten Threads ist nur über die Start-Taste am smartPAD möglich. Durch Drücken der Start-Taste wird die pausierte Bewegung fortgesetzt. Der pausierte Thread wird mit der im Quellcode auf halt() folgenden Anweisung fortgeführt.

#### Syntax

```
getApplicationControl().halt();
```

### 15.28.2 Bewegungsausführung pausieren

#### Beschreibung

Die Bewegungsausführung kann über die Methode pause() pausiert werden.

Das Verhalten entspricht dem Pausieren der Applikation über das smartPAD. Durch pause() wird die aktuell ausgeführte Bewegung pausiert und der Applikationszustand auf der smartHMI wechselt zu **Bewegung pausiert**.

pause() verursacht kein blockierendes Warten. Die Applikation wird bis zum Erreichen eines synchronen Bewegungsbefehls weiter ausgeführt.

Ein Fortsetzen der Bewegungsausführung ist nur über die Start-Taste am smartPAD möglich.

#### Syntax

```
getApplicationControl().pause();
```

### 15.28.3 for-Schleife

#### Beschreibung

Die for-Schleife, auch Zählschleife genannt, wiederholt einen Anweisungsblock solange eine definierte Bedingung erfüllt ist.

Dafür wird ein Zähler definiert, der nach jedem Schleifendurchlauf um einen konstanten Wert erhöht oder verringert wird. Zu Beginn eines Schleifendurchlaufs wird geprüft, ob die definierte Bedingung noch erfüllt ist. Diese Bedingung wird in der Regel durch einen Vergleich des Zählers mit einem Grenzwert formuliert. Ist die Bedingung nicht mehr erfüllt, wird die Schleife nicht mehr durchlaufen und die Programmausführung nach der Schleife fortgesetzt.

Die for-Schleife wird in der Regel verwendet, wenn bekannt ist, wie oft eine Schleife durchlaufen werden muss.

for-Schleifen können verschachtelt werden. (>>> 15.28.8 "Beispiele für verschachtelte Schleifen" Seite 399)

#### Syntax

```
for (int Zähler = Startwert; Bedingung; Zählanweisung) {
    Anweisung_1;
```

```
<...
Anweisung_n; >
}
```

### Erläuterung der Syntax

Element	Beschreibung
Zähler	Zähler für die Anzahl der Schleifendurchläufe. Dem Zähler wird ein Startwert zugewiesen. Nach jedem Schleifendurchlauf wird der Zähler um einen konstanten Wert erhöht oder verringert.
Startwert	Startwert des Zählers
Bedingung	Bedingung für das Ausführen eines Schleifendurchlaufs In der Regel wird der Zähler mit einem Grenzwert verglichen. Das Ergebnis des Vergleichs ist vom Typ boolean. Die Schleife wird beendet, sobald der Vergleich FALSE liefert, die Bedingung also nicht mehr erfüllt ist.
Zählanweisung	Die Zählanweisung bestimmt um welchen Wert sich der Zähler bei jedem Schleifendurchlauf ändert. Schrittweite und Zählrichtung können auf unterschiedliche Arten angegeben werden. Beispiele: <ul style="list-style-type: none"> <li>■ Startwert ++ --: Der Startwert wird nach jedem Durchlauf um den Wert 1 erhöht oder verringert.</li> <li>■ Startwert + - Schrittweite: Der Startwert wird nach jedem Schleifendurchlauf um die angegebene Schrittweite erhöht oder verringert.</li> </ul>

### Beispiel

```
for (int i = 0; i < 10; i++) {
    getLogger().info(i);
}
```

Der Wert der Variablen `i` wird nach jedem Durchlauf um 1 erhöht. Der aktuelle Wert von `i` wird bei jedem Durchlauf auf der smartHMI ausgegeben. Die Schleife wird insgesamt 10-mal durchlaufen. Dabei werden die Werte 0 bis 9 ausgegeben.

#### 15.28.4 while-Schleife

##### Beschreibung

Die while-Schleife wiederholt einen Anweisungsblock, so lange eine bestimmte Bedingung erfüllt ist. Sie wird auch abweisende Schleife genannt, da die Bedingung vor jedem Schleifendurchlauf geprüft wird.

Wenn die Bedingung nicht mehr erfüllt ist, wird der Anweisungsblock der Schleife kein weiteres Mal durchlaufen und die Programmausführung nach der Schleife fortgesetzt. Wenn die Bedingung von vorneherein nicht erfüllt ist, wird der Anweisungsblock keinmal ausgeführt.

Die while-Schleife wird in der Regel verwendet, wenn unbekannt ist, wie oft eine Schleife durchlaufen werden muss. Z. B. weil die Wiederholbedingung berechnet wird oder ein bestimmtes Signal ist.

while-Schleifen können verschachtelt werden. ([>>> 15.28.8 "Beispiele für verschachtelte Schleifen"](#) Seite 399)

##### Syntax

```
while (Wiederholbedingung) {
    Anweisung_1;
    <...
```

```
Anweisung_n; >
```

```
}
```

## Erläuterung der Syntax

Element	Beschreibung
<i>Wiederholbedingung</i>	<p>Typ: boolean</p> <p>Möglich:</p> <ul style="list-style-type: none"> <li>■ Variable vom Typ boolean</li> <li>■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ boolean</li> </ul>

## Beispiel 1

```
while(input1 == true) {
    getLogger().info("Input 1 is TRUE.");
}
getLogger().info("Input 1 is FALSE.");
```

Vor dem Schleifendurchlauf wird geprüft, ob ein Eingangssignal anliegt. Solange dies der Fall ist, wird die Schleife immer wieder durchlaufen und auf der smartHMI ausgegeben, dass der Eingang true ist. Wurde das Eingangssignal zurückgesetzt, wird die Schleife nicht (mehr) durchlaufen und ausgegeben, dass der Eingang false ist.

## Beispiel 2

```
int w = 0;
Random num = new Random();

while (w <= 21) {
    w = w + (num.nextInt(6) + 1);
}
```

Bei jedem Schleifendurchlauf wird der Wert der Variablen `w` um eine Zufallszahl zwischen 1 und 6 erhöht. Solange die Summe aller Zufallszahlen kleiner 21 ist, wird die Schleife durchlaufen. Die genaue Anzahl der Durchläufe ist nicht vorhersehbar. Es kann sein, dass die Schleife nach 4 Durchläufen ( $3 \times 6$  und  $1 \times 3$ ) beendet ist oder erst nach 21 Durchläufen ( $21 \times 1$ ).

## 15.28.5 do-while-Schleife

### Beschreibung

Die do-while-Schleife wiederholt einen Anweisungsblock so lange, bis eine bestimmte Bedingung erfüllt ist. Sie wird auch annehmende Schleife genannt, da die Bedingung erst nach jedem Schleifendurchlauf geprüft wird.

Der Anweisungsblock wird mindestens einmal ausgeführt. Wenn die Bedingung erfüllt ist, wird die Schleife beendet und das Programm fortgesetzt.

Die do-while-Schleife wird in der Regel verwendet, wenn eine Schleife mindestens einmal durchlaufen werden muss, aber unbekannt ist, wie oft insgesamt. Z. B. weil die Abbruchbedingung berechnet wird oder ein bestimmtes Signal ist.

do-while-Schleifen können verschachtelt werden. ([>>> 15.28.8 "Beispiele für verschachtelte Schleifen" Seite 399](#))

### Syntax

```
do {
    Anweisung_1;
    <...
    Anweisung_n; >
} while (Abbruchbedingung);
```

## Erläuterung der Syntax

Element	Beschreibung
Abbruchbedingung	<p>Typ: boolean</p> <p>Möglich:</p> <ul style="list-style-type: none"> <li>■ Variable vom Typ boolean</li> <li>■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ boolean</li> </ul>

## Beispiel

```
int num;

do {
    num = (int) (Math.random() * 6 + 1);
} while (num != 6);
```

Es werden solange Zufallszahlen zwischen 1 und 6 gewürfelt bis eine 6 gefallen ist. Es muss mindestens einmal gewürfelt werden.

### 15.28.6 if-else-Verzweigung

#### Beschreibung

Die if-else-Verzweigung wird auch bedingte Verzweigung genannt. Abhängig von einer Bedingung wird entweder der erste Anweisungsblock (if-Block) oder der zweite Anweisungsblock (else-Block) ausgeführt.

Der else-Block wird ausgeführt, wenn die if-Bedingung nicht zutrifft. Der else-Block darf fehlen. Dann werden, falls die if-Bedingung nicht zutrifft, keine weiteren Anweisungen ausgeführt.

Es ist möglich, nach dem if-Block mit `else if` weitere Bedingungen zu prüfen und mit Anweisungen zu verknüpfen. Sobald eine dieser Bedingungen zutrifft und die zugehörigen Anweisungen ausgeführt sind, werden die folgenden Zweige nicht mehr geprüft.

Mehrere if-Anweisungen können ineinander verschachtelt werden.

#### Syntax

```
if (Bedingung_1) {
    Anweisung_1;
    <...
    Anweisung_n;>
}

<else if (Bedingung_2) {
    Anweisung_1;
    <...
    Anweisung_n;>
}>

<else {
    Anweisung_1;
    <...
    Anweisung_n;>
}>
```

## Erläuterung der Syntax

Element	Beschreibung
Bedingung	<p>Typ: boolean</p> <p>Möglich:</p> <ul style="list-style-type: none"> <li>■ Variable vom Typ boolean</li> <li>■ Verknüpfung, z. B. ein Vergleich, mit Ergebnis vom Typ boolean</li> </ul>

### Beispiel 1

if-Verzweigung ohne else

```
int a;
int b;

if (a == 17) {
    b = 1;
}
```

Wenn die Variable `a` den Wert 17 hat, wird der Variablen `b` der Wert 1 zugewiesen.

### Beispiel 2

if-Verzweigung innerhalb einer for-Schleife, ohne else

```
for(int a = 1; a <= 10; a++) {
    if(a == 3) {
        a = a + 5;
    }
    getLogger().info(a);
}
```

Die Schleife wird 5-mal durchlaufen. Wenn die Variable `a` den Wert 3 hat, wird der Wert von `a` einmalig um 5 erhöht.

Es werden die Werte 1, 2, 8, 9 und 10 auf der smartHMI ausgegeben.

### Beispiel 3

if-else-Verzweigung mit else if

```
double velAct = 0.0;
double velDesired = 130.0;

...

if (velAct < velDesired) {
    accelerating();
}
else if (velAct > velDesired) {
    braking();
}
else {
    testrun();
}
```

In einem Programm soll für ein Fahrzeug ein Testlauf durchgeführt werden. Dieser Testlauf ist nur bei einer bestimmten Sollgeschwindigkeit aussagekräftig.

Mit der if-Abfrage wird geprüft, ob die Istgeschwindigkeit `velAct` kleiner ist als die Sollgeschwindigkeit `velDesired`. Falls ja, wird das Fahrzeug beschleunigt. Falls nein, geht es weiter mit else if.

Mit der if-else-Abfrage wird geprüft, ob die Istgeschwindigkeit `velAct` größer ist als die Sollgeschwindigkeit `velDesired`. Falls ja, wird das Fahrzeug gebremst. Falls nein, wird der else-Block mit dem Testlauf ausgeführt.

### 15.28.7 switch-Verzweigung

#### Beschreibung

Die switch-Verzweigung wird auch Mehrfachverzweigung genannt. Im Prinzip entspricht eine switch-Verzweigung einer mehrfach verschachtelten if-Verzweigung.

In einem switch-Block können verschiedene case-Blöcke ausgeführt werden, die durch case-Marken (Sprungmarken) gekennzeichnet sind. Abhängig vom Ergebnis eines Ausdrucks wird der entsprechende case-Block gewählt und abgearbeitet. Das Programm springt an die case-Marke und wird an dieser Stelle fortgesetzt.

Das Schlüsselwort `break` am Ende eines case-Blockes bedeutet, dass der switch-Block verlassen wird. Folgt kein `break` am Ende eines Anweisungsblocks, werden auch alle nachfolgenden Anweisungen (nicht nur Anweisungen mit case-Marken) ausgeführt, bis entweder eine break-Marke kommt oder alle Anweisungen durchlaufen wurden.

Optional kann ein default-Block programmiert werden. Wird keine Bedingung zum Sprung an eine case-Marke erfüllt, wird der default-Block ausgeführt.

#### Syntax

```
switch (Ausdruck) {
    case Konstante_1:
        Anweisung_1;
        ...
        Anweisung_n;>
        < break;>
        <...
        case Konstante_n:
            Anweisung_1;>
            ...
            Anweisung_n;>
            < break;>
            < default:
                Anweisung_1;>
                ...
                Anweisung_n;>
                < break;>
            }
}
```

#### Erläuterung der Syntax

Element	Beschreibung
<code>Ausdruck</code>	Typ: int, byte, short, char, enum
<code>Konstante</code>	Typ: int, byte, short, char, enum  Der Datentyp der Konstante muss mit dem Datentyp des Ausdrucks übereinstimmen.  <b>Hinweis:</b> Konstanten vom Typ char müssen mit ' angegeben werden, z. B. case 'a'

#### Beispiel

switch-Verzweigung mit break- und default-Anweisung:

```
int a, b;
switch (a){
```

```

case 1:
    b = 10;
case 2:
case 3:
    b = 20;
    break;
case 4:
    b = 30
    break;
default:
    b = 40;
}

```

Wenn die Variable `a` den Wert 1 hat, springt das Programm an die Marke `case 1`. Der Variablen `b` wird der Wert 10 zugewiesen.

Das Programm wird fortgesetzt. Bei der Marke `case 2` wird kein Befehl ausgeführt. Das Programm wechselt zur Marke `case 3`. Der Variablen `b` wird der Wert 20 zugewiesen. Hat die Variable `a` den Wert 2 oder 3, wird der Variablen `b` ebenfalls der Wert 20 zugewiesen. Mit der `break`-Anweisung wird der switch-Block verlassen.

Wenn die Variable `a` zu Beginn den Wert 4 hat, wird der Variablen `b` der Wert 30 zugewiesen.

Mit der `break`-Anweisung wird der switch-Block verlassen.

Hat die Variable `a` einen anderen Wert (z. B. 5), wird der Variablen `b` der Wert 40 zugewiesen. Die Ausführung des switch-Blockes wird beendet.

### 15.28.8 Beispiele für verschachtelte Schleifen

Zunächst wird die äußere Schleife durchlaufen bis man zur inneren Schleife kommt. Die innere Schleife wird dann komplett durchlaufen. Danach wird die äußere Schleife bis zum Ende durchlaufen und geprüft, ob die äußere Schleife erneut durchlaufen werden muss. Ist dies der Fall wird auch die innere Schleife noch einmal durchlaufen.

Schleifen können beliebig tief verschachtelt werden. Die inneren Schleifen werden immer so oft ausgeführt, wie die äußere Schleife durchlaufen wird.

#### for-in-for-Schleife

```

for (int i = 1; i < 4; i++) {
    getLogger().info(i + ".Cycle begins");

    for (int k = 10; k > 0; k--) {
        getLogger().info("..." + k);
    }
}

```

Die äußere Schleife legt fest, dass die innere Schleife 3-mal durchlaufen wird. Der Zähler der äußeren Schleife startet mit dem Wert `i = 1`.

Nachdem auf der smartHMI ausgegeben wurde, dass der 1. Durchlauf beginnt, startet der Zähler der inneren Schleife mit dem Wert `k = 10`. Der Wert der Variablen `k` wird nach jedem Durchlauf um 1 verringert. Der aktuelle Wert von `k` wird bei jedem Durchlauf auf der smartHMI ausgegeben. Wenn die Variable `k` den Wert 1 hat, wird die innere Schleife das letzte Mal durchlaufen.

Dann wird die äußere Schleife beendet und der Wert der Variablen `i` um 1 erhöht. Der 2. Durchlauf beginnt.

#### for-in-while-Schleife

```

int sum = 0;
int round = 1;
int diceRoll = 0;

```

```

Random num = new Random();

while (sum < 21) {
    round++;

    for (int i = 1; i <= 3; i++) {
        diceRoll = (num.nextInt(6) + 1);
        if (diceRoll % 2 == 0)
            sum += diceRoll;
    }
}

```

In einem Würfelspiel gelten folgende Regeln:

- Die Gesamtsumme aller Würfe muss mindestens 21 ergeben (Abfrage mit while-Schleife).
- In jeder Runde wird 3-mal gewürfelt werden (for-schleife).
- Es werden nur gerade Ziffern (2, 4 und 6) gezählt (if-Abfrage mit Modulo).

## 15.29 Pausierte Applikation im Automatikbetrieb fortsetzen (Recovery)

### Beschreibung

Wenn eine pausierte Roboter-Applikation im Automatikbetrieb fortgesetzt werden soll, muss die übergeordnete Steuerung feststellen können, ob sich der Roboter noch auf seiner programmierten Bahn befindet. Befindet er sich nicht mehr auf der Bahn, z. B. nach einem nicht bahntreuen Stopp oder weil er während der Pause manuell verfahren wurde, muss der Roboter mit einer geeigneten Strategie automatisiert rückpositioniert werden können.

Diese Rückkehrstrategie darf nur zur Anwendung kommen, wenn sichergestellt ist, dass es auf dem Weg zurück zur Bahn zu keiner Kollision kommen kann. Ist dies nicht sichergestellt, muss der Roboter manuell vom Bediener rückpositioniert werden.

Für das automatische Rückpositionieren stellt RoboticsAPI die Schnittstelle IRecovery zur Verfügung. Auf die Schnittstelle kann von Roboter-Applikationen und Hintergrund-Tasks aus zugegriffen werden:

- IRecovery getRecovery()

### Übersicht

Die Schnittstelle IRecovery stellt Methoden zur Verfügung, um abzufragen, ob Roboter zum Fortsetzen einer pausierten Applikation rückpositioniert werden müssen, und welche Rückkehrstrategie angewendet wird.

Methode	Beschreibung
isRecoveryRequired()	Rückgabetyp: boolean  Prüft für eine pausierte Applikation, ob einer oder mehrere in der Applikation verwendete Roboter rückpositioniert werden müssen.  <b>true:</b> Mindestens ein Roboter muss rückpositioniert werden, damit die Applikation fortgesetzt werden kann.  <b>false:</b> Applikation kann sofort fortgesetzt werden.

<b>Methode</b>	<b>Beschreibung</b>
isRecoveryRequired(...)	<p>Rückgabetyp: boolean</p> <p>Prüft für eine pausierte Applikation, ob ein bestimmter Roboter rückpositioniert werden muss. Der Roboter wird als Parameter übergeben (Typ: Robot).</p> <p><b>true:</b> Roboter muss rückpositioniert werden, damit die Applikation fortgesetzt werden kann.</p> <p><b>false:</b> Applikation kann sofort fortgesetzt werden.</p>
getRecoveryStrategy(...)	<p>Rückgabetyp: RecoveryStrategy</p> <p>Abfrage der Strategie, die angewendet wird, um einen bestimmten Roboter zurück auf die Bahn zu bringen. Der Roboter wird als Parameter übergeben (Typ: Robot).</p> <ul style="list-style-type: none"> <li>■ <b>PTPRecoveryStrategy:</b> Der Roboter wird mit einer PTP-Bewegung rückpositioniert. Es wird mit 20 % der maximal möglichen Achsgeschwindigkeit und dem effektiven Override verfahren. Aktuell stehen keine weiteren Strategien zur Verfügung.</li> </ul> <p>In folgenden Fällen gibt die Methode <code>null</code> zurück:</p> <ul style="list-style-type: none"> <li>■ Es ist keine Rückkehrstrategie erforderlich oder verfügbar.</li> <li>■ Die Applikation ist nicht pausiert.</li> </ul>

**PTPRecoveryStrategy** Die Klasse PTPRecoveryStrategy stellt get-Methoden zur Verfügung, mit denen die Eigenschaften der PTP-Bewegung abgefragt werden können. Mithilfe dieser Methoden kann bewertet werden, ob die Rückkehrstrategie im Automatikbetrieb ausgeführt werden darf.

<b>Methode</b>	<b>Beschreibung</b>
getStartPosition()	<p>Rückgabetyp: JointPosition</p> <p>Abfrage der Startposition der PTP-Bewegung (= Achsposition, von der aus rückpositioniert wird)</p> <p>Die Startposition ist die aktuell kommandierte Sollposition des Roboters und nicht die aktuell gemessene Istposition.</p>
getMotion()	<p>Rückgabetyp: PTP</p> <p>Abfrage der PTP-Bewegung, die beim Ausführen der Strategie ausgeführt wird</p> <p>Von dem zurückgegebenen Bewegungsobjekt können weitere Informationen abgefragt werden:</p> <ul style="list-style-type: none"> <li>■ <code>getDestination():</code> Zielposition der PTP-Bewegung (= Achsposition, an der die Bahn verlassen wurde)</li> <li>■ <code>getMode():</code> Reglermodus der Bewegung, die unterbrochen wurde</li> </ul>

**Externe Steuerung** Die Robotersteuerung muss der übergeordneten Steuerung mitteilen, ob rückpositioniert werden muss. Die übergeordnete Steuerung darf das Ausführen der Rückkehrstrategie nur zulassen, wenn dies gefahrlos möglich ist. Andernfalls darf nur manuell rückpositioniert werden.

Folgende Systemsignale stehen zur Verfügung:

- Ausgang AutExt\_AppReadyToStart  
Mit diesem Ausgang teilt die Robotersteuerung der übergeordneten Steuerung mit, ob die Applikation fortgesetzt werden darf.

- a. Wenn `isRecoveryRequired(...)` den Wert **false** liefert (= kein Rückpositionieren erforderlich), kann der Ausgang auf TRUE gesetzt werden.
- b. Wenn `getRecoveryStrategy(...)` **null** liefert (= keine Rückkehrstrategie verfügbar), muss der Ausgang auf FALSE gesetzt werden.
- c. Wenn die Bewertung der Rückkehrstrategie ergibt, dass sie im Automatikbetrieb ausgeführt werden darf, kann der Ausgang auf TRUE gesetzt werden.

Ist dies nicht der Fall, muss der Ausgang auf FALSE gesetzt werden.

#### ■ Eingang App\_Start

Über eine steigende Flanke dieses Eingangs teilt die übergeordnete Steuerung der Robotersteuerung mit, dass die Applikation fortgesetzt werden soll. (Voraussetzung: `AutExt_AppReadyToStart` ist TRUE)

Die übergeordnete Steuerung muss das Startsignal `App_Start` 2-mal schicken:

1. Startsignal zum Rückpositionieren
2. Startsignal zum Fortsetzen der Applikation

## 15.30 Fehlerbehandlung

### 15.30.1 Behandlung fehlgeschlagener Bewegungsbefehle

Bewegungsbefehle, die an die Robotersteuerung übermittelt werden, können aus unterschiedlichen Gründen fehlschlagen, z. B.:

- Zielpunkt liegt außerhalb eines Arbeitsraumes
- Zielpunkt ist mit der gegebenen Achskonfiguration nicht erreichbar
- der verwendete Frame ist in den Applikationsdaten nicht vorhanden

Ein fehlgeschlagener Bewegungsbefehl führt defaultmäßig zu einem Abbruch der Applikation. Um im Fehlerfall einen Abruch der Applikation zu verhindern, können Behandlungsroutinen festgelegt werden.

Abhängig vom Fehlerfall bestehen folgende Möglichkeiten zur Behandlung:

- Behandlung fehlgeschlagener synchroner Bewegungsbefehle erfolgt mithilfe eines try-catch-Blocks
- Behandlung fehlgeschlagener asynchroner Bewegungsbefehle erfolgt mithilfe eines Event-Handlers

### 15.30.2 Behandlung fehlgeschlagener synchroner Bewegungsbefehle

#### Beschreibung

Synchron ausgeführte Bewegungsbefehle (`.move (...);`) werden schrittweise an die Echtzeitsteuerung gesendet und abgearbeitet. Die weitere Programmausführung ist solange unterbrochen bis die Bewegung ausgeführt wurde. Erst dann wird der nächste Befehl gesendet.

Mithilfe eines try-catch-Blocks können im Programmablauf absehbare Laufzeitfehler oder Ausnahmen abgearbeitet werden, ohne dass die Applikation abgebrochen wird.

Innerhalb eines try-catch-Blocks wird eine festgelegte Methode zur Fehlerbehandlung ausgelöst. Beim Aufruf des Schlüsselwortes `try` soll versucht werden, den aufgeführten Befehl abzuarbeiten. Tritt bei der Abarbeitung ein Fehler auf, wird die entsprechende Behandlungsroutine im catch-Block gestartet.

#### Syntax

```
try {
    // Code, bei dessen Ausführung ein Laufzeitfehler auftreten könnte
```

```

}

catch (Exception e) {
    // Code zur Behandlung des Laufzeitfehlers
}

< finally{
    // Abschlussbehandlung (optional)
}>

```

### Erläuterung der Syntax

Element	Beschreibung
try{...}	Der try-Block enthält einen Code, der zu einem Laufzeitfehler führen könnte. Tritt ein Fehler auf, wird die Ausführung des try-BLOCKS abgebrochen und der catch-Block ausgeführt.
catch (...){...}	Der catch-Block enthält den Code zur Behandlung des Laufzeitfehlers. Der catch-Block wird nur dann durchlaufen, wenn im try-Block ein Fehler auftritt.
Exception e	Über den Fehler-Datentyp (hier: Exception) kann die Fehlerart festgelegt werden, die im catch-Block behandelt werden soll. Die Fehlerart Exception ist Oberklasse der meisten Fehler-Datentypen. Es kann jedoch auch auf spezifischere Fehler eingegangen werden. Über den Parameter e können Informationen über aufgetretenen Fehler abgefragt werden. Zur Behandlung fehlgeschlagener Bewegungsbefehle ist vor allem der Fehler-Datentyp CommandInvalidException (Paket: com.kuka.roboticsAPI.executionModel) von Bedeutung. Er tritt unter anderem dann auf, wenn der Zielpunkt der Bewegung nicht erreicht werden kann.
finally{...}	Der finally-Block ist optional. Hier kann eine Abschlussbehandlung festgelegt werden, die auf jeden Fall ausgeführt werden soll. Unabhängig davon, ob im try-Block ein Fehler aufgetreten ist oder nicht.

