

Rexroth IndraMotion MLC 12VRS RCL-Programmieranleitung

R911333851
Ausgabe 02

Anwenderbeschreibung



Titel Rexroth IndraMotion
MLC 12VRS
RCL-Programmieranleitung

Art der Dokumentation Anwenderbeschreibung

Dokumentations-Type DOK-MLC***-RCL*PRO*V12-AP02-DE-P

Interner Ablagevermerk RS-e3eb11e15b29718d0a6846a00109078e-2-de-DE-19

Zweck der Dokumentation Das vorliegende Handbuch informiert über die Robot Control. Die Programmiersprache RCL (Robot Control Language) steht dabei im Vordergrund. Beschrieben werden die Programmstruktur, Variablen, Funktionen, Bewegungsanweisungen sowie die notwendigen Parameter des Systems.

Änderungsverlauf

Ausgabe	Stand	Bemerkung
Ausgabe 01	05.2011	Erstausgabe für MLC 12VRS
Ausgabe 02	02.2012	Umstellung mm -> mm/inch, K-0-0054

Schutzvermerk © Bosch Rexroth AG 2012

Alle Rechte bei Bosch Rexroth AG, auch für den Fall von Schutzrechtsanmeldungen. Jede Verfügungsbefugnis, wie Kopier- und Weitergaberecht, bei uns.

Verbindlichkeit Die angegebenen Daten dienen allein der Produktbeschreibung und sind nicht als zugesicherte Eigenschaften im Rechtssinne zu verstehen. Änderungen im Inhalt der Dokumentation und Liefermöglichkeiten der Produkte sind vorbehalten.

Herausgeber Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2 • D-97816 Lohr a. Main
Telefon +49 (0)93 52 / 18-0 • Fax +49 (0)93 52 / 18 49 41
<http://www.boschrexroth.com/>

System IndraLogic & Basis Motion Logic, GBU (vha)

Hinweis Diese Dokumentation ist auf chlorfrei gebleichtem Papier gedruckt.

Inhaltsverzeichnis

	Seite
1 Zu dieser Dokumentation.....	9
1.1 Gültigkeit der Dokumentation.....	9
1.2 Erforderliche und ergänzende Dokumentationen.....	9
1.3 Darstellung von Informationen.....	13
1.3.1 Sicherheitshinweise.....	13
1.3.2 Verwendete Symbole.....	14
1.3.3 Bezeichnungen und Abkürzungen.....	14
2 Koordinaten und Koordinatensysteme.....	15
2.1 Allgemeines.....	15
2.2 Gelenk-Koordinatensystem (JCS).....	16
2.3 Basis-Koordinatensystem (OCS).....	16
2.4 Welt-Koordinatensystem (WCS).....	18
2.5 Werkstück-Koordinatensystem (WpCS).....	19
3 Anwendung der Koordinatensysteme.....	21
3.1 Definition der Koordinatensysteme.....	21
3.2 Definition des Basis-Koordinatensystems relativ zum Welt-Koordinatensystem.....	21
3.3 Definition des Werkstück-Koordinatensystems relativ zum Welt-Koordinatensystem.....	23
3.4 Werkstückkoordinaten-Datei.....	25
3.5 Werkstückkoordinaten-Editor.....	25
3.6 Bibliotheksfunktionen.....	26
4 Verfahrbereichsbegrenzungen in der Robot Control.....	33
5 RCL-Programmstruktur.....	35
5.1 Allgemeines.....	35
5.2 Arbeitsweise des Compilers.....	35
5.2.1 Verwendung.....	35
5.2.2 Quelldateien.....	35
5.2.3 Ausgabedateien.....	35
5.3 Compileranweisungen.....	36
5.3.1 Allgemeines.....	36
5.3.2 Kinematikdefinition.....	36
5.3.3 Welt-Koordinatennamen (WC name definition).....	36
5.3.4 Achsnamen (JC name definition).....	37
5.3.5 Einfügen von Dateien.....	37
5.3.6 Wahlfreie Extension innerhalb der Einfügeanweisung.....	38
5.3.7 Testinformation.....	38
5.4 Hauptprogramm-Aufbau.....	39
5.4.1 Allgemeines.....	39
5.4.2 Deklarationsteil.....	39
5.4.3 Anweisungsteil.....	40

Inhaltsverzeichnis

	Seite
5.5	Programmvereinbarung..... 41
5.6	Hauptprogrammaufruf im Hauptprogramm..... 42
5.7	Unterprogrammvereinbarung..... 45
5.8	Programmablauf 46
6	RCL-Konstanten..... 51
6.1	RCL-Konstantenvereinbarung..... 51
6.2	Vordefinierte RCL-Konstanten..... 51
7	RCL-Variablen..... 53
7.1	Allgemeines..... 53
7.2	RCL-Datentypen..... 53
7.2.1	Übersicht..... 53
7.2.2	Einfache RCL-Datentypen..... 54
7.2.3	Zusammengesetzte RCL-Datentypen..... 55
7.2.4	Benutzerdefinierte RCL-Typen..... 55
7.3	RCL-Variablendeklaration..... 57
7.4	Punktvariablen..... 59
7.4.1	Vereinbarung von Punktvariablen..... 59
7.4.2	Punkte und Punktedatei "pkt"..... 59
7.4.3	Vollständige Wertzuweisung..... 60
7.4.4	Zuweisung von Variablen für Einzelkomponenten..... 60
7.5	Arbeit mit Textvariablen..... 62
7.6	Arbeit mit Feldvariablen (ARRAYs)..... 62
7.7	Kommunikationskanäle zur SPS..... 64
7.7.1	Kommunikationskanäle zur SPS, Allgemeines..... 64
7.7.2	Kommunikationskanäle zur SPS, Vereinbarung und Programmierung 67
8	RCL-Programmsteuerung..... 71
8.1	WAIT-Anweisung..... 71
8.2	PAUSE-Anweisung..... 75
8.3	HALT-Anweisung..... 76
8.4	REPEAT-Anweisung..... 76
8.5	JUMP-Anweisung..... 77
8.6	IF-THEN-(ELSE)-Anweisung..... 79
8.7	CASE-Anweisung..... 82
9	Wertzuweisungen und Verknüpfungen..... 83
9.1	Wertzuweisungen..... 83
9.2	Verknüpfungen..... 83
9.2.1	Arithmetische Ausdrücke..... 83
9.2.2	Vergleichsoperationen..... 85
9.2.3	Logische Ausdrücke..... 85
9.2.4	Rangfolge der Operationen..... 86

Inhaltsverzeichnis

	Seite
10 RCL-Funktionen.....	89
10.1 SIN-Funktion.....	89
10.2 COS-Funktion.....	89
10.3 ATAN-Funktion.....	90
10.4 SQRT-Wurzel-Funktion.....	90
10.5 ABS-Funktion.....	91
10.6 TRUNC-Funktion.....	91
10.7 ORD-Funktion.....	91
10.8 CHR-Funktion.....	91
10.9 ROUND-Funktion.....	91
10.10 EOF-Funktion.....	92
10.11 ASSIGN-Funktion.....	92
10.12 INT_ASC-Funktion.....	92
10.13 ASC_INT-Funktion.....	93
10.14 SIZEOF-Funktion.....	94
10.15 LIMIT_MIN, LIMIT_MAX, LIMIT_OFF-Funktionen.....	95
10.16 JC-Standardfunktion.....	95
10.17 WC-Standardfunktion.....	95
11 RCL-Spezialfunktionen.....	97
11.1 RCL-Spezialfunktionen, Allgemeines.....	97
11.2 RCL-Spezialfunktionen, tabellarische Auflistung.....	98
11.3 Positionsgenaueres Schalten.....	99
11.3.1 Positionsgenaueres Schalten, Allgemeines.....	99
11.3.2 SPC_FCT: 1, Positionsgenaueres Schalten digitaler Ausgänge auf der Bahn.....	100
11.3.3 SPC_FCT: 2, Positionsgenaueres Schalten von Prozessdaten auf der Bahn.....	101
11.4 SPC_FCT: 21, Bandart.....	102
11.5 SPC_FCT: 23, DateTime.....	103
11.6 SPC_FCT: 24, SysClk.....	103
11.7 SPC_FCT: 28, Bandzähler setzen.....	104
11.8 SPC_FCT: 53, bandsynchroner Arbeitsbereich der Bandart 4.....	105
11.9 RCL-Programmierung "Überschleifmodus".....	107
11.9.1 SPC_FCT: 60, Überschleifmodus setzen.....	107
11.9.2 SPC_FCT: 59, Überschleifmodus abfragen.....	108
12 RCL-Bewegungsanweisungen.....	111
12.1 Allgemeines.....	111
12.2 Direkte Bewegungsanweisungen.....	111
12.2.1 Überblick.....	111
12.2.2 Bewegungsbefehle.....	112
12.2.3 Interpolationsart	116
12.2.4 Wirkungsweise bewegungsbeeinflussender RCL-Anweisungen und Parameter.....	120
12.2.5 Zusatzinformationen.....	121
Zusatzinformationen, Allgemeines.....	121
Zusatzinformationen Geschwindigkeit	121

Inhaltsverzeichnis

	Seite
Zusatzinformationen Geschwindigkeitsfaktor.....	123
Zusatzinformationen Beschleunigung	124
Zusatzinformationen Beschleunigungsfaktor	126
Zusatzinformationen Bremsverzögerungsfaktor	127
Zusatzinformationen Ruck	128
12.3 Bewegungsbeeinflussende Anweisungen	129
12.3.1 Allgemeines.....	129
12.3.2 Bandsynchronisation.....	129
12.3.3 Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)	131
12.3.4 Look-Ahead über mehrere Sätze bei PROGR_SLOPE und CONT_SLOPE.....	136
12.3.5 Spätes Kommandieren.....	136
12.3.6 Räumliches Überschleifen.....	137
Überblick.....	137
Räumliches Überschleifen bei PROGR_SLOPE und CONT_SLOPE.....	138
Räumliches Überschleifen bei BLOCK_SLOPE.....	142
Spätes Überschleifen.....	144
Überschleifmodus.....	146
13 Einrichtebetrieb.....	149
13.1 Einrichtebetrieb, Allgemeines.....	149
13.2 Tippen in unterschiedlichen Koordinatensystemen.....	150
13.2.1 Einstellende Parameter für das Tippen in Achskoordinaten.....	150
13.2.2 Einstellende Parameter für das Tippen in Weltkoordinaten.....	151
13.2.3 Tippen in Weltkoordinaten.....	152
14 RCL-Parameter.....	153
14.1 Achsbezogene Parameter, genutzt durch Kinematiken einer MLC.....	153
14.1.1 Allgemeines.....	153
14.1.2 A-0-0028, Fahrbereichsgrenzscharter.....	153
14.1.3 A-0-0029, Positionspolaritäten.....	153
14.1.4 A-0-0030, Positionsgrenzwert, positiv.....	154
14.1.5 A-0-0031, Positionsgrenzwert, negativ.....	154
14.1.6 A-0-0032, Geschwindigkeitsgrenzwert, positiv.....	155
14.1.7 A-0-0033, Geschwindigkeitsgrenzwert, negativ.....	155
14.1.8 A-0-0034, Beschleunigungsgrenzwert, bipolar.....	155
14.1.9 A-0-0036, Ruckgrenzwert, bipolar.....	155
14.1.10 A-0-0038, Drehmoment-/Kraftgrenzwert, bipolar.....	156
14.1.11 A-0-0048, Geschwindigkeitspolaritäten.....	156
14.1.12 A-0-0049, Drehmoment-/Kraft-Polaritäten.....	157
14.1.13 A-0-0080, Joggen, kurze Distanz.....	158
14.1.14 A-0-0081, Joggen, lange Distanz.....	158
14.1.15 A-0-0082, Joggen, langsame Geschwindigkeit.....	159
14.1.16 A-0-0083, Joggen, schnelle Geschwindigkeit.....	160
14.1.17 A-0-0084, Joggen, Beschleunigung.....	161
14.1.18 A-0-0085, Joggen, Bremsverzögerung.....	161

Inhaltsverzeichnis

	Seite
14.1.19 A-0-0086, Joggen, Beschleunigungsruck.....	162
14.1.20 A-0-0087, Joggen, Bremsruck.....	162
14.1.21 A-0-0224, Schnellhalt-Verzögerung.....	163
14.1.22 A-0-0225, Schnellhalt-Ruck.....	163
14.1.23 A-0-0230, Überschleifdistanz PTP.....	163
14.1.24 A-0-0240, Feedrate-Override.....	163
14.1.25 A-0-0241, Feedrate-Override-Steuerwort.....	164
14.1.26 A-0-0242, Lageregelung, Aufsynchronisierbeschleunigung.....	164
14.1.27 A-0-0243, Lageregelung, Aufsynchronisiergeschwindigkeit.....	165
14.2 Gemeinsame C-Parameter aller Kinematiken einer MLC.....	166
14.2.1 Allgemeines.....	166
14.2.2 C-0-0123, Maximale Anzahl an Kinematiken.....	166
14.2.3 C-0-2480, Konfigurierte Kinematikenanzahl.....	166
14.2.4 C-0-2482, Liste der nicht verknüpften Achsen.....	167
14.2.5 C-0-2483, Liste der konfigurierten Kinematiken.....	167
14.2.6 C-0-2484, Kinematikkonfigurationsliste.....	167
14.3 Parameter einer Kinematik in der MLC.....	168
14.3.1 Allgemeines.....	168
14.3.2 K-0-0001, Logische Kinematiknummer.....	168
14.3.3 K-0-0002, Kinematikbezeichnung.....	168
14.3.4 K-0-0003, Maximale Bahngeschwindigkeit.....	168
14.3.5 K-0-0004, reserviert.....	169
14.3.6 K-0-0005, Software-Endschalter, positiv, WC.....	169
14.3.7 K-0-0006, Software-Endschalter, negativ, WC.....	169
14.3.8 K-0-0007, Achsanzahl der Kinematik.....	169
14.3.9 K-0-0008, Achskonfigurationsliste.....	169
14.3.10 K-0-0009, Oszi: Liste aller oszilloskopierbaren K-Parameter.....	171
14.3.11 K-0-0010, Liste aller K-Parameter.....	172
14.3.12 K-0-0011, Liste aller ungültigen K-Parameter.....	172
14.3.13 K-0-0012, Prüfsumme aller K-Parameter.....	172
14.3.14 K-0-0013, IDN-Liste der K-Prüfsummenparameter.....	172
14.3.15 K-0-0014, Bahnbeschleunigung, Änderungszeit.....	173
14.3.16 K-0-0015, Bahnbremsverzögerung, Änderungszeit.....	175
14.3.17 K-0-0016, Projektidentifikationsnummer.....	178
14.3.18 K-0-0020, Kinematikdiagnosemeldung.....	178
14.3.19 K-0-0021, Kinematikstatus.....	179
14.3.20 K-0-0022, Erweiterter Kinematikstatus.....	181
14.3.21 K-0-0023, Kinematikdiagnosenummer.....	183
14.3.22 K-0-0031, Transformationsschema.....	183
14.3.23 K-0-0032, Referenzieren der Achsen, Sollkonfiguration.....	184
14.3.24 K-0-0033, IPOS, Aktualisierung bei Koordinatenwechsel.....	184
14.3.25 K-0-0036, WC-Achsüberwachung.....	184
14.3.26 K-0-0040, Transformation, Längen der Mechanik.....	185
14.3.27 K-0-0041, Transformation, Faktoren.....	185
14.3.28 K-0-0042, Verschiebung des WC-Systems.....	186
14.3.29 K-0-0043, Werkstückkoordinatenzuordnungsliste.....	186

Inhaltsverzeichnis

	Seite
14.3.30 K-0-0050, Distanzeinheiten WC.....	187
14.3.31 K-0-0051, Geschwindigkeitseinheiten WC.....	188
14.3.32 K-0-0052, Beschleunigungseinheiten WC.....	189
14.3.33 K-0-0053, Ruckeinheiten WC.....	189
14.3.34 K-0-0054, Maßeinheit für Kinematikdaten.....	190
14.3.35 K-0-0070, Bandanzahl der Kinematik.....	190
14.3.36 K-0-0071, Kopplungsfaktoren Band 1.....	190
14.3.37 K-0-0072, Kopplungsfaktoren Band 2.....	191
14.3.38 K-0-0073, Kopplungsfaktoren Band 3.....	192
14.3.39 K-0-0074, Kopplungsfaktoren Band 4.....	192
14.3.40 K-0-0080, Joggen, Distanz kurz im WC-System.....	192
14.3.41 K-0-0081, Joggen, Distanz lang im WC-System.....	193
14.3.42 K-0-0082, Joggen, Geschwindigkeit langsam im WC-System.....	194
14.3.43 K-0-0083, Joggen, Geschwindigkeit schnell im WC-System.....	195
14.3.44 K-0-0084, Joggen, Beschleunigung im WC-System.....	196
14.3.45 K-0-0085, Joggen, Bremsverzögerung im WC-System.....	196
14.3.46 K-0-0086, Joggen, Beschleunigungsruck im WC-System.....	197
14.3.47 K-0-0087, Joggen, Verzögerungsruck im WC-System.....	197
14.3.48 K-0-0100, Aktuelle Position, Kartesische Originalkoordinaten.....	198
14.3.49 K-0-0102, Aktuelle Bahngeschwindigkeit.....	198
14.3.50 K-0-0103, Aktuelle Bahnbeschleunigung.....	198
14.3.51 K-0-0104, Aktueller Bahnruck.....	199
14.3.52 K-0-0105, Aktuelle Position, Kartesische Werkstückkoordinaten.....	199
14.3.53 K-0-0106, Koordinatenwerte vom Werkzeug.....	199
14.3.54 K-0-0107, Koordinatenwerte vom Werkstück.....	199
14.3.55 K-0-0121 bis K-0-0136, Koordinatennamen	199
14.3.56 K-0-0121, Koordinatennamen K1.....	199
14.3.57 K-0-0140, Werkzeugbezeichnung.....	199
14.3.58 K-0-0141, Werkstückbezeichnung.....	199
14.3.59 K-0-0150, Interpolierte Sollposition.....	199
14.3.60 K-0-0152, Interpolierte Bahngeschwindigkeit.....	199
14.3.61 K-0-0153, Interpolierte Bahnbeschleunigung.....	200
14.3.62 K-0-0154, Interpolierter Bahnruck.....	200
14.3.63 K-0-0155, Schutzzonen ein/aus.....	200
14.3.64 K-0-0156, Schutzzone1 min. Punkt.....	200
14.3.65 K-0-0157, Schutzzone1 max. Punkt.....	200
14.3.66 K-0-0156/K-0-0157 Schutzzone1 min./max. Punkt bis K-0-0218/K-0-0219 Schutzzone32 min./max. Punkt.....	201
14.3.67 K-0-0220, Dateitransfername.....	202
14.3.68 K-0-0221, Name der zu kompilierenden Datei.....	203
14.3.69 K-0-0240, Feedrate-Override.....	203
14.3.70 K-0-0241, Feedrate-Override-Steuerwort.....	203
14.3.71 K-0-0242, Acceleration-Override.....	204
14.3.72 K-0-0243, Deceleration-Override.....	204
14.3.73 K-0-1060, Kommando: Lade Datei von CF.....	204
14.3.74 K-0-1061, Kommando: Speichere Datei auf CF.....	204

Inhaltsverzeichnis

	Seite
14.3.75 K-0-1062, Kommando: Compiliere Datei auf MLC.....	205
14.3.76 K-0-2200, Programmierte Position 1 (aktiv).....	205
14.3.77 K-0-2201, Programmierte Position 2 (aktiv).....	205
14.3.78 K-0-2202, Typ der aktiven programmierten Position.....	205
14.3.79 K-0-2203, Interpolationstyp.....	205
14.3.80 K-0-2204, Aktuell aktiver QLL-Dateiname.....	205
14.3.81 K-0-2205, Aktuell aktive Zeilennummer.....	205
14.3.82 K-0-2300, SPS-Bit-Eingang (RCL).....	205
14.3.83 K-0-2301, SPS-Ganzzahl-Eingang (RCL).....	205
14.3.84 K-0-2302, SPS-Gleitkommazahl-Eingang (RCL).....	205
15 Kinematik-Benutzeroberfläche.....	207
15.1 Allgemeines.....	207
15.2 Robot Control Menüstruktur.....	208
15.2.1 Menüpunkt Robot.....	208
15.2.2 Menüpunkt Kinematik.....	208
15.2.3 Menüpunkt Programme.....	208
15.2.4 Menüpunkt Prog.QLL.....	209
15.3 RCL-Editor.....	209
15.3.1 Allgemeines.....	209
15.3.2 Merkmale.....	209
15.3.3 Optionen.....	210
15.3.4 Neues RCL-Programm erstellen.....	210
15.3.5 Arbeiten mit dem Editor.....	210
15.4 Compiler.....	214
15.5 Debugger.....	215
15.5.1 Allgemeines.....	215
15.5.2 Setzen und Löschen von Haltepunkten.....	215
15.5.3 Programm debuggen.....	216
15.5.4 Variablen überwachen.....	217
16 Anwendung der RoCo-Multikinematik-Technik.....	219
16.1 Allgemeines.....	219
16.2 Anlegen von Kinematiken.....	219
16.2.1 Allgemeines.....	219
16.2.2 Festlegung des Kinematiktyps (Robotertyp, K-0-0031).....	221
16.2.3 Festlegung der Achszuordnung (K-0-0008).....	221
16.2.4 Löschen von Kinematiken.....	223
16.3 Bewegungssteuerung der Kinematiken.....	224
16.4 RCL-Programme der Kinematiken.....	225
16.5 Punktetabellen der Kinematiken.....	226
16.5.1 Allgemeines.....	226
16.5.2 Punktetabellen, Zuordnung und Verhalten.....	227
16.6 PLC-Shared-Memory-Interfaces (KinData) der Kinematiken.....	228

Inhaltsverzeichnis

	Seite
17 Robot Control-Funktionalität, programmiert auf der SPS.....	231
18 Transformationsvorschriften.....	233
18.1 Transformationsvorschriften, Überblick.....	233
18.2 Robotertyp 00, Karteser.....	233
18.3 Robotertyp 07, 2-Achs-Delta mit optional 1 Handachse.....	237
18.4 Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse.....	240
18.5 Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse.....	242
18.6 Robotertyp 12, 2-Achs-Parallelogramm.....	245
18.7 Robotertyp 13, 2-Achs-Parallelogramm.....	247
18.8 Robotertyp 18, Umschlingungskinematik.....	249
18.9 Robotertyp 19, 2-Achs-Parallelogramm V1.....	250
18.10 Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse.....	252
18.11 Robotertyp 21, 2-Achs-Parallelogramm V2.....	257
18.12 Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen.....	259
18.13 Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen.....	265
18.14 Robotertyp 29, 2-Achs-Delta mit rotatorischer und translatorischer Achse.....	272
18.15 Robotertyp 30, Adabot.....	275
18.16 Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP.....	278
18.17 Robotertyp 100, SPS.....	279
19 RCL-Schlüsselworte.....	281
20 Service und Support.....	291
Index.....	293

1 Zu dieser Dokumentation

1.1 Gültigkeit der Dokumentation

Zielgruppe Diese Dokumentation richtet sich an Anwender, die sich über die Robot Control der IndraMotion MLC informieren möchten.

Die Programmiersprache RCL (Robot Control Language) steht dabei im Vordergrund.

Beschrieben werden

- Programmstruktur,
- Variablen, Funktionen, Bewegungsanweisungen,
- die notwendigen Parameter des Systems,
- Programmierreferenzen zum detaillierten Nachschlagen,
- Transformationsvorschriften und
- Besonderheiten bei der SPS-Programmierung.

Diese Dokumentation unterstützt den Anwender in den Phasen

- Programmierung des RCL-Programms,
- Debugging und Inbetriebnahme.

1.2 Erforderliche und ergänzende Dokumentationen

Dokumentationstitel mit Typenschlüssel und Materialnummer

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 12VRS Software-Installation DOK-IWORKS-SOFTINS*V12-COxx-DE-P, R911334395 Die Dokumentation beschreibt den Installationsprozess von IndraWorks.	X	X
/5/	Rexroth IndraWorks 12VRS Engineering DOK-IWORKS-ENGINEE*V12-APxx-DE-P, R911334387 Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.	X	X
/20/	Rexroth IndraMotion MLC 12VRS Funktionsbeschreibung DOK-MLC***-FUNC****V12-APxx-DE-P, R911333847 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 12VRS Funktionsbeschreibung DOK-XLC***-FUNC****V12-APxx-DE-P, R911333877 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraLogic XLC.		X
/7/	Rexroth IndraWorks 12VRS CamBuilder DOK-IWORKS-CAMBUIL*V12-APxx-DE-P, R911333841 Diese Dokumentation dient zur Beschreibung der Grundlagen und der Bedienung des CamBuilder, des Werkzeugs zur Kurvenscheibenbearbeitung.	X	X

Zu dieser Dokumentation

/37/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Automation Interface DOK-XLCMLC-AUT*INT*V12-APxx-DE-P, R911334177 Diese Dokumentation beschreibt den skriptbasierten Zugriff auf IndraWorks-Projektdateien über die Schnittstelle des Automation Interfaces.	X	X
/38/	Rexroth IndraWorks 12VRS FDT-Container DOK-IWORKS-FDT*CON*V12-APxx-DE-P, R911334397 Diese Dokumentation beschreibt die FDT-Container-Funktionalität von IndraWorks. Beschrieben wird das Aktivieren der Funktionalität im Projekt und das Arbeiten mit DTMs.	X	X
/29/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Projektkonvertierung DOK-XLCMLC-PROCONV*V12-APxx-DE-P, R911334186 Diese Dokumentation beschreibt die Projektkonvertierung von IndraLogic 04VRS und IndraMotion MLC04VRS auf IndraWorks Version 12 mit IndraLogic 2G. Es wird insbesondere auf die Änderungen im Bereich Motion und SPS eingegangen.	X	X
/28/	Rexroth IndraMotion MLC 12VRS Inbetriebnahme DOK-MLC***-STARTUP*V12-COxx-DE-P, R911333857 Das vorliegende Handbuch beschreibt die Inbetriebnahme- und Servicearbeitsschritte für das System IndraMotion MLC. Es beinhaltet Checklisten für häufig durchzuführende Arbeiten und eine ausführliche Beschreibung der Arbeitsschritte.	X	

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS PLCopen-Bibliotheken DOK-XLCMLC-FUNLIB**V12-LIxx-DE-P, R911334181 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes, ML_Base und ML_PLCopen für die IndraLogic XLC/IndraMotion MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Generic Application Template DOK-XLCMLC-TF*GAT**V12-APxx-DE-P, R911334190 Dieses Handbuch stellt dem IndraLogic SPS-Programmierer ein strukturiertes Template, welches zum Hinzufügen und Bearbeiten des SPS-Programmiercodes verwendet werden kann, zur Verfügung. Beschrieben werden das Template selbst, der Template Wizard und Beispielapplikationen.	X	X
/31/	Rexroth IndraMotion MLC 12VRS RCL-Programmieranleitung DOK-MLC***-RCL*PRO*V12-APxx-DE-P, R911333851 Das vorliegende Handbuch informiert über die Robot Control. Die Programmiersprache RCL (Robot Control Language) steht dabei im Vordergrund. Beschrieben werden die Programmstruktur, Variablen, Funktionen, Bewegungsanweisungen sowie die notwendigen Parameter des Systems.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Parameter DOK-XLCMLC-PARAM***V12-RExx-DE-P, R911334185 Diese Dokumentation beschreibt die Parameter der Systeme XLC/MLC, sowie das Wechselspiel zwischen Parametrieren und Programmieren. Beschrieben werden die Achs-Parameter, die Steuerungs-Parameter, die Kinematik-Parameter, die Messtaster- und die Nockenschaltwerks-Parameter.	X	X
/10/	Rexroth IndraDrive Firmware für Antriebsregelgeräte MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKxx-DE-P, R911328669		
/11/	Rexroth IndraDrive MPx-16 Funktionen DOK-INDRV*-MP*-16VRS**-APxx-DE-P, R911326484		

Zu dieser Dokumentation

Feldbusse		MLC	XLC
/39/	Rexroth IndraMotion MLC 11VRS PLCopen Feldbus DOK-IM*ML*-PLCFBUS*V11-APxx-DE-P, R911333895 Die Dokumentation beschreibt die Anlage von Feldbusantrieben innerhalb eines IndraWorks-Projekts, Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes.library (Auszug für Feldbusantriebe), RMB_PLCPopenFieldBus.library, RIL_Utilitys.library (Auszug für Feldbusantriebe). Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.	X	X
/4/	Rexroth IndraWorks 12VRS Feldbusse DOK-IWORKS-FB*****V12-APxx-DE-P, R911334393 Diese Dokumentation dient der Beschreibung der unterstützten Feldbusse und ihrer Diagnose-Funktionsbausteine.	X	X

HMI		MLC	XLC
/8/	Rexroth IndraWorks 12VRS HMI DOK-IWORKS-HMI*****V12-APxx-DE-P, R911334391 Diese Dokumentation beschreibt die Funktionen, die Konfiguration und die Bedienung der Bedienoberflächen IndraWorks HMI Engineering und IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 12VRS WinStudio DOK-IWORKS-WINSTUD*V12-APxx-DE-P, R911333843 Diese Dokumentation beschreibt die Installation der Software, das Arbeiten mit WinStudio und das Erstellen und den Betrieb von Applikationen.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS HMI-Anbindung DOK-XLCMLC-HMI*****V12-APxx-DE-P, R911334183 Dieses Dokument beschreibt die von der IndraLogic XLC und IndraMotion MLC unterstützten Visualisierungssysteme sowie deren Anbindung.	X	X

SPS		MLC	XLC
/3/	Rexroth IndraWorks 12VRS IndraLogic 2G Programmieranleitung DOK-IWORKS-IL2GPRO*V12-APxx-DE-P, R911334389 Diese Dokumentation beschreibt das SPS-Programmierwerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.	X	X
/33/	Rexroth IndraWorks 12VRS Basis-Bibliotheken IndraLogic 2G DOK-IL*2G*-BASLIB**V12-LIxx-DE-P, R911333833 Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.	X	X

Technologie			
/30/	Rexroth IndraMotion MLC 12VRS Technologie-Bibliotheken DOK-MLC***-TF*LIB**V12-LIxx-DE-P, R911333867 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" und "ML_TechBase.library" sowie Bibliotheken zur Wicklerfunktionalität, Registerreglerfunktionalität und Querschneiderfunktionalität.	X	
/60/	Rexroth IndraMotion MLC 12VRS RegisterControl (Bibliothek) DOK-MLC***-REGI*CO*V12-LIxx-DE-P, R911333855 Diese Dokumentation beschreibt die Ein- und Ausgänge der einzelnen Funktionsbausteine und gibt Hinweise für deren Verwendung.	X	

Zu dieser Dokumentation

/62/	Rexroth IndraMotion MLC 12VRS RegisterControl (Anwendungsbeschreibung) DOK-MLC***-REGI*CO*V12-APxx-DE-P, R911333853 Dieses Dokument beschreibt die Anwendung der integrierten Registerregelung für eine Tiefdruckmaschine. Beschrieben werden die Komponenten des Markenstromsensors, die HMI-Anwendung und die Fehlerbehebung. Diese Anleitung dient dazu, die Registerregelung zu bedienen und auf Fehler zu reagieren sowie Diagnosen abzufragen. Die Zielgruppe für diese Dokumentation sind Maschineneinrichter und Maschinenbediener.	X	
/49/	Rexroth IndraMotion MLC 12VRS Anwendung Wicklerfunktionen DOK-MLC***-TF*WIND*V12-APxx-DE-P, R911333869 Diese anwendungsbezogene Systemdokumentation beschreibt die Anwendung der Wickler-Technologiefunktionen.	X	

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65 DOK-CONTRL-IC*L45*L65*-PRxx-DE-P, R911324660	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-DE-P, R911328473	X	X
/24/	Rexroth IndraControl Lxx 12VRS Funktionsmodule DOK-CONTRL-FM*LXX**V12-APxx-DE-P, R911333832 Diese Dokumentation dient der Beschreibung aller Funktionsmodule der Lxx Steuerungen inkl. Engineering und Diagnose.	X	X
/12/	Rexroth IndraDrive Antriebsregelgeräte MPx-02 bis MPx-07 DOK-INDRV*-GEN-**VRS**-PAxx-DE-P, R911297316		
/13/	Rexroth IndraDrive MPx-02 bis MPx-07 und HMV DOK-INDRV*-GEN-**VRS**-WAxx-DE-P, R911297318		
/35/	Rexroth IndraDrive Antriebsregelgeräte Steuerteile CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PR08-DE-P, R911295011		

Diagnose und Service		MLC	XLC
/26/	Rexroth IndraMotion MLC/XLC 11VRS Service Tool DOK-IM*ML*-IMST***V11-RExx-DE-P, R911331939	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Diagnosen DOK-XLCMLC-DIAG***V12-RExx-DE-P, R911334179 Diese Dokumentation enthält alle in den Steuerungssystemen IndraLogic XLC und IndraMotion MLC implementierten Steuerungsparameter.	X	X

Systemübersicht		MLC	XLC
/48/	Rexroth IndraMotion for Printing 12VRS Systemübersicht DOK-IM*PR*-SYSTEM**V12-PRxx-DE-P, R911333839 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Printing. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	
/48/	Rexroth IndraMotion for Packaging 12VRS Systemübersicht DOK-IM*PA*-SYSTEM**V12-PRxx-DE-P, R911333837 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Packaging. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	

Zu dieser Dokumentation

/9/	Rexroth IndraMotion MLC 12VRS Systemübersicht DOK-MLC***-SYSTEM**V12-PRxx-DE-P, R911333859 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraMotion MLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.	X	
/9/	Rexroth IndraLogic XLC 12VRS Systemübersicht DOK-XLC***-SYSTEM**V12-PRxx-DE-P, R911333879 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraLogic XLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.		X

Erste Schritte		MLC	XLC
/25/	Rexroth IndraMotion MLC 12VRS Die ersten Schritte DOK-MLC***-F*STEP**V12-COxx-DE-P, R911333845 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraMotion MLC und RobotControl. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.	X	
/25/	Rexroth IndraLogic XLC 12VRS Die ersten Schritte DOK-XLC***-F*STEP**V12-COxx-DE-P, R911333875 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraLogic XLC. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.		X

xx entsprechender Ausgabestand
Abb. 1-1: XLC/MLC-Dokumentationsübersicht

1.3 Darstellung von Informationen

1.3.1 Sicherheitshinweise

Die Sicherheitshinweise, soweit sie in der vorliegenden Anwendungsdokumentation vorhanden sind, beinhalten bestimmte Signalwörter (Gefahr, Warnung, Vorsicht, Hinweis) und ggf. eine Signalgrafik (nach ANSI Z535.6-2006).

Das Signalwort soll die Aufmerksamkeit auf den Sicherheitshinweis lenken und bezeichnet die Schwere der Gefährdung.

Die Signalgrafik (Warndreieck mit Ausrufezeichen), welche den Signalwörtern Gefahr, Warnung und Vorsicht vorangestellt wird, weist auf Gefährdungen für Personen hin.

GEFAHR

Bei Nichtbeachtung dieses Sicherheitshinweises **werden** Tod oder schwere Körperverletzung eintreten.

WARNUNG

Bei Nichtbeachtung dieses Sicherheitshinweises **können** Tod oder schwere Körperverletzung eintreten.

VORSICHT

Bei Nichtbeachtung dieses Sicherheitshinweises können mittelschwere oder leichte Körperverletzung eintreten.

Zu dieser Dokumentation

HINWEIS

Bei Nichtbeachtung dieses Sicherheitshinweises können Sachschäden eintreten.

1.3.2 Verwendete Symbole

Hinweis Hinweise werden wie folgt dargestellt:



Das ist ein Hinweis für den Anwender.

Tipp Tipps werden wie folgt dargestellt:



Das ist ein Tipp für den Anwender.

1.3.3 Bezeichnungen und Abkürzungen

Begriff	Erklärung
JCS	Gelenk-Koordinatensystem = Joint Coordinate System
OCS	Basis-Koordinatensystem = Original Coordinate System
RCL	Programmiersprache Robot Control Language
TCP	Werkzeugarbeitspunkt = Tool Center Point
WCS	Welt-Koordinatensystem = World Coordinate System
WpCS	Werkstück-Koordinatensystem = Workpiece Coordinate System

Abb. 1-2: Verwendete Bezeichnungen und Abkürzungen

2 Koordinaten und Koordinatensysteme

2.1 Allgemeines

In der Robotik nehmen die Begriffe Roboterachsen, Koordinatenachsen und Koordinatensysteme eine zentrale Stellung ein.

Hierbei sind die Namensvergabe für Roboterachsen, die Bezugnahme auf Koordinatensysteme und der Einsatz von Koordinatentransformationen von besonderer Bedeutung.

Im Folgenden werden die in der Robot Control unterstützten Koordinatensysteme vorgestellt.

Im Allgemeinen sind die Koordinatenachsen kartesisch, d.h., rechtwinklig zueinander liegend angeordnet. Die positiven Richtungen der Koordinatenachsen werden durch ein rechtshändiges, rechtwinkliges Koordinatensystem bestimmt.

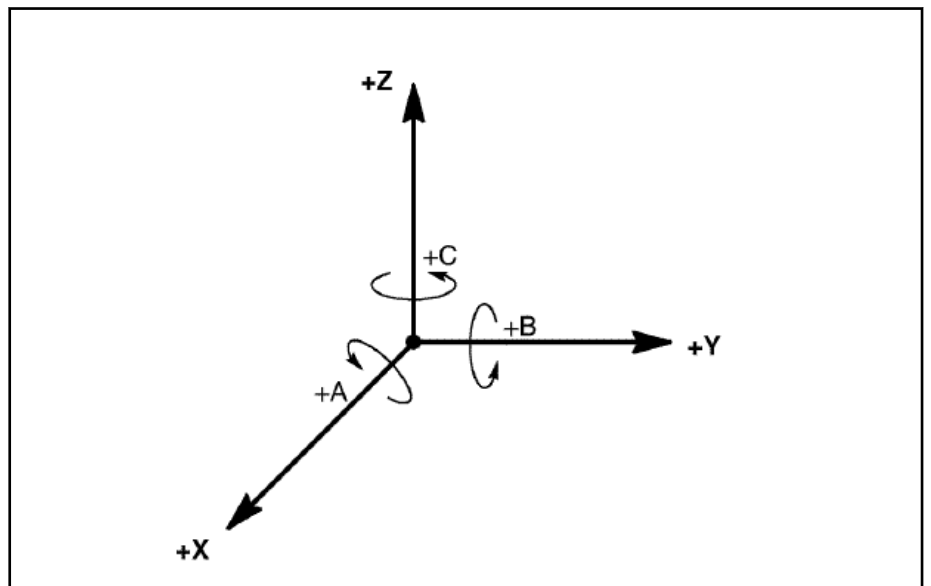


Abb.2-1: Kartesisches Koordinatensystem, Achsbezeichnung

Um ein Objekt in einem kartesischen Koordinatensystem frei positionieren und orientieren zu können werden allg. 6 Freiheitsgrade benötigt. Dabei setzt sich die Lage aus 3 Freiheitsgraden der Positionierung und 3 Freiheitsgraden der Orientierung zusammen.



Die Anordnung der einzelnen Roboterachsen zueinander kann in der Praxis auch nicht rechtwinklig sein. Man erhält damit ein nicht orthogonales Gelenk-Koordinatensystem.

In der Robot Control werden aktuell die folgenden Koordinatensysteme unterstützt:

Übersicht der Koordinatensysteme

- [Gelenk-Koordinatensystem \(JCS\), Seite 16,](#)
- [Basis-Koordinatensystem \(OCS\), Seite 16,](#)
- [Welt-Koordinatensystem \(WCS\), Seite 18,](#)
- [Werkstück-Koordinatensystem \(WpCS\), Seite 19.](#)

2.2 Gelenk-Koordinatensystem (JCS)

Das Gelenk-Koordinatensystem (JCS = Joint Coordinate System) eines Roboters wird durch die Gesamtheit aller Positionierungs- und Orientierungsachsen, die einer Kinematik zugeordnet sind, definiert.

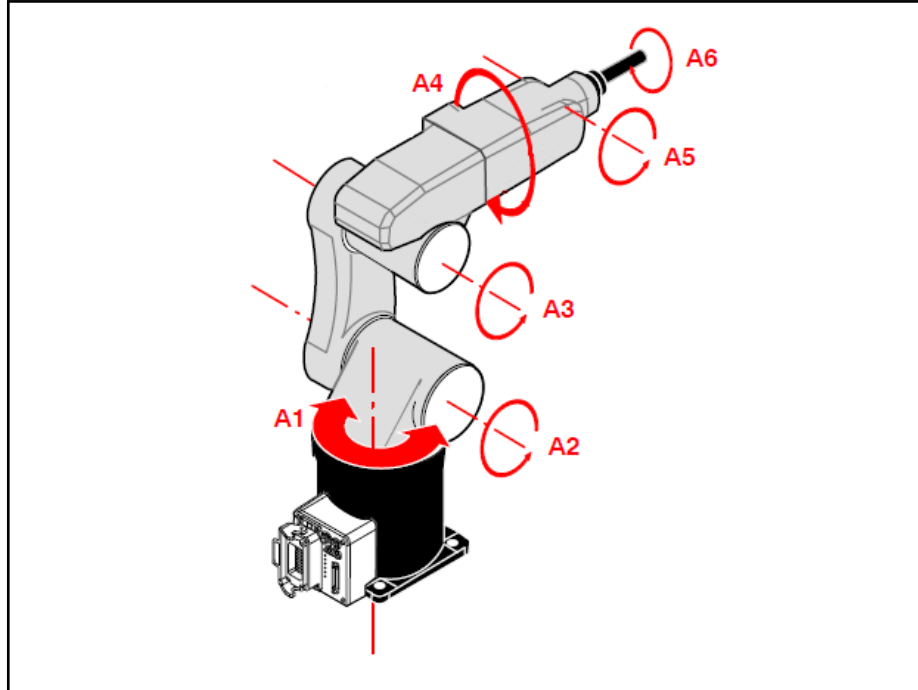


Abb.2-2: Gelenk-Koordinaten (A1,A2,A3,A4,A5,A6)

Zum Einrichten des Gelenk-Koordinatensystems werden die Roboterachsen in die Nullstellungen gebracht. Diese sind der Transformationsvorschrift der jeweiligen Kinematik zu entnehmen. Anschließend sind die Lage-Messsysteme zu referenzieren.

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellungen der Achs-Nullstellungen,
- exakte Vermessung der Armlängen bzw. Außermittigkeiten,
- exakte Ermittlung der Kopplungsfaktoren (falls vorhanden),
- korrekte Drehrichtungen der Achsen.



Ist eine dieser Grundvoraussetzungen nicht zu 100% erfüllt, so kann keine exakte Bewegung des Roboters garantiert werden. Dies kann im ungünstigsten Fall zu falschen und unerwarteten Bewegungen des Roboters führen.

2.3 Basis-Koordinatensystem (OCS)

Das Basis-Koordinatensystem (OCS = Original Coordinate System) ist ein kartesisches, fest mit dem Roboter verbundenes Koordinatensystem.

Koordinaten und Koordinatensysteme

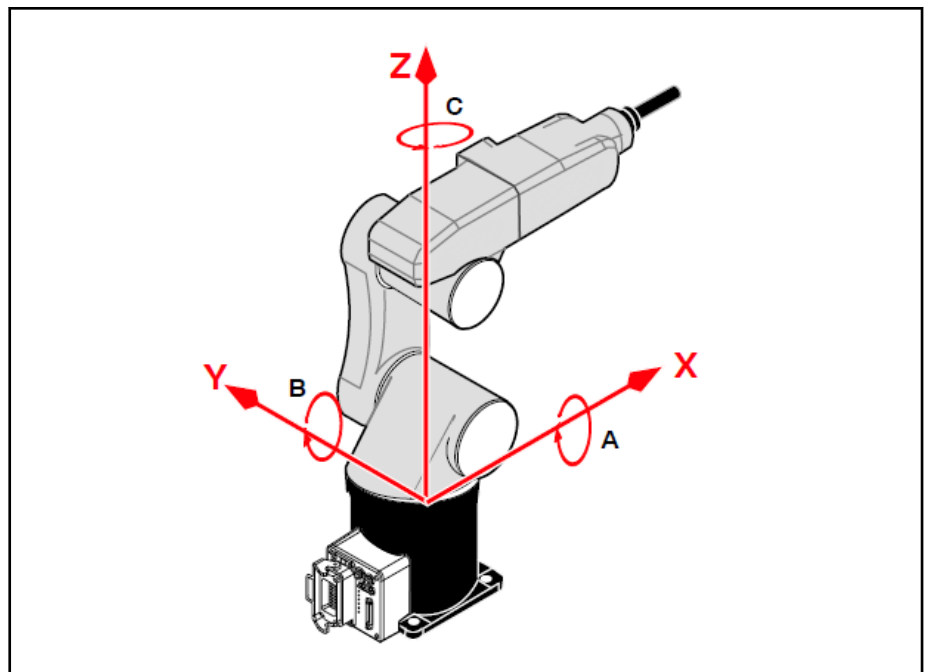


Abb.2-3: Basis-Koordinaten ($X_0, Y_0, Z_0, A_0, B_0, C_0$)

Besitzt die Roboter-Kinematik keinen kartesischen Aufbau, wird die Beziehung zwischen den Gelenk- und den kartesischen Basis-Koordinaten mittels einer Transformation hergestellt.

Kinematische Transformationen in schematischer Darstellung

Vorwärtstransformation / direktes kinematisches Problem

Muss die augenblickliche Lage des Werkzeugarbeitspunktes im Raum ermittelt werden, so ist es am einfachsten, die Gelenk-Koordinaten zu erfassen und daraus die Position und Orientierung des Werkzeuges im Basis-Koordinatensystem zu errechnen.

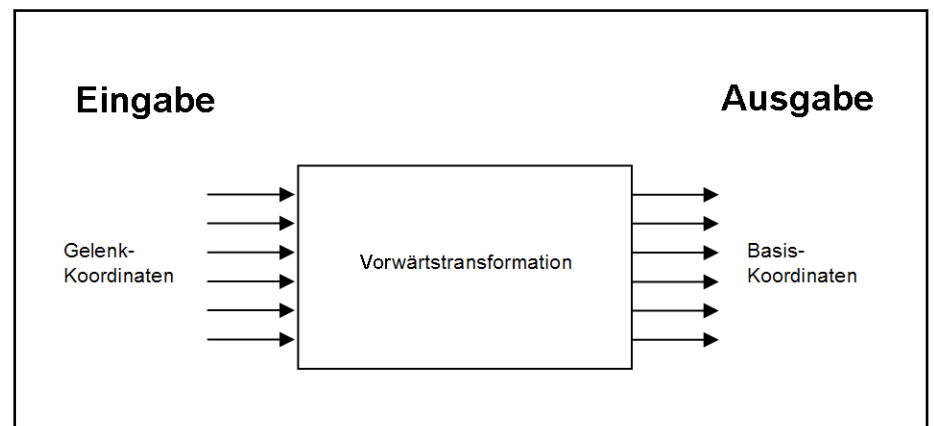


Abb.2-4: Vorwärtstransformation

Rückwärtstransformation / inverse Kinematik

Ist die Lage des Werkzeugarbeitspunktes durch die Position und Orientierung im Basis-Koordinatensystem gegeben, müssen aus diesen Informationen die dazugehörigen Gelenk-Koordinaten berechnet werden.

Koordinaten und Koordinatensysteme

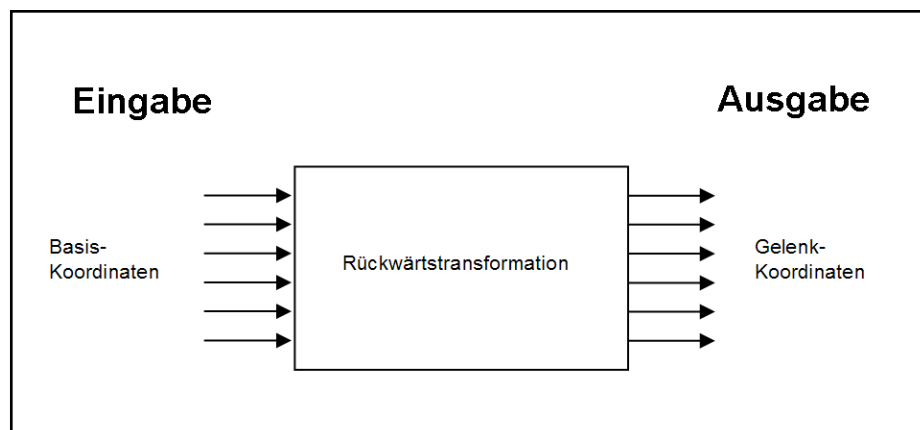
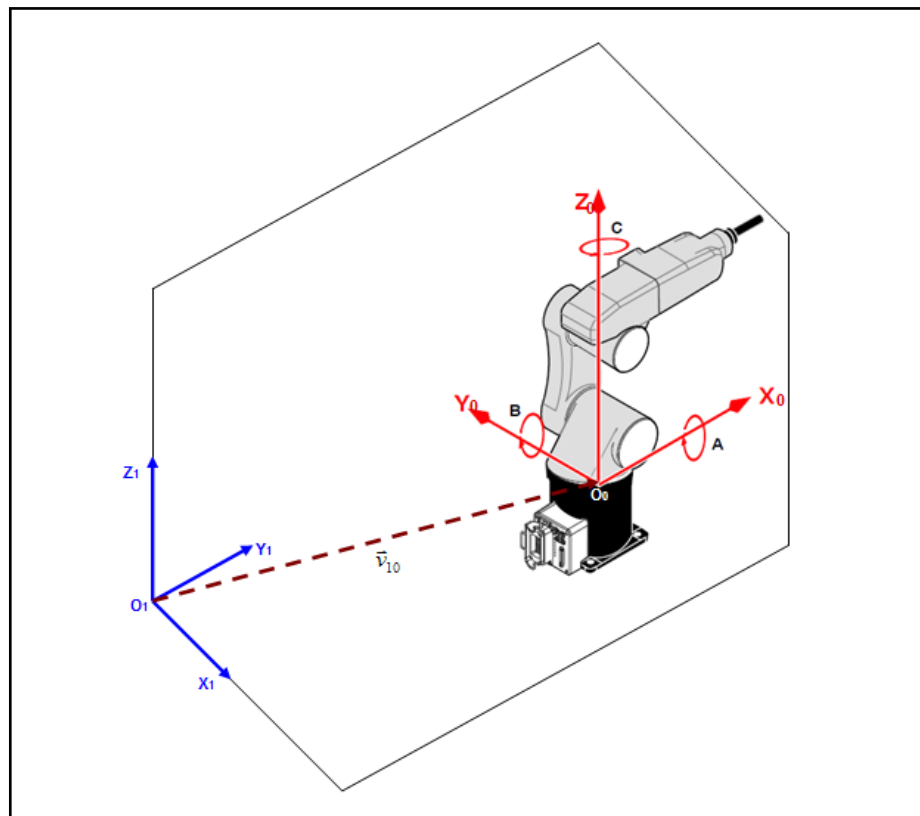


Abb.2-5: Rückwärtstransformation

2.4 Welt-Koordinatensystem (WCS)

Das Welt-Koordinatensystem (WCS = **World Coordinate System**) ist ein ortsfestes Koordinatensystem. Das Welt-Koordinatensystem orientiert sich zu-
meist an einem markanten Punkt der Roboterarbeitszelle.

Abb.2-6: Welt-Koordinaten ($X1, Y1, Z1, A1, B1, C1$)

In der Praxis können innerhalb einer Arbeitszelle mehrere Roboter montiert sein. Überlappen sich die Arbeitsräume der jeweiligen Roboter, so muss die Bewegung der jeweiligen Kinematiken in Bezug zueinander kontrolliert ablaufen.

Hierfür ist es zweckmäßig, die jeweiligen Bewegungen im selben Koordinatensystem darzustellen.

2.5 Werkstück-Koordinatensystem (WpCS)

Das Werkstück-Koordinatensystem (WpCS = **W**orkpiece**C**oordinate **S**ystem) besitzt eine feste Position und Orientierung in Bezug zum Welt-Koordinatensystem.

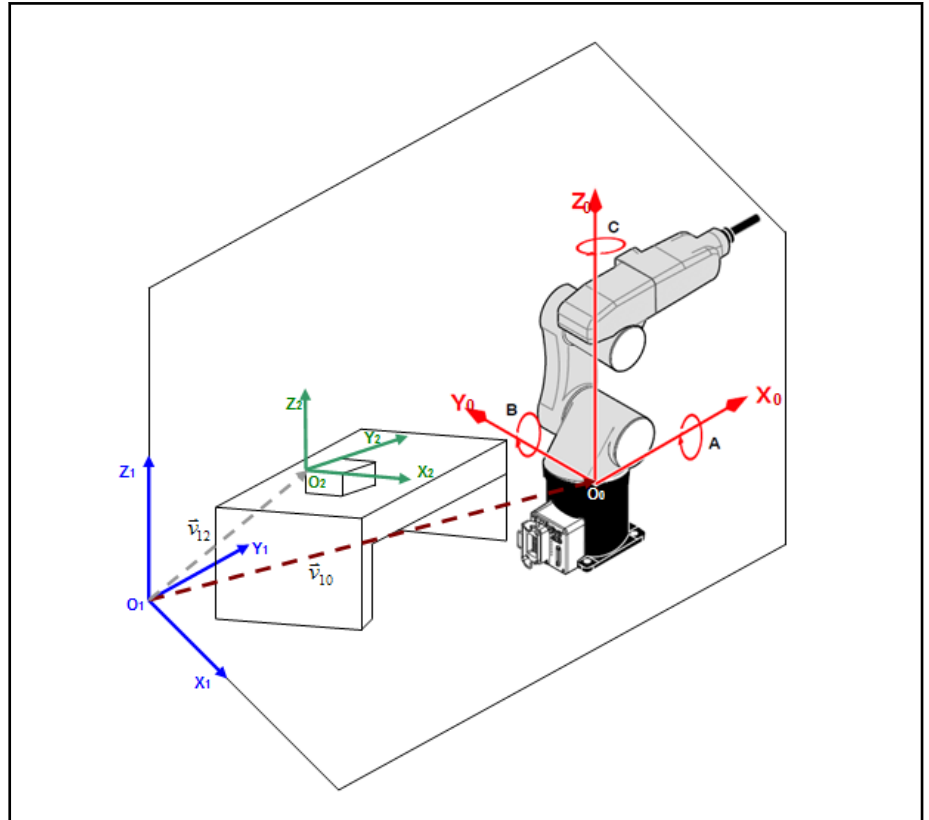


Abb.2-7: Werkstück-Koordinaten ($X_2, Y_2, Z_2, A_2, B_2, C_2$)

Das Werkstück-Koordinatensystem hat insbesondere bei der Programmierung von Bearbeitungsaufgaben eine hohe Bedeutung.

Mit diesen Werkstück-Koordinaten ist es für den Programmierer einfach, die verschiedenen Verarbeitungspositionen am Werkstück exakt anzufahren und die Bearbeitung zu programmieren.



Alle programmierten Bewegungen im RCL-Programm beziehen sich auf das Werkstück-Koordinatensystem.

Die programmierten Koordinaten werden mit dem TCP (TCP = Tool Center Point) angefahren.

Der TCP liegt im Werkzeugarbeitspunkt.

Per Default entspricht das Werkstück-Koordinatensystem dem Weltkoordinatensystem.

3 Anwendung der Koordinatensysteme

3.1 Definition der Koordinatensysteme

Der Übergang zwischen den Koordinatensystemen, d.h. zwischen Welt - und Basis-Koordinaten bzw. zwischen Welt- und Werkstückkoordinaten erfolgt jeweils mit Hilfe von Transformationen in der Steuerung.

Dem Anwender stehen zur Definition der Koordinatensysteme die unten aufgeführten Möglichkeiten zur Verfügung.

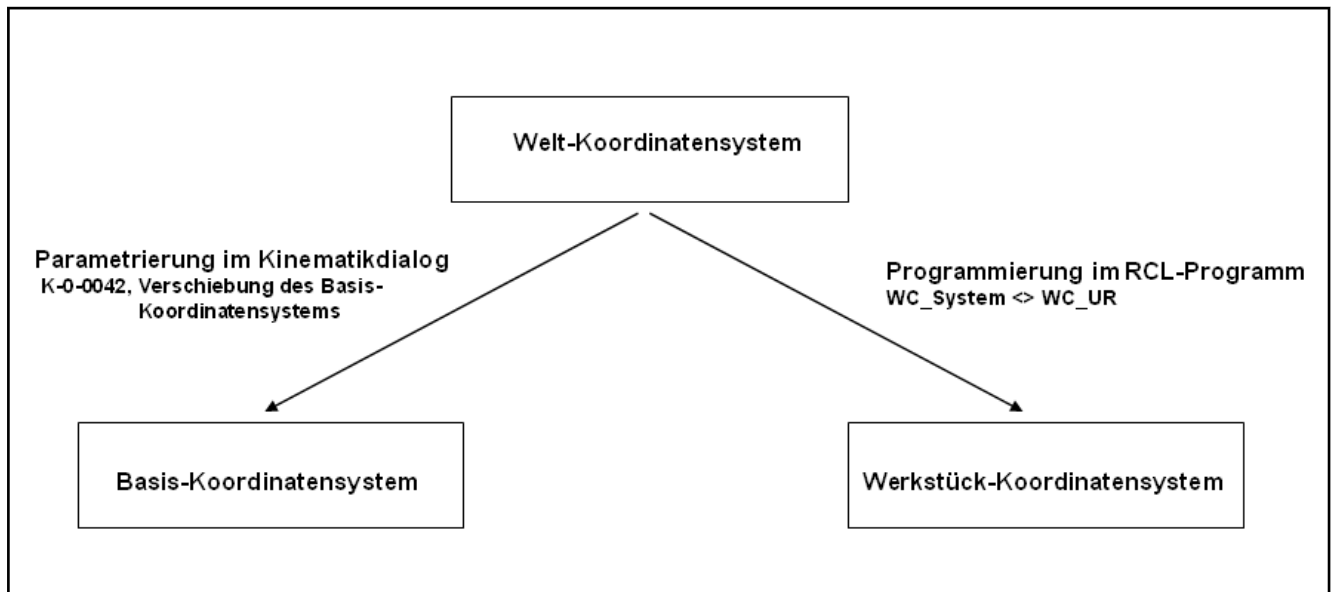


Abb.3-1: Übersicht der Koordinatensysteme

3.2 Definition des Basis-Koordinatensystems relativ zum Welt-Koordinatensystem

Beschreibung der Funktion

Der Zusammenhang zwischen dem Welt-Koordinatensystem und dem Basis-Koordinatensystem wird über den Kinematikparameter K-0-0042, Verschiebung des Raumkoordinatensystems, beschrieben.

Der Parameter K-0-0042 besteht dabei aus 6 Elementen. Drei Elemente dienen der Positionierung und drei Elemente der Orientierung des Basis-Koordinatensystems relativ zum Welt-Koordinatensystem.



Das Werkstück-Koordinatensystem kann nur bei 6-Achs-Kinematiken in voller Allgemeinheit eingesetzt werden. Hat die Kinematik weniger als 6 Freiheitsgrade, so muss über den Parameter K-0-0043 festgelegt werden, welche Achsen berücksichtigt werden. In diesem Fall kann das Werkstück-Koordinatensystem in alle 3 Richtungen beliebig verschoben und um maximal eine Achse verdreht werden.

Welches Element des Parameters K-0-0042 auf welche Verschiebung/ Verdrehung im Raum wirkt, kann im Parameter K-0-0043 eingestellt werden.

Es ist zu beachten, dass die Anzahl der Freiheitsgrade von der verwendeten Kinematik abhängt.

Anwendung der Koordinatensysteme

Der **Verschiebevektor** verläuft vom Ursprung des Welt-Koordinatensystems zum Ursprung des Basis-Koordinatensystems.

Die Verdrehung des Koordinatensystems ist nur für spezielle Transformationschemata (siehe K-0-0031) möglich. Dies ist in den entsprechenden Beschreibungen dokumentiert. (Kein Hinweis über die Verdrehung des Koordinatensystems bedeutet, dass diese Option mit diesem Transformationsschema nicht unterstützt wird).

Relevante Parameter

K-0-0042	Verschiebung des Raumkoordinatensystems
K-0-0043	Werkstückkoordinatenzuordnungsliste
K-0-0031	Transformationsschema

Die Parameter sind im Kapitel [RCL-Parameter](#), Seite 153, näher beschrieben.

Aktivieren

Die Festlegung der Lage des Basis-Koordinatensystems erfolgt in Bezug zum Welt-Koordinatensystem.

Dazu muss der Verschiebevektor in die ersten drei Elemente eingetragen werden.

Die Festlegung der Orientierung erfolgt mit dem Beschreiben der letzten 3 Elemente.

Die Verschiebungen werden in mm und die Verdrehungen in Grad eingegeben.

Zum Beschreiben des Parameters K-0-0042 kann der Kinematikdialog "Koordinatenverschiebung" oder der Parametereditor verwendet werden.



Die Übernahme der Parameter erfolgt mit der Phasenschaltung von P2 nach BB.

Nach der Definition des Basis-Koordinatensystems relativ zum Welt-Koordinatensystems beziehen sich die Koordinaten in der KinData-Schnittstelle

`KinData[x].arPositionOriginalCoord_i`

auf das Welt-Koordinatensystem.

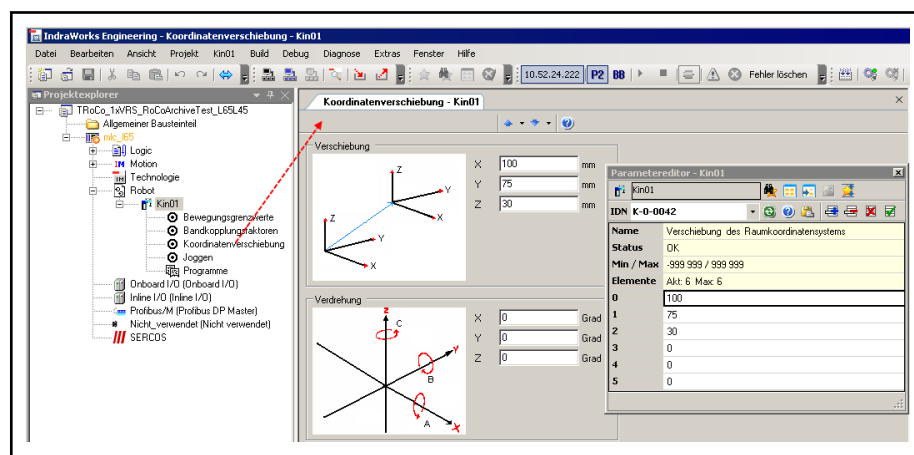


Abb.3-2: Möglichkeiten zur Aktivierung des Basis-Koordinatensystems

Deaktivieren

Alle Elemente des Parameters K-0-0042, Verschiebung des Raumkoordinatensystems sind mit 0 zu belegen.