### Beispiel

Ein Roboter führt eine Bewegung impedanzgeregelt mit sehr geringer Steifigkeit aus. Daher ist nicht sichergestellt, ob er die Zielposition erreicht. Anschließend soll er sich relativ um 50 cm in positiver Z-Richtung des Flansch-Koordinatensystems bewegen. Steht der Roboter nach der impedanzgeregelten Bewegung in einer ungünstigen Position, kann die lineare Bewegung nicht ausgeführt werden und ein Laufzeitfehler tritt auf. Um in diesem Fall einen Abbruch der Applikation zu verhindern, wird die kritische lineare Bewegung in einem try-catch-Block programmiert. Schlägt die Bewegungsplanung fehl, soll der Roboter vor Fortsetzung des Applikationsablaufs an einen Zwischenpunkt gefahren werden.

```

CartesianImpedanceControlMode softMode = new
CartesianImpedanceControlMode();

softMode.parametrize(CartDOF.ALL).setStiffness(10.0);
exampleRobot.move(ptp(getFrame("/Start"))
    .setMode(softMode).setJointVelocityRel(0.3));

try{

```

```

        getLogger().info("1: Versuche, lineare Bewegung auszuführen");
        exampleRobot.move(linRel(0.0, 0.0, 500.0)
            .setJointVelocityRel(0.5));
    }

    catch(CommandInvalidException e){
        getLogger().info("2: Bewegung nicht möglich");
        exampleRobot.move(ptp(getFrame("/Zwischenpunkt"))
            .setJointVelocityRel(0.5));
    }

    finally{
        getLogger().info(
            "3: Abschlussbehandlung im finally-Block wird ausgeführt");
    }

    getLogger().info("4: Weiter im Programm");
}

```

### 15.30.3 Behandlung fehlgeschlagener asynchroner Bewegungsbefehle

#### Beschreibung

Bei asynchron ausgeführten Bewegungsbefehlen (.moveAsync(...);) wird direkt nach dem Senden des Bewegungsbefehls die nächste Programmzeile ausgeführt.

Um auf einen fehlgeschlagenen asynchronen Bewegungsbefehl zu reagieren, wird ein Event-Handler verwendet.

Dieser Event-Handler ist ein Objekt vom Typ IErrorHandler und definiert die Methode handleError(...). Während der Ausführung der Methode handleError(...) wird das Übermitteln weiterer Bewegungsbefehle an die Echtzeitsteuerung blockiert, die Applikation bleibt stehen.

Mit handleError(...) wird die Behandlungsroutine festgelegt. Über die Eingangsparameter der Methode kann auf Informationen zu dem fehlgeschlagenen Bewegungsbefehl zugegriffen werden. Die Methode gibt einen Parameter vom Typ ErrorHandlingAction zurück. Über diesen wird die abschließende Reaktion auf den Fehler ausgewählt.

Folgende Reaktionen stehen zur Auswahl:

- Die Applikation wird mit einem Fehler beendet.
- Die Bewegungsausführung wird pausiert und muss vom Bediener über die Start-Taste am smartPAD fortgesetzt werden.
- Der Fehler wird ignoriert und die Applikation fortgesetzt.

Der definierte Event-Handler muss registriert werden, bevor er in der Applikation verwendet werden kann. Hierfür wird die Methode getApplicationControl().registerMoveAsyncErrorHandler(...) verwendet. Die Methode gehört zur Schnittstelle IApplicationControl.

#### Syntax

Event-Handler definieren:

```

IErrorHandler errorHandler = new IErrorHandler() {
    @Override
    public ErrorHandlingAction handleError
        (Device device, IMotionContainer failedContainer,
        List<IMotionContainer> canceledContainers) {
        // Code, der im Fehlerfall ausgeführt wird
        return ErrorHandlingAction.reaction;
}

```

```

    }
};

Event-Handler registrieren:
getApplicationControl().registerMoveAsyncErrorHandler(errorHandler);

```

### Erläuterung der Syntax

Element	Beschreibung
errorHandler	Typ: IErrorHandler Name des Event-Handlers, der für die Behandlung fehlgeschlagener asynchroner Bewegungsbefehle zuständig ist.
Eingangsparameter der Methode handleError(...):	
device	Typ: Device Über den Parameter kann auf den Roboter zugegriffen werden, für den der fehlgeschlagene Bewegungsbefehl kommandiert wurde.
failed Container	Typ: IMotionContainer Über den Parameter kann auf den fehlgeschlagenen Bewegungsbefehl zugegriffen werden.
canceled Containers	Typ: List<IMotionContainer> Über den Parameter kann auf eine Liste aller gelöschten Bewegungsbefehle zugegriffen werden. Es sind alle Bewegungsbefehle enthalten, die bei Aufruf der Methode handleError(...) bereits an die Echtzeitsteuerung geschickt wurden.
reaction	Typ: Enum vom Typ ErrorHandlingAction Rückgabewert der Methode handleError(...), über den die abschließende Reaktion auf den Fehler festgelegt wird. <ul style="list-style-type: none"> <li>■ <b>ErrorHandlingAction.EndApplication:</b> Die Applikation wird mit einem Fehler beendet.</li> <li>■ <b>ErrorHandlingAction.PauseMotion:</b> Die Bewegungsausführung wird pausiert, bis der Bediener die Applikation über das smartPAD fortsetzt.</li> <li>■ <b>ErrorHandlingAction.Ignore:</b> Der Fehler wird ignoriert und die Applikation fortgesetzt.</li> </ul>

### Beispiel

In einer Applikation sollen nacheinander mehrere asynchrone Bewegungsbefehle ausgeführt werden. Über die Registrierung eines Event-Handlers vom Typ IErrorHandler wird mit der Methode handleError(...) eine Behandlungsroutine für den Fall festgelegt, dass einer der asynchronen Bewegungsbefehle fehlschlägt:

- Auf der smartHMI wird ausgegeben, welcher Bewegungsbefehl fehlgeschlagen ist.
- Auf der smartHMI wird ausgegeben, welche Bewegungsbefehle nicht mehr ausgeführt werden.

Die Methode handleError(...) wird mit der Rückgabe des Wertes ErrorHandlingAction.Ignore beendet.

```

public void initialize(){
    kuka_Sunrise_Cabinet_1 = getController("kuka_Sunrise_Cabinet_1");
    robot = (LBR) getRobot(kuka_Sunrise_Cabinet_1, "LBR_iwa_14_R820_1");

    IErrorHandler errorHandler = new IErrorHandler()

```

```
    @Override
    public ErrorHandlingAction handleError(Device device,
        IMotionContainer failedContainer,
        List<IMotionContainer> canceledContainers) {
        getLogger().warn("Folgender Bewegungsbefehl ist
fehlgeschlagen: " + failedContainer.toString());
        getLogger().info("Folgende Bewegungsbefehle werden nicht
ausgeführt: ");
        for(int i = 0; i < canceledContainers.size(); i++){
            getLogger().info(canceledContainers.get(i).toString());
        }
        return ErrorHandlingAction.Ignore
    }
};

getApplicationControl().
registerMoveAsyncErrorHandler(errorHandler);
}

public void run(){
...
    robot.moveAsync(ptp(getFrame("/P1")));
    robot.moveAsync(ptp(getFrame("/P2")));

    robot.moveAsync(lin(getFrame("/P3")));

    robot.moveAsync(ptp(getFrame("/P4")));
    robot.moveAsync(ptp(getFrame("/P5")));
    robot.moveAsync(ptp(getFrame("/P6")));

    robot.moveAsync(ptp(getFrame("/P7")));
    robot.moveAsync(ptp(getFrame("/P8")));
    robot.moveAsync(ptp(getFrame("/P9")));
...
}
```

Zur Erläuterung des Systemverhaltens wird angenommen, dass die Linearbewegung zu P3 nicht geplant werden kann. Dadurch wird die Methode `handleError(...)` aufgerufen. Zu diesem Zeitpunkt befindet sich in unserem Beispiel der Roboter am Zielpunkt P2.

Wenn sich gleichzeitig zu diesem Zeitpunkt beispielsweise die Bewegungsbefehle zu P4, P5, P6 bereits in der Echtzeitsteuerung befinden, werden diese Bewegungsbefehle gelöscht und nicht mehr ausgeführt.

Bei Aufruf der Methode `handleError(...)` wird das Abschicken weiterer Bewegungsbefehle an die Echtzeitsteuerung blockiert. Die Applikation wird in diesem Fall vor dem Bewegungsbefehl zu P7 angehalten. Wenn die Methode `handleError(...)` mit der Rückgabe des Wertes `ErrorHandlingAction.Ignore` beendet wird, wird die Applikation fortgesetzt. Der Roboter fährt dann von der aktuellen Position P2 direkt zu P7.

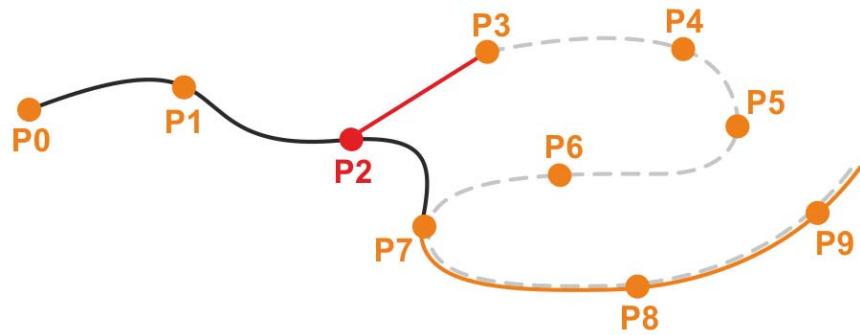


Abb. 15-22: Bewegung zu P3 fehlgeschlagen (Beispiel Bahnverlauf)



# 16 Hintergrund-Tasks

## 16.1 Verwendung von Hintergrund-Tasks

### Aufgaben

Hintergrund-Tasks werden eingesetzt, um parallel zu einer laufenden Roboter-Applikation Aufgaben ausführen zu können, oder um zyklische Prozesse zu realisieren, die ständig im Hintergrund ausgeführt werden sollen. Es können mehrere Hintergrund-Tasks gleichzeitig und unabhängig von der laufenden Roboter-Applikation ausgeführt werden.

Hintergrund-Tasks werden insbesondere zum Steuern und Überwachen von Peripheriegeräten eingesetzt und dazu, die zugehörige übergeordnete Logik zu implementieren. Beispiele:

- Schalten von Signalleuchten
- Überwachen und Auswerten von Sensorinformationen

Für kleinere Anwendungen ist somit keine übergeordnete Steuerung, z. B. eine SPS notwendig, da die Robotersteuerung solche Aufgaben mit übernehmen kann.



Bei Ausgängen, die über einen Hintergrund-Task geschalten werden, sind folgende Punkte zu beachten:

- Die Ausgänge werden unabhängig davon geschalten, ob aktuell eine Roboter-Applikation ausgeführt wird.
- Die Ausgänge werden auch dann geschalten, wenn die Roboter-Applikation wegen eines NOT-HALTs oder fehlender Zustimmung pausiert ist.
- Die Ausgänge werden auch dann geschalten, wenn eine Stopp-Anforderung der Sicherheitssteuerung anliegt. (Gilt auch, wenn Ausgänge über eine Roboter-Applikation geschalten werden.)



**WARNING** Hintergrund-Tasks dürfen nicht dazu genutzt werden, den Roboter zu bewegen oder Parameter zu beeinflussen, die sich auf Bewegungen auswirken können. Dies ist die Aufgabe der Roboter-Applikation. Der Aufruf von Bewegungskommandos oder die Änderung bewegungsbezogener Parameter in einem Hintergrund-Task kann zu einem nicht spezifizierten Verhalten des Roboters und damit zu Sach- und Personenschaden führen.

### Eigenschaften

Hintergrund-Tasks sind wie Roboter-Applikationen als Java-Klassen realisiert. Sie sind ähnlich aufgebaut wie Roboter-Applikationen: Sie besitzen eine run()-Methode, die die auszuführenden Befehle enthält.

Hintergrund-Tasks sind Bestandteil des Sunrise-Projekts. Sie werden in Sunrise.Workbench erstellt und mit der Projekt-Synchronisation auf die Robotersteuerung übertragen.

(>>> 5.5 "Neuen Hintergrund-Task erstellen" Seite 55)

Es gibt 2 Arten von Hintergrund-Tasks, die sich in ihrer Lebensdauer unterscheiden:

- Zyklische Hintergrund-Tasks  
Werden zyklisch ausgeführt. Das zyklische Verhalten kann je nach Aufgabe, die der Task erfüllen soll, vom Programmierer angepasst werden.
- Nicht-zyklische Hintergrund-Tasks  
Werden einmalig ausgeführt.

Hintergrund-Tasks unterscheiden sich außerdem in ihrem Starttyp:

■ **Manuell**

Der Task muss manuell über das smartPAD gestartet werden. (Funktion wird zur Zeit noch nicht unterstützt.)

■ **Automatisch**

Der Task wird in folgenden Fällen automatisch gestartet:

- Nach einem Hochlauf der Robotersteuerung
- Nach einer Projekt-Synchronisation von Sunrise.Workbench auf die Robotersteuerung

Laufende Hintergrund-Tasks werden in folgenden Fällen automatisch beendet:

- Beim Herunterfahren der Robotersteuerung
- Bei einer Projekt-Synchronisation von Sunrise.Workbench auf die Robotersteuerung
- Bei Auftreten eines nicht behandelten Fehlers während der Ausführung des Task



Wenn ein Hintergrund-Task wegen eines nicht behandelten Fehlers beendet wurde, kann der Task nur durch einen Neustart der Robotersteuerung oder durch eine erneute Projekt-Synchronisation von Sunrise.Workbench auf die Robotersteuerung neu gestartet werden.

Jeder Hintergrund-Task besitzt eine eigene Instanz vom Typ Controller. Darüber ist z. B. der Zugriff auf konfigurierte E/A-Gruppen zum Steuern von Peripherie oder das Abfragen von Roboterdeaten möglich.

Benötigt ein Hintergrund-Task Zugang zu Informationen aus der laufenden Roboter-Applikation oder aus anderen Hintergrund-Tasks, steht eine eigene Schnittstelle für den Datenaustausch zwischen Tasks zur Verfügung.

(>>> 16.4 "Datenaustausch zwischen Tasks" Seite 414)

## 16.2 Zyklischer Hintergrund-Task

### Aufbau

```

(1) package backgroundTask;

(2) import java.util.concurrent.TimeUnit;

(3) public class BackgroundTaskCyclic extends RoboticsAPICyclicBackgroundTask {
    private Controller kuka_Sunrise_Cabinet_1; (4)

(5)     public void initialize() {
        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        initializeCyclic(0, 500, TimeUnit.MILLISECONDS,
                        CycleBehavior.BestEffort);
    }

(6)     public void runCyclic() {
    }
}

```

Abb. 16-1: Aufbau eines zyklischen Hintergrund-Task

Pos.	Beschreibung
1	Zeile mit dem Namen des Pakets, in dem der Task liegt
2	Import-Bereich Der Bereich enthält die importierten Klassen, die zur Programmierung des Task benötigt werden.
3	Kopfzeile des Tasks Der zyklische Hintergrund-Task ist eine Unterklasse von RoboticsAPICyclicBackgroundTask.

Pos.	Beschreibung
4	<p>Deklarations-Bereich</p> <p>Beim Erstellen des Task wird automatisch eine Instanz vom Typ Controller angelegt. Über die Controller-Instanz können beispielsweise Instanzen von E/A-Gruppen erzeugt werden, die wiederum das Ansteuern angeschlossener Ein-/Ausgänge ermöglichen.</p>
5	<p>Methode initialize()</p> <p>Hier werden die im Deklarations-Bereich angelegten Datenfelder mit Anfangswerten belegt.</p> <p>Die Methode initializeCyclic(...) ist defaultmäßig vorhanden. Mit initializeCyclic(...) wird das zyklische Verhalten des Task festgelegt.</p> <p>(&gt;&gt;&gt; "Initialisierung" Seite 411)</p> <p><b>Hinweis:</b> Die Methode darf nicht gelöscht oder umbenannt werden.</p>
6	<p>Methode runCyclic()</p> <p>Hier wird der Code programmiert, der zyklisch ausgeführt werden soll.</p> <p><b>Hinweis:</b> Die Methode darf nicht gelöscht oder umbenannt werden.</p>

## Initialisierung

Mit initializeCyclic(...) wird das zyklische Verhalten des Hintergrund-Task festgelegt.

Beim Erstellen eines zyklischen Hintergrund-Task wird der Aufruf von initializeCyclic(...) automatisch eingefügt. Die Eingangsparameter der Methode werden dabei mit Anfangswerten belegt, die zu folgendem zyklischen Verhalten führen:

- Zeitliche Verzögerung: 0 ms
- Periodendauer: 500 ms
- Verhalten bei Überschreitung der definierten Periodendauer: runCyclic() wird weiter ausgeführt.

Die Anfangswerte können vom Programmierer geändert werden.

```
initializeCyclic(long initialDelay, long period, TimeUnit timeUnit,  
CycleBehavior behavior);
```

Element	Beschreibung
<i>initialDelay</i>	<p>Zeitliche Verzögerung, mit der der zyklische Hintergrund-Task nach dem Start erstmalig ausgeführt wird. Alle weiteren Zyklen werden ohne Zeitverzögerung ausgeführt.</p> <p>Die Zeiteinheit wird mit <i>timeUnit</i> festgelegt.</p>
<i>period</i>	<p>Periodendauer (= Zeitspanne zwischen 2 Aufrufen von runCyclic())</p> <p>Die Periodendauer wird auch dann eingehalten, wenn die Ausführungszeit von runCyclic() die festgelegte Periodendauer unterschreitet. Das Verhalten, wenn runCyclic() die Periodendauer überschreitet, wird mit <i>behavior</i> festgelegt.</p> <p>Die Zeiteinheit wird mit <i>timeUnit</i> festgelegt.</p>

Element	Beschreibung
<i>timeUnit</i>	Zeiteinheit von <i>initialDelay</i> und <i>period</i> Das Enum TimeUnit ist Bestandteil der Java-Standardbibliothek.
<i>behavior</i>	Verhalten bei Zeitüberschreitung Es wird festgelegt, wie sich der Hintergrund-Task verhalten soll, wenn die mit <i>period</i> festgelegte Periodendauer durch die Laufzeit von runCyclic() überschritten wird. <ul style="list-style-type: none"> <li>■ <b>CycleBehavior.BestEffort</b> runCyclic() wird vollständig ausgeführt und anschließend erneut aufgerufen.</li> <li>■ <b>CycleBehavior.Strict</b> Die Ausführung des Hintergrund-Task wird mit einem Fehler vom Typ CycleExceededException abgebrochen.</li> </ul>

**Beispiel**

Ein Roboter soll Werkstücke fügen, die er aus einem Magazin entnimmt. Das Magazin kann maximal 100 Werkstücke enthalten und wird manuell aufgefüllt. Unterschreitet der Restbestand im Magazin eine Menge von 20 Werkstücken, wird dies der Robotertsteuerung über einen digitalen Eingang gemeldet. Daraufhin soll eine LED im 500 ms-Takt blinken, um dem Werker zu signalisieren, dass ein Befüllen des Magazins erforderlich ist. Zusätzlich soll eine weitere LED blinken, wenn die am Roboterflansch ermittelte Kraft eine Grenze von 150 N übersteigt.

Zur Datenauswertung und Ansteuerung der LEDs wird ein zyklischer Hintergrund-Task genutzt. Der Hintergrund-Task wird im 500 ms-Takt ausgeführt.

```
public class LEDTask extends RoboticsAPICyclicBackgroundTask {
    private Controller _sunrise_Cabinet;
    private LBR _lbr_iwa;
    private ProcessParametersIOGroup _processParalOs;
    private ProcessParametersLEDsIOGroup _LED_IOs;

    public void initialize() {
        _sunrise_Cabinet = getController("KUKA_Sunrise_Cabinet_1");
        initializeCyclic(0, 500, TimeUnit.MILLISECONDS,
            CycleBehavior.BestEffort);

        // initialize LBR instance
        getRobot(_sunrise_Cabinet, "LBR_iwa_7_R800_1");

        // IOs for process parameters
        _processParalOs = new
        ProcessParametersIOGroup(_sunrise_Cabinet);
        // IOs for controlling LED-signals
        _LED_IOs = new
        ProcessParametersLEDsIOGroup(_sunrise_Cabinet);
    }

    public void runCyclic() {
        // Check if refill is required (value true)
        boolean refillRequired =
        _processParalOs.getSensor_RefillRequired();

        if (refillRequired) {
            /*
             * If refill is required, the appropriate LED changes
            */
        }
    }
}
```

```

 * its state with every execution of runCyclic()
 */
boolean currentState = _LED_IOs.getLED_RefillRequired();
_LED_IOs.setLED_RefillRequired(!currentState);
}

else{
/*
 * If refill is not required, the LED remains off
*/
_LED_IOs.setLED_RefillRequired(false);
}

// Query the applied force
Vector forceVector = _lbr_iwa.getExternalForceTorque(
_lbr_iwa.getFlange()).getForce();
// Calculate the absolute force value
double forceValue = forceVector.length();
// Check if force exceeds 150N
if(forceValue > 150.0){
/*
 * If the force limit is exceeded, the appropriate LED
 * changes its state with every execution of runCyclic()
*/
boolean currentStateForceLED =
_LED_IOs.getLED_ForceExceeded();
_LED_IOs.setLED_ForceExceeded(!currentStateForceLED);
} else{
    _LED_IOs.setLED_ForceExceeded(false);
}

```

### 16.3 Nicht-zyklischer Hintergrund-Task

## Aufbau

```
① package backgroundTask;  
② import com.kuka.roboticsAPI.applicationModel.tasks.RoboticsAPIBackgroundTask;  
③ public class BackgroundTask extends RoboticsAPIBackgroundTask {  
    private Controller kuka_Sunrise_Cabinet_1;  
    ④  
    ⑤ public void initialize() {  
        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");  
    }  
    ⑥ public void run() {  
    }  
}
```

Abb. 16-2: Aufbau eines nicht-zyklischen Hintergrund-Task

Pos.	Beschreibung
1	Zeile mit dem Namen des Pakets, in dem der Task liegt
2	Import-Bereich  Der Bereich enthält die importierten Klassen, die zur Programmierung des Task benötigt werden.
3	Kopfzeile des Tasks  Der nicht-zyklische Hintergrund-Task ist eine Unterklasse von RoboticsAPIBackgroundTask.

Pos.	Beschreibung
4	<p>Deklarations-Bereich</p> <p>Beim Erstellen des Tasks wird automatisch eine Instanz der Typ Controller angelegt. Über die Controller-Instanz können beispielsweise Instanzen von E/A-Gruppen erzeugt werden, die wiederum das Ansteuern angeschlossener Ein-/Ausgänge ermöglichen.</p>
5	<p>Methode initialize()</p> <p>Hier werden die im Deklarations-Bereich angelegten Datenfelder mit Anfangswerten belegt.</p> <p><b>Hinweis:</b> Die Methode darf nicht gelöscht oder umbenannt werden.</p>
6	<p>Methode run()</p> <p>Hier wird der Code programmiert, der einmalig ausgeführt werden soll. Die Laufzeit ist nicht begrenzt.</p> <p><b>Hinweis:</b> Die Methode darf nicht gelöscht oder umbenannt werden.</p>

## 16.4 Datenaustausch zwischen Tasks

### Beschreibung

Über den hier beschriebenen Mechanismus können Daten zwischen laufenden Tasks ausgetauscht werden. Ein Task kann Task-Funktionalitäten zur Verfügung stellen (anbietender Task), auf die andere Tasks (anfragende Tasks) zugreifen können. Damit ist es z. B. möglich, in einem Hintergrund-Task auf Informationen aus der laufenden Roboter-Applikation zuzugreifen.

Es ist für die Programmierung unerheblich, ob Daten zwischen einem Hintergrund-Task und einer Roboter-Applikation ausgetauscht werden oder zwischen 2 Hintergrund-Tasks. Es spielt auch keine Rolle, ob die Roboter-Applikation oder der Hintergrund-Task der anbietende Task ist. Aus diesem Grund sind hier Hintergrund-Tasks und Roboter-Applikationen unter dem Begriff Task zusammengefasst.