Anwendung der Koordinatensysteme

Nach der Phasenschaltung P2 nach BB ist die Lage des Basis-Koordinatensystems identisch mit der Lage des Welt-Koordinatensystems. Vorher applizierte Verschiebungen und Verdrehungen werden damit aufgehoben.

Nach dem Ausschalten der Translation und Rotation des Basis-Koordinatensystems beziehen sich die Koordinaten in der KinData-Schnittstelle

`KinData[x].arPositionOriginalCoord_i`

weiterhin auf das Welt-Koordinatensystem, welches nun identisch mit dem Basis-Koordinatensystem ist.

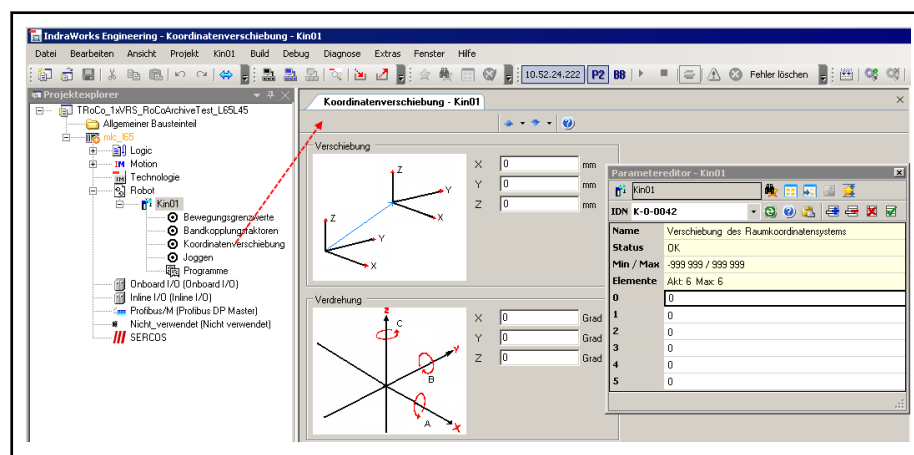


Abb.3-3: Möglichkeiten zur Deaktivierung des Basis-Koordinatensystems

Restriktionen

Es ist zu beachten, dass die Werte des Parameters K-0-0042 erst bei der Phasenschaltung von P2 (P2 = Parametriermodus) nach BB (BB = Betriebsbereit) aktiv werden.

3.3 Definition des Werkstück-Koordinatensystems relativ zum Welt-Koordinatensystem

Beschreibung der Funktion

Der Zusammenhang zwischen dem Werkstück-Koordinatensystem und dem Welt-Koordinatensystem wird über den RCL-Befehl `WC_SYSTEM`, beschrieben. Dieser Befehl besteht aus 6 Elementen. 3 Elemente dienen der Positionierung und 3 weitere der Orientierung des Werkstück-Koordinatensystems. Die Möglichkeit, Verdrehungen zu applizieren, hängt vom Transformationschema ab. Es ist zu beachten, dass die Anzahl der Freiheitsgrade von der verwendeten Kinematik abhängt.

Der Translationsvektor verläuft immer vom Ursprung des Welt-Koordinatensystems zum Ursprung des Werkstück-Koordinatensystems.



Die Verschiebungen werden in mm und die Verdrehungen in Grad eingegeben.

Relevante RCL-Befehle

WC_SYSTEM ist eine kinematikbezogene Standardvariable des Typs `WC_FRAME`. Über diese Standardvariable wird das gültige Werkstück-Koordinatensystem mitgeteilt und aktiviert.

WC_FRAME ist ein kinematikbezogener Standardtyp, der aus 6 Komponenten des Typs `REAL` besteht. Die 6 Komponenten beschreiben die Verschiebung und Verdrehung des Werkstück-Koordinatensystem gegenüber dem Welt-Koordinatensystem.

Die Reihenfolge der Komponenten ist wie folgt:

Anwendung der Koordinatensysteme

1. Komponente	deltaX	x-Richtung
2. Komponente	deltaY	y-Richtung
3. Komponente	deltaZ	z-Richtung
4. Komponente	deltaO1	Erste Orientierungsachse
5. Komponente	deltaO2	Zweite Orientierungsachse
6. Komponente	deltaO3	Dritte Orientierungsachse

Die spezielle Definition und Reihenfolge der 3 Orientierungen hängt vom Robotertyp und ggf. der dort gewählten Orientierungsdefinition ab (vergleiche Dokumentation der entsprechenden Transformationsvorschriften).

WC_UR ist eine vordefinierte Konstante des Typs **WC_FRAME**. Ihre sechs Komponenten haben alle den Wert Null. Wird diese Konstante **WC_SYSTEM** zugewiesen, so werden alle Verschiebungen und Verdrehungen gegenüber dem Welt-Koordinatensystem zu Null.

Aktivieren*Variablen-Deklaration und Zuweisung*

```
<KinematicsName>.WC_FRAME : <WC-VarName>
```

<WC-VarName> ist der vom Anwender frei wählbare Name des Werkstück-Koordinatensystems. Die deklarierte Variable <WC-VarName> vom Typ WC_FRAME kann nun im RCL Programm mit Werten besetzt werden. Nach dem Besetzen mit gültigen Werten muss die Variable <WC-VarName> dem Befehl WC_SYSTEM zugewiesen werden. Mit der Abarbeitung des Befehls durch die Satzabarbeitung wird das Werkstück-Koordinatensystem definiert.

Transformation Welt-Koordinatensystem -> Werkstück-Koordinatensystem

```

;-----
; Transformation Welt-Koordinatensystem -> Werkstück-Koordinatensystem
;-----
; Deklaration der verschiedenen Werkstück-Koordinatensysteme
WC_FRAME      : Palette1, Palette2
REAL          : Komp1, Komp2, Komp3, Komp4, Komp5, Komp6
BEGIN
; Definition der verschiedenen Werkstück-Koordinatensysteme
Palette1 = WC_FRAME(100,100,0,0,0,0)
  Komp1=50
  Komp2=50
  Komp3=75
Palette2 = WC_FRAME(Komp1,Komp2,Komp3,Komp4,Komp5,Komp6)
; Aktivierung des Werkstück-Koordinatensystems der Palette1
WC_SYSTEM = Palette1

```

Die Koordinaten in der KinData-Schnittstelle

KinData[x].arPositionWorkpieceCoords_i

beziehen sich auf das Werkstück-Koordinatensystem, das mit Palette1 relativ zum Welt-Koordinatensystem definiert ist.

Deaktivieren

Ein aktives Werkstück-Koordinatensystem wird folgendermaßen deaktiviert.

WC_SYSTEM = WC_UR

Mit dieser Zuweisung werden alle 6 Komponenten des WC_SYSTEM Befehls mit dem Wert 0 belegt. Nach dem Deaktivieren ist das Werkstück-Koordinatensystem identisch mit dem Welt-Koordinatensystem.

Deaktivierung des Werkstück-Koordinatensystems

```
WC_SYSTEM = WC_UR
```

Anwendung der Koordinatensysteme

Die Koordinaten in der KinData-Schnittstelle

`KinData[x].arPositionWorkpieceCoords_i`

sind gleich den Koordinaten

`KinData[x].arPositionOriginalCoords_i.`

Restriktionen Eine Verdrehung des Werkstück-Koordinatensystem um eine nicht existierende Kinematikachse wird nicht unterstützt.

3.4 Werkstückkoordinaten-Datei

Beschreibung der Funktion Alternativ zur direkten Programmierung gibt es die ASCII-Systemdatei WCSYST.DAT, in der alle lokalen Werkstückkoordinaten namentlich definiert sind

Erzeugen

Die Datei kann, sofern nicht vorhanden, über das Kontextmenü des Orders "Global Data" erzeugt werden. Der Ordner "Global Data" befindet sich unterhalb des Robot-Knotens im Projektexplorer.



Abb.3-4: Werkstückkoordinaten-Datei erzeugen

Der Ordner "Global Data" kann vom Anwender jederzeit gelöscht beziehungsweise über das Kontextmenü des Robot-Knotens wieder erzeugt werden.



Beim Löschen des Ordners "Global Data" werden auch alle in diesem Ordner abgelegten Dateien gelöscht !

Bearbeiten

Über das Kontextmenü der Datei "WCSYST.DAT" kann der Editor für die Eingabe beziehungsweise Bearbeitung der Werkstückkoordinaten geöffnet werden

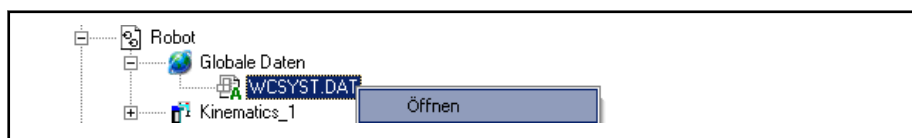


Abb.3-5:

3.5 Werkstückkoordinaten-Editor

Allgemeines Mit dem Editor können die Werkstückkoordinaten eingegeben und bearbeitet werden.

Anwendung der Koordinatensysteme

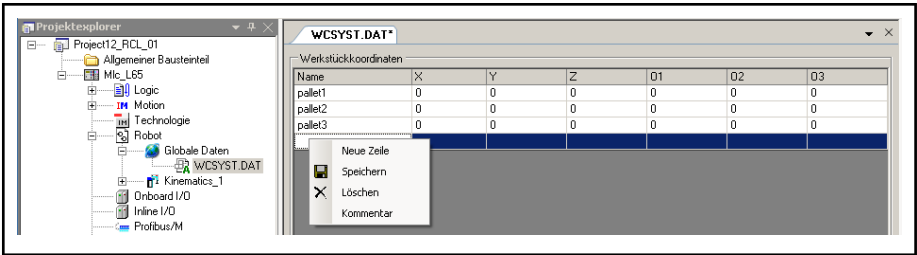


Abb.3-6: Editor

Die einzelnen Raumkoordinaten werden mit einem vom Benutzer frei wählbaren Namen benannt. Unter diesem Namen werden die zugehörigen Koordinatenwerte in der Datei "WCSYST.DAT"abgelegt.

 Der Name darf maximal 12 Zeichen lang sein

Für die Koordinatenwerte sind nur die Eingaben "0" ... "9", ".", "+", und "-" erlaubt

Der Editor bietet folgende Funktionen:

- Neue Zeile einfügen,
- Speichern der Werkstückkoordinaten nach Datei,
- Zeile löschen,
- Zeilenkommentar eingeben,

3.6 Bibliotheksfunktionen

Die Klasse 2000 der RCL-Bibliotheksfunktionen enthält Bibliotheksfunktionen, welche den Einsatz allgemeiner Raum-Koordinatensysteme erleichtern. Zu der exakten Beschreibung der Parameterlisten sei auf die Header-Datei "rk_En.inc" verwiesen.

rKxGWcSystem Liefert Name und Koordinaten des aktiven RK-Systems. Der Funktion muss eine Struktur von Typ TKxGWcSystem übergeben werden. Diese Struktur ist wie folgt aufgebaut:

Übergabeparameter TKxGWcSystem der Funktion rKxGWcSystem()

Name	Typ	Datentyp	Bedeutung
ChannelId	Eingang	INTEGER	Als ChannelId muss der Wert 0 gesetzt werden.
KinNr	Eingang	INTEGER	Kinematik-Nummer Zulässig sind ganze Zahlen 1 bis n. Die maximal zulässige KinNr ist von der Einstellung des Parameter C-0-2484 abhängig
WcSysCoord	Ausgang	FARRAY_WC	Feld enthält die sechs Koordinaten-Werte des Werkstückkoordinatensystems

Anwendung der Koordinatensysteme

WcSysName	Ausgang	NAMSTRING	Name des aktiven Werkstückkoordinatensystems
Reserved	Reserviert	ARRAY[1..32] IN-TEGERS	Reserviert

Abb.3-7: Übergabeparameter Funktion rKxGWcSystem()

Mögliche Rückgabewerte der Funktion:

Wert	Bedeutung
RK_NO_ERROR	Kein Fehler, die Funktion wurde korrekt ausgeführt
RK_KIN_NR	Unzulässige Kinematik-Nummer

Abb.3-8: Mögliche Rückgabewerte der Funktion:

rKSWcSystem

Aktiviert das namentlich ausgewählte RK-System. Der Name und die zugehörigen Koordinatenwerte müssen in der System-Datei "RKSYS.DAT" definiert sein.

VORSICHT

Bei Benutzung dieser Funktion werden die Eigenschaften der Kinematik verändert (Tool Center Point).

Der Benutzer hat die einschlägigen Sicherheitsvorkehrungen zu treffen und die geltenden Unfallverhütungsvorschriften einzuhalten, so dass es zu keinerlei Gefährdung von Personen und Gerätschaften kommen kann. Für die bestimmungsgemäße Anwendung ist alleine der Benutzer dieser Funktion verantwortlich.

Der Funktion muss eine Struktur von Typ TKSWhSystem übergeben werden. Diese Struktur ist wie folgt aufgebaut:

Übergabeparameter TKSWhSystem der Funktion rKSWcSystem()

Name	Typ	Datentyp	Bedeutung
ChannelId	Eingang	INTEGER	Als ChannelId muss der Wert 0 gesetzt werden.
KinNr	Eingang	INTEGER	Kinematik-Nummer Zulässig sind ganze Zahlen 1 bis n. Die maximal zulässige KinNr ist von der Einstellung des Parameter C-0-2484 abhängig
WcSysName	Eingang	NAMSTRING	Name des zu aktivierenden Werkstückkoordinatensystems Die Funktion sucht den Namen in der Datei WCSYST.DAT
Reserved	Reserviert	ARRAY[1..32] IN-TEGERS	Reserviert

Abb.3-9: Übergabeparameter Funktion rKSWcSystem()

Anwendung der Koordinatensysteme

Mögliche Rückgabewerte der Funktion:

Wert	Bedeutung
RK_NO_ERROR	Kein Fehler, die Funktion wurde korrekt ausgeführt
RK_KIN_NR	Unzulässige Kinematik-Nummer
RK_HEAPOVERF	Heap-Überlauf
RK_TOOL_FILE	Fehler in WCSYST.DAT: Name des Werkstückkoordinatensystems fehlt in WCSYST.DAT oder WCSYST.DAT fehlt
RK_WRONGK43	Unzulässiger Maschinenparameter K-0-0043

Abb.3-10: Mögliche Rückgabewerte der Funktion:

rKCWpFrame

Berechnet aus 3 Positionen P0, P1, P2, aufgenommen in Koordinaten des Ur-Koordinatensystems RK_{ur} , den zugehörigen Nullpunkt des lokalen RK-Systems RK_{loc} inklusive Verdrehung gegenüber dem Ur-Koordinatensystem.



Die Funktion ist nur einsetzbar bei Kinematiken, welche alle 6 Freiheitsgrade besitzen; d.h. welche mindestens 6 Achsen haben. Andernfalls ist das Problem mathematisch unterbestimmt und es wird der Return-Code (RK_AX_NR = -2) ausgegeben.

Zur Berechnung müssen folgende Voraussetzungen erfüllt sein:

- Die 3 Punkte P0, P1 und P2 liegen in einer Palettenebene, d.h. sie haben die gleiche z_{loc} -Koordinate.
- Der Vektor P0P1 liegt parallel zur x-Achse
- Die (x, y, z)-Koordinaten des Punktes P0 im lokalen RK-System RK_{loc} müssen bekannt sein.

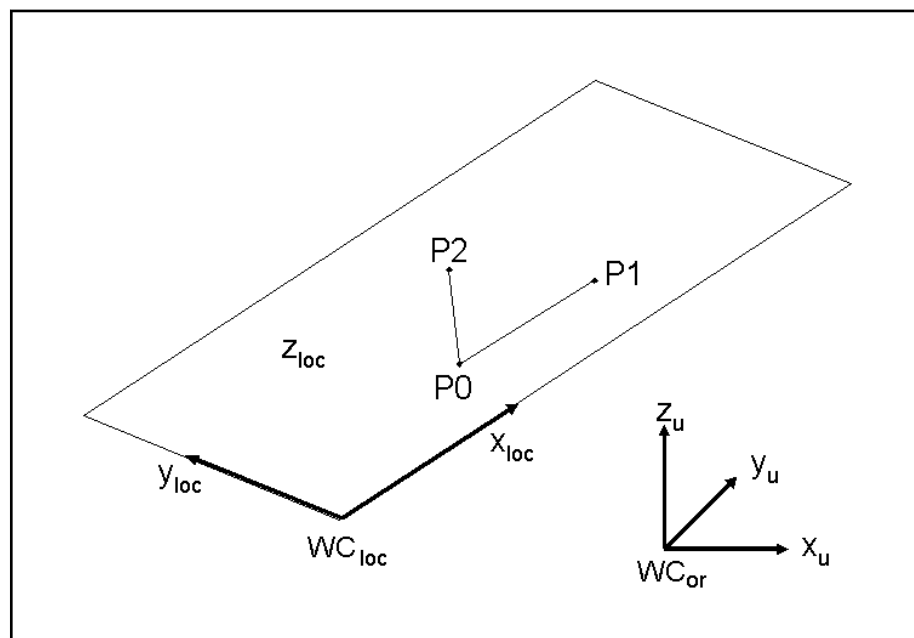


Abb.3-11: Parameter der Funktion rKCWpFrame schematisch

Anwendung der Koordinatensysteme

Übergabeparameter TKCWpFrame der Funktion rKCWpFrame()

Name	Typ	Datentyp	Bedeutung
ChannelId	Eingang	INTEGER	Als ChannelId muss der Wert 0 gesetzt werden.
KinNr	Eingang	INTEGER	Kinematik-Nummer Zulässig sind ganze Zahlen 1 bis n. Die maximal zulässige KinNr ist von der Einstellung des Parameter C-0-2484 abhängig
P0loc	Eingang	FARRAY_AXIS	Position des Punktes P0 lokalen Werkstück-Koordinaten
P0base	Eingang	FARRAY_AXIS	Position des Punktes P0 in Roboter-Urkoordinaten
P1base	Eingang	FARRAY_AXIS	Position des Punktes P1 in Roboter-Urkoordinaten
P2base	Eingang	FARRAY_AXIS	Position des Punktes P2 in Roboter-Urkoordinaten
WpFrame	Ausgang	FARRAY_WC	Koordinaten des lokalen Werkstück-Koordinatensystems. Das Feld enthält sechs Werte, welche die Position (drei Werte) und die Orientierung (drei Werte) des lokalen Werkstück-Koordinatensystems in Einheiten des Roboter-Urkoordinatensystems beschreiben. Die Grundebene des Werkstück-Koordinatensystems wird von den drei Punkten P0, P1 und P2 aufgespannt.
Reserved	Reserviert	ARRAY[1..32] INTEGER	Reserviert

Abb.3-12: Übergabeparameter Funktion rKCWpFrame()

Mögliche Rückgabewerte der Funktion:

Wert	Bedeutung
RK_NO_ERROR	Kein Fehler, die Funktion wurde korrekt ausgeführt
RK_KIN_NR	Unzulässige Kinematik-Nummer
RK_AX_NR	Kinematik besitzt nicht alle sechs Freiheitsgrade
RK_ANGLENOTC	Winkel sind nicht berechenbar, Punkte P0, P1, P2 sind ungeeignet

Abb.3-13: Mögliche Rückgabewerte der Funktion:

Anwendung der Koordinatensysteme

rKTrfLocOri Berechnet aus dem lokalen Koordinaten eines Punktes die zugehörigen Ur-Koordinaten.

 Die Bibliotheksfunktion kann nur bei 6-Achs-Kinematiken eingesetzt werden. Bei fehlenden Freiheitsgraden ist die Funktion unterbestimmt.

Die Funktion arbeitet namensorientiert bezüglich des lokalen Werkstückkoordinatensystems. Der Name mit den zugehörigen Koordinaten wird in der Systemdatei WCSYST.DAT der Steuerung erwartet. P_{loc} entspricht dem Punkt in Koordinaten des lokalen Werkstückkoordinatensystems. P_{ur} ist der gleiche Punkt in Koordinaten des Ur-Koordinatensystem RK_{ur} .

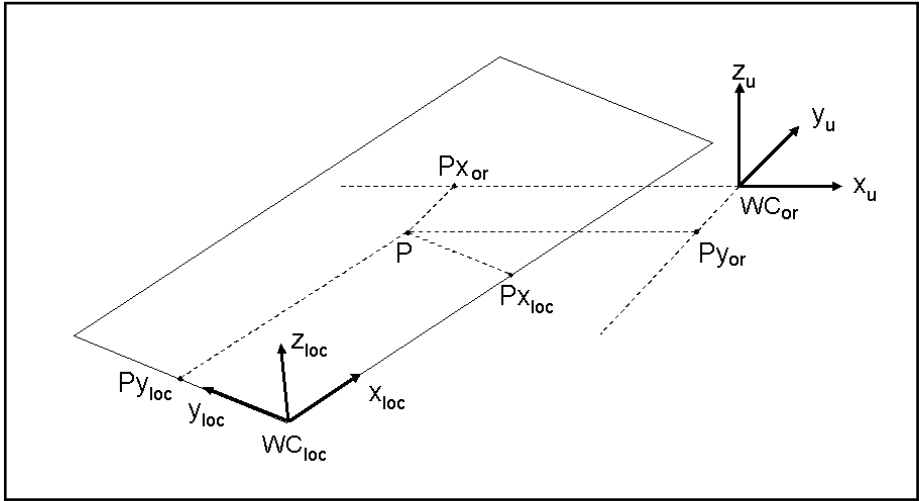


Abb.3-14: Punkt P im Koordinatensystem RK_{loc} und RK_{ur} schematisch

Übergabeparameter TKTrfLocOri der Funktion rKTrfLocOri()

Name	Typ	Datentyp	Bedeutung
ChannelId	Eingang	INTEGER	Als ChannelId muss der Wert 0 gesetzt werden.
KinNr	Eingang	INTEGER	Kinematik-Nummer Zulässig sind ganze Zahlen 1 bis n. Die maximal zulässige KinNr ist von der Einstellung des Parameter C-0-2484 abhängig
PntLoc	Eingang	FARRAY_AXIS	Punkt in Koordinaten des lokalen Werkstückkoordinaten-Systems 'WcSysName'
WcSysName	Eingang	NAMSTRING	Name des lokalen Werkstückkoordinaten-Systems Die Funktion sucht den Namen in der System-Datei WCSYST.DAT

Anwendung der Koordinatensysteme

PntOri	Ausgang	FARRAY_AXIS	Zugehörige Urkoordinaten des Punktes PntLoc; d.h. Transformation des Punktes PntLoc ins Ur-System des Roboters.
Reserved	Reserviert	ARRAY[1..32] IN-TEGER	Reserviert

Abb.3-15: Übergabeparameter Funktion rKTrfLocOri()

Mögliche Rückgabewerte der Funktion:

Wert	Bedeutung
RK_NO_ERROR	Kein Fehler, die Funktion wurde korrekt ausgeführt
RK_KIN_NR	Unzulässige Kinematik-Nummer
RK_TOOL_FILE	Fehler in WCSYST.DAT: Name des Werkstückkoordinatensystems fehlt in WCSYST.DAT oder WCSYST.DAT fehlt

Abb.3-16: Mögliche Rückgabewerte der Funktion:

rKTrfOriLoc

Berechnet aus den Ur-Koordinaten eines Punktes die zugehörigen lokalen Koordinaten.



Die Bibliotheksfunktion kann nur bei 6-Achs-Kinematiken eingesetzt werden. Bei fehlenden Freiheitsgraden ist die Funktion unterbestimmt.

Die Funktion arbeitet namensorientiert bezüglich des lokalen Werkstückkoordinatensystems. Der Name mit den zugehörigen Koordinaten wird in der Systemdatei WCSYST.DAT der Steuerung erwartet. P_{loc} entspricht dem Punkt in Koordinaten des lokalen Werkstückkoordinatensystems. P_{ur} ist der gleiche Punkt in Koordinaten des Ur-Koordinatensystems RK_{ur} .

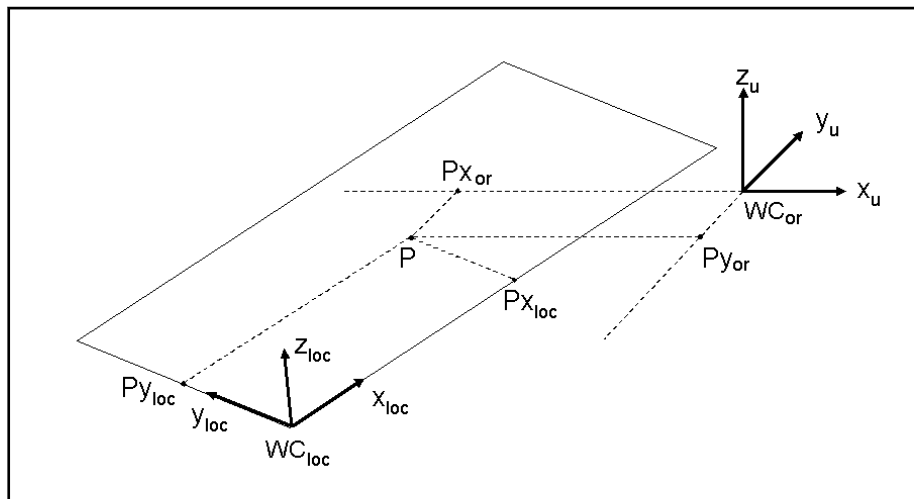


Abb.3-17: Punkt P im Koordinatensystem RK_{loc} und RK_{ur} schematisch

Übergabeparameter TKTrfOriLoc der Funktion rKTrfOriLoc()

Name	Typ	Datentyp	Bedeutung
ChannelId	Eingang	INTEGER	Als ChannelId muss der Wert 0 gesetzt werden.

Anwendung der Koordinatensysteme

KinNr	Eingang	INTEGER	Kinematik-Nummer Zulässig sind ganze Zahlen 1 bis n. Die maximal zulässige KinNr ist von der Einstellung des Parameter C-0-2484 abhängig
PntOri	Eingang	FARRAY_AXIS	Punkt in Ur-Koordinaten des Roboters
WcSysName	Eingang	NAMSTRING	Name des lokalen Werkstückkoordinaten-Systems Die Funktion sucht den Namen in der System-Datei WCSYST.DAT
PntLoc	Ausgang	FARRAY_AXIS	Zugehörige Koordinaten des Punktes PntOri in Wertes des Werkstückkoordinatensystems; d.h. Transformation des Punktes PntOri ins lokale Werkstückkoordinatensystem.
Reserved	Reserviert	ARRAY[1..32] INTEGER	Reserviert

Abb.3-18: Übergabeparameter Funktion rKTrfOriLoc()

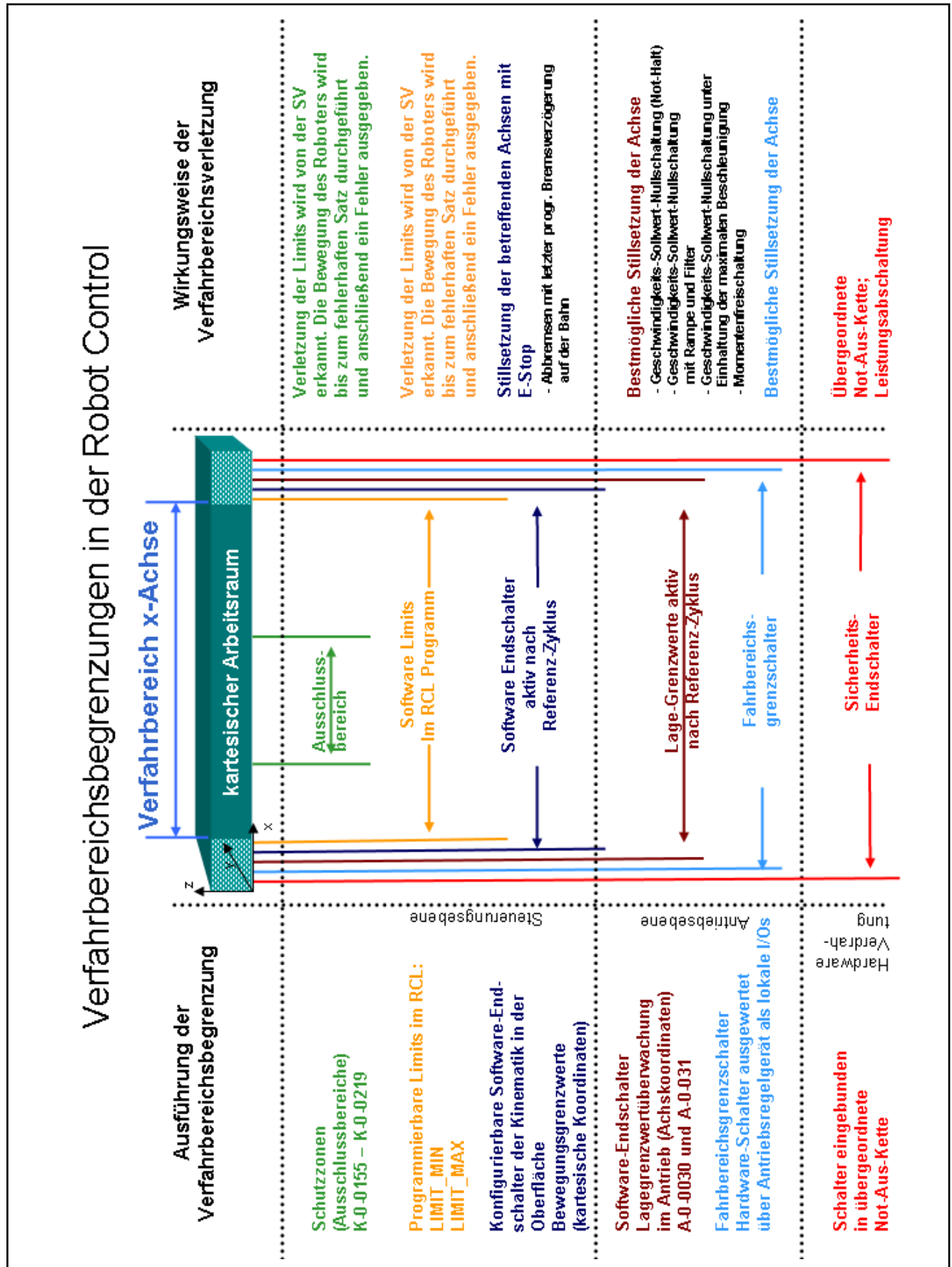
Mögliche Rückgabewerte der Funktion:

Wert	Bedeutung
RK_NO_ERROR	Kein Fehler, die Funktion wurde korrekt ausgeführt
RK_KIN_NR	Unzulässige Kinematik-Nummer
RK_TOOL_FILE	Fehler in WCSYST.DAT: Name des Werkstückkoordinatensystems fehlt in WCSYST.DAT oder WCSYST.DAT fehlt

Abb.3-19: Mögliche Rückgabewerte der Funktion:

4

Verfahrbereichsbegrenzungen in der Robot Control



Verfahrenbereichsbegrenzungen in der Robot Control

SV Satzvorbereitung
Abb.4-1: *Verfahrenbereichsbegrenzungen*

5 RCL-Programmstruktur

5.1 Allgemeines

RCL steht für "Robot Control Language" und ist eine aufgabenorientierte Programmiersprache zur Programmierung der MLC-Steuerungsfamilie.

Als Programmiersprache für Roboter- und Handling-Systeme enthält RCL einen mächtigen aber leicht zu erlernenden Sprachumfang. Dieser ermöglicht dem Anwender eine schnelle und wartungsfreundliche Realisierung seiner Aufgabenstellung.

Die Sprachbefehle sind in englischer Sprache.

In dem vorliegenden Dokument wird vor jeder ausführlichen Anweisungsbeschreibung die allgemeine Syntax der Anweisung angegeben.

Dabei werden folgende Symbole zur Beschreibung verwendet:

GROSS- BUCHSTABEN	bedeutet Teil des Sprachelementes, also ein RCL-Schlüsselwort und muss genauso geschrieben werden
{ }	kann optional mehrmals angegeben werden
[]	kann optional einmal angegeben werden



Symbolische Bezeichner wie z.B. Variablennamen dürfen aus nicht mehr als 12 Zeichen bestehen.

Das erste Zeichen muss ein Buchstabe sein, Buchstaben und Ziffern sind in Folge erlaubt, keine Umlaute, kein "ß".

5.2 Arbeitsweise des Compilers

5.2.1 Verwendung

Der RCL-Compiler kann auf zwei verschiedene Arten verwendet werden.

- In IndraWorks ist ein Editor vorhanden um ein RCL-Programm einfach und schnell erstellen und kompilieren zu können.
- Ist schon eine Quelldatei vorhanden (z.B. über einen Texteditor erstellt worden) besteht die Möglichkeit den Compiler über die Parameter K-0-0221 und K-0-1062 zu starten.

5.2.2 Quelldateien

qll-Datei Diese Textdatei enthält den RCL-Programmcode.
Der Datei- und der Programmname müssen identisch sein.

5.2.3 Ausgabedateien

Die Ausgabedateien werden beim Compilervorgang erzeugt. Der Name der Ausgabedateien setzt sich zusammen aus dem Namen der qll-Datei und der entsprechenden Endung (z.B. aus "Prog1.qll" entsteht "Prog1.ird").

ird-Datei Diese Datei enthält den Programmcode den die Steuerung abarbeiten soll.
Außerdem wird der Speicherbereich der für die im Programm verwendeten Variablen benötigt wird festgelegt. Für Punktvariablen die nicht mit DEF vereinbart sind und denen im Programm ein Wert zugewiesen wird, wird in dieser Datei ebenfalls Speicherplatz reserviert.

Diese Datei wird nur erzeugt, wenn das Programm fehlerfrei übersetzt wurde.

RCL-Programmstruktur

pkt-Datei	In dieser Datei ist der Speicherbereich für die Punktvariablen reserviert. Es wird Speicher für Punktvariablen reserviert die • im Programm mit DEF vereinbart sind. • nicht vereinbart sind. • denen im Programm kein Wert zugewiesen wird.
sym-Datei	Diese Datei enthält Informationen über die im Programm verwendeten Variablennamen. Zum Debuggen ist diese Datei notwendig.
err-Datei	Diese Textdatei enthält eine Übersicht über die Fehler die beim Compilervorgang aufgetreten sind.

5.3 Compileranweisungen

5.3.1 Allgemeines

in der qll-Datei sind Compileranweisungen enthalten. Diese dienen der Steuerung des Compilers und reduzieren den Schreibaufwand.

Die Compileranweisung beginnt immer mit zwei Semikolons.

Compileranweisungen:

```
;;COMPILER = RCL3 ; RCL Compiler Version
;;INCLUDE name ; optional
;;[kin_name.]INT = CIRCULAR | PTP | LINEAR
;;KINEMATICS: ({ integer_constant = kin_variable | , })
;;KINEMATICS = kin_variable
;;(kin_variable.JC_NAMES | WC_NAMES) = { name || , }
;;DEBUGINFO [+/-] ; optional
```

Die Compileranweisung für die Achsnamen und für die Zielsteuerung muss vor dem ersten Quellsymbol stehen, d. h. vor der Programmvereinbarung.

5.3.2 Kinematikdefinition

Ab MLC 04V12 ist es möglich, anstelle einer Kinematik bis zu 16 Kinematiken pro Steuerung zu unterstützen (siehe auch [Anwendung der RoCo-Multi-kinematik-Technik, Seite 219](#)).

Jede dieser Kinematiken wird von einem eigenen Programm gesteuert.

Positionier- und Orientierungsachsen sollten jeweils nur einer Kinematik zugeordnet werden.

Bandachsen können mehreren Kinematiken zugeordnet werden.

Technologiebedingt ist es möglich, Achsen und Bänder von einer Kinematik an eine andere zu übergeben und steuern zu lassen.

5.3.3 Welt-Koordinatennamen (WC name definition)

Die Raumkoordinatenpunkte (world coordinate points) enthalten die Komponenten für die Position, die Orientierung und ggf. für die Bandkoordinate(n). Die Komponentennamen können definiert werden.

Syntax

```
;;WC_NAMES=x_k,y_k,z_k,a_k ;WC_NAMES=WC_name,...
```

Beispiel mit Kinematiknamen

```
;;Kin_1.WC_NAMES=x_k,y_k,z_k,a_k ;kinematics name.WC_NAMES=WC name,...
```



Die Koordinatendeklaration muss in einer Zeile stehen und darf nicht durch Zeilenumbruch unterbrochen sein.

Beim Anlegen einer Kinematik in IndraWorks werden die Namen für die WC-Achsen eingegeben und beim Erzeugen einer neuen qll-Datei automatisch angelegt.

Sie können nachträglich verändert werden.

5.3.4 Achsnamen (JC name definition)

Die Maschinenkoordinatenpunkte enthalten die Komponenten für die einzelnen Achsen und ggf. für die Bandkoordinate(n).

Die Achsnamen können definiert werden.

Syntax

```
;;JC_NAMES=a_1,a_2,a_3,a_4 ;JC_NAMES=JC name,...
```

Beispiel mit Kinematiknamen

```
;;Kin_1.JC_NAMES=a_1,a_2,a_3,a_4
```



Die Achsnamendeklaration muss in einer Zeile stehen und darf nicht durch Zeilenumbruch unterbrochen sein.

Beim Erzeugen einer neuen qll-Datei werden die Achsnamen von IndraWorks automatisch übernommen.

Sie können nachträglich geändert werden.

5.3.5 Einfügen von Dateien

Mit der Übersetzeranweisung ;;INCLUDE 'Dateiname' können Teile von Quellprogrammen in das Programm eingefügt werden.

Beispiel

```
;;INCLUDE rcl ;rcl.qll, Datei zum automatischen Einfügen
;des Vereinbarungsteils beim Übersetzen
INPUT REAL: 1=grip_contact,4=measheight ;Deklaration von Eingängen
INPUT: 11=gateswitch_1,15=li_barrier
OUTPUT REAL: 1=print,2=met_unit ;Deklaration von Ausgängen
OUTPUT: 7=alarm
```

Die Datei "rcl.qll" beinhaltet z. B. die Vereinbarungen ihrer Ein- und Ausgänge. Diese sind festgelegt bezüglich

- Datentyp
- Kanalnummer
- Variablenname des Signals.

Details siehe [Kommunikationskanäle zur SPS, Seite 64](#).

RCL-Programmstruktur



Die Datei rcl.qll muss im selben Verzeichnis, wie die übrigen zu übersetzenden Dateien stehen.

5.3.6 Wahlfreie Extension innerhalb der Einfügeanweisung

Die Extension bei Einfügedateien kann frei gewählt werden. Wird nur der Dateiname angegeben, wird aus Kompatibilitätsgründen die Extension "qll" unterlegt.

Soll eine Datei ohne Extension eingebunden werden, ist der Dateiname mit einem Punkt abzuschließen, siehe Beispiel Einfügeanweisung.

Syntax

```
;;INCLUDE filename ;Compiler statement 'INCLUDE'
```

In obiger Syntax gilt:

filename: Name der Datei, die bei der Übersetzung eingebunden werden soll.

Beispiel

```
;;INCLUDE head.inc
PROGRAM demo
;;INCLUDE constant ;constant.qll file is included
;;INCLUDE types.INC ;File types.INC are included
;;INCLUDE variable. ;File variable. (without extension) is included
BEGIN
PROGRAM_END
```



Die Dateien head.inc, demo.qll, constant.qll, types.inc und variable müssen in einem Verzeichnis liegen.



Einfüge-Dateien selbst dürfen keine INCLUDE-Befehle beinhalten (keine Schachtelung möglich).

5.3.7 Testinformation

Mit der Übersetzeranweisung ";;DEBUGINFO-" kann man die Erstellung von Testinformationen zum Irdata-Code für den komfortablen Testbetrieb abschalten.

Das ist im allgemeinen nur für vollständig ausgetestete Anwendungsprogramme sinnvoll.

Es fehlen dann die Debug-Zusatzinformationen im Irdata-Code, der Irdata-Code ist kürzer und wird schneller abgearbeitet, jedoch ist ein Test mit dem Testsystem dann nicht mehr möglich.

; ;DEBUGINFO+ ist voreingestellt.

Syntax

```
;;DEBUGINFO- ;Test info generation is switched off
```

Weitere Einschränkungen bei "DEBUGINFO-":

Eventuell auftretende Programmfehler werden ebenfalls ohne Zeilennummer angezeigt.

Mit ";;DEBUGINFO+" wird die Erzeugung von Testinformation wieder eingeschaltet.

5.4 Hauptprogramm-Aufbau

5.4.1 Allgemeines

Jedes Hauptprogramm (Main program) besteht aus

- Deklarationsteil (Declarations)
- Anweisungsteil (Instructions) und optional
- Unterprogrammvereinbarung(en) (Subroutines).

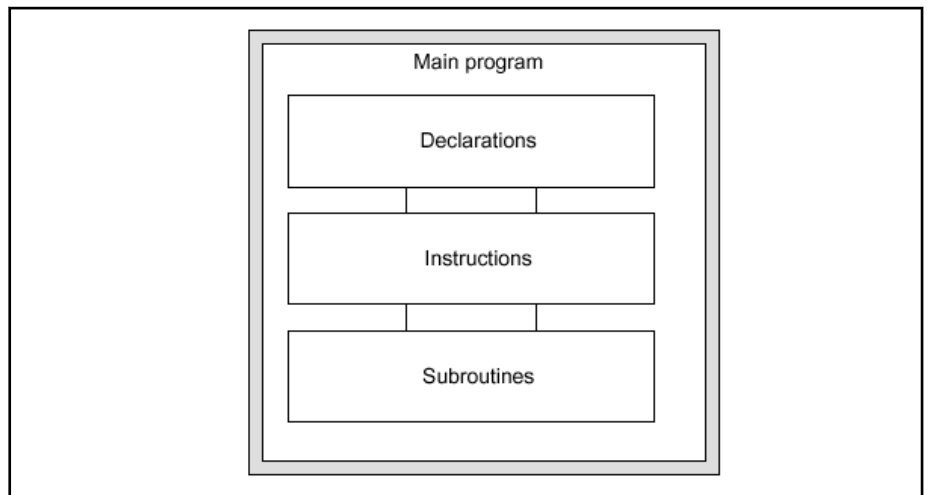


Abb.5-1: Struktur des Hauptprogramms

5.4.2 Deklarationsteil

Am Anfang des Hauptprogramms steht der Deklarationsteil. Im Deklarationsteil werden die im Hauptprogramm vorkommenden Namen vereinbart.

Dies betrifft:

- Programmkopf,
der Name des Hauptprogramms
- Externvereinbarung,
die Namen der im Hauptprogramm aufgerufenen externen Hauptprogramme
- Kanalvereinbarung,
die Namen der im Programm benutzten Ein- und Ausgangskanäle
- Variablen- und Konstantenvereinbarung,
die Namen der im Programm vorkommenden Variablen und Konstanten

RCL-Programmstruktur

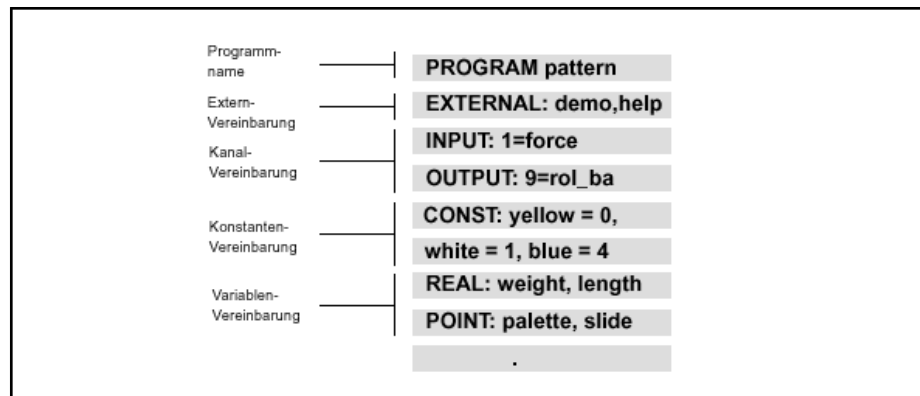


Abb.5-2: Deklarationsteil des Hauptprogramms

Die Deklaration von Variablen kann optional durch das Schlüsselwort "VAR" eingeleitet werden.

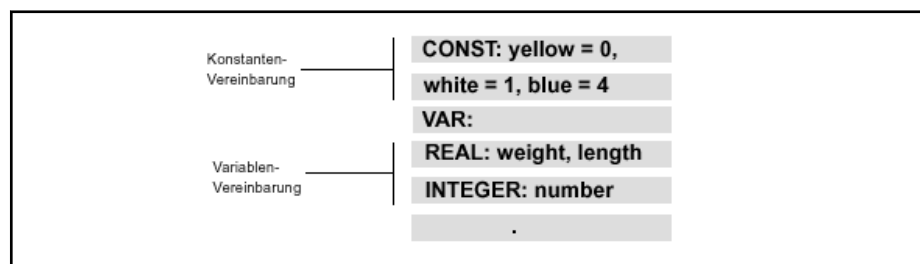


Abb.5-3: Deklaration von Variablen

Der Deklarationsteil ist vom Anweisungsteil durch das Schlüsselwort "BEGIN" zu trennen.



Nicht vereinbarte Variable werden als Variablen vom Typ "POINT" oder "JC_POINT" angenommen.

5.4.3 Anweisungsteil

Im Anweisungsteil (Statements) werden die auszuführenden Anweisungen programmiert.

Dies sind zum Beispiel

- Bewegungsanweisungen,
- Verzögerungen und Halt,
- Hauptprogrammaufrufe,
- Unterprogrammaufrufe,
- Programmteiwiederholungen,
- Programmsprünge,
- arithmetische Operationen , sowie
- Funktionsaufrufe.

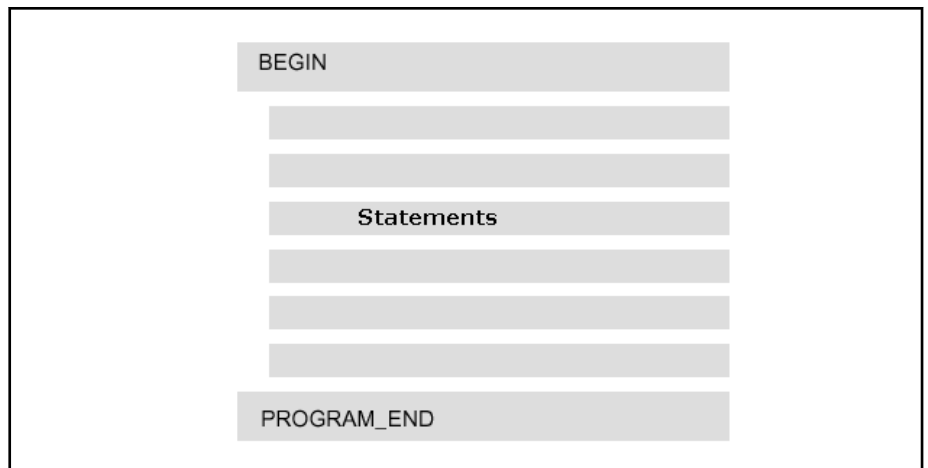


Abb.5-4: Anweisungsteil

Der Anweisungsteil steht zwischen den Schlüsselworten "BEGIN" und "PROGRAM_END".

5.5 Programmvereinbarung

Ein Hauptprogramm ist am Anfang durch das RCL-Wort "PROGRAM" und seinen Programmnamen gekennzeichnet.

Der Programmname besteht aus maximal acht Zeichen. Zulässig sind Buchstaben, Ziffern und Unterstrich.



Das erste Zeichen muss ein Buchstabe sein. Groß- und Kleinschreibung sind dabei gleichwertig

Den Abschluss des Programms bildet das RCL-Wort "PROGRAM_ END" oder "SUB_END", falls Unterprogramme aufgeführt sind.



Programmname und Dateiname müssen identisch sein

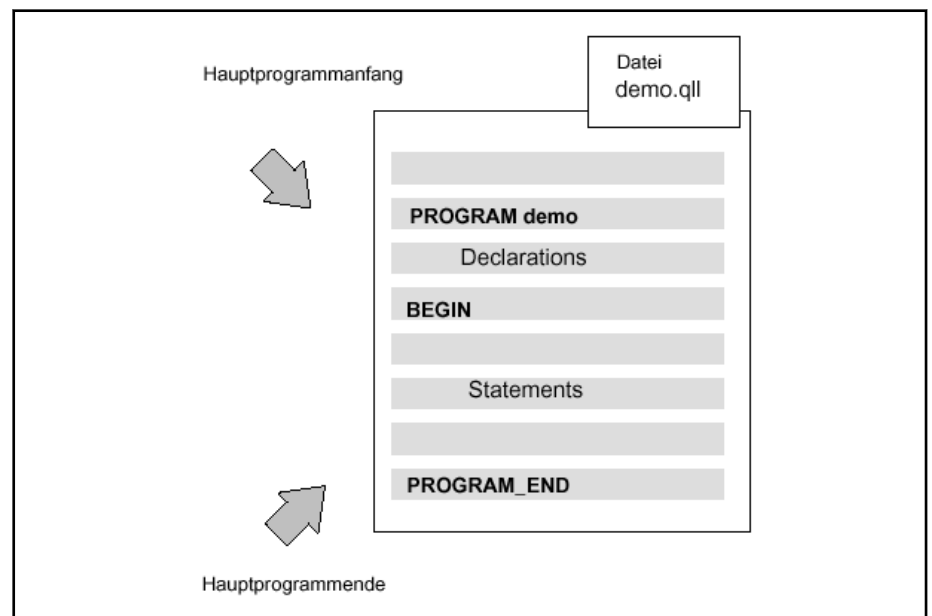


Abb.5-5: Aufbau des Hauptprogramms

5.6 Hauptprogrammaufruf im Hauptprogramm

Externvereinbarung:

Ein Hauptprogramm kann aus mehreren externen Einzel-Hauptprogrammen bestehen. Dabei ist jedes Hauptprogramm eine separate QLL-Datei.

Die externen Hauptprogramme müssen zu Beginn des Deklarationsteils mit "EXTERNAL" vereinbart werden und bei Aufruf als ird-Datei in der Steuerung vorhanden sein.

Externe Hauptprogramme können sowohl mit, als auch ohne Übergabeparameter vereinbart werden. Bei Parameterübergabe müssen die Anzahl, die Reihenfolge sowie die Datentypen mit der Vereinbarung übereinstimmen.

Die Übergabe eines Parameters kann als Wert oder als Referenz erfolgen. Bei Übergabe eines Parameters als Wert muss das Schlüsselwort "VALUE" verwendet werden. Bei dieser Art der Parameterübergabe wird nur der Wert übergeben. Bei Parameterübergabe als Referenz wird nicht der Wert, sondern eine Referenz auf den Wert übergeben. Auf diese Art ist es möglich, übergebene Parameter in einem externen Programm zu ändern, und im aufrufenden Programm diese Änderung zu verarbeiten. Als Parameter sind folgende Datentypen zulässig.

Typen für Parameter externer Programme

RCL Definition	Übergabetyp
VALUE CHAR	Zeichen als Wert
VALUE REAL	Gleitkommazahl als Wert
VALUE BINARY	Binärzahl als Wert
VALUE INTEGER	Ganzzahl als Wert
VALUE TEXT	Zeichenkette als Wert
VALUE POINT	Raumkoordinaten als Wert
VALUE JC_POINT	Maschinenkoordinaten als Wert
CHAR	Zeichen als Referenz
REAL	Gleitkommazahl als Referenz
BINARY	Binärzahl als Referenz
INTEGER	Ganzzahl als Referenz
POINT	Raumkoordinaten als Referenz
JC_POINT	Maschinenkoordinaten als Referenz

Abb.5-6: Typen für Parameter externer Programme

Externe Hauptprogramme mit Übergabeparametern können nicht als selbstständige Programme gestartet werden, sondern nur durch einen Programmaufruf aus einem übergeordneten Hauptprogramm.

Für den Aufruf externer Programme im aktiven Hauptprogramm genügt dann die Angabe des vereinbarten Programmnamens.

RCL-Programmstruktur

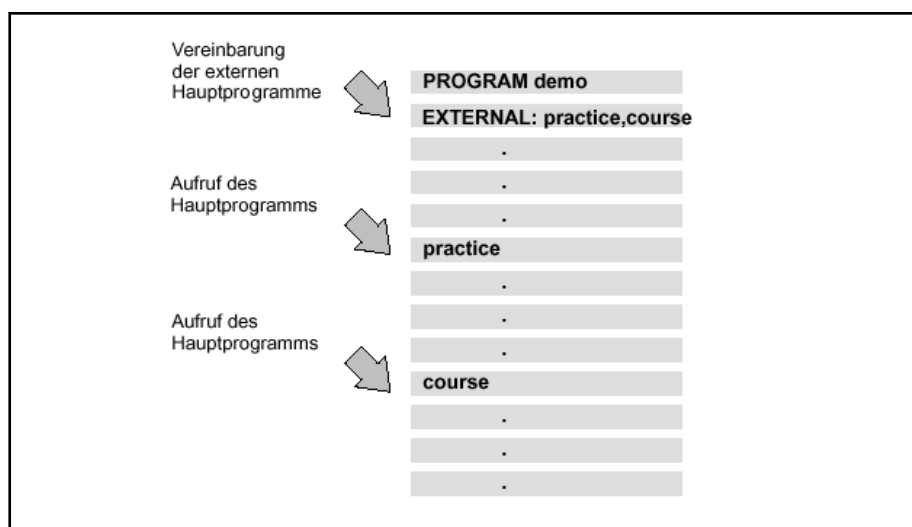


Abb.5-7: Arbeit mit mehreren Haupt-Programmen



Das Hauptprogramm und das aufzurufende externe Programm werden unabhängig voneinander übersetzt. Zum Übersetzungszeitpunkt ist daher keine Überprüfung der übergebenen Parameter auf Übereinstimmung mit der Vereinbarung im externen Hauptprogramm möglich. Sie wird beim Programmlauf vorgenommen. Die Anzahl, die Typen, die Reihenfolge sowie die Art (Wert oder Referenz) der Übergabeparameter muss der Vereinbarung des aufgerufenen externen Hauptprogrammes entsprechen.

Programmierung

Die Programmnamen der externen Hauptprogramme werden mit der Anweisung "EXTERNAL:" vereinbart.

Beispiel

```
EXTERNAL: practice,course
```

Externvereinbarung mit Parameterübergabe

```
EXTERNAL: withpar(VALUE INTEGER: number REAL: r2)
```

Es wird definiert, dass ein Hauptprogramm existiert, dass als ersten Parameter eine Ganzzahl als Referenz und als zweiten Parameter eine Gleitkommazahl als Wert übergeben wird. Das Hauptprogramm muss entsprechend vereinbart sein.

Hauptprogramm

```
PROGRAM withpar(VALUE INTEGER: number REAL: r2)
```

Hauptprogrammaufruf im Hauptprogramm

Die Steuerung arbeitet das aktive Hauptprogramm bis zum externen Programmaufruf ab.

Anschließend erfolgt ein Sprung zum Anfang des Programms "practice".

Das Programm "practice" wird bis zur "HALT"-Anweisung (oder zum "PROGRAM_END") abgearbeitet.

"HALT" (bzw. "PROGRAM_END") bewirkt einen Rücksprung in das Hauptprogramm "demo".

Die Steuerung setzt den Programmablauf mit der auf den Aufruf folgenden Anweisung fort.

RCL-Programmstruktur

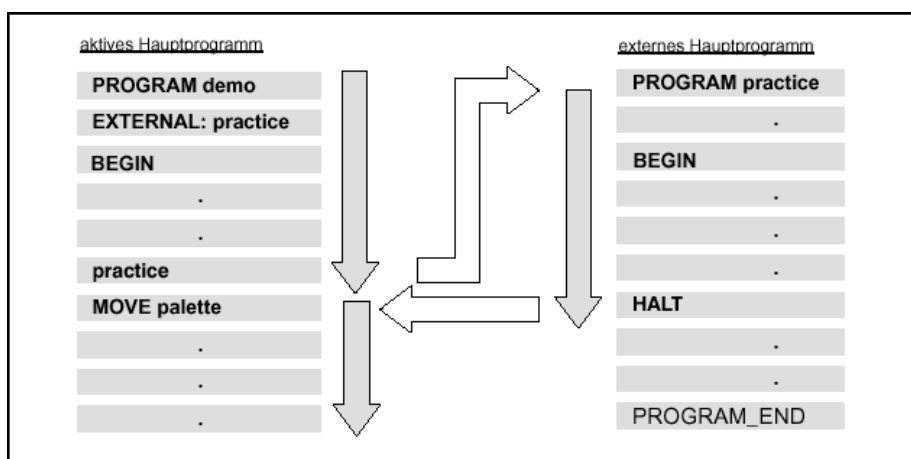


Abb.5-8: Ablauf eines Hauptprogrammaufrufs im Hauptprogramm

Auswirkungen von globalen Angaben

Bei einem externen Hauptprogrammaufruf werden die aktuellen globalen Angaben wie z.B. V, A, J oder die Slopeart mit übernommen. Änderungen dieser Werte im aufgerufenen Programm werden beim Rücksprung aus dem Programm aber nicht an das aufrufende Programm übergeben. Dazu ein Beispiel:

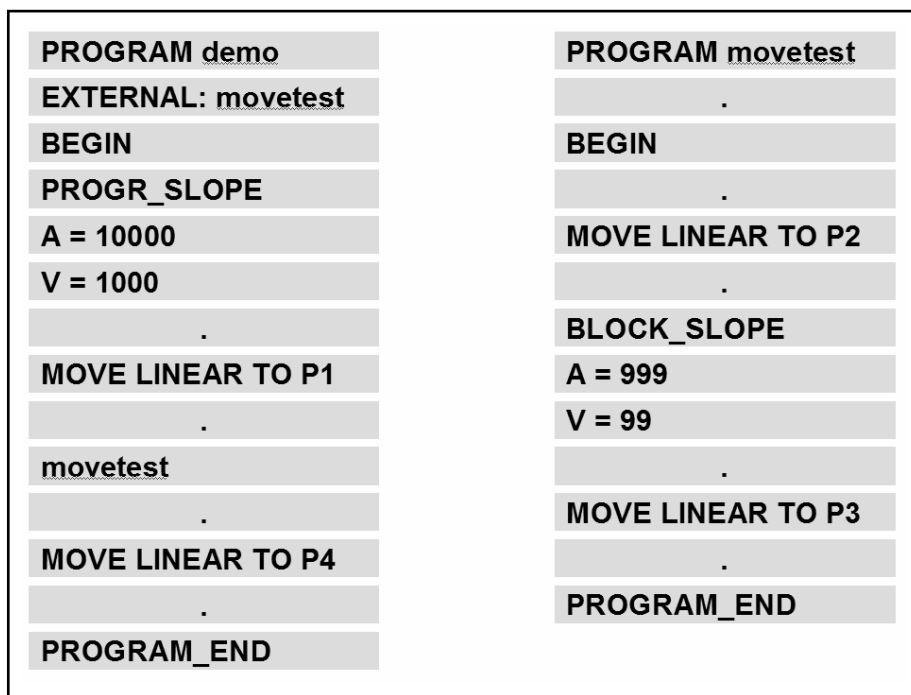


Abb.5-9: Beispiel externer Programmaufruf mit geänderten globalen Angaben

Folgende Tabelle zeigt mit welchen globalen Angaben die einzelnen Punkte angefahren werden.

Globale Angaben für die Bewegung

Punkt	A	V	Slopeart
P1	10000	1000	PROGR_SLOPE
P2	10000	1000	PROGR_SLOPE

P3	999	999	BLOCK_SLOPE
P4	10000	1000	PROGR_SLOPE

Abb.5-10: Globale Angaben für die Bewegung

5.7 Unterprogrammvereinbarung

Sind an verschiedenen Stellen eines Programms gleiche Arbeitsgänge durchzuführen, können Sie diese Arbeitsgänge zu Unterprogrammen zusammenfassen. Mit der Unterprogrammtechnik sparen Sie Speicherplatz und erhöhen zusätzlich die Übersichtlichkeit Ihres Programms.

Variablen, die im Hauptprogramm definiert sind, also globale Variablen, können auch im Unterprogramm verarbeitet werden.

Variablen, die im Unterprogramm deklariert sind, also lokale Variablen, können nur im Unterprogramm verarbeitet werden. Eine Übergabe in das Hauptprogramm erfolgt nicht.

Die Unterprogrammvereinbarungen liegen hinter dem Hauptprogramm nach der PROGRAM_END-Anweisung.

Kennzeichnung Ein Unterprogramm ist am Anfang durch das Wort "SUBROUTINE" und den Unterprogrammnamen gekennzeichnet.

Der Unterprogrammname besteht aus maximal zwölf Zeichen. Zulässig sind Buchstaben und Ziffern.



Das erste Zeichen muss ein Buchstabe sein. Groß- und Kleinschreibung sind dabei gleichwertig.

Programmierung

Den Schluss bildet das RCL-Schlüsselwort "SUB_END"

Ein Unterprogramm beinhaltet Anweisungen für den Greifer und soll den Namen "gripper" erhalten.

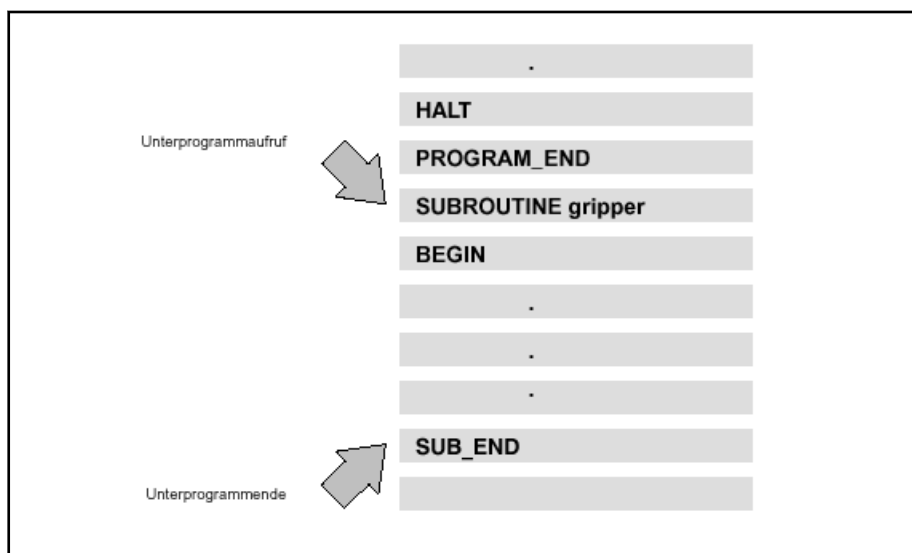


Abb.5-11: Platz des Unterprogramms nach dem Hauptprogramm



"RETURN" kann innerhalb eines Unterprogramms öfter verwendet werden, z. B. mit Programmsprüngen und bedingten Anweisungen.

RCL-Programmstruktur

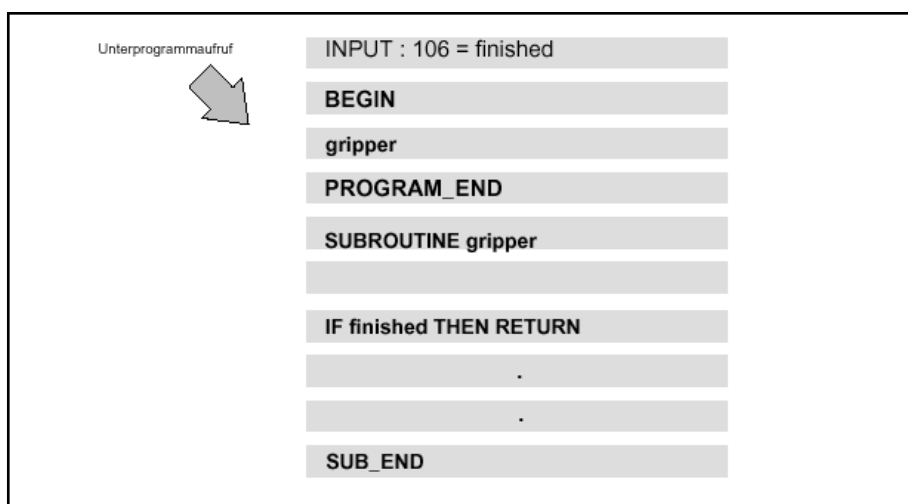


Abb.5-12: Unterprogrammaufruf und bedingter Rücksprung

Ist aus der Programmstruktur der Unterprogrammrücksprung erkennbar, z. B. am Unterprogrammende, generiert der Übersetzer den Befehl "RETURN" selbständig.

Unterprogrammaufruf

Für den Unterprogrammaufruf genügt die Angabe des vereinbarten Unterprogrammnamens, im Beispiel "gripper".



Das Programm und das aufzurufende Unterprogramm werden unabhängig voneinander übersetzt. Zum Übersetzungszeitpunkt erfolgt keine Überprüfung der übergebenen Parameter auf Übereinstimmung. Sie wird beim Programmaufbau vorgenommen. Die Anzahl, die Typen, die Reihenfolge sowie die Art ("VALUE" oder Adressierung) der Übergabeparameter muss der Vereinbarung des aufgerufenen externen Hauptprogrammes entsprechen.

5.8 Programmablauf

Die Steuerung arbeitet das Hauptprogramm bis zu einem Unterprogrammaufruf ab.

Anschließend erfolgt ein Sprung zum Anfang des aufgerufenen Unterprogramms. Das UP wird bis RETURN (oder SUB_END) abgearbeitet. Der Befehl "RETURN" bewirkt einen Rücksprung in das Hauptprogramm.

Die Steuerung setzt den Programmaufbau mit der auf den Unterprogrammaufruf folgenden Anweisung fort.

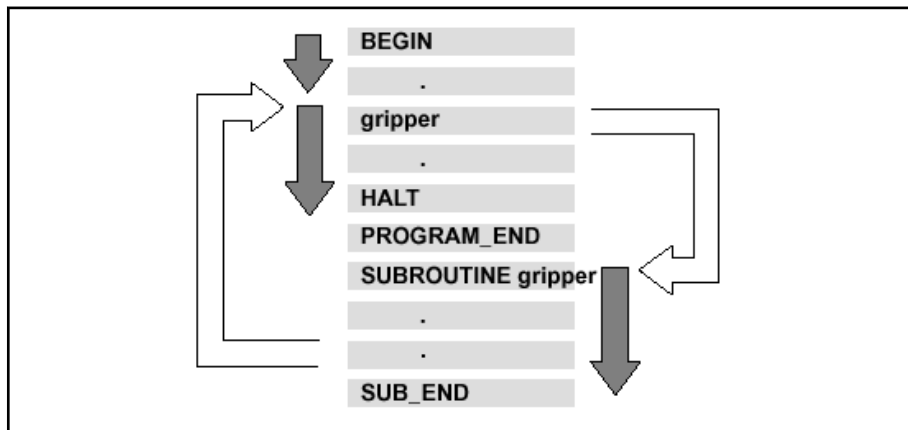


Abb.5-13: Programmabarbeitung Haupt-/ Unterprogramm

RCL-Programmstruktur

Mit einem Unterprogrammaufruf können Sie Variable oder Werte übergeben. Dazu müssen die Variablen in der Unterprogrammvereinbarung entsprechend deklariert werden.

Beispiel

```
gripper (1.5,6.0)
SUBROUTINE grip(VALUE REAL: force1,force2) ;Wertzuweisung
                                           ;beim Unterprogrammaufruf bekommen die
                                           ;Variable force1 den Wert 1.5 und die
                                           ;Variable force2 den Wert 6.0.
```

Erfolgt die Deklaration mit dem vorangestellten RCL-Befehl "VALUE", wird der Wert, andernfalls die Adresse der Variablen übergeben, d.h. dass die Variable im Hauptprogramm deklariert sein muss.

Ist "VALUE" in der Deklaration vorangestellt, übergibt das aufrufende Programm Informationen an das aufgerufene Unterprogramm. Das aufgerufene Unterprogramm liefert aber auf diesem Weg keine Informationen zurück.

Wird dagegen die Adresse übergeben, wirken sich Zuweisungen im Unterprogramm auf diese Variable auch nach Rückkehr in das aufrufende Programm aus.

Beispiel

```
;;INCLUDE head ;Übersetzeranweisungen wie Steuerungstyp, TESTINFO+ usw. aus
               ;der Datei "head" ins Programm einfügen
PROGRAM demo   ;Testprogramm für Unterprogramm mit Parameterübergabe
REAL: x        ;Variablenvereinbarung
BEGIN
  x=5
  anix(X)       ;Aufruf des UP anix
               ; x hat jetzt den Wert 10
  awas(X)       ;Aufruf des UP awas
               ; x hat immer noch den Wert 10
  HALT
PROGRAM_END
SUBROUTINE awas(REAL: aw) ;UPvereinbarung awas. Übergabe des Wertes und der
                        ;Variable. Das UP verrechnet die Variable und
                        ;gibt sie ins HP zurück.
BEGIN
  aw=aw*2       ; aw und x haben jetzt den Wert 10
  RETURN
SUB_END

SUBROUTINE anix(VALUE REAL: aw) ;UPvereinbarung anix. Übergabe des Wertes, die
                                ;Variable x im HP bleibt unverändert.
                                ;Das UP verarbeitet die Variable aw (eigener Speicherplatz).
                                ;Nach Verlassen des UP ist die Variable im HP unverändert.
BEGIN
  aw=aw*2       ;aw hat den Wert 20 und x hat immer noch den Wert 10
  RETURN
SUB_END
```

Schachtelung Innerhalb von aufgerufenen Haupt- oder Unterprogrammen (HP oder UP) können Sie weitere Hauptprogrammaufrufe bzw. Unterprogrammaufrufe programmieren. Man spricht in diesen Fällen von einer Schachtelung.

Programmbeispiele für Unterprogrammsschachtelung:

- Im Hauptprogramm ist ein Aufruf des UP "stacker" programmiert.
- Im Unterprogramm "stacker" ist ein weiterer Unterprogrammaufruf, das UP "gripper", programmiert.
- Die Steuerung arbeitet das Hauptprogramm bis zum Aufruf "stacker" ab.
- Dann erfolgt ein Sprung in das UP "stacker".
- Die Steuerung arbeitet das UP "stacker" bis zum Aufruf "gripper" ab.
- Dann erfolgt ein Sprung in das nächste UP "gripper".
- Das UP "gripper" wird im gezeigten Beispiel vollständig abgearbeitet.

RCL-Programmstruktur

Mit dem Befehl "RETURN" springt die Steuerung in das Unterprogramm "stacker" zurück, setzt den Programmlauf bis "RETURN" fort und springt schließlich wieder ins Hauptprogramm zurück.



Bei Schachtelung ist darauf zu achten, dass keine Endlos-Schleifen gebildet werden.

Die Schachtelungstiefe ist beliebig, sie ist nur durch die Größe des Stacks (4KByte) begrenzt.

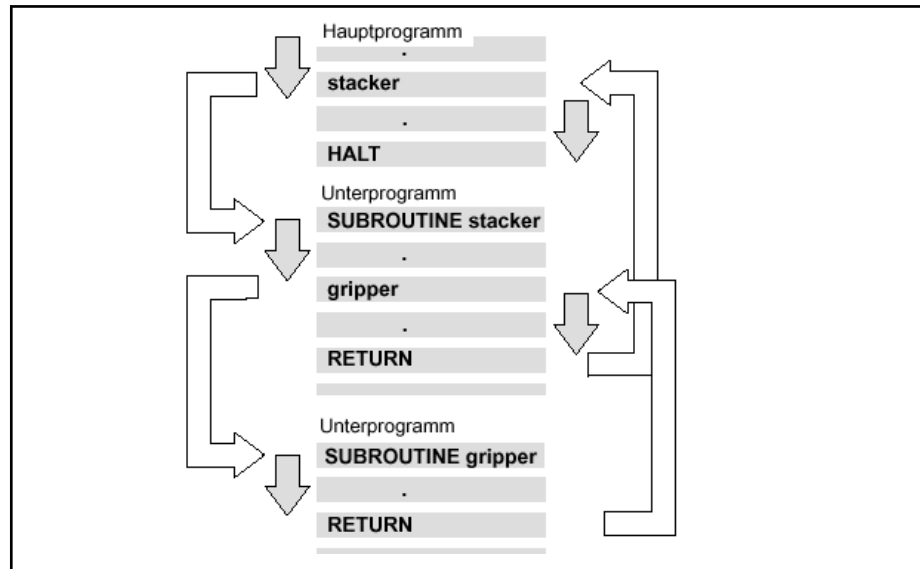


Abb.5-14: Beispiel 1

Auch Programmteilmiederholungen können geschachtelt werden. Innerhalb einer Programmteilmiederholung ist eine zweite Wiederholung programmiert.

Das Programm läuft wie folgt ab:

- Die Steuerung arbeitet den Programmteil bis zum Beginn des zweiten Programmteils einmal ab.
- Der zweite Programmteil wird dreimal wiederholt.
- Anschließend setzt die Steuerung den Programmlauf bis zum Ende des ersten Programmteils weiter fort; der erste Programmteil ist also einmal durchlaufen.
- Zum zweiten Durchlauf springt die Steuerung zum Anfang des ersten Programmteils zurück.
- Der ganze Ablauf wiederholt sich ein zweites Mal.

Die Schachtelungstiefe ist beliebig, sie ist nur durch die Größe des Stacks (4KByte) begrenzt.

RCL-Programmstruktur

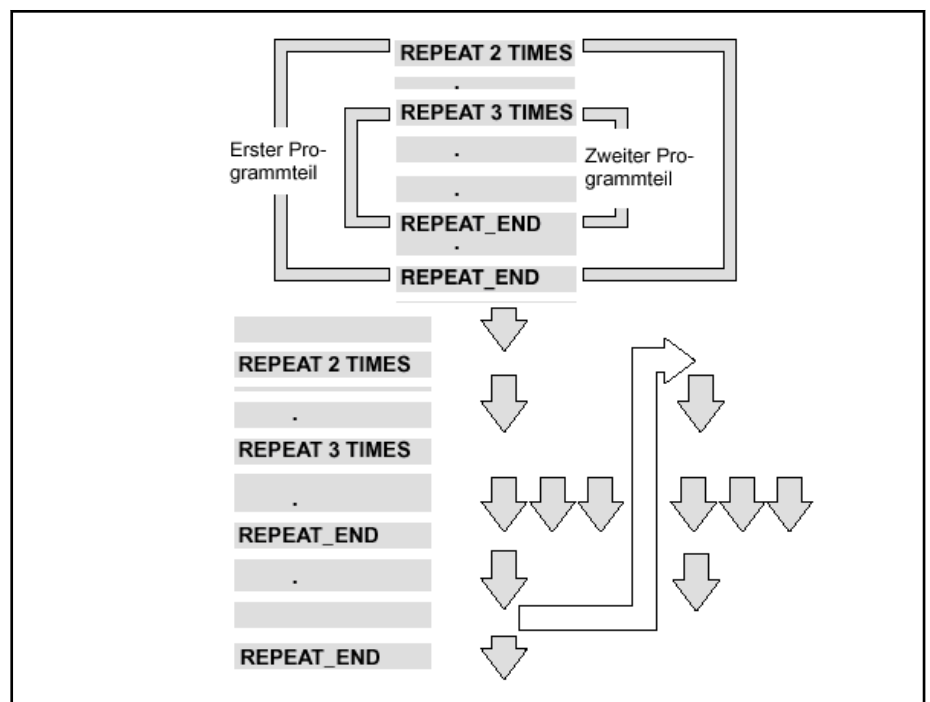


Abb.5-15: Beispiel 2

6 RCL-Konstanten

6.1 RCL-Konstantenvereinbarung

Innerhalb eines Programms können Zahlenwerte, z. B. die Anzahl der Wiederholungen, Zeichenfolgen oder Zeichen, durch Konstanten angegeben werden.

Der Wert einer Konstante kann auch durch eine Rechenvorschrift angegeben werden.

Die Konstanten sind mit der Konstantenvereinbarung vor den Variablen eines Programms festzulegen.

Beispiel

```
CONST: red=1,yellow=2*red,green=2*yellow, lines=4,      ;Constants
      columns=20, limit_value=2.5*7.25
INTEGER: color, l_index, c_index                        ;Variables
REAL: meas_value
ARRAY[lines .. columns] REAL: distance
BEGIN....
IF meas_value > limit_value THEN ....                  ;Implementation
```

6.2 Vordefinierte RCL-Konstanten

In RCL stehen Standardkonstanten zur Verfügung. Diese Konstanten sind im Sprachumfang enthalten und müssen deshalb nicht vereinbart werden.

WC_UR "WC_UR" ist eine in RCL vordefinierte Konstante des Typs WC_FRAME. Ihre sechs Komponenten haben alle den Wert Null.

Wird diese Konstante WC_SYSTEM zugewiesen, werden alle Verschiebungen und Verdrehungen gegenüber dem Ursprungskoordinatensystem zu Null; z.B. durch WC_SYSTEM = WC_UR.

Das Werkstück-Koordinatensystem ist damit wieder gleich dem ursprünglichen Raumkoordinatensystem (Siehe auch [Definition des Werkstück-Koordinatensystems relativ zum Welt-Koordinatensystem, Seite 23](#)).

7 RCL-Variablen

7.1 Allgemeines

In einem Programm können Zahlenwerte z.B. Koordinatenwerte bei Positionen oder die Anzahl der Wiederholungen durch Variablen ersetzt werden.

Eine Variable reserviert unter ihrem Namen einen Speicherplatz.

Diesem Speicherplatz kann beliebig oft ein Zahlenwert zugewiesen werden. Die Steuerung speichert den jeweils zuletzt zugewiesenen Zahlenwert ab.

Taucht während des Programmlaufes ein Variablenname auf, so ersetzt die Steuerung die Variable durch den zu diesem Zeitpunkt unter ihrem Namen abgespeicherten Wert.

Jede Variable hat einen Namen. Unterschiedliche Variablen müssen unterschiedliche Namen haben. Die Namen sollten möglichst so gewählt werden, dass man auch die Bedeutung der Variablen erkennen kann, z. B. bei Positionszeichnungen statt "b1" besser "bohrung_1" oder bei Rechenvariablen statt "i" besser "zaehler" usw.



Der Name besteht aus maximal 12 Zeichen.

Zulässig sind Buchstaben, Ziffern und der Unterstrich "_". Groß- und Kleinschreibung sind gleichwertig.

Das erste Zeichen muss ein Buchstabe sein.

Leerzeichen sind nicht erlaubt.

Eine Ausnahme bilden die Punkt-Variablen vom Typ "JC_POINT".

Diesen Variablen muss das "@" vorangestellt werden.

7.2 RCL-Datentypen

7.2.1 Übersicht

Der Datentyp einer Variablen bestimmt ihren Wertebereich und die zulässigen Zuweisungs- und Rechenoperationen. Die zulässigen Operationen finden Sie im Abschnitt [Vollständige Wertzuweisung, ab Seite 60](#).

Bei den Datentypen ist zwischen einfachen Datentypen und zusammengesetzten, strukturierten Datentypen zu unterscheiden.

Einfache Datentypen

- INTEGER
- REAL
- BINARY
- CHAR

Strukturierte Datentypen

- POINT
- JC_POINT
- BELT
- TEXT
- ARRAY
- WC_FRAME
- FILE.

RCL-Variablen

Ein zusammengesetzter Datentyp setzt sich aus zwei oder mehreren einfachen Datentypen zusammen.

7.2.2 Einfache RCL-Datentypen

Die Deklaration von Variablen erfolgt vor deren Verwendung im Deklarations-
teil (vor "BEGIN"). Eine Abgrenzung dieses Abschnittes gegenüber einer
Typdefinition kann durch das Schlüsselwort "VAR" erfolgen.

INTEGER Variablen vom Typ "INTEGER" dürfen nur ganzzahlige Werte, also Zahlen
ohne Dezimalpunkt, positiv oder negativ zugewiesen werden.

Wertebereich -2^{31} bis $+(2^{31}-1)^1$



Der RCL-Datentyp INTEGER entspricht dem SPS-Datentyp DINT!

REAL Variablen vom Typ "REAL" dürfen nur reelle Zahlen, also Zahlen mit Dezi-
malpunkt, positiv oder negativ zugewiesen werden.

Wertebereich: 4-BYTE-Gleitkommazahl mit einfacher Genauigkeit²⁾.



Die Angaben von Werten für Variablen vom Typ "REAL" muss mit
dem Dezimalpunkt erfolgen, nicht mit einem Komma.

BINARY Variablen vom Typ "BINARY" dürfen nur die Ziffern 0 oder 1 zugewiesen
werden.



Mit Variablen vom Typ "BINARY" dürfen keine ganzzahligen Re-
chenoperationen durchgeführt werden.

Die Ziffern 0 und 1 stellen keine Werte im herkömmlichen Sinn dar.

Sie beschreiben zwei definierte Zustände, z. B.:

Variablenbezeichnung	0 (FALSE)	1 (TRUE)
Schalter	aus	ein
Frage	nein	ja
Topf	leer	voll
gefüllt	falsch	richtig

Abb. 7-1: Interpretation binärer Werte

Variablen vom Typ "BINARY" sind z.B. Signale der binären Ein- und Aus-
gangskanäle.

¹⁾ Variablen vom Datentyp INTEGER (lange Ganzzahl) werden als 32-Bit-Zahlen (4 Bytes) mit Vorzeichen im Bereich von -2.147.483.648 bis 2.147.483.647 gespeichert.

²⁾ Variablen vom Datentyp REAL werden als 32-Bit-Zahlen (4 Bytes) -3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte gespeichert.

CHAR Variablen vom Typ CHAR dürfen nur ASCII-Zeichen³⁾ zugewiesen werden.

7.2.3 Zusammengesetzte RCL-Datentypen

POINT Variablen vom Typ "POINT" dürfen nur Positionen in Raumkoordinaten (World Coordinates) zugewiesen werden. Die einzelnen Koordinatenwerte einer Position vom Typ "POINT" müssen vom Typ "REAL" sein.

JC_POINT Variablen vom Typ "JC_POINT" dürfen nur Positionen in Maschinenkoordinaten (Joint Coordinates) zugewiesen werden.
Die einzelnen Koordinatenwerte einer Position vom Typ "JC_POINT" müssen vom Typ "REAL" sein.



Variablennamen vom Typ "JC_POINT" müssen mit dem Zeichen "@" beginnen.

Variablen vom Typ "POINT" und "JC_POINT" heißen auch Punktvariablen.
Die Anzahl der Komponenten richtet sich nach der Achsanzahl der angegebenen Kinematik.

TEXT Variablen vom Typ "TEXT" dürfen nur Texte, bestehend aus bis zu 80 ASCII-Zeichen, zugeordnet werden. Die einzelnen Zeichen können mit Index direkt wie Feldelemente angesprochen werden.

Text-Variablen, Deklaration und Zuweisung

```
TEXT: char_string           ;Declaration of text variables
char_string[1]='A'         ;Assigning individual components
char_string[2]='B'
char_string='This is ASCII text' ;Assigning a character string
```

FILE *Dateien, Deklaration*

FILE: cad_dat

Mit dem Datentyp "FILE" wird ein Dateinamen definiert. Dieser wird beim Zugriff mit "WRITE" oder "READ" bzw. "OPEN" oder "CLOSE" als Parameter verwendet.

7.2.4 Benutzerdefinierte RCL-Typen

Durch die Erweiterung des Vereinbarungsteils eines RCL-Programms um den Typdefinitionsteil "TYPE" ist es dem Anwender in RCL möglich, eigene Typen zu vereinbaren.

ALIAS-Name Damit können beliebigen Datentypen neue Namen zugewiesen werden.

Dies gilt sowohl für die Standardtypen

- INTEGER
- REAL
- BINARY
- CHAR

als auch für die daraus gebildeten strukturierten Typen.

³⁾ Der ASCII (American Standard Code for Information Interchange)-Zeichensatz (7-Bit) wird verwendet, um Buchstaben und Symbole darzustellen, die auf einer US-Standard-Tastatur verwendet werden. Der ASCII-Zeichensatz entspricht den ersten 128 Zeichen (0-127) des ANSI-Zeichensatzes.

RCL-Variablen

- TEXT
- ARRAY
- RECORD

Beispiele

```

PROGRAM types
TYPE:
  tmatrix=  ARRAY[0..7] ARRAY[0..7] INTEGER
  myinteger= INTEGER
;Type definition
;2 dim. ARRAY "tmatrix"
;Direct derivation from "INTEGER"

VAR:
  ARRAY [0..7] ARRAY[0..7] INTEGER: matrixone
  tmatrix: matrixtwo
  myinteger: integervar
;Variable declaration

BEGIN
  initold(matrixone)
  initnew(matrixtwo)
PROGRAM_END
SUBROUTINE initold(ARRAY[0..7] ARRAY[0..7] INTEGER: matrix) ;Long spelling
  BEGIN....
SUB_END
SUBROUTINE initnew(tmatrix: matrix) ;Short spelling
  BEGIN....
SUB_END

```

Verbundtypen, RECORDs

Eine weitere Verbesserung der Übersichtlichkeit und der Strukturierung von Programmen kann durch die Verwendung der Verbundtypen erreicht werden.

Ein Verbund hat einen Namen und besteht aus 1..n Variablen (Komponenten).

Jede Komponente hat einen eigenen Namen und Datentyp.

Eine Komponente kann selbst wieder ein Verbund sein.

In PASCAL sind diese Verbundtypen als "RECORD" bekannt, in C heißen sie "STRUCT".

Beispiel Barcode-Erkennung

Die Barcode-Information (r_barcode) besteht aus

- dem Namen (name, TEXT),
- dem Gewicht (weight, REAL) und
- dem Volumen (r_volume), selbst Verbund aus
 - Länge (length, REAL),
 - Breite (width, REAL) und
 - Höhe (height, REAL)

Verbunde sind nützlich, um komplizierte Daten zu organisieren, insbesondere in großen Programmen. Sie erlauben sowohl den Zugriff auf den kompletten Datensatz als auch auf jede Komponente.

Der Name, der bei der Typdeklaration auf der linken Seite des Zuweisungszeichens, vor dem reservierten Wort "RECORD" erscheint, steht für den gesamten Verbund und kann im folgenden als Abkürzung für die ausführliche Deklaration verwendet werden.

Variablen, die in einem Verbund erwähnt sind, werden Komponenten genannt.

RCL-Variablen

Eine Komponente kann den gleichen Namen besitzen, wie eine gewöhnliche Variable, ohne dass dadurch ein Konflikt entsteht. Sie können immer aus dem Kontext unterschieden werden.

Der Zugriff auf eine Komponente erfolgt ausschließlich über den Namen des Verbundes. Hierbei wird der Komponentename vom Verbundnamen durch einen Punkt getrennt.

Stellt die Komponente selbst einen Verbund dar, werden deren Komponenten ebenfalls durch einen Punkt getrennt.

Beispiel

```
PROGRAM records
TYPE:                                ;Type definition part

r_volume=RECORD
REAL: lenght
REAL: width
REAL: height
RECORD_END

r_barcode=RECORD
TEXT: name
REAL: weight
t_volume: volume
RECORD_END

r_barcode: code                      ;Variable declaration for the "code" of a packing
BEGIN
;
; code.name='cuboid'                  ;Accessing a component
; code.volume.lenght= 12.5            ;Accessing a subcomponent
; code.weight= 155.5
;
PROGRAM_END
```

Bei der Verwendung von Verbundvariablen innerhalb der WRITE- bzw. READ-Anweisung ist folgendes zu beachten:

- Verbundvariablen können auf Dateien vom Typ "FILE" (Text-Dateien) nur komponentenweise gelesen bzw. geschrieben werden.
- Verbundvariablen können auf Dateien vom Typ "BNR_FILE" (Binär-Datei) wahlweise komplett oder auch komponentenweise gelesen bzw. geschrieben werden.

ARRAY Sie haben die Möglichkeit, Variablen gleichen Typs zu einem Feld zusammenzufassen.

Diese Variablen haben alle denselben Namen und unterscheiden sich nur im Index. Deshalb heißen diese Variablen auch indizierte Variablen.

Syntax

```
ARRAY[( $\pm$  integer_constant)...( $\pm$  integer_constant)] <Type> : variablenname
```

Beispiel

```
ARRAY[-10..10] INTEGER: variablenname
```

7.3 RCL-Variablendeklaration

Die Steuerung muss vor Ausführung einer Anweisung mit Variablen wissen, welche Werte oder Zeichen die Variable annehmen darf.

RCL-Variablen

Deshalb muss jede im Programm verwendete Variable vorher vereinbart werden, d.h. es muss festgelegt werden, von welchem Datentyp die Variable ist.

Syntax

[DEF] Type: [channel =]name{ , [channel =]name}

- "DEF" kann vor Punktvariablen "POINT" und "JC_POINT" gestellt werden,
- "Type" ist durch den konkreten Typ, z.B. "INTEGER", zu ersetzen.
- "channel" ist nur bei den Typen "INPUT", "OUTPUT" oder "BELT" vorgesehen.



Alle Punkte, die mit "DEF" vereinbart werden und alle Punkte, denen an keiner Stelle im Programm ein Wert zugewiesen wird, werden in einer Punktedatei mit der Kennung "pkt" abgespeichert.

Punkte, die als Parameter an Unterprogramme, (Spz-Fkt und rho-Fkt) übergeben werden, erscheinen nicht in der Punktedatei.

Programmierung

Die Deklaration besteht aus dem Datentyp und den zuzuordnenden Variablennamen. Der Datentyp ist vom Variablennamen durch einen Doppelpunkt getrennt. Mehrere Variablennamen des gleichen Typs werden durch Komma getrennt.

Deklarationen sind möglich zwischen:

- "PROGRAM" und "BEGIN" und
- "SUBROUTINE" und "BEGIN".

Beispiel

```
PROGRAM Prog2
POINT: BasePnt
OUTPUT: 1=valve_1,2=valve_2
BINARY: ready
INTEGER: counter
REAL: divisor,xvalue,yvalue
TEXT: message_1,message_2
BEGIN
```



Variablen vom Typ "POINT" und "JC_POINT" müssen nicht deklariert werden, d.h. der Übersetzer interpretiert nicht deklarierte Variablen als Punktvariablen.



"JC_POINT"-Variablen beginnen mit dem Zeichen "@".



[RCL-Schlüsselworte \(siehe Seite 281\)](#) dürfen nicht als Namen für Anwendervariablen benutzt werden.

7.4 Punktvariablen

7.4.1 Vereinbarung von Punktvariablen

Punktvariablen sind zusammengesetzte, strukturierte Datentypen und bestehen aus Komponenten.

Die Komponenten sind die Koordinaten oder die Achsen der Punktvariablen.

Neben der vollständigen Wertzuweisung können Sie den Punktvariablen auch komponentenweise Werte zuweisen, z. B. `ecke.z_k = hoehe`.

Die Komponentenbezeichnung geben Sie mit den Übersetzeranweisungen

- `;;JC_NAMES = achsname, ...`
sowie
- `;;WC_NAMES = koordinatenname, ...`

an.

Beispiel

```
;;KINEMATICS: ( 1 = Kin_1)
;;KINEMATICS= Kin_1      ; Default Kinematics here
;;WC_NAMES= kx, ky, kz
;;JC_NAMES= a1, a2, a3
```

```
PROGRAM Prog2
  POINT: corner
  JC_POINT: @wpos
BEGIN
  corner.kx = 100.0
  corner.ky = 200.0
  corner.kz = 300.0
  @wpos.a1 = 45.3
  @wpos.a2 = 0.0
  @wpos.a3 = 10.2

  MOVE TO @wpos
  MOVE TO corner
PROGRAM_END
```



Namen von Punktvariablen des Typs "POINT" beginnen mit einem Buchstaben.

Namen von Punktvariablen des Typs "JC_POINT" beginnen mit dem Sonderzeichen "@".

Wertzuweisung für Punktvariablen erfolgt erst nach "BEGIN".

7.4.2 Punkte und Punktedatei "pkt"

Während alle anderen Variablen vereinbart werden müssen, brauchen Punktvariablen nicht ausdrücklich vereinbart werden.



Der RCL-Übersetzer interpretiert alle nicht vereinbarten Variablen als POINT oder JC_POINT.

Alle Punkte, die mit "DEF" vereinbart werden und alle Punkte, denen an keiner Stelle im Programm ein Wert zugewiesen wird, werden in einer Punktedatei mit der Kennung "pkt" abgespeichert. Details im Multi-Kinematik-Betrieb siehe [Punktetabellen der Kinematiken, Seite 226](#).

Punkte, die als Parameter an Unterprogramme, (Spz-Fkt.) übergeben werden, erscheinen nicht in der Punktedatei.

RCL-Variablen

Dies gilt auch für die Übergabe von einzelnen Punktkomponenten (z.B. "Pos.X_K").



Punkten in der pkt-Datei kann man nur dann während des Programmablaufes Werte zuweisen, wenn sie mit "DEF" vereinbart wurden.



Punkte, die nicht vereinbart sind, denen aber im Programm Werte zugewiesen sind, werden nicht in der Punktedatei sondern in der ird-Datei abgelegt.

7.4.3 Vollständige Wertzuweisung

Bei vollständigen Wertzuweisungen werden allen Koordinaten der Punktvariablen Werte zugewiesen. Innerhalb einer Zuweisung dürfen nur Punktvariablen eines Datentyps stehen.

Ausnahme:

Innerhalb einer Zuweisung dürfen auch gemischte Operationen mit den Standard-Funktionen "WC" bzw. "JC" stehen.

Zuweisung von Zahlenwerten

```
position=(50,0,100,0,15,10)
@edge=@(45,5.8,70,10,5.8,0)
```

7.4.4 Zuweisung von Variablen für Einzelkomponenten

Zuweisung bei Einzelkomponenten

```
hole=(xvalue,yvalue,zvalue,0,0,0)
@seat=@(50,95.8,height,40,38,0)
hole.z_c=height
```

Die Variablen "xvalue", "yvalue", "zvalue" und "height" sind vom Typ "REAL".

Zuweisung durch Addition und Subtraktion

Der Punktvariablen "shelf" ist die Summe aus der Punktvariablen "position" und der Konstanten zugewiesen worden.

Konstanten bestehen aus der in Klammern gesetzten Aufzählung ihrer Koordinaten. Konstanten sind im Gegensatz zur Variablen während des Programmablaufes unveränderlich.

Bei Addition und Subtraktion addieren bzw. subtrahieren sich jeweils die Einzelkomponenten.

Zuweisung Basis $\pm$ Offset

```
shelf= position + (10,-30,100,5,0,0,0)
```

Zuweisung mit Multiplikation und Division

Durch die Multiplikation und Division mit Zahlenwerten oder Variablen des Typs "REAL" können Sie alle Koordinatenwerte von Punktvariablen bzw. Konstanten verändern.

Zuweisung mittels Ausdruck (1)

```
pos_1=(10,10,10,0,0,0)
pos_2=pos_1*2
```

Jede Einzelkomponente der pos_1 verdoppelt sich durch die Multiplikation mit 2 und wird der neuen Punktvariablen pos_2 zugewiesen; pos_2 hat dann die Koordinaten: pos_2 = (20, 20, 20, 0, 0, 0)

Zuweisung mittels Ausdruck (2)

```
div=4.0  
pos_3=pos_2/div
```

pos_3 hat dann die Koordinaten: pos_3 = (5, 5, 5, 0, 0, 0).

Gemischte Operation mit Punktvariablen

Über die Standardfunktionen "WC" und "JC" können Sie gemischte Operationen mit Punktvariablen des Typs "POINT" und "JC_POINT" ausführen.

Soll das Ergebnis der Rechenoperationen einer Punktvariablen des Typs "POINT" zugewiesen werden, ist die Rechnung in Raumkoordinaten (world coordinates) durchzuführen.

Soll das Ergebnis der Rechenoperationen einer Punktvariablen des Typs "JC_POINT" zugewiesen werden, ist die Rechnung in Maschinenkoordinaten (joint coordinates) durchzuführen.

Gemischte Operationen

```
PROGRAM Prog2  
  DEF POINT: p2  
  DEF JC_POINT: @p1, @p3  
  BEGIN  
    @p3=@(0,100,0)  
    @p1= JC(p2)+@p3  
  PROGRAM_END
```

Die Steuerung rechnet die Raumkoordinaten des Punktes "p2" in Maschinenkoordinaten um und addiert sie zu den Koordinaten von @p3.

Das Ergebnis wird der Punktvariablen "@p1" zugewiesen.

Lesen der Ist-Position POS, @POS

Mit den Standardpunktvariablen POS (Raumkoordinaten, World Coordinates) und @POS (Maschinenkoordinaten, Joint Coordinates) können sie während des Programmlaufs Punktvariablen die jeweils aktuelle Position des Roboters zuweisen.

Lesen der Ist-Position

```
act_pos= POS  
@joint_pos= @POS
```

Die Standardpunktvariablen stehen auf der rechten Seite der Zuweisung. Nach Komponenten getrennte Zuweisung ist ebenfalls möglich.

Lesen der Ist-Position, komponentenweise

```
cpos.c_3= POS.c_3  
xvalue= POS.c_1
```



Eine Abfrage ist allerdings nur bei Stillstand des Roboters möglich.

Wird die Abfrage im Anschluss an einen "MOVE..TO"-Befehl angewandt, erhält man durch die gesamte Wirkkette in der Regel einen geschwindigkeitsabhängigen Positionsfehler.

Komponentenweise Zuweisung

Bei Punktvariablen können Sie einzelnen Koordinaten neue Werte zuweisen. Umgekehrt ist es möglich, Variablen vom Typ "REAL" Koordinatenwerte von Punktvariablen zuzuweisen.

RCL-Variablen

Die Koordinatennamen sind in den Maschinenparametern oder durch Übersetzeranweisung festgelegt. In den folgenden Beispielen sind die Koordinaten einer Position in Raumkoordinaten mit k1, k2, k3, usw. (k = Koordinaten), die Koordinaten einer Position in Maschinenkoordinaten mit a1, a2, a3, usw. (a = Achse) bezeichnet.

Die Koordinatenbezeichnung wird an den Namen der Punktvariablen durch einen Punkt angehängt.

Beispiel

```
position.k3= 100      ;Der Koordinate k3 wird der Wert 100 zugewiesen
zvalue= anchor.k3    ;Der Variablen zvalue ist der dritte Koordinatenwert der
                    ;Punktvariablen anchor zugewiesen
@pal_pos.a4= radius  ;Der Koordinate a4, der in Maschinenkoordinaten beschriebenen
                    ;Position, ist der Wert der Variable radius vom Typ REAL zugewiesen
```

7.5 Arbeit mit Textvariablen

Innerhalb eines Programms können Sie Textvariablen Texte zuweisen.

'message' und 'note' als TEXT-Variable deklarieren

```
TEXT: message, note
```

Textzuweisung

Der zuzuweisende Text muss in Hochkommata stehen und darf maximal 80 Zeichen haben. Der Text muss innerhalb einer Zeile stehen.

TEXT-Variable erhält Zeichenkette zugewiesen

```
message= 'gripper is defective'
note= 'change pallet'
```

Variablenverwendung

Textvariablen können im Programm mit den RCL-Befehlen WRITE auf einen Ausgabekanal, z. B. Datei ausgegeben und mit READ eingelesen werden.

READ/WRITE auf Ein-/Ausgabe-Kanal

```
WRITE File1, note
READ File1, entry
```



Die Variable selbst darf nicht in Hochkommata stehen.

Die Steuerung gibt bei **WRITE 'note'** das Wort "note" aus.

7.6 Arbeit mit Feldvariablen (ARRAYs)

Variablen vom gleichen Typ können Sie zu Feldern zusammenfassen.

Felder bestehen aus einer frei wählbaren Anzahl von Feldplätzen, die mit aufsteigenden Nummern bezeichnet sind. Jedem Feldplatz kann über die Angabe einer Nummer (Index) eine Variable zugeordnet werden. Variablen eines Feldes haben alle denselben Namen und unterscheiden sich nur im Index. Der Index stimmt mit der Nummer des zugewiesenen Feldplatzes überein.

Alle Typen von Variablen können in Feldern abgelegt werden.

Feldvereinbarung

Die Feldvereinbarung besteht aus:

- Vereinbarungsbefehl "ARRAY"
- Feldgrenzen (Platznummern)
- Vereinbarung der Feldvariablen.

RCL-Variablen

Die Feldgrenzen werden in eckigen Klammern angegeben und aus der ersten Platznummer (erster Index = Untergrenze) und der letzten Platznummer (letzter Index = Obergrenze) gebildet.

Deklaration von Array-Instanzen

```
ARRAY[1..9] POINT: placing_pos           ;1-dim. ARRAY
ARRAY[0..10] TEXT: message
ARRAY[-10..10] INTEGER: number_pos
ARRAY[1..9]ARRAY[5..10]POINT: picking_pos ;multidim. ARRAY
```

Die Untergrenze ist von der Obergrenze durch zwei Punkte zu trennen. z.B. [3..8].

Die Obergrenze des Feldes darf nicht kleiner als die Untergrenze sein. Der Index ist vom Typ "INTEGER".

Beispiel

```
ARRAY[1..5] POINT: hole           ;Vereinbarung eines Feldes mit 5 Plätzen für die
                                   ;Punktvariable 'hole', der erste Index soll 1 sein
```

Wertzuweisung

Jede Feldvariable wird durch ihren Namen und den in eckigen Klammern gesetzten Index angesprochen. Für den Index können Sie auch eine Variable oder einen Ausdruck vom Typ "INTEGER" einsetzen. Dann ist eine Wertzuweisung über die "REPEAT"-Anweisung oder über die anderen Programmschleifen möglich.

Wertzuweisung bei Feld-Elementen

```
PROGRAM Prog2
ARRAY[-5..0] INTEGER: number
INTEGER: i,k           ;Index i und Wert k
BEGIN
;Beispiel mit direkter Zuweisung
number[-5]=1           ;Die Variable number auf dem Feldplatz -5 hat den Wert 1,
number[-4]=2           ;die Variable number auf dem Feldplatz -4 hat den Wert 2,
number[-3]=4           ;usw.
number[-2]=8
number[-1]=16
number[0]=32

;Beispiel mit REPEAT-Anweisung
i=-5                   ;Erster Index
k=1                    ;Erster Wert
REPEAT 6 TIMES
number[i]=k            ;Wertzuweisung
i=i+1                 ;erhöhe Index
k=k*2                 ;ändere Wert
                     ;beliebige weitere Zuweisungen
REPEAT_END

;Beispiel Programmschleife
i=-5                   ;erster Index
k=1                    ;erster Wert
mark:                  ;Sprungziel
number[i]=k            ;Wertzuweisung
i=i+1                 ;Erhöhung Index
k=k*2                 ;Änderung des Wertes
                     ;beliebige Zuweisungen
IF i<=0 THEN JUMP mark ;Bedingte Zuweisung für Programmschleife
PROGRAM_END
```

Bestimmung von Palettenpositionen

```
;;INCLUDE head           ;Einfügen von Übersetzeranweisungen
PROGRAM palpos           ;Bestimmung von Palettenpositionen
INTEGER: line,column,c,l,k ;Deklarationen
ARRAY[1..12] POINT: palpos ;Die Palette hat vier Zeilen und drei Spalten,
                             ;also 12 Positionen
BEGIN                   ;Programmstart
dx=(30,0,0,0,0)         ;Der Abstand der Positionen zueinander ist
```

RCL-Variablen

```

dy=(0,20,0,0,0)           ;unter dx und dy im Inkrementalmaß angegeben
line=4
column=3
c=0
l=0
k=0
mark_1:
k=k+1
palpos[k]=pos+c*dx+l*dy    ;Anweisungen, Punktzuweisungen
                             ;Die Position pos ist ein Teachpunkt, d.h.
                             ;sie wird durch Anfahren und Abspeichern definiert,
                             ;Sie ist die erste Position palpos[1].
                             ;Deshalb stehen im Initialisierungssteil s=0 und z=0.
                             ;Erhöhe Spaltenzahl
c=c+1
IF c<column THEN JUMP mark_1 ;die nächste Position palpos[2] befindet sich in
ELSE c=0                    ;der benachbarten Spalte.
                             ;Die Spaltenzahl muss also erhöht werden.
                             ;Gleichzeitig darf die Spaltenzahl c nicht
                             ;die Gesamtanzahl der Spalten überschreiten.
                             ;Bleibt c kleiner als column, springt die
                             ;Steuerung auf die Sprungmarke mark_1,
                             ;erhöht den Index k um 1
                             ;und weist der Position palpos[2] den
                             ;Wert pos+1*dx+0*dy zu. Wird c größer als
                             ;column, weist die Steuerung der Variablen c
                             ;den Wert Null zu und erhöht die Zeilenzahl.
                             ;erhöhe Zeilennummer
l=l+1
IF l<line THEN JUMP mark_1  ;Die Erhöhung der Zeilenzahl läuft analog
                             ;zur Erhöhung der Spaltenzahl ab. In der
                             ;IF-THEN-Anweisung ohne ELSE-Anweisung
                             ;die Steuerung fährt bei Nichterfüllung der
                             ;Bedingung l<line mit der Fahrmanweisung fort

;;INT= LINEAR
V=1000
FACTOR=9.999
k=0

REPEAT 12 TIMES
k=k+1
MOVE TO palpos[k]           ;Fahrmanweisung
MOVE_REL with V=36 EXACT (0,0,-20,0,0)
WAIT 2
MOVE_REL with V=59 EXACT (0,0,20,0,0)
REPEAT_END
MOVE_REL CIRCULAR((-50,-50,100,0,0),(-100,-100,0,0,0))
HALT
PROGRAM_END

```

7.7 Kommunikationskanäle zur SPS

7.7.1 Kommunikationskanäle zur SPS, Allgemeines

Mit der RCL ist es möglich Ein- und Ausgänge zur SPS hin zu beschreiben oder von dieser zurück zu lesen:

- BINARY** Die Schnittstelle verfügt über 32 binäre Ein- und Ausgänge, die erweiterte Schnittstelle verfügt über 32 + 160 = 192 binäre Ein- und Ausgänge von und zur SPS.
- INTEGER** Die Schnittstelle verfügt über 4 Ein- und Ausgänge vom Typ "INTEGER", 32 Bit
 die erweiterte Schnittstelle verfügt über 4 + 46 = 50 Ein- und Ausgänge vom Typ "INTEGER" von und zur SPS.
- REAL** Die Schnittstelle verfügt über 4 Ein- und Ausgänge vom Typ "REAL",
 die erweiterte Schnittstelle verfügt über 4 + 46 = 50 Ein- und Ausgänge vom Typ "REAL" von und zur SPS.

RCL-Variablen

- POINT** Punkte können von der SPS zum RCL-Programm übertragen werden, jedoch nicht vom RCL-Programm zur SPS.
 Beispiel
 INPUT POINT: 701 = point1
 OUTPUT POINT... ist nicht verfügbar!
 Die Schnittstelle verfügt über 4 Eingänge vom Typ "POINT", die erweiterte Schnittstelle verfügt über 4 + 46 = 50 Eingänge vom Typ POINT.
- BELT** Bandkanäle dienen der Synchronisation auf eine beliebige Achse, d.h. reale Achse, virtuelle Achse, Verbundachse oder Geberachse (Standardwegmesssystem).
 Bandkanäle sind Eingänge vom Typ "INTEGER", keine erweiterte Schnittstelle.



Datenaustausch mit der SPS

- Eingänge auf der RCL-Seite sind Ausgänge auf der SPS-Seite und umgekehrt.
- Auf Daten der Schnittstelle kann SPS-seitig sowohl über die KinData-Schnittstelle als auch per Funktion zugegriffen werden.
 Auf Daten der erweiterten Schnittstelle kann SPS-seitig nur per Funktion zugegriffen werden.
- Der Funktions-Zugriff auf die User-Daten ist nicht abhängig von der Satzvorbereitung des RCL-Programms, sondern wird direkt ausgeführt.

ML_KINEMATICDATA_SM	FUNCTION	INDEX
dwUserBits_i	ML_KinGetUserBits	1..32
	ML_KinGetUserBits	33..192
dwUserBits_i	ML_KinGetUserBit_DWord	1
	ML_KinGetUserBit_DWord	2..6
adiUserDInt_i[1..4]	ML_KinGetUserDInt	1..4
	ML_KinGetUserDInt	5..50
arUserReals_i[1..4]	ML_KinGetUserReal	1..4
	ML_KinGetUserReal	5..50
dwUserBits_q	ML_KinSetUserBits	1..32
	ML_KinSetUserBits	33..192
adiUserDInt_q[1..4]	ML_KinSetUserDInt	1..4
	ML_KinSetUserDInt	5..50

RCL-Variablen

ML_KINEMATICDATA_SM	FUNCTION	INDEX
arUserReals_q[1..4]	ML_KinSetUserReal	1..4
	ML_KinSetUserReal	5..50
arUserPoints_q[1..4]	ML_KinSetUserPoint	1..4
	ML_KinSetUserPoint	5..50

Abb.7-2: Erweiterung des Shared Memory-Bereiches durch Funktionszugriff aus Sicht der SPS

Die Adressierung des jeweiligen Ein- oder Ausgangs erfolgt durch Angabe einer Kanalnummer bei der Vereinbarung der Ein- oder Ausgangsvariablen.

Synchrone und asynchrone Abarbeitung

Im RCL-Programm kann über die INPUT / OUTPUT Variablen auf die User-Daten zugegriffen werden.

Es ist sowohl ein synchroner, als auch ein asynchroner Zugriff (Kanalnummer + 1000) möglich; synchron $\Leftrightarrow$ wirksam in Satzabarbeitung, asynchron $\Leftrightarrow$ wirksam in Satzvorbereitung.

Auf die User-Daten kann wie folgt zugegriffen werden:

- INPUT / OUTPUT BINARY,
- INPUT / OUTPUT REAL,
- INPUT / OUTPUT INTEGER ,
- INPUT POINT.

Beispiele für RCL-Zugriffe:

Zugriff auf 108. User-Bit, Output, asynchron:

```
OUTPUT BINARY:    1108 = bool1
```

```
bool1 = TRUE
```

Zugriff auf 41. User-Point, Input, synchron:

```
INPUT POINT:      741 = point1
```

```
varPoint = point1
```

Folgende Kanalnummern (bei asynchronem Zugriff +1000) stehen bei der MLC 10VRS pro Kinematik zur Verfügung:

Kanalnummer ⁴⁾	Typ und Bedeutung
1 bis 32, bis 192	BOOL, binäre Ein- und Ausgänge
201 bis 204, bis 250	REAL, Ein- und Ausgänge
401 bis 404, bis 450	INTEGER, Ein- und Ausgänge (auf der SPS-Seite DINT!)
501 bis 504	Band-Eingänge (keine Ausgänge)
701 bis 704, bis 750	Punkt-Eingänge (keine Ausgänge)

Abb.7-3: Kanalvereinbarungen

Band-Eingänge

Für den **RCL-Programmierer** werden die Band-Daten über die Definition

BELT: 501...504 zugänglich gemacht.

Aus der Sicht **Parameter** sind die Daten des "BELT: 501" im Parameter K-0-0008, Achskonfigurationsliste, erste Bandachse, u.s.w. zu finden.

⁴⁾ Dritte Zahl: erweiterte Schnittstelle.

Die Koordinaten des Tool Center Point (TCP) werden auf diese Weise um die Bandkoordinaten ergänzt.

Die **SPS** adressiert den gleichen Speicher (Shared Memory) über eine Instanz des Datentyps "ML_KINEMATICS_DATA_SM" (Details siehe Rexroth IndraMotion MLC10VRS, ML_Robot).

So kann die aktuelle Bandposition aus "KinData[n].arPositionOriginalCoord_i" entnommen werden (erst TCP-Koordinaten, dann Band-Koordinaten).

7.7.2 Kommunikationskanäle zur SPS, Vereinbarung und Programmierung

In der Kanalvereinbarung wird einer Kanalnummer der Datentyp "BINARY", "INTEGER" oder "REAL" und der Variablenname des zu übertragenden Signals zugeordnet. Dabei ist festzulegen, ob es sich um Eingangs- oder Ausgangssignale handelt.

Die einzelnen Zuordnungen der Kanäle müssen durch ein Komma getrennt werden. Nach der letzten Zuordnung darf kein Komma stehen.



Fehlt die Angabe des Datentyps nach "INPUT" bzw. "OUTPUT", unterlegt die Steuerung den Typ "BINARY".

Kommunikationskanäle, Deklaration und Verwendung

```
PROGRAM Prog1
INPUT:
  1=gate_switch1,
  2=met_unit,
  5=li_barrier
OUTPUT:
  7=alarm
BEGIN
  IF NOT gate_switch1 AND li_barrier THEN alarm=1
PROGRAM_END
```

Bei Abfrage binärer Signale auf 1 kann die Angabe "=1" entfallen. Die Steuerung fragt dann selbstständig auf "1" ab.

Beispiel: Test auf "1" (TRUE)

```
IF gate_switch1 THEN ..
IF NOT gate_switch1 THEN ..
```

Beispiel: Signale vom Typ "INTEGER"

```
PROGRAM Prog2
INPUT:
  1=gate_switch1
INPUT INTEGER:
  401=grip_force,
  403=meas_height
OUTPUT INTEGER:
  401=pressure,
  402=print
BEGIN
  WAIT UNTIL gate_switch= 1
  IF grip_force>=26 THEN...

  WAIT UNTIL meas_height >= 212
  MOVE LINEAR UNTIL meas_height >= 200 TO pos ; ok
PROGRAM_END
```

Beispiel: Signale vom Typ "REAL"

```
PROGRAM Prog3
INPUT REAL:
  201=torque,
```

RCL-Variablen

```

206=force
OUTPUT REAL:
203=speed
BEGIN
;...
PROGRAM_END

```

Beispiel: Punkt-Eingänge

```

PROGRAM Prog4
;handshake for the points pick and place by the PLC
INPUT BINARY: 1= ppavail
INPUT POINT: 701= pick, 702= place
;handshake to PLC, points processed pick and place
OUTPUT BINARY: 1= ppready

BEGIN
Loop:
;wait till PLC sets pick and place
WAIT UNTIL ppavail= 1
MOVE TO pick
MOVE TO place
;handshake to PLC, points processed pick and place
ppready= 1
JUMP Loop
PROGRAM_END

```

Abfrage von Kanälen/ Signalen

Die Abfrage und Auswertung der Kanäle bzw. deren zugeordneter Namen erfolgt in Bedingungen.

Beispiel: Abfrage von Kanälen/ Signalen im Programm

```

PROGRAM Prog5
INPUT: 2=met_unit
INPUT INTEGER: 401=grip_force
BEGIN
IF met_unit THEN ...
IF grip_force>=26 THEN...
PROGRAM_END

```



- Ausgangssignale können nicht abgefragt werden.
 - Eine Abfrage ist nur bei Eingangssignalen möglich.
-

Beispiel: fehlerhafte Abfrage

```
MOVE UNTIL grip_force=100 AND meas_height=200 TO pallet
```

Falsch ist:

- MOVE UNTIL grip_force=100 AND meas_height=200 TO pallet

Der Übersetzer würde diese Anweisung wie folgt interpretieren:

- MOVE UNTIL grip_force= ((100 AND meas_height) =200) TO pallet
-



Die Abfrage in Verbindung mit einer "MOVE UNTIL"-Anweisung ist nur jeweils für 1 Eingangssignal zulässig.

Setzen von Signalen

Das Setzen von Signalen erfolgt durch Wertzuweisungen, siehe [Arithmetische Ausdrücke, Seite 83](#).

Beispiel: Setzen von Signalen

```
PROGRAM Prog6
OUTPUT INTEGER: 402=print
BEGIN
  print= 75
PROGRAM_END
```



- Eingangssignale können nicht gesetzt werden.
- Ein Setzen ist nur bei Ausgangssignalen möglich.

Binäre Signale können mit "AND", "OR", bzw. "NOT" verknüpft werden.

Ausnahmen sind die "WAIT UNTIL"-Bedingung und "MOVE UNTIL"-Bedingung. Hierbei sind keine Verknüpfungen zulässig.

8 RCL-Programmsteuerung

8.1 WAIT-Anweisung

Syntax:

WAIT expression

Verweilzeit (Dwell time)

Mit der "WAIT"-Anweisung können Sie Verzögerungen und Unterbrechungen in der Programmausführung programmieren.

Soll der Roboter auf einer Position eine bestimmte Zeit verweilen, kann diese Zeitdauer direkt vorgegeben werden.

Verweilzeit bei "WAIT"

- Maßeinheit: Sekunde
- Eingabebereich: 0.0 bis MAX_REAL s

Programmierung:

Die Zeitvorgabe erfolgt als Zahlenwert im Anschluss an die Anweisung "WAIT", z.B. WAIT 8.5

Beispiel:

Der Roboter transportiert ein Gefäß zum Dosierer, um es auffüllen zu lassen.

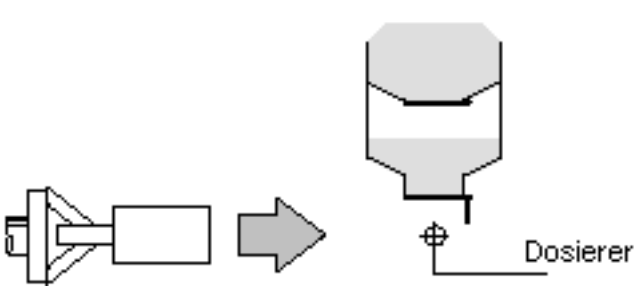
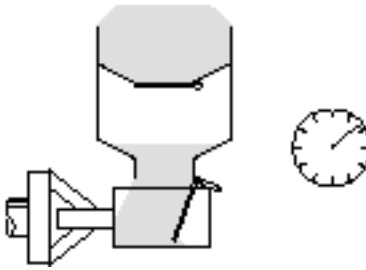
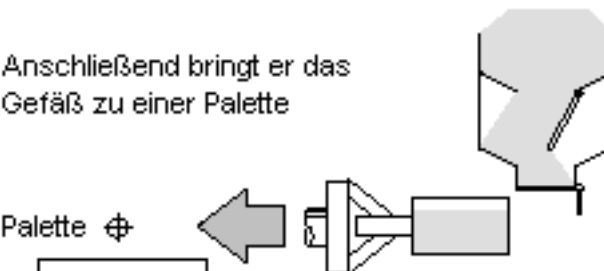
Syntax	Beschreibung
MOVE TO met_unit	
WAIT 8.5	Die Füllzeit beträgt ca. 8,5 Sekunden 
MOVE TO pallet	Anschließend bringt er das Gefäß zu einer Palette 

Abb.8-1: Programmierbeispiel 1 zur WAIT-Anweisung

RCL-Programmsteuerung

Warten auf Eintreffen einer Bedingung*Syntax:*

`WAIT UNTIL variable rel_operator expression [max_time=expression [ERROR statement]]`

`rel_operator: =, <>, <=, >=, <, >`

Soll der Roboter auf einer Position das Eintreffen einer Bedingung abwarten, können Sie die Bedingung zusammen mit der "WAIT"-Anweisung angeben.

Die Programmausführung ist dann solange unterbrochen, bis die Bedingung erfüllt ist.



Die Bedingungen können nur über Eingangskanal-Variablen gesetzt werden siehe auch [Kommunikationskanäle zur SPS, Vereinbarung und Programmierung](#) , Seite 67.

Programmierung:

Die Bedingung wird mit dem Wort "UNTIL " an die "WAIT"-Anweisung angehängt, z.B.:

Programm:

`WAIT UNTIL pal_empty=1`

Besteht die Bedingung aus mehreren Eingangskanälen, muss die "WAIT"-Anweisung in mehrere Schritte aufgeteilt werden.

Beispiele:

`WAIT UNTIL signal=1``WAIT UNTIL light=1``MOVE TO pallet`

Falsch wäre:

- `WAIT UNTIL signal=1 AND licht=1`

Der Übersetzer würde diese Anweisung wie folgt interpretieren:

- `WAIT UNTIL signal= ((1 AND licht) =1)`

Beispiel:

Der Roboter transportiert ein Werkstück zu einem Palettenwechsler.

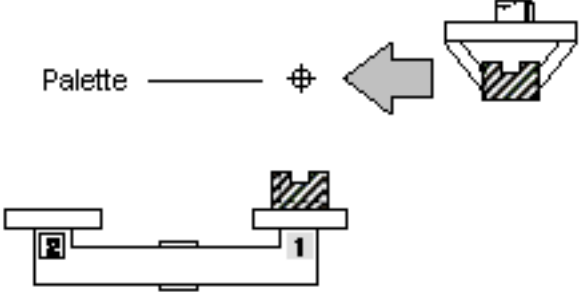
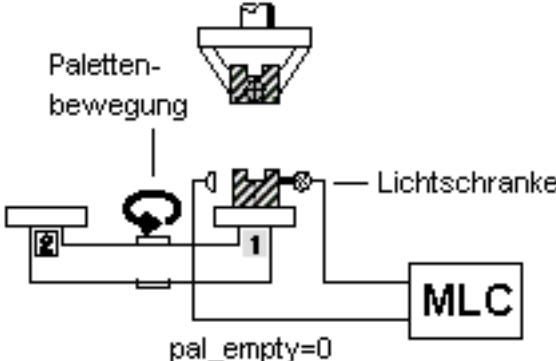
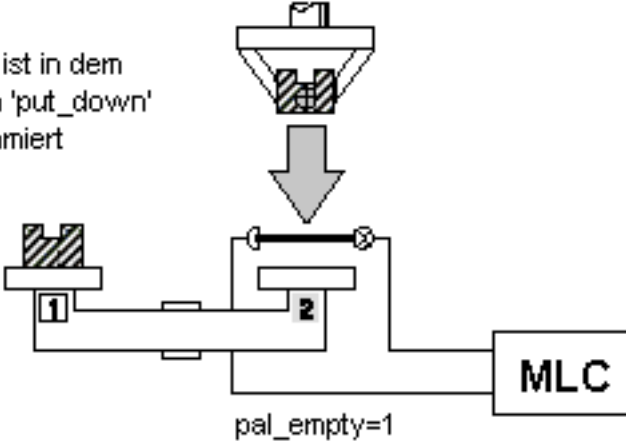
Syntax	Beschreibung
MOVE TO pallet	
WAIT UNTIL pal_empty=1	Der Palettenwechsler wartet, bis die Palette leer ist. Er gibt das Signal pal_empty= 1 aus, wenn eine leere Palette in der Wechsellposition eingetroffen ist. 
put_down	Das Ablegen ist in dem Unterprogramm 'put_down' programmiert 

Abb.8-2: Programmierbeispiel 2 zur WAIT-Anweisung

Maximale Wartezeit *MAX_TIME*

- Maßeinheit: Sekunde
- Eingabebereich: 0.5 bis 32000 s

RCL-Programmsteuerung

In Verbindung mit einer WAIT-Bedingung können Sie eine maximale Wartezeit vorgeben.

Die Programmausführung ist dann solange unterbrochen, bis die Bedingung erfüllt oder die angegebene Maximalzeit überschritten ist.

Mit der maximalen Wartezeit können sie eine Fehleranweisung programmieren. Wird die maximale Wartezeit überschritten, führt die Steuerung die Fehleranweisung aus.

Programmierung:

Die maximale Wartezeit wird mit dem RCL-Befehl MAX_TIME angegeben.

Die Programmierung erfolgt im Anschluss an die Warte-Bedingung.

Beispiel:

```
WAIT UNTIL sig=1 MAX_TIME=60
```

Die Programmierung der Fehleranweisung erfolgt im Anschluss an "MAX_TIME".

Beispiel (Fehler):

```
WAIT UNTIL sig=1 MAX_TIME=60 ERROR PAUSE
```

Ablaufbeispiel

Ein Behälter ist mit einer Flüssigkeit gefüllt.

Ein Sensor misst das Gewicht des Behälters.

Ist der Behälter voll, gibt der Sensor das Signal "weight= 1" aus. Die Abfüllanlage schließt das Ventil und der Roboter transportiert den Behälter ab. Die durchschnittliche Füllzeit beträgt ca. 25 s.

Der Behälter darf maximal 45 s unterhalb der Abfüllvorrichtung bleiben, um den Produktionsablauf nicht zu gefährden.

Syntax	Beschreibung
WAIT UNTIL weight=1 MAX_TIME=45	Fehlerbehandlung
ERROR JUMP f_end	
.. f_end:	

Abb.8-3: Fehlerbehandlung zum Ablaufbeispiel

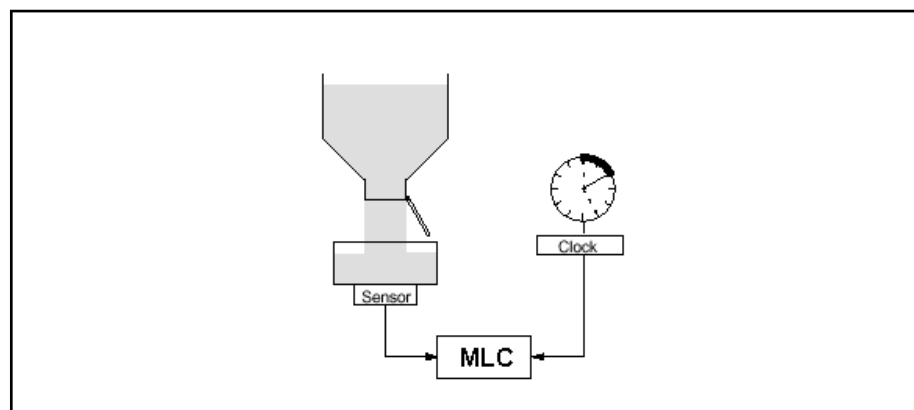


Abb.8-4: Abfüllvorgang begonnen

Fall 1

- Der Behälter wird vollständig aufgefüllt.

RCL-Programmsteuerung

Der Sensor meldet der MLC "weight = 1".

- Der Roboter transportiert den Behälter ab.
- Die "MAX_TIME" wurde nicht erreicht.