### Übersicht

Damit anbietender und anfragender Task miteinander kommunizieren können, sind folgende Schritte erforderlich:

Schritt	Beschreibung
1	<p>Schnittstelle erstellen, die vom Typ ITaskFunction (com.kuka.roboticsAPI.applicationModel.tasks) ableitet, und die gewünschten Task-Funktionalitäten deklarieren.</p> <p>(&gt;&gt;&gt; 16.4.1 "Task-Funktionalitäten deklarieren" Seite 415)</p>
2	<p>Klasse erstellen, die die von ITaskFunction abgeleitete Schnittstelle implementiert, und die deklarierten Task-Funktionalitäten programmieren.</p> <p>(&gt;&gt;&gt; 16.4.2 "Task-Funktionalitäten implementieren" Seite 416)</p> <p><b>Hinweis:</b> Die Schnittstelle kann direkt vom anbietenden Task implementiert werden oder alternativ durch eine eigens für diesen Zweck erstellte Klasse.</p>

Schritt	Beschreibung
3	<p>Anbietenden Task erstellen.</p> <p>Der anbietende Task muss die Schnittstelle ITaskFunction-Provider implementieren und bekanntgeben, dass er die in der von ITaskFunction abgeleiteten Schnittstelle definierten Task-Funktionalitäten zur Verfügung stellt.</p> <p>(&gt;&gt;&gt; 16.4.3 "Anbietenden Task erstellen" Seite 417)</p>
4	<p>Im anfragenden Task ein Objekt vom Typ ITaskFunctionAccess- sor über eine statische Methode holen. Über dieses Objekt wird auf die Task-Funktionalitäten zugegriffen.</p> <p>Für die ITaskFunctionAccess- sor-Instanz muss die von ITaskFunction abgeleitete Schnittstelle angegeben werden, in der die benötigten Task-Funktionalitäten deklariert sind.</p> <p>(&gt;&gt;&gt; 16.4.4 "Task-Funktionalitäten nutzen" Seite 420)</p>

**Beispiel**

Der Datenaustausch zwischen Tasks wird in den nachfolgenden Abschnitten schrittweise anhand des folgenden Beispiels beschrieben:

Durch die Roboter-Applikation "AssemblyApplication" soll ein Fügeprozess realisiert werden. Während des Fügeprozesses soll eine LED blinken. Kommt der Roboter während der Applikation von der Bahn ab und muss rückpositio-  
niert werden, soll eine weitere LED blinken.

Die LEDs werden durch den Hintergrund-Task "LEDTask" angesteuert. Der Hintergrund-Task ist in diesem Beispiel der anfragende Task.

Die Roboter-Applikation ist der anbietende Task. Sie muss den Zugriff auf ihre Recovery-Schnittstelle ermöglichen, über die geprüft wird, ob ein Rückpositio-  
nieren des Roboters erforderlich ist. Außerdem muss sie Start und Ende des Fügeprozesses bekanntgeben.

**16.4.1 Task-Funktionalitäten deklarieren****Beschreibung**

Über eine Schnittstelle, die vom Typ ITaskFunction ableitet, werden die ge-  
wünschten Task-Funktionalitäten deklariert.



Die Schnittstelle darf nur diejenigen Methoden deklarieren, die dem anfragenden Task zur Verfügung gestellt werden sollen. Es wird daher empfohlen, in der erstellten Schnittstelle keine set-Methoden zum Setzen von Feldern zu deklarieren. Stattdessen können solche Methoden von der implementierenden Klasse angeboten werden.

**Beispiel**

Deklaration der Task-Funktionalitäten über die Schnittstelle IApplicationInfor-  
mationFunction

Die Methode zum Setzen der Recovery-Schnittstelle soll dem anfragenden Task nicht zur Verfügung stehen und wird daher von der Schnittstelle nicht de-  
klariert.

```

1 public interface IApplicationInformationFunction extends
2     ITaskFunction {
3
4     /**
5      * Signifies whether assembly is currently executed
6      * @return true, if assembly is executed
7      */
8     public boolean isAssemblyRunning();
9     /**
10      * Called from application when assembly is

```

```

11     * started and finished
12     * @param assembly Set to true when assembly is started.
13     * Reset when assembly is stopped.
14     */
15     public void setAssemblyRunning(boolean assembly);
16
17 /**
18     * Returns whether the application requires
19     * repositioning of the robot
20     * @return true if repositioning is required
21     */
22     public boolean isApplicationRecoveryRequired();
23 /**
24     * Called from application to give access to its
25     * recovery interface
26     * @param applicationRecoveryInterface Recovery
27     * interface of the application
28     */
29 }

```

Zeile	Beschreibung
1 ... 29	Schnittstelle IApplicationInformationFunction
8	Methode isAssemblyRunning() Wird vom anfragenden Task aufgerufen, um abzufragen, ob aktuell der Fügeprozess abläuft.
15	Methode setAssemblyRunning(...) Wird von der Roboter-Applikation aufgerufen, wenn der Fügeprozess gestartet oder beendet wird.
22	Methode isApplicationRecoveryRequired() Wird vom anfragenden Task aufgerufen, um abzufragen, ob ein Rückpositionieren des Roboters erforderlich ist.

#### 16.4.2 Task-Funktionalitäten implementieren

##### Beschreibung

Es muss eine Klasse zur Verfügung gestellt werden, die die von ITaskFunction abgeleitete Schnittstelle implementiert und in der die deklarierten Task-Funktionalitäten programmiert sind. Dafür kann der anbietende Task, aber auch eine eigens für diesen Zweck erstellte Klasse verwendet werden.

##### Beispiel

Implementierung der Schnittstelle IApplicationInformationFunction durch die Klasse ApplicationInformation

Die Roboter-Applikation benötigt eine Methode, um ihre Recovery-Schnittstelle zur Verfügung zu stellen. Diese Methode wird nicht von der Schnittstelle IApplicationInformationFunction, sondern von der Klasse ApplicationInformation deklariert und implementiert.

```

1 public class ApplicationInformation implements
2     IApplicationInformationFunction {
3     private boolean _assembly;
4     private IRecovery _applicationRecoveryInterface;
5
6     @Override
7     public boolean isAssemblyRunning() {
8         return _assembly;
9     }
10
11    @Override

```

```

12  public void setAssemblyRunning(boolean assembly) {
13      _assembly = assembly;
14  }
15
16  @Override
17  public boolean isApplicationRecoveryRequired() {
18      return _applicationRecoveryInterface.isRecoveryRequired();
19  }
20
21  @Override
22  public void setApplicationRecoveryInterface(
23      IRecovery applicationRecoveryInterface) {
24      _applicationRecoveryInterface =
25          applicationRecoveryInterface;
26  }
27 }
```

Zeile	Beschreibung
1 ... 27	Klasse ApplicationInformation In der Klasse sind die Task-Funktionalitäten programmiert.
3, 4	Deklaration der Datenfelder <ul style="list-style-type: none"> <li>■ <code>_assembly</code>: Speichert den aktuellen Status des Fügeprozesses</li> <li>■ <code>_applicationRecoveryInterface</code>: Verweist auf die Recovery-Schnittstelle der Roboter-Applikation</li> </ul>
6 ... 9	Methode <code>isAssemblyRunning()</code> Wird vom anfragenden Task aufgerufen, um abzufragen, ob aktuell der Fügeprozess läuft.
11 ... 14	Methode <code>setAssemblyRunning(...)</code> Wird von der Roboter-Applikation aufgerufen, wenn der Fügeprozess gestartet oder beendet wird.
16 ... 19	Methode <code>isApplicationRecoveryRequired()</code> Wird vom anfragenden Task aufgerufen, um abzufragen, ob ein Rückpositionieren des Roboters erforderlich ist.
21 ... 26	Methode <code>setApplicationRecoveryInterface(...)</code> Wird von der Roboter-Applikation aufgerufen, um ihre Recovery-Schnittstelle zur Verfügung zu stellen.

#### 16.4.3 Anbietenden Task erstellen

Ein Task kann Task-Funktionalitäten mehrerer unterschiedlicher von `ITaskFunction` abgeleiteten Schnittstellen anbieten.

Damit ein Task Task-Funktionalitäten anbieten kann, müssen an der Task-Klasse einige Anpassungen vorgenommen werden.

##### Annotation

Der anbietende Task muss bekanntgeben, dass er die in der von `ITaskFunction` abgeleiteten Schnittstelle definierten Task-Funktionalitäten zur Verfügung stellt.

Dafür wird folgende Annotation über dem Klassenkopf der Roboter-Applikation eingefügt:

```
@ProvidedFunctions (ITaskFunctionType.class)
```

- *ITaskFunctionType*: Name der von ITaskFunction abgeleiteten Schnittstelle. Der Task bietet die von dieser Schnittstelle definierten Task-Funktionalitäten an.

Werden von einem Task Task-Funktionalitäten mehrerer von ITaskFunction abgeleiteter Schnittstellen angeboten, müssen die Datentypen als Array angegeben werden:

```
@ProvidedFunctions({ITaskFunctionType_1.class,  
ITaskFunctionType_2.class, ...})
```



Jede von ITaskFunction abgeleitete Schnittstelle darf nur 1-mal zur Verfügung gestellt werden. Das bedeutet, es darf keine 2 Tasks geben, die die gleiche Schnittstelle in ihrer @ProvidedFunctions-Annotation angeben.

#### Beispiel:

Die Roboter-Applikation "AssemblyApplication" bietet die durch die Schnittstelle IApplicationInformationFunction deklarierten Task-Funktionalitäten an. Dies wird durch die Annotation über dem Klassenkopf der Roboter-Applikation bekanntgegeben.

```
@ProvidedFunctions(IApplicationInformationFunction.class)  
  
public class AssemblyApplication ...
```

#### Schnittstelle

Der anbietende Task muss die Schnittstelle ITaskFunctionProvider implementieren und die Methode createTaskFunctions() anbieten:

```
public Map<Class<? extends ITaskFunction>, ITaskFunction>  
createTaskFunctions () {  
...  
}
```



Wenn der anbietende Task die von ITaskFunction abgeleitete Schnittstelle nicht selbst implementiert, benötigt er eine Instanz der implementierenden Klasse. Es wird empfohlen, diese Instanz als Feld anzulegen.

Die Methode createTaskFunctions() gibt ein Objekt vom Typ Map zurück, das alle ITaskFunction-Instanzen des anbietenden Task enthält. Die Map-Instanz wird durch folgenden Befehl erzeugt und kann ohne weitere Anpassung in den eigenen Quellcode übernommen werden:

```
Map<Class<? extends ITaskFunction>, ITaskFunction> map  
= new HashMap<Class<? extends ITaskFunction>, ITaskFunction>();
```

- *map*: Bezeichner der Map-Instanz

Zur Speicherung der ITaskFunction-Instanzen in dem Map-Objekt wird der Map-Befehl put(...) verwendet. Dem Befehl werden der Typ der von ITaskFunction abgeleiteten Schnittstelle sowie die zugehörige ITaskFunction-Instanz übergeben:

```
map.put (iTaskFunctionType.class, value);
```

- *iTaskFunctionType*: Bezeichner der von ITaskFunction abgeleiteten Schnittstelle, die die deklarierten Task-Funktionalitäten enthält.
- *value*: ITaskFunction-Instanz des anbietenden Tasks. Der Datentyp von *value* ist durch den Parameter *iTaskFunctionType.class* definiert.



Implementiert der anbietende Task die Schnittstelle selbst, wird für den Parameter `value` die Instanz des Task (Referenz `this`) übergeben:

```
map.put(iTaskFunctionType.class, this)
```

## Gesamtbeispiel

Die Roboter-Applikation erhält ein Datenfeld vom Typ `ApplicationInformation`. Über dessen Methode `setApplicationRecoveryInterface(...)` wird die Recovery-Schnittstelle der Roboter-Applikation zur Verfügung gestellt. Durch Aufruf der Methode `setAssembly(...)` wird bekanntgegeben, wann der Fügeprozess ausgeführt wird.

```
// Annotation, required for all tasks which provide task functions
@ProvidedFunctions(IApplicationInformationFunction.class)

public class AssemblyApplication extends RoboticsAPIApplication
implements ITaskFunctionProvider {
    private Controller _sunrise_Cabinet;
    private LBR _lbr_iwa;

    /**
     * ITaskFunction instance with which
     * the task's functions are provided
     */
    private ApplicationInformation _function;

    public void initialize() {
        _sunrise_Cabinet = getController("KUKA_sunrise_Cabinet_1");
        _lbr_iwa = (LBR) getRobot(_sunrise_Cabinet,
        "LBR_iwa_7_R800_1");

        _function = new ApplicationInformation();
        // Gives access to recovery interface
        _function.setApplicationRecoveryInterface(getRecovery());
    }

    public void run() {
        // Moves robot to initial pose
        _lbr_iwa.move(ptp(getFrame("/StartPos")));
        // Announces that assembly is running
        _function.setAssemblyRunning(true);
        assembly();
        // Announces that assembly is finished
        _function.setAssemblyRunning(false);

        // Moves robot to initial pose
        _lbr_iwa.move(ptp(getFrame("/StartPos")));
    }

    /**
     * Implements the assembly process
     */
    private void assembly() {
        ...
    }

    public static void main(String[] args) {
        AssemblyApplication app = new AssemblyApplication();
        app.runApplication();
    }
}
```

```

    /**
     * ITaskFunctionProvider method that has to be
     * implemented by the task
     */
    @Override
    public Map<Class<? extends ITaskFunction>, ITaskFunction>
    createTaskFunctions() {
        // Creation of the map containing all ITaskFunction instances
        Map<Class<? extends ITaskFunction>, ITaskFunction> map = new
        HashMap<Class<? extends ITaskFunction>, ITaskFunction>();
        // Fill the map with functions and corresponding interface
        map.put(IApplicationInformationFunction.class, _function);
        // Return map which references the ITaskFunction instances
        return map;
    }
}

```

#### 16.4.4 Task-Funktionalitäten nutzen

Task-Funktionalitäten, die ein Task zur Verfügung stellt, können von anderen Tasks genutzt werden.

##### Zugriff ermöglichen

Der Zugriff auf die Task-Funktionalitäten erfolgt im anfragenden Task über ein Objekt vom Typ ITaskFunctionAccessor. Beim Erzeugen der ITaskFunctionAccessor-Instanz muss die von ITaskFunction abgeleitete Schnittstelle angegeben werden, die die benötigten Task-Funktionalitäten deklariert. Dafür wird der folgende Befehl verwendet:

```
ITaskFunctionAccessor< iTaskFunctionType> accessor =
getTaskFunction( iTaskFunctionType.class );
```

- *iTaskFunctionType*: Bezeichner der von ITaskFunction abgeleiteten Schnittstelle, die die deklarierten Task-Funktionalitäten enthält.
- *accessor*: Variable vom Typ ITaskFunctionAccessor<*iTaskFunctionType*.class>, die auf die erzeugte Instanz verweist.

##### Beispiel:

Im anfragenden Hintergrund-Task "LEDTask" soll der Zugriff auf die durch IApplicationInformationFunction definierten Funktionen ermöglicht werden. Die hierfür benötigte ITaskFunctionAccessor<IApplicationInformationFunction>-Instanz wird als Datenfeld angelegt und in der initialize()-Methode des Task erzeugt:

```

public class LEDTask extends RoboticsAPICyclicBackgroundTask {
    ...
    private ITaskFunctionAccessor<IApplicationInformationFunction>
    _accessor;

    public void initialize() {
        ...
        _accessor =
        getTaskFunction(IApplicationInformationFunction.class);
        ...
    }
}

```

##### Funktionalitäten nutzen

Über eine ITaskFunctionAccessor-Instanz können die Methoden der zugehörigen von ITaskFunctions abgeleiteten Schnittstelle genutzt werden. Über die ITaskFunctionAccessor-Methode *get()* erhält man zunächst Zugriff auf die ITaskFunction-Instanz des anbietenden Tasks. Über den Punktoperator stehen dann die einzelnen Methoden der Schnittstelle zur Verfügung.

```
iTaskFunctionType taskFunction = accessor.get();
```

- *iTaskFunctionType*: Bezeichner der von ITaskFunction abgeleiteten Schnittstelle, die die deklarierten Task-Funktionalitäten enthält.
- *taskFunction*: Variable vom Typ der Schnittstelle
- *accessor*: Variable vom Typ *ITaskFunctionAccessor<iTaskFunctionType>*, über die auf die durch die Schnittstelle definierten Task-Funktionalitäten zugegriffen werden kann.

#### Beispiel:

In der Methode runCyclic() des Hintergrund-Task "LEDTask" soll abgefragt werden, ob der Fügeprozess aktuell ausgeführt wird. Dafür bietet die Schnittstelle IApplicationInformationFunction die Methode isAssemblyRunning() an.

Die Methoden der Schnittstelle können unmittelbar hinter dem Aufruf von get() aufgerufen werden:

```
if (_accessor.get().isAssemblyRunning()) {
    ...
}
```

#### Verfügbarkeit prüfen

Die Task-Funktionalitäten des anbietenden Task stehen nur dann zur Verfügung, wenn eine Instanz dieses Task existiert. Beispielsweise stehen die von einer Roboter-Applikation bereitgestellten Task-Funktionalitäten nur zur Verfügung, wenn die Roboter-Applikation ausgeführt wird oder pausiert ist.



Wenn versucht wird, über die ITaskFunctionAccessor-Instanz auf nicht verfügbare Task-Funktionalitäten zuzugreifen, kommt es im anfragenden Task zu einem Laufzeitfehler. Wird dieser Fehler nicht behandelt, wird die Task-Ausführung abgebrochen.

Der Datentyp ITaskFunctionAccessor bietet folgende Methoden an, um zu prüfen, ob der anbietende Task verfügbar ist:

Methode	Beschreibung
isAvailable()	Rückgabetyp: boolean  Gibt an, ob die Task-Funktionalitäten des anbietenden Task verfügbar sind ( <b>true</b> = verfügbar).
await( <i>time</i> , <i>unit</i> )	Rückgabetyp: boolean  Stehen die Task-Funktionalitäten bei Aufruf des anbietenden Task nicht zur Verfügung, wird eine definierbare Zeit auf die Verfügbarkeit gewartet ( <b>true</b> = Task-Funktionalitäten innerhalb der definierten Wartezeit verfügbar).  Parameter: <ul style="list-style-type: none"> <li>■ <i>time</i> (Typ: long): Dauer der maximalen Wartezeit. Die Einheit wird durch den Parameter <i>unit</i> festgelegt.</li> <li>■ <i>unit</i> (Typ: TimeUnit): Einheit von <i>time</i></li> </ul>

#### Beispiel:

Der anfragende Hintergrund-Task "LEDTask" kann nur dann abfragen, ob der Fügeprozess ausgeführt wird, wenn die Roboter-Applikation läuft oder pausiert ist. Daher muss vor Aufruf von isAssemblyRunning() die Verfügbarkeit der Funktionalität geprüft werden:

```
if (_accessor.isAvailable()) {
    if (_accessor.get().isAssemblyRunning()) {
        ...
    }
}
```

## Gesamtbeispiel

Der anfragende Hintergrund-Task "LED Task" wird zyklisch alle 500 ms ausgeführt. Dabei prüft er zunächst, ob die benötigten Task-Funktionalitäten der Roboter-Applikation verfügbar sind. Wenn sie verfügbar sind, wird abgefragt, ob der Fügeprozess ausgeführt wird und entsprechend die zugehörige LED angesteuert. Anschließend wird abgefragt, ob ein Rückpositionieren des Roboters erforderlich ist. Ist dies der Fall, wird eine weitere LED angesteuert.

```
public class LEDTask extends RoboticsAPICyclicBackgroundTask {
    private Controller _sunrise_Cabinet;
    private ProcessParametersLEDsIOGroup _LED_IOs;

    /**
     * ITaskFunctionAccessor for functions defined by
     * IApplicationInformationFunction
     */
    private ITaskFunctionAccessor<IApplicationInformationFunction>
        _accessor;

    public void initialize() {
        _sunrise_Cabinet = getController("KUKA_Sunrise_Cabinet_1");
        initializeCyclic(0, 500, TimeUnit.MILLISECONDS,
            CycleBehavior.BestEffort);

        // Create ITaskFunctionAccessor for IApplicationInformation
        _accessor =
            getTaskFunction(IApplicationInformationFunction.class);

        // IOs for controlling LED-signals
        _LED_IOs = new ProcessParametersLEDsIOGroup(_sunrise_Cabinet);
    }

    public void runCyclic() {

        // Check if task functions are available
        if(_accessor.isAvailable()){
            /*
             * Use task function to check if assembly is
             * currently executed
             */
            if(_accessor.get().isAssemblyRunning()){
                /*
                 * If assembly is running, the appropriate LED changes
                 * its state with every execution of runCyclic()
                 */
                boolean currentStateAssemblyLED =
                    _LED_IOs.getLED_Assembly();
                _LED_IOs.setLED_Assembly(!currentStateAssemblyLED);

            } else{
                _LED_IOs.setLED_Assembly(false);
            }

            /*
             * Use task function to check whether the application
             * requires repositioning
             */
            boolean recoveryRequired =
                _accessor.get().isApplicationRecoveryRequired();

            if(recoveryRequired){
                /*

```

```
*If recovery is required, the appropriate LED changes
* its state with every execution of runCyclic()
*/
boolean currentStateRecoveryLED =
_LED_IOS.getLED_RecoveryRequired();
_LED_IOS.setLED_ForceExceeded(!currentStateRecoveryLED);
} else{
    _LED_IOS.setLED_RecoveryRequired(false);

}
} else{
    // If application is not running, LEDs remain off
    _LED_IOS.setLED_Assembly(false);
    _LED_IOS.setLED_RecoveryRequired(false);
}
}
}
```



# 17 Programmierung mit nachgiebigem Roboter

## 17.1 Sensorik und Regelung

Ein Standard-Industrieroboter kann ohne zusätzliche Hilfsmittel nur positionsgeregt betrieben werden. Ziel der Positionsregelung ist, die Differenz zwischen vorgegebener und tatsächlicher Roboterposition stets minimal zu halten.

Der KUKA LBR iiwa besitzt neben den Positionssensoren zur Ermittlung der aktuellen Gelenkstellung in jeder Achse Gelenkmomenten-Sensoren, mit denen das aktuelle Moment im Gelenk gemessen wird. Diese Daten ermöglichen zusätzlich zur Positionsregelung die Verwendung eines Impedanzreglers, durch den ein nachgiebiges Verhalten des Roboters realisiert werden kann. Das zugrundeliegende Modell ist ein virtuelles Feder-Dämpfer-System mit einstellbaren Werten für Steifigkeit und Dämpfung. Außerdem können zusätzlich Kräfte und Kraftschwingungen aufgeschaltet werden.

Durch die besondere Sensorik und die verfügbaren Reglermechanismen ist der KUKA LBR iiwa sensitiv und feinfühlig. Er kann sehr schnell auf auftretende Prozesskräfte reagieren und eignet sich besonders für unterschiedlichste Füge-Aufgaben sowie für die Zusammenarbeit mit dem Menschen.

## 17.2 Verfügbare Regler – Übersicht

Der KUKA LBR iiwa kann mit unterschiedlichen Reglern betrieben werden. Für jede Regelungsart stellt das RoboticsAPI im Paket com.kuka.roboticsAPI.motionModel.controlModeModel eine eigene Klasse zur Verfügung. Gemeinsame Oberklasse ist AbstractMotionControlMode.

Regler	Beschreibung
Positionsregler	Datentyp: PositionControlMode  Ziel der Positionsregelung ist es, die vorgegebene Bahn möglichst positionsgenau und ohne Bahnabweichung abzufahren. Einflüsse von außen, beispielsweise Hindernisse oder auftretende Prozesskräfte, werden defaultmäßig nicht berücksichtigt.
Kartesischer Impedanzregler	Datentyp: CartesianImpedanceControlMode  Der kartesische Impedanzregler basiert auf dem Modell eines virtuellen Feder-Dämpfer-Systems mit einstellbaren Werten für Steifigkeit und Dämpfung. Diese Feder wird zwischen der Soll- und Ist-Position des TCP aufgespannt. Dadurch reagiert der Roboter nachgiebig auf äußere Einflüsse.
Kartesischer Impedanzregler mit aufgeschalteter Kraftschwingung	Datentyp: CartesianSineImpedanceControlMode  Sonderform des kartesischen Impedanzreglers. Zusätzlich zum nachgiebigen Verhalten können konstante Sollkräfte und sinusförmige Kraftschwingungen aufgeschaltet werden. Mit diesem Regler können beispielsweise kraftabhängige Suchfahrten und Rüttelbewegungen für Fügeprozesse realisiert werden.
Achsspezifischer Impedanzregler	Datentyp: JointImpedanceControlMode  Der achsspezifische Impedanzregler basiert auf dem Modell eines virtuellen Feder-Dämpfer-Systems mit einstellbaren Werten für Steifigkeit und Dämpfung für jede Achse.

## 17.3 Regler in Roboter-Applikation verwenden

**Beschreibung** In Roboter-Applikationen wird der zu verwendende Regler für jeden Bewegungsbefehl eigens eingestellt. Hierfür sind defaultmäßig folgende Schritte erforderlich:

- Vorgehensweise**
1. Regler-Objekt des gewünschten Regler-Datentyps anlegen.
  2. Regler-Objekt parametrieren, um das Reglerverhalten festzulegen.
  3. Regler als Bewegungsparameter für einen Bewegungsbefehl einstellen.

### 17.3.1 Regler-Objekt anlegen

**Beschreibung** Um einen Regler verwenden zu können, muss zunächst eine Variable des gewünschten Regler-Datentyps angelegt und initialisiert werden. Das Regler-Objekt wird defaultmäßig mit dem Standardkonstruktor erzeugt.

**Syntax** *Reglermodus controlMode;*

```
controlMode = new Reglermodus();
```

**Erläuterung der Syntax**

Element	Beschreibung
<i>Reglermodus</i>	Datentyp des Reglers. Unterklasse von AbstractMotionControlMode.
<i>controlMode</i>	Name des Regler-Objekts

**Beispiel**

Anlegen eines kartesischen Impedanzreglers:

```
CartesianImpedanceControlMode cartImpCtrlMode;
cartImpCtrlMode = new CartesianImpedanceControlMode();
```

### 17.3.2 Regler-Parameter festlegen

Die einstellbaren Parameter sind abhängig vom Typ des verwendeten Reglers. Die einzelnen Regler-Klassen in der RoboticsAPI stellen für jeden Parameter eigene set- und get-Methoden zur Verfügung.

(>>> 17.5.2 "Parametrierung des kartesischen Impedanzreglers" Seite 430)

(>>> 17.6.3 "Parametrierung des Impedanzreglers mit aufgeschalteter Kraftschwingung" Seite 437)

(>>> 17.8 "Achsspezifischer Impedanzregler" Seite 447)

### 17.3.3 Regler-Objekt als Bewegungsparameter übergeben

**Beschreibung** Das Regler-Objekt wird einer Bewegung mit dem Befehl `setMode(...)` als Parameter übergeben. Wird einer Bewegung kein Regler-Objekt als Parameter übergeben, wird die Bewegung automatisch positionsgeregelt ausgeführt.