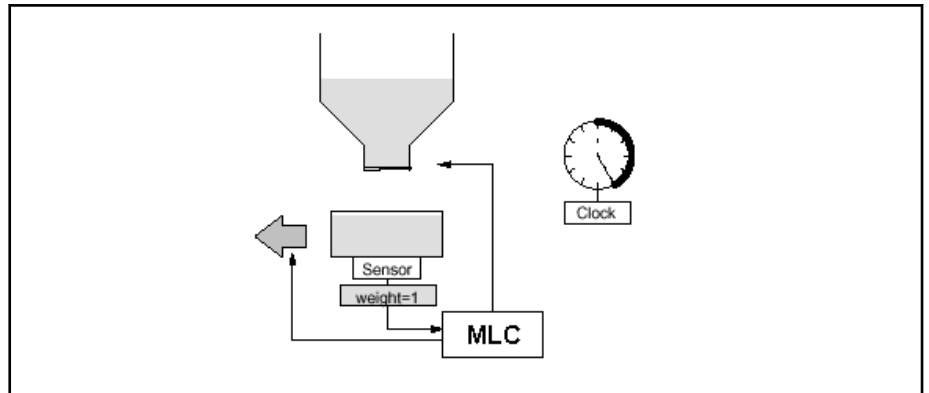


Abb.8-5: Fall 1, Vorrat ausreichend

Fall 2

- Die Abfüllanlage ist leer, und der Behälter ist nicht vollständig gefüllt.
- Die "MAX_TIME" ist erreicht.
Das Programm wird bei "f_ende" fortgesetzt.
- Das Programm "f_ende" weist den Roboter an, den teilgefüllten Behälter auszusondern.

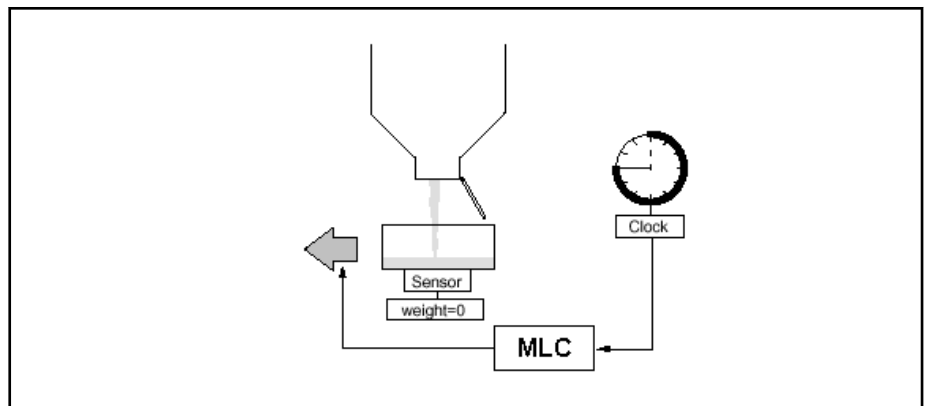


Abb.8-6: Fall 2, Behälter leer

8.2 PAUSE-Anweisung

Mit der "PAUSE"-Anweisung können Sie die Programmausführung anhalten. Zur Fortführung des Programmlaufs muss das SPS-Programm erneut den Funktionsbaustein "MB_IRDProgStart" aufrufen.

In K-0-0021, Kinematik-Status, wird dabei das Bit 26, IRD_PROGRAM_PAUSE gesetzt.

Damit wird gleichzeitig auch `KinData[n].dwKinStatus_i.26` beeinflusst.

Programmierung Die "PAUSE"-Anweisung besteht aus dem RCL-Befehl "PAUSE".



Es empfiehlt sich, vor dem Befehl "PAUSE" einen Ausgang zu setzen, um den Bediener über den Programmfluss zu informieren.

8.3 HALT-Anweisung

Die "HALT"-Anweisung beendet die Ausführung einer Anweisungsfolge im Hauptprogramm.

Am Befehl "HALT" erkennt die Steuerung während des Programmlaufs, dass das Programm beendet ist.



Bei aufgerufenen externen Unterprogrammen bewirkt "HALT" den Rücksprung in das aufrufende, aktive Hauptprogramm.

Programmierung

Der Befehl "HALT" wird vor der Unterprogramm-Vereinbarung bzw. vor dem Programmende eingegeben. Der HALT-Befehl kann innerhalb eines Programms mehrfach verwendet werden.



Ist aus der Programmstruktur der Programm-Halt erkennbar, z. B. bei der RCL-Anweisung "PROGRAM_END", generiert die Steuerung beim Übersetzen den Befehl "HALT" selbsttätig.

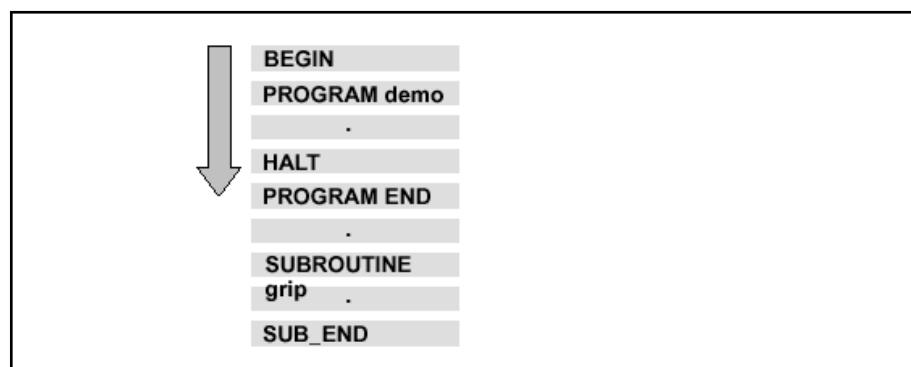


Abb.8-7: HALT-Anweisung im Programmablauf

8.4 REPEAT-Anweisung

Syntax:

```

REPEAT [expression] TIMES
  statement(s)
REPEAT_END
  
```

Ein Programmteil kann mit einer "REPEAT"-Anweisung mehrmals abgearbeitet werden.

Man spricht in diesem Fall von einer Programmteil-Wiederholung.

Der Programmteil ist am Anfang durch die Wiederhol-Anweisung "REPEAT n TIMES" gekennzeichnet.

Für die Anzahl der Wiederholungen können Zahlen, Variablen oder Ausdrücke vom Typ "INTEGER", siehe [Übersicht, Seite 53](#), eingesetzt werden.

Ist Anzahl = 0 oder negativ, wird die Schleife nicht ausgeführt.

Den Schluss des Programmteils bildet der RCL-Befehl "REPEAT_END".

Programmierung

Ein bestimmter Programmteil soll acht mal ausgeführt werden, d.h. die RCL-Anweisung lautet "REPEAT 8 TIMES".

Der Programmteil wird mit "REPEAT_END" abgeschlossen.

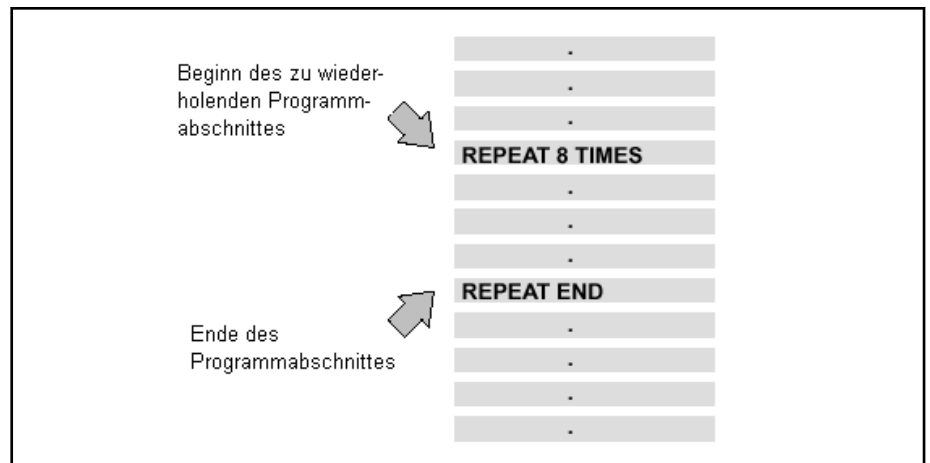


Abb.8-8: REPEAT-Anweisung im Programmablauf

8.5 JUMP-Anweisung

Innerhalb von Haupt- und Unterprogrammen können Sie Marken (Labels) setzen, auf die mit einer Sprung-Anweisung gesprungen werden kann. Dabei sind Vorwärts- und Rückwärtssprünge möglich.

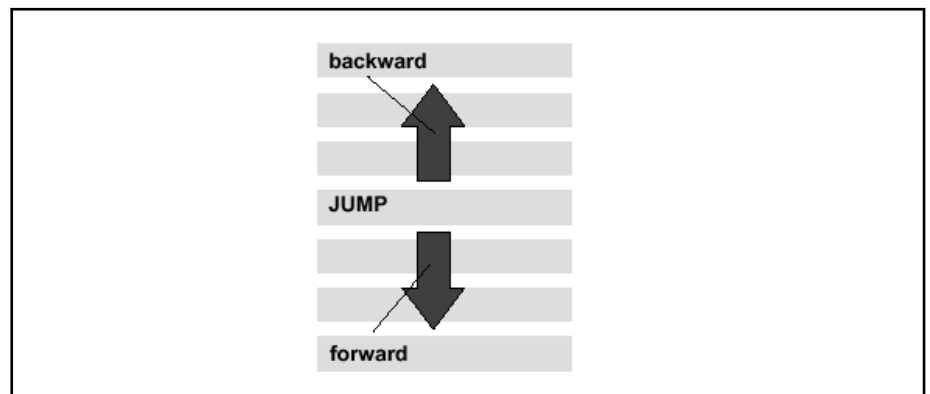


Abb.8-9: Vorwärts- (forward) und Rückwärtssprung (backward)

Die Sprunganweisung besteht aus dem RCL-Befehl "JUMP" und dem Namen der gesetzten Sprungmarke.

Die Marken (Sprungziele) werden mit Namen gekennzeichnet. Der Name besteht aus maximal zwölf Zeichen.



Das erste Zeichen muss ein Buchstabe sein. Als Sonderzeichen ist nur das Underline "_" zugelassen. Groß- und Kleinschreibung sind gleichwertig.

Zur Unterscheidung gegenüber Unterprogramm-Aufrufen ist das Setzen einer Sprungmarke mit einem Doppelpunkt ":" kenntlich zu machen.



Eine bestimmte Sprungmarke darf nur einmal gesetzt werden! Die Anzahl verschieden benannter Sprungmarken ist beliebig.



Sprünge aus dem Hauptprogramm in ein Unterprogramm und umgekehrt sind nicht erlaubt!

RCL-Programmsteuerung

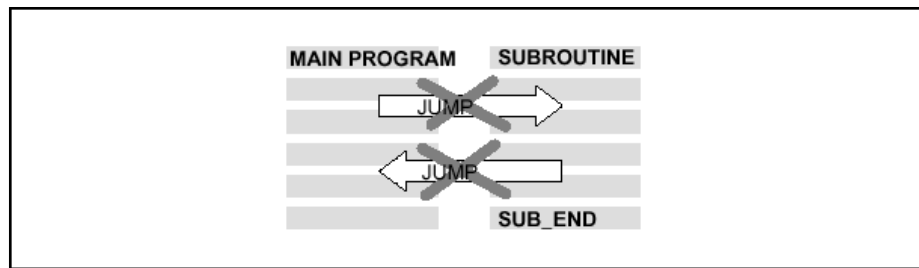


Abb.8-10: Verbotene Sprünge zwischen Haupt- und Unterprogramm



Sprünge in Programmteil-Wiederholungen sind nicht erlaubt.
Sprünge aus Programmteil-Wiederholungen heraus sind erlaubt.

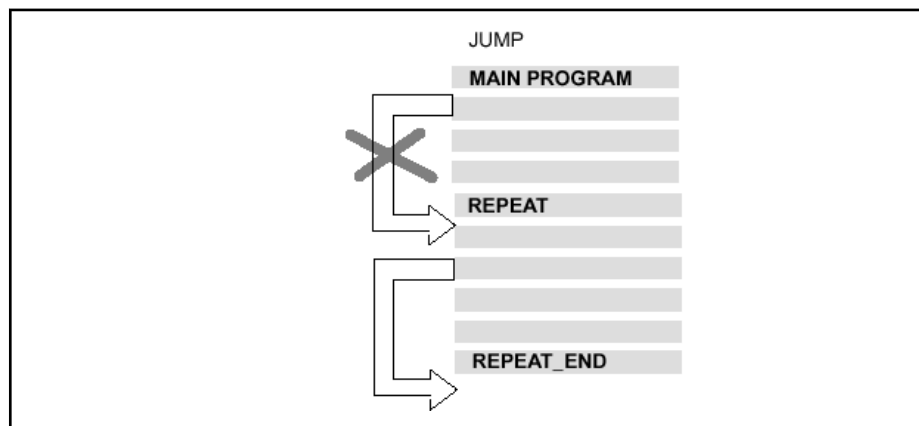


Abb.8-11: Verbotener Sprung in Schleife; erlaubter Sprung aus Schleife

Programmierung Innerhalb eines Programms soll auf die Sprungmarke "table" (vorwärts) gesprungen werden.

Syntax:

`JUMP table ;Jump-Anweisung zur Sprungmarke 'table'`

`table: ;Setzen der Sprungmarke namens table`

Das Programm wird bis zur Sprunganweisung "JUMP table" abgearbeitet.

Es erfolgt ein Sprung zur Sprungmarke "table".

Anschließend setzt die Steuerung das Programm von dieser Marke aus weiter fort.

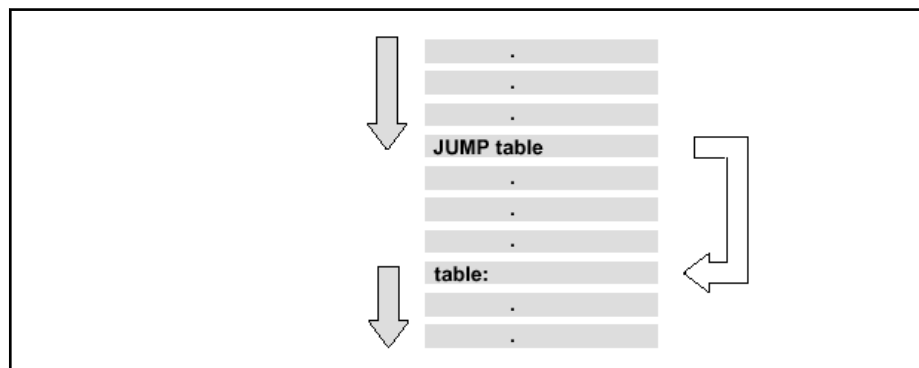


Abb.8-12: JUMP-Anweisung im Programmablauf

8.6 IF-THEN-(ELSE)-Anweisung

Syntax:

```
IF condition THEN statement  
[ELSE statement]
```

```
IF condition THEN  
  BEGIN  
    statement(s)  
  END  
[ELSE statement]
```

```
IF condition THEN  
  BEGIN  
    statement(s)  
  END  
ELSE  
  BEGIN  
    statement(s)  
  END
```

An frei wählbaren Stellen innerhalb eines Haupt- oder Unterprogramms kann der weitere Programmablauf von einer Bedingung abhängig gemacht werden (bedingte Anweisungen).

Unter Bedingung versteht man einen Ausdruck vom Typ "BINARY".

Die Bedingung ist erfüllt, wenn die Aussage richtig ist, also die Variable den Wert tatsächlich hat, bzw. einen Wert innerhalb des angegebenen Wertebereichs angenommen hat.

Die Bedingung ist nicht erfüllt, wenn die Aussage falsch ist.

channel	=	1
l	=	15
sig1	AND	sig2
word	=	j
force	>	115.0
torque	>=	thresholdval



Bedingungen müssen bei Verknüpfung mit AND, OR, bzw. NOT ggf. in Klammern gesetzt werden, siehe [Logische Ausdrücke, Seite 85](#), um die gewünschten Prioritäten der Bedingungs-Operation zu erhalten.

Programmierung

Die bedingte Anweisung hat die Form:

```
IF condition THEN statement 1  
ELSE statement 2
```

"IF" die Bedingung ist erfüllt, "THEN" führt die Steuerung Anweisung 1 aus, "ELSE", also wenn die Bedingung nicht erfüllt ist, führt die Steuerung Anweisung 2 aus.

RCL-Programmsteuerung

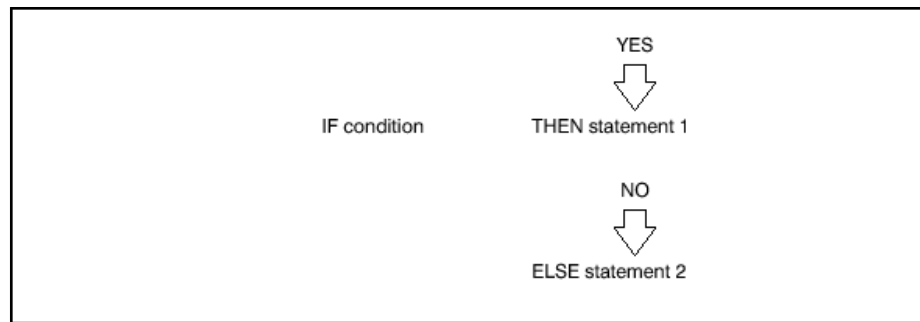


Abb.8-13: IF..THEN..ELSE

Mögliche Anweisungen sind zum Beispiel:

- Programm-Aufrufe (Haupt-, Unterprogramm-Aufrufe)
- Sprung-Anweisungen (JUMP)
- Pause-Anweisungen (PAUSE), Halt-Anweisungen (HALT), Verzögerungen (WAIT)
- Wiederholungen (REPEAT)
- Bewegungsbefehle (MOVE, MOVE UNTIL)
- Bedingte Anweisungen (IF...THEN...ELSE) usw.

Ist in der "THEN"-Anweisung oder "ELSE"-Anweisung kein Sprung programmiert, führt die Steuerung den Programmlauf mit denen auf die bedingte Anweisung folgenden Programmschritten fort.

Die "ELSE"-Anweisung kann entfallen. In diesem Fall führt die Steuerung ebenfalls den Programmlauf mit den auf die bedingte Anweisung folgenden Programmschritten fort.

Beispiel

Der Roboter soll in einem Regal eine mit einem Werkstück beladene Palette, Gewicht ca. 200 kg, suchen. Dazu hat er einen Sensor in seinem Hebewerkzeug, der der Steuerung das Gewicht der Palette überträgt.

Hat er das Werkstück gefunden, soll er es zur Maschine bringen.

Ausgangsposition: Der Roboter steht vor der obersten Palette.

Programm:

```

next:
  in                ;Aufruf UP 'in': Einfahren und Anheben der Palette
IF weight >= 180.0
  THEN MOVE TO machine
  ELSE JUMP search
  sig= 1            ;Signal zur SPS, Palette abgeliefert
PAUSE
search:
  down              ;SUBROUTINE ablassen, rausfahren und runterfahren
  JUMP next
  
```

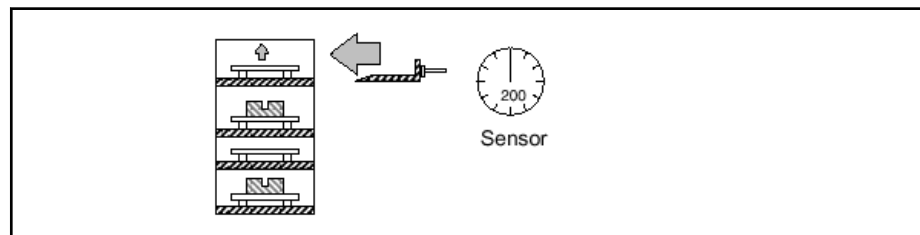


Abb.8-14: Fahren an den Palettenständer

RCL-Programmsteuerung

Programm:

```
IF weight >= 180.0      ;Auswerten des Signals < 180.0 kg, leere Palette
  THEN MOVE TO machine
  ELSE JUMP search      ;Steuerung entscheidet sich für ELSE-Anweisung
                        ;Programmsprung auf die Sprungmarke 'search:'
```

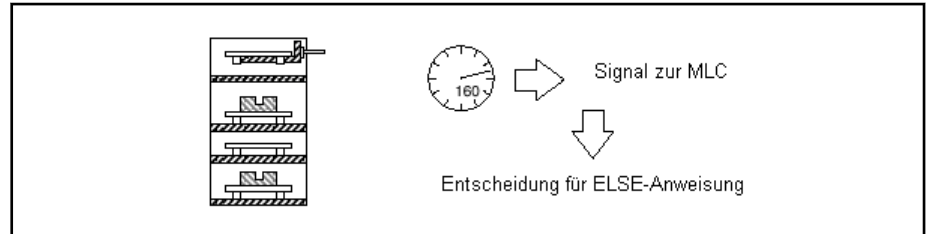


Abb. 8-15: Anheben einer leeren Palette

Programm:

```
search:
down      ;Aufruf UP 'down': Ablassen der Palette,
          ;Herausfahren und Tieferfahren.
```

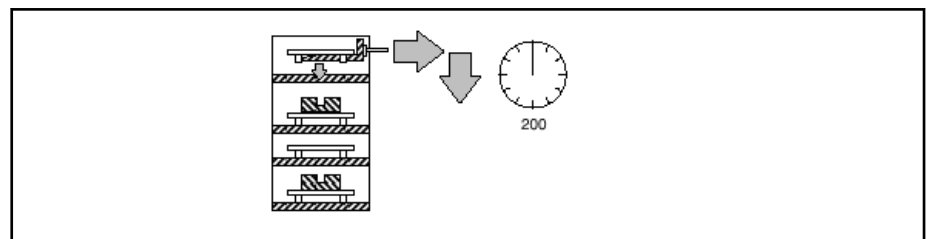


Abb. 8-16: Leere Palette wieder absetzen

Programm:

```
JUMP next      ;Programm-Sprung auf die Sprungmarke 'next:'

next:
in             ;Aufruf UP 'in': Einfahren und Anheben der Palette
```

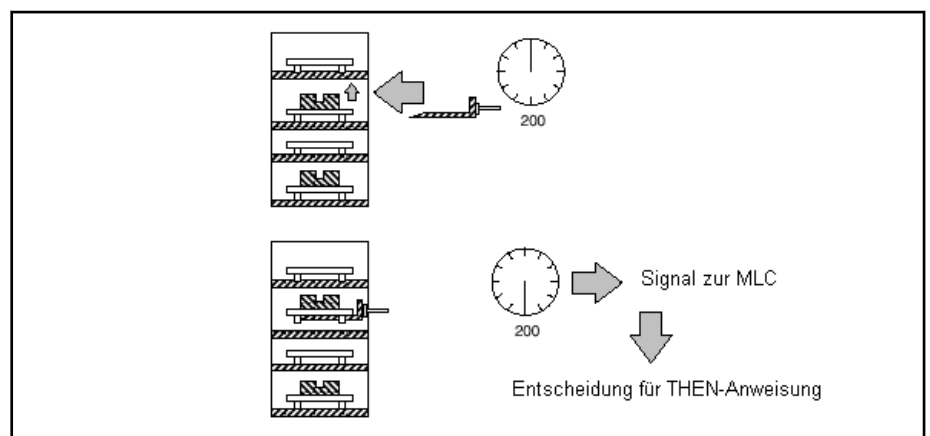


Abb. 8-17: Wiederholung, Palette voll...

Programm:

```
IF weight >= 180.0      ;Auswerten des Signals 200 kg ==> THEN
  THEN MOVE TO machine  ;Fahrt in Richtung 'Maschine'
  ELSE JUMP search
```

Programm:

```
sig= 1      ;Signal zur SPS, Palette abgeliefert
PAUSE      ;Programm anhalten und auf SPS warten
```

RCL-Programmsteuerung

8.7 CASE-Anweisung

Durch die Verzweigungsanweisung "CASE" kann eine Auswahl aus mehreren Alternativen realisiert werden. Verschachtelte "IF-THEN"-Abfragen können hiermit vermieden und die Ausführungszeit des Programms verkürzt werden.

Programmierung Die bedingte Anweisung hat die Form

Syntax:

```
CASE selection_expression
  EQUAL: statement
  EQUAL: statement
  DEFAULT: statement
CASE_END
```

Zulässige Typen für **selection_expression** sind "BINARY", "INTEGER", "CHAR" und "TEXT".

Die Konstanten der Konstantenliste müssen kompatibel zum Auswahl Ausdruck sein. Eine Konstante darf nur einmal innerhalb der Verzweigungsanweisung verwendet werden.

Die Angabe des "DEFAULT"-Teils ist optional.



Fehlt der "DEFAULT"-Teil in der Verzweigungsanweisung, wird bei der Programmabarbeitung kein Laufzeitfehler gemeldet.

Dies gilt auch, wenn keine der Anweisungen zutrifft.

Programm:

```
PROGRAM case_ex
INPUT INTEGER: amount      ;Wert kommt von der SPS
INTEGER: offset            ;Deklarationen
BEGIN                     ;Programmstart
  offset=0
  CASE amount
    EQUAL 9,10: offset=0
    EQUAL 11,12: offset=1
    EQUAL 13: offset=2
    DEFAULT offset=5
  CASE_END
PROGRAM_END
```

9 Wertzuweisungen und Verknüpfungen

9.1 Wertzuweisungen

Über Wertzuweisungen werden Variablen für Geschwindigkeit und Beschleunigung im Verlauf eines Programms Werte zugeordnet. Dies gilt auch für Standard-Variablen.

Der zugewiesene Wert muss vom gleichen Datentyp wie die Variable sein.

Im folgenden Abschnitt ist die Wertzuweisung von allgemeinen Variablen beschrieben. Die Wertzuweisung von Positionen finden Sie in [Punktvariablen, Seite 59](#).

Programmierung *Syntax:*

variable=expression

Die Zuweisung erfolgt über das "="-Zeichen.

Auf der linken Seite der Zuweisung ist der Name der Variablen anzugeben, der ein Wert zugewiesen wird. Auf der rechten Seite steht ein Ausdruck.

Für den Ausdruck können

- Zahlenwerte (Konstanten)
- Variable
- Arithmetische Ausdrücke
- Standard-Funktionen

einggegeben werden.

Das Vorzeichen "+" kann entfallen.

Das Vorzeichen "-" steht unmittelbar vor der Variablen oder Konstanten. Falls zwei Operatoren unmittelbar aufeinander folgen, ist der negative Ausdruck in Klammern zu setzen.

Sollen alle Komponenten einer Variablen vom Typ "POINT" oder "JC_POINT" negiert werden, sind alle Komponenten mit -1 zu multiplizieren. Auch in diesem Fall ist -1.0 in Klammern zu setzen.

Syntax:

counter=1	;Variable und Zahlenwert
xvalue=yvalue+2.5	;Variable und arithmetischer Ausdruck
distance=SIN(alpha)	;Variable und Standardfunktion
value=-value	;Variable vom Typ REAL und Negation
d=d*(-2.0)	;Variable und negativer Ausdruck
corner=(-1.0)*corner	;Variable vom Typ POINT und Negation

9.2 Verknüpfungen

9.2.1 Arithmetische Ausdrücke

Arithmetische Ausdrücke sind Verknüpfungen von

- Zahlenwerten und/oder
- Variablen und/oder
- Standard-Funktionen und/oder
- weiteren arithmetischen Ausdrücken.

Die Art von Verknüpfung ist durch den Operator festgelegt.

RCL kennt fünf arithmetische Operatoren

Wertzuweisungen und Verknüpfungen

+	Addition z. B. $k = 1 + 5$
-	Subtraktion z. B. $\text{wert} = \text{gewicht} - 1$
*	Multiplikation z. B. $\text{laenge} = \text{breite} * 2$
/	Division z. B. $\text{hoehe\_neu} = \text{hoehe} / 2.0$
MOD	Modulo-Rechnung, nur bei Typ INTEGER z.B. $\text{rest} = \text{zahl} \text{ MOD } \text{teiler}$



- Die Zeichen + und – können auch als Vorzeichen für Variable und Zahlenwerte stehen.
- Variable und Zahlenwerte mit Vorzeichen müssen in Klammern stehen, damit nicht zwei Operatoren unmittelbar aufeinander folgen.

Auf der linken Seite einer Zuweisung darf kein Vorzeichen stehen.

Die Operationen Addition und Subtraktion können mit Variablen und Zahlenwerten (Konstanten) des Typs "INTEGER", "REAL", ""POINT"" und "JC_POINT" durchgeführt werden.

Die Operationen Multiplikation und Division mit Variablen des Typs "INTEGER" und "REAL".

Variable und Zahlenwerte vom Typ "REAL" können auch für die Multiplikation und Division von Punktvariablen verwendet werden, siehe [Punktvariablen, Seite 59](#).

Die Operatoren "*", "/" und "MOD" werden vor + und – abgearbeitet.

Innerhalb dieser Klassen erfolgt die Abarbeitung von links nach rechts.

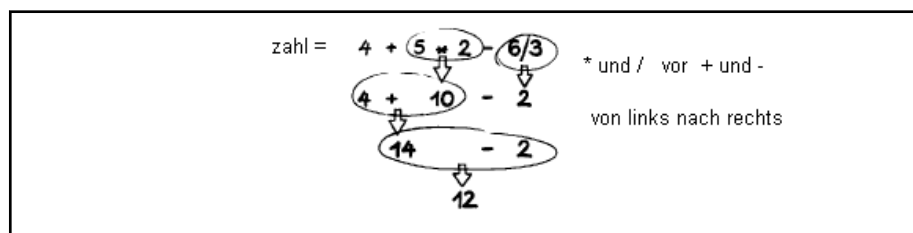


Abb.9-1: Berechnungsbeispiel 1

Darüber hinaus können Sie zusammengehörige Ausdrücke in Klammern setzen.

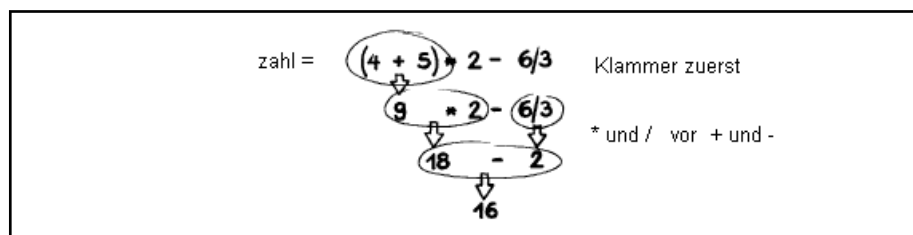


Abb.9-2: Berechnungsbeispiel 2

Modulo-Funktion wert1 MOD wert2

Die Modulo-Funktion, Datentyp "INTEGER", ermittelt aus der Division von "wert1" durch "wert2" den ganzzahligen Rest.

"wert1", "wert2" und das Ergebnis sind vom Typ "INTEGER".

Ermittlung der Spalte einer Palette nach Angabe der Platzzahl und der Spaltenzahl.

Wertzuweisungen und Verknüpfungen

	Spalte 0	Spalte 1	Spalte 2	Spalte 3	Spalte 4
Zeile 0	0	1	2	3	4
Zeile 1	5	6	7	8	9
Zeile 2	10	11	12	13	14
Zeile 3	15	16	17	18	19
Zeile 4	20	21	22	23	24

Abb.9-3: Berechnungsbeispiel 3

Die Palette hat 5 Spalten (0, 1, 2, 3, 4). Welche Spalte und welche Reihe nimmt der Platz 13 ein?

Die Platzzahl ist für "wert1" einzusetzen, die Anzahl der Spalten für "wert2".

Programm:

```
column=13 MOD 5 ;Berechnung: 13/5 = 2 Rest 3. Der Rest gibt die Spalte an, also
;ist Spalte 3 die gesuchte
line=13 / 5 ;Berechnung: 13/5 = 2 Rest 3. Der Quotient gibt die Zeile an, also
;ist Zeile 2 die gesuchte
```

Bei dem Aufruf spalte ermittelt die Modulo-Funktion anhand des Wertes des Restes, im Beispiel= 3, die Spalte in der sich der gesuchte Platz befindet.

Bei dem Aufrufzeile ermittelt die Divisions-Funktion anhand des Wertes des Ergebnisses, im Beispiel= 2, die Zeile in der sich der gesuchte Platz befindet.

9.2.2 Vergleichsoperationen

In Bedingungen, z. B. UNTIL-Bedingung oder IF-Bedingung, fragt die Steuerung Werte und Zustände ab. Dieses Abfragen geschieht über Vergleiche.

Programmierung Folgende Zeichen stehen zur Verfügung:

=	gleich, z.B.: p = 1
<>	ungleich, z.B.: p <> 1
>	größer, z.B.: p > 1
>=	größer oder gleich, z.B.: p >= 1
<	kleiner, z.B.: p < 1
<=	kleiner oder gleich, p <= 1



Variablen vom Typ "POINT", "JC_POINT" und "RECORDS" (Verbundvariablen) können nur auf "=", gleich, oder "<>", ungleich, abgefragt werden.

9.2.3 Logische Ausdrücke

Bedingungen prüft die Steuerung auf ihren Wahrheitsgehalt, siehe auch [Vergleichsoperationen](#), Seite 85, und [IF-THEN-\(ELSE\)-Anweisung](#), Seite 79. Bedingungen können nur zwei Werte annehmen.

"Wert 1" für wahr und "Wert 0" für falsch.

Dies gilt auch für Variablen des Typs "BINARY".

Wertzuweisungen und Verknüpfungen

Diese Variablen können ebenfalls immer nur einen der beiden Werte 0 oder 1 einnehmen.

Verknüpfung von Bedingungen

Oft ist der weitere Programmablauf von mehreren Bedingungen gleichzeitig abhängig.



Variablen und Ausdrücke (Bedingungen) vom Typ "BINARY" können mit den logischen Operationen AND, OR und NOT verknüpft werden.

bed_1	bed_2	bed_1 AND bed_2
1	1	1
1	0	0
0	1	0
0	0	0

Abb.9-4: Verknüpfung zweier Bedingungen bed_1 und bed_2 mit AND-Wahrheitstafel

bed_1	bed_2	bed_1 OR bed_2
1	1	1
1	0	1
0	1	1
0	0	0

Abb.9-5: Verknüpfung zweier Bedingungen bed_1 und bed_2 mit OR-Wahrheitstafel

Verneinung von Bedingungen

Der Wahrheitsgehalt von Bedingungen und Variablen vom Typ "BINARY" kann mit dem Wort "NOT" umgekehrt werden.

Beispiel:

Bedingung bed_1 ist wahr, also gilt: (bed_1) = 1

Setzt man das Wort "NOT" vor die Bedingung bed_1, gilt für den Wahrheitsgehalt von "NOT" bed_1:

(NOT bed_1) = 0

oder

Beispiel:

Die Bedingung bed_2 ist unwahr, also gilt (bed_2) = 0

gilt für die Umkehrung mit "NOT":

(NOT bed_2) = 1

9.2.4 Rangfolge der Operationen

Programmierung von Verknüpfungen mehrerer Bedingungen

Ergebnisse von Vergleichsoperationen sind immer vom Datentyp "BINARY". Werden mehrere Bedingungen miteinander verknüpft, muss die Rangfolge der Operatoren beachtet werden.

1. NOT

Wertzuweisungen und Verknüpfungen

2. *, /, MOD, AND
3. +, -, OR
4. =, <>, >, >=, <, <=.

Beispiel:

Abfrage nach Zahlenwerten der Variablen i und j vom Typ "INTEGER"

IF i = 10 AND j = 50 THEN...

In diesem Beispiel verarbeitet die Steuerung zuerst den Ausdruck 10 AND j.
10 AND j bedeutet für die Steuerung aber einen Typ-Konflikt, denn die Konstante 10 ist vom Typ "INTEGER" und nicht vom Datentyp "BINARY".

Um die Reihenfolge der Verarbeitung von Ausdrücken festzulegen, werden Klammern gesetzt.

IF (i = 10) AND (j = 50) THEN...

10 RCL-Funktionen

10.1 SIN-Funktion

Die Sinus-Funktion, Datentyp "REAL", stellt den mathematischen Zusammenhang zwischen einem Winkel und den Seitenlängen im rechtwinkligen Dreieck her.

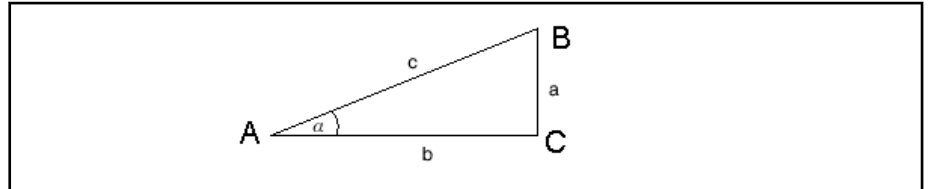


Abb. 10-1: Rechtwinkliges Dreieck

$$\sin \alpha = \frac{\text{Gegenkathete}}{\text{Hypothenuse}} = \frac{a}{c}$$

Programmierung Der Winkel α muss im Bogenmaß als Radiant (rad) angegeben werden. Die Zahl ist vom Typ "REAL".

$$\text{rad} = \alpha \frac{\pi}{180^\circ} \Leftrightarrow \alpha = \text{rad} * \frac{180^\circ}{\pi}; \pi = 3,14$$

Syntax:

avalue= c*SIN(rad)



Die Bezeichnung "avalue" statt "a" wurde gewählt, weil "A" als reservierter Name für die Beschleunigung schon vergeben ist.

10.2 COS-Funktion

Die Cosin-Funktion, Datentyp "REAL", stellt den mathematischen Zusammenhang zwischen einem Winkel und den Seitenlängen im rechtwinkligen Dreieck her.

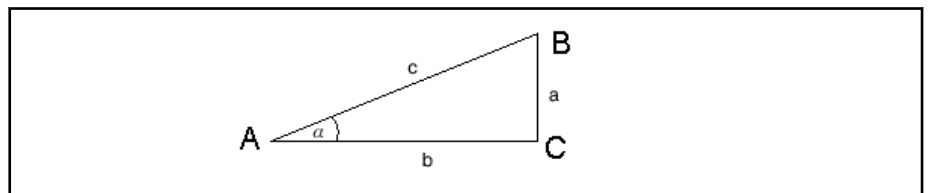


Abb. 10-2: Rechtwinkliges Dreieck

$$\cos \alpha = \frac{\text{Ankathete}}{\text{Hypothenuse}} = \frac{b}{c}$$

Programmierung Der Winkel α muss im Bogenmaß als Radiant (rad) angegeben werden. Die Zahl ist vom Typ "REAL".

Syntax:

b= c*COS(rad)

RCL-Funktionen

10.3 ATAN-Funktion

Die Arcustangens-Funktion, Datentyp "REAL", ermittelt den Winkel im rechtwinkligen Dreieck durch Angabe der Seitenlänge.

Die Arcustangens-Funktion ist die Umkehrung der Tangens-Funktion, diese ist wie folgt definiert:

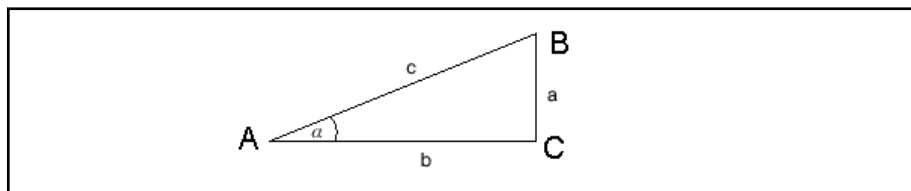


Abb. 10-3: Rechtwinkliges Dreieck

$$\tan \alpha = \frac{\text{Gegenkathete}}{\text{Ankathete}} = \frac{a}{b}$$

Die Umkehrung der Tangens-Funktion lautet dann

$$\alpha = \text{atan} \frac{a}{b}$$

Der Winkel liegt im Bogenmaß als Radiant "rad" vor und kann umgerechnet werden.

$$\text{rad} = \alpha \frac{\pi}{180^\circ} \Leftrightarrow \alpha = \text{rad} * \frac{180^\circ}{\pi}; \pi = 3,14$$

Syntax:

alpha= c*ATAN(ava lue/b)



Die Bezeichnung "ava lue" statt "a" wurde gewählt, weil "A" als reservierter Name für die Beschleunigung schon vergeben ist.

10.4 SQRT-Wurzel-Funktion

Die Wurzel-Funktion, Datentyp "REAL", ermittelt den Wert der Quadratwurzel (Square root) eines Ausdrucks.

Berechnung der Länge c im rechtwinkligen Dreieck.

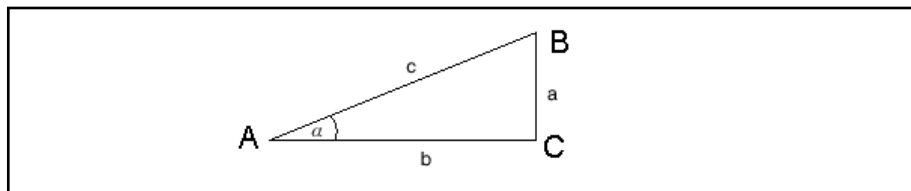


Abb. 10-4: Rechtwinkliges Dreieck

Satz des Pythagoras:

$$a^2 + b^2 = c^2 \Rightarrow c = \sqrt{a^2 + b^2}$$

Syntax:

c=SQRT(ava lue*ava lue+b*b)



Die Bezeichnung "avalue" statt "a" wurde gewählt, weil "A" als reservierter Name für die Beschleunigung schon vergeben ist.

10.5 ABS-Funktion

Syntax:

ABS(argument)

Die Funktion liefert als Ergebnis den absoluten Betrag des übergebenen Arguments. Das Argument darf vom Typ "INTEGER" oder "REAL" sein. Das Ergebnis ist vom gleichen Typ wie das Argument.

Beispiel:

deviation=**ABS**(delta)

10.6 TRUNC-Funktion

Syntax:

TRUNC(argument)

Die Funktion wandelt das Argument vom Typ "REAL" durch Abschneiden in einen Wert vom Typ "INTEGER".

Bei positivem Argument erhält man als Ergebnis die größte ganze Zahl kleiner oder gleich dem Argument, z. B. 3.7 wird zu 3.

Bei negativem Argument erhält man als Ergebnis die kleinste ganze Zahl größer oder gleich dem Argument, z. B. -3.7 wird zu -3.

10.7 ORD-Funktion

Syntax:

ORD(char_variable)

Diese Funktion liefert den INTEGER-Wert der Variablen vom Typ "CHAR".

Beispiel:

char_num=**ORD**(asc_zeich)

char_num=**ORD**('i')

10.8 CHR-Funktion

Syntax:

CHR(integer)

Diese Funktion liefert entsprechend der Berechnung "integer MOD 256" einen Wert vom Typ "CHAR".

Beispiel:

asc_char=**CHR**(34)

10.9 ROUND-Funktion

Syntax:

ROUND(argument)

RCL-Funktionen

Die Funktion wandelt das Argument vom Typ "REAL" in einen Wert vom Typ "INTEGER".

Die Rundung erfolgt jeweils zu der dem Argument am nächsten liegenden ganzen Zahl.

Bei Argumenten die genau zwischen zwei benachbarten Ganzzahlen liegen (z.B. 0.5, 1.5) wird immer zur geraden Ganzzahl gerundet, d.h. 1.5 wird auf 2 gerundet und 6.5 wird auf 6 gerundet.

Beispiel:

```
value_3=ROUND(7.81) ;value_3 wird 8 zugewiesen
value_4=ROUND(-5.43) ;value_4 wird -5 zugewiesen
```

10.10 EOF-Funktion

Syntax:

```
END_OF_FILE(argument)
```

Diese Funktion ermöglicht das Abfragen, ob das Dateieinde beim Lesen einer dat-Datei erreicht ist.

Das Argument ist eine Variable vom Typ "FILE". Die Funktion liefert den Wert 1 (wahr), wenn das Ende der Datei erreicht ist und den Wert 0 (falsch), wenn das Dateieinde noch nicht erreicht ist.

Beispiel 1:

```
IF END_OF_FILE(dat_values) ;Verwendung der Funktion in einer Bedingungsanweisung
THEN statements
ELSE statements
```

Beispiel 2:

```
EOF=END_OF_FILE(dat_values) ;Verwendung der Funktion in einer Zuweisung
```

10.11 ASSIGN-Funktion

Die Anweisung "ASSIGN" ermöglicht, den Namen von Dateien zur Programmlaufzeit durch das Anwenderprogramm zu ändern.



Die ASSIGN-Anweisung ist nur auf geschlossene Dateien und Standardkanäle anwendbar.

Beispiel:

```
FILE datvar,wrvar ;Dateien datvar, wrvar
ASSIGN wrvar,wrname ;Dateivariablen wrvar <-- Textvariable wrname
ASSIGN datvar,datname ;Dateivariablen datvar <-- Textvariable datname
READ_BEGIN datvar ;Lesezeiger auf Anfang der Datei datvar
READ datvar, oneline ;Lesen einer Zeile
WRITE_END wrvar ;Schreibzeiger auf Ende der Datei wrvar
WRITE wrvar, oneline ;Schreiben einer Zeile
CLOSE datvar ;Schließen der Datei
CLOSE wrvar ;Schließen der Datei
PROGRAM_END
```

10.12 INT_ASC-Funktion

Die Standardprozedur "INT_ASC" konvertiert einen ganzzahligen Wert in ein Feld von Zeichen.



Das Format ist rechtsbündig mit evtl. führenden Leerzeichen. Wird bei der Wandlung ein Fehler erkannt, bleibt das Zeichenfeld unverändert und der entsprechende Fehlercode wird zurückgegeben.

Syntax:

```
OUTPUT INTEGER: 401= toPLC
INTEGER: int_number      ;Vorgabe
INTEGER: index_na        ;int_number = zu konvertierende Zahl.
INTEGER: length_char     ;Nur vom Typ INTEGER, z.B.
INTEGER: error_ret       ;INT-Zahl <= 2147483647
ARRAY[1..10] CHAR: ascii_array ;Ergebnisrückmeldung ascii_feld = Ergebnisfeld vom Typ ARRAY[ ] CHAR
BEGIN
  int_number=-1234      ;Vorgabe
  index_zf=1           ;Startindex im Zeichenfeld
  length_char=5         ;Maximale Anzahl von Zeichen, die
                        ;für die zu konvertierende Zahl reserviert werden.
  INT_ASC(int_number,ascii_array,index_na,length_char,error_ret)
                        ;Zielbereich wird vor Umwandlung mit Leerzeichen initialisiert.
  IF error_ret<>0       ;Ergebnisrückmeldung

  THEN toPLC= error_ret ;error_ret= Fehlernummerausgabe
  ELSE toPLC= int_number
```

Bedeutung der Fehlernummern		
0		Keine Fehler
-1		Startindex liegt außerhalb der Feldgrenzen
-2		Endindex (Startindex + Länge) liegt außerhalb der Feldgrenzen
-3		reservierte Länge ist zu klein
-4		Bereichsüberschreitung (Wert zu groß)
-5		Feldlänge < 0
-6		Feldlänge = 0

10.13 ASC_INT-Funktion

Die Standardprozedur "ASC_INT" konvertiert ein Feld von Zeichen in einen ganzzahligen Wert.

Die Prozedur liest ab der Startposition solange Zeichen ein, bis ein Zeichen erkannt wird, bei dem es sich nicht um eine Ziffer handelt oder die maximale Anzahl an Zeichen gelesen, bzw. das Ende des Zeichenfeldes erreicht ist.

Syntax:

```
OUTPUT INTEGER: 401= toPLC
INTEGER: int_number      ;Ergebnis: konvertierte Zahl nur INTEGER zulässig
INTEGER: index_na        ;Position im Feld ab der die Zahl gelesen werden soll
INTEGER: length_char     ;maximale Anzahl zu lesender Zeichen
INTEGER: error_ret
ARRAY [1..10] CHAR: ascii_array ;zu konvertierendes Feld vom Typ CHAR
BEGIN
  ascii_array='AaBc'
  index_na=1
  length_char=5
  ASC_INT(ascii_array,int_number,index_na,length_char,error_ret)
  IF error_ret<>0       ;Ergebnisrückmeldung error_ret= Fehlernummerausgabe
  THEN toPLC= error_ret ;error_ret= Fehlernummerausgabe
  ELSE toPLC= int_number
```

RCL-Funktionen

Bedeutung Fehlernummer	0	Keine Fehler
	-1	Startindex liegt außerhalb der Feldgrenzen
	-2	Endindex (Startindex + Länge) liegt außerhalb der Feldgrenzen
	-3	nicht verwendeter Fehlercode
	-4	Bereichsüberschreitung (Wert zu groß)
	-5	Feldlänge < 0
	-6	Feldlänge = 0
	-7	Zeichenfolge beginnt weder mit Zahl noch mit Vorzeichen

10.14 SIZEOF-Funktion

Die Standardfunktion "SIZEOF" dient der Ermittlung des Speicherplatzbedarfs für beliebige RCL-Variablen bzw. Typen. Insbesondere bei der Dimensionierung von Übergabepuffern, z.B. bei Kommunikation über TCP/IP-Verbindungen, ist es wichtig, den Speicherbedarf einer gegebenen Variablen (Datenpuffer) zu kennen.

Die Funktion liefert den Speicherbedarf des angegebenen Typs oder der Variablen in Bytes zurück. Die Standardfunktion "SIZEOF" kann innerhalb des Konstantendefinitionsteils verwendet werden.

Syntax der impliziten Deklaration:

```
sizeof=INTEGER: SIZEOF(type_or_variable) ;Standardfunktion
```

In obiger Syntax gilt:

- type_or_variable:
Beliebiger Typ- bzw. Variablenname.

In nachfolgender Tabelle ist der Speicherplatz der Standardtypen aufgeführt.

Datentyp	Größe [BYTES]	korespondierender Datentyp (C/C++)	Wertebereich in RCL
BINARY	4	long	[0, 1]
INTEGER	4	long	[-2147483648 ... 2147483647]
REAL	4, einfache Genauigkeit	float	3.4E +/-38 (7 digits)
CHAR	1	unsigned char	[CHR(0)...CHR(255)]
POINT	Achszahl * 4 (REAL)	typedef float POINT [Achszahl]	-
JC_POINT	Achszahl * 4 (REAL)	typedef float JC_POINT [Achszahl]	-
WC_FRAME	6 * 4 (REAL)	typedef float WC_FRAME[6]	-
TEXT	80 * 1 (CHAR)	typedef char TEXT[80]	-
INPUT <type>	4 + SIZEOF <type>	typedef struct IN<type> {<type> value; long address} IN<type>	siehe <type>
OUTPUT <type>	4 + SIZEOF <type>	typedef struct OUT<type> {<type> value; long address} OUT<type>	siehe <type>

Datentyp	Größe [BYTES]	korespondierender Datentyp (C/C++)	Wertebereich in RCL
FILE	28	-	-
BNR_FILE	28	-	-

Abb. 10-5: Speicherbedarf von RCL-Datentypen

Beispiel für die Funktion *SIZEOF*:

```

;; JC_NAMES=a1, a2, a3, a4                ;Achsnamen
;; WC_NAMES=c1, c2, c3, c4                ;Koordinaten
OUTPUT INTEGER: 401= szofReal, 402= szofPoint ;Übergabe an SPS
PROGRAM size
POINT: EndPos
BEGIN
    szofReal= SIZEOF (REAL)                ;Ausgabe: 4, Typ REAL
    szofPoint= SIZEOF (EndPos)              ;Ausgabe: 16 (4 axes * size of REAL)
PROGRAM_END

```

10.15 LIMIT_MIN, LIMIT_MAX, LIMIT_OFF-Funktionen

Mit Hilfe der Zuweisungen

Programm:

```

LIMIT_MIN= min_position
LIMIT_MAX= max_position

```

lassen sich Arbeitsraumbegrenzungen definieren.

Dabei stellen die Koordinaten der Punkte **min_position** und **max_position** die Endpunkte der Raumdiagonalen des aufgespannten Quaders dar.

Rein formal sind MIN_POSITION und MAX_POSITION normale Raumkoordinatenpunkte.

Diese beziehen sich, wie alle WC-Punkte, auf das lokale Werkstück- Koordinatensystem. Daraus ergibt sich, dass LIMIT_MIN und LIMIT_MAX in Werkstück-Koordinaten programmiert werden.

LIMIT_OFF schaltet die Überwachung ab.

10.16 JC-Standardfunktion

Die Standardfunktion "JC" liefert eine Umrechnung von Raum- in Maschinen-Koordinaten, z.B.

Programm:

```
@start_pos= JC(start_pos)
```

Als Ergebnis erhält man die Armstellung der Kinematik. Da der Ausgangspunkt ein Raumkoordinatenpunkt ist, ist das Ergebnis abhängig vom aktiven Werkstück-Koordinatensystem.



"JC" liefert stets Werte aus dem Hauptbereich, falls die Maschinen- oder Raumkoordinaten mehrdeutig sein können.

10.17 WC-Standardfunktion

Die Standardfunktion "WC" liefert eine Umrechnung von Maschinen- in Raum-Koordinaten, z.B.

RCL-Funktionen

Programm:

```
start_pos= WC(@start_pos)
```

Als Ergebnis erhält man die Koordinaten, bezogen auf das lokale Werkstück-Koordinatensystem, d.h. das Ergebnis ist abhängig vom aktiven Werkstück-Koordinatensystem.



"WC" liefert stets Werte aus dem Hauptbereich, falls die Maschinen- oder Raumkoordinaten mehrdeutig sein können.

11 RCL-Spezialfunktionen

11.1 RCL-Spezialfunktionen, Allgemeines

Mit den Spezialfunktionen werden in der Steuerung vorhandene Funktionen, für die keine RCL-Sprachelemente reserviert sind, dem RCL-Programmierer zugänglich gemacht. Spezialfunktionen stellen eine Erweiterung des RCL-Sprachumfangs dar.



Sie können in einem Programm aufgerufen werden, wenn sie vor dem Aufruf, ähnlich wie eine Variable, vereinbart wurden.

Für die IndraMotion MLC sind zur Zeit folgende Spezialfunktionen verfügbar:

SPC_FCT	Funktionskurzbeschreibung
1	iol (Positionsgenaueres Schalten digitaler Ausgänge auf der Bahn)
2	ppo (Positionsgenaueres Schalten von Prozessdaten auf der Bahn)
21	belt_kind (Bandart)
23	clock_date (Systemdatum und Uhrzeit)
24	sys_count (Systemzähler, realtime counter)
28	set_belt (Bandzähler Rücksetzwert setzen)
53	belt_area (Bandbereich bei Bandart 4)
59	Überschleifmodus abfragen
60	Überschleifmodus setzen

Abb. 11-1: RCL-Spezialfunktionen

Vereinbarung der Spezialfunktion

Die Vereinbarung einer Spezialfunktion beinhaltet Kennzahl und Namen der Funktion sowie Namen und Datentypen der Übergabeparameter. Durch die Übergabeparameter legen Sie fest, wann, wo und wie die Funktion wirksam sein soll. Die Vereinbarung muss im Deklarationsteil des Programms erfolgen.

Der Name der Spezialfunktion und die Namen der Übergabeparameter sind dabei frei wählbar.



Die Datentypen sind durch die Spezifikation der jeweiligen Spezialfunktion fest vorgegeben.

Beispiel:

SPC_FCT: x = beispiel (VALUE INTEGER: wap)

SPC_FCT: x	Funktionsnummer
beispiel	Spezialfunktionsname
VALUE INTEGER:	Datentypbezeichnung des Übergabeparameters, z.B. dezimaler Wert
wap	Name des Übergabeparameters (Platzhalter)

Abb. 11-2: Beispiel für eine Spezialfunktion

RCL-Spezialfunktionen

Aufruf der Spezialfunktion Der Aufruf im Anweisungsteil des Programms erfolgt durch die Angabe des Spezialfunktionsnamens und die Festlegung der vereinbarten Übergabeparameter.

Im Aufruf müssen der Name der Spezialfunktion sowie die Typen der Übergabeparameter in der Weise eingehalten werden, wie sie in der Vereinbarung im Programm festgelegt wurden.

11.2 RCL-Spezialfunktionen, tabellarische Auflistung

Syntax	Beschreibung
SPC_FCT:1=IOL (VALUE INTEGER: IONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: RateTime)	Positionsgenaues Schalten digitaler Ausgänge auf der Bahn Die Vereinbarung der Spezialfunktion beinhaltet die Kennzahl 1 und die Spezialfunktionsbezeichnung sowie die Namen und Typbezeichnungen der Funktionsgrößen, durch die im Aufruf der Spezialfunktion festgelegt wird, an welcher Position der entsprechenden Achse welcher Steuerwert ausgegeben werden soll.
SPC_FCT: 2=PPO (VALUE INTEGER: PPONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: RateTime)	Positionsgenaues Schalten von Prozessdaten auf der Bahn Die Vereinbarung der Spezialfunktion beinhaltet die Kennzahl 2 und die Spezialfunktionsbezeichnung sowie die Namen und Typbezeichnungen der Funktionsgrößen, durch die im Aufruf der Spezialfunktion festgelegt wird, an welcher Position der entsprechenden Achse welcher Prozesswert ausgegeben werden soll.
SPC_FCT: 21 = belt_kind (VALUE INTEGER: belt_no VALUE INTEGER: kind_belt)	Bandsynchroner Arbeitsbereich der Bandart 4 Mit dieser Spezialfunktion wird für das gewünschte Band die gewünschte Art der Bandsynchronisation gewählt werden. Folgende Synchronisationen sind z. Z. realisierbar: kind_belt = 4: Standard
SPC_FCT: 23 = clock_date (INTEGER: hours,minutes,day,month,year)	Systemuhr Mit der Spezialfunktion 23 kann von einem RoCo-Programm aus auf die Systemuhr der MLC zugegriffen werden. Es werden Tag, Monat, Jahr, Stunden und Minuten ermittelt.
SPC_FCT: 24= sys_count (INTEGER: c_count)	Systemzähler Die beim Aufruf der Spezialfunktion 24 übergebene Variable wird mit dem Wert des Systemzählers beschrieben. Der Wert der Variablen hat die Einheit Millisekunden [ms]. Das Zählraster entspricht der Clockzeit. Mit jedem Hochlauf der Steuerung wird der Echtzeitzähler wieder auf Null gesetzt.
SPC_FCT: 28= belt_set (VALUE INTEGER: belt_no VALUE REAL: reset_value)	Bandzähler setzen Mit der Spezialfunktion 28 wird den Bändern ein Rücksetzwert (Reset- Wert) zugewiesen. Mit dem RCL-Befehl "SYNC" kann der Bandzähler auf seinen Rücksetzwert gesetzt werden (Default = 0).

RCL-Spezialfunktionen

Syntax	Beschreibung
SPC_FCT: 53 = belt_area (VALUE INTEGER: belt_no JC_POINT: @Base REAL: FullLength REAL: BeginLength)	Geometrische Lage des Bandbereichs bei Bandart 4 belt_no: Nummer des Bandes, für welches der Wert gesetzt, bzw. gelesen werden soll (1..4). Bei negativer belt_no werden die Werte ausgelesen @Base: MK_PUNKT des Beginns des bandsynchronen Arbeitsbereiches FullLength: Gesamtlänge des bandsynchronen Arbeitsbereiches in Bandrichtung. Der Wert muss größer Null sein BeginLength: Spätester Beginn mit der bandsynchronen Arbeit. Der Wert muss größer Null und kleiner oder gleich der Gesamtlänge (FullLength) sein.
SPC_FCT: 59 = getBlendMode (VALUE INTEGER : bIMode)	Überschleifmodus abfragen bIMode=0: Der Überschleifradius wird auf die halbe Distanz der Wegstrecken limitiert (Default) bIMode=1: Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies möglich ist
SPC_FCT: 60 = setBlendMode (VALUE INTEGER : bIMode)	Überschleifmodus setzen bIMode=0: Der Überschleifradius wird auf die halbe Distanz der Wegstrecken limitiert (Default) bIMode=1: Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies möglich ist

Abb. 11-3: Spezialfunktionen

11.3 Positionsgenaues Schalten

11.3.1 Positionsgenaues Schalten, Allgemeines

Die MLC bietet die Möglichkeit auf einer Bahn zu schalten. Man kann damit während einer Verfahrbewegung aus einem RoCo-Programm heraus positionsgenau und mit zeitlichem Vorhalt Signale über die KinData-Schnittstelle an die SPS übergeben.

Siehe auch [Robot Control-Funktionalität, programmiert auf der SPS, Seite 231](#).

Dazu stehen zwei Spezialfunktionen zur Verfügung.

- Mit der Spezialfunktion 1 können am Interface digitale Ausgänge geschaltet werden.

Spezialfunktion 1 wird im folgenden als EA-Logik (Ein-, Ausschalt-Logik, IOL) bezeichnet.

- Mit der Spezialfunktion 2 können ganzzahlige Werte ausgegeben werden.

Spezialfunktion 2 wird im folgenden als PPO-Logik (Prozess-Parameter-Ausgabe-Logik) bezeichnet. Mit diesen Werten können Sie Ihre Techno-

RCL-Spezialfunktionen

logie steuern, z.B. bei Lackieranwendungen die Farbmenge dosieren oder die Pistole ein- oder ausschalten.

Positionsgenaue Ausgabe Positionsgenaue Ausgabe bedeutet, dass im Programm eine bestimmte Position des Tool-Center-Point (Greiferposition) festgelegt ist, an der die Ausgabe an die KinData-Schnittstelle an die SPS gesetzt wird.

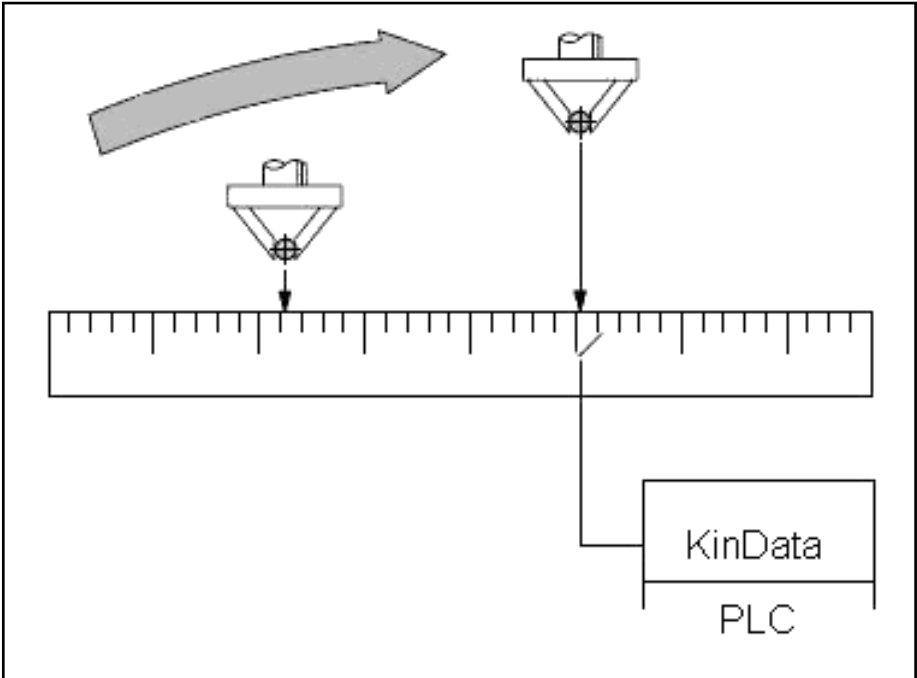


Abb. 11-4: Positionsgenaue Ausgabe an die KinData-Schnittstelle

Programmierung Die Spezialfunktionen müssen vor ihrem Aufruf, ähnlich wie Variablen, im Deklarationsteil des RoCo-Programms vereinbart werden. Es ist unbedingt darauf zu achten, dass die Parameterliste mit der nachfolgenden Deklaration exakt übereinstimmt.

11.3.2

SPC_FCT: 1, Positionsgenaues Schalten digitaler Ausgänge auf der Bahn

Die Vereinbarung der Spezialfunktion beinhaltet die Kennzahl 1 und die Spezialfunktionsbezeichnung sowie die Namen und Typbezeichnungen der Funktionsgrößen, durch die im Aufruf der Spezialfunktion festgelegt wird, an welcher Position der entsprechenden Achse welcher Steuerwert ausgegeben werden soll.

Deklaration *Programm:*

```
SPC_FCT:1=IOL(VALUE INTEGER: IONo
VALUE INTEGER: KinNo
VALUE INTEGER: CoordNo
VALUE REAL: SwitchPos
VALUE REAL: ParamValue
VALUE INTEGER: RateTime)
```

- IONo Im KinData-Interface stehen 32 Ausgänge zur Verfügung, die mit der Spezialfunktion 1 geschaltet werden können (OUTPUT BINARY 1..32).
- KinNo Nummer der Kinematik (1..16), auf die sich die Schaltposition bezieht.
- CoordNo Nummer der Koordinate innerhalb der Kinematik, auf die sich die Schaltposition bezieht.

RCL-Spezialfunktionen

SwitchPos	Position an der geschaltet, d.h. der Parameterwert ausgegeben wird.
ParamValue	Der Parameterwert wird als dezimaler Wert vorgegeben, wobei ein Wert < 0.5 als logisch 0 (low) und ein Wert ≥ 0.5 als logisch 1 (high) ausgegeben wird.
RateTime	Zeit in [ms], die zwischen der tatsächlichen Ausgabe des Parameterwertes und dem Erreichen der Schaltposition verstreicht. Der digitale Ausgang wird also bereits vor Erreichen der gewünschten Position geschaltet (Vorhaltezeit). Damit können Reaktionszeiten der Peripherie kompensiert werden.



Mit dieser Funktion können keine bandsynchrone Koordinaten verwendet werden!

Freigabesignale

Für eine korrekte Ausführung der Spezialfunktion müssen die folgenden Zustände gesetzt sein:

- Automatik,
- kein Not-Aus,
- alle Achsen sind der Kinematik mit MB_AddAxisToGroup zugeordnet,
- Im K-0-0241 darf Vorschub Halt nicht gesetzt sein.

Programm:

```
PROGRAM prog1
VAR:
SPC_FCT:1=ioout(VALUE INTEGER: chan, kin, coord
                VALUE REAL: po, va
                VALUE INTEGER: tm)
OUTPUT BINARY: 7= ToVaOn ; entspricht KinData.dwUserBits_i.6
BEGIN
ioout(7,1,3,-300,1.0,50) ; 50ms (tm) bevor die kartesische z-Achse (coord)
                        ; die Position (po) -300 erreicht hat wird auf dem Ausgabekanal (chan)
                        ; 7 der Wert (va) 1 ausgegeben
MOVE LINEAR TO (10,20,-300)
PROGRAM_END
```

11.3.3 SPC_FCT: 2, Positionsgenaues Schalten von Prozessdaten auf der Bahn

Die Vereinbarung der Spezialfunktion beinhaltet die Kennzahl 2 und die Spezialfunktionsbezeichnung sowie die Namen und Typbezeichnungen der Funktionsgrößen, durch die im Aufruf der Spezialfunktion festgelegt wird, an welcher Position der entsprechenden Achse welcher Steuerwert ausgegeben werden soll.

Deklaration *Programm:*

```
SPC_FCT:2=PP0(VALUE INTEGER: PP0No
              VALUE INTEGER: KinNo
              VALUE INTEGER: CoordNo
              VALUE REAL: SwitchPos
              VALUE REAL: ParamValue
              VALUE INTEGER: RateTime)
```

PP0No	Im KinData-Interface stehen 4 ganzzahlige Ausgänge zur Verfügung, die mit der Spezialfunktion 2 geschaltet werden können (OUTPUT INTEGER 1.4).
KinNo	Nummer der Kinematik (1..16), auf die sich die Schaltposition bezieht.

RCL-Spezialfunktionen

CoordNo	Nummer der Koordinate innerhalb der Kinematik, auf die sich die Schaltposition bezieht.
SwitchPos	Position an der geschaltet, d.h. der Parameterwert ausgegeben wird.
ParamValue	Der Parameterwert wird als dezimaler Wert vorgegeben.
RateTime	Zeit in [ms], die zwischen der tatsächlichen Ausgabe des Parameterwertes und dem Erreichen der Schaltposition verstreicht. Der digitale Ausgang wird also bereits vor Erreichen der gewünschten Position geschaltet (Vorhaltezeit). Damit können Reaktionszeiten der Peripherie kompensiert werden.



Mit dieser Funktion können keine bandsynchrone Koordinaten verwendet werden!

- Freigabesignale
- Für eine korrekte Ausführung der Spezialfunktion müssen die folgenden Zustände gesetzt sein:

 - Automatik,
 - kein Not-Aus,
 - alle Achsen sind der Kinematik mit MB_AddAxisToGroup zugeordnet,
 - Im K-0-0241 darf Vorschub Halt nicht gesetzt sein.

Programm:

```
PROGRAM prog2
VAR:
SPC_FCT:2=ppo(VALUE INTEGER: chan, kin, coord
               VALUE REAL: po, va
               VALUE INTEGER: tm)
OUTPUT INTEGER: 403=ToVaOn ; entspricht KinData.adiUserDInt_i[3]
BEGIN
  ioout(403,1,3,-300,42.0,50) ; 50ms (tm) bevor die kartesische z-Achse (coord)
                             ; die Position (po) -300 erreicht hat wird
                             ; auf dem Ausgabekanal (chan) 403 der Wert (va) 42 ausgegeben
  MOVE LINEAR TO (10,20,-310)
PROGRAM_END
```

11.4 SPC_FCT: 21, Bandart

Mit der Spezialfunktion 21 wird die Art der Bandsynchronisation gewählt. Nach dem Hochlauf der Steuerung ist die Synchronisationsart 1 aktiv.



Aktuell ist nur die Synchronisationsart 4 zur Nutzung freigegeben.

Wird die Synchronisationsart geändert, gilt diese Einstellung (für alle Prozesse) solange, bis sie erneut geändert wird oder die Steuerung neu hochläuft.

Deklaration Programm:

```
SPC_FCT:21 = belt_kind (VALUE INTEGER: belt_no
                       VALUE INTEGER: kind_belt)
```

belt_no	Nummer des Bandes (1 bis 4)
kind_belt	Bandsynchronisationsart: 4=Bandart 4 mit bandparalleler Achse und bandsynchroner PTP-Verfahrenmöglichkeit.

Beispiel Programm:

```

PROGRAM SPZ21                                ;Programmname
SPC_FCT : 21 = belt_kind(VALUE INTEGER: belt_no ;Deklaration der Spezialfunktion
                        VALUE INTEGER: kind_belt)
BEGIN
belt_kind(1,4)                                ;Anwahl der Bandart 4 für Band 1
PROGRAM_END

```

11.5 SPC_FCT: 23, DateTime

Mit der Spezialfunktion 23 kann von einem RCL-Programm aus auf die Systemuhr der MLC zugegriffen werden. Es werden Tag, Monat, Jahr, Stunden und Minuten ermittelt.

Deklaration Programm:

```

SPC_FCT:23 = DateTime(INTEGER: hour, minute,
                      day, month, year)

```

hour	Stunde
minute	Minuten
day	Tag
month	Monat
year	Jahr

Beispiel Programm:

```

PROGRAM SPZ23                                ;Programmname
VAR:
INTEGER: hr, mi, dy, mo, ye
SPC_FCT : 23 = DateTime(INTEGER: hour,
                        minute,
                        day,
                        month,
                        year)                ;Deklaration der Spezialfunktion
BEGIN
DateTime(hr, mi, dy, mo, ye)                ;liefert in hr, mi, dy mo, and ye
                                           ;das aktuelle Datum und die Uhrzeit zurück
PROGRAM_END

```

11.6 SPC_FCT: 24, SysClk

Die beim Aufruf der Spezialfunktion 24 übergebene Variable wird mit dem Wert des Systemzählers beschrieben.

Der Wert der Variablen hat die Einheit Millisekunden [ms]. Das Zählraster entspricht der MC-Zykluszeit (s. C-0-0410). Mit jedem Hochlauf der Steuerung wird der Echtzeitzähler wieder auf Null gesetzt.

Deklaration Programm:

```

SPC_FCT:24 = SysClk(INTEGER: clks)

```

clks	Anzahl motion cycles (MC) die nach dem Hochlauf vergangen sind

RCL-Spezialfunktionen

Beispiel *Programm:*

```

PROGRAM SPZ24                                     ;Programmname
VAR:
  INTEGER: clks                                   ;Deklaration der Spezialfunktion
  SPC_FCT : 24 = SysClk(INTEGER: clks)
BEGIN
  SysClk( clks )                                ;liefert in clks die Anzahl motion cycles zurück
PROGRAM_END

```

11.7 SPC_FCT: 28, Bandzähler setzen

Mit der Spezialfunktion 28 wird den Bändern ein Rücksetzwert (Reset-Wert) zugewiesen.

Mit RCL-Befehl "SYNC" kann der Bandzähler gesetzt werden.

Deklaration *Programm:*

```

SPC_FCT:28=belt_set(VALUE INTEGER: belt_no
                    VALUE REAL: reset_value)

```

belt_no	Nummer des Bandes (1 bis 4)
reset_value	Rücksetz-Wert in [mm] für den Bandzähler.

Freigabesignale *Für eine korrekte Ausführung der Spezialfunktion müssen die folgenden Zustände gesetzt sein:*

- Automatik,
- kein Not-Aus,
- alle Achsen sind der Kinematik mit MB_AddAxisToGroup zugeordnet,
- Im K-0-0241 darf Vorschub Halt nicht gesetzt sein.

Beispiel *Programm:*

```

;;INCLUDE head                                   ;head enthält Übersetzeranweisungen
PROGRAM SPC28                                     ;Programmname
SPC_FCT:28=belt_set(VALUE INTEGER: belt_no       ;Deklaration der Spezialfunktion
                    VALUE REAL: reset_value)
BEGIN
  belt_set(2,200)
PROGRAM_END

```

Der Reset-Wert bleibt bis zum nächsten Aufruf der Spezialfunktion 28 für das gleiche Band aktiv. Ist die Steuerung neu hochgelaufen, ist der Reset-Wert 0 hinterlegt.

Bei der Bandsynchronisation in einem RCL-Programm werden die internen Bandzähler ebenfalls auf die zur Zeit des Aufrufs aktiven Reset- Werte gesetzt, beispielsweise bei

Programm:

```

SYNC band1, E1=1 ; band 1 wird auf den Rücksetzwert gesetzt

```

Programm:

```

SYNC band3 ≥ 200 ; vom aktuellen Bandzählerwert von band3 wird einmalig
                  ; 200 abgezogen.
                  ; Ist der aktuelle Bandzähler < 200, wartet die Funktion
                  ; bis der Zähler auf 200 gestiegen ist und zieht dann 200 ab.

```

11.8 SPC_FCT: 53, bandsynchroner Arbeitsbereich der Bandart 4

Vor Nutzung des SPC_FCT: 53, bandsynchroner Arbeitsbereich der Kinematik muss mittels SPC_FCT: 21, die Bandart 4 festgelegt werden.

Das Arbeitsprinzip wird in nachstehender Abbildung verdeutlicht.

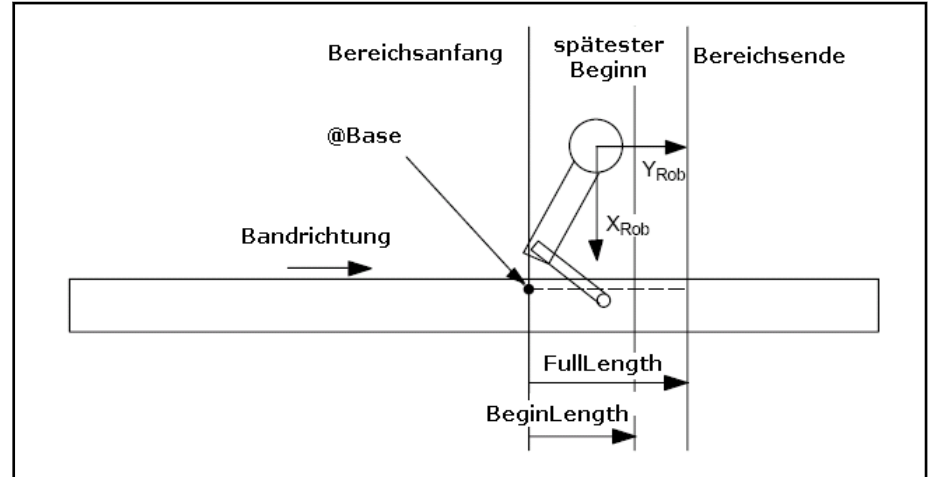


Abb. 11-5: Bandsynchroner Arbeitsbereich

Die Bandrichtung ist durch Parameter K-0-0071, Kopplungsfaktoren Band 1, (K-0-0072/Band 2... K-0-0074/Band 4) festgelegt.

Hier im Bild ist das Band um +90 Grad gegenüber dem Roboterkoordinatensystem gedreht, also Koord1 = 0, Koord2 = 1, Koord3 = 0. (Der Wert für eine Koordinate entspricht dem Kosinuswert ($\cos a$) des Winkels, der zwischen dem Band und der entsprechenden Koordinaten liegt).

Die Startebene geht durch den Maschinenkoordinatenpunkt @Base und steht senkrecht zur Bandrichtung.

Der Bereichsanfang und das Bereichsende liegen parallel zur Startebene.

Bei der ersten Verfahrbewegung im bandsynchronen Abschnitt der Bandart 4 wird überprüft, ob sich der Endpunkt der Verfahrbewegung zwischen der Startebene und der Beginnsgrenze befindet. Hat er die Startebene noch nicht erreicht, so wird die Ausführung der Verfahrbewegung so lange verzögert, bis der Endpunkt die Startebene passiert hat. Hat der Endpunkt der ersten bandsynchronen Verfahrbewegung die Beginnsgrenze schon überschritten, erfolgt eine Fehlermeldung (F12C00B4, Bandsync.: 1. BS-Verfahrsatz zu spät aktiviert).

Bei der Ausführung des bandsynchronen Abschnittes der Bandart 4 wird der Tool Center Point (TCP) auf das Bereichsende überwacht. Wird festgestellt, dass der TCP das Bereichsende überschreiten würde oder schon überschritten hat, erfolgt gleichfalls eine Fehlermeldung (F12C00AE, Bandsync.: Verfahrsatz zu spät aktiviert).

Im letzten bandsynchronen Verfahrsatz (Absynchronisieren und Fahrt zur ortsfesten Position) wird das Bereichsende nicht überwacht.

Deklaration Programm:

```
SPC_FCT:53 = belt_area (VALUE INTEGER : belt_no
                        JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)
```


RCL-Spezialfunktionen

```

SPC_FCT : 21 = belt_kind (VALUE INTEGER: belt_no
                           VALUE INTEGER: kind_belt)
SPC_FCT : 53 = belt_area (VALUE INTEGER: belt_no
                           JC_POINT: @Base
                           REAL: rTotalLength
                           REAL: rBeginLength)
BEGIN
    belt_kind ( C_BELT_NO_1, C_BK_PTP ) ; Festlegung Bandart
    belt_area ( C_BELT_NO_1, @Base, rTotalLength, rBeginLength ) ; Festlegung bandsynchroner Arbeitsbereich
    rTotalLength = 560
    rBeginLength = 400
    Base = (280, 140, -600, 0, 0, 0)
    @Base = JC(Base)
    belt_area ( C_BELT_NO_1, @Base, rTotalLength, rBeginLength)

```

Mehrere Bänder innerhalb einer Kinematik

Hat eine Kinematik mehrere Bänder, kann die Spezialfunktion mit gleicher Deklaration für jedes Band der Kinematik benutzt werden. Die Unterscheidung erfolgt über "belt_no".

Mehrere Kinematiken



Ein RCL-Programm kann genau eine Kinematik bewegen. In der RoCo wird folgende Philosophie vertreten:

1. das RCL-Programm ist für die Bewegung des Roboters zuständig,
2. das PLC-Programm erledigt alle anderen Aufgaben (Kommunikation, Koordination, Anlagen-/Prozesssteuerung, ...).

Variablen vom Typ "JC_POINT" sind kinematikbezogen. Soll diese Spezialfunktion für mehrere Kinematiken benutzt werden, muss sie unter verschiedenen Namen deklariert werden, z.B. "belt_area1".

Die Nummer (53 = ...) muss für jede Kinematik mit kinematikbezogenem Typ "JC_POINT", z.B. Robi_1.JC_POINT, gleich sein.

Beispiel:

Programm:

```

SPC_FCT :53=belt_area1 (VALUE INTEGER: belt_no
                        Robi_1.JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)
SPC_FCT :53=belt_area2 (VALUE INTEGER: belt_no
                        Robi_2.JC_POINT: @Base
                        REAL: FullLength
                        REAL: BeginLength)

```

11.9 RCL-Programmierung "Überschleifmodus"

11.9.1 SPC_FCT: 60, Überschleifmodus setzen

Mit Hilfe der SPC_FCT 60 kann im RCL-Programm der Überschleifmodus angewählt werden. Dieser Modus gilt für alle nachfolgenden Befehle die in den Satzpuffer gestellt werden.

Programm:

```
SPC_FCT: 60 = setBlendMode(VALUE INTEGER : blMode)
```

RCL-Spezialfunktionen

- blMode=0:** Der Überschleifradius wird auf die halbe Distanz der Wegstrecken limitiert (Default)
Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies möglich ist
- blMode=1:** Der Überschleifradius wird auf die halbe Distanz der Wegstrecken limitiert (Default)
Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies möglich ist

Der programmierte Überschleifradius R wird in beiden Modi entsprechend berücksichtigt.

Im Falle **blMode = 0**

ergibt sich der wirksame Blendradius R_w aus dem Minimum des programmierten Radius R , die Hälfte der Wegstrecke s_1 und die Hälfte der Wegstrecke s_2 .

$$R_w = \min \left(R, \frac{1}{2} s_1, \frac{1}{2} s_2 \right)$$

Abb. 11-6: Wirkradius beim Überschleifen mit Überschleifmodus 0

Im Falle **blMode=1**

kann sich der wirksame Blendradius R_w aus dem Minimum des programmierten Radius R , die gesamte Wegstrecke s_1 und die gesamte Wegstrecke s_2 ergeben, wenn dies möglich ist.

$$R_w = (R, n_1 \cdot s_1, n_2 \cdot s_2)$$

Abb. 11-7: Wirkradius beim Überschleifen mit Überschleifmodus 1

$n_{(1,2)} = 1$ bzw. $1/2$

s_1 = Wegstrecke 1

s_2 = Wegstrecke 2

$n_{(1,2)}$ wird von der Steuerung automatisch anhand der programmierten Verfahrbefehle gewählt.

Ist der Verfahrbefehl ein "MOVE TO"-Befehl, wird für diese Wegstrecke $n = 1$.

Ist der Verfahrbefehl ein "MOVE VIA"-Befehl, hängt n vom vorhergehenden Befehl ab.

11.9.2 SPC_FCT: 59, Überschleifmodus abfragen

Mit Hilfe der SPC_FCT 59 kann im RCL-Programm der in der Satzvorbereitung gesetzte Überschleifmodus abgefragt werden.

Programm:

```
SPC_FCT: 59 = getBlendMode(VALUE INTEGER : blMode)
```

- blMode=0:** Der Überschleifradius wird derzeit auf die halbe Distanz der Wegstrecken limitiert (Default)
Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies geometrisch möglich ist.
- blMode=1:** Der Überschleifradius wird derzeit auf die halbe Distanz der Wegstrecken limitiert (Default)
Der Überschleifradius kann die gesamte Distanz der kürzeren Wegstrecke erreichen, wenn dies geometrisch möglich ist.

Beispiel:

```
;;COMPILER = RCL3
;;INT = LINEAR
;;KINEMATICS: ( 1= Kin1)
;;Kin1.JC_NAMES = vAchse1, vAchse2, vAchse3, vAchse4, vAchse5, vAchse6
;;Kin1.WC_NAMES = kx, ky, kz, kc, kb1, kb2

PROGRAM halfFull

VAR:
SPC_FCT : 60 = setBlendMode (VALUE INTEGER : blMode)
INTEGER : blMode
POINT : P1, P2, P3, P4
BEGIN

V=200 ; 0.2 m/s
A=1000 ; 1 m/s^2
R=1000 ; 1 meter = control calculates the max. blend radius
; because the moved distances will be lower then 1 meter

BLOCK_SLOPE ; has no effect because this is the default
setBlendMode( 0 ) ; has no effect because this is the default

; points for the rectangle
P1 = ( -300, 0, -650, 0, 0, 0)
P2 = ( -300, 200, -650, 0, 0, 0)
P3 = ( 300, 200, -650, 0, 0, 0)
P4 = ( 300, 0, -650, 0, 0, 0)

; no blending
MOVE LINEAR TO P1
MOVE LINEAR TO P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with half distances

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with half distances

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR VIA P3
MOVE LINEAR TO P4

; blMode = 1
setBlendMode( 1 )

; no blending

MOVE LINEAR TO P1
MOVE LINEAR TO P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with complete distances of P1P2 and P2P3

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR TO P3
MOVE LINEAR TO P4

; blending with complete distance of P1P2
; and half distance of P2P3
; blending with half distance of P2P3
; and complete distance of P3P4

MOVE LINEAR TO P1
MOVE LINEAR VIA P2
MOVE LINEAR VIA P3
MOVE LINEAR TO P4

PROGRAM_END
```


12 RCL-Bewegungsanweisungen

12.1 Allgemeines

Bewegungen des Roboters werden durch Bewegungsanweisungen ausgelöst. Bewegungsanweisungen beschreiben die Bewegung des Roboters von einer aktuellen Position und Orientierung zu einem Zielpunkt.

Wir unterscheiden zwischen:

- Direkte Bewegungsanweisungen (Angabe von **V**, **A** etc.)
- Bewegungsbeeinflussende Bewegungsanweisungen

12.2 Direkte Bewegungsanweisungen

12.2.1 Überblick

Eine direkte Bewegungsanweisung setzt sich aus den folgenden Einzelanweisungen zusammen:

- Bewegungsbefehl
- Interpolationsart
- Zusatzinformationen
- Abbruchbedingung (opt.)
- Anfahrtart
- Zielposition

Folgende Tabelle stellt eine Übersicht über die möglichen Eingaben dar. Die Kombinationsmöglichkeiten ergeben sich aus den einzelnen Beschreibungen im weiteren Verlauf.

Bewegungsanweisungen				
Bewegungs-befehl	Interpolations-art	Zusatzinfor-mationen	Abbruchbedin-gung	Anfahrtart
MOVE	LINEAR	V	UNTIL	TO
MOVE_REL	CIRCULAR	A		VIA
	PTP	J		APPROX
		R		EXACT
		VFACTOR		
		AFACTOR		
		DFACTOR		
		V_PTP		
		R_PTP		

Abb. 12-1: Überblick zu Bewegungsanweisungen

Die Programmierung einer Abbruchbedingung ist optional.

Anfahrtart und Zielposition müssen zu jedem Bewegungsbefehl mit angegeben werden.

RCL-Bewegungsanweisungen

Anweisungsbezogene Informationen

Über den WITH-Befehl werden die Zusatzinformationen **V**, **A**, **J** und **R** ausschließlich anweisungsbezogen definiert. Die Definition der Interpolationsart erfolgt ohne den WITH-Befehl direkt nach dem Bewegungsbefehl.

Für die weiteren Bewegungsbefehle sind diese Angaben dann nicht mehr gültig.

Es ist nicht möglich **AFACTOR**, **VFACTOR** und **DFACTOR** anweisungsbezogen zu programmieren.

Anweisungsübergreifende Informationen

Für den Fall, dass keine anweisungsbezogenen Zusatzinformationen mit dem Bewegungsbefehl kommandiert werden, sind die anweisungsübergreifenden Befehle gültig. Hierzu zählen sämtliche Zusatzinformationen (V, A, J, R etc.) und die Interpolationsart (LINEAR, CIRCULAR und PTP).

Zu Anfang sind jeweils die Defaultwerte gültig. Sobald diese definiert sind gelten sie für die folgenden Bewegungsbefehle.



Werden keine Zusatzinformationen zum Bewegungsbefehl angegeben, gelten die anweisungsübergreifenden Zusatzbefehle.

12.2.2 Bewegungsbefehle

Die Steuerung kennt folgende Bewegungsbefehle:

- MOVE
- MOVE_REL

MOVE

Syntax:

```
MOVE[Interpolationsart][Zusatzinformation][Abbruchbedingung]
[TO]Zielpunkt[VIA Zielpunkt[TO Zielpunkt]
```

Die Steuerung interpretiert alle nach MOVE programmierten Positionsangaben als Absolutmaße. Die Koordinatenwerte beziehen sich auf den Nullpunkt des Raum- oder Maschinenkoordinatensystems.



Der Bewegungsbefehl MOVE kann mit allen Interpolationsarten kombiniert werden.

Programm:

```
target=(10,10)
MOVE TO target
```

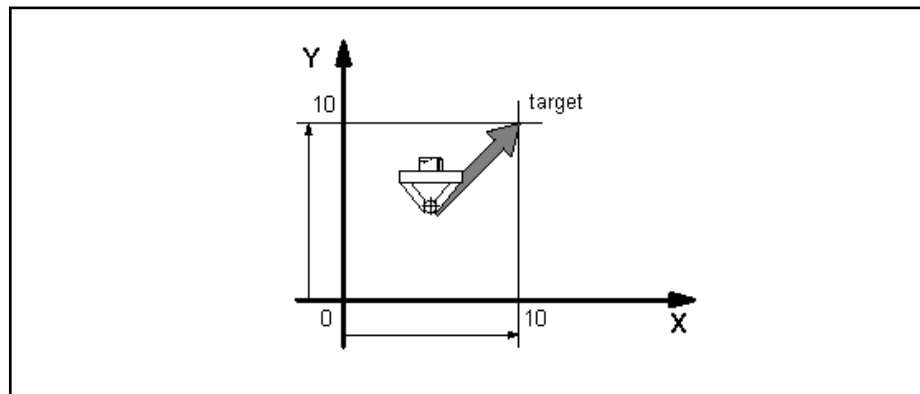


Abb. 12-2: MOVE-Befehl

RCL-Bewegungsanweisungen

Innerhalb des Fahrbefehles entscheiden Zusatzangaben darüber, ob der Roboter die programmierten Punkte genau - innerhalb der festgelegten Toleranz- anfährt, oder ob er nur an den Punkten - ohne Halt - vorbeifährt.

Beim Fahren im Absolutmaß mit MOVE sind dies die Angaben TO und VIA.

MOVE VIA ... (Überschleifen)

Der Roboter fährt an den Positionen ohne Zwischenhalt vorbei. Siehe "R" und "Überschleifen".

Programm:

`MOVE VIA pos1, pos2, pos3`

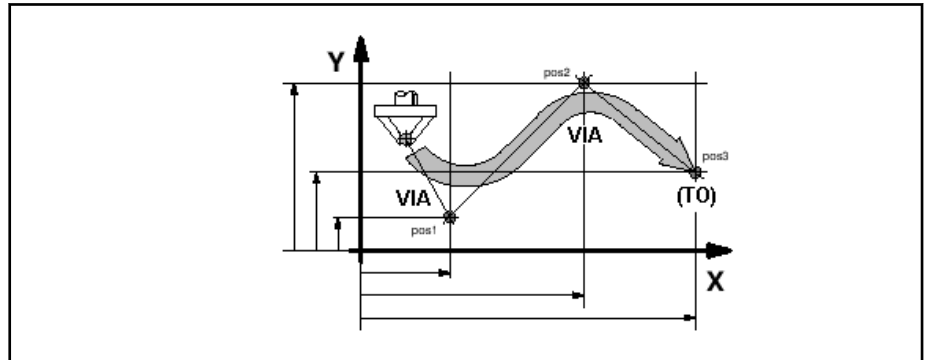


Abb. 12-3: MOVE VIA ... (Überschleifen)

MOVE TO ...

Der Roboter fährt die Positionen nacheinander mit Zwischenhalt an. In diesem Fall wird "TO" als Standardzusatz gewählt.

Programm:

`MOVE TO pos1, pos2, pos3`

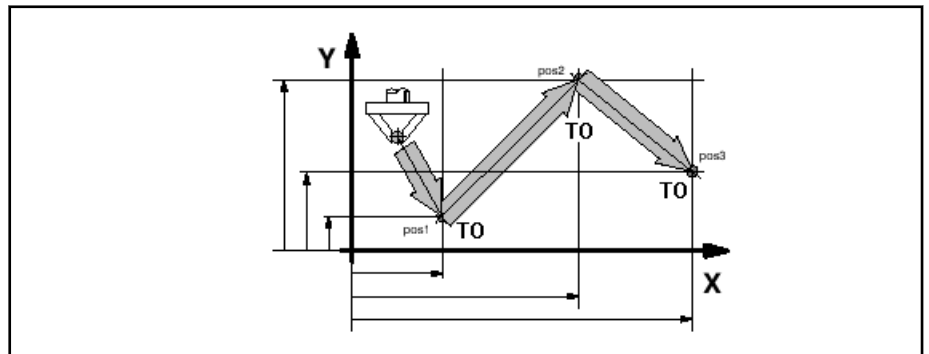


Abb. 12-4: MOVE TO ...

Eine Verbindung von VIA und TO innerhalb eines Fahrbefehles ist möglich.

MOVE VIA ... TO ...

Der Roboter fährt an der Position 1 ohne Zwischenhalt vorbei und fährt die Position 2 und 3 nacheinander mit Zwischenhalt an.

Programm:

`MOVE VIA pos1 TO pos2, pos3`

RCL-Bewegungsanweisungen

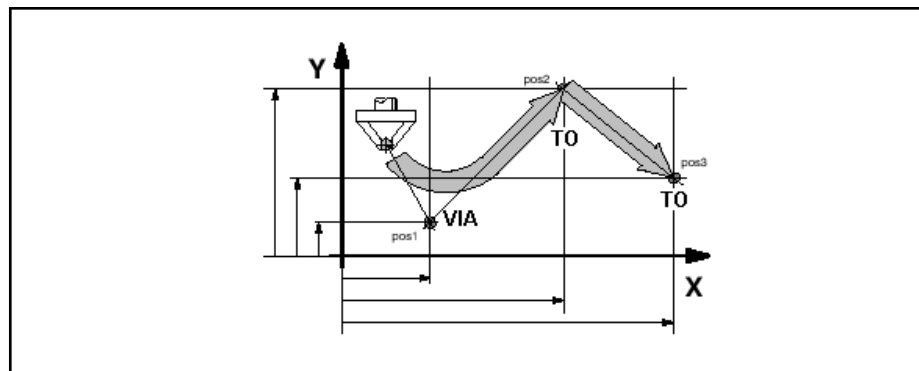


Abb. 12-5: MOVE VIA ... TO ...

MOVE_REL

Syntax:

```
MOVE_REL[Interpolationsart][Zusatzinformationen][Abbruchbedingung]
([EXACT]Zielpunkt|APPROX Zielpunkt[EXACT Zielpunkt] )
```

Die Steuerung interpretiert alle nach MOVE_REL programmierten Positionsangaben als Kettenmaße.

Die Koordinatenwerte sind hier Distanzangaben im jeweiligen Koordinatensystem und beziehen sich auf die letzte kommandierte Position des Roboters.



Der Bewegungsbefehl MOVE_REL kann mit allen Interpolationsarten kombiniert werden.

Programm:

```
target=(10,10)
MOVE_REL target
```

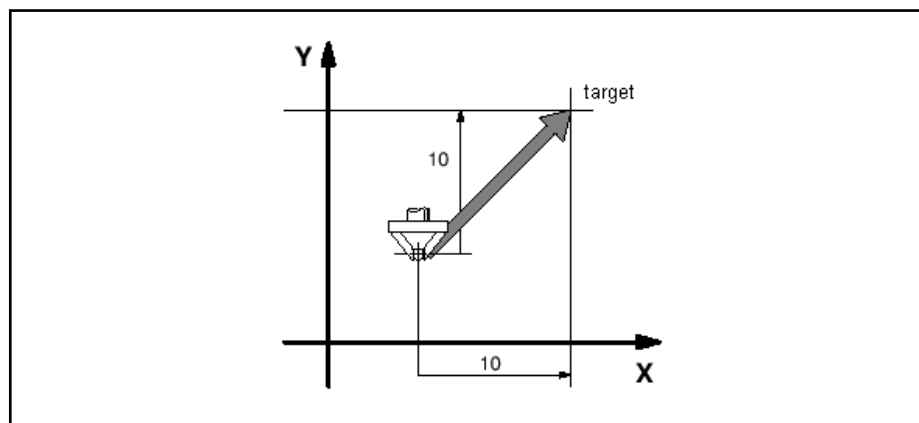


Abb. 12-6: MOVE_REL

Beim Fahren im Kettenmaß mit MOVE_REL entscheiden die Wörter "APPROX" und "EXACT" über Genau-Halt oder Vorbeifahren.

MOVE_REL APPROX...

Der Roboter fährt an den im Kettenmaß definierten Positionen ohne Zwischenhalt vorbei. Siehe "R" und "Überschleifen".

RCL-Bewegungsanweisungen

Programm:

`MOVE_REL APPROX dis1,dis2,dis3`

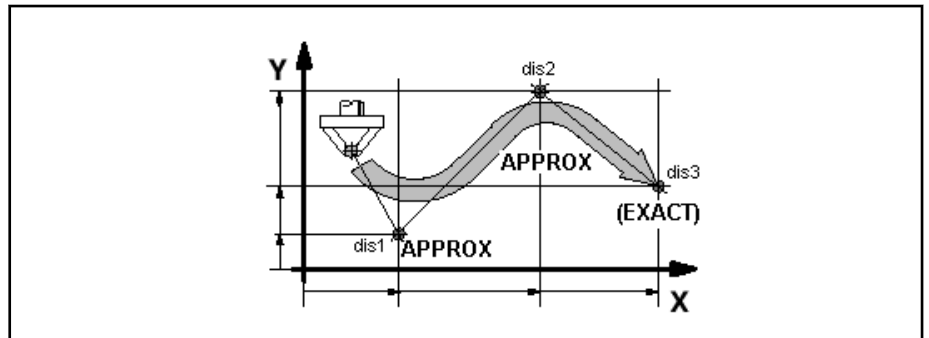


Abb. 12-7: *MOVE_REL APPROX...*

MOVE_REL EXACT...

Der Roboter fährt die im Kettenmaß definierten Positionen nacheinander mit Zwischenhalt an. Das Wort "EXACT" kann bei der Programmierung des Fahrbefehles entfallen. Die Steuerung generiert den Befehl selbstständig.

Programm:

`MOVE_REL EXACT dis1,dis2,dis3`

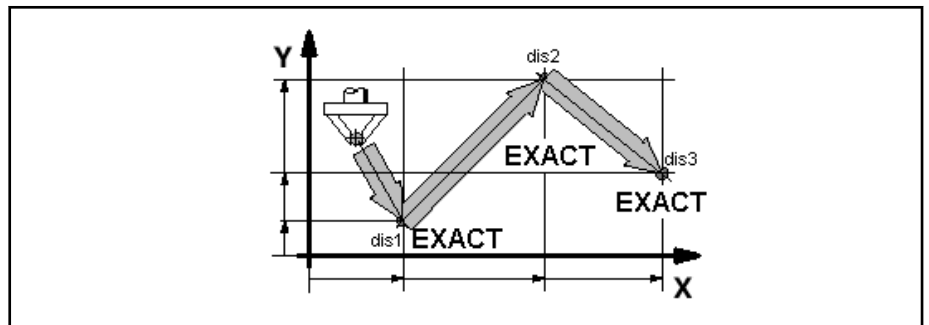


Abb. 12-8: *MOVE_REL EXACT...*

Eine Verbindung von "APPROX" und "EXACT" innerhalb des Fahrbefehles ist möglich.

MOVE_REL APPROX ... EXACT ...

Der Roboter fährt ohne Zwischenhalt an der ersten Position vorbei und fährt die nächsten Positionen 2 und 3 nacheinander mit Zwischenhalt an.

Programm:

`MOVE_REL APPROX dis1 EXACT dis2,dis3`

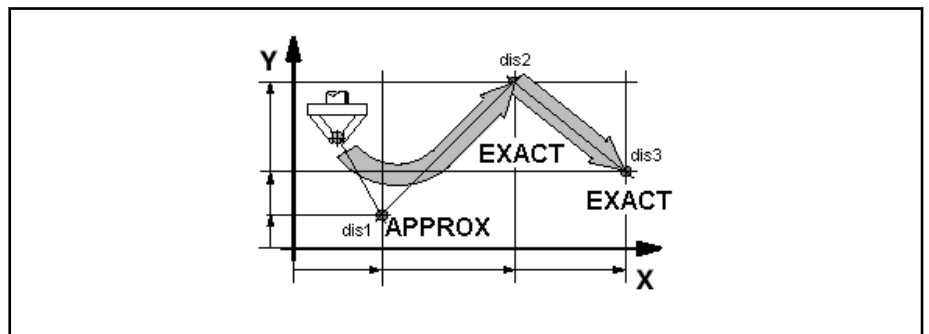


Abb. 12-9: *MOVE_REL APPROX ... EXACT ...*

RCL-Bewegungsanweisungen

Zielpunktbezeichnungen

Die Bezeichnungen der Punktvariablen sind frei wählbar, siehe [Punktvariablen, Seite 59](#). So können z.B. den Palettenpunkten die Namen "palette1", "palette2" ... zugeordnet werden.

Der Einfachheit halber werden in den Beispielen auf den nächsten Seiten Punktangaben im Absolutmaß mit "pos" bezeichnet. z.B. "MOVE TO pos"

Punktangaben im Kettenmaß werden mit "dis" (Distanzangaben) bezeichnet, z. B. "MOVE_REL dis".

12.2.3 Interpolationsart

Die MLC Robot Control unterstützt das Verfahren einer Kinematik in drei Interpolationsarten:

- LINEAR (Gerade im Raum)
- CIRCULAR (Kreisbahn im Raum)
- PTP (synchron Punkt zu Punkt)

Die Interpolationsarten beziehen sich auf die Interpolation im Raum. Abhängig von der Kinematik des Roboters folgen die Achsbewegungen entsprechend nach Rückwärtstransformation (von Raumkoordinaten nach Achskoordinaten).



Orientierungsachsen einer Kinematik werden nicht transformiert. Bei der Interpolation werden diese jedoch vollsynchron (identische Rampenzeiten) mit angesteuert.

Interpolationsart LINEAR

Der Roboter fährt den Zielpunkt auf einer Geraden an. Die Gerade ist geometrisch durch zwei Punkte definiert. Da der Steuerung die momentane Position P1 des Roboters bekannt ist, genügt die Angabe eines Zielpunktes P2.

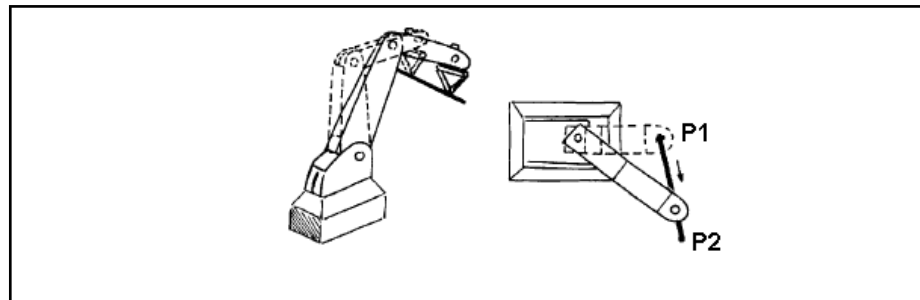


Abb. 12-10: Interpolationsart LINEAR

Interpolationsart CIRCULAR

Der Roboter fährt den Zielpunkt auf einer Kreisbahn im Raum an.

Der Kreis ist geometrisch definiert durch drei Punkte.

Zusätzlich zum Zielpunkt P2 müssen Sie also noch einen Zwischenpunkt "PZ" angeben damit die Steuerung die Kreisbahn eindeutig berechnen kann.

Der Punkt P1 ist der letzte angefahren Punkt und somit der Steuerung bekannt. Der Zwischenpunkt PZ ist ein Punkt auf dem Kreisbogen, der vom Roboter überfahren wird.

RCL-Bewegungsanweisungen

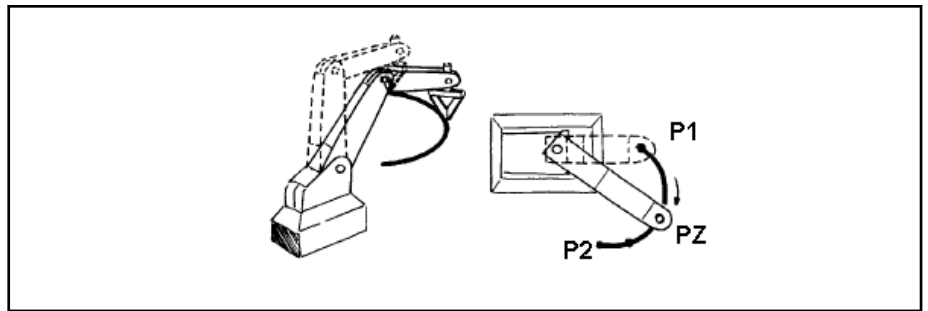


Abb. 12-11: Interpolationsart CIRCULAR

Interpolationsart PTP

PTP: Point-To-Point (Punkt-zu-Punkt)

Die Steuerung berechnet die Bewegung aller Achsen so, dass sie gleichzeitig die Bewegung beginnen und beenden. Dabei gelten für alle dieselben Rampenzeiten. Dies wird allgemein auch als Vollsynchron-PTP bezeichnet. Abhängig vom Hebelverhältnis der Roboterarme und der Punkte P1 und P2 kann dies einen nicht weiter definierten Raumweg ergeben.

Beim PTP-Verfahren genügt die Angabe eines Zielpunktes.

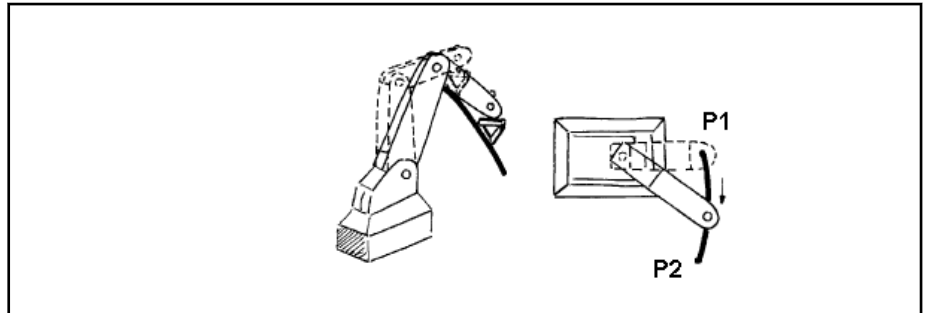


Abb. 12-12: Interpolationsart PTP

Anweisungsbezogene Interpolationsart

Soll eine bestimmte Interpolationsart nur für eine Bewegungsanweisung Gültigkeit haben, programmiert man die Interpolationsart in der Bewegungsanweisung.

Die Interpolationsart steht innerhalb des Fahrbefehles, direkt im Anschluss an "MOVE" oder "MOVE_REL".



Innerhalb einer Bewegungsfunktion darf nur eine Interpolationsart stehen!

Beispiel LINEAR

Programm:

```
MOVE LINEAR TO pos1
MOVE_REL LINEAR APPROX dis1 EXACT dis2
```

RCL-Bewegungsanweisungen

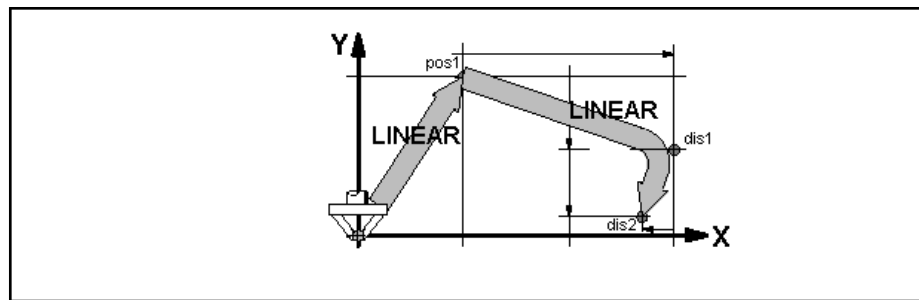


Abb. 12-13: Beispiel LINEAR

Beispiel CIRCULAR

Die für CIRCULAR (Kreis) notwendigen Punktangaben, wie Zwischenpunkt "pz" bzw. "dz", Endposition "pos1" bzw. dis1, müssen durch ein Komma getrennt in Klammern geschrieben werden.

Programm:

```
MOVE CIRCULAR TO (pz, pos1)
```

```
MOVE_REL CIRCULAR EXACT (dz, dis1)
```

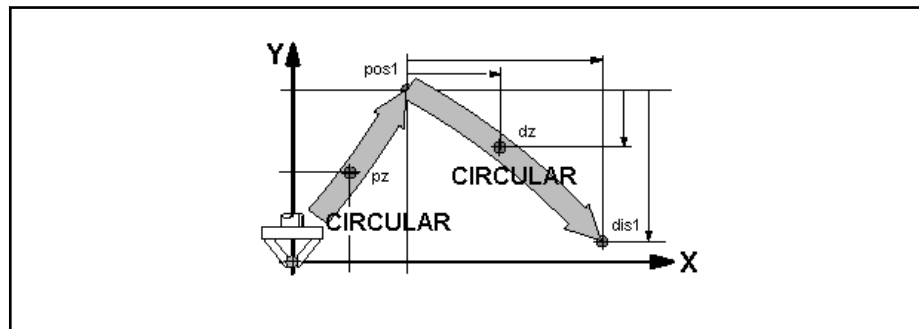


Abb. 12-14: Beispiel CIRCULAR

Beispiel PTP

Programm:

```
MOVE PTP TO pos1
```

```
MOVE_REL PTP EXACT dis1
```

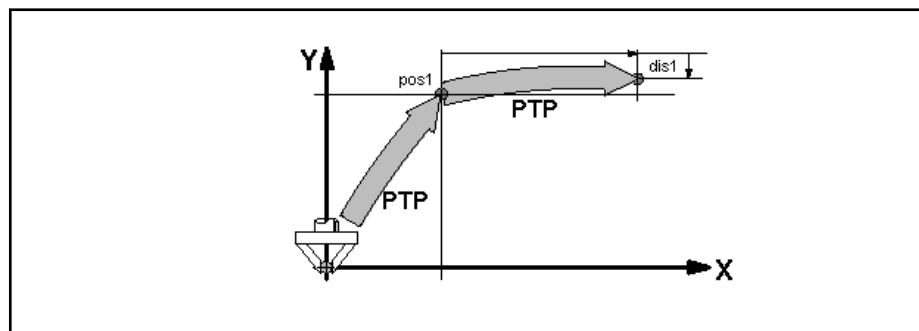


Abb. 12-15: Beispiel PTP

Anweisungsübergreifende Interpolationsart

Soll eine Interpolationsart für mehrere Bewegungsanweisungen gelten, kann die Interpolationsart anweisungsübergreifend vorgegeben werden. Die anweisungsübergreifende Interpolationsart wird mit der folgenden Übersetzeranweisung angegeben.

RCL-Bewegungsanweisungen

Syntax:

```
;;INT= [LINEAR, CIRCULAR, PTP] ;Interpolation mode, all kinematics
```

Diese ist standardmäßig bereits im RCL-Programmkopf integriert. Die so festgelegte Interpolationsart gilt dann für alle nachfolgenden Bewegungsanweisungen, die keine spezifischen Interpolationsvorgaben enthalten.

Beispiel:

```
;;INT=LINEAR
MOVE pos1 ;Die Position pos1 und den über dis1 definierten Punkt
           ;fährt der Roboter jeweils auf einer Geraden an.
MOVE_REL dis1
MOVE CIRCULAR(pz,pos2) ;Der Punkt pos2 dagegen wird auf einer Kreisbahn angefahren.
```

Die anweisungsübergreifende Interpolationsart hat solange Gültigkeit, bis sie durch eine andere Interpolationsart ersetzt wird.

Beispiel:

```
;;INT=LINEAR
MOVE pos1 ;Der Roboter fährt die Achsen der Kinematik 1 auf die Positionen
MOVE pos2 ;pos1 und pos2 jeweils auf einer Geraden an
;;INT=CIRCULAR
MOVE (pz1,pos3) ;Die Position pos3 und pos4 und alle weiteren werden nach der
MOVE (pz2,pos4) ;Interpolationsvorgabe CIRCULAR auf einer Kreisbahn angefahren
```



Bei anweisungsübergreifender Vorgabe einer CIRCULAR-Interpolation müssen die Punkangaben innerhalb der Bewegungsanweisung für die die Interpolationsvorgabe gültige Punktepaare sein.



Fehlt die Angabe einer Interpolationsart und ist keine anweisungsübergreifende Vorgabe einer Interpolationsart erfolgt, setzt die Steuerung selbsttätig PTP ein.



Sind mehrere anweisungsübergreifende Interpolationsarten programmiert, erfolgt die Vorgabe der Interpolationsart in aufsteigender Zeilenfolge und nicht dem Programmverlauf (Unterprogramm, Sprünge usw.) entsprechend.

Beispiel:

Die Position "pos2" wird mit PTP angefahren, obwohl die Übersetzer-Anweisung ";;INT = PTP" im Programmverlauf übersprungen wurde.

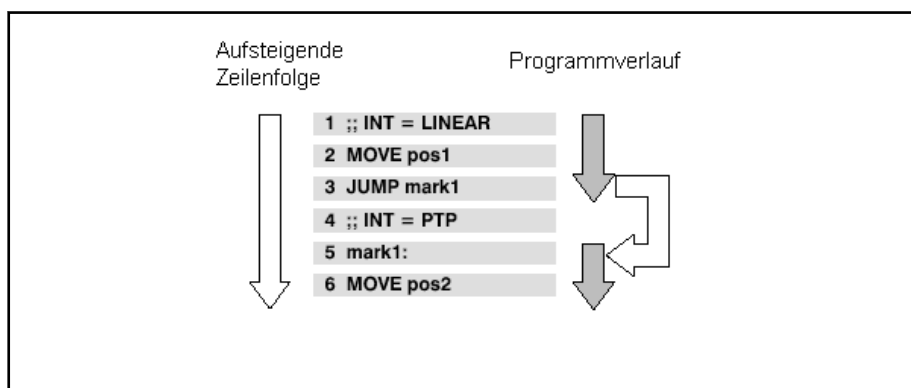


Abb. 12-16: Aufsteigende Zeilenfolge kontra Programmverlauf

RCL-Bewegungsanweisungen

12.2.4 Wirkungsweise bewegungsbeeinflussender RCL-Anweisungen und Parameter

Interpolationsart: LINEAR/ CIRCULAR (RCL-Syntax)

	Erläuterung	Berechnung	Wirkung
V	Bahngeschwindigkeit		SV
VFACTOR	Bahngeschwindigkeits-Override	$V_{\text{wirk}} = V * VFACTOR$	SV
VFIX	Deaktiviert K-0-0240, RoCoFeedrate-Override		SV
A	Bahnbeschleunigung		SV
J	Bahnrudd		SV
AFACTOR	Bahnbeschleunigung-Override	$A_{\text{wirk}} = A * AFACTOR$	SV
DFACTOR	Bahnbremsverzögerung-Override	$D_{\text{wirk}} = A * DFACTOR$	SV
AFIX	Deaktiviert K-0-0242, RoCoAcceleration-Override		SV
R	Überschleifradius		SV

SV Satzvorbereitung

Abb. 12-17: Wirkungsweise bewegungsbeeinflussender RCL-Anweisungen

K-Parameter (SPS/ Parametereditor)

	Erläuterung	Berechnung	Wirkung
K-0-0240	RoCoFeedrate-Override	$V_{\text{wirk}} = V_{\text{wirk}} * (K-0-0240 / 100)$	SA
K-0-0242	RoCoAcceleration-Override	$A_{\text{wirk}} = A_{\text{wirk}} * (K-0-0242 / 100)$	SA
K-0-0243	RoCoDeceleration-Override	$D_{\text{wirk}} = D_{\text{wirk}} * (K-0-0243 / 100)$	SA
K-0-0014	Bahnbeschleunigung, Änderungszeit	$J_{\text{wirk}} = A_{\text{wirk}} / K-0-0-0014$ (Beschleunigungs-Dreiecks-Betrieb) $J_{\text{wirk}} = A_{\text{wirk}} / t$ für $t \leq K-0-0-0014$ (Beschleunigungs-Trapez-Betrieb)	SA
K-0-0015	Bahnbremsverzögerung, Änderungszeit	$J_{\text{wirk}} = D_{\text{wirk}} / K-0-0-0014$ (Beschleunigungs-Dreiecks-Betrieb) $J_{\text{wirk}} = D_{\text{wirk}} / t$ für $t \leq K-0-0-0014$ (Beschleunigungs-Trapez-Betrieb)	SA

SA Satzausführung, Satzabarbeitung

Abb. 12-18: Wirkungsweise bewegungsbeeinflussender Parameter

RCL-Bewegungsanweisungen

Interpolationsart: PTP (RCL-Syntax)

	Erläuterung	Berechnung	Wirkung
V_PTP	dezimaler Faktor für den kleineren Wert in A-0-0032/ A-0-0033 der an der Bewegung beteiligten langsamsten Achse	$V\_PTP_{\text{wirk}} = (0,8 * A-0-0032/33) * V\_PTP$	SV
AFACTOR	dezimaler Faktor für A-0-0034, Beschleunigungs-Grenzwert bipolar der an der Bewegung beteiligten langsamsten Achse	$A\_PTP_{\text{wirk}} = (0,8 * A-0-0034) * AFAC-TOR$	SV
R_PTP	dezimaler Faktor für A-0-0230, Überschleifdistanz PTP	$R\_PTP_{\text{wirk}} = A-0-0230 * R\_PTP$	SV

SV Satzvorbereitung
Abb. 12-19: Wirkungsweise bewegungsbeeinflussender RCL-Anweisungen



A-0-0032/ A-0-0033/ A-0-0034: Die Grenzwerte der Achse können in P2 und BB modifiziert werden.

Die Änderung der Werte wird erst mit der Phasenschaltung von P2 nach BB wirksam.

12.2.5 Zusatzinformationen

Zusatzinformationen, Allgemeines

Neben der Position des Zielpunktes und der Bahn müssen Sie der Steuerung mitteilen, wie schnell oder in welcher Zeit der Roboter den Zielpunkt anfahren soll.

Die Steuerung benötigt also Angaben über die Zeitdauer der Bewegung oder die Bahngeschwindigkeit des Roboters.

Wie schnell der Roboter diese Bahngeschwindigkeit erreichen soll, können Sie über die Beschleunigung vorgeben.

Zusatzinformationen Geschwindigkeit

Die Verfahrgeschwindigkeit wird bei LINEAR und CIRCULAR Interpolation in mm/s programmiert. Bei PTP Interpolation erfolgt die Programmierung als Faktor der maximalen Achsgeschwindigkeit.



Wenn Sie für die Geschwindigkeit keine Angaben programmieren, fährt der Roboter mit den Standard-Geschwindigkeitswerten. Diese sind 10 mm/s bzw 10 inch/s (siehe K-0-0054) bei Bahninterpolation bzw. 10 % der maximalen Achsgeschwindigkeit bei Synchron-PTP-Interpolation.

Die Geschwindigkeit V_PTP wird bei der Synchron-PTP-Interpolation als dezimaler Faktor der maximalen Achsgeschwindigkeit angegeben.

Eine Angabe in Prozent ist ebenfalls zulässig, d.h. V_PTP = 80 % ist gleichbedeutend mit V_PTP = 0.8.

RCL-Bewegungsanweisungen

V

Bezeichnung	V
Maßeinheit	mm/s bzw. inch/s (siehe K-0-0054)
Eingabebereich	1 bis 99999
Einschaltzustand	25

Abb. 12-20: Geschwindigkeit bei LINEAR und CIRCULAR

V_PTP

Bezeichnung	V_PTP
Maßeinheit	Dezimaler Faktor
Eingabebereich	0.0001 bis 9.9999 0.01% bis 999.99%
Einschaltzustand	0.1

Abb. 12-21: Geschwindigkeit bei PTP

Bewegungsoptimierung bei Orientierungsachsen

Der Interpolator errechnet im Vorfeld die bei Einhaltung der Pfadgeschwindigkeit resultierenden Beschleunigungen und Geschwindigkeiten der Orientierungsachsen.

Werden diese Grenzwerte verletzt, reduziert sich entsprechend die resultierende Pfadgeschwindigkeit bzw. -beschleunigung. Dies gilt auch für Linear-Interpolationen.



Die Bewegungsvorgabe erfolgt stets mit Einhaltung der Grenzwerte aller Orientierungsachsen!

Möglichkeiten der Programmierung

Die Programmierung der Geschwindigkeit kann erfolgen als

- anweisungsübergreifende Geschwindigkeitsvorgabe,
- anweisungsbezogene Geschwindigkeitsvorgabe,
- anweisungsbezogene Zeitvorgabe,

Anweisungsübergreifende Geschwindigkeitsvorgabe

Bleibt die Bahngeschwindigkeit für das ganze Programm oder für einen größeren Abschnitt gleich, ist es sinnvoll, die Geschwindigkeit anweisungsübergreifend vorzugeben.

Der Geschwindigkeitswert hat solange Gültigkeit, bis er durch eine weitere anweisungsübergreifende Geschwindigkeitsvorgabe geändert wird.

Beispiel:

```

V=750                ;V = dezimaler Ausdruck. Alle nachfolgenden
                     ;Bewegungsfunktionen mit den Interpolationsarten LINEAR und
                     ;CIRCULAR haben als Geschwindigkeit den Wert V = 750 mm/s vorgegeben

MOVE LINEAR pos1
MOVE CIRCULAR (pz, pe)

V=750                ;Die Geschwindigkeit beim linearen Anfahren der Punkte pos1
MOVE LINEAR pos1      ;und pos2 beträgt V = 750 mm/s
MOVE LINEAR pos2

V=300                ;Die Punkte place3 und place4 und alle weiteren werden mit
MOVE LINEAR place3    ;V= 300 angefahren
MOVE LINEAR place4

```

RCL-Bewegungsanweisungen

Anweisungsbezogene Geschwindigkeitsvorgabe (WITH)

Soll die Bahngeschwindigkeit nur für eine Bewegungsanweisung gelten, wird die Geschwindigkeit anweisungsbezogen, d.h. mit der Bewegungsanweisung vorgegeben.

Geschwindigkeitsbezeichnung und Wertzuweisung sind Bestandteil der Bewegungsfunktion, für die sie Gültigkeit haben soll.

Die Programmierung erfolgt im Anschluss an die Interpolationsart mit dem Schlüsselwort WITH.

Beispiel LINEAR:

```
MOVE_REL LINEAR WITH V=500 TO pos
```

Beispiel PTP:

```
MOVE PTP WITH V_PTP=70% TO pos
```

Die anweisungsübergreifenden Geschwindigkeitsvorgaben haben auf die anweisungsbezogenen Vorgaben keinen Einfluss.

Override

Über die Kinematik-Parameter

K-0-0240, RoCoFeedrate-Override,

K-0-0241, RoCoFeedrate-Override-Steuerwort,

K-0-0242, RoCoAcceleration-Override, und

K-0-0243, RoCoDeceleration-Override,

können global für die Kinematik gültige Override-Parameter eingestellt werden. Diese haben direkten Einfluss auf die Bewegungsvorgabe.

VFIX, VFIX_PTP, AFIX: Override deaktivieren

Mit den RCL-Befehlen VFIX, VFIX_PTP und AFIX können nachstehende für die Kinematik gültige Override-Parameter deaktiviert werden:

Bezeichnung	deaktiviert Parameter
VFIX, VFIX_PTP	K-0-0240, RoCoFeedrate-Override
AFIX	K-0-0242, RoCoAcceleration-Override

Abb. 12-22: VFIX, AFIX: Override deaktivieren

VFIX, AFIX und VFIX_PTP wirken nur satzweise, d.h., dass diese Werte nur zusammen mit dem Schlüsselwort WITH innerhalb eines MOVE Befehls programmiert werden können/dürfen. Nach diesem Satz ist die Sperre automatisch wieder aufgehoben. Eine Beeinflussung der Geschwindigkeit oder der Beschleunigung über Poti ist für diesen Satz nicht wirksam/möglich.

Beispiele:

```
MOVE PTP WITH VFIX_PTP = 0.8 TO @pos1 ; poti wird nicht berücksichtigt
```

```
MOVE PTP WITH V_PTP = 0.8 TO @pos1 ; poti wird berücksichtigt
```

Zusatzinformationen Geschwindigkeitsfaktor

Geschwindigkeits- und Zeitvorgaben können mit dem programmierten Geschwindigkeitsfaktor "VFACTOR" geändert werden.

Alle Geschwindigkeitsvorgaben werden mit dem Geschwindigkeitsfaktor multipliziert.

Zeitvorgaben werden durch den "VFACTOR" dividiert. Die so ermittelten Werte gelten dann für die nachfolgenden Bewegungsfunktionen.

RCL-Bewegungsanweisungen

VFACTOR

Der Faktor bezieht sich auf die vorgegebenen Geschwindigkeits- bzw. Zeitangaben im Programm.

Der Faktor kann auch in Prozent angegeben werden, d.h. VFAC-
TOR = 180 % ist gleichbedeutend mit VFACTOR = 1.8.

Bezeichnung	VFACTOR
Maßeinheit	1 oder %
Eingabebereich	0.0001 (0.01%) bis 6 (600%)
Einschaltzustand	1.0 (100%)

Abb. 12-23: Geschwindigkeitsfaktor



Beim Start eines RCL-Programms werden AFACTOR, DFACTOR und VFACTOR auf 1.0 (100 %) gesetzt

Beispiel:

VFACTOR=180%

;Es wirkt der VFACTOR von 180% (= Faktor 1.8) auf
;die Bahngeschwindigkeit zu den Positionen 1, 2,
;3 und 4. pos1 wird mit 180% von Vmax angefahren
;(Einschaltzustand 10%, Faktor 1.8)

MOVE PTP TO pos1

V=200

MOVE LINEAR TO pos2

MOVE LINEAR WITH V=100 TO pos3

MOVE LINEAR WITH T=10 TO pos4

;pos2 wird mit 360 mm/s angefahren
;pos3 wird mit 180 mm/s angefahren
;pos4 wird mit einer Zeit von 5.6 Sekunden
;angefahren (10/1.8= 5.6)

Wirksame Geschwindigkeit

Die tatsächlich wirksame Geschwindigkeit wird aus der programmierten Geschwindigkeit (V), dem Geschwindigkeits-Faktor (VFACTOR) und dem Geschwindigkeits-Override (K-0-0240) berechnet.