Bewegungen, die den kartesischen Impedanzregler verwenden, dürfen keine Posen enthalten, die sich in der Nähe singulärer Stellungen befinden.

**Syntax**

```
movableObject.move(motion.setMode(controlMode));
```

**Erläuterung der Syntax**

<b>Element</b>	<b>Beschreibung</b>
<i>motion</i>	Typ: Motion Auszuführende Bewegung
<i>controlMode</i>	Typ: UnterkLASSE von AbstractMotionControlMode Name des Regler-Objekts

## 17.4 Positionsregler

Bei der Positionsregelung werden die Motoren so angesteuert, dass die aktuelle Position des Roboters zu jedem Zeitpunkt mit der Sollposition, die von der Steuerung vorgegeben wird, bis auf eine minimale Differenz übereinstimmt. Der Positionsregler eignet sich insbesondere, wenn ein genaues Positionieren nötig ist.

Der Positionsregler wird durch die Klasse PositionControlMode repräsentiert. Der Datentyp besitzt keine einstellbaren Parameter zur Anpassung des Roboters.

Wird der Reglermodus einer Bewegung nicht explizit angegeben, wird der Positionsregler verwendet.

## 17.5 Kartesischer Impedanzregler

Der kartesische Impedanzregler wird durch die Klasse CartesianImpedanceControlMode repräsentiert.

Die Impedanzregelung bezieht sich defaultmäßig auf das Koordinatensystem, mit dem das Bewegungskommando ausgeführt wird.

Beispiele:

- `robot.move(...);`  
Impedanzregelung bezieht sich auf das Flansch-Koordinatensystem des Roboters.
- `gripper.move(...);`  
Impedanzregelung bezieht sich auf das aktuell verwendete Werkzeug-Koordinatensystem des Greifers oder auf den für den Greifer festgelegten Standard-Frame für Bewegungen.
- `gripper.getFrame("/TipCenter").move(...);`  
Impedanzregelung bezieht sich auf das Werkzeug-Koordinatensystem, das von dem Frame "TipCenter" am Greifer aufgespannt wird.

**Verhalten des Roboters**

In Impedanzregelung verhält sich der Roboter nachgiebig. Er ist feinfühlig und kann auf äußere Einflüsse wie Hindernisse oder Prozesskräfte reagieren. Der Roboter kann durch externe Krafteinwirkung von der geplanten Bahn abgebracht werden.

Das zugrundeliegende Modell sind virtuelle Federn und Dämpfer, die durch die Differenz zwischen aktuell gemessener und kommandierter Position des TCP aufgespannt werden. Die Eigenschaften der Federn werden durch Steifigkeiten und die der Dämpfer durch Dämpfungen beschrieben. Diese Parameter können für jede translatorische und rotatorische Dimension einzeln eingestellt werden.



Wenn der Roboter impedanzgeregt verfahren wird, kann die programmierte Roboterkonfiguration, z. B. der Status, nicht garantiert werden.

### 17.5.1 Berechnung der Kräfte nach dem Federgesetz

Stimmen gemessene und kommandierte Roboterposition überein, sind die virtuellen Federn entspannt. Da der Roboter sich nachgiebig verhält, kommt es bei externer Krafteinwirkung oder aufgrund eines Verfahrkommandos zu einer Abweichung zwischen Soll- und Istposition des Roboters. Dadurch ergibt sich eine Auslenkung der virtuellen Federn, was nach dem Federgesetz zu einer Kraft führt.

Die resultierende Kraft  $F$  lässt sich nach dem Federgesetz mit der eingestellten Federsteifigkeit  $C$  und der Auslenkung  $\Delta x$  berechnen:

$$F = C \cdot \Delta x$$

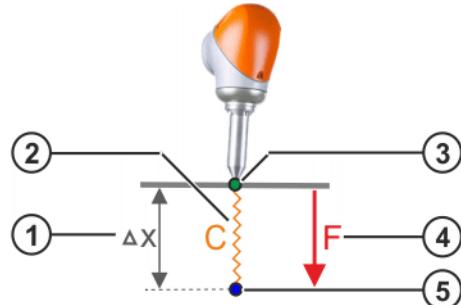


Abb. 17-1: Virtuelle Feder mit Federsteifigkeit C

- |                         |                           |
|-------------------------|---------------------------|
| 1 Auslenkung $\Delta x$ | 4 Resultierende Kraft $F$ |
| 2 Virtuelle Feder       | 5 Sollposition            |
| 3 Istposition           |                           |

Befindet sich der Roboter an einem Widerstand, übt er die berechnete Kraft aus. Befindet er sich im freien Raum, bewegt er sich zur kommandierten Position, wobei es auch hier aufgrund interner Reibungskräfte in den Gelenken zu Bahnabweichungen kommt, deren Ausmaß von der eingestellten Federsteifigkeit abhängt. Höhere Steifigkeiten führen zu geringeren Abweichungen.

Befindet sich der Roboter bereits an der kommandierten Position und wird eine externe Kraft auf das System ausgeübt, gibt er dieser so lange nach bis die Kräfte aus der Nachgiebigkeitsregelung die externen Kräfte ausgleichen.

#### Beispiele

Die Kraft, die bei Kontakt ausgeübt wird, ist abhängig von der Differenz zwischen Soll- und Istposition und der eingestellten Steifigkeit.

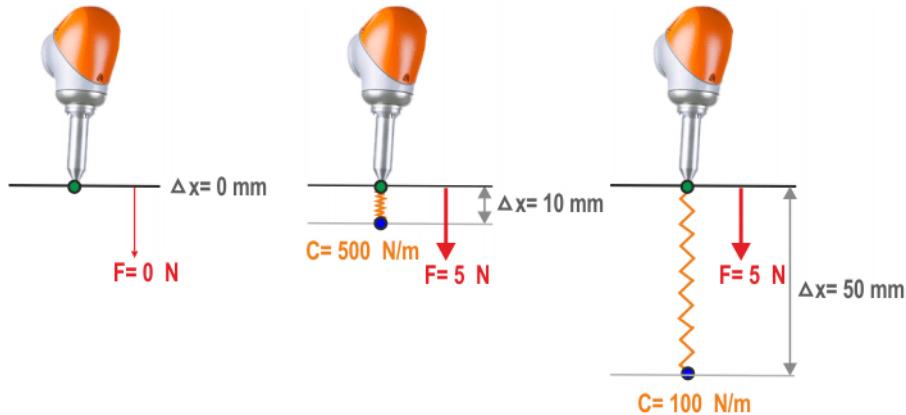


Abb. 17-2: Ausgeübte Kraft bei Kontakt

Wie die Abbildung zeigt (>>> Abb. 17-2 ), kann mit einer großen Positionsabweichung und geringer Steifigkeit die gleiche Kraft erzeugt werden wie mit einer kleineren Positionsabweichung und höherer Steifigkeit. Wird die Kraft durch eine Bewegung in einer Kontaktsituation aufgebaut, unterscheidet sich bei identi-

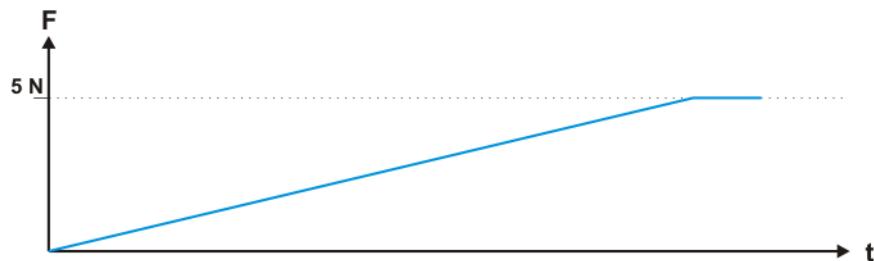
scher kartesischer Geschwindigkeit die Zeit, die benötigt wird um die Kraft zu erreichen.

Werden höhere Steifigkeiten verwendet, kann eine gewünschte Kraft früher erreicht werden, da nur eine kleine Positionsunterschied notwendig ist. Da die Sollposition schnell erreicht ist, kann auf diese Weise ein Ruck hervorgerufen werden.



**Abb. 17-3: Kraft im Zeitverlauf (Steifigkeit hoch, Positionsunterschied klein)**

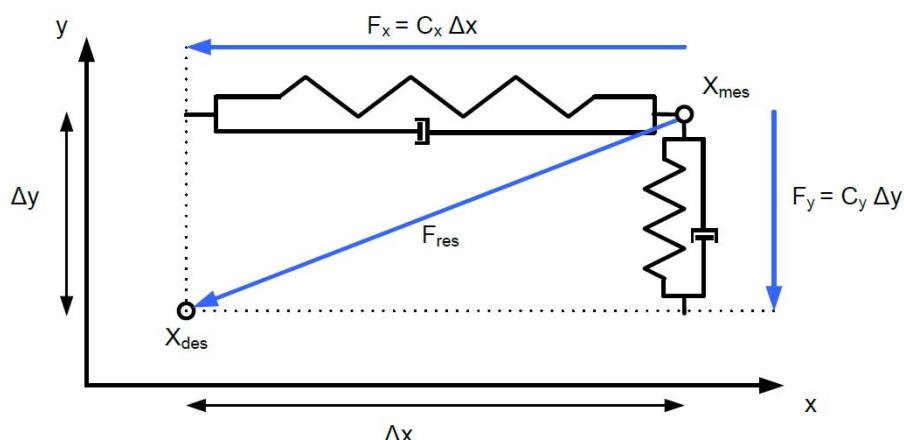
Bei großer Positionsunterschied und geringer Steifigkeit wird die Kraft langsamer aufgebaut. Dies kann z. B. genutzt werden, wenn der Roboter auf Kontakt fährt und die Belastungen beim Auftreffen reduziert werden sollen.



**Abb. 17-4: Kraft im Zeitverlauf (Steifigkeit gering, Positionsunterschied groß)**

Soll-Ist-Abweichungen in mehreren Richtungen führen zur Auslenkung aller betroffenen virtuellen Federn. Betrag und Richtung der Gesamtkraft ergeben sich aus Vektoraddition der Einzelkräfte für jede Richtung.

Die Auslenkung in x-Richtung um  $\Delta x$  und in y-Richtung um  $\Delta y$  führen zur Kraft  $F_x = C_x \Delta x$  in x-Richtung und  $F_y = C_y \Delta y$  in y-Richtung. Per Vektoraddition ergibt sich die Gesamtkraft zu  $F_{res}$ .



**Abb. 17-5: Gesamtkraft bei Auslenkung in 2 Richtungen**

## 17.5.2 Parametrierung des kartesischen Impedanzreglers

In Impedanzregelung verhält sich der Roboter wie eine Feder. Die Eigenschaften dieser Feder werden durch unterschiedliche Parameter beschrieben. Daraus ergibt sich das Verhalten des Roboters.

Mit einem kartesischen Impedanzregler können Kräfte für alle kartesischen Freiheitsgrade aufgeschaltet werden. Durch Kräfte, die um eine Achse wirken, wird ein Drehmoment erzeugt. Daher wird für die rotatorischen Freiheitsgrade nicht die aufgeschaltete Kraft, sondern das aufgeschaltete Moment angegeben. Zur Vereinfachung schließen im Folgenden die Begriffe "Kraft" und "Kraftschwingung" die Begriffe "Moment" und "Momentschwingung" für die rotatorischen Freiheitsgrade mit ein.



**VORSICHT** In Impedanzregelung können ungenaue Sensorinformationen oder falsch gewählte Parameter (z. B. fehlerhafte Lastdaten, falsches Werkzeug) als äußere Kräfte interpretiert werden und zu unvorhersehbaren Bewegungen des Roboters führen.

Folgende Regler-Eigenschaften können für jeden kartesischen Freiheitsgrad einzeln festgelegt werden:

- Steifigkeit
- Dämpfung
- Zusätzlich zur Feder aufzuwendende Kraft

Folgende Regler-Eigenschaften können unabhängig vom Freiheitsgrad festgelegt werden:

- Steifigkeit des Redundanz-Freiheitsgrades
- Dämpfung des Redundanz-Freiheitsgrades
- Begrenzung der maximalen Kraft am TCP
- Maximale kartesische Geschwindigkeit
- Maximale kartesische Bahnabweichung

### 17.5.2.1 Darstellung der kartesischen Freiheitsgrade

In der RoboticsAPI werden die Freiheitsgrade des kartesischen Impedanzreglers durch das Enum CartDOF (Paket com.kuka.roboticsAPI.geometricModel) repräsentiert. Mit den Werten dieses Enums kann entweder jeder Freiheitsgrad einzeln beschrieben werden oder die Kombination mehrerer Freiheitsgrade.

Enum-Wert	Beschreibung
CartDOF.X	Translatorischer Freiheitsgrad in X-Richtung
CartDOF.Y	Translatorischer Freiheitsgrad in Y-Richtung
CartDOF.Z	Translatorischer Freiheitsgrad in Z-Richtung
CartDOF.TRANSL	Kombination der translatorischen Freiheitsgrade in X-, Y- und Z-Richtung
CartDOF.A	Rotatorischer Freiheitsgrad um die Z-Achse
CartDOF.B	Rotatorischer Freiheitsgrad um die Y-Achse
CartDOF.C	Rotatorischer Freiheitsgrad um die X-Achse
CartDOF.ROT	Kombination der rotatorischen Freiheitsgrade um die Z-, Y- und X-Achse
CartDOF.ALL	Kombination aller kartesischen Freiheitsgrade

### 17.5.2.2 Regler-Parameter für einzelne Freiheitsgrade festlegen

**Beschreibung** Einige Parameter des kartesischen Impedanzreglers können für jeden kartesischen Freiheitsgrad einzeln definiert werden.

Bei der Programmierung wird zuerst angegeben für welche kartesischen Freiheitsgrade der Regler-Parameter gelten soll. Dafür wird die Methode parametrize(...) der Regler-Datentypen verwendet. Zur Festlegung der Freiheitsgrade werden dieser Methode ein oder mehrere Parameter vom Typ CartDOF übergeben.

Anschließend ruft man über den Punktoperator die set-Methode des gewünschten Regler-Parameters auf. Dieser Regler-Parameter wird für alle in parametrize(...) angegebenen Freiheitsgrade auf den als Eingangsparameter der set-Methode angegebenen Wert gesetzt.

**Syntax**

```
controlMode.parametrize(CartDOF.degreeOfFreedom_1
<, CartDOF.degreeOfFreedom_2, ...>).setParameter(value);
```

**Erläuterung der Syntax**

Element	Beschreibung
controlMode	Typ: CartesianImpedanceControlMode Name des Regler-Objekts
degreeOfFreedom_1, degreeOfFreedom_2, ...	Typ: CartDOF Aufzählung der Freiheitsgrade, die beschrieben werden sollen
setParameter(value)	Methode zum Setzen eines Regler-Parameters Für jeden einstellbaren Parameter steht eine eigene Methode zur Verfügung (value = Wert des Parameters).

**Beispiel**

Eine LIN-Bewegung zu einem definierten Punkt soll impedanzgeregelt ausgeführt werden. Der kartesische Impedanzregler ist so konfiguriert, dass der aktuell verwendete TCP, hier der Roboterflansch-Frame, in Z-Richtung nachgiebig ist.

```
CartesianImpedanceControlMode cartImpCtrlMode = new
CartesianImpedanceControlMode();

cartImpCtrlMode.parametrize(CartDOF.X,
CartDOF.Y).setStiffness(3000.0);
cartImpCtrlMode.parametrize(CartDOF.Z).setStiffness(1.0);
cartImpCtrlMode.parametrize(CartDOF.ROT).setStiffness(300.0);
cartImpCtrlMode.parametrize(CartDOF.ALL).setDamping(0.7);

lbr.move(lingetApplicationData().getFrame("/")
P1").setCartVelocity(800).setMode(cartImpCtrlMode));
```

### 17.5.2.3 Freiheitsgradspezifische Regler-Parameter

**Übersicht** Folgende Methoden stehen für die freiheitsgradspezifischen Parameter des kartesischen Impedanzreglers zur Verfügung:

Methode	Beschreibung
setStiffness(...)	<p>Federsteifigkeit (Typ: double)</p> <p>Die Federsteifigkeit bestimmt, wie stark der Roboter bei Krafteinwirkung nachgibt und dabei von seiner geplanten Bahn abweicht.</p> <p>Translatorische Freiheitsgrade (Einheit: N/m):</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 5000.0</b> Default: 2000.0</li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm/rad):</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 300.0</b> Default: 200.0</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Federsteifigkeit festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>
setDamping(...)	<p>Federdämpfung (Typ: double)</p> <p>Die Federdämpfung bestimmt, wie stark die virtuellen Federn nach Auslenkung schwingen.</p> <p>Für alle Freiheitsgrade (ohne Einheit: Lehrsches Dämpfungsmaß):</p> <ul style="list-style-type: none"> <li>■ <b>0.1 ... 1.0</b> Default: 0.7</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Federdämpfung festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>
setAdditionalControl-Force(...)	<p>Zusätzlich zur Feder wirkende Kraft (Typ: double)</p> <p>Durch die Zusatzkraft wird eine kartesische Kraft am TCP aufgebracht. Diese wirkt zusätzlich zu den Kräften, die aufgrund der Federsteifigkeit entstehen.</p> <p>Translatorische Freiheitsgrade (Einheit: N):</p> <ul style="list-style-type: none"> <li>■ Negative und positive Werte möglich. Default: 0.0</li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm):</p> <ul style="list-style-type: none"> <li>■ Negative und positive Werte möglich. Default: 0.0</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Zusatzkraft festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p> <p><b>Hinweis:</b> Die Kraft wird ohne Verzögerung aufgeschalten. Wenn die aufzuschaltende Kraft zu groß ist, kann dies zu einer Überlastung des Roboters und zu einem Programmabbruch führen. Die Klasse <code>CartesianSinelImpedanceControlMode</code> verfügt über die Möglichkeit Kräfte zeitlich verzögert aufzuschalten.</p>

#### 17.5.2.4 Freiheitsgradunabhängige Regler-Parameter

Einige Einstellungen gelten unabhängig von den kartesischen Freiheitsgraden. Die set-Methoden, mit denen diese Regler-Parameter festgelegt werden, gehören zur Klasse `CartesianImpedanceControlMode` und werden direkt auf dem Regler-Objekt aufgerufen.

##### Übersicht

Folgende Methoden stehen für die freiheitsgradunabhängigen Parameter des kartesischen Impedanzreglers zur Verfügung:

<b>Methode</b>	<b>Beschreibung</b>
setNullSpaceStiffness(...)	<p>Federsteifigkeit des Redundanz-Freiheitsgrades (Typ: double; Einheit: Nm/rad)</p> <p>Die Federsteifigkeit bestimmt, wie stark der Roboter bei Krafteinwirkung nachgibt und dabei von seiner geplanten Bahn abweicht.</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p><b>Hinweis:</b> Wird für den Redundanz-Freiheitsgrad keine Federsteifigkeit festgelegt, wird für diesen Freiheitsgrad ein Default-Wert verwendet.</p>
setNullSpaceDamping(...)	<p>Federdämpfung des Redundanz-Freiheitsgrades (Typ: double)</p> <p>Die Federdämpfung bestimmt, wie stark die virtuellen Federn nach Auslenkung schwingen.</p> <ul style="list-style-type: none"> <li>■ <b>0.3 ... 1.0</b></li> </ul> <p><b>Hinweis:</b> Wird für Redundanz-Freiheitsgrad keine Federdämpfung festgelegt, wird für diesen Freiheitsgrad ein Default-Wert verwendet.</p>
setMaxControlForce(...)	<p>Begrenzung der maximalen Kraft am TCP</p> <p>Die Kraft, die maximal am TCP durch die virtuellen Federn aufgebracht wird, wird begrenzt. Damit ist auch die Kraft, die maximal nötig ist, um die virtuelle Feder auszulenken, definiert. Zusätzlich wird festgelegt, ob die Bewegung bei Überschreiten der maximalen Kraft am TCP abgebrochen werden soll.</p> <p><b>Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <code>setMaxControlForce (maxForceX, maxForceY, maxForceZ, maxTorqueA, maxTorqueB, maxTorqueC, addStopCondition)</code></li> </ul> <p><b>Erläuterung der Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <i>maxForceX/Y/Z</i>: Maximale Kraft am TCP in der entsprechenden kartesischen Richtung (Typ: double, Einheit: N) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> <li>■ <i>maxTorqueA/B/C</i>: Maximales Moment am TCP in der entsprechenden rotatorischen Richtung (Typ: double, Einheit Nm) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> <li>■ <i>addStopCondition</i>: Abbruch der Bewegung bei Überschreiten der maximalen Kraft am TCP (Typ: boolean) <ul style="list-style-type: none"> <li>■ <b>true</b>: Bewegung wird abgebrochen.</li> <li>■ <b>false</b>: Bewegung wird nicht abgebrochen.</li> </ul> </li> </ul>

Methode	Beschreibung
setMaxCartesianVelocity(...)	<p>Maximale kartesische Geschwindigkeit Die Bewegung wird abgebrochen, wenn die festgelegte Geschwindigkeitsgrenze überschritten wird.</p> <p><b>Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <code>setMaxCartesianVelocity(maxVelocityX, maxVelocityY, maxVelocityZ, maxVelocityA, maxVelocityB, maxVelocityC)</code></li> </ul> <p><b>Erläuterung der Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <i>maxVelocityXYZ</i>: Maximale erlaubte translatorische Geschwindigkeit am TCP in der entsprechenden kartesischen Richtung (Typ: double, Einheit: mm/s) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> <li>■ <i>maxVelocityAIBIC</i>: Maximale erlaubte rotatorische Geschwindigkeit am TCP in der entsprechenden rotatorischen Richtung (Typ: double, Einheit: rad/s) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> </ul>
setMaxPathDeviation(...)	<p>Maximale kartesische Bahnabweichung Legt die kartesische Bahnabweichung von der aktuell geplanten Sollposition fest, die bei einer nachgiebigen Bewegung maximal auftreten darf. Die Bewegung wird abgebrochen, wenn die festgelegte maximale Bahnabweichung überschritten wird.</p> <p><b>Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <code>setMaxPathDeviation(maxDeviationX, maxDeviationY, maxDeviationZ, maxDeviationA, maxDeviationB, maxDeviationC)</code></li> </ul> <p><b>Erläuterung der Syntax:</b></p> <ul style="list-style-type: none"> <li>■ <i>maxDeviationXYZ</i>: Maximale erlaubte Bahnabweichung am TCP in der entsprechenden kartesischen Richtung (Typ: double, Einheit: mm) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> <li>■ <i>maxDeviationAIBIC</i>: Maximale erlaubte rotatorische Abweichung am TCP in der entsprechenden rotatorischen Richtung (Typ: double, Einheit: rad) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> </li> </ul>

**Beispiel 1**

Ein impedanzgeregelter Roboter soll sich in seinem redundanten Freiheitsgrad nachgiebig verhalten, um während der Bewegung auf Hindernisse reagieren zu können. Dafür werden Steifigkeit und Dämpfung des redundanten Freiheitsgrades für den Impedanzregler parametert.

```
CartesianImpedanceControlMode mode = new
CartesianImpedanceControlMode();

mode.setNullSpaceStiffness(10.0);
mode.setNullSpaceDamping(0.7);
```

**Beispiel 2**

Ein Roboter soll nachgiebig auf einer Tischplatte entlangfahren. Dafür wird ein kartesischer Impedanzregler parametert. In der Z-Richtung des Werkzeug-Koordinatensystems im TCP wird eine hohe Steifigkeit eingestellt. Außerdem soll eine zusätzliche Kraft von 20 N aufgebracht werden. Die Bewegung wird abgebrochen, wenn eine Kraftgrenze von 50 N in Z-Richtung überschritten wird. In der XY-Ebene wird eine geringe Steifigkeit eingestellt. Die kartesische Abweichung in X- und Y-Richtung darf jedoch nicht größer sein als 1 cm. Für alle weiteren Parameter werden geeignete größere Werte angegeben.

```

CartesianImpedanceControlMode mode = new
CartesianImpedanceControlMode();

mode.parametrize(CartDOF.Z).setStiffness(3000.0);
mode.parametrize(CartDOF.Z).setAdditionalControlForce(20.0);
mode.setMaxControlForce(100.0, 100.0, 50.0, 20.0, 20.0, 20.0, true);

mode.parametrize(CartDOF.X, CartDOF.Y).setStiffness(10.0);
mode.setMaxPathDeviation(10.0, 10.0, 50.0, 2.0, 2.0, 2.0);

```

## 17.6 Kartesischer Impedanzregler mit aufgeschalteter Kraftschwingung

Der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung ist eine Sonderform des kartesischen Impedanzreglers. Die Kraftaufschaltung ist für jeden kartesischen Freiheitsgrad einzeln möglich.

Kraftschwingungen um eine Achse erzeugen Drehmoment-Schwingungen. Durch Aufschalten von Momentschwingungen können Drehschwingungen erzeugt werden.

Das Aufschalten konstanter oder sinusförmiger Kräfte führt zu einer Bewegung des Roboters. Durch geeignete Kombination der Schwingungen in den einzelnen Freiheitsgraden können unterschiedliche Bewegungsmuster erzeugt werden.

Mithilfe von aufgeschalteten Schwingungen lassen sich z. B. nachgiebige Pendelbewegungen für Suchfahrten und Vibrationen im Werkzeug für Fügeprozesse realisieren.

Der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung wird durch die Klasse `CartesianSinelImpedanceControlMode` repräsentiert.

### Verhalten des Roboters

Bei dieser Form der Impedanzregelung wird der Roboter durch das Aufschalten einer Kraft gezielt von der geplanten Bahn abgebracht. Der neue Bahnverlauf wird dabei durch eine Vielzahl an unterschiedlichen Parametern bestimmt.

Zusätzlich zu Steifigkeit und Dämpfung sind noch weitere Parameter definierbar, z. B. die Frequenz und eine Amplitude. Auch die programmierte Geschwindigkeit des Roboters spielt für den konkreten Bahnverlauf eine wesentliche Rolle.



Das Aufschalten zusätzlicher Kräfte hat einen starken Einfluss auf die Roboterbewegung und die vom Roboter ausgeübten Kräfte. Beispielsweise kann es bei geringer Steifigkeit und hohen aufgeschalteten Kräften zu einer starken Beschleunigung des Roboters kommen. Wird mit Kraftaufschaltungen gearbeitet, muss daher mit Vorsicht parametriert werden. Beispielsweise indem man mit dem Aufschalten geringer Kräfte beginnt und sich schrittweise den passenden Kraftwerten nähert. Zusätzlich muss die Bewegung, die aus der aufgeschalteten Kraft entsteht, immer zuerst in der Betriebsart T1 getestet werden.

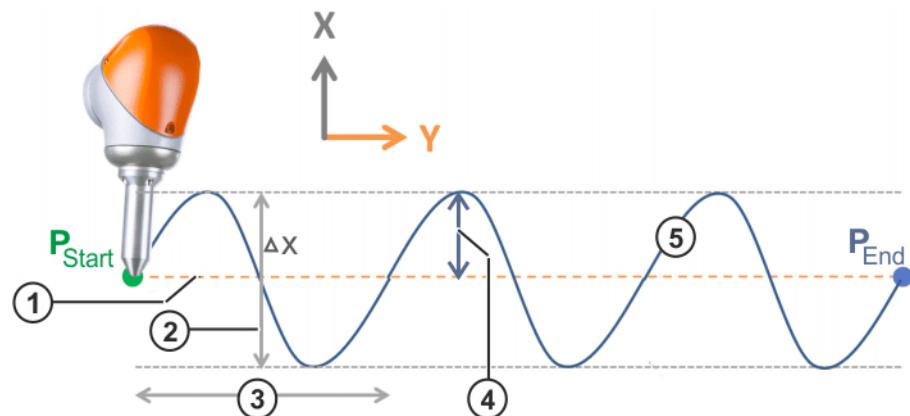
### 17.6.1 Aufschalten einer einfachen Kraftschwingung

Durch das Aufschalten einer einfachen Kraftschwingung wird der Arbeitspunkt von der geplanten Bahn (= Bahnverlauf ohne aufgeschaltete Schwingungen) abgelenkt und stattdessen in einer Wellenbahn vom Start- zum Endpunkt der Bewegung geführt.

#### Beispiel

Der Roboter führt eine relative Bewegung in Y-Richtung des Werkzeug-Koordinatensystems im TCP aus. Dabei wird eine sinusförmige Kraftschwingung

in X-Richtung aufgeschaltet. Das Ergebnis ist eine wellenförmige Bahn in der XY-Ebene des Koordinatensystems.



**Abb. 17-6: Aufschalten einer einfachen Kraftschwingung**

- |                         |             |
|-------------------------|-------------|
| 1 Ursprüngliche Bahn    | 4 Amplitude |
| 2 Auslenkung $\Delta x$ | 5 Neue Bahn |
| 3 Wellenlänge           |             |

Die maximale Auslenkung  $\Delta x$  ist die Abweichung von der Ursprungsbahn in die positive und negative X-Richtung. Die maximale Auslenkung wird durch die Steifigkeit und die Amplitude bestimmt, die für den Impedanzregler in der kartesischen X-Richtung definiert sind, z. B.:

- Kartesische Steifigkeit:  $C = 500 \text{ N/m}$
- Amplitude:  $F = 5 \text{ N}$

Die maximale Auslenkung ergibt sich aus dem Federgesetz:

$$\Delta x = F / C = 5 \text{ N} / (500 \text{ N/m}) = 1 / (100 \text{ 1/m}) = 1 \text{ cm}$$

Über die Wellenlänge kann festgelegt werden, wie viele Schwingungen der Roboter zwischen dem Start- und dem Endpunkt der Bewegung ausführen soll. Die Wellenlänge wird durch die Frequenz bestimmt, die für den Impedanzregler mit aufgeschalteter Kraftschwingung definiert ist, sowie durch die programmierte Robotergeschwindigkeit.

Die Wellenlänge  $\lambda$  lässt sich wie folgt berechnen:

$$\lambda = c / f = \text{Robotergeschwindigkeit} / \text{Frequenz}$$

### 17.6.2 Aufschalten überlagerter Kraftschwingungen (Lissajous-Figuren)

Lissajous-Figuren entstehen, indem eine sinusförmige Kraftschwingung in 2 verschiedenen kartesischen Richtungen aufgeschaltet wird. Durch die Überlagerung der beiden Schwingungen können die unterschiedlichsten Formen für den Bahnverlauf erzeugt werden. Der exakte Bahnverlauf hängt von einer Vielzahl von Parametern ab.

#### Anwendung

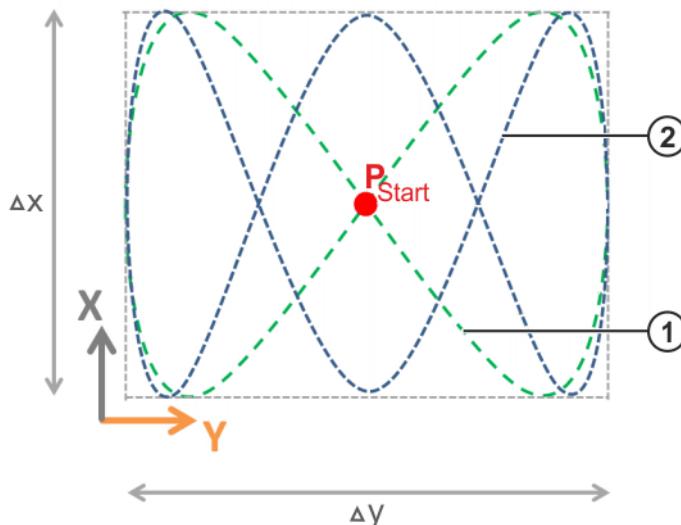
Mit der Überlagerung von 2 sinusförmigen Kraftschwingungen, die sich in der Frequenz unterscheiden, können Vibrationen am TCP erzeugt werden. Wenn beispielsweise während eines Montageprozesses Verspannungen und Verklemmungen auftreten, können diese durch die so erzeugten Vibrationen gelöst werden.

#### Beispiel

Es wird je eine sinusförmige Kraftschwingung in die X-Richtung und die Y-Richtung des Werkzeug-Koordinatensystems im TCP aufgeschaltet. Die maximalen Auslenkungen  $\Delta x$  und  $\Delta y$  werden durch die Steifigkeit und die Ampli-

tude bestimmt, die für den Impedanzregler in der kartesischen X- und Y-Richtung definiert sind.

Zusätzlich zu den bereits bekannten Parametern des Impedanzreglers spielt für den Bahnverlauf die Phasenverschiebung zwischen den beiden Schwingungen eine wesentliche Rolle.



**Abb. 17-7: Bahnverlauf bei einer Lissajous-Figur**

- 1 Bahnverlauf ohne Phasenverschiebung (Verhältnis Frequenz X:Y = 2:1)
- 2 Bahnverlauf mit Phasenverschiebung (Verhältnis Frequenz X:Y = 3:1)

Die Form des Bahnverlaufs wird hauptsächlich von dem Verhältnis der beiden Frequenzen und der Phasenverschiebung zwischen den beiden Schwingungen bestimmt. Die entstehende Figur ist immer achsen- und punktsymmetrisch. Aus der eingestellten Kraftamplitude und Steifigkeit für eine Schwingungsrichtung ergibt sich deren Positionsamplitude. Das Verhältnis der beiden Positionsamplituden bestimmt das Verhältnis von Breite zu Höhe der Figur.

### 17.6.3 Parametrierung des Impedanzreglers mit aufgeschalteter Kraftschwingung

Der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung ist eine Sonderform des Standard-Impedanzreglers.

Mit einem kartesischen Impedanzregler mit aufgeschalteter Kraftschwingung können Kräfte für alle kartesischen Freiheitsgrade aufgeschaltet werden. Durch Kräfte, die um eine Achse wirken, wird ein Drehmoment erzeugt. Daher wird für die rotatorischen Freiheitsgrade nicht die aufgeschaltete Kraft, sondern das aufgeschaltete Moment angegeben. Zur Vereinfachung schließen im Folgenden die Begriffe "Kraft" und "Kraftschwingung" die Begriffe "Moment" und "Momentschwingung" für die rotatorischen Freiheitsgrade mit ein.



In Impedanzregelung können ungenaue Sensorinformationen oder falsch gewählte Parameter (z. B. fehlerhafte Lastdaten, falsches Werkzeug) als äußere Kräfte interpretiert werden und zu unvorhersehbaren Bewegungen des Roboters führen.

Der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung wird analog zum Standard-Impedanzregler parametriert. Die für den Standard-Impedanzregler beschriebenen freiheitsgradspezifischen und freiheitsgradunabhängigen Regler-Parameter können auf die gleiche Weise für den Impedanzregler mit aufgeschalteter Kraftschwingung verwendet werden.

(>>> 17.5.2 "Parametrierung des kartesischen Impedanzreglers" Seite 430)



Ausnahme: Die Methode `setAdditionalControlForce(...)` der Klasse `CartesianImpedanceControlMode` zum Aufschalten einer zusätzlich zur Feder aufzuwendenden Kraft steht zwar für die Klasse `CartesianSinelImpedanceControlMode` zur Verfügung, sollte jedoch nicht verwendet werden.  
Um konstante Kräfte aufzuschalten, steht in der Klasse `CartesianSinelImpedanceControlMode` die Methode `setBias(...)` zur Verfügung.

Folgende zusätzlichen Regler-Eigenschaften können für jeden kartesischen Freiheitsgrad einzeln festgelegt werden:

- Amplitude der Kraftschwingung
- Frequenz der Kraftschwingung
- Phasenverschiebung der Kraftschwingung
- Überlagerte konstante Kraft
- Kraftbegrenzung der Kraftschwingung
- Begrenzung der Auslenkung durch die Kraftschwingung

Folgende zusätzlichen Regler-Eigenschaften können unabhängig vom Freiheitsgrad festgelegt werden:

- Anstiegszeit der Kraftschwingung
- Haltezeit der Kraftschwingung
- Abfallzeit der Kraftschwingung
- Gesamtdauer der Kraftschwingung

#### 17.6.3.1 Freiheitsgradspezifische Regler-Parameter

##### Übersicht

Folgende Methoden stehen für die freiheitsgradspezifischen Parameter des kartesischen Impedanzreglers mit aufgeschalteter Kraftschwingung zur Verfügung:

Methode	Beschreibung
<code>setAmplitude(...)</code>	<p>Amplitude der Kraftschwingung (Typ: double) Amplitude und Steifigkeit bestimmen die Positionsamplitude. Translatorische Freiheitsgrade (Einheit: N):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: 0.0</li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: 0.0</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Amplitude festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>
<code>setFrequency(...)</code>	<p>Frequenz der Kraftschwingung (Typ: double; Einheit: Hz) Frequenz und kartesische Geschwindigkeit bestimmen die Wellenlänge der Kraftschwingung.</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 15.0</b> Default: 0.0</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Frequenz festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>

<b>Methode</b>	<b>Beschreibung</b>
setPhaseDeg(...)	<p>Phasenverschiebung der Kraftschwingung zu Beginn der Kraftaufschaltung (Typ: double; Einheit: °)</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Default: 0.0</p> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Phasenverschiebung festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>
setBias(...)	<p>Aufgeschaltete konstante Kraft (Typ: double)</p> <p>Mithilfe von setBias(...) kann zusätzlich zur aufgeschalteten Kraftschwingung eine konstante Kraft aufgeschaltet werden. Diese addiert sich zu der Kraft, die aufgrund der Federsteifigkeit und der festgelegten Kraftschwingung entsteht.</p> <p>Wird eine konstante Kraft ohne zusätzliche Kraftschwingung aufgeschaltet, ergibt sich ein Kraftverlauf der abhängig von der mit setRiseTime(...) definierten Anstiegszeit ansteigt und anschließend konstant bleibt. setRiseTime(...) gehört zu den freiheitsgradunabhängigen Regler-Parametern (<b>&gt;&gt;&gt;</b> 17.6.3.1 "Freiheitsgradspezifische Regler-Parameter" Seite 438).</p> <p>Wird eine konstante Kraft zusätzlich zu einer Kraftschwingung aufgeschaltet, verschiebt sich die Kraftschwingung in der definierten Richtung.</p> <p>Translatorische Freiheitsgrade (Einheit: N):</p> <ul style="list-style-type: none"> <li>■ Negative und positive Werte möglich.</li> </ul> <p>Default: 0.0</p> <p>Rotatorische Freiheitsgrade (Einheit: Nm):</p> <ul style="list-style-type: none"> <li>■ Negative und positive Werte möglich.</li> </ul> <p>Default: 0.0</p> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine zusätzliche konstante Kraft aufgeschaltet, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>

Methode	Beschreibung
setForceLimit(...)	<p>Kraftbegrenzung der Kraftschwingung (Typ: double)</p> <p>Legt den Grenzwert fest, den die Gesamtkraft, d. h. die Summe aus Amplitude der Kraftschwingung und zusätzlich aufgeschalteter konstanter Kraft, nicht überschreiten darf. Überschreitet die Gesamtkraft den Grenzwert, wird die aufgeschaltete Kraft auf den Grenzwert reduziert.</p> <p>Translatorische Freiheitsgrade (Einheit: N):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: Nicht begrenzt.</li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: Nicht begrenzt.</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine Kraftbegrenzung festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>
setPositionLimit(...)	<p>Maximale Auslenkung durch die Kraftschwingung (Typ: double)</p> <p>Bei Überschreiten der maximal zulässigen Auslenkung wird die Kraft abgeschaltet. Sobald sich der Roboter wieder im zulässigen Bereich befindet, wird die Kraft wieder aufgeschaltet.</p> <p>Translatorische Freiheitsgrade (Einheit: mm):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: Nicht begrenzt.</li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: rad):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b> Default: Nicht begrenzt.</li> </ul> <p><b>Hinweis:</b> Wird für einen Freiheitsgrad keine maximale Auslenkung festgelegt, wird für diesen Freiheitsgrad der Default-Wert verwendet.</p>

**Beispiel**

Während eines Fügeprozesses soll eine Schwingung um die Z-Achse des Werkzeug-Koordinatensystems im TCP ausgeführt werden. Dafür wird der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung verwendet. Bei einer Steifigkeit von 10 Nm/rad und einer Amplitude von 15 Nm beträgt die Positionsamplitude ca. 1.5 rad. Die Frequenz wird auf 5 Hz gesetzt. Um eine zusätzliche Druckkraft in Bewegungsrichtung auszuüben, wird in Z-Richtung eine konstante Kraft von 5 N erzeugt, die die um die Z-Achse aufgeschaltete Kraftschwingung überlagert.

```
CartesianSineImpedanceControlMode sineMode = new
CartesianSineImpedanceControlMode();

sineMode.parametrize(CartDOF.Z).setStiffness(4000.0);
sineMode.parametrize(CartDOF.Z).setBias(5.0);

sineMode.parametrize(CartDOF.A).setStiffness(10.0);
sineMode.parametrize(CartDOF.A).setAmplitude(15.0);
sineMode.parametrize(CartDOF.A).setFrequency(5.0);

tool.getFrame("/TCP").move(linRel(0.0, 0.0,
10.0).setCartVelocity(10.0).sineMode(sineMode));
```

**17.6.3.2 Freiheitsgradunabhängige Regler-Parameter**

Einige Einstellungen gelten unabhängig von den kartesischen Freiheitsgraden. Die set-Methoden, mit denen diese Regler-Parameter festgelegt werden,

gehören zur Klasse `CartesianSinelImpedanceControlMode` und werden direkt auf dem Regler-Objekt aufgerufen.

## Übersicht

Folgende Methoden stehen für die freiheitsgradunabhängigen Parameter des kartesischen Impedanzreglers mit aufgeschalteter Kraftschwingung zur Verfügung:

Methode	Beschreibung
<code>setTotalTime(...)</code>	Gesamtdauer der Kraftschwingung (Typ: double; Einheit: s) <b>(&gt;&gt;&gt; "Gesamtdauer der Kraftschwingung" Seite 441)</b> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Default: Unbegrenzt
<code>setRiseTime(...)</code>	Anstiegszeit der Kraftschwingung (Typ: double; Einheit: s) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Default: 0.0 <p><b>Hinweis:</b> Wird keine Anstiegszeit festgelegt, wird der Default-Wert verwendet. Das bedeutet, die Amplitude steigt übergangslos und ruckartig auf den festgelegten Wert an. Wenn die aufzuschaltende Kraft zu groß ist, kann dies zu einer Überlastung des Roboters und zu einem Programmabbruch führen.</p>
<code>setHoldTime(...)</code>	Haltezeit der Kraftschwingung (Typ: double; Einheit: s) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Default: Unbegrenzt <p><b>Hinweis:</b> Wird keine Haltezeit festgelegt, wird der Default-Wert verwendet. Das bedeutet, die aufgeschaltete Kraftschwingung endet mit der zugehörigen Bewegung.</p>
<code>setFallTime(...)</code>	Abfallzeit der Kraftschwingung (Typ: double; Einheit: s) <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> Default: 0.0 <p><b>Hinweis:</b> Wird keine Abfallzeit festgelegt, wird der Default-Wert verwendet. Das bedeutet, die Amplitude fällt übergangslos und ruckartig auf Null ab. Wenn der Kraftabfall zu groß ist, kann dies zu einer Überlastung des Roboters und zu einem Programmabbruch führen.</p>
<code>setStayActiveUntil-PatternFinished(...)</code>	Verhalten bei Überschreiten der Bewegungsdauer (Typ: boolean) Für den Fall, dass die Kraftschwingung länger dauert als die Bewegung, kann festgelegt werden, ob die Schwingung nach Bewegungsende abgebrochen oder fortgesetzt wird. <ul style="list-style-type: none"> <li>■ <b>true:</b> Schwingung wird nach Bewegungsende fortgesetzt.</li> <li>■ <b>false:</b> Schwingung wird mit Bewegungsende abgebrochen.</li> </ul> Default: false <p><b>Hinweis:</b> Wird das Verhalten bei Überschreiten der Bewegungsdauer nicht festgelegt, wird der Default-Wert verwendet.</p>

## Gesamtdauer der Kraftschwingung

Die Gesamtdauer ist die Summe aus Anstiegszeit, Haltezeit und Abfallzeit der Kraftschwingung:

- **Anstiegszeit**  
Zeit, in der die Amplitude der Kraftschwingung aufgebaut wird
- **Haltezeit**  
Zeit, in der die Kraftschwingung mit der festgelegten Amplitude ausgeführt wird

- Abfallzeit

Zeit, in der die Amplitude der Kraftschwingung wieder auf Null abgebaut wird

Anstiegszeit, Haltezeit und Abfallzeit der Kraftschwingung können einzeln festgelegt werden, oder indirekt, indem die Gesamtdauer der Kraftschwingung definiert wird.

Wird die Gesamtdauer über `setTotalTime(...)` definiert, werden Anstiegs- und Abfallzeit automatisch festgelegt.

Berechnung:

- Anstiegszeit = Abfallzeit =  $(1/\text{Frequenz}) \cdot 0.5$
- Unter den für die Kraftschwingung definierten Frequenzen (bezogen auf alle Freiheitsgrade), wird diejenige Frequenz für die Berechnung verwendet, die die größtmögliche Anstiegs- und Abfallzeit ergibt.
- Werden ausschließlich konstante Kräfte aufgeschaltet, beträgt die Frequenz aller Freiheitsgrade 0.0 Hz. Anstiegs- und Abfallzeit werden auf 0.0 s eingestellt.
- Übersteigt die so berechnete Summe aus Anstiegs- und Abfallzeit die festgelegte Gesamtdauer, werden Anstiegs- und Abfallzeit auf jeweils 25 % und die Haltezeit auf 50 % der Gesamtzeit gesetzt.

Ist die Gesamtdauer der Kraftschwingung kürzer als die Dauer der zugehörigen Bewegung, endet die Kraftschwingung vor Bewegungsende. Das Verhalten bei Überschreiten der Bewegungsdauer wird über `setStayActiveUntilPatternFinished(...)` festgelegt.

## 17.7 Statische Methoden für Impedanzregler mit überlagerter Kraftschwingung

### Übersicht

Der kartesische Impedanzregler mit aufgeschalteter Kraftschwingung kann auch über statische Methoden der Klasse `CartesianSineImpedanceControlMode` konfiguriert werden. Dies erleichtert die Programmierung, insbesondere bei Lissajous-Figuren, da nur wenige Parameter durch den Anwender angegeben werden müssen. Die übrigen für die Umsetzung wichtigen Parameter werden automatisch berechnet und gesetzt. Für alle weiteren Parameter werden Default-Werte verwendet. Zusätzliche Einstellungen werden wie beschrieben über die `parametrize(...)`-Funktion und die `set-`Methoden von `CartesianSineImpedanceControlMode` durchgeführt.

- `createDesiredForce(...)`: Statische Methode für konstante Kraft
- `createSinePattern(...)`: Statische Methode für einfache Kraftschwingungen
- `createLissajousPattern(...)`: Statische Methode für Lissajous-Figuren
- `createSpiralPattern(...)`: Statische Methode für Spiralen

### Angabe kartesische Ebenen

Bei Lissajous-Figuren und Spiralen wird im Gegensatz zu einfachen Schwingungen kein einzelner Freiheitsgrad übergeben, sondern die Ebene, in der die Bahn verlaufen soll. Die Ebene wird über das Enum `CartPlane` angegeben (Paket `com.kuka.roboticsAPI.geometricModel`).

Enum-Wert	Beschreibung
<code>CartPlane.XY</code>	Bahnverlauf in der XY-Ebene
<code>CartPlane.XZ</code>	Bahnverlauf in der XZ-Ebene
<code>CartPlane.YZ</code>	Bahnverlauf in der YZ-Ebene

### 17.7.1 Konstante Kraft aufschalten

**Beschreibung** Mit der Methode `createDesiredForce(...)` wird in einer kartesischen Richtung eine konstante, zeitlich nicht veränderliche Kraft aufgeschaltet.

**Syntax**

```
controlMode = CartesianSineImpedanceControlMode.createDesiredForce(CartDOF.degreeOfFreedom, force, stiffness);
```

**Erläuterung der Syntax**

Element	Beschreibung
<code>controlMode</code>	Typ: <code>CartesianSineImpedanceControlMode</code> Name des Regler-Objekts
<code>degreeOfFreedom</code>	Typ: <code>CartDOF</code> Freiheitsgrad, für den die konstante Kraft aufgeschaltet werden soll
<code>force</code>	Typ: <code>double</code> Wert der aufgeschalteten konstanten Kraft. Entspricht dem Aufruf von <code>setBias(...)</code> für den angegebenen Freiheitsgrad. Translatorische Freiheitsgrade (Einheit: N): ■ <b>≥ 0.0</b> Rotatorische Freiheitsgrade (Einheit: Nm): ■ <b>≥ 0.0</b>
<code>stiffness</code>	Typ: <code>double</code> Steifigkeitswert für den angegebenen Freiheitsgrad Translatorische Freiheitsgrade (Einheit: N/m): ■ <b>0.0 ... 5000.0</b> Rotatorische Freiheitsgrade (Einheit: Nm/rad): ■ <b>0.0 ... 300.0</b>

### 17.7.2 Einfache Kraftschwingung aufschalten

**Beschreibung** Mit der Methode `createSinePattern(...)` wird in einer kartesischen Richtung eine einfache Kraftschwingung aufgeschaltet.

**Syntax**

```
controlMode = CartesianSineImpedanceControlMode.createSinePattern(CartDOF.degreeOfFreedom, frequency, amplitude, stiffness);
```

**Erläuterung der Syntax**

Element	Beschreibung
<code>controlMode</code>	Typ: <code>CartesianSineImpedanceControlMode</code> Name des Regler-Objekts
<code>degreeOfFreedom</code>	Typ: <code>CartDOF</code> Freiheitsgrad, für den die Kraftschwingung aufgeschaltet werden soll
<code>frequency</code>	Typ: <code>double</code> Frequenz der Schwingung (Einheit: Hz) ■ <b>0.0 ... 15.0</b>

Element	Beschreibung
<i>amplitude</i>	<p>Typ: double</p> <p>Amplitude der Schwingung, die in Richtung des angegebenen Freiheitsgrads aufgeschaltet wird</p> <p>Translatorische Freiheitsgrade (Einheit: N):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm):</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul>
<i>stiffness</i>	<p>Typ: double</p> <p>Steifigkeitswert für den angegebenen Freiheitsgrad</p> <p>Translatorische Freiheitsgrade (Einheit: N/m):</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 5000.0</b></li> </ul> <p>Rotatorische Freiheitsgrade (Einheit: Nm/rad):</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 300.0</b></li> </ul>

**Beispiel**

Von der aktuellen Position aus soll eine relative Bewegung von 15 cm in Y-Richtung ausgeführt werden. Die Bewegung soll in einer Wellenbahn mit einer Auslenkung von ca. 10 cm (ergibt sich aus Amplitude und Steifigkeit) und einer Frequenz von 2 Hz in X-Richtung verlaufen.

```
CartesianSineImpedanceControlMode sineMode;

sineMode =
CartesianSineImpedanceControlMode.createSinePattern(CartDOF.X, 2.0,
50.0, 500.0);

lbr.move(linRel(0.0, 150.0,
0.0).setCartVelocity(100).setMode(sineMode));
```

**17.7.3 Lissajous-Schwingung aufschalten****Beschreibung**

Mit der Methode `createLissajousPattern(...)` wird eine 2-dimensionale Schwingung in einer Ebene erzeugt. Die Ebene wird als Wert vom Typ `CartPlane` übergeben. Die weiteren übergebenen Parameter beziehen sich auf den ersten Freiheitsgrad der angegebenen Ebene (Beispiel: Für `CartPlane.XY` beziehen sich die Angaben auf `CartDOF.X`).