Beispiel:

K-0-0054 Bit 0 = 0 (mm)

Programmiertes V = 1000.00 mm/s, VFACTOR = 80% und K-0-0240 = 100 %
% daraus ergibt sich ein wirksames

$V_{\text{wirk}} = V \cdot \text{VFACTOR} \cdot \text{K-0-0240} = 800 \text{ mm/s}$

Zusatzinformationen Beschleunigung

Aus den Beschleunigungsangaben erkennt die Steuerung, wie schnell der Roboter die ihm vorgegebene Geschwindigkeit erreichen muss.



Beschleunigungsangaben können nur bei LINEAR- und CIRCULAR-Interpolation programmiert werden.

Beim PTP-Verfahren beschleunigt der Roboter mit den im Achs-Datensatz festgelegten Maximalwerten. Eine Veränderung der Beschleunigung ist mit dem AFACTOR möglich.

Möglichkeiten der Programmierung der Beschleunigungsvorgabe

Bei der LINEAR- und CIRCULAR-Interpolation kann die Beschleunigungsvorgabe erfolgen als

- anweisungsübergreifende Vorgabe
- anweisungsbezogene Vorgabe

RCL-Bewegungsanweisungen

- Beschleunigungsfaktor.

Der Beschleunigungsfaktor wirkt auch beim PTP-Verfahren.

Anweisungsübergreifende Beschleunigungsvorgabe (Globale Vorgabe)

Interpolationsart: LINEAR und CIRCULAR

Bleibt die Beschleunigung für das ganze Programm oder für einen größeren Abschnitt gleich, ist es sinnvoll die Beschleunigung global vorzugeben.

Bezeichnung und Wertzuweisung bilden eine eigenständige Anweisung zu Beginn des Programms bzw. des Programmabschnittes.

Der Beschleunigungswert hat solange Gültigkeit, bis er durch eine weitere globale Vorgabe ersetzt wird.

Beispiel:

```
A=30 ;Alle nachfolgenden Bewegungsfunktionen mit den
;Interpolationsarten LINEAR und CIRCULAR haben als
;Beschleunigung den Wert A=30 mm/s² vorgegeben
V=100 ;Die Beschleunigung bis zum Erreichen der
;Bahngeschwindigkeit V=100 beträgt beim Anfahren der Punkte
;pos1 bis pos3 A=30 mm/s²
MOVE CIRCULAR (pz,pos1)
MOVE LINEAR pos2
MOVE LINEAR pos3

A=15 ;Die Positionen pos4, pos6 und alle weiteren mit LINEAR- und
;CIRCULAR-Interpolationen haben als Beschleunigungsvorgabe den
;Wert A=15 mm/s²
MOVE LINEAR pos4
MOVE PTP pos5
MOVE LINEAR pos6
```



pos5 wird mit PTP angefahren. Die Beschleunigungsvorgabe A = 15 mm/s² ist hier unwirksam. (K-0-0054 Bit 0 = 0 (mm))

Anweisungsbezogene Beschleunigungsvorgabe (WITH)

Soll eine Bahnbeschleunigung nur innerhalb eines Bewegungsbefehls wirksam sein, so wird die Beschleunigung lokal vorgegeben.

A

Bezeichnung	A
Maßeinheit	mm/s² bzw inch/s² (siehe K-0-0054)
Einschaltzustand	10

Abb. 12-24: Beschleunigung bei LINEAR- und CIRCULAR-Interpolation



Lokale Beschleunigungsvorgaben sind nur bei LINEAR- und CIRCULAR-Interpolation möglich

Beim PTP-Verfahren beschleunigt der Roboter mit den im Achs-Datensatz festgelegten Maximalwerten. Eine Veränderung der Beschleunigung ist mit dem AFACTOR möglich.

Bezeichnung und Wertzuweisung sind Bestandteil der Bewegungsfunktion, für die die Beschleunigung Gültigkeit haben soll.

Die Programmierung erfolgt im Anschluss an die Interpolationsart mit dem Schlüsselwort WITH.

Beispiel LINEAR:

```
MOVE LINEAR WITH J=1000 TO pos
```

RCL-Bewegungsanweisungen

Beispiel CIRCULAR:

```
MOVE_REL CIRCULAR WITH J=2000 EXACT (dz,dis1)
```



Die globalen Beschleunigungsvorgaben haben auf die anweisungsbezogenen Vorgaben keinen Einfluss.

Es besteht die Möglichkeit, innerhalb einer Bewegungsfunktion Geschwindigkeit und Beschleunigung zusammen lokal vorzugeben. Bei der Programmierung werden die beiden Vorgaben durch ein Komma getrennt. Die Reihenfolge ist beliebig.

Programm:

```
MOVE_REL CIRCULAR WITH V=120, A=35, J=1000 EXACT (dz,dis1)
```

Zusatzinformationen Beschleunigungsfaktor

Beschleunigungsvorgaben können mit dem Beschleunigungsfaktor "AFACTOR" nochmals beeinflusst werden. Der Beschleunigungsfaktor wird auf alle Beschleunigungsvorgaben multipliziert. Die daraus resultierenden Werte gelten dann für alle nachfolgenden Bewegungsfunktionen.



Der AFACTOR wirkt bei anweisungsbezogener Beschleunigungsvorgabe nicht (im Gegensatz zum DFACTOR).

Beispiel:

K-0-0054 Bit 0 = 0 (mm)

Programmiertes A = 300.00 mm/s² und AFACTOR = 90% daraus ergibt sich ein wirksames

$$A_{\text{wirk}} = A * \text{AFACTOR} = 270 \text{ mm/s}^2.$$

AFACTOR

Bezeichnung	AFACTOR
Maßeinheit	%
Eingabebereich	0.0001 (0.01%) bis 1.2 (120%)
Einschaltzustand	1.0 (100%)

Abb. 12-25: Beschleunigungsfaktor



Der AFACTOR wirkt nicht, wenn die Beschleunigung anweisungsbezogen programmiert wird.



Nach Programmstart werden AFACTOR, DFACTOR und VFACTOR auf 1.0 (100 %) zurückgesetzt.

Beispiel:

```

A = 1000 ;1000 [mm/s²]
V = 200 ;200 [mm/s]

AFACTOR = 50% ;Es wirkt der AFACTOR von 50 %
                ;(=Faktor 0.5) auf die Bahnbeschleunigung

MOVE PTP TO pos1 ;Die Bahnbeschleunigung hat keinen Einfluss auf die PTP-
                ;Interpolation
MOVE CIRCULAR TO (pz,pos2) ;Wirksames A = 500 mm/s² (0.5 * 1000)
MOVE LINEAR WITH A=100 TO pos3 ;Wirksames A = 100 mm/s², AFACTOR hat hier keine Wirkung

```

RCL-Bewegungsanweisungen

Wirksame Beschleunigung

Die tatsächlich wirksame Beschleunigung setzt sich zusammen aus der programmierten Beschleunigung (A), dem Beschleunigungs-Faktor (AFACTOR) und dem Beschleunigungs-Override (K-0-0242).

Beispiel:

K-0-0054 Bit 0 = 0 (mm)

Programmiertes A = 300.00 mm/s², AFACTOR = 90% und K-0-0242 = 100 %
daraus ergibt sich ein wirksames

$$A_{\text{wirk}} = A * AFACTOR * K-0-0242 = 270 \text{ mm/s}^2.$$

Zusatzinformationen Bremsverzögerungsfaktor

Analog zum Beschleunigungsfaktor kann für die Beeinflussung der Bremsverzögerung der Bremsverzögerungsfaktor über den Befehl "DFACTOR" eingestellt werden.

Der Bremsverzögerungsfaktor wird auf sämtliche Bremsverzögerungsvorgaben multipliziert. Die daraus resultierenden Werte gelten dann für alle nachfolgenden Bewegungsfunktionen.

Beispiel:

K-0-0054 Bit 0 = 0 (mm)

Programmiertes A = 300.00 mm/s² und DFACTOR = 90% daraus ergibt sich ein wirksames

$$D_{\text{wirk}} = A * DFACTOR = 270 \text{ mm/s}^2.$$

DFACTOR

Bezeichnung	DFACTOR
Maßeinheit	%
Eingabebereich	0.0001 (0.01%) bis 1.2 (120%)
Einschaltzustand	1.0 (100%)

Abb. 12-26: Bremsverzögerungsfaktor

Wirksame Bremsverzögerung

Die tatsächlich wirksame Bremsverzögerung setzt sich zusammen aus der programmierten Beschleunigung (A), dem Bremsverzögerungsfaktor (DFACTOR) und dem Bremsverzögerungs-Override (K-0-0243).

Beispiel:

K-0-0054 Bit 0 = 0 (mm)

Programmiertes A = 300.00 mm/s², DFACTOR = 90% und K-0-0243 = 100 %
daraus ergibt sich ein wirksames

$$D_{\text{wirk}} = A * DFACTOR * K-0-0243 = 270 \text{ mm/s}^2.$$



Beschleunigung und Bremsverzögerung verwenden als Basis den gleichen A-Wert.

Die Überlagerung erfolgt getrennt mit AFACTOR und DFACTOR.

Beispiel:

A = 1000
V = 200

DFACTOR = 50%

```
;1000 [mm/s2]  
;200 [mm/s]
```

```
;Es wirkt der DFACTOR von 50 %  
; (=Faktor 0.5) auf die Bahnbremsverzögerung
```

RCL-Bewegungsanweisungen

```

MOVE PTP TO pos1                ;Die Bahnbeschleunigung hat keinen Einfluss auf die PTP-
MOVE CIRCULAR TO (pz,pos2)      ;Interpolation
MOVE LINEAR WITH A=100 TO pos3  ;Wirksames D = 500 mm/s² (0.5 * 1000)
                                ;Wirksames D = 50 mm/s² (0.5 * 100)

```

Zusatzinformationen Ruck

Aus den Ruckangaben erkennt die Steuerung, wie schnell der Roboter die ihm vorgegebene Beschleunigung erreichen muss.

Bezeichnung	J
Maßeinheit	mm/s³ bzw inch/s³ (siehe K-0-0054)
Einschaltzustand	1000000

Abb. 12-27: Bahnruck bei LINEAR- und CIRCULAR-Interpolation



Ruckangaben können nur bei LINEAR- und CIRCULAR-Interpolation in der Satzübergangsart BLOCK_SLOPE und CONT_SLOPE programmiert werden.

In der Satzübergangsart PROGR_SLOPE kann eine Beschleunigungsänderungszeit mit dem Parameter K-0-0014 und eine Bremsverzögerungsänderungszeit mit dem Parameter K-0-0015 eingestellt werden.

Möglichkeiten der Programmierung der Ruckvorgabe

Die Ruckvorgabe kann erfolgen als

- anweisungsübergreifende Vorgabe
- anweisungsbezogene Vorgabe

Anweisungsübergreifende Ruckvorgabe (Globale Vorgabe)

Bleibt der Ruck für das ganze Programm oder für einen größeren Abschnitt gleich, ist es sinnvoll den Ruck global vorzugeben.

Bezeichnung und Wertzuweisung bilden eine eigenständige Anweisung zu Beginn des Programms bzw. des Programmabschnittes.

Der Ruckwert hat solange Gültigkeit, bis er durch eine weitere globale Vorgabe ersetzt wird.

Beispiel:

```

J=1000                ;Alle nachfolgenden Bewegungsfunktionen mit den
                      ;Interpolationsarten LINEAR und CIRCULAR haben als
                      ;Ruck den Wert J=1000 mm/s³ vorgegeben.
A=30                  ;Der Ruck bis zum Erreichen der Bahnbeschleunigung A=30 mm/s² beträgt
                      ;beim Anfahren der Punkte pos1 bis pos3 J=1000 mm/s³
MOVE CIRCULAR (pz,pos1)
MOVE LINEAR pos2
MOVE LINEAR pos3

J=500                 ;Die Positionen pos4, pos5 und alle weiteren Bewegungsfunktionen
                      ;mit LINEAR- und CIRCULAR-Interpolation haben als Ruckvorgabe den
                      ;Wert J=500 mm/s³
MOVE LINEAR pos4
MOVE LINEAR pos5
MOVE PTP pos6         ;Pos6 wird mit PTP angefahren, daher ist die Ruckvorgabe unwirksam.

PROGR_SLOPE           ;Die Satzübergangsart PROGR_SLOPE wird für alle nachfolgenden
                      ;Bewegungsfunktionen wirksam.
MOVE LINEAR pos7      ;Pos7 wird mit der Satzübergangsart PROGR_SLOPE angefahren,
                      ;daher ist die Ruckvorgabe unwirksam.

```

Anweisungsbezogene Ruckvorgabe (WITH)

Soll ein Bahnruck nur innerhalb eines Bewegungsbefehls wirksam sein, so wird der Ruck lokal vorgegeben.

RCL-Bewegungsanweisungen

Bezeichnung und Wertzuweisung sind Bestandteil der Bewegungsfunktion, für die der Ruck Gültigkeit haben soll.

Die Programmierung erfolgt im Anschluss an die Interpolationsart mit dem Schlüsselwort WITH.

Beispiel LINEAR:

```
MOVE LINEAR WITH J=1000 TO pos
```

Beispiel CIRCULAR:

```
MOVE_REL CIRCULAR WITH J=2000 EXACT (dz,dis1)
```



Die anweisungsübergreifenden Ruckvorgaben haben auf die anweisungsbezogenen Vorgaben keinen Einfluss.

Es besteht die Möglichkeit, innerhalb einer Bewegungsfunktion Geschwindigkeit, Beschleunigung und Ruck zusammen lokal vorzugeben. Bei der Programmierung werden die Vorgaben durch ein Komma getrennt. Die Reihenfolge ist beliebig.

Programm:

```
MOVE_REL CIRCULAR WITH V=120, A=35, J=1000 EXACT (dz,dis1)
```

12.3 Bewegungsbeeinflussende Anweisungen

12.3.1 Allgemeines

Neben den oben beschriebenen direkten Bewegungsanweisungen gibt es noch folgende Anweisungen, die die Bewegungsabläufe mit beeinflussen:

- Synchronisationsanweisungen SYNC, SYNCHRON, SYNCHRON_END
- Beschleunigung, Verzögerung zwischen Bewegungsanweisungen (Satzübergänge) BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE

12.3.2 Bandsynchronisation

Die Funktion "Bandsynchron" ermöglicht, dass die Roboterbewegung zu einem Montage- oder Förderband hinsichtlich Position und Orientierung synchronisiert werden kann.

Dabei kann das Band beliebig vor- oder rückwärts fahren, seine Geschwindigkeit ändern oder stoppen.

Die Voreinstellungen sind über die Spezialfunktionen [SPC_FCT: 21, Bandart, Seite 102](#), und [SPC_FCT: 53, bandsynchroner Arbeitsbereich der Bandart 4, Seite 105](#), vorzunehmen.

Das Band muss eine Gerade sein. Diese kann beliebig im Raum stehen, siehe MLC-Parameter "K-0-0070" und folgende. Die Bewegung des Bandes wird durch ein Wegmesssystem aufgenommen oder von der MLC mit z.B. "MC_MoveVelocity" kommandiert.

In RCL stehen zur Programmierung der Funktionalität die Anweisungen SYNC, SYNCHRON sowie SYNCHRON_END zur Verfügung.

Syntax:

```
SYNC belt variable[, variable] condition  
SYNCHRON [kinematic name] belt variable  
SYNCHRON_END [kinematic name] belt variable
```

RCL-Bewegungsanweisungen

Die Bandvariable ist im Vereinbarungsteil des Programms zu deklarieren, siehe auch [Kommunikationskanäle zur SPS, Allgemeines, Seite 64](#), und dabei einer Kinematik zuzuordnen.

Beispiel:

Deklaration der Bandvariablen mehrere Bänder zu einer Kinematik

```
Kin1.belt:    501=belt1, 502=belt2
screwdr.belt: 503=screw_belt ;same belt like belt1
```



Es können insgesamt bis zu 4 Bänder pro Kinematik vereinbart werden. Die Bandnamen und Bandnummern müssen unterschiedlich sein, auch wenn mehrere Kinematiken für eine Bandvariable ein Messsystem mehrfach nutzen.

Es kann nur auf Bändern synchron verfahren werden, die auch für diese Kinematik deklariert sind.

Die Komponentennamen und Achsnamen sind für das zugewiesene Band entsprechend anzugeben.



Die Bandsynchronisation mit der Satzübergangsart `PROGR_SLOPE` und `CONT_SLOPE` wird zur Zeit nicht unterstützt.

Programmierung Bandsynchron

Die Bandvariable vom Datentyp "REAL" enthält den Zählerstand des Bandmesssystems.

Die Bandvariable kann nur über Vergleichsoperationen `>=` und `<=` abgefragt oder im Programm mit "WAIT UNTIL", "IF" bzw. "SYNC" verwendet werden.

Beispiel:

```
WAIT UNTIL belt_1>=6 ;belt variable in WAIT UNTIL
```



Das Programm wartet beim SYNC-Befehl so lange, bis die Bandvariable das Kriterium erfüllt hat.

Der Befehl "SYNC" setzt die Bandvariable auf Null, bzw. auf den eingestellten Rücksetzwert (`SPC_FCT 28`).

Das Nullsetzen kann abhängig von einer Bedingung erfolgen.¹⁾

In nachstehendem Beispiel sind die Möglichkeiten, wie der SYNC-Befehl verwendet werden kann, aufgeführt.

Beispiel:

```
SYNC belt_1>=300      ;subtrahiert 300 vom Bandzaehler, wenn der aktuelle
SYNC belt_1, li_scr=1 ;Zaehlerstand ≥ 300 ist
                     ;Nullsetzen erfolgt abhängig von einer Bedingung
```

Mit folgendem Beispiel kann man die Modulo-Verarbeitung nichtwartend realisieren.

Beispiel:

```
IF belt_1>=300      ;Prüft zunächst, ob der Zaehlerstand ≥ 300 ist
THEN
```

¹⁾ subtrahiert einen Wert (ähnlich Modulo).

RCL-Bewegungsanweisungen

```
BEGIN
  SYNC belt_1>=300          ;Der Bandzähler wird hier in jedem Fall zurück gesetzt,
                              ;da die Bedingung stimmt.
END
```

Mit dem Befehl "SYNCHRON" wird die Bandsynchronisation eingeschaltet. Von nun an sind alle programmierten Bewegungen hinsichtlich Position und Orientierung mit dem Band synchronisiert. Das Aufsynchronisieren erfolgt sobald der programmierte Zielpunkt (wird mit dem Zähler bewegt) im Bandsynchron-Arbeitsbereich liegt.

Mit "SYNCHRON_END" wird die Synchronisation abgeschaltet.

Beispiel:

```
SYNCHRON Kin1 belt_1          ;Bandvariable und SYNCHRON-Anweisung
  MOVE LINEAR TO pos_1        ;(Der Bandzähler von pos_1 hat gültigen einen Wert > 0)
SYNCHRON_END belt_1
```

Beispiel:

```
;;COMPILER= RCL3
;;KINEMATICS: (1=robi_1)      ;Definition der Kinematiknamen
;;robi_1.JC_NAMES=a1,a2,a3,b1 ;b1 ist ein dummy für Bandvariablen
;;robi_1.WC_NAMES=k1,k2,k3,b_c1

PROGRAM beltsyn
VAR:
  CONST :
    C_BELT_NO_1 = 1,          ; Bandnummer
    C_BK_PTP = 4              ; Bandart PTP
  BELT :
    500 + C_BELT_NO_1 = belt1 ; Kanal / Bandname

    ; Deklaration der Spezialfunktionen
  SPC_FCT : 21 = belt_kind (VALUE INTEGER: belt_no
                             VALUE INTEGER: kind_belt)
  SPC_FCT : 53 = belt_area (VALUE INTEGER: BeltNo
                             JC_POINT: @Base
                             REAL: rTotalLength
                             REAL: rBeginLength)

INPUT: 1=i1,2=i2

BEGIN
    ; Festlegung Bandart
    belt_kind ( C_BELT_NO_1, C_BK_PTP )
    ; Festlegung bandsynchroner Arbeitsbereich
    rTotalLength = 560
    rBeginLength = 400
    Base = (280, 140, -600, 0, 0, 0)
    @Base = JC(Base)
    belt_area ( C_BELT_NO_1, @Base, rTotalLength, rBeginLength)

    SYNC belt1,i1=1          ;wartet bis die SPS i1=1 gesetzt hat, um den
                              ;Bandzähler belt1 (501) zurückzusetzen
    SYNCHRON robi_1 belt1    ;Synchronisation der Kinematik robi_1 zu belt1

    MOVE robi_1 LINEAR TO robi_1.POS
    WAIT UNTIL belt1>=1000
    SYNCHRON_END robi_1 belt1

PROGRAM_END
```

12.3.3 Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)

Syntax:

BLOCK_SLOPE

PROGR_SLOPE

CONT_SLOPE

Bei einem einfachen Bewegungsablauf werden die angesteuerten Achsen mit jeder MOVE-Anweisung auf die programmierte Geschwindigkeit be-

RCL-Bewegungsanweisungen

schleunigt, mit der programmierten Geschwindigkeit verfahren und zum Erreichen der programmierten Position wieder auf $V = 0$ abgebremst.

Die Geschwindigkeitsänderung bezeichnet man als "SLOPE" (eng. für "Rampe" oder "Neigung"). Entsprechend bezeichnet man die Beschleunigungsphase als Up-Slope und die Bremsphase als Down-Slope.

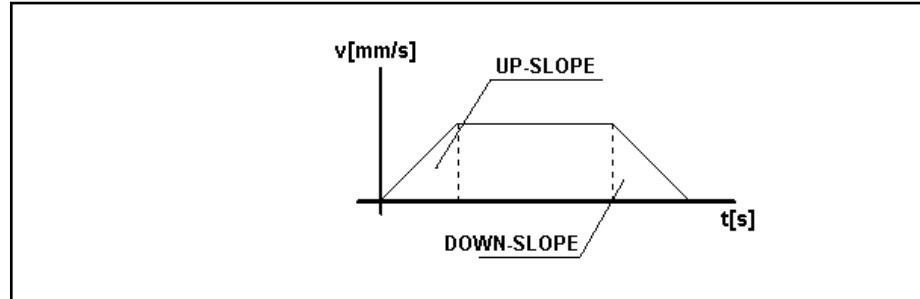


Abb. 12-28: Up-Slope und Down-Slope

Allgemeines Mit der Satzübergangsart `PROGR_SLOPE`, `CONT_SLOPE` werden die überschleifenden Bewegungsbefehle ohne zwischenzeitiges Anhalten zusammenhängend Verfahren. Die Satzübergänge erfolgen mit konstanter Sollgeschwindigkeit. Werden die Sätze mit unterschiedlichen Geschwindigkeiten programmiert erfolgt eine rampenförmige Geschwindigkeitsänderung zwischen den Bewegungsbefehlen.

Mit der Satzübergangsart `BLOCK_SLOPE` erfolgt ein satzweises Beschleunigen und Abbremsen.

Aktivierung der SLOPE-Art Die Initial-Einstellung der Slope-Art ist `BLOCK_SLOPE`.

Die Slope-Art kann im RCL-Programm umgeschaltet werden. Dazu werden die RCL-Standard-Funktionen `BLOCK_SLOPE`, `PROGR_SLOPE` und `CONT_SLOPE` definiert.

Mit der Aktivierung einer SLOPE-Art wird die vorherige ausgeschaltet.



Bei einem Slope-Arten-Wechsel ist die Übergangs-Geschwindigkeit stets 0 !

BLOCK_SLOPE Die Betriebsart `BLOCK_SLOPE` behandelt sämtliche Geschwindigkeits- und Beschleunigungsvorgaben satzbezogen. Beschleunigen und Abbremsen erfolgt zu Beginn und am Ende eines Bewegungsbefehls. Es gibt somit keine Übergangsgeschwindigkeit zwischen zwei Sätzen. Über den Überschleifradius wird ein Überblenden der Geschwindigkeit durch verfrühtest Starten des Folgesatzes erreicht.

PROGR_SLOPE Bei der Betriebsart `PROGR_SLOPE` handelt es sich um eine satzübergreifende Geschwindigkeits- und Beschleunigungsvorgabe. Der Roboter wird am Anfang eines zusammenhängenden Bewegungsablaufs beschleunigt bzw. am Ende abgebremst. An den Satzübergängen wird die Geschwindigkeit konstant gehalten, falls keine Geschwindigkeitsänderung programmiert wurde. Über den Überschleifradius wird ein räumliches Überschleifen mittels Kreissegmenten durchgeführt.

CONT_SLOPE Bei der Betriebsart `CONT_SLOPE` handelt es sich um eine satzübergreifende Geschwindigkeits- und Beschleunigungsvorgabe. Der Roboter wird am Anfang eines zusammenhängenden Bewegungsablaufs beschleunigt bzw. am Ende abgebremst. An den Satzübergängen wird die Geschwindigkeit kon-

RCL-Bewegungsanweisungen

stant gehalten, falls keine Geschwindigkeitsänderung programmiert wurde. Über den Überschleifradius wird ein räumliches Überschleifen mittels Spline-segmenten durchgeführt.



Näheres Informationen zum räumlichen Überschleifen mittels Überschleifradius befinden sich Hier: [Kap. 12.3.6 "Räumliches Überschleifen" auf Seite 137](#)

Nachfolgend ist die Programmierung der Slope-Betriebsart in einigen RCL-Programmbeispielen erläutert. Die Wirkung wird anhand von Graphiken dargestellt.

Beispiel 1:

PROGRAM ex_1	;(1)
BEGIN	;(2)
;;INT=LINEAR	;(3)
V=800,A=1000	;(4)
MOVE TO beg_point	;(5) Programmslope nicht aktiv, d.h. der Roboter wird in
	;(6) jedem Satz mit A=1000 mm/s ² beschleunigt und am Ende
	;(7) des Satzes wieder abgebremst.
MOVE TO point_center	;(6)
MOVE TO end_point	;(7)
PROGR_SLOPE	;(8) PROGR_SLOPE wird aktiviert
MOVE VIA beg_point	;(9)
MOVE VIA point_center	;(10)
MOVE TO end_point	;(11)
MOVE VIA RAMP_1	;(12)
MOVE VIA RAMP_2	;(13)
MOVE VIA beg_point	;(14)
MOVE VIA point_center	;(15)
MOVE TO end_point	;(16)
BLOCK_SLOPE	;(17)
HALT	;(18)
PROGRAM_END	;(19)

In dem Beispiel ist im Parameter K-0-0054 Bit 0 = 0 (mm) konfiguriert.

Im ersten Teil des Programms, Satz 5 bis 7, ist die Default-Slope-Art BLOCK_SLOPE aktiv. Der Roboter wird in jedem Satz mit A = 1000 mm/s² beschleunigt und am Ende des Satzes wieder abgebremst, siehe Slope, Bild 1.

In Satz 8 wird der PROGR_SLOPE eingeschaltet. Dadurch wird erreicht, dass der Roboter in Satz 9 mit A=1000 mm/s² beschleunigt wird.

Die Satzübergänge von 9 nach 10 und 10 nach 11 werden mit konstanter Geschwindigkeit ausgeführt.

Am Ende von Satz 11 wird der Roboter mit der programmierten Bremsverzögerung von 1000 mm/s² abgebremst, siehe Slope, Bild 2.

Falls die Verfahrstrecke im MOVE TO-Satz (Satz 11) nicht ausreicht um den Roboter über Slope-Funktion abzubremsen, wird der Bremsvorgang bereits im vorhergehenden Satz (Satz 10, Slope, Bild 3) eingeleitet.

Die Beschleunigungsphase kann über beliebig viele Sätze laufen, z.B. Satz 12 bis 13, Slope, Bild 3. Dies gilt auch für die Abbremsphase, z.B. Satz 15 bis 16 Bild 3.

RCL-Bewegungsanweisungen

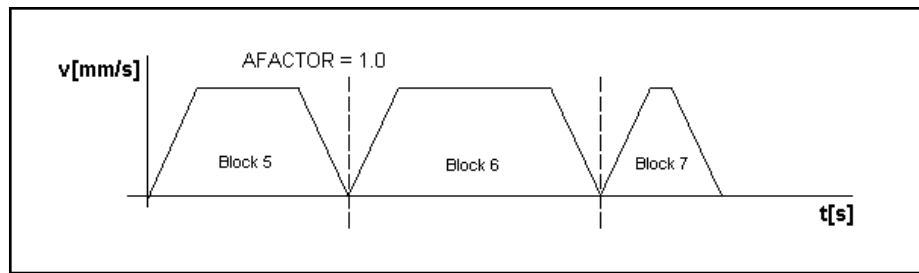


Abb. 12-29: Slope, Bild 1 - BLOCK_SLOPE

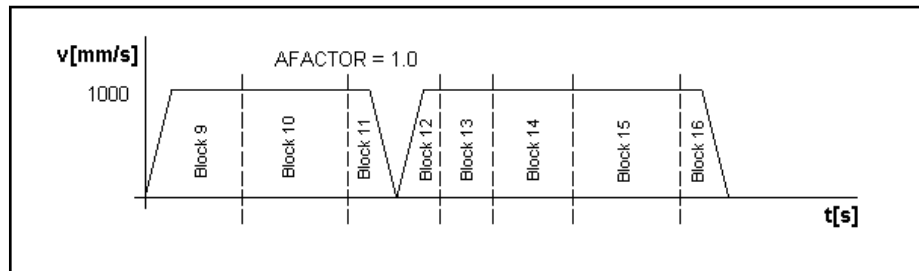


Abb. 12-30: Slope, Bild 2 - PROGR_SLOPE

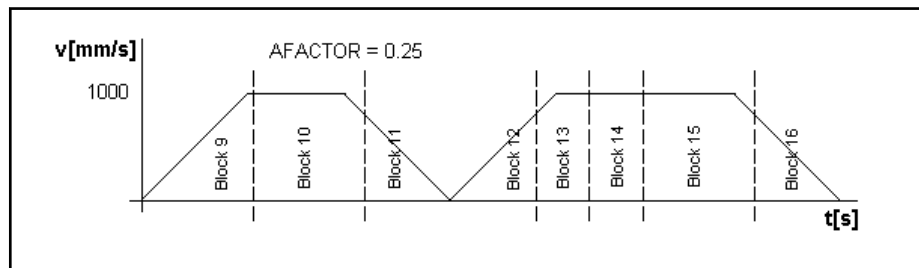


Abb. 12-31: Slope, Bild 3 - PROGR_SLOPE

Bewegungsabbruch

Abbruch einer zusammenhängenden Bewegung im RCL-Programm

Ein Bewegungsablauf mit der Slope-Betriebsarten PROGR_SLOPE und CONT_SLOPE wird durch folgende RCL-Befehle unterbrochen:

- Synchroner Eingangsabfrage
- WAIT
- PAUSE
- HALT
- BLOCK_SLOPE
- mehrere Sätze ohne Verfahreninformation, z.B. Berechnungen, Variablenzuweisungen, Setzen von Ausgangssignalen.

Die Anzahl der möglichen Sätze ist abhängig von der Länge der vorausgegangenen Verfahrensätze und der Art der durchgeführten Zuweisungen bzw. Berechnungen.

Beispiel 2:

```
PROGRAM ex_3
;;INT=LINEAR
BEGIN
V=800, A=1000
PROGR_SLOPE

MOVE LINEAR VIA beg_point

WAIT 1
MOVE LINEAR VIA point_center
```

;Interpolationsart LINEAR gilt für folgende Bewegungsanweisungen

;PROGR_SLOPE wird eingeschaltet

;Endgeschwindigkeit = 0, da der nächste Befehl ein synchroner Befehl ist

;Hier wird geschwindigkeitsüberschleifend gefahren.

RCL-Bewegungsanweisungen

```
MOVE TO end_point  
PROGRAM_END
```

;Die Endgeschwindigkeit wird berechnet.

Umschaltung der Interpolationsart

Das satzübergreifende Beschleunigungs- und Verzögerungsverhalten kann unabhängig von der Interpolationsart aktiviert werden. Satzübergänge ohne Wechsel der Interpolationsart werden ausgeführt, wie unter [Satzübergänge \(BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE\)](#) , Seite 131. beschrieben.

Umschaltung zwischen Linear- und Zirkularinterpolation

Beim Wechsel von Linear- nach Zirkularinterpolation und umgekehrt werden die Satzübergänge ausgeführt, wie unter [Satzübergänge \(BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE\)](#) , Seite 131 beschrieben.

Beispiel 3:

```
PROGRAM ex_4  
;;INT=LINEAR  
BEGIN  
V=800,A=1000  
PROGR_SLOPE  
MOVE VIA beg_point  
MOVE CIRCULAR VIA (int_point1,circular_end1)  
MOVE CIRCULAR TO (int_point2,circular_end2)  
MOVE TO pnt_center  
MOVE CIRCULAR TO (int_point3, circular_end3)  
HALT  
PROGRAM_END
```

Umschaltung zwischen LINEAR- und PTP-Interpolation

Vor dem Umschalten von LINEAR- auf PTP-Interpolation und umgekehrt sollte der Bewegungsablauf durch einen "MOVE TO"-Satz beendet werden, damit ein gesteuerter Übergang durchgeführt werden kann.

Erfolgt die Umschaltung während einer zusammenhängenden Bewegung, wird die Geschwindigkeit sprunghaft geändert.

Beispiel 4:

```
PROGRAM ex_5  
BEGIN  
PROGR_SLOPE  
V_PTP=1  
MOVE PTP TO beg_point  
MOVE LINEAR TO p1  
MOVE PTP VIA p2  
MOVE LINEAR VIA end_pnt  
V_PTP=0.5  
MOVE PTP TO end_pnt_1  
HALT  
PROGRAM_END
```

Überschreitung von Achsgrenzwerten

Bedingt durch die Koordinatentransformation können Überschreitungen von Grenzwerten einzelner Maschinenachsen bei Bahnbetrieb nicht ausgeschlossen werden. Es kann lediglich eine Überwachung während des Programmablaufs durchgeführt werden.

Diese Überwachung löst im Fehlerfall eine der beiden folgenden Fehlermeldungen aus:

- Interpolator-Stop, Achse X
- Achsgeschwindigkeit zu groß, Achse X.

RCL-Bewegungsanweisungen

Dabei ist X die Nummer der entsprechenden Maschinenachse.

Der Programmierer wird dadurch veranlasst, das Programm an den entsprechenden Stellen zu verändern. Eine automatische Geschwindigkeitsanpassung kann nicht durchgeführt werden, da dies der Forderung nach möglichst konstanter Bahngeschwindigkeit widersprechen würde.

12.3.4 Look-Ahead über mehrere Sätze bei PROGR_SLOPE und CONT_SLOPE

In den Slope-Art PROGR_SLOPE und CONT_SLOPE erfolgt die Berechnung der möglichen Satzübergangsgeschwindigkeiten im Vorfeld über mehrere Sätze. Dabei wird berücksichtigt, dass der Roboter ausreichend Distanz hat, um am Ende der Verfahrbewegung anhalten zu können.

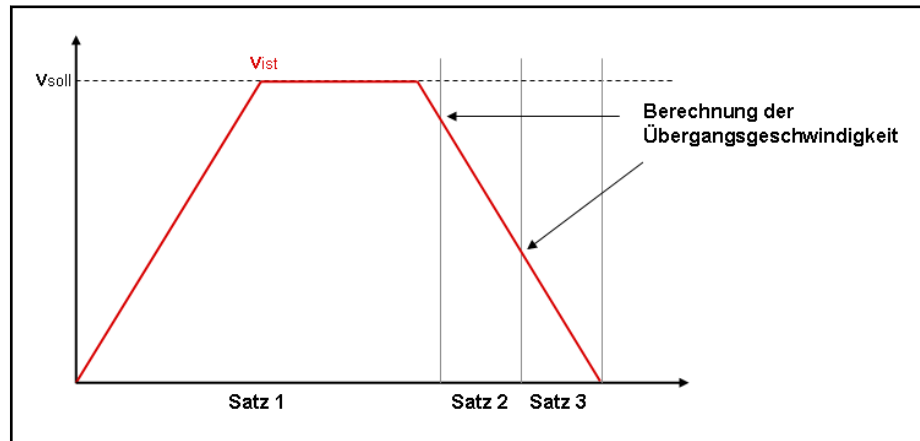


Abb. 12-32: Look-Ahead über mehrere Sätze

Die Abbildung verdeutlicht die Arbeitsweise des integrierten Look-Aheads. Die programmierte Sollgeschwindigkeit wird in Satz 2 und Satz 3 nicht erreicht. Die Endgeschwindigkeit des ersten Satzes wird entsprechend des Bremswegs angepasst.

12.3.5 Spätes Kommandieren

Die MLC Robot Control bietet die Möglichkeit, Bewegungsbefehle spät abzusetzen. Dies ist sowohl bei PROGR_SLOPE und CONT_SLOPE als auch bei BLOCK_SLOPE möglich. Wird während des Verfahrens eines Satzes ein weiterer Satz kommandiert, beeinflusst dieser gegebenenfalls die Endgeschwindigkeit des aktuellen Satzes.

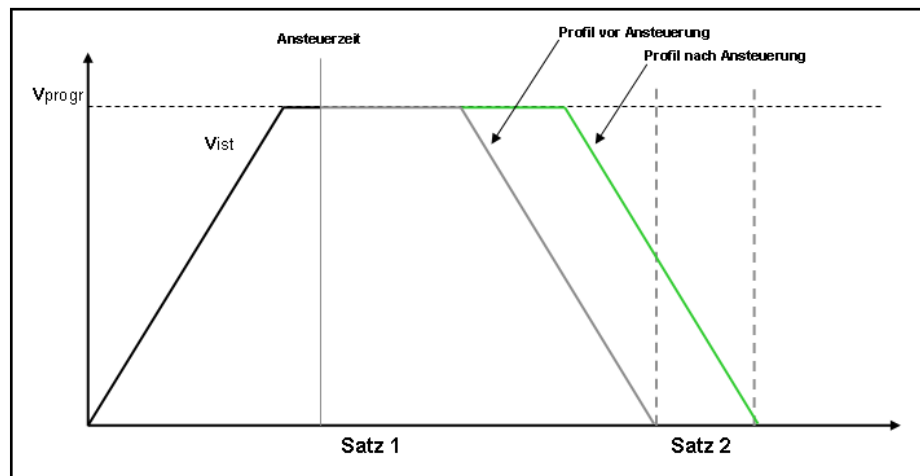


Abb. 12-33: Anhebung der Endgeschwindigkeit durch spätes Kommandieren

RCL-Bewegungsanweisungen

Die Abbildung verdeutlicht einen solchen Vorgang. Satz 1 beginnt mit dem Beschleunigen auf die programmierte Geschwindigkeit. Der Satz wird so interpoliert, dass die Kinematik nach dem programmierten Satz zum Stillstand kommt. Zum Zeitpunkt "Ansteuerzeit" wird während Satz 1 aktiv ist Satz 2 kommandiert. Dieser hebt die mögliche Endgeschwindigkeit des ersten Satzes an, obwohl der Satz bereits gestartet wurde.

Das Anpassen der Endgeschwindigkeit bezieht mehrere Sätze im Voraus mit ein. Dieser Mechanismus arbeitet vollständig im Hintergrund.

12.3.6 Räumliches Überschleifen

Überblick

Um die Mechanik des Roboters zu schonen und gleichzeitig den Bewegungsablauf zeitlich zu verkürzen, lassen sich die programmierten Zielpunkte überschleifen. Die Folge ist eine kontinuierliche und harmonische Bewegung bei Satzübergängen.

Überschleifen bedeutet: Durch definierte Abweichen von der programmierten Bahn einen fließenden Geschwindigkeitsverlauf bei Satzübergängen zu gewährleisten.

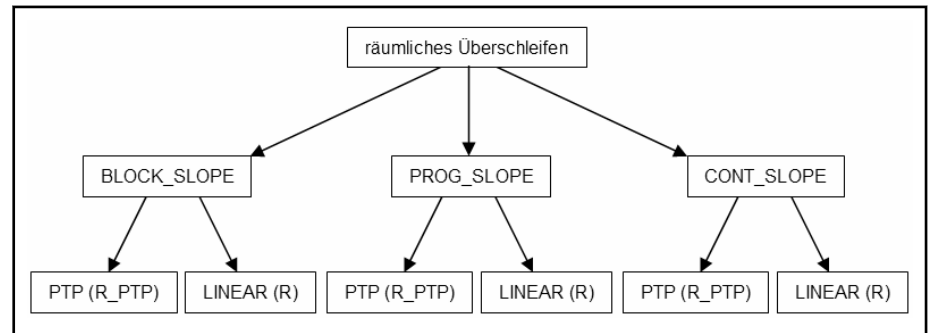


Abb. 12-34: Räumliches Überschleifen

Überschleifradius

Das räumliche Überschleifen wird je nach Interpolationsart mit den Standardvariablen "R" oder "R_PTP" im RCL-Programm beeinflusst.

Für die PTP-Interpolation wird die Standardvariable "R_PTP" programmiert. Dieser Faktor kann dezimal oder als Prozentwert programmiert werden. Er wird mit dem Parameter A-0-0230, Überschleifdistanz PTP, aller Achsen verrechnet.

Für die LINEAR-Interpolation wird die Standardvariable "R" wie Radius verwendet. Für "R" werden absolute Werte programmiert. Die Einheit von "R" ist durch den ausgewählten Robotertyp festgelegt.

R, R_PTP

Bezeichnung	R	R_PTP
Maßeinheit	mm bzw. inch bzw. ° (siehe K-0-0054)	-
Einschaltzustand	0	0
Wirkung	absolut	A-0-0230, Überschleifdistanz PTP

Abb. 12-35: Überschleifradius

Beispiel

MOVE-Anweisung mit R bzw. R_PTP

```
;;INT=LINEAR
R = 50
```

```
MOVE LINEAR WITH R=20 VIA p1
```

;Interpolationsart LINEAR gilt für folgende Bewegungsanweisungen
;R = 50 mm gilt nun satzübergreifend

;R = 20 mm, da satzbezogen definiert

RCL-Bewegungsanweisungen

```

MOVE LINEAR TO p2                ;R = 50 mm
;;INT=PTP                        ;Voreinstellung der Interpolationsart
R_PTP = 1.4                      ;R_PTP = 140% von A-0-0230 der beteiligten Achsen,
                                ;gilt nun satzübergreifend
MOVE PTP WITH R_PTP=1.2 VIA p3   ;R = 120%, da satzbezogen definiert
MOVE PTP TO p4                  ;R = 140%

```



Sind R = 0 oder R_PTP = 0 programmiert, ist das räumliche Überschleifen ausgeschaltet.

Allgemein gilt für BLOCK_SLOPE

- Die räumliche Abweichung im Überschleifbereich ist **geschwindigkeitsabhängig**.
- Die einzelnen Verfahrssätze werden verfrüht um die eingestellten Überschleifdistanzen gestartet. Die Bahngeschwindigkeit resultiert aus der geometrischen Addition der Einzelachsgeschwindigkeiten.

Allgemein gilt für PROGR_SLOPE und CONT_SLOPE

- Die räumliche Abweichung im Überschleifbereich ist **nicht geschwindigkeitsabhängig**.
- Im Überschleifbereich wird die Bahn durch einen von der Steuerung generierten Satz vorgegeben. In einem zusammenhängenden Bewegungsablauf wird versucht die Bahngeschwindigkeit konstant zu halten.



Punkte die in Zirkularinterpolation angefahren werden oder ab denen mit Zirkularinterpolation verfahren wird, werden nicht räumlich überschleift.

Dies gilt sowohl für BLOCK_SLOPE als auch für PROGR_SLOPE und CONT_SLOPE.

Räumliches Überschleifen bei PROGR_SLOPE und CONT_SLOPE

PTP-Überschleifen bei PROGR_SLOPE und CONT_SLOPE

Für jede Achse kann bezüglich ihrer Endposition eine Überschleifdistanz in Grad oder mm im Parameter A-0-0230, Überschleifdistanz PTP, abgelegt werden. Mittels dieser Distanzen wird ein Überschleifbereich um den Zielpunkt definiert. Im RCL-Programm kann der Anwender mittels eines Faktors (R_PTP) diese Distanzen verändern. Alle abgelegten Distanzen für die einzelnen Achsen werden gleichmäßig beeinflusst, sodass ein Stauchen oder Strecken des Überschleifbereichs stattfindet.

Um jede Zielposition kann somit ein eigener Überschleifbereich definiert werden, der optimal an die jeweiligen Erfordernisse angepasst werden kann.

Tritt der Werkzeugarbeitspunkt während der Verfahrbewegung in diesen Überschleifbereich ein, verfährt der Roboter auf einer von der Steuerung generierten Bahn, die sicherstellt, dass sich die einzelnen Achsgeschwindigkeiten nicht sprungförmig ändern. Nach Austritt aus dem Überschleifbereich verhält sich der Roboter identisch zu Verfahrbewegungen ohne programmiertes Überschleifen.

Jeder Bewegungssatz beinhaltet Informationen bezüglich Geschwindigkeit und zu verfahrender Distanz. Wird in aufeinanderfolgenden Sätzen unterschiedliche Geschwindigkeit programmiert, so wird die Geschwindigkeitsänderung an den Satzübergang vollzogen. Ist der definierte Überschleifbereich relativ klein und die Differenz der programmierten Satzgeschwindigkeit relativ hoch, so muss die Differenzgeschwindigkeit der aufeinanderfolgenden Bewegungen in kurzer Zeit überwunden werden. Die Abweichung von der ur-

RCL-Bewegungsanweisungen

sprönglich programmierten Bahn, also ohne Überschleifen, ist auf einen kleinen Bereich beschränkt. Werden große Überschleifbereiche gewählt, ist der Geschwindigkeitsübergang sanfter, jedoch wird auf größerer Strecke von der Bahn abgewichen.

Die Übergänge zwischen programmierter Bahn und Überschleifbahn sind tangential.

Folgendes Beispiel soll die kontinuierliche Geschwindigkeitsänderung bei einem Satzübergang verdeutlichen. Zugrunde gelegt wird ein kartesisches System mit zwei Achsen, eine in X- und eine in Y-Richtung. Die Slope-Art ist `PROGR_SLOPE`. Der Überschleifbereich in X- und Y-Richtung wird über Parameter A-0-0230 der jeweiligen Achse festgelegt. Wird ohne Überschleifen verfahren, kommt es beim Satzübergang für die Y-Achse zu einem Geschwindigkeitssprung von +V nach -V.

Mit eingeschaltetem Überschleifen wird die Geschwindigkeit innerhalb des Überschleifbereichs kontinuierlich von +V nach -V verändert. Es entstehen weichere Satzübergänge.

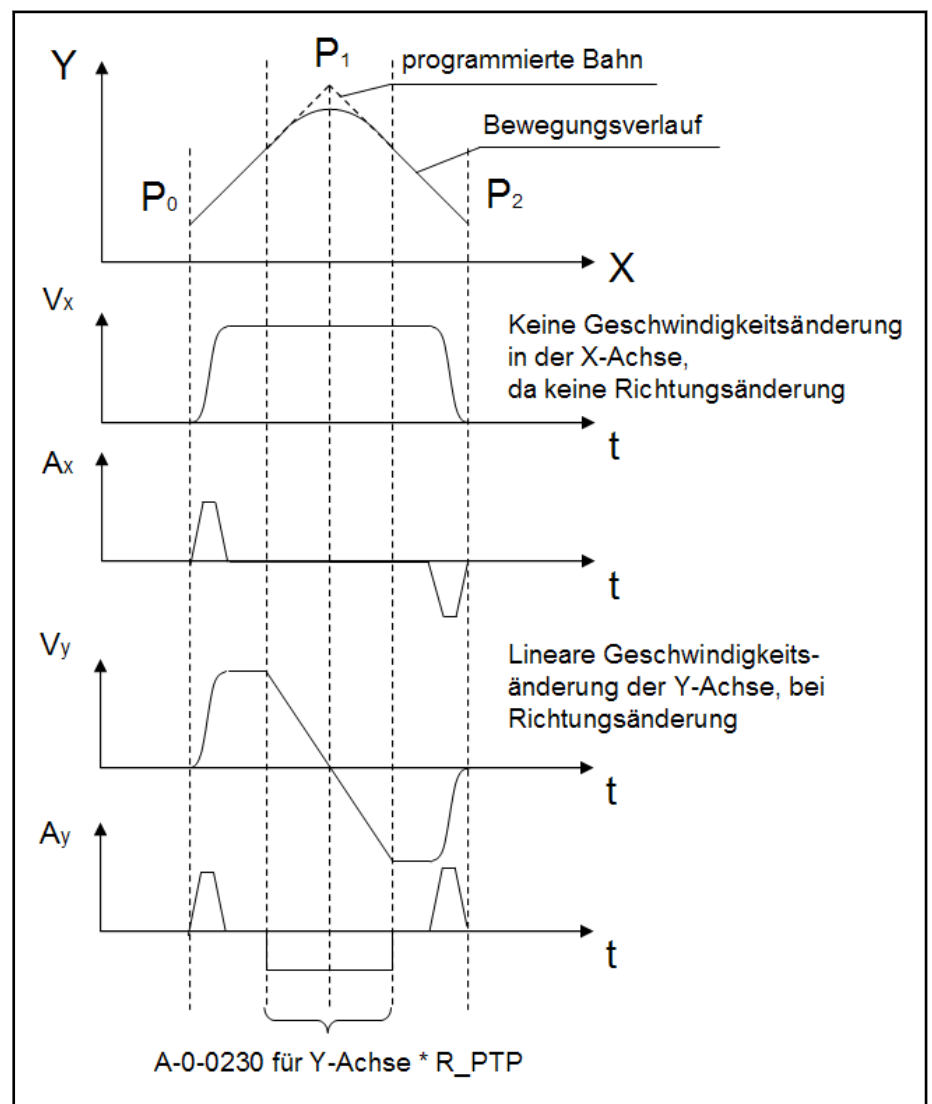


Abb. 12-36: Geschwindigkeitsverlauf der X- und Y-Achse beim PTP Überschleifen in `PROGR_SLOPE`

RCL-Bewegungsanweisungen

Linear-Überschleifen bei PROGR_SLOPE und CONT_SLOPE

Beim Bahn-Überschleifen wird eine Raumkugel um jeden Punkt, der überschleifen werden soll, definiert. Der Radius dieser Kugel wird im RCL-Programm mit der Standardvariablen R festgelegt. Somit ist es möglich beim Bahn-Überschleifen jeden programmierten Raumpunkt mit einem ihm eigenen Überschleifbereich auszustatten. Zwischen Eintritt und Austritt aus dieser Raumkugel wird die originale Bahn zu Gunsten einer konstanten Bahngeschwindigkeit verlassen. Die Bewegung innerhalb der Raumkugel erfolgt auf einer von der Steuerung generierten Bahn. Außerhalb dieser Raumkugel verfährt der Roboter auf der programmierten Bahn.

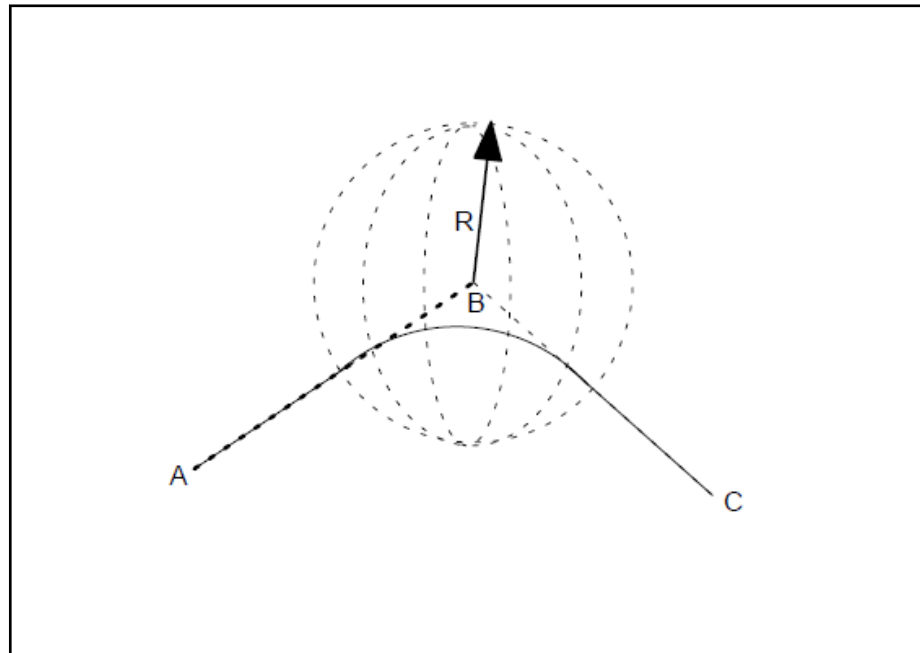


Abb. 12-37: Überschleifkugel bei Linearinterpolation in PROGR_SLOPE

Überschleifradius PROG_SLOPE

Reichen die Distanzen zwischen den zu überschleifenden Punkten nicht aus, um das programmierte Überschleifkriterium einzuhalten, wird von der Steuerung automatisch auf den maximal möglichen Überschleifbereich reduziert. Dieser Sachverhalt ist in

[Abb. 12-38 "Überlappende Überschleifbereiche " auf Seite 141](#) dargestellt.

RCL-Bewegungsanweisungen

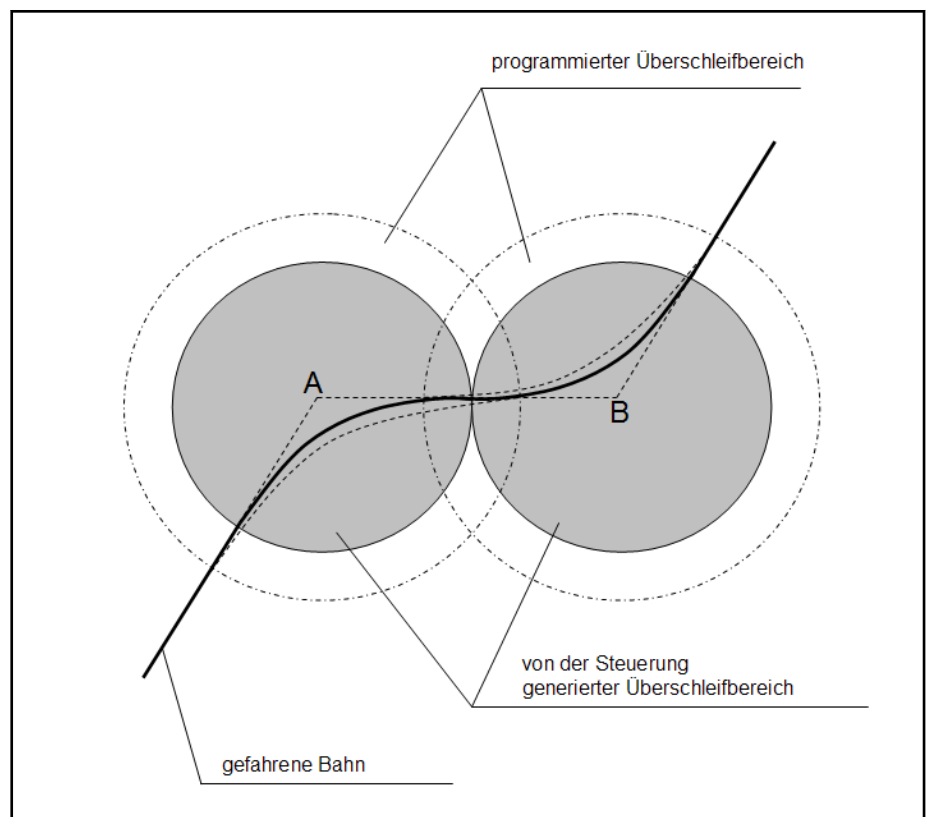


Abb. 12-38: Überlappende Überschleifbereiche

Erklärung zum Bild:

Überlappende Überschleifbereiche werden von der Steuerung automatisch so reduziert, dass sich der hierfür maximal mögliche Überschleifbereich ergibt. Er ist mit der durchgezogenen Linie dargestellt. Weitere Einzelheiten zum Wirkradius beim Überschleifen können dem Thema [Überschleifmodus](#), Seite 146, entnommen werden.

Überschleifradius CONT_SLOPE

Der Maximal mögliche Radius zum Überschleifen ist die halbe Strecke zwischen 2 Punkten. Reichen die Distanzen zwischen den zu überschleifenden Punkten nicht aus, um das programmierte Überschleifkriterium einzuhalten, wird von der Steuerung automatisch auf den maximal möglichen Überschleifbereich reduziert. Anders als bei der Slope-Art PROG_SLOPE sind asymmetrische Überschleifradien möglich. Dabei gilt: Zwischen mehreren zu überschleifenden Sätzen ist der maximale Überschleifradius die jeweilige Hälfte des Abstandes zwischen 2 Punkten. Bei Start bzw. Endsätzen einer Bewegung kann die ganze Distanz zum Überschleifen verwendet werden.

RCL-Bewegungsanweisungen

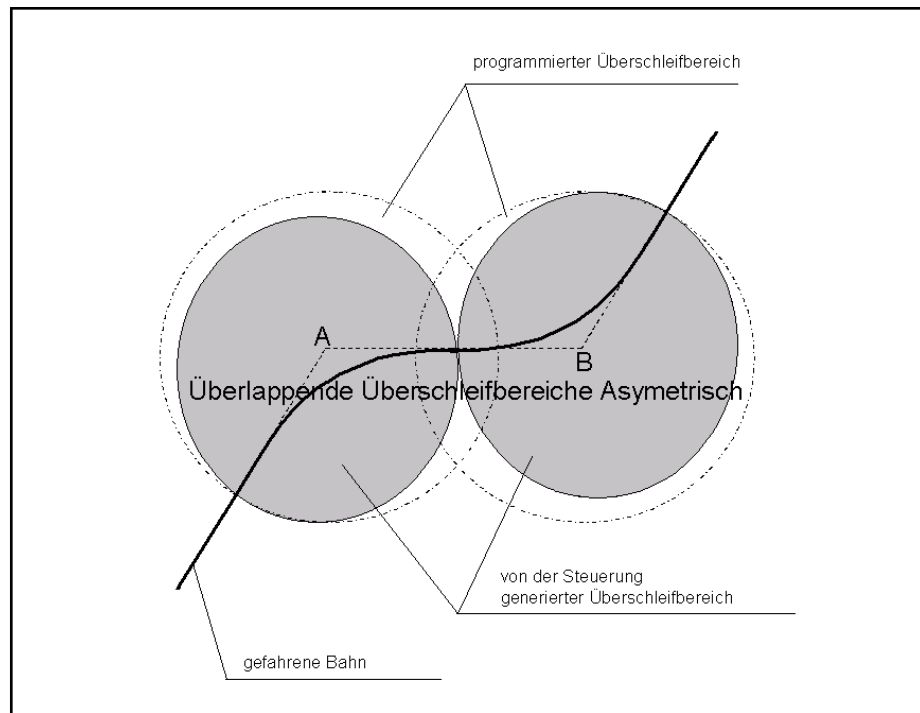


Abb. 12-39: Überlappende Überschleifbereiche

Erklärung zum Bild:

Überlappende Überschleifbereiche werden von der Steuerung automatisch so reduziert, dass sich der hierfür maximal mögliche Überschleifbereich ergibt. Bei CONT_SLOPE ist es möglich, dass sich asymmetrische Überschleifbereiche ergeben. Mit der durchgezogenen Linie ist der tatsächliche Pfad dargestellt. Weitere Einzelheiten zum Wirkradius beim Überschleifen können dem Thema [Überschleifmodus](#), Seite 146, entnommen werden.



Die Bandsynchronisation mit der Satzübergangsart PROGR_SLOPE und CONT_SLOPE wird zur Zeit nicht unterstützt.

Überschleifen bei Wechsel der Interpolationsart

Das Überschleifen bei aktivem PROGR_SLOPE ist auch beim Wechsel der Interpolationsarten wirksam. Zum Beispiel kann mit PTP auf einen Punkt zu und LINEAR von einem Punkt weg verfahren werden. Umgekehrt wäre dies auch denkbar. Vor dem Eintreten und nach dem Austreten aus der Überschleifhülle wird von der ursprünglich programmierten Bahn nicht abgewichen. Hierbei ist es nicht ausschlaggebend ob von PTP nach Linear oder umgekehrt verfahren wird.



Bei der Betriebsart CONT_SLOPE wird nur bei der Bewegungskombination Linear - Linear und PTP - PTP das Überschleifen unterstützt. Bei Wechsel der Interpolationsart findet kein Überschleifen statt.

Räumliches Überschleifen bei BLOCK_SLOPE

PTP Überschleifen bei BLOCK_SLOPE

Bei der Slope-Art BLOCK_SLOPE wird der nächste Bewegungssatz in Abhängigkeit von A-0-0230, Überschleifdistanz PTP und dem im RCL-Pro-

RCL-Bewegungsanweisungen

gramm programmierten R_PTP Faktor verfrüht gestartet. Aus den für jede Achse bezüglich ihrer Endposition angegebenen Distanzen, in Grad oder mm, wird ein gemeinsamer Zeitpunkt bestimmt, bei dem der darauffolgende Bewegungssatz gestartet wird. Beide Sätze werden nun parallel ausgeführt. Die Bahngeschwindigkeit resultiert aus der geometrischen Addition der Einzelachsgeschwindigkeiten. Hierdurch ist gewährleistet, dass der Werkzeugarbeitspunkt des Roboters nach Austritt aus dem Überschleifbereich, die ursprünglich ohne Überschleifen verfahrenene Bahn, wieder erreicht. Um ein gleichzeitiges Aktivieren von mehr als zwei Bewegungssätzen zu verhindern, wird von der Steuerung automatisch der früheste Zeitpunkt des Starts auf die Mitte der vorherigen Bewegung begrenzt. Ein gestarteter Bewegungssatz wird bis zur Mitte des darauffolgenden Satzes beendet. Je nach wirksamen Überschleifmodus kann das beschriebene Verhalten bezüglich der erwähnten Distanzen abweichen. Weitere Einzelheiten zum Wirkradius beim Überschleifen können dem Thema [Überschleifmodus, Seite 146](#), entnommen werden.

Folgendes Beispiel soll das Fahrverhalten verdeutlichen.

Zugrunde gelegt wird ein kartesisches System mit 3 Positionierachsen, wobei zunächst die Z-Achse aus dem Arbeitsbereich nach oben verfahren werden soll, danach werden die Achsen X und Y verfahren um dann wieder mit Z-Achse in den Arbeitsbereich einzutauchen.