Die Parameter des zweiten Freiheitsgrades der Ebene werden so berechnet, dass das Ergebnis eine Lissajous-Figur mit folgenden Eigenschaften ist:

- Verhältnis Amplitude 1. Freiheitsgrad : 2. Freiheitsgrad: 1 : 1
- Verhältnis Frequenz 1. Freiheitsgrad : 2. Freiheitsgrad: 1 : 0.4
- Phasenverschiebung zwischen 1. und 2. Freiheitsgrad:  $\frac{1}{2} \cdot \pi$

**Syntax**

```
controlMode = CartesianSineImpedanceControlMode.createLissajousPattern(CartPlane.plane, frequency, amplitude, stiffness);
```

## Erläuterung der Syntax

Element	Beschreibung
<i>controlMode</i>	Typ: CartesianSineImpedanceControlMode Name des Regler-Objekts
<i>plane</i>	Typ: Enum vom Typ CartPlane Ebene, in der die Lissajous-Schwingung aufgeschaltet werden soll
<i>frequency</i>	Typ: double Frequenz der Schwingung für den ersten Freiheitsgrads der angegebenen Ebene (Einheit: Hz) <ul style="list-style-type: none"> <li>■ <b>0.0 ... 15.0</b></li> </ul> <p>Die Frequenz für den zweiten Freiheitsgrad wird wie folgt berechnet:</p> <ul style="list-style-type: none"> <li>■ <math>frequency \cdot 0.4</math></li> </ul>
<i>amplitude</i>	Typ: double Amplitude der Schwingung für beide Freiheitsgrade der angegebenen Ebene (Einheit: N) <ul style="list-style-type: none"> <li>■ <b><math>\geq 0.0</math></b></li> </ul>
<i>stiffness</i>	Typ: double Steifigkeitswert für beide Freiheitsgrade der angegebenen Ebene (Einheit: N/m) <ul style="list-style-type: none"> <li>■ <b>0.0 ... 5000.0</b></li> </ul>

## Beispiel

Am Roboterflansch soll eine Schwingung in Form einer Lissajous-Figur mit einem Frequenzverhältnis X : Y von 1 : 0.4 und einer Phasenverschiebung in Y von  $\pi/2$  erzeugt werden. Bahnverlauf mit Phasenverschiebung (= blaue Linie (**>>> Abb. 17-7**)).

```
CartesianSineImpedanceControlMode lissajousMode;

lissajousMode =
CartesianSineImpedanceControlMode.createLissajousPattern(CartPlane.XY
, 10.0, 50.0, 500.0);

lbr.move(linRel(0.0, 150.0,
0.0).setCartVelocity(100).setMode(lissajousMode));
```

## 17.7.4 Spiralförmige Kraftschwingung aufschalten

### Beschreibung

Mit der Methode `createSpiralPattern(...)` wird eine spiralförmige Kraftschwingung in einer Ebene erzeugt.

Der Kraftverlauf wird durch die Überlagerung von 2 sinusförmigen Schwingungen erzeugt. Die Schwingungen sind zueinander um  $\pi/2$  ( $90^\circ$ ) phasenverschoben. Die Amplituden der Schwingungen steigen bis zum festgelegten Wert stetig an und fallen anschließend wieder auf Null ab. Auf diese Weise ergibt sich eine sich bis auf den festgelegten Amplitudenwert ausdehnende und anschließend wieder zusammenziehende Spiralform.

Bei der daraus entstehenden Roboterbewegung bewegt sich der TCP entlang dieser Spirale. Die kartesische Ausdehnung der Spirale ist abhängig von den für Steifigkeit und Amplitude festgelegten Werten und vorhandenen Hindernissen.

Die Ebene, in der die spiralförmige Schwingung aufgeschaltet werden soll, wird als Wert vom Typ `CartPlane` übergeben. Die für die Parameter Steifigkeit,

Frequenz und Amplitude festgelegten Werte sind für beide Freiheitsgrade der Ebene identisch.

Zusätzlich wird ein Wert für die Gesamtzeit der Kraftschwingung übergeben. Die Zeit wird je zur Hälfte für das Auf- und Abschwingen der Schwingung verwendet:

Anstiegszeit = Gesamtzeit / 2

Haltezeit = 0

Abfallzeit = Gesamtzeit / 2

#### Syntax

```
controlMode = CartesianSineImpedanceControlMode.createSpiralPattern(CartPlane.plane, frequency, amplitude, stiffness, totalTime);
```

#### Erläuterung der Syntax

Element	Beschreibung
<i>controlMode</i>	Typ: <code>CartesianSineImpedanceControlMode</code> Name des Regler-Objekts
<i>plane</i>	Typ: <code>Enum</code> vom Typ <code>CartPlane</code> Ebene, in der die spiralförmige Schwingung aufgeschaltet werden soll
<i>frequency</i>	Typ: <code>double</code> Frequenz der Schwingung für beide Freiheitsgrade der angegebenen Ebene (Einheit: Hz) <b>■ 0.0 ... 15.0</b>
<i>amplitude</i>	Typ: <code>double</code> Amplitude der Schwingung für beide Freiheitsgrade der angegebenen Ebene (Einheit: N) <b>■ ≥ 0.0</b>
<i>stiffness</i>	Typ: <code>double</code> Steifigkeitswert für beide Freiheitsgrade der angegebenen Ebene (Einheit: N/m) <b>■ 0.0 ... 5000.0</b>
<i>totalTime</i>	Typ: <code>double</code> Gesamtzeit der spiralförmigen Schwingung. Die Zeit wird je zur Hälfte für das Auf- und Abschwingen der Schwingung verwendet (Einheit: s). <b>■ ≥ 0.0</b>

#### Beispiel

Auf der aktuellen Position des Roboterflansches soll eine spiralförmige Kraftschwingung in der XY-Ebene des Flansch-Koordinatensystems aufgeschaltet werden. Die Kraft soll spiralförmig bis zu einem Maximalwert von 100 N ansteigen. Der Kraftverlauf soll sich einmal pro Sekunde um den Startpunkt der Spirale winden (Frequenz der Kraftschwingung: 1.0 Hz). Innerhalb von 10 Sekunden soll die Kraftspirale an- und wieder absteigen.

```
CartesianSineImpedanceControlMode spiralMode;
spiralMode =
CartesianSineImpedanceControlMode.createSpiralPattern(CartPlane.XY,
1.0, 100, 500, 10);
lbr.move(positionHold(spiralMode, 10, TimeUnit.SECONDS));
```

Die Anzahl der Windungen ergibt sich aus der Gesamtzeit durch die Zeit für eine Windung ( $t_{\text{Periode}}$ ). Die Zeit für eine Windung entspricht der Periodendauer einer Schwingung, z. B.:

- Frequenz der Kraftschwingung:  $f = 1.0 \text{ Hz}$
- Gesamtzeit:  $t = 10 \text{ s}$

Die Anzahl der Windungen wird wie folgt berechnet:

$$\text{Anzahl}_{\text{Windungen}} = \text{Gesamtzeit} / t_{\text{Periode}} = 10 \text{ s} / 1 \text{ s} = 10$$

$$t_{\text{Periode}} = 1 / f = 1 / 1.0 \text{ Hz} = 1 \text{ s}$$

Die maximale Auslenkung ergibt sich aus dem Federgesetz:

$$\Delta x = F / C = 100 \text{ N} / (500 \text{ N/m}) = 0.2 \text{ m} = 20 \text{ cm}$$

## 17.8 Achsspezifischer Impedanzregler

Der achsspezifische Impedanzregler wird durch die Klasse JointImpedanceControlMode repräsentiert. In diesem Regelungsmodus verhält sich der Roboter nachgiebig.

Das zugrundeliegende Modell verwendet virtuelle Federn und Dämpfer. Im Gegensatz zum kartesischen Impedanzregler werden diese Federn und Dämpfer allerdings durch die Differenz zwischen aktuell gemessenen und kommandierten Achspositionen aufgespannt. Aus diesem Grund haben singuläre Stellungen des Roboters keinen Einfluss auf das Impedanzverhalten.

### 17.8.1 Parametrierung des achsspezifischen Impedanzreglers



**VORSICHT** In Impedanzregelung können ungenaue Sensorinformationen oder falsch gewählte Parameter (z. B. fehlerhafte Lastdaten, falsches Werkzeug) als äußere Kräfte interpretiert werden und zu unvorhersehbaren Bewegungen des Roboters führen.



Wird die Applikation in Impedanzregelung bei gespannter Feder pausiert, wird der Bewegungsbefehl unterbrochen. Beim Fortsetzen der Applikation wird die Feder wieder gespannt. Es kann zu einer ruckartigen Bewegung des Roboters kommen.

Folgende Regler-Eigenschaften können für jede Achse einzeln festgelegt werden:

- Steifigkeit
- Dämpfung

## 17.8.2 Methoden des achsspezifischen Impedanzreglers

### Übersicht

Methode	Beschreibung
setStiffness(...)	<p>Federsteifigkeit (Typ: double[]; Einheit: Nm/rad)</p> <p>Die achsspezifische Federsteifigkeit bestimmt, wie stark eine Achse bei Krafteinwirkung nachgibt.</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul> <p><b>Hinweis:</b> Die Federsteifigkeit muss für jede Achse angegeben werden.</p>
setDamping(...)	<p>Federdämpfung (Typ: double[]; ohne Einheit: Lehrsches Dämpfungsmaß)</p> <p>Die achsspezifische Federdämpfung bestimmt, wie stark die virtuellen Federn nach Auslenkung schwingen.</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul> <p>Default: 0.7</p> <p><b>Hinweis:</b> Die Federdämpfung muss für jede Achse angegeben werden.</p>
setStiffness ForAllJoints(...)	<p>Federsteifigkeit (Typ: double; Einheit: Nm/rad)</p> <p>Ein Wert bestimmt, wie stark alle Achsen bei Krafteinwirkung nachgeben.</p> <ul style="list-style-type: none"> <li>■ <b>≥ 0.0</b></li> </ul>
setDamping ForAllJoints(...)	<p>Federdämpfung (Typ: double; ohne Einheit: Lehrsches Dämpfungsmaß)</p> <p>Ein Wert bestimmt, wie stark die virtuellen Federn in allen Achsen nach Auslenkung schwingen.</p> <ul style="list-style-type: none"> <li>■ <b>0.0 ... 1.0</b></li> </ul>

### Konstruktorsyntax

```
JointImpedanceControlMode jointImp = new JointImpedanceControlMode(A1, A2, ... A7);
```

### Erläuterung der Syntax

Element	Beschreibung
jointImp	Typ: JointImpedanceControlMode Name des Regler-Objekts
A1 ... A7	Typ: double; Einheit: Nm/rad Achsspezifische Federsteifigkeiten Die Anzahl der Werte ist abhängig von der Achsenanzahl (hier: 7 Achsen).

### Beispiel 1

Mithilfe des achsspezifischen Impedanzreglers sollen 7 Achsen geregelt werden. Im Konstruktor des Reglers werden die achsspezifischen Federsteifigkeiten initial festgelegt. Später soll die Steifigkeit für Achse A4 geändert werden. Die Federdämpfung soll für alle Achsen gleich sein.

```
JointImpedanceControlMode jointImp
= new JointImpedanceControlMode(2000.0, 2000.0, 2000.0, 2000.0,
100.0, 100.0, 100.0);
...
jointImp.setStiffness(2000.0, 2000.0, 2000.0, 1500.0, 100.0, 100.0,
100.0);
jointImp.setDampingForAllJoints(0.5);
```

### Beispiel 2

Mithilfe des achsspezifischen Impedanzreglers sollen 7 Achsen geregelt werden. Im Konstruktor des Reglers werden die achsspezifischen Federsteifigkeiten

ten initial festgelegt. Später sollen Federsteifigkeit und Federdämpfung für alle Achsen gleich sein.

```
JointImpedanceControlMode jointImp
= new JointImpedanceControlMode(2000.0, 2000.0, 2000.0, 2000.0,
100.0, 100.0, 100.0);
...
jointImp.setStiffnessForAllJoints(100);
jointImp.setDampingForAllJoints(0.5);
```

## 17.9 Halten der Position in Regelung

### Beschreibung

Mithilfe des Bewegungsbefehls `positionHold(...)` kann der Roboter über einen einstellbaren Zeitraum hinweg seine kartesische Sollposition halten und dabei in Regelung bleiben.

Wird der Roboter in Nachgiebigkeitsregelung betrieben, kann er sich von seiner Sollposition entfernen. Ob, wie weit und in welche Richtung sich der Roboter dabei von der aktuellen kartesischen Sollposition (= Position bei Start des Befehls `positionHold(...)`) weg bewegt, hängt von den eingestellten Regler-Parametern und den daraus resultierenden Kräften ab. Außerdem kann der nachgiebig geregelte Roboter durch äußere Kräfte von seiner Sollposition abgebracht werden.

### Syntax

```
object.move(positionHold(controlMode, time, unit));
```

### Erläuterung der Syntax

Element	Beschreibung
<code>controlMode</code>	Typ: Unterklasse von <code>AbstractMotionControlMode</code> Name des Regler-Objekts
<code>time</code>	Typ: long  Gibt die Zeit an, für die der angegebene <code>controlMode</code> gehalten werden soll. Der Wert muss $\geq 0$ sein. Ein Wert $< 0$ bedeutet endlos.
<code>unit</code>	Typ: Enum vom Typ <code>TimeUnit</code>  Einheit der angegebenen Zeit.  Das Enum <code>TimeUnit</code> ist Bestandteil der Java-Standardbibliothek.

### Beispiel

Der Roboter soll für 10 Sekunden an seiner aktuellen Position gehalten werden. Der Roboter ist in dieser Zeit in der kartesischen X-Richtung weich geschalten.

```
CartesianImpedanceControlMode controlMode = new
CartesianImpedanceControlMode();

controlMode.parametrize(CartDOF.X).setStiffness(1000.0);
controlMode.parametrize(CartDOF.ALL).setDamping(0.7);

lbr.move(positionHold(controlMode, 10, TimeUnit.SECONDS));
```



## 18 Diagnose

### 18.1 Feldbus-Diagnose



Für eine genaue Fehleranalyse kann WorkVisual genutzt werden. Weitere Informationen zur Feldbus-Diagnose mit WorkVisual sind in der Dokumentation **WorkVisual** zu finden.



Wenn die Robotersteuerung als PROFINET-Master oder -Device verwendet wird, können Hardware-Probleme dazu führen, dass Busteilnehmer nicht erreicht werden können. In diesem Fall wird der Einsatz eines Diagnosewerkzeugs, beispielsweise WorkVisual, Step 7 oder Wireshark empfohlen.

#### 18.1.1 Allgemeine Feldbusfehler anzeigen

**Beschreibung** Der allgemeine Fehlerzustand der angeschlossenen Feldbusse kann auf der smartHMI angezeigt werden.

**Vorgehensweise**

1. Stationssicht öffnen.
2. Kachel **KUKA\_Sunrise\_Cabinet** wählen.  
Die Statusanzeige der Kachel **Feldbusse** zeigt den Sammelzustand aller an die Steuerung angeschlossenen Feldbusse an.
3. Kachel **Feldbusse** wählen.  
Die Detailansicht mit Fehlerinformationen zu den aktuell angeschlossenen Feldbussen öffnet sich.

#### 18.1.2 Fehlerzustand von E/As und E/A-Gruppen anzeigen

**Beschreibung** In der Navigationsleiste der smartHMI zeigt die Statusanzeige im Bereich **E/A-Gruppen** den Zustand der konfigurierten E/A-Gruppen an:

- Die untere Anzeige bildet den Sammelzustand aller konfigurierten E/A-Gruppen ab.
- Die obere Anzeige bildet den Zustand der ausgewählten E/A-Gruppe ab.

**Vorgehensweise**

- In der Navigationsleiste unter **E/A-Gruppen** die gewünschte E/A-Gruppe auswählen.  
Die Detailansicht der E/A-Gruppe öffnet sich. Jeder fehlerhafte Ein-/Ausgang ist gekennzeichnet.



Wenn PROFINET verwendet wird und Fehler an einzelnen Klemmen auftreten, werden nicht nur die betroffenen Ein-/Ausgänge, sondern alle konfigurierten PROFINET-E/As als fehlerhaft markiert.

### 18.2 Protokoll anzeigen

Auf der smartHMI kann ein Protokoll der Ereignisse und Zustandsänderungen des Systems angezeigt werden.

**Vorgehensweise**

1. Stationssicht oder Robotersicht öffnen.
2. Kachel **Protokoll** wählen. Die Ansicht **Protokoll** öffnet sich.  
Wird die Ansicht über die Robotersicht geöffnet, werden defaultmäßig nur die Protokolleinträge angezeigt, die den in der Navigationsleiste ausgewählten Roboter betreffen.

## 18.2.1 Ansicht Protokoll

### Übersicht

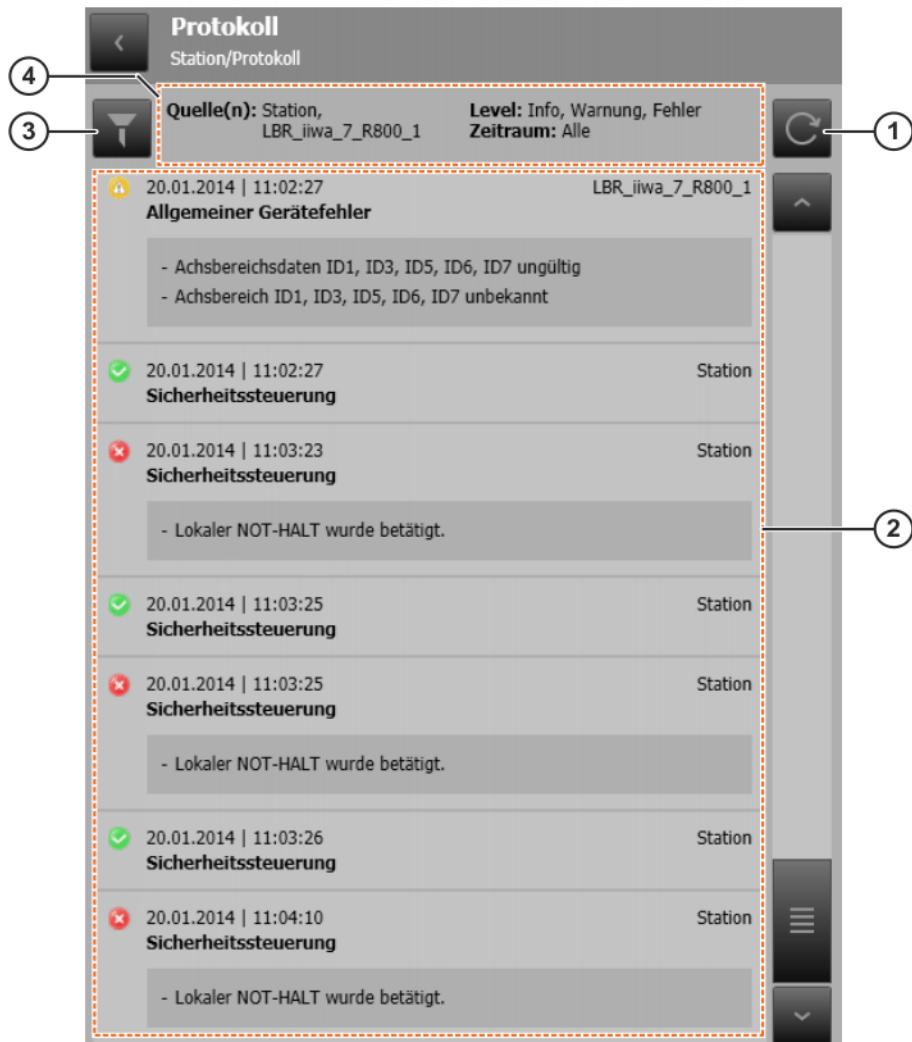


Abb. 18-1: Ansicht Protokoll

Pos.	Beschreibung
1	Button <b>Aktualisieren</b> Aktualisiert die angezeigten Protokolleinträge. Defaultmäßig wird nach dem Aktualisieren der aktuellste Eintrag zuoberst angezeigt. Ist ein Zeitfilter aktiv, wird der älteste Eintrag zuoberst angezeigt.
2	Liste der Protokolleinträge (>>> "Log-Ereignis" Seite 452)
3	Button <b>Filtereinstellungen</b> Öffnet das Fenster <b>Filtereinstellungen</b> , in dem die Protokolleinträge anhand unterschiedlicher Kriterien gefiltert werden können.
4	Anzeige Filtereinstellungen Hier werden die aktuell aktiven Filter angezeigt.

### Log-Ereignis

Die Protokolleinträge enthalten verschiedene Informationen zum jeweiligen Log-Ereignis.

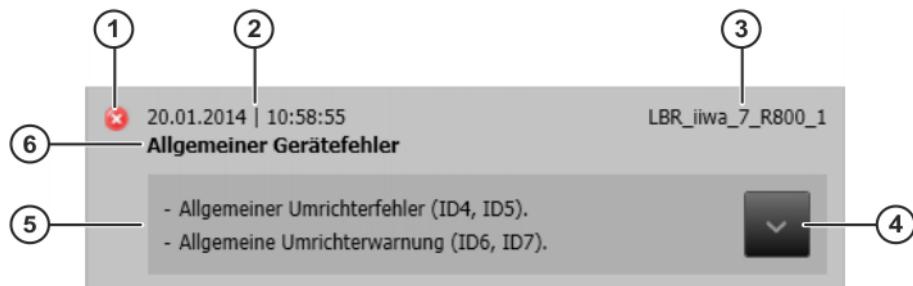


Abb. 18-2: Informationen zum Log-Ereignis

Pos.	Beschreibung
1	Log-Level des Ereignisses (>>> "Log-Level" Seite 453)
2	Datum und Uhrzeit des Log-Ereignisses (Systemzeit der Robotersteuerung)
3	Quelle des Log-Ereignisses (Roboter oder Station)
4	Button zum Maximieren/Minimieren der Detailanzeige Der Button steht nur zur Verfügung, wenn zu einem Ereignis mehr als 2 Symptome vorliegen.
5	Symptome des Log-Ereignisses (Detailanzeige) Defaultmäßig werden bis zu 2 Symptome pro Ereignis angezeigt.
6	Kategorie oder Kurzbeschreibung des Log-Ereignisses

## Log-Level

Der Log-Level eines Ereignisses wird durch folgende Symbole dargestellt:

Symbol	Beschreibung
✗	Fehler Kritisches Ereignis, das einen Fehlerzustand des Systems zur Folge hat
⚠	Warnung Kritisches Ereignis, das zu einem Fehler führen kann
✓	Information Unkritisches Ereignis oder Information über Zustandsänderung

## 18.2.2 Protokolleinträge filtern

### Voraussetzung

- Ansicht **Protokoll** ist geöffnet.

### Vorgehensweise

- Den Button **Filtereinstellungen** berühren. Das Fenster **Filtereinstellungen** öffnet sich.
- Gewünschte Filter über die zugehörigen Button auswählen.
- Den Button **Filtereinstellungen** oder einen Bereich außerhalb des Fens-ters berühren.

Das Fenster **Filtereinstellungen** wird geschlossen und die ausgewählten Filter werden aktiviert.



Durch Schließen der Ansicht **Protokoll** werden die Filter zurückgesetzt. Wird die Ansicht erneut geöffnet, sind wieder die Default-Einstellungen aktiv.

## Beschreibung



Abb. 18-3: Fenster Filtereinstellungen

Pos.	Beschreibung
1	<p><b>Filter Quelle(n)</b></p> <p>Die Protokolleinträge können nach den Quellen gefiltert werden, die das Log-Ereignis verursacht haben.</p> <ul style="list-style-type: none"> <li>■ <b>Station:</b> Es werden alle Protokolleinträge angezeigt, die die Station und die Ein-/Ausgänge von Feldbussen betreffen.</li> <li>■ <b>Roboter:</b> Es werden nur die Protokolleinträge angezeigt, die den in der Navigationsleiste ausgewählten Roboter betreffen, hier ein LBR iiwa 7 R800.</li> </ul> <p>Default bei Protokoll über Stationssicht: Beide Quellen sind ausgewählt.</p> <p>Default bei Protokoll über Robotersicht: Quelle ist der in der Navigationsleiste ausgewählte Roboter.</p>
2	<p><b>Filter Zeitraum</b></p> <p>Es kann ein Zeitfilter aktiviert werden, so dass nur die Protokolleinträge eines bestimmten Zeitraums angezeigt werden.</p> <p>Default: <b>Alle</b> (kein Zeitfilter aktiv)</p>
3	<p><b>Filter Level</b></p> <p>Die Protokolleinträge können nach ihrem Log-Level gefiltert werden.</p> <p>Default: <b>Info, Warnung, Fehler</b> (kein Filter nach Log-Level aktiv)</p>

### 18.3 Anzeige Fehlermeldungen (Applikationssicht)

Treten beim Ausführen einer Applikation Fehler auf, werden die zugehörigen Fehlermeldungen auf der smartHMI angezeigt.