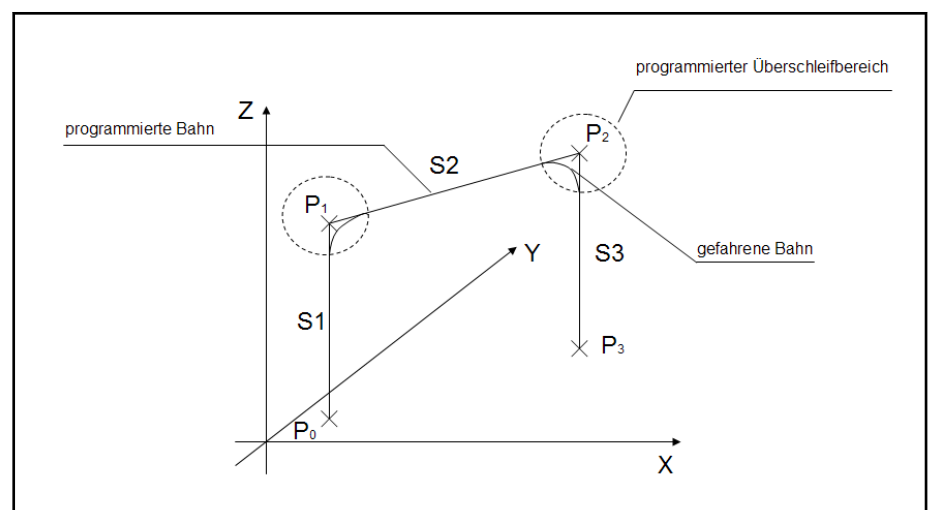


Abb. 12-40: Bahnverlauf der Pick- und Place-Anwendung

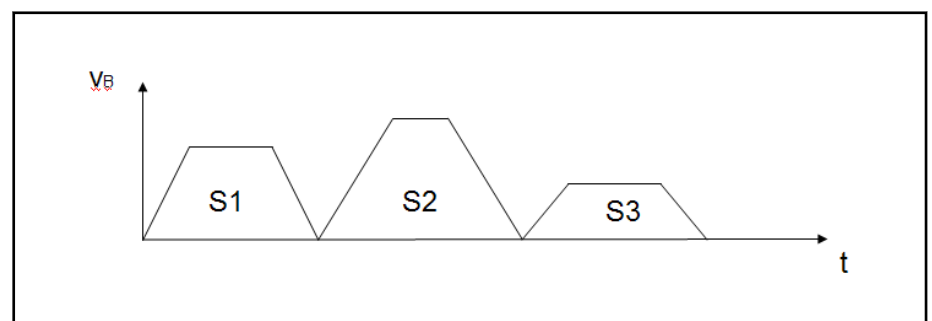


Abb. 12-41: Bahngeschwindigkeitsverlauf der Pick- und Place-Anwendung ohne Überschleifen

RCL-Bewegungsanweisungen

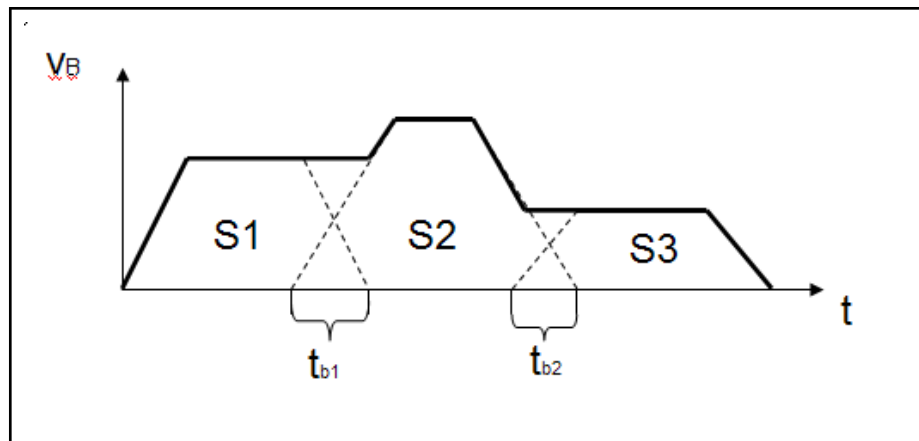


Abb. 12-42: Bahngeschwindigkeitsverlauf der Pick- und Place-Anwendung mit Überschleifen

Gewonnene Zeit: $t_{b1} + t_{b2}$.

Linear-Überschleifen bei BLOCK_SLOPE

Das Bahn-Überschleifen mit BLOCK_SLOPE wirkt, bezogen auf die Bahn, prinzipiell wie das PTP-Überschleifen mit BLOCK_SLOPE.

Überschleifen bei Wechsel der Interpolationsart

Bei aktivem BLOCK_SLOPE wird bei Interpolationsartwechsel sowohl von LINEAR nach PTP als auch von PTP nach LINEAR überschleift.

Spätes Überschleifen

Der Mechanismus des "Späten Überschleifens" bezieht sich auf das räumliche Überschleifen sowohl in BLOCK_SLOPE, CONT_SLOPE als auch Progr_SLOPE.

In der Praxis kommt es häufig vor, dass der Zielpunkt der Verfahrensbewegung erst zu einem späten Zeitpunkt bekannt gemacht wird. Um den Bewegungsablauf zeitlich zu optimieren kann der Roboter bereits vor diesem Zeitpunkt zum Zwischenpunkt kommandiert werden. Wird auf dem Weg zum Zwischenpunkt der Zielpunkt bekannt gemacht, startet das späte Überschleifen. Dabei wird gegebenenfalls der programmierte Radius reduziert auf zirka den Abstand zwischen der aktuellen Position und dem Zwischenpunkt.

RCL-Bewegungsanweisungen

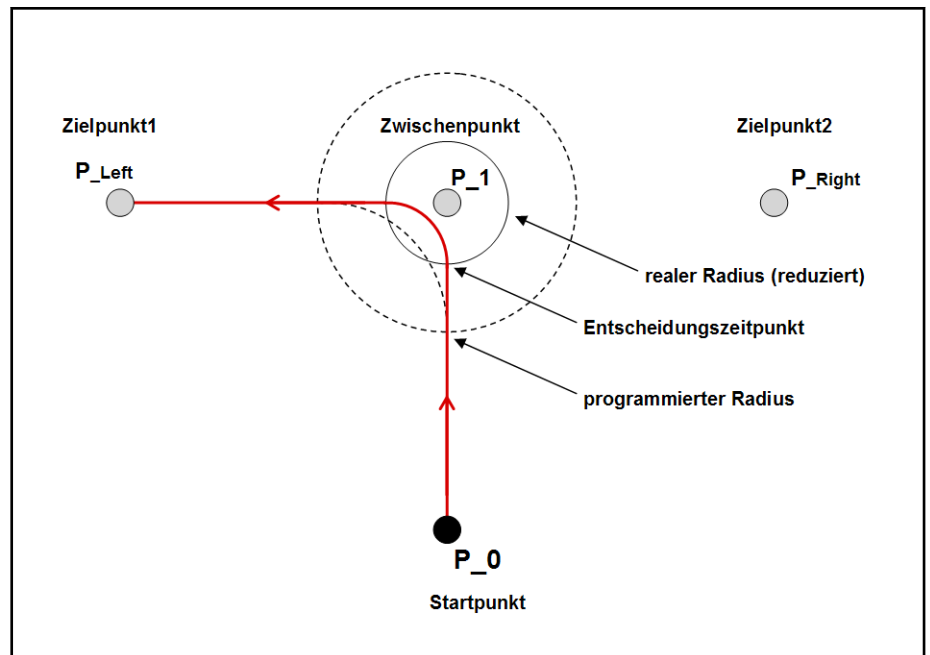


Abb. 12-43: Spätes Überschleifen bei PROGR_SLOPE

Das obere Beispiel zeigt einen Bewegungsablauf, der mit Hilfe des "Spätes Überschleifens" erst zu einem späten Zeitpunkt vollständig definiert wird. Der Startpunkt P_0 wird zunächst angefahren. Die Bewegung wird mit dem Anfahren des Zwischenpunktes P_1 begonnen. Während des Verfahrens wird zu einem späten Zeitpunkt entschieden den Punkt "P_Left" anzufahren. Da die Bewegung bereits in einem fortgeschrittenen Zustand ist und der ursprünglich programmierte Radiusbereich bereits überschritten wurde, wird der Radius automatisch reduziert.

Programm:

```

PROGRAM LBlend                                     ;(1)
INPUT BINARY : 1001 = bAsyncGo                    ;(2) "Time of commanding" signal
INPUT BINARY : 1002 = bAsyncIn                    ;(3) Decision whether to turn left or right
POINT : P_0, P_1, P_Left, P_Right                ;(4) Definition of sample points
BEGIN                                              ;(5)
;;INT=LINEAR                                       ;(6) Default MOVE is LINEAR

P_0 = (0,0)                                       ;(7) Initial Point
P_1 = (0,100)                                    ;(8) Intermediate Point
P_Left = (-100,100)                              ;(9) Turn Left End Point
P_Right = (100,100)                              ;(10) Turn Right End Point

PROGR_SLOPE                                       ;(11) Activate Program Slope

A = 10000                                         ;(12)
V = 2000                                         ;(13)
R = 20                                           ;(14)

MOVE TO P_0                                       ;(15) Move to the Initial Point
MOVE VIA P_1                                     ;(16) Move to the first intermediate point

; Decide whether to turn left or right
; By the time when this decision is done, switch on the bAsyncGo input.
;
; The program shall wait here for this plc signal.
WAIT UNTIL bAsyncGo                             ;(17)

; The decision was made. Check the result.

; 1: Left, 0: Right
IF bAsyncIn = 1
THEN
MOVE TO P_Left
ELSE

```

RCL-Bewegungsanweisungen

MOVE TO P_Right

PROGRAM_END

Überschleifmodus

Allgemeines



Dieses Überschleifverhalten gilt sowohl für BLOCK_SLOPE als auch für PROGR_SLOPE. Das Überschleifen mit Punkt zu Punkt Bewegungen (PTP) sowie Linearinterpolation (LINEAR) ist gleichermaßen implementiert. Bei CONT_SLOPE gilt das Verhalten nur für das Überschleifen bei Linearinterpolation (LINEAR).

Bei vielen Anwendungen ist es vorteilhaft, dass der Überschleifradius der ersten und letzten Bewegung nicht durch die halbe Wegstrecke der kürzesten Distanz begrenzt wird (obere Kurve, blau dargestellt).

Alternativ ist es ab der Version MLC-FW 10V06 möglich, die gesamte Wegstrecke der kürzesten Distanz als Überschleifradius zu verwenden, wenn dies geometrisch möglich ist (untere Kurve, grün dargestellt).

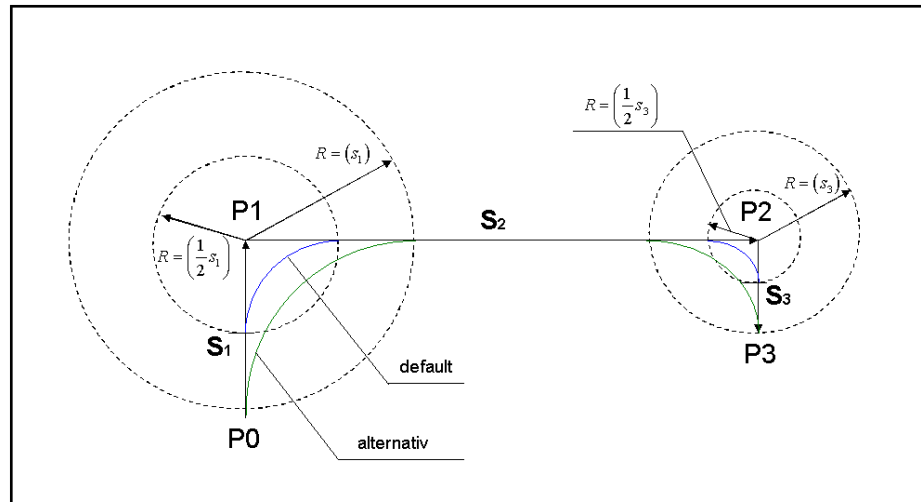


Abb. 12-44: Anpassung des wirksamen Radius mittels Überschleifmodus



Eine Überlappung der Überschleifradien ist in beiden Fällen nicht zulässig und wird durch die Steuerung verhindert. In diesem Fall

wird der Wirkradius auf $R_w = \min \left(R, \frac{1}{2} s_1, \frac{1}{2} s_2 \right)$ begrenzt.



Der Begriff „**halbe Distanz**“ ist aus technischen Gründen etwas weniger als die halbe Distanz (1/2,3). Aus gleichem Grund ist die „**ganze Distanz**“ ebenfalls etwas kleiner (1/1,1).

Die Programmierung dieser Funktion kann sowohl in der SPS als auch im RCL-Programm erfolgen. Es wird jedoch ausdrücklich empfohlen, nicht beide Zugriffsmöglichkeiten gleichzeitig einzusetzen.

Siehe auch [Robot Control-Funktionalität, programmiert auf der SPS, Seite 231](#).

RCL-Bewegungsanweisungen

SPS-Programmierung "Überschleifmodus" siehe ML_Robot, Bibliotheksbeschreibung:

- SPS-Funktion ML_KinGetDistanceBlendOption,
- SPS-Funktion ML_KinSetDistanceBlendOption.

RCL-Programmierung "Überschleifmodus"

- [SPC_FCT: 59, Überschleifmodus abfragen, Seite 108,](#)
- [SPC_FCT: 60, Überschleifmodus setzen, Seite 107](#)

13 Einrichtebetrieb

13.1 Einrichtebetrieb, Allgemeines

Dem Anwender steht eine Schnittstelle zur SPS zur Einrichtung der Kinematik(en) zur Verfügung.

Der Datenaustausch erfolgt dabei über einen Shared Memory zwischen SPS und Motion. Die Daten sind im ARRAY KinData je nach Steuerungstyp für die maximale Anzahl an Kinematiken (Achsgruppen) abgelegt.

Für jede einzelne Kinematik steht ihre Struktur ML_KINEMATICSDATA_SM, ML_Robot.library, zur Verfügung.

Innerhalb dieser Struktur sind die erforderlichen Elemente zum Tippen entlang der einzelnen Achsen unter "Kommandoschnittstelle zum Tippen"

stJogCmd_q :ML_KIN_JOG; // x-x-xxxx -RW- Command interface for jog mode

zu finden.

Siehe auch [Robot Control-Funktionalität, programmiert auf der SPS, Seite 231](#).

Einrichtebetrieb

13.2 Tippen in unterschiedlichen Koordinatensystemen

13.2.1 Einzustellende Parameter für das Tippen in Achskoordinaten

Kinematikachsen können in Achskoordinaten von Hand verfahren werden. Um eine einzelne Achse im Einrichtbetrieb zu tippen müssen je nach Tippmodus die entsprechenden Parameter eingestellt werden.

Inkrementelles Tippen, Distanz [mm]

Kurz:	A-0-0080, Jog-Distanz kurz	
	Geschwindigkeit [mm/s] bzw. [inch/s]	
	Schnell:	A-0-0083, Jog-Geschwindigkeit schnell
	Beschleunigung [mm/s²] bzw. [inch/s²]	
	Beschleunigung:	A-0-0084, Jog-Beschleunigung
	Bremsverzögerung:	A-0-0085, Jog-Bremsverzögerung
	Ruck [mm/s³] bzw. [inch/s³]	
	Beschleunigungsruck:	A-0-0086, Jog-Beschleunigungsruck
	Bremsruck:	A-0-0087, Jog-Bremsruck
Lang:	A-0-0081, Jog-Distanz lang	
	Geschwindigkeit [mm/s] bzw. [inch/s]	
	Schnell:	A-0-0083, Jog-Geschwindigkeit schnell
	Beschleunigung [mm/s²] bzw. [inch/s²]	
	Beschleunigung:	A-0-0084, Jog-Beschleunigung
	Bremsverzögerung:	A-0-0085, Jog-Bremsverzögerung
	Ruck [mm/s³] bzw. [inch/s³]	
	Beschleunigungsruck:	A-0-0086, Jog-Beschleunigungsruck
	Bremsruck:	A-0-0087, Jog-Bremsruck

Kontinuierliches Tippen, Geschwindigkeit [mm/s]

Langsam:	A-0-0082, Jog-Geschwindigkeit langsam	
	Beschleunigung [mm/s²] bzw. [inch/s²]	
	Beschleunigung:	A-0-0084, Jog-Beschleunigung
	Bremsverzögerung:	A-0-0085, Jog-Bremsverzögerung
	Ruck [mm/s³] bzw. [inch/s³]	
	Beschleunigungsruck:	A-0-0086, Jog-Beschleunigungsruck
Schnell:	A-0-0083, Jog-Geschwindigkeit schnell	
	Beschleunigung [mm/s²] bzw. [inch/s²]	
	Beschleunigung:	A-0-0084, Jog-Beschleunigung
	Bremsverzögerung:	A-0-0085, Jog-Bremsverzögerung
	Ruck [mm/s³] bzw. [inch/s³]	
	Beschleunigungsruck:	A-0-0086, Jog-Beschleunigungsruck
	Bremsruck:	A-0-0087, Jog-Bremsruck

13.2.2 Einstellende Parameter für das Tippen in Weltkoordinaten

Kinematikachsen können in Weltkoordinaten von Hand verfahren werden. Um entlang einer einzelnen Koordinate im Einrichtbetrieb zu tippen müssen je nach Tippmodus die entsprechenden Parameter eingestellt werden.

Inkrementelles Tippen, Distanz [mm] bzw. [inch]

Kurz: K-0-0080, Jog-Distanz kurz im Weltkoordinatensystem (WC)
Geschwindigkeit [mm/s] bzw. [inch/s]
Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)

Beschleunigung [mm/s²] bzw. [inch/s²]

Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Lang: K-0-0081, Jog-Distanz lang im Weltkoordinatensystem (WC)
Geschwindigkeit [mm/s] bzw. [inch/s]
Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)

Beschleunigung [mm/s²] bzw. [inch/s²]

Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Kontinuierliches Tippen, Geschwindigkeit [mm/s] bzw [inch/s]

Langsam: K-0-0082, Jog-Geschwindigkeit langsam im Weltkoordinatensystem (WC)
Beschleunigung [mm/s²] bzw. [inch/s²]
Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)
Beschleunigung [mm/s²] bzw. [inch/s²]
Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Einrichtebetrieb

13.2.3 Tippen in Weltkoordinaten

Kinematikachsen können in Weltkoordinaten von Hand verfahren werden. Um entlang einer einzelnen Koordinate im Einrichtbetrieb zu tippen, müssen je nach Tippmodus die entsprechenden Parameter eingestellt werden.

Inkrementelles Tippen, Distanz [mm] bzw [inch]

Kurz: K-0-0080, Jog-Distanz kurz im Weltkoordinatensystem (WC)
Geschwindigkeit [mm/s] bzw. [inch/s]
 Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)

Beschleunigung [mm/s²] bzw. [inch/s²]

Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
 Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
 Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Lang: K-0-0081, Jog-Distanz lang im Weltkoordinatensystem (WC)
Geschwindigkeit [mm/s] bzw. [inch/s]
 Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)

Beschleunigung [mm/s²] bzw. [inch/s²]

Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
 Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
 Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Kontinuierliches Tippen, Geschwindigkeit [mm/s] bzw [inch/s]

Langsam: K-0-0082, Jog-Geschwindigkeit langsam im Weltkoordinatensystem (WC)
Beschleunigung [mm/s²] bzw. [inch/s²]
 Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
 Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
 Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

Schnell: K-0-0083, Jog-Geschwindigkeit schnell im Weltkoordinatensystem (WC)
Beschleunigung [mm/s²] bzw. [inch/s²]
 Beschleunigung: K-0-0084, Jog-Beschleunigung im Weltkoordinatensystem (WC)
 Bremsverzögerung: K-0-0085, Jog-Bremsverzögerung im Weltkoordinatensystem (WC)

Ruck [mm/s³] bzw. [inch/s³]

Beschleunigungsruck: K-0-0086, Jog-Beschleunigungsruck im Weltkoordinatensystem (WC)
 Bremsruck: K-0-0087, Jog-Verzögerungsruck im Weltkoordinatensystem (WC)

14 RCL-Parameter

14.1 Achsbezogene Parameter, genutzt durch Kinematiken einer MLC

14.1.1 Allgemeines

Grenzwerte und achsbezogene Daten werden in A-Parametern der jeweiligen Achse abgelegt.

Details siehe "IndraMotion MLC Parameterbeschreibung".

14.1.2 A-0-0028, Fahrbereichsgrenzschalter

Funktion In diesem Parameter wird das Signalverhalten der Fahrbereichsgrenzschalter-Eingänge und das Verhalten des Antriebs bezüglich Fahrbereichsüberschreitung festgelegt.

Aufbau

Bit	Bezeichnung/ Funktion
0	Signalverhalten Fahrbereichsgrenzschalter Eingänge (nur reale Achsen) 0: nicht invertiert, hohe Spannung wird als Fahrbereich überschritten gewertet, 1: invertiert, niedrige Spannung wird als Fahrbereich überschritten gewertet.
1	Aktivierung Fahrbereichsgrenzschalter (nur reale Achsen) 0: nicht aktiv, 1: aktiv.
2	Behandlung der Fahrbereichsüberschreitung 0: als Fehler (gemäß Fehlerreaktion), 1: als fatale Warnung (Stillsetzung).
3..15	reserviert

14.1.3 A-0-0029, Positionspolaritäten

Funktion Nur reale Achsen gilt:

Mit den ersten 4 Bits dieses Parameter können die Polaritäten der angegebenen Lagedaten invertiert werden. Die Polaritäten werden außerhalb der Regelstrecke umgeschaltet, d. h. am Eingang und Ausgang der Regelstrecke. Der Parameter wirkt hierzu auf den S-0-0055 des Antriebsgerätes.



Die Festlegung der Lage-Polarität muss bei der Erstinbetriebnahme einer Achse vor der Herstellung des Maschinen-Nullpunktbezugs für die Mess-Systeme durchgeführt werden, da sich bei Änderung der Polarität andere Lageistwerte ergeben.

Bei rotatorischen (drehenden) Motoren gilt:

Drehung der Motorabtriebswelle im Uhrzeigersinn bei positiver Lagesollwertdifferenz und positiver Polarität.

Bei Linearmotoren gilt:

Bewegung des Primärteils in Richtung der Kabelanschluss-Seite bei positiver Lagesollwertdifferenz und positiver Polarität.

Siehe auch: A-0-0048 und A-0-0049

RCL-Parameter

Funktion Mit dem Bit 4 dieses Parameters werden die Software-Positionsgrenzwert, "A-0-0030, Positionsgrenzwert, positiv" bzw. "A-0-0031, Positionsgrenzwert, negativ" aktiviert bzw. deaktiviert.

Aufbau

Bit	Bezeichnung/ Funktion
0	Lage-Sollwert 0: Polarität positiv 1: Polarität negativ
1	Lage-Sollwert additiv 0: Polarität positiv 1: Polarität negativ
2	Lage-Istwert 1 0: Polarität positiv 1: Polarität negativ
3	Lage-Istwert 2 0: Polarität positiv 1: Polarität negativ
4	Positionsgrenzwert 0: nicht aktiv, 1: aktiv.
15..5	reserviert

14.1.4 A-0-0030, Positionsgrenzwert, positiv

Funktion Der erlaubte Verfahrbereich einer Achse kann durch Positionsgrenzwerte festgelegt werden. In diesen Parameter wird der maximal erlaubte positive Positionswert (positive Verfahrrichtung) eingetragen.

Der positive Grenzwert ist nur aktiv, wenn alle Positionsdaten auf den Achsnullpunkt bezogen sind, d.h. die Achse referenziert ist "A-0-0021, Achsstatus, Bit 10".

Verwendung Wenn die "A-0-2200, Sollposition" der Achse außerhalb des positiven Positionsgrenzwertes gesetzt ist, wird das Fehlerbit im "A-0-0021, Achsstatus, Bit 9" TRUE und erzeugt den Fehler "F2112053, Zielposition außerhalb des Verfahrbereiches".

Überschreitet der "A-0-0100, Positionswert" den "A-0-0030, Positionsgrenzwert, positiv" oder den "A-0-0031, Positionsgrenzwert, negativ", setzt die Achse das Fehlerbit im "A-0-0021, Achsstatus, Bit 9" und erzeugt den Fehler "F2116029, Positionsgrenzwert positiv überschritten" bzw. "F2116030, Positionsgrenzwert negativ überschritten".

Siehe auch Funktionsbeschreibung: "Positionsgrenzwerte".

14.1.5 A-0-0031, Positionsgrenzwert, negativ

Funktion Der erlaubte Verfahrbereich einer Achse kann durch Positionsgrenzwerte festgelegt werden. In diesen Parameter wird der maximal erlaubte negative Positionswert (negative Verfahrrichtung) eingetragen.

Der negative Grenzwert ist nur aktiv, wenn alle Positionsdaten auf den Achsnullpunkt bezogen sind, d.h. die Achse referenziert ist "A-0-0021, Achsstatus, Bit 10".

- Verwendung** Wenn die "A-0-2200, Sollposition" der Achse außerhalb des negativen Positionsgrenzwertes gesetzt ist, wird das Fehlerbit im "A-0-0021, Achsstatus, Bit 9" TRUE und erzeugt den Fehler "F2112053, Zielposition außerhalb des Verfahrbereiches".
- Überschreitet der "A-0-0100, Positionswert" den "A-0-0030, Positionsgrenzwert, positiv" oder den "A-0-0031, Positionsgrenzwert, negativ", setzt die Achse das Fehlerbit im "A-0-0021, Achsstatus, Bit 9" und erzeugt den Fehler "F2116029, Positionsgrenzwert, positiv, überschritten" bzw. "F2116030 Positionsgrenzwert, negativ, überschritten".
- Siehe auch Funktionsbeschreibung: "Positionsgrenzwerte".

14.1.6 A-0-0032, Geschwindigkeitsgrenzwert, positiv

- Funktion** Der Parameter beschreibt die maximal zulässige Geschwindigkeit in positiver Richtung.
- Verwendung** Wenn der "A-0-0102, Geschwindigkeitswert" den Betrag der Grenzwerte "A-0-0032, Geschwindigkeitsgrenzwert, positiv" bzw. "A-0-0033, Geschwindigkeitsgrenzwert, negativ" überschreitet, setzt die Achse das Warnungsbit "A-0-0021, Achsstatus, bit 8" und generiert die Warnung "E0112063, Geschwindigkeit größer Grenzwert A-0-0032 / A-0-0033".
- Der eingegebene Wert dient als Maximalwert für alle Geschwindigkeitsgrößen in positiver Richtung.
- Siehe auch Funktionsbeschreibung "Geschwindigkeitsbegrenzung".

14.1.7 A-0-0033, Geschwindigkeitsgrenzwert, negativ

- Funktion** Der Parameter beschreibt die maximal zulässige Geschwindigkeit in negativer Richtung.
- Verwendung** Wenn der "A-0-0102, Geschwindigkeitswert" den Betrag der Grenzwerte "A-0-0032, Geschwindigkeitsgrenzwert, positiv" bzw. "A-0-0033, Geschwindigkeitsgrenzwert negativ" überschreitet, setzt die Achse das Warnungsbit "A-0-0021, Achsstatus, bit 8" und generiert die Warnung "E0112063, Geschwindigkeit größer Grenzwert A-0-0032 / A-0-0033".
- Der eingegebene Wert dient als Maximalwert für alle Geschwindigkeitsgrößen in negativer Richtung.
- Siehe auch Funktionsbeschreibung "Geschwindigkeitsbegrenzung".

14.1.8 A-0-0034, Beschleunigungsgrenzwert, bipolar

- Funktion** Der Parameter beschreibt die maximal zulässige Beschleunigung symmetrisch in beide Richtungen (Beschleunigen und Bremsen).
- Verwendung** Wenn der Parameter "A-0-0103, Beschleunigungswert" den Parameter "A-0-0034, Beschleunigungsgrenzwert, bipolar" überschreitet, setzt die Achse das Warnungsbit "A-0-0021, Achsstatus, bit 8" und erzeugt die Warnung "E0112039, Maximale Beschleunigung überschritten".

14.1.9 A-0-0036, Ruckgrenzwert, bipolar

- Funktion** Der Parameter beschreibt die maximal zulässige Beschleunigungsänderung pro Zeit (= Ruck) symmetrisch in beide Richtungen (Beschleunigen und Bremsen).
- Dieser Grenzwert wirkt gegen den kommandierten Ruck im Parameter "A-0-0216, Ruck".

RCL-Parameter



Der Parameter begrenzt die Beschleunigungsänderung pro Zeit bei "Antrieb-Halt" und bei Kommandos, die sich selbst Positionswerte generieren.

Der Wert "0" schaltet den Ruckgrenzwert ab.

Er wird beeinflusst durch den Parameter "A-0-0053, Wichtungsart für Beschleunigungsdaten".

14.1.10 A-0-0038, Drehmoment-/Kraftgrenzwert, bipolar

Funktion Durch diesen Parameter kann der Drehmoment-/Kraftsollwert zum Schutz für mechanische Komponenten im Antriebsstrang auf zulässige Maximalwerte begrenzt werden.

Der Parameter wirkt bipolar, d.h. positive und negative Sollwerte werden auf den eingetragenen Wert begrenzt. Die Begrenzung wirkt auf die Summe aller Drehmomentsollwerte, d.h. auf die Summe der Sollwertanteile des Geschwindigkeitsreglerausgangs und der Sollwertanteile aus beschleunigungsabhängigen Vorsteuerungen ("S-0-0348, Verstärkung Beschleunigungsvorsteuerung"; Geschw.-Regelkreis: "P-0-1126, Beschleunigungsvorsteuerung").

Die Einheit von "A-0-0038" ist abhängig von der eingestellten Wichtung.

Siehe auch Parameterbeschreibung "A-0-0050, Wichtungsart für Drehmoment-/Kraftdaten"



Der wirksame Grenzwert wird in "P-0-0049, Wirksamer Momenten-/Kraft-Sollwert" angezeigt. Dieser Wert kann gegenüber "A-0-0038" aufgrund eines kleineren Wertes in "P-0-0109, Spitzendrehmoment-/ Kraftbegrenzung" oder aufgrund einer Strombegrenzung durch entsprechende Auslastung von Verstärker oder Motor eingeschränkt sein.

Soll dem Antrieb im stationären Betrieb (konstante Drehzahl) ein geringeres Drehmoment zur Verfügung stehen als bei Beschleunigungsvorgängen, so kann dies über entsprechende Werte von "S-0-0082, Drehmoment-/ Kraft-Grenzwert positiv" und "S-0-0083, Drehmoment- / Kraft-Grenzwert negativ" in Verbindung mit Beschleunigungsvorsteuerung (S-0-0348 oder P-0-1126) erreicht werden.

Der Drehmoment-/Kraft-Grenzwert bipolar wirkt auch auf "C1300 Kommando: Fahren auf Festanschlag".

Der Wert von "A-0-0038" ist ein Kriterium für die Quittierung von "C1300".

Siehe auch Funktionsbeschreibung "Drehmoment-/Kraftregelung".

14.1.11 A-0-0048, Geschwindigkeitspolaritäten

Funktion In diesem Parameter kann die Polarität der Geschwindigkeitsdaten auf die Anwendung bezogen umgeschaltet werden.

Die Polarität wird nicht innerhalb, sondern außerhalb (am Eingang und Ausgang) einer Regelstrecke umgeschaltet.

RCL-Parameter

Aufbau

Bit	Bezeichnung/ Funktion
0	Geschwindigkeitssollwert 0: Polarität positiv 1: Polarität negativ
1	Positionssollwert additiv 0: Polarität positiv 1: Polarität negativ
2	Geschwindigkeitsistwert 0: Polarität positiv 1: Polarität negativ
15..3	reserviert



Die Bits 1 und 2 sind Kopien von Bit 0.

Änderungen sind nur in Bit 0 wirksam. Unterschiedliche Einstellungen der einzelnen Bits sind nicht zulässig!

Bei **rotatorischen (drehenden) Motoren** gilt: Drehung der Motorabtriebswelle im Uhrzeigersinn führt zu positivem Geschwindigkeitsistwert (positive Polarität).

Bei **Linearmotoren** gilt: Bewegung des Primärteils in Richtung der Kabelanschluss-Seite führt zu positivem Geschwindigkeitsistwert (positive Polarität).

14.1.12 A-0-0049, Drehmoment-/Kraft-Polaritäten

Funktion

In diesem Parameter kann die Polarität der angegebenen Drehmoment-/Kraftdaten auf die Anwendung bezogen invertiert werden.

Die Polarität wird außerhalb der Regelstrecke umgeschaltet, d.h. am Eingang und Ausgang der Regelstrecke.

Aufbau

Bit	Bezeichnung/ Funktion
0	Drehmomentsollwert 0: Polarität positiv 1: Polarität negativ
1	Drehmomentsollwert additiv 0: Polarität positiv 1: Polarität negativ
2	Drehmomentistwert 0: Polarität positiv 1: Polarität negativ
15..3	reserviert



Die Bits 1 und 2 sind Kopien von Bit 0.

Änderungen sind nur in Bit 0 wirksam. Unterschiedliche Einstellungen der einzelnen Bits sind nicht zulässig!

Bei **rotatorischen (drehenden) Motoren** gilt: Drehung der Motorabtriebswelle im Uhrzeigersinn bei positivem Drehmoment und positiver Polarität.

RCL-Parameter

Bei **Linearmotoren** gilt: Bewegung des Primärteils in Richtung der Kabelanschluss-Seite bei positiver Kraft und positiver Polarität.

14.1.13 A-0-0080, Joggen, kurze Distanz

Funktion In diesem Parameter ist die kurze Distanz für das inkrementelle Joggen der Achse hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm,
für rotatorisch gewichtete Achsen in Grad.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 1.0 mm bzw. Grad.

Verwendung Dieser Parameter wird von der Robot-Control für das inkrementelle Joggen der Achse verwendet.

Wenn die Achse mit einer anderen Achse gekoppelt ist (z.B. Hub-Drehspindel) wird die andere Achse so angesteuert, dass ihre logische Position unverändert bleibt.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das inkrementelle Joggen mit kurzer Distanz aktivieren:

Z.B. Inkrementelles Joggen mit kurzer Distanz der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SMALL;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die maximale Geschwindigkeit steht in A-0-0083 (für inkrementelles Joggen immer schnell).
- Die maximale Beschleunigung steht in A-0-0084 und
- die maximale Bremsverzögerung steht in A-0-0085.
- Der maximale Ruck für die Beschleunigung steht in A-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in A-0-0087.

Ein vorzeitiges Beenden der Jog-Bewegung ist durch Setzen von enJogMode = KIN_JOG_STOP möglich.

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.1.14 A-0-0081, Joggen, lange Distanz

Funktion In diesem Parameter ist die lange Distanz für das inkrementelle Joggen der Achse hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm,
für rotatorisch gewichtete Achsen in Grad.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 10.0 mm bzw. Grad.

Verwendung Dieser Parameter wird von der Robot-Control für das inkrementelle Joggen der Achse verwendet.

Wenn die Achse mit einer anderen Achse gekoppelt ist (z.B. Hub-Drehspindel) wird die andere Achse so angesteuert, dass ihre logische Position unverändert bleibt.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das inkrementelle Joggen mit langer Distanz aktivieren:

Z.B. Inkrementelles Joggen mit langer Distanz der 4. Achse in negative Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_LARGE;
KinData[1].stJogCmd_q.wJogCmdNegative := 8; (*4. Bit für 4.Achse der Kinematik bzw. Gruppe*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

- Die Distanz für das Joggen der Achse mit langer Distanz steht in A-0-0081.
- Die maximale Geschwindigkeit steht in A-0-0083 (für inkrementelles Joggen immer schnell).
- Die maximale Beschleunigung steht in A-0-0084 und
- die maximale Bremsverzögerung steht in A-0-0085.
- Der maximale Ruck für die Beschleunigung steht in A-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in A-0-0087.

Ein vorzeitiges Beenden der Jog-Bewegung ist durch Setzen von enJogMode = KIN_JOG_STOP möglich.

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.1.15 A-0-0082, Joggen, langsame Geschwindigkeit

Funktion In diesem Parameter ist die langsame Geschwindigkeit für das kontinuierliche joggen der Achse hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s,
für rotatorisch gewichtete Achsen in Grad/s.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 10.0 mm/s bzw. Grad/s.

Verwendung Dieser Parameter wird von der Robot-Control für das kontinuierliche Joggen der Achse verwendet.

RCL-Parameter

Wenn die Achse mit einer anderen Achse gekoppelt ist (z.B. Hub-Drehspindel), wird die andere Achse so angesteuert, dass ihre logische Position unverändert bleibt.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das kontinuierliche Joggen mit langsamer Geschwindigkeit aktivieren:

Z.B. kontinuierliches Joggen mit langsamer Geschwindigkeit der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_SLOW;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die maximale Beschleunigung steht in A-0-0084 und
- die maximale Bremsverzögerung steht in A-0-0085.
- Der maximale Ruck für die Beschleunigung steht in A-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in A-0-0087.

Das kontinuierliche Joggen kann durch 2 Arten beendet werden:

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

und / oder

Programm:

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

14.1.16 A-0-0083, Joggen, schnelle Geschwindigkeit

Funktion In diesem Parameter ist die schnelle Geschwindigkeit für das kontinuierliche Joggen der Achse hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s,
für rotatorisch gewichtete Achsen in Grad/s.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 25.0 mm/s bzw. Grad/s.

Verwendung Dieser Parameter wird von der Robot-Control für das kontinuierliche **und inkrementelle** Joggen der Achse verwendet.

Wenn die Achse mit einer anderen Achse gekoppelt ist (z.B. Hub-Drehspindel) wird die andere Achse so angesteuert, dass ihre logische Position unverändert bleibt.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das kontinuierliche Joggen mit schneller Geschwindigkeit aktivieren:

Z.B. kontinuierliches Joggen mit schneller Geschwindigkeit der 4. Achse in negative Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys := KIN_CORD_SYS_JOINT_COORD;  
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_FAST;  
KinData[1].stJogCmd_q.wJogCmdNegative := 8; (*4. Bit für 4.Achse der Kinematik bzw. Gruppe *)  
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

- Die maximale Beschleunigung steht in A-0-0084 und
- die maximale Bremsverzögerung steht in A-0-0085.
- Der maximale Ruck für die Beschleunigung steht in A-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in A-0-0087.

Das kontinuierliche Joggen kann auf 2 Arten beendet werden:

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

und / oder

Programm:

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

14.1.17 A-0-0084, Joggen, Beschleunigung

Funktion In diesem Parameter ist die maximale Beschleunigung für das Joggen (inkrementell und kontinuierlich) der Achse hinterlegt.



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s²,
für rotatorisch gewichtete Achsen in Grad/s².
 - Wertebereich: > 0.0 (Fließkommazahl, muss ≤ A-0-0034 sein)
 - Default: 370 mm/s² bzw. Grad/s².

Verwendung Dieser Parameter wird von der Robot-Control für das Joggen der Achse verwendet. Die Jog-Beschleunigung gilt für alle Jog-Bewegungen (inkrementell und kontinuierlich). Die Jog-Bremsverzögerung wird im A-0-0085 angegeben. Die Bremsverzögerungs- und Beschleunigungswerte für das Joggen im Raumkoordinatensystem (WC) werden in K-0-0084 und K-0-0085 festgelegt.

14.1.18 A-0-0085, Joggen, Bremsverzögerung

Funktion In diesem Parameter ist die maximale Bremsverzögerung für das Joggen (inkrementell und kontinuierlich) der Achse hinterlegt.



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s²,
für rotatorisch gewichtete Achsen in Grad/s².
 - Wertebereich: > 0.0 (Fließkommazahl, muss ≤ A-0-0034 sein)
 - Default: 100 mm/s² bzw. Grad/s².

RCL-Parameter

- Verwendung** Dieser Parameter wird von der Robot-Control für das Joggen der Achse verwendet.
- Die Jog-Bremsverzögerung gilt für alle Jog-Bewegungen (inkrementell und kontinuierlich).
- Die Jog-Beschleunigung wird im A-0-0084 angegeben.
- Die Bremsverzögerungs- und Beschleunigungswerte für das Joggen im Raumkoordinatensystem (WC) werden in K-0-0084 und K-0-0085 festgelegt.

14.1.19 A-0-0086, Joggen, Beschleunigungsruck

- Funktion** In diesem Parameter ist der maximale Ruck bei Beschleunigung für das Joggen (inkrementell und kontinuierlich) der Achse hinterlegt (indirekte Definition der Beschleunigungsänderungszeit).



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s³,
für rotatorisch gewichtete Achsen in Grad/s³.
 - Wertebereich: ≥0.0 (Fließkommazahl, ≤ A-0-0036)
 - Default: 0.0 (keine Ruckbegrenzung).
- Verwendung** Dieser Parameter wird von der Robot-Control für das Joggen der Achse verwendet. Der Jog-Beschleunigungsruck gilt für alle Jog-Bewegungen (inkrementell und kontinuierlich).

Aus $\Delta a = j \cdot t$ kann man die Beschleunigungsänderungszeit $t = \frac{\Delta a}{j}$ bzw.

$$t = \frac{A-0-0084}{A-0-0086} \text{ errechnen.}$$

Der maximale Ruck j bei der Bremsverzögerung wird in A-0-0087 angegeben.

Die maximale Ruckwerte für das Joggen in Raumkoordinaten werden mit K-0-0086 und K-0-0087 appliziert.

14.1.20 A-0-0087, Joggen, Bremsruck

- Funktion** In diesem Parameter ist der maximale Ruck bei Bremsverzögerung für das Joggen (inkrementell und kontinuierlich) der Achse hinterlegt (indirekte Definition der Bremsverzögerungsänderungszeit).



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Einheit:
für translatorisch gewichtete Achsen in mm/s³,
für rotatorisch gewichtete Achsen in Grad/s³.
 - Wertebereich: ≥0.0 (Fließkommazahl, ≤ A-0-0036)
 - Default: 0.0 (keine Ruckbegrenzung).
- Verwendung** Dieser Parameter wird von der Robot-Control für das Joggen der Achse verwendet. Der Jog-Verzögerungsruck gilt für alle Jog-Bewegungen (inkrementell und kontinuierlich).

Aus $\Delta a = j \cdot t$ kann man die Verzögerungsänderungszeit $t = \frac{\Delta a}{j}$ bzw.
 $t = \frac{A-0-0085}{A-0-0087}$ errechnen.

Der maximale Ruck bei Beschleunigung wird in A-0-0086 angegeben.

Die maximale Ruckwerte für das Joggen in Raumkoordinaten werden mit K-0-0086 und K-0-0087 appliziert.

14.1.21 A-0-0224, Schnellhalt-Verzögerung

Funktion In diesem Parameter wird eingestellt, mit welchem maximalen Verzögerungswert die Achse bei

- Schnellhalt mit Rampe

stillgesetzt wird.

Die Einheit dieses Parameters wird durch "A-0-0053, Wichtungsart für Beschleunigungsdaten" festgelegt.

Siehe auch Funktionsbeschreibung: "Fehlerreaktion"

14.1.22 A-0-0225, Schnellhalt-Ruck

Funktion In diesem Parameter wird eingestellt, mit welchem maximalen Ruckwert die Achse bei

- Schnellhalt mit Rampe

stillgesetzt wird.

Die Einheit dieses Parameters wird durch "A-0-0053, Wichtungsart für Beschleunigungsdaten" festgelegt.

Siehe auch Funktionsbeschreibung: "Fehlerreaktion".



Dieser Parameter ist nur wirksam falls für die Achse "Interpolation in der Steuerung" gewählt wurde.

14.1.23 A-0-0230, Überschleifdistanz PTP

Funktion Dieser Parameter ist nur für Achsen relevant, die innerhalb einer Kinematik mit dem Befehl "MOVE PTP" verfahren werden.

Bei diesen Achsbewegungen kann die Stärke des Überschleifens bei Verfahrtsatzübergängen mit dem Faktor R_PTP programmiert werden.

Der Wert dieses Parameters gibt die Wegstrecke an, auf die sich der Faktor R_PTP bezieht.

Ist der Wert 0, kann die Achse nicht überschleift werden.

14.1.24 A-0-0240, Feedrate-Override

Funktion In diesem Parameter wird eingestellt, mit welchem Faktor die Geschwindigkeit der Achse verändert wird.

Ein Wert von 100% bedeutet keine Veränderung.

Bei einem Faktor von 0% wird die Warnung "E0112055, Feedrate-Override A-0-0240 = 0" ausgegeben.

RCL-Parameter

14.1.25 A-0-0241, Feedrate-Override-Steuerwort

Funktion Dieser Parameter wirkt auf den Feedrate-Override, ohne dass der Feedrate-Override-Parameter A-0-0240 verändert wird.

Aufbau

Bit	Bezeichnung/ Funktion
0	reserviert
1	Feedrate-Override = 0
15-2	reserviert

Verwendung **Bit 1:**

Mit dem Setzen des Bit 1 wird ein Feedrate-Override von 0 wirksam, ohne dass der eigentliche Parameterwert A-0-0240 verändert wird.



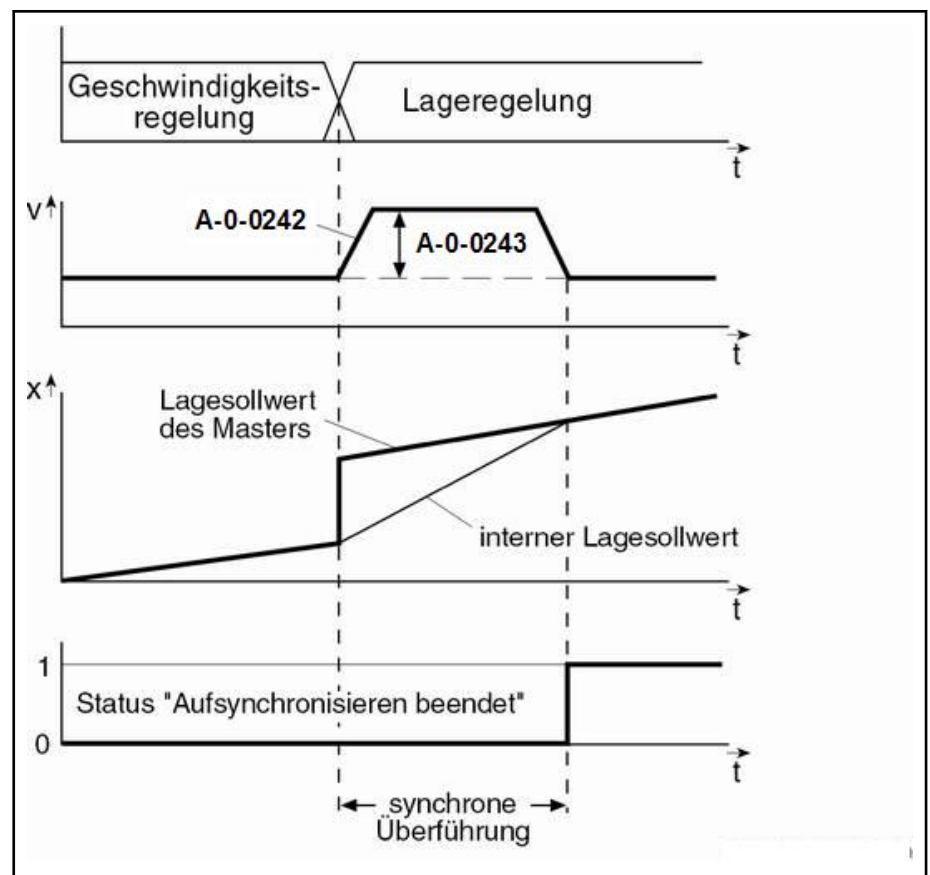
Feedrate-Override = 0 bedeutet "Halt", jedoch in diesem Fall ohne die Warnung "E0112055, Feedrate-Override A-0-0240 = 0"

14.1.26 A-0-0242, Lageregelung, Aufsynchronisierbeschleunigung

Funktion Der Parameter enthält den Wert der Beschleunigung, der beim Betriebsartenwechsel von Geschwindigkeitsregelung nach Positionsregelung, antriebsgeführt, beim Aufsynchronisieren berücksichtigt wird.

Der Antrieb sorgt beim Betriebsartenwechsel in die zyklische Positionsregelung für einen entsprechenden Aufsynchronisiervorgang, d.h. er erzeugt intern anhand der internen Aufsynchronisierparameter (A-0-0242, A-0-0243) einen sanften Übergang des internen Positionssollwertes von der aktuellen Istposition auf den neuen, von der Steuerung vorgegebenen Sollwertverlauf. Nach Abschluss des Aufsynchronisiervorganges folgt der Antrieb dann den vom Master im Motion-Takt vorgegebenen Positionssollwerten

Es wird antriebsintern sichergestellt, dass beim Wechseln der Betriebsart selbst bei sprunghaften Änderungen des Sollwertes eine synchronisierte Überführung erfolgt.



A-0-0242 Aufsynchronisierbeschleunigung
A-0-0243 Aufsynchronisiergeschwindigkeit
Abb. 14-1: Zeitdiagramm "Antriebsgeführter Betriebsartenwechsel"



Der Wert dieses Parameters entspricht im Defaultzustand dem Wert des Parameters A-0-0034, Beschleunigungsgrenzwert, bipolar.

Individuelle Werte können konfiguriert werden, wenn im Parameter A-0-0007, Achskonfiguration, Bit 4 gesetzt wird.

Verwendung Dieser Wert wird beim Phasenschalten von P2 nach BB zum Antrieb hin synchronisiert.

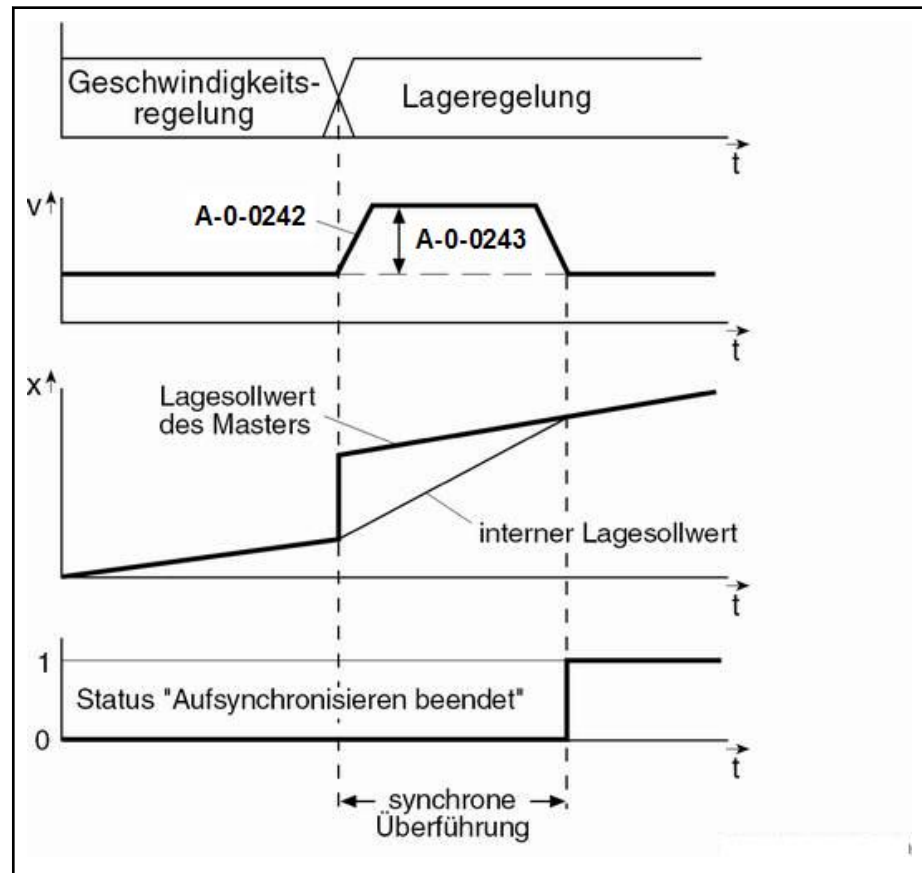
14.1.27 A-0-0243, Lageregelung, Aufsynchronisiergeschwindigkeit

Funktion Der Parameter enthält den Wert der Geschwindigkeit, der beim Betriebsartenwechsel von Geschwindigkeitsregelung nach Positionsregelung, antriebsgeführt, beim Aufsynchronisieren berücksichtigt wird.

Der Antrieb sorgt beim Betriebsartenwechsel in die zyklische Positionsregelung für einen entsprechenden Aufsynchronisiervorgang, d.h. er erzeugt intern anhand der internen Aufsynchronisierparameter (A-0-0242, A-0-0243) einen sanften Übergang des internen Positionssollwertes von der aktuellen Istposition auf den neuen, von der Steuerung vorgegebenen Sollwertverlauf. Nach Abschluss des Aufsynchronisiervorganges folgt der Antrieb dann den vom Master im Motion-Takt vorgegebenen Positionssollwerten

Es wird antriebsintern sichergestellt, dass beim Wechseln der Betriebsart selbst bei sprunghaften Änderungen des Sollwertes eine synchronisierte Überführung erfolgt.

RCL-Parameter



A-0-0242 Aufsynchronisierbeschleunigung

A-0-0243 Aufsynchronisiergeschwindigkeit

Abb. 14-2: Zeitdiagramm "Antriebsgeführter Betriebsartenwechsel"



Der Wert dieses Parameters entspricht im Defaultzustand dem maximalen Absolutwert der Parameter A-0-0032 / A-0-0033, Geschwindigkeitsgrenzwert, positiv/negativ.

Individuelle Werte können konfiguriert werden, wenn im Parameter A-0-0007, Achskonfiguration, Bit 4 gesetzt wird.

Verwendung Dieser Wert wird beim Phasenschalten von P2 nach BB zum Antrieb hin synchronisiert.

14.2 Gemeinsame C-Parameter aller Kinematiken einer MLC

14.2.1 Allgemeines

Zur Verwaltung der Daten **aller** Kinematiken einer Steuerung werden nachstehende C-Parameter angelegt:

Details siehe "IndraMotion MLC Parameterbeschreibung".

14.2.2 C-0-0123, Maximale Anzahl an Kinematiken

Funktion Dieser Parameter gibt die maximale Anzahl an Kinematiken an.

14.2.3 C-0-2480, Konfigurierte Kinematikenanzahl

Funktion Dieser Parameter zeigt die aktuelle Anzahl konfigurierter Kinematiken an.

14.2.4 C-0-2482, Liste der nicht verknüpften Achsen

- Funktion** Eine Achse kann nur einer Kinematik zugeordnet werden.
Dieser Listenparameter zeigt die Achsen an, die noch keiner Kinematik zugeordnet sind.
Ist keine Kinematik konfiguriert, so entspricht diese Liste dem Parameter "C-0-0483, Liste der konfigurierten Achsen".
Die Zuordnung der Achsen zu den Kinematiken erfolgt über den Parameter "K-0-0008, Achskonfigurationsliste".

14.2.5 C-0-2483, Liste der konfigurierten Kinematiken

- Funktion** Dieser Parameter zeigt die konfigurierten Kinematiken des Systems.
Verwendung Beispiel:
3 Kinematiken (Kinematik 1, 2 und 3) sind im System konfiguriert.
C-0-2483 zeigt:

Element	Wert	Kommentar
1	1	Kinematik 1 ist konfiguriert
2	2	Kinematik 2 ist konfiguriert
3	3	Kinematik 3 ist konfiguriert

14.2.6 C-0-2484, Kinematikkonfigurationsliste

- Funktion** Dieser Parameter definiert das Kinematiksystem. Er konfiguriert die Kinematiken. Es kann nur auf K-Parameter von Kinematiken zugegriffen werden, die hier konfiguriert sind.
Die Zuordnung der Achsen zur Kinematik erfolgt über den Parameter "K-0-0008, Achskonfigurationsliste" (Beispiel K01:K-0-0008).

- Aufbau** Gültige Werte einer Kinematikkonfiguration:

Wert	Kommentar
0	keine Kinematik konfiguriert
1	Kinematik konfiguriert

- Verwendung** Die Liste darf keine Lücken mit nicht konfigurierten Kinematiken besitzen. Dies wird beim Umschalten nach Phase 4 geprüft.

Ist dies nicht der Fall, wird der Parameter "C-0-2484, Kinematikkonfigurationsliste" in den Parameter "C-0-0111, Liste der ungültigen C-Parameter" eingetragen und eine fehlerhafte Konfiguration angezeigt (F0160024, Fehler im Cross-check C-Parameter).

Ein Beschreiben des Parameters "C-0-2484, Kinematikkonfigurationsliste" aktualisiert den Parameter "C-0-2480, Konfigurierte Kinematikenanzahl" und den Parameter "C-0-2483, Liste der konfigurierten Kinematiken".

Beispiel A:

Es soll eine Kinematik konfiguriert werden.

C-0-2484 zeigt:

RCL-Parameter

Element	Wert	Kommentar
1	1	Kinematik 1 ist konfiguriert
2	0	Keine Kinematik 2 verwendet/benötigt
3	0	Keine Kinematik 3 verwendet/benötigt
4	0	Keine Kinematik 4 verwendet/benötigt

Beispiel B:

Es sollen drei Kinematiken konfiguriert werden (ab Version 04V12).
C-0-2484 zeigt:

Element	Wert	Kommentar
1	1	Kinematik 1 ist konfiguriert
2	1	Kinematik 2 ist konfiguriert
3	1	Kinematik 3 ist konfiguriert
4	0	Keine Kinematik 4 verwendet/benötigt

14.3 Parameter einer Kinematik in der MLC

14.3.1 Allgemeines

Zur Verwaltung der Daten **einer** Kinematik in der Steuerung werden nachstehende K-Parameter angelegt. Jeder Parameter existiert genau einmal pro Kinematik.

Details siehe "IndraMotion MLC Parameterbeschreibung".

14.3.2 K-0-0001, Logische Kinematiknummer

Funktion Aus diesem Parameter kann die logische Kinematiknummer gelesen werden.

14.3.3 K-0-0002, Kinematikbezeichnung

Funktion In diesem Parameter steht der Name der Kinematik (z.B. Robi_1).

Mit diesem Namen wird automatisch im SPS-Programm eine globale Variable vom Typ AXIS_GROUP_REF angelegt.



Folgende Einschränkungen gelten:

- maximal 12 Zeichen,
- A...Z, a...z, 0..9, '_',
- erstes Zeichen muss ein Buchstabe sein,
- "Robi_1" identisch "ROBI_1".

14.3.4 K-0-0003, Maximale Bahngeschwindigkeit

Funktion Maximal programmierbare Bewegungsgeschwindigkeit in Raumkoordinaten (RK) am Tool Center Point (TCP).

Die Eingabe erfolgt in Abhängigkeit vom Parameter K-0-0054 in mm/s oder inch/s.

Die Bestimmung einer sinnvollen maximalen Bahngeschwindigkeit ist abhängig von der Auflösung des Messsystems, dem minimalen programmierten Punktabstand, der Robotermechanik und der Abtastzeit des Positionsregelkreises (Clock-Startzeit).

14.3.5 K-0-0004, reserviert

Funktion Dieser Parameter ist reserviert für interne Anwendungen.

14.3.6 K-0-0005, Software-Endschalter, positiv, WC

Funktion Mit dem Parameter werden die Werte der positiven Software-Endschalter in Raumkoordinaten festgelegt.
Die Eingabe erfolgt in Abhängigkeit vom Parameter K-0-0054 in mm oder inch bzw. in Grad.
Wirkt nur im Jog-Betrieb.
Kinematikbedingt kann im Automatik- oder Handbetrieb beim Verfahren in Raumkoordinaten eine interne Fahrbereichsgrenze erreicht werden.
Die Bewegung wird sofort gestoppt und unter der Betriebsart Diagnose-Fehler wird 'Fahrbereichsgrenze erreicht' angezeigt.
Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert ist erst beim nächsten Phasenschalten der Steuerung von P2 nach BB wirksam.

14.3.7 K-0-0006, Software-Endschalter, negativ, WC

Funktion Mit dem Parameter werden die Werte der negativen Software-Endschalter in Raumkoordinaten festgelegt.
Die Eingabe erfolgt in Abhängigkeit vom Parameter K-0-0054 in mm oder inch bzw. in Grad.
Wirkt nur im Jog-Betrieb.
Kinematikbedingt kann im Automatik- oder Handbetrieb beim Verfahren in Raumkoordinaten eine interne Fahrbereichsgrenze erreicht werden.
Die Bewegung wird sofort gestoppt und unter der Betriebsart Diagnose-Fehler wird "Fahrbereichsgrenze erreicht" angezeigt.
Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert ist erst beim nächsten Phasenschalten der Steuerung von P2 nach BB wirksam.

14.3.8 K-0-0007, Achsanzahl der Kinematik

Funktion Dieser Parameter enthält die aktuelle Achsanzahl der Kinematik.
Der Wert ergibt sich aus "K-0-0008, Achskonfigurationsliste".
Er enthält die Summe aller dort konfigurierten Achsen und Bänder.
Der Parameter "K-0-0070, Bandanzahl der Kinematik" enthält nur die Summe der Bänder.

14.3.9 K-0-0008, Achskonfigurationsliste

Funktion Dieser Parameter konfiguriert die Achsen für die Kinematik.

Aufbau Der Parameter ist ein Listenparameter mit folgender Dreifach-Struktur:

- Element 1: Achskonfiguration der Kinematikachse 1
- Element 2: Steuerungsnummer
- Element 3: zugeordnete Achse zur Kinematikachse 1
- Element 4: Achskonfiguration der Kinematikachse 2
- Element 5: Steuerungsnummer
- Element 6: zugeordnete Achse zur Kinematikachse 2
- ...

Die Elementanzahl muss geradzahlig sein.

RCL-Parameter

Wert	Kommentar
0	keine Achse konfiguriert
8	Positionierachse konfiguriert
16	Orientierungsachse konfiguriert
32	Bandachse konfiguriert

Abb. 14-3: Gültige Werte einer Achskonfiguration

Wert	Kommentar
0	eigene Steuerung
1-64	Steuerungsnummer im C2C-Verbund

Abb. 14-4: Gültige Werte für die Steuerungsnummer

Verwendung**Aufbau der Dreifachliste**

Die Liste darf keine Lücken enthalten.

Einer Kinematik muss mindestens eine Positionierungsachse zugeordnet sein.

Orientierungsachsen müssen hinter den Positionierungsachsen liegen.

Bandachsen müssen hinter Positionierungsachsen und Orientierungsachsen liegen.

Abhängig vom eingestellten Parameter "K-0-0031, Transformationsschema" wird eine unterschiedliche minimale Anzahl von Positionierungsachsen und Orientierungsachsen benötigt (siehe gewünschte Transformationsbeschreibung).

Es dürfen maximal vier Bandachsen definiert sein.

Die Dreifachliste darf in Summe maximal 16 Achsen konfigurieren.

Hilfen bei der Zuordnung

Jede Achse darf höchstens einer Kinematik als Positionierungsachse oder Orientierungsachse zugeordnet werden.

Im Parameter "C-0-2482, Liste der nicht verknüpften Achsen" sind die Achsen aufgelistet, die noch keiner Kinematik zugeordnet wurden.

Als Bandachse konfigurierte Kinematikachsen beeinflussen den Parameter "C-0-2482, Liste der nicht verknüpften Achsen" nicht.

Bandachsen können aus der "C-0-0483, Liste der konfigurierten Achsen" ausgewählt und mehreren Kinematiken zugewiesen werden.

Einfluss auf andere Parameter

Die Anzahl der Dreifach-Einträge in der Liste bestimmt den Parameter "K-0-0007, Achsanzahl der Kinematik" .

Die Anzahl der in der Liste konfigurierten Bandachsen bestimmt den Parameter "K-0-0070, Bandanzahl der Kinematik" .