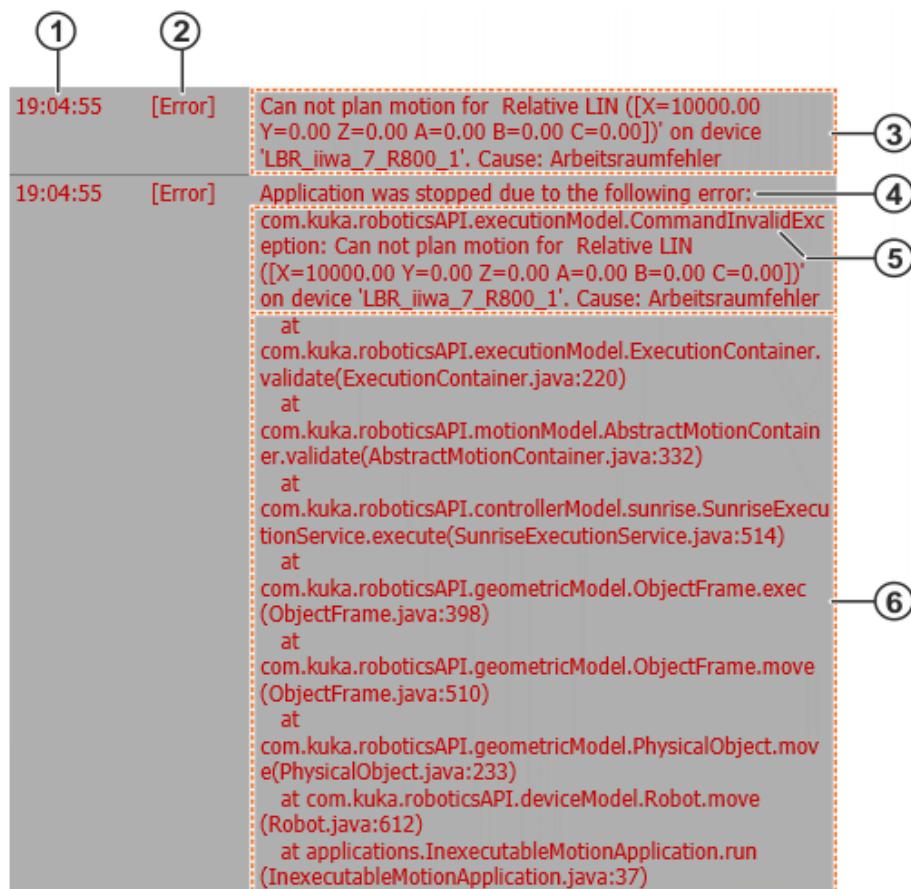


Abb. 18-4: Aufbau Fehlermeldung (Beispiel)

Pos.	Beschreibung
1	<b>Zeitstempel</b> Zeitpunkt, zu dem der Fehler aufgetreten ist
2	<b>Stufe</b> Log-Level der Meldung. Fehler besitzen den Log-Level <b>Error</b> .
3	Fehlermeldung
4	Information bei Abbruch der Applikation, z. B. nach einem Laufzeitfehler

Pos.	Beschreibung
5	Fehlerotyp Fehler sind als Java-Klassen definiert. Der Name der Klasse sowie das zugehörige Paket werden angezeigt. Dahinter folgt die Fehlermeldung (siehe Pos. 3).
6	Stacktrace Die Methodenaufrufe, die zum Fehler geführt haben, werden in aufsteigender Reihenfolge angezeigt. Die Methoden werden mit ihrem vollständigen Bezeichner angegeben. Außerdem wird die Nummer der Programmzeile angegeben, in der der Fehler aufgetreten ist. Anhand des Stacktrace kann man feststellen, an welcher Programmstelle die Methode aufgerufen wurde, die letztendlich zum Fehler geführt hat. Beispiel von unten nach oben gelesen: <ul style="list-style-type: none"> <li>■ Ursprung des Fehlers: Methode run() der Applikation InexecutableMotion.java, Zeile 37</li> <li>■ In der Zeile 37 der Applikation wurde die Methode move(...) der Klasse Robot aufgerufen. Im Quellcode der Klasse robot.java ist der Fehler in Zeile 612 aufgetreten, und zwar beim Aufruf der Methode move(...) der Klasse PhysicalObject.</li> <li>■ ...</li> <li>■ Der eigentliche Fehler ist in Zeile 220 im Quellcode der Klasse ExecutionContainer.java aufgetreten, und zwar beim Aufruf der Methode validate(...).</li> </ul>

Häufig ist ein Fehler die Folge einer Verkettung vorausgehender Fehler. In diesem Fall wird die gesamte Fehlerkette in absteigender Reihenfolge angezeigt.

```

19:07:38 [Error] Application was stopped due to the following error:
java.lang.RuntimeException: Es ist ein Fehler aufgetreten!
    at
applications.EmbeddedExceptionApplication.getNextPosition(EmbeddedExceptionApplication.java:46)
    at
applications.EmbeddedExceptionApplication.run(EmbeddedExceptionApplication.java:38)
Caused by: java.lang.Exception: Fehler bei der Berechnung
    at
applications.Utils.calculateValue(Utils.java:8)
    at
applications.EmbeddedExceptionApplication.getNextPosition(EmbeddedExceptionApplication.java:43)
    ... 1 more

```

Abb. 18-5: Anzeige Fehlerkette (Beispiel)

Pos.	Beschreibung
1	Folgefehler Hier wird das letzte Glied der Fehlerkette angezeigt. Im Beispiel ist dies ein Fehler vom Typ RuntimeException, der beim Ausführen der Methode run() in Zeile 38 der Applikation EmbeddedExceptionApplication.java aufgetreten ist.
2	Verursachende Fehler Die Anzeige der verursachenden Fehler wird immer wie folgt eingeleitet: <ul style="list-style-type: none"> <li>■ Caused by: <i>FehlerTyp</i></li> </ul> <p>Im Beispiel ist der verursachende Fehler vom Typ Exception und trat bei Aufruf der Methode calculateValue(...) der Klasse Utils auf. Auf diese Art wird die gesamte Fehlerkette bis zur eigentlichen Fehlerursache angegeben.</p>

## 18.4 Diagnoseinformationen sammeln für Fehleranalyse bei KUKA

Zur Fehleranalyse benötigt der KUKA Customer Support Diagnosedaten von der Robotersteuerung.

Hierfür wird eine ZIP-Datei mit dem Namen **KRCDiag** erzeugt, die auf der Robotersteuerung unter D:\DiagnosisPackages oder auf einem an der Robotersteuerung angesteckten USB-Stick archiviert werden kann. Das Diagnosepaket **KRCDiag** enthält die Daten, die der KUKA Customer Support benötigt, um einen Fehler zu analysieren. Hierzu gehören Informationen über die Systemressourcen, Maschinendaten und vieles mehr.

Auf die Diagnoseinformationen kann auch über Sunrise.Workbench zugegriffen werden. Dabei wird entweder ein bestehendes Diagnosepaket von der Robotersteuerung geladen oder ein neues Paket erstellt.



Projekte und Applikationen werden nicht in das Diagnosepaket aufgenommen. Es wird empfohlen diese Daten separat zu übermitteln, da sie wichtige Informationen für die Fehlersuche enthalten können.



Empfehlung: Diagnoseinformationen möglichst nur dann sammeln, wenn der Roboter stillsteht.



Schlägt das Sammeln der Diagnoseinformationen bei laufender Applikation fehl, Applikation anhalten und abwählen und Diagnoseprozess erneut starten.

### 18.4.1 Diagnosepaket über smartHMI erstellen

#### Beschreibung

Mit dieser Vorgehensweise kann das Diagnosepaket **KRCDiag** erstellt und auf der Robotersteuerung unter D:\DiagnosisPackages oder auf einem USB-Stick archiviert werden.

#### Vorgehensweise

1. Für die Archivierung auf einem USB-Stick: USB-Stick an der Robotersteuerung anstecken und warten bis die LED am USB-Stick nur noch dauerhaft leuchtet.
2. Im Hauptmenü **Diagnose > Diagnosepaket erstellen** und den gewünschten Speicherort wählen:
  - Festplatte
  - USB-Stick

Die Diagnoseinformationen werden zusammengestellt. Der Fortschritt wird in einem Fenster angezeigt. Wenn der Vorgang abgeschlossen ist, wird dies ebenfalls in dem Fenster angezeigt. Danach blendet sich das Fenster selbstständig wieder aus.

#### 18.4.2 Diagnosepaket über smartPAD erstellen

##### Beschreibung

Diese Vorgehensweise verwendet keine Menüpunkte, sondern Tasten auf dem smartPAD. Sie kann deshalb auch dann eingesetzt werden, wenn die smartHMI nicht zur Verfügung steht.

Das Diagnosepaket **KRCdiag** wird erstellt und auf der Robotersteuerung unter D:\DiagnosisPackages archiviert.



Die in der Vorgehensweise beschriebene Tastenfolge muss innerhalb von 2 Sekunden ausgeführt werden.

##### Vorgehensweise

1. Die Hauptmenü-Taste drücken und halten.
2. Die Tastatur-Taste 2-mal drücken.
3. Die Hauptmenü-Taste loslassen.

Die Diagnoseinformationen werden zusammengestellt. Der Fortschritt wird in einem Fenster angezeigt. Wenn der Vorgang abgeschlossen ist, wird dies ebenfalls in dem Fenster angezeigt. Danach blendet sich das Fenster selbstständig wieder aus.

#### 18.4.3 Diagnosepaket über Sunrise.Workbench erstellen

##### Voraussetzung

- Netzwerk-Verbindung zur Robotersteuerung

##### Vorgehensweise

1. Im **Paket-Explorer** auf das Projekt rechtsklicken und im Kontextmenü **Sunrise > Diagnose-Paket erstellen** wählen. Der Assistent zum Erstellen des Diagnosepakets öffnet sich.
2. **Öffnen...** wählen und zu dem Verzeichnis navigieren, in dem das Diagnosepaket **KRCdiag** erstellt werden soll. Bei Bedarf über **Neuen Ordner erstellen** einen Ordner für das Diagnosepaket anlegen. Mit **OK** bestätigen.
3. Auf **Weiter >** klicken. Das Diagnosepaket wird im angegebenen Ordner erstellt.
4. Über **Ziel-Ordner in Windows-Explorer öffnen** kann zum Ordner, in dem das Diagnosepaket erstellt wurde, navigiert werden, z. B. um es direkt per E-Mail zu versenden.
5. Auf **Fertigstellen** klicken. Der Assistent wird geschlossen.



Projekte und Applikationen werden nicht in das Diagnosepaket aufgenommen. Es wird empfohlen diese Daten separat zu übermitteln, da sie wichtige Informationen für die Fehlersuche enthalten können.

#### 18.4.4 Bestehende Diagnosepakete von Robotersteuerung laden

##### Voraussetzung

- Netzwerk-Verbindung zur Robotersteuerung

##### Vorgehensweise

1. Im **Paket-Explorer** auf das Projekt rechtsklicken und im Kontextmenü **Sunrise > Diagnose-Paket erstellen** wählen. Der Assistent zum Erstellen des Diagnosepakets öffnet sich.
2. **Öffnen...** wählen und zu dem Verzeichnis navigieren, in dem das Diagnosepaket **KRCdiag** kopiert werden soll. Bei Bedarf über **Neuen Ordner erstellen** einen Ordner für das Diagnosepaket anlegen. Mit **OK** bestätigen.

3. Den Radio-Button **Bestehende Diagnose-Pakete von Steuerung laden** aktivieren und gewünschte Diagnosepakete markieren.
4. Auf **Weiter >** klicken. Das Diagnosepaket wird in den angegebenen Ordner kopiert.

Wenn der Ordner bereits ein Diagnosepaket mit gleichem Dateinamen enthält, wird ein Benutzerdialog angezeigt. Der Kopievorgang kann abgebrochen werden.

5. Über **Ziel-Ordner in Windows-Explorer öffnen** kann zum Ordner, in den das Diagnosepaket kopiert wurde, navigiert werden, z. B. um es direkt per E-Mail zu versenden.
6. Auf **Fertigstellen** klicken. Der Assistent wird geschlossen.



## 19 KUKA Service

### 19.1 Support-Anfrage

**Einleitung**

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

**Informationen****Zur Abwicklung einer Anfrage werden folgende Informationen benötigt:**

- Problembeschreibung inkl. Angaben zu Dauer und Häufigkeit der Störung
- Möglichst umfassende Informationen zu den Hardware- und Software-Komponenten des Gesamtsystems

Die folgende Liste gibt Anhaltspunkte, welche Informationen häufig relevant sind:

- Typ und Seriennummer der Kinematik, z. B. des Manipulators
- Typ und Seriennummer der Steuerung
- Typ und Seriennummer der Energiezuführung
- Bezeichnung und Version der System Software
- Bezeichnungen und Versionen weiterer/anderer Software-Komponenten oder Modifikationen
- Diagnosepaket **KrcDiag**
  - Für KUKA Sunrise zusätzlich: Vorhandene Projekte inklusive Applikationen
  - Für Versionen der KUKA System Software älter als V8: Archiv der Software (**KrcDiag** steht hier noch nicht zur Verfügung.)
- Vorhandene Applikation
- Vorhandene Zusatzachsen

### 19.2 KUKA Customer Support

**Verfügbarkeit**

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

**Argentinien**

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

**Australien**

KUKA Robotics Australia Pty Ltd  
45 Fennell Street  
Port Melbourne VIC 3207  
Australien  
Tel. +61 3 9939 9656  
[info@kuka-robotics.com.au](mailto:info@kuka-robotics.com.au)  
[www.kuka-robotics.com.au](http://www.kuka-robotics.com.au)

<b>Belgien</b>	KUKA Automatisering + Robots N.V. Centrum Zuid 1031 3530 Houthalen Belgien Tel. +32 11 516160 Fax +32 11 526794 <a href="mailto:info@kuka.be">info@kuka.be</a> <a href="http://www.kuka.be">www.kuka.be</a>
<b>Brasilien</b>	KUKA Roboter do Brasil Ltda. Travessa Claudio Armando, nº 171 Bloco 5 - Galpões 51/52 Bairro Assunção CEP 09861-7630 São Bernardo do Campo - SP Brasilien Tel. +55 11 4942-8299 Fax +55 11 2201-7883 <a href="mailto:info@kuka-roboter.com.br">info@kuka-roboter.com.br</a> <a href="http://www.kuka-roboter.com.br">www.kuka-roboter.com.br</a>
<b>Chile</b>	Robotec S.A. (Agency) Santiago de Chile Chile Tel. +56 2 331-5951 Fax +56 2 331-5952 <a href="mailto:robotec@robotec.cl">robotec@robotec.cl</a> <a href="http://www.robotec.cl">www.robotec.cl</a>
<b>China</b>	KUKA Robotics China Co., Ltd. No. 889 Kungang Road Xiaokunshan Town Songjiang District 201614 Shanghai P. R. China Tel. +86 21 5707 2688 Fax +86 21 5707 2603 <a href="mailto:info@kuka-robotics.cn">info@kuka-robotics.cn</a> <a href="http://www.kuka-robotics.com">www.kuka-robotics.com</a>
<b>Deutschland</b>	KUKA Roboter GmbH Zugspitzstr. 140 86165 Augsburg Deutschland Tel. +49 821 797-4000 Fax +49 821 797-1616 <a href="mailto:info@kuka-roboter.de">info@kuka-roboter.de</a> <a href="http://www.kuka-roboter.de">www.kuka-roboter.de</a>

<b>Frankreich</b>	KUKA Automatisme + Robotique SAS Techvallée 6, Avenue du Parc 91140 Villebon S/Yvette Frankreich Tel. +33 1 6931660-0 Fax +33 1 6931660-1 <a href="mailto:commercial@kuka.fr">commercial@kuka.fr</a> <a href="http://www.kuka.fr">www.kuka.fr</a>
<b>Indien</b>	KUKA Robotics India Pvt. Ltd. Office Number-7, German Centre, Level 12, Building No. - 9B DLF Cyber City Phase III 122 002 Gurgaon Haryana Indien Tel. +91 124 4635774 Fax +91 124 4635773 <a href="mailto:info@kuka.in">info@kuka.in</a> <a href="http://www.kuka.in">www.kuka.in</a>
<b>Italien</b>	KUKA Roboter Italia S.p.A. Via Pavia 9/a - int.6 10098 Rivoli (TO) Italien Tel. +39 011 959-5013 Fax +39 011 959-5141 <a href="mailto:kuka@kuka.it">kuka@kuka.it</a> <a href="http://www.kuka.it">www.kuka.it</a>
<b>Japan</b>	KUKA Robotics Japan K.K. YBP Technical Center 134 Godo-cho, Hodogaya-ku Yokohama, Kanagawa 240 0005 Japan Tel. +81 45 744 7691 Fax +81 45 744 7696 <a href="mailto:info@kuka.co.jp">info@kuka.co.jp</a>
<b>Kanada</b>	KUKA Robotics Canada Ltd. 6710 Maritz Drive - Unit 4 Mississauga L5W 0A1 Ontario Kanada Tel. +1 905 670-8600 Fax +1 905 670-8604 <a href="mailto:info@kukarobotics.com">info@kukarobotics.com</a> <a href="http://www.kuka-robotics.com/canada">www.kuka-robotics.com/canada</a>

<b>Korea</b>	KUKA Robotics Korea Co. Ltd. RIT Center 306, Gyeonggi Technopark 1271-11 Sa 3-dong, Sangnok-gu Ansan City, Gyeonggi Do 426-901 Korea Tel. +82 31 501-1451 Fax +82 31 501-1461 <a href="mailto:info@kukakorea.com">info@kukakorea.com</a>
<b>Malaysia</b>	KUKA Robot Automation (M) Sdn Bhd South East Asia Regional Office No. 7, Jalan TPP 6/6 Taman Perindustrian Puchong 47100 Puchong Selangor Malaysia Tel. +60 (03) 8063-1792 Fax +60 (03) 8060-7386 <a href="mailto:info@kuka.com.my">info@kuka.com.my</a>
<b>Mexiko</b>	KUKA de México S. de R.L. de C.V. Progreso #8 Col. Centro Industrial Puente de Vigas Tlalnepantla de Baz 54020 Estado de México Mexiko Tel. +52 55 5203-8407 Fax +52 55 5203-8148 <a href="mailto:info@kuka.com.mx">info@kuka.com.mx</a> <a href="http://www.kuka-robotics.com/mexico">www.kuka-robotics.com/mexico</a>
<b>Norwegen</b>	KUKA Sveiseanlegg + Roboter Sentrumsvegen 5 2867 Hov Norwegen Tel. +47 61 18 91 30 Fax +47 61 18 62 00 <a href="mailto:info@kuka.no">info@kuka.no</a>
<b>Österreich</b>	KUKA Roboter CEE GmbH Gruberstraße 2-4 4020 Linz Österreich Tel. +43 7 32 78 47 52 Fax +43 7 32 79 38 80 <a href="mailto:office@kuka-roboter.at">office@kuka-roboter.at</a> <a href="http://www.kuka.at">www.kuka.at</a>

<b>Polen</b>	KUKA Roboter Austria GmbH Spółka z ograniczoną odpowiedzialnością Oddział w Polsce Ul. Porcelanowa 10 40-246 Katowice Polen Tel. +48 327 30 32 13 or -14 Fax +48 327 30 32 26 ServicePL@kuka-roboter.de
<b>Portugal</b>	KUKA Sistemas de Automatización S.A. Rua do Alto da Guerra nº 50 Armazém 04 2910 011 Setúbal Portugal Tel. +351 265 729780 Fax +351 265 729782 kuka@mail.telepac.pt
<b>Russland</b>	KUKA Robotics RUS Werbnaia ul. 8A 107143 Moskau Russland Tel. +7 495 781-31-20 Fax +7 495 781-31-19 info@kuka-robotics.ru www.kuka-robotics.ru
<b>Schweden</b>	KUKA Svetsanläggningar + Robotar AB A. Odnhrs gata 15 421 30 Västra Frölunda Schweden Tel. +46 31 7266-200 Fax +46 31 7266-201 info@kuka.se
<b>Schweiz</b>	KUKA Roboter Schweiz AG Industriestr. 9 5432 Neuenhof Schweiz Tel. +41 44 74490-90 Fax +41 44 74490-91 info@kuka-roboter.ch www.kuka-roboter.ch

<b>Spanien</b>	KUKA Robots IBÉRICA, S.A. Pol. Industrial Torrent de la Pastera Carrer del Bages s/n 08800 Vilanova i la Geltrú (Barcelona) Spanien Tel. +34 93 8142-353 Fax +34 93 8142-950 <a href="mailto:Comercial@kuka-e.com">Comercial@kuka-e.com</a> <a href="http://www.kuka-e.com">www.kuka-e.com</a>
<b>Südafrika</b>	Jendamark Automation LTD (Agentur) 76a York Road North End 6000 Port Elizabeth Südafrika Tel. +27 41 391 4700 Fax +27 41 373 3869 <a href="http://www.jendamark.co.za">www.jendamark.co.za</a>
<b>Taiwan</b>	KUKA Robot Automation Taiwan Co., Ltd. No. 249 Pujong Road Jungli City, Taoyuan County 320 Taiwan, R. O. C. Tel. +886 3 4331988 Fax +886 3 4331948 <a href="mailto:info@kuka.com.tw">info@kuka.com.tw</a> <a href="http://www.kuka.com.tw">www.kuka.com.tw</a>
<b>Thailand</b>	KUKA Robot Automation (M)SdnBhd Thailand Office c/o Maccall System Co. Ltd. 49/9-10 Soi Kingkaew 30 Kingkaew Road Tt. Rachatheva, A. Bangpli Samutprakarn 10540 Thailand Tel. +66 2 7502737 Fax +66 2 6612355 <a href="mailto:atika@ji-net.com">atika@ji-net.com</a> <a href="http://www.kuka-roboter.de">www.kuka-roboter.de</a>
<b>Tschechien</b>	KUKA Roboter Austria GmbH Organisation Tschechien und Slowakei Sezemická 2757/2 193 00 Praha Horní Počernice Tschechische Republik Tel. +420 22 62 12 27 2 Fax +420 22 62 12 27 0 <a href="mailto:support@kuka.cz">support@kuka.cz</a>

**Ungarn**            KUKA Robotics Hungaria Kft.  
Fö út 140  
2335 Taksony  
Ungarn  
Tel. +36 24 501609  
Fax +36 24 477031  
[info@kuka-robotics.hu](mailto:info@kuka-robotics.hu)

**USA**            KUKA Robotics Corporation  
51870 Shelby Parkway  
Shelby Township  
48315-1787  
Michigan  
USA  
Tel. +1 866 873-5852  
Fax +1 866 329-5852  
[info@kukarobotics.com](mailto:info@kukarobotics.com)  
[www.kukarobotics.com](http://www.kukarobotics.com)

**Vereinigtes Königreich**    KUKA Robotics UK Ltd  
Great Western Street  
Wednesbury West Midlands  
WS10 7LL  
Vereinigtes Königreich  
Tel. +44 121 505 9970  
Fax +44 121 505 6589  
[service@kuka-robotics.co.uk](mailto:service@kuka-robotics.co.uk)  
[www.kuka-robotics.co.uk](http://www.kuka-robotics.co.uk)



# Index

## Zahlen

2004/108/EG 42  
2006/42/EG 42  
3-Punkt-Methode 110  
89/336/EWG 42  
95/16/EG 42

## A

Abbruchbedingungen für Bewegungen 360  
Abbruchbedingungen, auswerten 361  
ABC 2-Punkt-Methode 107  
ABC Welt-Methode 109  
Abfrage, Roboterposition 328  
Abwählen (Button) 91  
Abweisende Schleife 394  
Achsbereich 25, 227  
Achsgrenze 227  
Achsmomente, abfragen 322  
Achsmomenten-Bedingung 343  
Achsmomentenüberwachung 233  
Achsspezifische Position, abfragen 328  
Achsspezifische Überwachungsräume, definieren 227  
Achsspezifischer Impedanzregler 425, 447  
addCartesianForce(...) 375  
addCartesianTorque(...) 376  
addCommandedCartesianPositionXYZ(...) 376  
addCommandedJointPosition(...) 376  
addControllerListener(...) 334, 337  
addCurrentCartesianPositionXYZ(...) 377  
addCurrentJointPosition(...) 376  
addDoubleUserKey(...) 382  
addExternalJointTorque(...) 375  
addInternalJointTorque(...) 375  
addUserKey(...) 382  
Allgemeine Sicherheitsmaßnahmen 36  
AMF 16  
Angewandte Normen und Vorschriften 42  
Anhalteweg 25, 28, 222  
Anlagenintegrator 26  
Annehmende Schleife 395  
ANSI/RIA R.15.06-2012 43  
Ansicht, Protokoll 452  
Anwender 27  
Anwender PSM, PSM-Tabelle 195  
API 16  
App\_Enable 178  
App\_Start 178, 402  
Applikation, pausieren 94, 393  
Applikations-Override 93, 339  
Applikationsdaten (Sicht) 48  
Applikationsmodus 89  
Applikationswerkzeug 79  
Arbeitsbereich 25, 28, 227  
Arbeitsbereich, neu 56  
Arbeitsbereich, Sunrise.Workbench 56  
Arbeitsbereich, wechseln 56  
Arbeitsraum 220, 222

Arbeitsspeicher 45  
areDataValid() 125  
attachTo(...) 310, 311  
Aufbau eines Bewegungsbefehls 294  
Aufbau, Roboter-Applikation 283  
Aufgaben (Sicht) 49  
Ausgang, ändern 99  
Ausschalten, Robotersteuerung 66  
AUT 25  
AutExt\_Active 178  
AutExt\_AppReadyToStart 178, 401  
Auto-Vervollständigen 284  
Automatik 25  
Automatikbetrieb 40  
Außerbetriebnahme 41  
awaitFileAvailable(...) 379

## B

Bahnbezogene Bedingung 356  
Bahnbezogene Schaltaktionen 342, 364  
Basis für Handverfahren 139  
Basis-Koordinatensystem 77, 110  
Basis, vermessen 110  
Bedienerschutz 29, 30  
Bedienhandgerät 19, 23  
Bedienoberfläche, KUKA smartHMI 67  
Bedienoberfläche, Sunrise.Workbench 47  
Bedienung, KUKA smartPAD 63  
Bedienung, KUKA Sunrise.Workbench 47  
Bedingte Verzweigung 396  
Bedingung für boolesche Signale 359  
Bedingung für Wertebereich eines Signals 359  
Bedingung, kartesisches Moment 351  
Bedingungen 341  
Begriffe, Sicherheit 25  
Begriffe, verwendet 16  
Behandlung fehlgeschlagener Bewegungsbefehle 402  
Benutzer 25  
Benutzerdialoge, programmieren 391  
Benutzermeldungen, programmieren 390  
Benutzertasten 64  
Benutzertasten-Leiste, erstellen 381  
Benutzertasten, aktivieren 95  
Benutzertasten, definieren 380  
Bestimmungsgemäße Verwendung 22, 23  
Betreiber 25, 27  
Betriebsart, wechseln 74  
Betriebsartenwahl 33  
Bewegungsarten 261  
Bewegungsausführung, pausieren 393  
Bewegungsparameter 302  
Bewegungsprogrammierung, Grundlagen 261  
Blockierendes Warten 372  
BooleanIOCondition 342  
BrakeState (Enum) 131  
BrakeTest (Klasse) 124, 127  
BrakeTestResult (Klasse) 130

breakWhen(...) 360, 362  
Bremsdefekt 36  
Bremse, defekt 119, 133  
Bremsentest 117  
Bremsentest-Applikation, Vorlage 120  
Bremsentest, Ausführung starten 128  
Bremsentest, Ausgangsposition ändern 123  
Bremsentest, auswerten 129  
Bremsentest, durchführen 133  
Bremsentest, Ergebnis (Anzeige) 134  
Bremsentest, Ergebnis abfragen 131  
Bremsentest, Programmierschnittstelle 124  
Bremsweg 25  
Bus, verschalten 172

**C**

CartesianTorqueCondition 342  
CE-Kennzeichnung 24  
CIB-SR 16  
CIRC 297  
CIRC, Bewegungsart 262  
clipApplicationOverride(...) 340  
clipManualOverride(...) 340  
Continuous Path 261  
CP-Bewegung 261  
CP-Spline-Block 261  
CP-Spline-Block, anlegen 300  
createAndEnableConditionObserver(...) 370  
createConditionObserver(...) 370  
createDesiredForce(...) 442  
createLissajousPattern(...) 442  
createNormalForceCondition(...) 345, 346  
createShearForceCondition(...) 345, 348  
createSinePattern(...) 442  
createSpatialForceCondition(...) 345  
createSpatialTorqueCondition(...) 352  
createSpiralPattern(...) 442  
createTiltingTorqueCondition(...) 352, 354  
createTurningTorqueCondition(...) 352, 353  
createUserKeyBar(...) 381

**D**

DataRecorder 374  
Daten, aufzeichnen und auswerten 374  
Datentypen 292  
Debuggen (Perspektive) 49  
DefaultApp\_Error 178  
Dejustieren 103  
detach() 313  
Diagnose 451  
Diagnoseinformationen, sammeln 457  
Diagnosepaket, erstellen 457, 458  
Diagnosepaket, von Robotersteuerung laden  
458  
displayModalDialog(...) 391  
do-while-Schleife 395  
Dokumentation, Industrieroboter 15  
Drehmoment 351

**E**

E/A-Gruppe, editieren 170

E/A-Gruppe, erstellen 169  
E/A-Gruppe, löschen 170  
E/A-Gruppe, Vorlage exportieren 170  
E/A-Gruppe, Vorlage importieren 171  
E/A-Konfiguration, exportieren 174  
E/A-Konfiguration, neu 166  
E/A-Konfiguration, öffnen 166  
EA-Verschaltung (Fenster) 172, 173  
Effektiver Override 93, 94, 339  
EG-Konformitätserklärung 24  
Eigenschaften (Sicht) 49  
Ein-/Ausgänge, anzeigen 99  
Einbauerklärung 23, 24  
Einfache Kraftschwingung, aufschalten 443  
Einleitung 15  
Einschalten, Robotersteuerung 66  
Einschaltverzögerung, für Sicherheitsfunktionen  
232  
Elektromagnetische Verträglichkeit (EMV) 43  
EMV-Richtlinie 24, 42  
EN 60204-1 + A1 43  
EN 61000-6-2 43  
EN 61000-6-4 + A1 43  
EN 614-1 + A1 43  
EN ISO 10218-1 43  
EN ISO 12100 43  
EN ISO 13849-1 42  
EN ISO 13849-2 42  
EN ISO 13850 42  
enable(), DataRecorder 377  
Entsorgung 41  
equals(...) 362  
ESM 16  
ESM-Mechanismus 200  
ESM-Tabelle 195  
ESM-Zustand, löschen 203  
ESM-Zustand, neu 201  
Event-driven Safety Monitoring 188  
Exception 16  
Externe Steuerung 177

**F**

Fahrbereitschaft-Signal, auf Änderung reagieren  
334  
Fahrbereitschaft, abfragen 334  
Federgesetz 428  
Fehlerbehandlung 402  
Feldbus-Diagnose 451  
Feldbusse, Übersicht 166  
Festplattenspeicher 45  
Filtereinstellungen 452  
Flansch-Koordinatensystem 77  
for-Schleife 393  
ForceComponentCondition 341  
ForceCondition 341  
Frame 16  
Frame-Verwaltung 137  
Frame, als Basis kennzeichnen 138  
Frames, anfahren 89  
Frames, anzeigen 85  
Frames, löschen 140

Frames, neu anlegen 138  
 Frames, teachen 86  
 Frames, verschieben 139  
 Freifahren, Roboter 82  
 FSoE 16  
 Funktionsprüfung 38

**G**

Gebrauchsdauer 25  
 Gefahrenbereich 25  
 Geschwindigkeit 80  
 Geschwindigkeitsüberwachungen 213  
 getAlphaRad() 331  
 getApplicationData().createFromTemplate() 310  
 getApplicationData().getFrame() 142  
 getApplicationOverride() 340  
 getApplicationUI() 381  
 getAxis() 130  
 getBetaRad() 331  
 getBrakeIndex() 130  
 getCommandedCartesianPosition(...) 328  
 getCommandedCartesianPosition() 367  
 getCommandedJointPosition() 328, 367  
 getCurrentCartesianPosition() 328, 367  
 getCurrentJointPosition() 328, 367  
 getEffectiveOverride() 340  
 getEmergencyStopEx() 336  
 getEmergencyStopInt() 336  
 getExecutionMode() 338  
 getExternalForceTorque(...) 324, 325  
 getExternalTorque() 323  
 getFiredBreakConditionInfo() 361  
 getFiredCondition() 362, 367  
 getFlange() 311  
 getForce() 325  
 getForcelnaccuracy() 326  
 getFrame(...) 311, 314  
 getFriction() 130  
 getGammaRad() 331  
 getGravity() 130  
 getHomePosition() 333  
 getLogLevel() 131  
 getManualOverride() 340  
 getMaxAbsTorqueValues() 125  
 getMaxBrakeHoldingTorque() 130  
 getMeasuredBrakeHoldingTorque() 130  
 getMeasuredTorque() 322  
 getMinBrakeHoldingTorque() 130  
 getMissedEvents() 367  
 getMotion() 401  
 getMotionContainer() 367  
 getMotorHoldingTorque() 130  
 getMotorIndex() 130  
 getMotorMaximalTorque() 130  
 getObserverManager() 370, 373  
 getOperationMode() 336  
 getOperatorSafetyState() 337  
 getPositionInfo() 362, 363  
 getPositionInformation(...) 328  
 getPositionInformation() 330, 367  
 getRecovery() 400

getRecoveryStrategy(...) 401  
 getRotationOffset() 330  
 getSafetyState() 336  
 getSafetyStopSignal() 337  
 getSingleMaxAbsTorqueValue(...) 125  
 getSingleTorqueValue(...) 323  
 getStartPosition() 401  
 getStartTimestamp() 125  
 getState() 131  
 getStoppedMotion() 362, 364  
 getStopTimestamp() 126  
 getTestedTorque() 131  
 getTimestamp() 131  
 getTorque() 325  
 getTorquelnaccuracy() 326  
 getTorqueValues() 323  
 getTranslationOffset() 330  
 getTriggerTime() 367  
 Grafikkarte 45

**H**

Haftungshinweis 23  
 halt() 393  
 Haltebremsen, öffnen 84  
 Hand-Override 68, 80  
 Handführen, Bewegungsart 269  
 Handführen, programmieren 304  
 Handführmodus 269, 304  
 handGuiding() 269, 304  
 Hardware 45  
 hasActiveMotionCommand() 335  
 Hauptmenü-Taste 64  
 Hauptmenü, aufrufen 73  
 Hilfspunkt 262, 297  
 Hintergrund-Task, neu 55  
 Hintergrund-Tasks 409  
 Hinweise 15  
 HOME-Position 331  
 HOME-Position, abfragen 333  
 HOME-Position, ändern 332  
 HOV 68, 80

**I**

IAnyEdgeListener 369  
 IApplicationOverrideControl (Schnittstelle) 339  
 ICallbackAction, Schnittstelle 365  
 ICondition, Schnittstelle 341  
 IControllerStateListener 334  
 if-else-Verzweigung 396  
 IFallingEdgeListener 369  
 Inbetriebnahme 37, 103  
 Industrieroboter 23  
 initialize() 284, 411, 414  
 initializeCyclic(...) 411  
 Installation 159  
 Installation, KUKA Sunrise.Workbench 45  
 Installationsverzeichnis 45  
 Instandsetzung 40  
 IORangeCondition 342  
 IP-Adresse, anzeigen 101  
 IP-Adressen 51

IRecovery, Schnittstelle 400  
IRisingEdgeListener 369  
isEnabled() 379  
isFileAvailable() 379  
isForceValid(...) 326  
isInHome() 333  
isMastered() 333  
isReadyToMove() 334  
isRecording() 379  
isRecoveryRequired(...) 401  
isRecoveryRequired() 400  
isTorqueMeasured() 126  
isTorqueValid(...) 326  
Istposition, achsspezifisch 97  
Istposition, kartesisch 98  
ISunriseControllerStateListener 337  
ITriggerAction, Schnittstelle 365  
IUserKeyBar, Schnittstelle 382

## J

Java-Datei, umbenennen 60  
Java-Editor 283  
Java-Editor, öffnen 283  
Java-Paket, neu 54  
Java-Projekt, neu 58  
Java-Projekte, referenzieren 59  
Javadoc 16  
Javadoc (Sicht) 49  
Javadoc-Browser, Aufbau 289  
Javadoc-Informationen, anzeigen 287  
Joint Path 261  
JointTorqueCondition 341  
JP-Bewegung 261  
JP-Spline-Block 261  
JP-Spline-Block, anlegen 301  
JRE 16  
Justage 103  
Justage, löschen 103  
Justagezustand, abfragen 333

## K

Kartesische Arbeitsräume, definieren 222  
Kartesische Position, abfragen 329  
Kartesische Schutzzäume, definieren 224  
Kartesische Soll-Ist-Differenz, abfragen 330  
Kartesischer Impedanzregler 425, 427, 435  
Kenntnisse, benötigt 15  
Kennzeichnungen 35  
Kippmoment 351  
KLI 16  
KMP 16  
Kollisionserkennung 234  
Komplexe Bedingungen 342  
Konformitätserklärung 24  
Konstante Kraft, aufschalten 443  
Koordinatensystem, für Verfahrtasten 68  
Koordinatensysteme 76  
Kraft-Komponenten-Bedingung 349  
Kraftbedingung 344  
KRCDiag 457  
Kreisbewegung 297

KRF 17, 26, 82  
KUKA Customer Support 461  
KUKA RoboticsAPI 17  
KUKA smartHMI 17, 67  
KUKA smartPAD 17, 26, 36, 63  
KUKA Sunrise Cabinet 17, 19  
KUKA Sunrise.OS 17  
KUKA PSM, PSM-Tabelle 195

## L

Lagerung 41  
Lastdaten 147  
Lastdaten, eingeben 147  
LIN 296  
LIN REL 297  
LIN, Bewegungsart 262  
Linearbewegung 296, 297  
Lissajous-Schwingung, aufschalten 444

## M

main() 284  
Manipulator 19, 23, 26, 28  
Manueller Betrieb 39  
Manueller Override 91, 93, 339  
Marken 16  
Maschinenrichtlinie 24, 42  
Medien-Flansch Touch 208, 210  
Mehrache Verzweigung 398  
Meldungsfenster 91  
Meldungsprogrammierung 390  
Menüleiste 48  
Methoden, extrahieren 286  
Moment-Komponenten-Bedingung 355  
Momente, achsspezifisch 98  
Momentenreferenzierung 241  
Momentenwert-Ermittlung 123  
Monitoring 342, 368  
Montagerichtung 51, 77  
MotionBatch 298  
MotionPathCondition 342  
move(...) 294, 313, 402  
moveAsync(...) 295, 313, 404  
MRK 17

## N

Navigationsleiste 68  
Nicht sicherheitsgerichtete Funktionen 33  
Nicht-zyklischer Hintergrund-Task 413  
Niederspannungsrichtlinie 24  
Normalkraft 345  
NOT-HALT 64  
NOT-HALT-Einrichtung 28, 29, 31  
NOT-HALT-Gerät 29  
NOT-HALT, extern 29, 31, 208  
NotificationType, Enum 371  
Nullraum-Bewegung 82

## O

Oberflächennormale 345  
Objektverwaltung 142  
Objektvorlagen (Sicht) 48

**O**  
 ObserverManager 370, 373  
 onIsReadyToMoveChanged(...) 334  
 onKeyEvent(...), IUserKeyListener 384  
 onSafetyStateChanged(...) 337  
 onTriggerFired(...) 365  
 Operatoren 343  
 Optionen 19, 23  
 Orientierungsführung 303  
 Orientierungsführung, LIN, CIRC, SPL 272  
 Override 68, 80, 91, 93, 339  
 Override, ändern und abfragen 339

**P**  
 Paket-Explorer (Sicht) 48  
 Panikstellung 30  
 Passwort, ändern 206  
 Pausieren, Applikation 94  
 Pausieren, Programm 94  
 Performance Level 24  
 Permanent Safety Monitoring 186  
 Personal 27  
 Perspektiven, anzeigen 49  
 Perspektiven, auswählen 48  
 Pflegearbeiten 41  
 Point to Point 261  
 positionHold(...) 449  
 Positions- und Momentenreferenzierung 240  
 Positionsreferenzierung 240  
 Positionsregler 425, 427  
 Produktbeschreibung 19  
 PROFINET 17  
 PROFIsafe 17  
 Programm, automatisch starten 94  
 Programm, manuell starten 94  
 Programm, pausieren 94  
 Programmablaufart, auswählen 92  
 Programmablaufart, ändern und abfragen 338  
 Programmablaufkontrolle 393  
 Programmausführung 90  
 Programmierung 283  
 Programmierung (Perspektive) 49  
 Projekt, auf Robotersteuerung übertragen 155  
 Projekt, synchronisieren 155  
 Projekt, von Robotersteuerung laden 158  
 Projekte, archivieren 57  
 Projekte, in Arbeitsbereich laden 57  
 Projektverwaltung 137  
 Protokoll, Ansicht 452  
 Protokoll, anzeigen 451  
 Protokolleinträge, filtern 453  
 Prozessor 45  
 Prüfsumme, Sicherheitskonfiguration 205  
 PSM 17  
 PSM-Mechanismus 197  
 PSM-Tabelle, Anwender PSM 195  
 PSM-Tabelle, KUKA PSM 195  
 PTP 295  
 PTP, Bewegungsart 261  
 PTPRecoveryStrategy (Klasse) 401  
 Punkt-zu-Punkt-Bewegung 295

**R**  
 Reaktionsweg 25  
 Redundanzinformationen 141, 276  
 Redundanzwinkel 277  
 Referenzierung, aufheben 59  
 Regler-Objekt, anlegen 426  
 Regler-Parameter, festlegen 426  
 Regler, Übersicht 425  
 Reinigungsarbeiten 41  
 Roboter-Applikation, anwählen 90  
 Roboter-Applikation, neu 54  
 Roboter, rückpositionieren 94  
 Roboteraktivität, abfragen 335  
 Roboterfuß-Koordinatensystem 77  
 Roboterposition, Abfrage 328  
 Robotersicht 72  
 Robotersteuerung 23  
 Robotersteuerung, ein-/ausschalten 66  
 Robotertyp, anzeigen 101  
 RoboticsAPI 17  
 RoboticsAPI, Version anzeigen 294  
 run() 284, 414  
 runCyclic() 411

**S**  
 SafetyConfiguration.sconf (Datei) 54, 195  
 Schablonen 285  
 Schablonen, benutzerspezifisch 286  
 Scherkraft 345  
 Schleifen, verschachteln 399  
 Schnelleingabe, Java 285  
 Schriftarten 292  
 Schulungen 15  
 Schutzausstattung 35  
 Schutzbereich 26, 28  
 Schutzeinrichtungen, extern 35  
 Schutzraum 220, 224  
 Seriennummer, anzeigen 101  
 Service, KUKA Roboter 461  
 set-Methoden 302  
 setAdditionalControlForce(...) 432  
 setAmplitude(...) 438  
 setApplicationOverride(...) 340  
 setAxisLimitsEnabled(...) 308  
 setAxisLimitsMax(...) 308  
 setAxisLimitsMin(...) 308  
 setAxisLimitViolationFreezesAll(...) 308  
 setBias(...) 439  
 setBlendingCart(...) 303  
 setBlendingOri(...) 303  
 setBlendingRel(...) 303  
 setCartAcceleration(...) 302  
 setCartJerk(...) 303  
 setCartVelocity(...) 302  
 setCriticalText(...), IUserKey 389  
 setDamping(...) 432, 448  
 setDampingForAllJoints(...) 448  
 setExecutionMode(...) 338  
 setFallTime(...) 441  
 setForceLimit(...) 440  
 setFrequency(...) 438

setHoldTime(...) 441  
setHomePosition(...) 332  
setJointAccelerationRel(...) 302, 304  
setJointJerkRel(...) 303, 304  
setJointVelocityRel(...) 302, 304  
setLED(...), IUserKey 388  
setMaxCartesianVelocity(...) 434  
setMaxControlForce(...) 433  
setMaxPathDeviation(...) 434  
setNullSpaceDamping(...) 433  
setNullSpaceStiffness(...) 433  
setOrientationReferenceSystem(...) 274, 303  
setOrientationType(...) 272, 303  
setPermanentPullOnViolationAtStart(...) 308  
setPhaseDeg(...) 439  
setPositionLimit(...) 440  
setRiseTime(...) 441  
setSafetyWorkpiece(...) 317  
setStayActiveUntilPatternFinished(...) 441  
setStiffness(...) 432, 448  
setStiffnessForAllJoints(...) 448  
setText(...), IUserKey 386  
setTotalTime(...) 441  
Sicherer Betriebshalt 231  
Sicherer Betriebshalt, extern 29, 32  
Sicherheit 23  
Sicherheit von Maschinen 42, 43  
Sicherheit, rechtliche Rahmenbedingungen 23  
Sicherheitsabnahme, Übersicht 243  
Sicherheitsfunktion, neu für ESM 203  
Sicherheitsfunktion, neu für PSM 199  
Sicherheitsfunktionen 24  
Sicherheitsfunktionen, konfigurieren 197, 200  
Sicherheitsgerichtete Funktionen 28  
Sicherheitsgerichtete Stopp-Reaktionen 32  
Sicherheitsgerichtete Werkstücke 152  
Sicherheitsgerichtetes Werkstücke, definieren 154  
Sicherheitsgerichtetes Werkzeug 148  
Sicherheitsgerichtetes Werkzeug, definieren 149  
Sicherheitshalt 26  
Sicherheitshalt 0 26  
Sicherheitshalt 1 26  
Sicherheitshalt 1 (bahntreu) 26  
Sicherheitshalt, extern 29, 31  
Sicherheitshinweise 15  
Sicherheitskonfiguration 183  
Sicherheitskonfiguration, aktivieren 205  
Sicherheitskonfiguration, deaktivieren 205  
Sicherheitskonfiguration, konvertieren 161  
Sicherheitskonfiguration, öffnen 195  
Sicherheitskonfiguration, wiederherstellen 206  
Sicherheitskonzept 184  
Sicherheitssignale, abfragen 335  
Sicherheitssignale, auswerten 335  
Sicherheitssteuerung, fortsetzen 83  
Sichten, anordnen 49  
Signalzustand, abfragen 336  
Signalzustand, auf Änderung reagieren 337  
Single Point of Control 41  
Singularität 273  
Singularitäten 279  
smartHMI 17, 67  
smartPAD 17, 26, 36, 63  
Software 19, 23, 45  
Software-Endschalter 35  
Software-Komponenten 20  
Software-Version, anzeigen 101  
Space Mouse 64  
Spiralförmige Kraftschwingung, aufschalten 445  
SPL, Bewegungsart 263  
Spline-Segment 263  
Spline, Bewegungsart 263  
SPOC 41  
Sprachpaket, installieren 46, 161  
SPS 17  
Standard-Frame für Bewegungen 146  
Start-Rückwärts-Taste 64  
Start-Taste 64, 65  
Starten, Programm 94  
Starten, System Software 66  
startEvaluation() 124  
startRecording() 377  
StartRecordingAction 377  
Station\_Error 178  
StationSetup.cat (Datei) 54, 159  
Stationskonfiguration 159  
Stationskonfiguration, öffnen 159  
Stationssicht 70  
Status 277  
Statusanzeige 69  
Step (Button) 91  
Stillstandsüberwachung 231  
STOP-Taste 64  
stopEvaluation() 124  
Stopp-Kategorie 0 26  
Stopp-Kategorie 1 26  
Stopp-Kategorie 1 (bahntreu) 26  
Stopp-Reaktionen, sicherheitsgerichtet 32  
stopRecording() 378  
StopRecordingAction 378  
Störungen 37  
Sunrise E/As, anlegen 167  
Sunrise E/As, ändern 170  
Sunrise E/As, löschen 170  
Sunrise-Projekt, neu 51  
Sunrise.Workbench, Bedienoberfläche 47  
Sunrise.Workbench, deinstallieren 45  
Sunrise.Workbench, installieren 45  
Sunrise.Workbench, starten 47  
SunriseExecutionService 338  
Support-Anfrage 461  
switch-Verzweigung 398  
Symbolleisten 48, 50  
Synchronisation, Projekt 155  
System Software, installieren 160  
Systemintegrator 24, 26, 27  
Systemvoraussetzungen, PC 45  
Systemzustände, abfragen 332

**T**

T1 26  
 T2 26  
 Tastatur 70  
 Tastatur-Taste 64  
 TCP 17, 104, 145  
 TCP-Kraftüberwachung 235  
 Template, für Sunrise-Projekt 51  
 Tippbetrieb 35  
 Tool Center Point 104  
 TorqueComponentCondition 342  
 TorqueEvaluator (Klasse) 121, 122, 124  
 TorqueStatistic (Klasse) 122, 124, 125  
 Touch-Screen 63  
 Transport 37  
 Trigger 342, 364  
 Trigger-Informationen, auswerten 366  
 Trigger, programmieren 364  
 triggerWhen(...) 364  
 Turn 278  
 Typenschild 65

**U**

Umbenennen, Variable 284  
 USB-Anschluss 65  
 UserKeyAlignment (Enum) 386

**Ü**

Überlast 36  
 Überschleifen 270  
 Überschleipunkt 270  
 Übersicht des Robotersystems 19  
 Übersicht, Sicherheitsabnahme 243  
 Überwachen von Prozessen 342, 368  
 Überwachung, Werkzeugorientierung 228  
 Überwachungsräume 220

**V**

Variable, umbenennen 284  
 Variablen 293  
 Verbindungsleitungen 19, 23  
 Verfahrtart 68  
 Verfahren, achsspezifisch 78, 80  
 Verfahren, kartesisch 78, 81  
 Verfahren, manuell, Roboter 78  
 Verfahrtasten 64, 80, 81  
 Vermessen 104  
 Vermessen, Basis 110  
 Vermessen, Werkzeug 104  
 Verriegelung trennender Schutzeinrichtungen  
 30  
 Verschalten, Ein-/Ausgänge 174  
 Versionsinformationen, RoboticsAPI 294  
 Verwendete Begriffe 16  
 Verwendung, nicht bestimmungsgemäß 23  
 Verwendung, unsachgemäß 23  
 VirensScanner, installieren 162  
 VirensScanner, Meldungen anzeigen 101  
 Vorlage, für Sunrise-Projekt 51

**W**

waitFor(...) 372  
 Wartung 40  
 Welt-Koordinatensystem 76  
 Werkstück-Frame, anlegen 145  
 Werkstück, anlegen 144  
 Werkstücke, deklarieren 309  
 Werkstücke, initialisieren 310  
 Werkzeug-Frame, anlegen 145  
 Werkzeug-Koordinatensystem 77, 104  
 Werkzeug-Lastdaten, ermitteln 112  
 Werkzeug, abschalten 208  
 Werkzeug, anlegen 144  
 Werkzeug, vermessen 104  
 Werkzeuge, deklarieren 309  
 Werkzeuge, initialisieren 310  
 Werkzeugorientierung 228  
 while-Schleife 394  
 Wiederinbetriebnahme 37, 103

**X**

XYZ 4-Punkt-Methode 105

**Z**

Zählschleife 393  
 Zeichen 292  
 Zielgruppe 15  
 Zubehör 19, 23  
 Zurücksetzen (Button) 91  
 Zustimmmeinrichtung 29, 30  
 Zustimmmeinrichtung, extern 29, 31  
 Zustimmungsschalter 30, 65, 66  
 Zyklischer Hintergrund-Task 410