Beim Umschalten nach Phase 4 wird überprüft, ob für eine konfigurierte Kinematik mindestens eine Achse konfiguriert ist und ob die konfigurierten Achsen vorhanden sind.

Ist bei den Kinematikachsen die "A-0-0059, Wichtungsart für Lagdaten" auf Moduloformat eingestellt, wird auch geprüft, ob der "A-0-0045, Modulowert" größer Null und ganzzahlig ist.

Ist dies nicht der Fall, wird der Parameter "K-0-0008, Achskonfigurationsliste" in den Parameter "K-0-0011, Liste der ungültigen K-Parameter" eingetragen und eine fehlerhafte Konfiguration angezeigt (F0160040, Fehler im Cross-check K-Parameter).

Beispiel:

Die Kinematik benutzt das Transformationsschema 10 (SCARA) mit vier Achsen.

Achse 1, 2 und 3 sind Positionierachsen.

Achse 4 ist eine Orientierungsachse.

K-0-0008 zeigt:

Element	Wert	Kommentar
1	8	Kinematikachse 1: Positionierachse
2	0	Steuerungsnummer 0: eigene Steuerung
3	1	Achse mit logischer Achsnummer 1 zugeordnet
4	8	Kinematikachse 2: Positionierachse
5	0	Steuerungsnummer 0: eigene Steuerung
6	2	Achse mit logischer Achsnummer 2 zugeordnet
7	8	Kinematikachse 3: Positionierachse
8	0	Steuerungsnummer 0: eigene Steuerung
9	3	Achse mit logischer Achsnummer 3 zugeordnet
10	16	Kinematikachse 4: Orientierungsachse
11	0	Steuerungsnummer 0: eigene Steuerung
12	4	Achse mit logischer Achsnummer 4 zugeordnet

Für Positionier- und Orientierungsachsen stehen folgende Achstypen zur Verfügung (siehe C-0-0484):

- reale Achse,
- virtuelle Achse,
- Reglerachse.

Für Bandachsen stehen zusätzlich noch zur Verfügung:

- Geberachse und
- Verbundachse (Querverbund SERCOS 2).

14.3.10 K-0-0009, Oszi: Liste aller oszilloskopierbaren K-Parameter

Funktion Dieser Parameter enthält die Liste aller oszilloskopierbaren K-Parameter.

RCL-Parameter

14.3.11 K-0-0010, Liste aller K-Parameter

Funktion In dieser Liste stehen alle K-Parameter. Der Eintrag erfolgt im ID-Nummern-Format.

Die Einträge sind nach Bedeutung und Zusammengehörigkeit sortiert, also nicht nach Ident-Nummern oder alphabetischer Reihenfolge.

In der Reihenfolge dieser Einträge werden K-Parameter abgespeichert.

Nachstehend das ID-Nummern-Format:

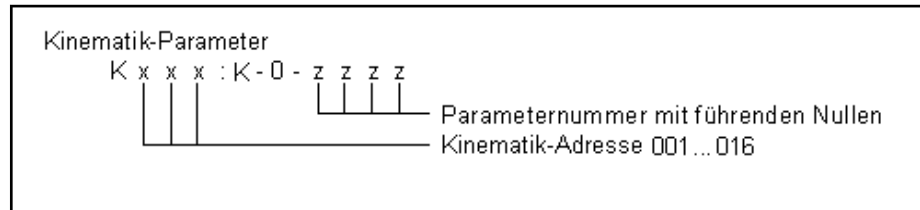


Abb. 14-5: Aufbau eines K-Parameters

Beispiel:

K001: K-0-0004 - Kinematik 1; Parameter K-0-0004.

14.3.12 K-0-0011, Liste aller ungültigen K-Parameter

Funktion Diese Liste zeigt alle ungültigen K-Parameter im Ergebnis

- eines Quersummentestes, z.B. nach einem Firmware-Update, oder
- eines Plausibilitätstestes, z.B. im Ergebnis einer illegalen Betriebsart.

Der Eintrag erfolgt im ID-Nummern-Format.

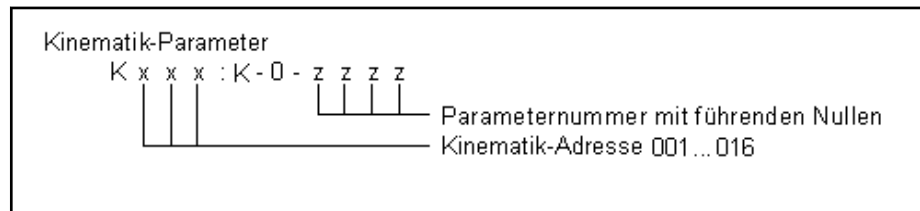


Abb. 14-6: Aufbau eines K-Parameters

Verwendung K001: K-0-0008 - Kinematik 1; Parameter K-0-0008.

14.3.13 K-0-0012, Prüfsumme aller K-Parameter

Funktion Beim Auslesen dieses Parameters wird die Prüfsumme über alle Parameterwerte gebildet, deren Identnummern im Parameter "K-0-0013, IDN-Liste der K-Prüfsummenparameter" aufgelistet sind.

14.3.14 K-0-0013, IDN-Liste der K-Prüfsummenparameter

Funktion In diesen Parameter werden die Identnummern der Parameter eingetragen, deren Checksumme gebildet werden soll.

Die Checksummenbildung wird mit dem Auslesen des Parameters "K-0-0012, Prüfsumme über alle K-Parameter" ausgelöst.

Im Defaultzustand sind in K-0-0013 noch keine Identnummern eingetragen.

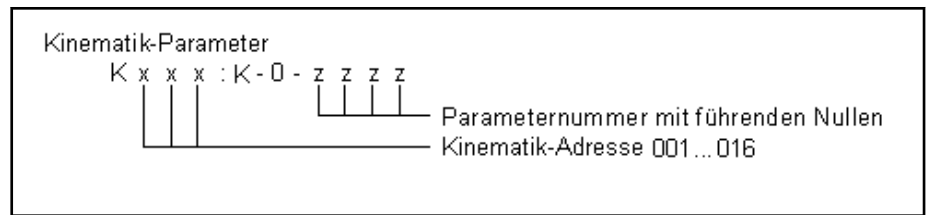


Abb. 14-7: Aufbau eines K-Parameters

Verwendung K001: K-0-0008 - Kinematik 1; Parameter K-0-0008.

14.3.15 K-0-0014, Bahnbeschleunigung, Änderungszeit

Funktion Dieser Parameter wirkt nur mit den RCL-Befehlen MOVE LINEAR und MOVE CIRCULAR in der Satzübergangsart PROGR_SLOPE.

In der Satzübergangsart BLOCK_SLOPE kann eine Ruckbegrenzung über das RCL-Schlüsselwort "J" programmiert werden.

Die Änderungszeit wirkt nur für die Beschleunigung, nicht aber für die Bremsverzögerung.

Die Bremsverzögerungsänderungszeit wird mit dem Parameter K-0-0015 eingestellt.



Die hier angegebene Änderungszeit ist die maximale Zeit, bis die im RCL-Programm programmierte Beschleunigung/Bremsverzögerung erreicht wird.

Wir unterscheiden 3 Bewegungsformen:

- Beschleunigung im Rechteck-Betrieb:
K-0-0014 und K-0-0015 sind 0.

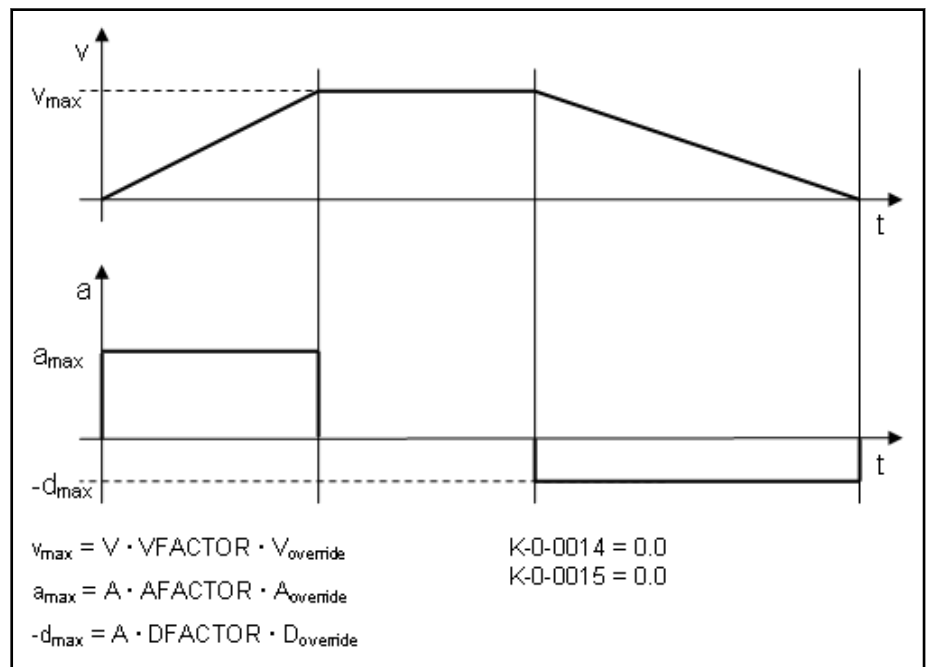


Abb. 14-8: Beschleunigung/Bremsverzögerung im Rechteck-Betrieb

Der daraus resultierende Ruck j (jerk) ist mathematisch unendlich.

- Beschleunigung im Trapez-Betrieb:
K-0-0014 und/oder K-0-0015 sind größer oder gleich der SERCOS-Zykluszeit (siehe auch C-0-0400), und die programmierte Beschleunigung

RCL-Parameter

muss eine gewisse Zeit konstant anstehen, um die programmierte Geschwindigkeit zu erreichen.

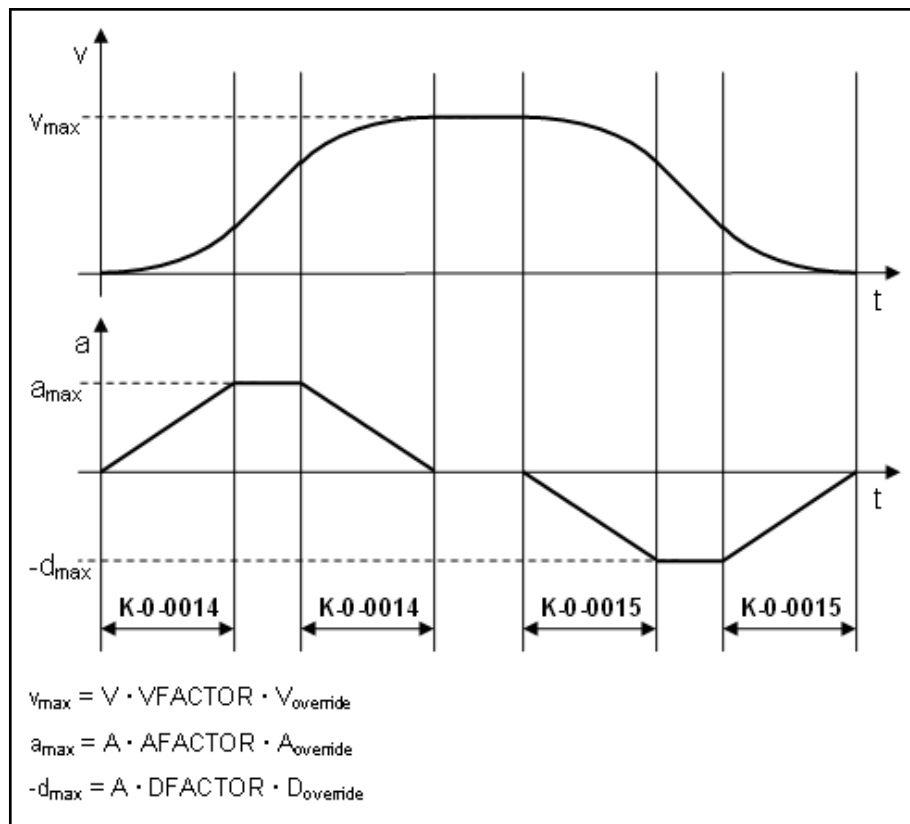


Abb. 14-9: Beschleunigung/Bremsverzögerung im Trapez-Betrieb

$$j = \frac{a_{\max}}{K-0-0014}$$

Der resultierende Ruck j ergibt sich zu

- Beschleunigung im Dreiecks-Betrieb:

Die in K-0-0014 und K-0-0015 angegebenen Zeiten können nicht eingehalten werden, um die programmierte Beschleunigung zu erreichen.

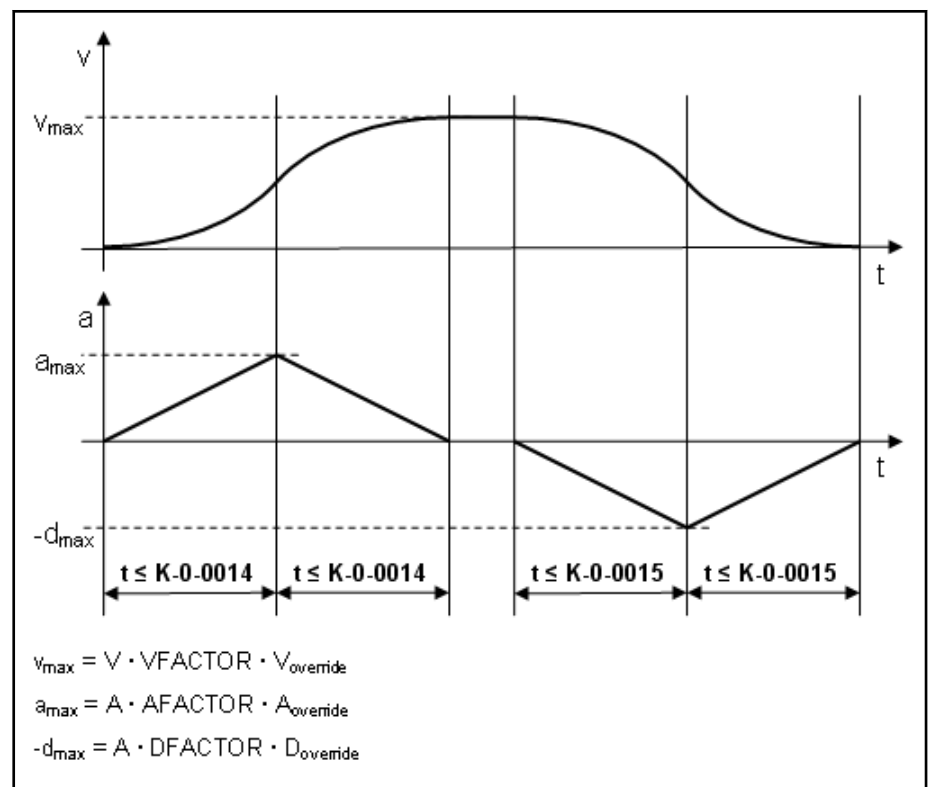


Abb. 14-10: Beschleunigung/Bremsverzögerung im Dreiecks-Betrieb

$$j = \frac{a_{\max}}{t}$$

Der resultierende Ruck j ergibt sich zu

Verwendung

Je größer die Änderungszeiten sind, desto weicher wird das Bewegungsprofil. Wenn die Beschleunigung/Bremsverzögerung zu groß programmiert wird geht die Bahnplanung in den oben beschriebenen Dreiecks-Betrieb über.



Je größer die programmierte Beschleunigung und je kleiner die Beschleunigungsänderungszeit desto größer der Ruck.

Durch kleine Geschwindigkeiten kann die Beschleunigungsänderungszeit t im Dreiecksbetrieb sehr klein werden was wiederum einen großen Ruck zur Folge hat.

14.3.16 K-0-0015, Bahnbremsverzögerung, Änderungszeit

Funktion

Dieser Parameter wirkt nur mit den RCL-Befehlen MOVE LINEAR und MOVE CIRCULAR in der Satzübergangsart PROGR_SLOPE.

In der Satzübergangsart BLOCK_SLOPE kann eine Ruckbegrenzung über das RCL-Schlüsselwort "J" programmiert werden.

Die Änderungszeit wirkt nur für die Bremsverzögerung, nicht aber für die Beschleunigung.

Die Beschleunigungsänderungszeit wird mit dem Parameter K-0-0014 eingestellt.



Die hier angegebene Änderungszeit ist die maximale Zeit, bis die im RCL-Programm programmierte Beschleunigung/Bremsverzögerung erreicht wird.

RCL-Parameter

Wir unterscheiden 3 Bewegungsformen:

- Beschleunigung im Rechteck-Betrieb:
K-0-0014 und K-0-0015 sind 0.

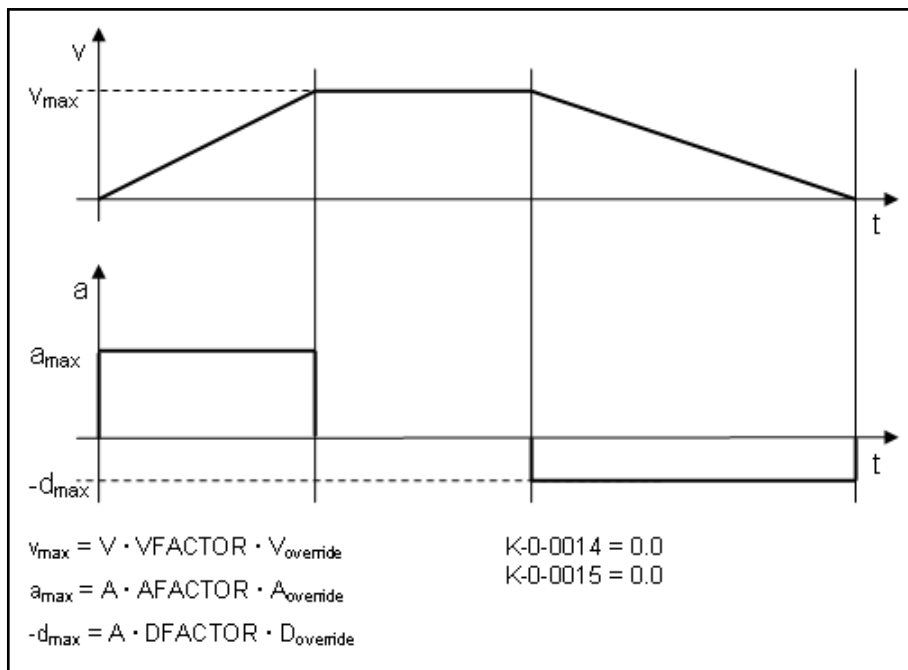


Abb. 14-11: Beschleunigung/Bremsverzögerung im Rechteck-Betrieb

Der daraus resultierende Ruck j (jerk) ist mathematisch unendlich.

- Beschleunigung im Trapez-Betrieb:
K-0-0014 und/oder K-0-0015 sind größer oder gleich der SERCOS-Zykluszeit (siehe auch C-0-0400), und die programmierte Beschleunigung muss eine gewisse Zeit konstant anstehen, um die programmierte Geschwindigkeit zu erreichen.

RCL-Parameter

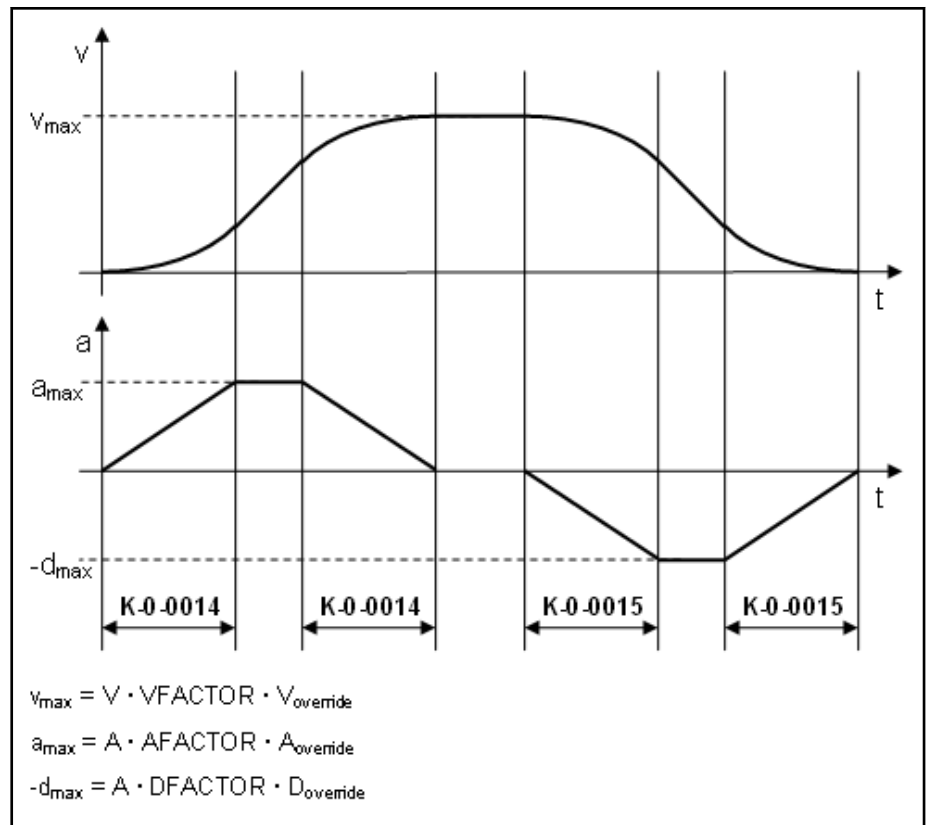


Abb. 14-12: Beschleunigung/Bremsverzögerung im Trapez-Betrieb

$$j = \frac{a_{\max}}{K-0-0015}$$

Der resultierende Ruck j ergibt sich zu

- Beschleunigung im Dreiecks-Betrieb:
Die in K-0-0014 und K-0-0015 angegebenen Zeiten können nicht eingehalten werden, um die programmierte Beschleunigung zu erreichen.

RCL-Parameter

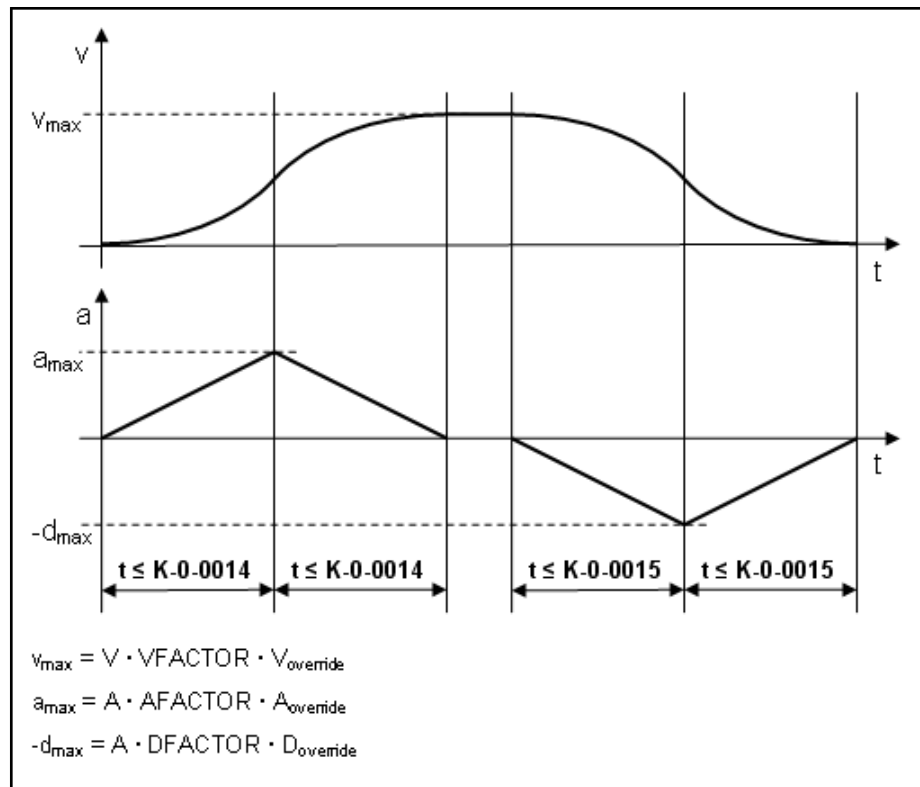


Abb. 14-13: Beschleunigung/Bremsverzögerung im Dreiecks-Betrieb

$$j = \frac{a_{\max}}{t}$$

Der resultierende Ruck j ergibt sich zu

Verwendung Je größer die Änderungszeiten sind, desto weicher wird das Bewegungsprofil. Wenn die Beschleunigung/Bremsverzögerung zu groß programmiert wird geht die Bahnplanung in den oben beschriebenen Dreiecks-Betrieb über.



Je größer die programmierte Beschleunigung und je kleiner die Beschleunigungsänderungszeit desto größer der Ruck.

Durch kleine Geschwindigkeiten kann die Beschleunigungsänderungszeit t im Dreiecksbetrieb sehr klein werden, was wiederum einen großen Ruck zur Folge hat.

14.3.17 K-0-0016, Projektidentifikationsnummer

Funktion IndraWorks generiert in diesem Parameter eine eindeutige Zeichenkette zur weltweiten Identifikation des Projektes.

Die Zeichenkette ist eineindeutig für jedes Projekt.



Dieser Parameter ist nur für interne Verwendung vorgesehen.

14.3.18 K-0-0020, Kinematikdiagnosemeldung

Funktion Dieser Parameter enthält die aktuelle Diagnosemeldung der Kinematik (Meldung, Warnung oder Fehler).

Wenn Bit 8 oder Bit 9 des Parameters "K-0-0021, Kinematikstatus" auf 1 gesetzt ist, enthält dieser Parameter den Warnungs- (Bit 8) bzw. Fehlertext (Bit 9).

RCL-Parameter

Aus der Sicht der SPS sind diese Informationen für die Kinematik "" verfügbar als

- KinData[*].dwKinStatus_i.8= TRUE, Diagnosemeldung in K-0-0020 ist eine Warnungstext bzw.
- KinData[*].dwKinStatus_i.9= TRUE, Diagnosemeldung in K-0-0020 ist ein Fehlertext.

14.3.19 K-0-0021, Kinematikstatus

Funktion Dieser Parameter beschreibt den aktuellen Status der Kinematik.

Dabei handelt es sich um interne Systeminformationen, die zum Beispiel über SPS-Funktionsbausteine ausgelöst werden. Es handelt sich dabei nicht um Informationen über den Zustand der Kinematik-Zustandsmaschine (siehe auch K-0-0022).

Aufbau

Bit	Inhalt	Name des Bits
2...0	001 - Automatik 010 - Manual	Kin_OpMode_Auto Kin_OpMode_Manual
4..3	Jog-Koordinaten-System 00 - JC 01 - WC original 10 - WC workpiece 11 - WC tool	Kin_JogCoord1 (Bit 3) Kin_JogCoord2 (Bit 4)
5	Alle Achsen der Kinematik stehen. (Bandachsen werden nicht berücksichtigt).	Kin_StandStill
6	Sollposition = Istposition Die programmierte Endposition ist erreicht.	Kin_InPosition
7	Bandsynchron	Kin_BeltSynchronous
8	Warnung	Kin_Warning
9	Fehler	Kin_Error
10	Homing erforderlich, nicht alle Achsen sind referenziert.	Kin_NotHomed
12	Transformierte Weltkoordinaten sind ungültig	Kin_TrafoValuesInvalid
13	Betriebsart MANUAL_JOG ist aktiv KinData[].stJogCom_q.enJogCmdVel KIN_JOG_STOP	Kin_JogEnabled
14	Alle Achsen betriebsbereit (AH oder AF)	Kin_DrivesOn
15	Vorschubfreigabe Freigabe für Verfahrbewegungen erteilt.	Kin_FeedEnabled
16	Satzverarbeitung freigegeben Verarbeitung des nächsten Satzes freigegeben	Kin_BlockEnabled
17	Fehlerreaktion aktiviert	Kin_ErrorReaction

RCL-Parameter

Bit	Inhalt	Name des Bits
19..18	JOG-MODE 00 - small steps 01 - big steps 10 - slow 11 - fast	Kin_JogMode1 (Bit 18) Kin_JogMode2 (Bit 19)
21	Programm angewählt	Kin_ProgSelected
23	Reset ist aktiv	Kin_ResetActive
24	Programm beendet	Kin_ProgEnded
25	Programm gestartet	Kin_ProgStarted
26	Programm: PAUSE Befehl aktiv	Kin_ProgPaused

Abb. 14-14: K-0-0021 Kinematikstatus



Die Bits 0..2 für die Betriebsarten werden als numerische Zahl (0..7) ausgewertet

Verwendung**Bit 5: Alle Achsen der Kinematik stehen**

Dieses Bit signalisiert, dass die durch die Kinematik positionierten Achsen im Stillstand sind. Dies bedeutet, dass die Geschwindigkeit Null ist.

Dieses Bit unterscheidet sich von Kis_Standstill (siehe Parameter K-0-0022) welches sich auf die Kinematik-Zustandsmaschine bezieht.

Bit 8: Warnung

Wird ein inkorrekt erkannt, der den ordnungsgemäßen Betrieb aber noch zulässt, dann wird eine Warnung generiert.

Beispiel:

Die programmierte Bahngeschwindigkeit kann nicht erreicht werden aufgrund der maximalen Geschwindigkeit einer oder mehrerer Achsen.



Einige Warnungen führen bei Nichtbeachtung zu einem Fehler.

Bit 9: Fehler

Wird ein inkorrekt erkannt, der den ordnungsgemäßen Betrieb nicht mehr zulässt, dann wird ein Fehler generiert.

Beispiel:

Die programmierte Position liegt außerhalb der Verfahrgrenze einer oder mehrerer Achsen.

Bit 12: Transformierte Weltkoordinaten sind ungültig

Die in der KinData angezeigten aktuellen Weltkoordinaten der Kinematik sind nicht gültig. Dies ist zum Beispiel der Fall, wenn nicht alle Achsen der Kinematik zugeordnet sind oder die Transformation einen Fehler hat.

Bit 17: Fehlerreaktion aktiv

Bei Fehlern mit Fehlerreaktion (F0-Fehler haben keine Fehlerreaktion) werden alle an der Kinematik beteiligten Achsen schnellstmöglich unter Einhaltung der programmierten Bahn gestoppt.

Das Bit 17 wird gelöscht, wenn alle Achsen stehen.

Bit 26: Programm: PAUSE Befehl aktiv

Im Roboterprogramm ist ein PAUSE Befehl aktiv. (siehe PAUSE-Befehl)

14.3.20 K-0-0022, Erweiterter Kinematikstatus

Funktion Dieser Parameter beschreibt den aktuellen Zustand der Kinematik-Zustandsmaschine mit Hilfe mehrerer Statusbits. Jeder Zustand ist dabei einer Statusbitkombination im Parameter K-0-0022 zugeordnet.

Die Abkürzung KIS der Bitbezeichnung steht dabei für "Kinematics State Machine", im Gegensatz zu den KIN-Bits. Die KIN-Bits beschreiben einzelne Statusmeldungen der Kinematik.

Aufbau Die Kinematik-Zustandsmaschine wird in der folgenden Abbildung verdeutlicht:

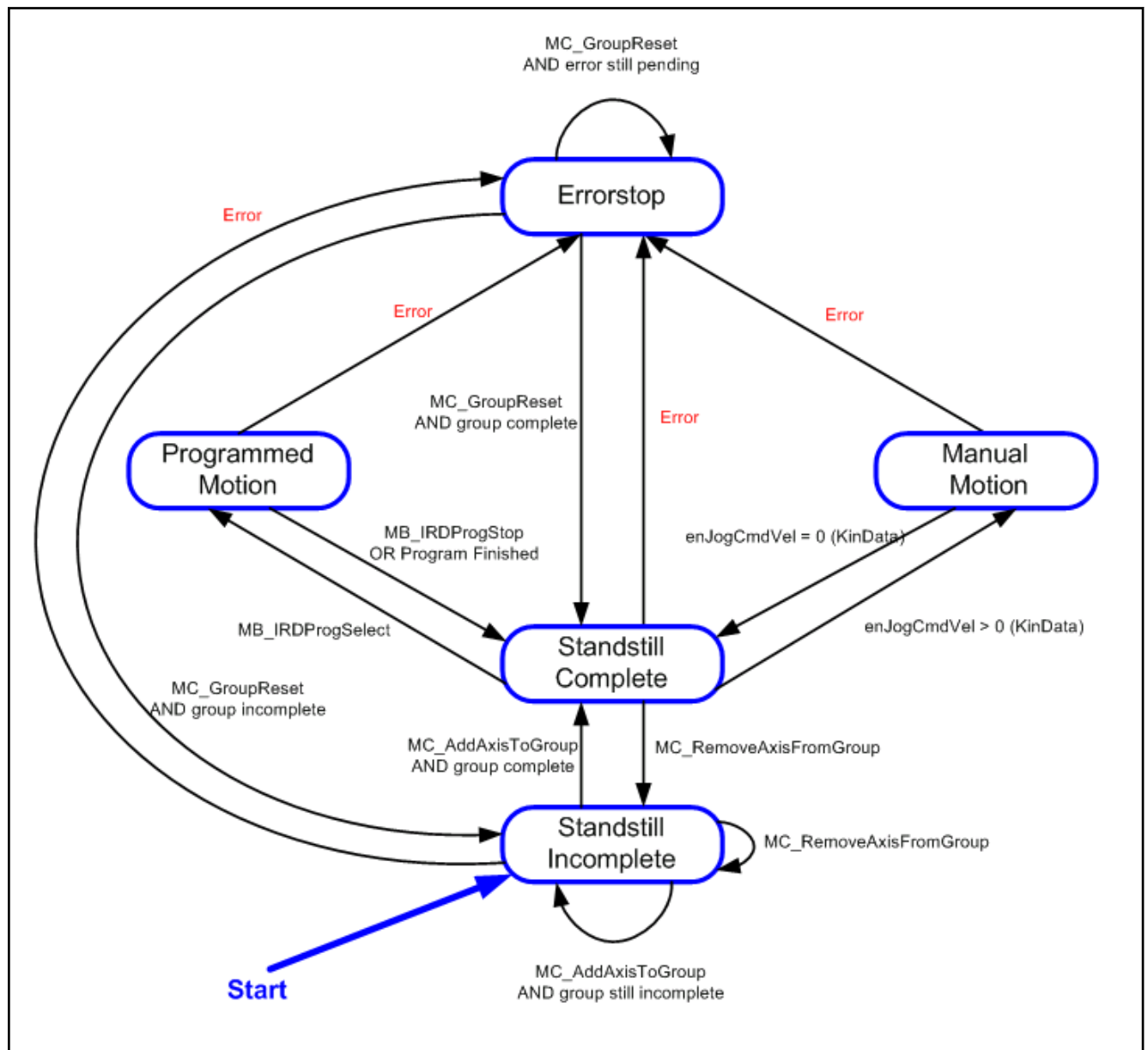


Abb. 14-15: Kinematik-Zustandsmaschine

"Programmed Motion" hat folgende Unterzustände:

RCL-Parameter

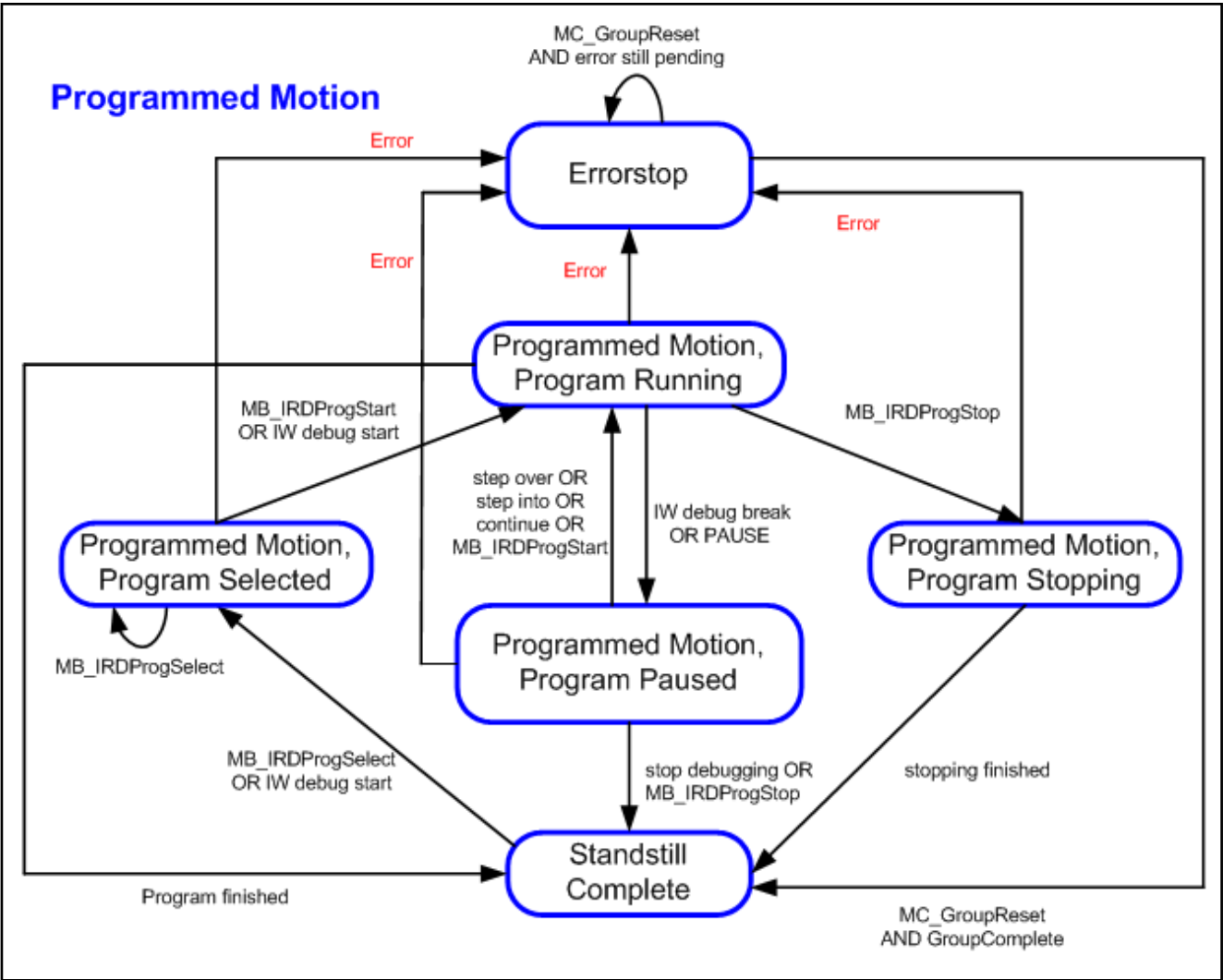


Abb. 14-16: "Programmed Motion" Zustandsmaschine

Zustand Errorstop

Der Zustand Errorstop wird erreicht, sobald ein Fehler während des Betriebs der Kinematik aufgetreten ist. Die Achsen bleiben im Zustand "Coordinated Motion" und sind weiterhin der Kinematik zugeordnet. Mit Ausnahme von Achsen, die einen Fehler verursacht haben bzw. sich im Fehlerzustand befinden.

Der Zustand Errorstop kann sowohl über den Baustein MB_GroupReset als auch über den Kommandoparameter "C-0-1030, Kommando: Alle Steuerungsfehler löschen" zurückgesetzt werden. Hiernach wird bei vollständiger Kinematik nach Zustand "Standstill Complete" bzw. bei unvollständiger Kinematik nach Zustand "Standstill Incomplete" gewechselt.

Bitfeld-Definition

Folgende Bits stehen im Parameter K-0-0022 zur Verfügung:

Bit	Inhalt	Name des Bits
0	Kinematik ist in "Errorstop"	Kis_Errorstop
1	Kinematik ist in "Stopping"	Kis_Stopping
3	Nicht alle Achsen sind der Kinematik zugeordnet (incomplete)	Kis_Incomplete
5	Kinematik ist in "Programmed Motion"	Kis_ProgrammedMotion

RCL-Parameter

Bit	Inhalt	Name des Bits
7	Kinematik ist in "StandStill"	Kis_StandStill
8	Programm ist angewählt	Kis_ProgramSelected
9	Kinematik ist aktiv (Running)	Kis_Running
10	Kinematik ist im Jog-Zustand KinData[].stJogCom_q.enJogCmdVel KIN_JOG_STOP	≠ Kis_ManualMotion
11	Kinematik ist in "Pause"	Kis_ProgramPaused

Abb. 14-17: K-0-0022 Erweiterter Kinematikstatus

Zuordnung Bit/Zustand

Die Zuordnung zwischen den skizzierten Zuständen und den dazu gehörigen Bits wird in folgender Tabelle ersichtlich.

1 : Bit gesetzt

0 : Bit nicht gesetzt

x : Bit nicht relevant für Zustand

Zustand	Bits 12...8	Bits 7...4	Bits 3...0
Standstill Incomplete	00000	1x0x	1x00
Standstill Complete	00000	1x0x	0x00
Programmed Motion, Program Selected	00001	0x1x	0x00
Programmed Motion, Program Running	00010	0x1x	0x00
Programmed Motion, Program Stopping	00000	0x1x	0x10
Programmed Motion, Program Paused	01000	0x1x	0x00
Errorstop	00000	0x0x	0x01
Manual Motion	00100	0x0x	0x00
GroupMoving	10000	0x0x	0x00

Abb. 14-18: Zuordnung Kinematikzustand - Statusbits

14.3.21 K-0-0023, Kinematikdiagnosenummer

Funktion Wenn die Bits 8 oder 9 des Parameters "K-0-0021, Kinematikstatus" gesetzt sind, kann in diesem Parameter die aktuelle Diagnosenummer gelesen werden.

14.3.22 K-0-0031, Transformationsschema

Funktion Dieser Parameter beschreibt das Schema, wie die Achsen der Kinematik mechanisch miteinander verkoppelt sind. Es legt fest, wie Gelenkwinkel oder Achspositionen in kartesische Koordinaten zu transformieren sind.

Verwendung Die Bedeutung der einzelnen Werte der Parameter "K-0-0040, Transformation, Längen der Mechanik" und "K-0-0041, Transformation, Faktoren" hängt von diesem eingestellten Transformationsschema ab.

Näheres ist der speziellen Transformationsdokumentation zu entnehmen.

Wert	Kommentar
0	Robotertyp 00, Karteser
9	Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse

RCL-Parameter

Wert	Kommentar
10	Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse
20	Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse
≥1000	Anwenderspezifizierte Transformation

Abb. 14-19: Gültige Werte des Transformationsschema



Dieser Parameter ist bei Inbetriebnahme nach DCC-Service-Angaben einzustellen.

Beim Umschalten nach Phase 4 wird überprüft, ob ausreichend Achsen für das Transformationsschema im Parameter "K-0-0008, Achskonfigurationsliste" konfiguriert sind.

Ist dies nicht der Fall, wird dieser Parameter in den Parameter "K-0-0011, Liste der ungültigen K-Parameter" eingetragen und eine fehlerhafte Konfiguration angezeigt ("F0160040, Fehler im Cross-check K-Parameter").

Beim anwenderdefiniertem Kinematiktyp werden keinerlei Überprüfungen der Parameter im Kinematik-Wizard vorgenommen.

14.3.23 K-0-0032, Referenzieren der Achsen, Sollkonfiguration

Funktion Für die verschiedene Robotertypen ist eine Voreinstellung zum Referenzieren aller Achsen der Kinematik einstellbar.

Referenzieren aller Achsen:

- 0 - nein, Standardeinstellung
- 1 - ja, nur bei bestimmten Transformationen zulässig.



Dieser Parameter ist bei Inbetriebnahme nach BRC-Service-Angaben einzustellen.

14.3.24 K-0-0033, IPOS, Aktualisierung bei Koordinatenwechsel

Funktion Für verschiedene Robotertypen ist eine Voreinstellung zur IPOS-Aufnahme bei Koordinatenwechsel einstellbar.

IPOS-Aufnahme bei Koordinatenwechsel:

- 0 - nein, Standardeinstellung,
- 1 - ja, nur bei bestimmten Transformationen zulässig.



Dieser Parameter ist bei Inbetriebnahme nach DCC-Service-Angaben einzustellen.

14.3.25 K-0-0036, WC-Achsüberwachung

Funktion Für verschiedene Robotertypen ist eine RK-Achsüberwachung (Überwachen der Achsgeschwindigkeit) projektierbar.

Verwendung *RK-Achsüberwachung:*

- 0 = Keine Überwachung
- 1 = Überwachung mit Bewegungsabbruch.
- 2 = Nur im Handbetrieb, Überwachung mit Begrenzung der Achsgeschwindigkeiten. Im Automatikbetrieb keine Überwachung.
- 3 = Diese Einstellung nicht verwenden.

RCL-Parameter

- 4 = Im Hand- und Automatikbetrieb, Überwachung mit Begrenzung der Achsgeschwindigkeiten. Die Begrenzung wirkt über die ganze Bewegung.
- 5 = Stetige Überwachung der Achsgeschwindigkeit im Linearbetrieb.

Bitte weiterhin beachten:

- Die Überwachung erfolgt im Handbetrieb auf die im Parameter A-0-0083 eingestellten schnellen MK-Jog-Geschwindigkeiten und im Automatikbetrieb auf
MIN {|A-0-0032| v |A-0-0033|}
eingestellten maximalen Achsgeschwindigkeiten.
- Tritt die Achsbegrenzung während der Bewegung ein, stimmt die Geschwindigkeit am TCP nicht mit der programmierten Geschwindigkeit überein.
- Tritt die Achsbegrenzung ein, ergeben sich Bahnungenauigkeiten.

14.3.26 K-0-0040, Transformation, Längen der Mechanik

Funktion In diesem Parameter werden Längen konfiguriert, die für die Transformation von Achspositionen in kartesische Koordinaten benötigt werden.
Die Bedeutung der einzelnen Werte hängt vom eingestellten "K-0-0031, Transformationsschema" ab.
Näheres ist der speziellen Transformationsdokumentation zu entnehmen.



Beim Umschalten nach Phase 4 werden die Werte abhängig vom Parameter "K-0-0031, Transformationsschema" überprüft. (Beim Transformationsschema SCARA wird z.B. geprüft, ob die ersten beiden Längen ungleich Null sind.)

Tritt bei der Überprüfung der Werte ein Fehler auf, wird

- dieser Parameter in den Parameter "K-0-0011, Liste der ungültigen K-Parameter" eingetragen und
- eine fehlerhafte Konfiguration angezeigt ("F0160040, Fehler im Cross-check K-Parameter").

14.3.27 K-0-0041, Transformation, Faktoren

Funktion In diesem Parameter werden Faktoren konfiguriert, die für die Transformation von Achspositionen in kartesische Koordinaten benötigt werden.
Die Bedeutung der einzelnen Werte hängt vom eingestellten "K-0-0031, Transformationsschema" ab. Näheres ist der speziellen Transformationsdokumentation zu entnehmen.



Beim Umschalten nach Phase 4 werden die Werte abhängig vom Parameter "K-0-0031, Transformationsschema" überprüft.

Tritt bei der Überprüfung der Werte ein Fehler auf, wird

- der Parameter "K-0-0041, Transformationsfaktoren" in den Parameter "K-0-0011, Liste der ungültigen K-Parameter" eingetragen und
- eine fehlerhafte Konfiguration angezeigt (F0160040, Fehler im Cross-check K-Parameter).

RCL-Parameter

14.3.28 K-0-0042, Verschiebung des WC-Systems

Funktion Dieser Parameter gestattet es, den Koordinatensystem-Ursprung des Transformationsschemas so zu verschieben, dass mit Koordinatenwerten gearbeitet werden kann, die sich auf einen Bezugspunkt in der Bearbeitungszelle beziehen.

Aufbau Dieser Parameter ist ein Listenparameter mit sechs Werten.

Element	Bedeutung
1	Verschiebung in kartesischer X-Richtung [in mm bzw. inch (siehe K-0-0054)]
2	Verschiebung in kartesischer Y-Richtung [in mm bzw. inch (siehe K-0-0054)]
3	Verschiebung in kartesischer Z-Richtung [in mm bzw. inch (siehe K-0-0054)]
4	Verdrehung um die kartesische X-Achse [in Grad] (siehe Hinweis)
5	Verdrehung um die kartesische Y-Achse [in Grad] (siehe Hinweis)
6	Verdrehung um die kartesische Z-Achse [in Grad] (siehe Hinweis)

Verwendung Die ersten drei Werte geben die Lage des Koordinatensystem-Ursprungs aus Sicht des Zellenkoordinatensystems an. Die Elemente 4 bis 6 geben die Rotation des Koordinatensystems an.



Die Verdrehung des Koordinatensystems ist nur für spezielle Transformationsschemas (siehe K-0-0031) möglich. Dies ist in den entsprechenden Beschreibungen dokumentiert (Kein Hinweis über die Verdrehung des Koordinatensystems bedeutet dass diese Option mit diesem Transformationsschema nicht unterstützt wird).

14.3.29 K-0-0043, Werkstückkoordinatenzuordnungsliste

Funktion Dieser Parameter beschreibt, welche Roboterkoordinate sich auf welche Verschiebung/Verdrehung des Werkstück-Koordinatensystems wirkt. Hat die Kinematik weniger als sechs Freiheitsgrade, so können nicht alle Verschiebungen/Verdrehungen des Werkstück-Koordinatensystems kompensiert werden. Mit Hilfe dieses Parameters wird der Steuerung mitgeteilt,

- welche der sechs allgemeinen Werkstück-Koordinaten von der Mechanik überhaupt berücksichtigt werden
- welche Roboterkoordinate zu welcher Werkstück-Koordinate korrespondiert

Aufbau Dieser Parameter ist ein Listenparameter mit sechs Werten.

Element	Bedeutung
1	Roboterkoordinate, die die Verschiebung in kartesischer X-Richtung beinhaltet
2	Roboterkoordinate, die die Verschiebung in kartesischer Y-Richtung beinhaltet
3	Roboterkoordinate, die die Verschiebung in kartesischer Z-Richtung beinhaltet

RCL-Parameter

Element	Bedeutung
4	Roboterkoordinate, die die Verdrehung um die kartesische X-Achse beinhaltet
5	Roboterkoordinate, die die Verdrehung um die kartesische Y-Achse beinhaltet
6	Roboterkoordinate, die die Verdrehung um die kartesische Z-Achse beinhaltet



Ist eine Verdrehung mechanisch nicht möglich, so muss in das entsprechende Element der Wert 0 eingetragen werden.



Die Verdrehung des Koordinatensystems ist nur für spezielle Transformationschemas (siehe K-0-0031) möglich. Dies ist in den entsprechenden Beschreibungen dokumentiert (Kein Hinweis über die Verdrehung des Koordinatensystems bedeutet, dass diese Option mit diesem Transformationsschema nicht unterstützt wird).

Verwendung *Beispiel:*

4-achs Scara

Ein 4-achs Scara kann in x, y, z - Richtung verfahren und eine Drehung um die z-Achse durchführen. Somit ergibt sich für den Parameter K-0-0043 folgende Belegung:

Belegung K-0-0043 bei einem 4-achs Scara

Element	Wert	Bedeutung
1	1	Der erste Achse der Kinematik wird für die Verschiebung in X-Richtung verwendet
2	2	Der zweite Achse der Kinematik K-0-0042 wird für die Verschiebung in Y-Richtung verwendet
3	3	Der dritte Achse der Kinematik K-0-0042 wird für die Verschiebung in Z-Richtung verwendet
4	0	Verdrehung des Werkstücks um die X-Achse ist mechanisch nicht möglich
5	0	Verdrehung des Werkstücks um die Y-Achse ist mechanisch nicht möglich
6	4	Der vierte Achse der Kinematik wird für die Verdrehung des Werkstückkoordinatensystems um Z-Richtung verwendet

Abb. 14-20: Belegung K-0-0043 bei einem 4-achs Scara

14.3.30 K-0-0050, Distanzeinheiten WC

Funktion In diesem Text-Parameter sind die Einheiten für Distanzen und Orientierung für jedes Element eines Listenparameters im Weltkoordinatensystem, mit Kommata getrennt, hinterlegt.

- Nur lesbar – wird automatisch ermittelt.

RCL-Parameter

- Für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag (mit Kommata getrennt)
- Einheit abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achsen K-0-0008.
 - für x-, y-, z-Achsen: mm
 - für alpha, beta, gamma: Grad

Verwendung *Beispiele*

1. Karteser mit X, Y, Z, C
K-0-0050 = „mm,mm,mm,Grad“
2. Scara mit Hub-Drehspindel (X, Y, Z, C)
K-0-0050 = „mm,mm,mm,Grad“
3. Karteser mit X, Y, Z, C, B
K-0-0050 = „mm,mm,mm,Grad,Grad“
4. Karteser mit X, Y, C
K-0-0050 = „mm,mm,Grad“
5. Delta mit C-Achse und 2 Bändern
K-0-0050 = „mm,mm,mm,Grad,“



Die Bänder werden in den Listen für das Weltkoordinatensystem nicht berücksichtigt.

14.3.31 K-0-0051, Geschwindigkeitseinheiten WC

Funktion In diesem Text-Parameter sind die Einheiten für Geschwindigkeiten für jedes Element eines Listenparameters, mit Kommata getrennt, hinterlegt.

- Nur lesbar – wird automatisch ermittelt.
- Für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag (mit Kommata getrennt)
- Einheit abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achsen K-0-0008.
 - für x-, y-, z-Achsen: mm/s
 - für alpha, beta, gamma: Grad/s

Verwendung *Beispiele*

1. Karteser mit X, Y, Z, C
K-0-0051 = „mm/s,mm/s,mm/s,Grad/s“
2. Scara mit Hub-Drehspindel (X, Y, Z, C)
K-0-0051 = „mm/s,mm/s,mm/s,Grad/s“
3. Karteser mit X, Y, Z, C, B
K-0-0051 = „mm/s,mm/s,mm/s,Grad/s,Grad/s“
4. Karteser mit X, Y, C
K-0-0051 = „mm/s,mm/s,Grad/s“
5. Delta mit C-Achse und 2 Bändern
K-0-0051 = „mm/s,mm/s,mm/s,Grad/s,“



Die Bänder werden in den Listen für das Weltkoordinatensystem nicht berücksichtigt.

14.3.32 K-0-0052, Beschleunigungseinheiten WC

Funktion In diesem Text-Parameter sind die Einheiten für Beschleunigung bzw. Bremsverzögerung für jedes Element eines Listenparameters, mit Kommata getrennt, hinterlegt.

- Nur lesbar – wird automatisch ermittelt.
- Für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag (mit Kommata getrennt)
- Einheit abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achsen K-0-0008.
 - für x-, y-, z-Achsen: mm/s^2
 - für alpha, beta, gamma: Grad/s^2

Verwendung *Beispiele*

1. Karteser mit X, Y, Z, C
K-0-0052 = „ $\text{mm/s}^2, \text{mm/s}^2, \text{mm/s}^2, \text{Grad/s}^2$ “
2. Scara mit Hub-Drehspindel (X, Y, Z, C)
K-0-0052 = „ $\text{mm/s}^2, \text{mm/s}^2, \text{mm/s}^2, \text{Grad/s}^2$ “
3. Karteser mit X, Y, Z, C, B
K-0-0052 = „ $\text{mm/s}^2, \text{mm/s}^2, \text{mm/s}^2, \text{Grad/s}^2, \text{Grad/s}^2$ “
4. Karteser mit X, Y, C
K-0-0052 = „ $\text{mm/s}^2, \text{mm/s}^2, \text{Grad/s}^2$ “
5. Delta mit C-Achse und 2 Bändern
K-0-0052 = „ $\text{mm/s}^2, \text{mm/s}^2, \text{mm/s}^2, \text{Grad/s}^2,$ “



Die Bänder werden in den Listen für das Weltkoordinatensystem nicht berücksichtigt.

14.3.33 K-0-0053, Ruckeinheiten WC

Funktion In diesem Text-Parameter sind die Einheiten für Ruck für jedes Element eines Listenparameters, mit Kommata getrennt, hinterlegt.

- Nur lesbar – wird automatisch ermittelt.
- Für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag (mit Kommata getrennt)
- Einheit abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achsen K-0-0008.
 - für x-, y-, z-Achsen: mm/s^3
 - für alpha, beta, gamma: Grad/s^3

Verwendung *Beispiele*

1. Karteser mit X, Y, Z, C
K-0-0053 = „ $\text{mm/s}^3, \text{mm/s}^3, \text{mm/s}^3, \text{Grad/s}^3$ “
2. Scara mit Hub-Drehspindel (X, Y, Z, C)
K-0-0053 = „ $\text{mm/s}^3, \text{mm/s}^3, \text{mm/s}^3, \text{Grad/s}^3$ “
3. Karteser mit X, Y, Z, C, B
K-0-0053 = „ $\text{mm/s}^3, \text{mm/s}^3, \text{mm/s}^3, \text{Grad/s}^3, \text{Grad/s}^3$ “
4. Karteser mit X, Y, C
K-0-0053 = „ $\text{mm/s}^3, \text{mm/s}^3, \text{Grad/s}^3$ “

RCL-Parameter

5. Delta mit C-Achse und 2 Bändern

$$K-0-0053 = „mm/s^3, mm/s^3, mm/s^3, Grad/s^3“$$


Die Bänder werden in den Listen für das Weltkoordinatensystem nicht berücksichtigt.

14.3.34 K-0-0054, Maßeinheit für Kinematikdaten

Funktion Dieser Parameter bestimmt die Wichtungseinheit der Kinematik.

Aufbau Folgende Einstellungen können vorgenommen werden:

Bit	Bezeichnung/ Funktion
0	Einheit 0: mm 1: inch
15..1	reserviert

14.3.35 K-0-0070, Bandanzahl der Kinematik

Funktion Dieser Parameter enthält die aktuelle Bandanzahl der Kinematik.

Der Wert ergibt sich aus "K-0-0008, Achskonfigurationsliste". Er enthält nur die Summe der dort konfigurierten Bänder.

Der Parameter "K-0-0007, Achsanzahl der Kinematik" enthält die Summe aller Achsen und Bänder.

14.3.36 K-0-0071, Kopplungsfaktoren Band 1

Funktion Die Band-Kopplungsfaktoren geben an, wie Bandpositionsänderungen des ersten Bandes auf Koordinatenänderungen in den Programmabschnitten zwischen SYNCHRON und SYNCHRON_END wirken.

Aufbau Die Werte dieses Parameters sind nur von Bedeutung, wenn im Parameter "K-0-0070, Bandanzahl der Kinematik" mindestens der Wert 1 steht.

Dieser Parameter ist ein Listenparameter. Für jede Koordinate ist ein Element in der Liste zuständig.

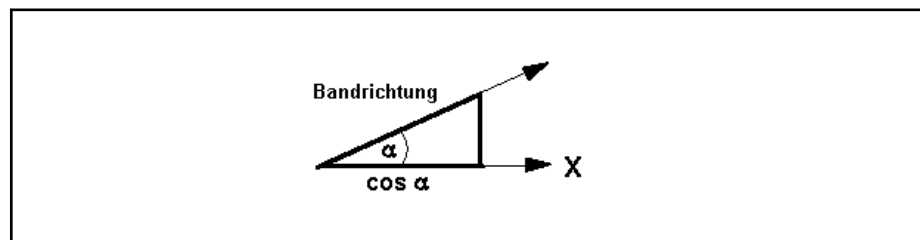


Abb. 14-21: Winkel Bandrichtung/ Achse

Der Wert für eine Koordinate entspricht dem Kosinuswertes ($\cos \alpha$) des Winkels, der zwischen dem Band und der entsprechenden Koordinaten liegt.



Am Windows-Taschenrechner (Ansicht wissenschaftlich) Einstellung DEG (nicht (Neu-) Grad) benutzen.

Verwendung Die Zuordnung der Koordinaten zu Raumrichtungen hängt im allgemeinen vom Parameter "K-0-0031, Transformationsschema" ab. In der Regel wird die erste Koordinate der X-Richtung, die zweite Koordinate der Y-Richtung und die dritte der Z-Richtung zugeordnet sein. In den folgenden Beispielen wird von dieser Definition ausgegangen.

Beispiel 1:

Das Band verläuft genau parallel zur X-Achse. Steigende Bandwerte führen dazu, dass ein auf dem Band befindliches Teil sich in positive X-Richtung bewegt.

K-0-0071 zeigt:

Element	Wert	Kommentar
1	1.000000	Cos (0 Grad)
2	0.000000	Cos (90 Grad)
3	0.000000	Cos (90 Grad)

Beispiel 2:

Das Band verläuft genau parallel zur Y-Achse. Steigende Bandwerte führen dazu, dass ein auf dem Band befindliches Teil sich in negative Y-Richtung bewegt.

K-0-0071 zeigt:

Element	Wert	Kommentar
1	0.000000	Cos (90 Grad)
2	-1.000000	Cos (180 Grad)
3	0.000000	Cos (90 Grad)

Beispiel 3:

Das Band bildet mit der X-Achse einen Winkel von 30 Grad und mit der Y-Achse einen Winkel von 60 Grad. Steigende Bandwerte führen dazu, dass ein auf dem Band befindliches Teil sich in positive X Richtung und in positive Y-Richtung bewegt.

K-0-0071 zeigt:

Element	Wert	Kommentar
1	0.866025	Cos (30 Grad)
2	0.500000	Cos (60 Grad)
3	0.000000	Cos (90 Grad)



Die Wurzel aus der Summe der Quadrate der Cosinuswerte muss möglichst genau 1 ergeben.

Ist der Wert größer als 1 entspricht dies einer Längendehnung in Bandrichtung, bei einem Wert kleiner als 1 einer Längenstauung.

14.3.37 K-0-0072, Kopplungsfaktoren Band 2

Funktion Die Band-Kopplungsfaktoren geben an, wie Bandpositionsänderungen des zweiten Bandes auf Koordinatenänderungen in den Programmabschnitten zwischen SYNCHRON und SYNCHRON_END wirken.

Die Werte dieses Parameters sind nur von Bedeutung, wenn im Parameter "K-0-0070, Bandanzahl der Kinematik" mindestens der Wert 2 steht.

Der Aufbau des Parameters entspricht dem Aufbau des Parameters "K-0-0071, Kopplungsfaktoren Band 1".

RCL-Parameter

14.3.38 K-0-0073, Kopplungsfaktoren Band 3

Funktion Die Band-Kopplungsfaktoren geben an, wie Bandpositionsänderungen des dritten Bandes auf Koordinatenänderungen in den Programmabschnitten zwischen SYNCHRON und SYNCHRON_END wirken.

Die Werte dieses Parameters sind nur von Bedeutung, wenn im Parameter "K-0-0070, Bandanzahl der Kinematik" mindestens der Wert 3 steht.

Der Aufbau des Parameters entspricht dem Aufbau des Parameters "K-0-0071, Kopplungsfaktoren Band 1".

14.3.39 K-0-0074, Kopplungsfaktoren Band 4

Funktion Die Band-Kopplungsfaktoren geben an, wie Bandpositionsänderungen des vierten Bandes auf Koordinatenänderungen in den Programmabschnitten zwischen SYNCHRON und SYNCHRON_END wirken.

Die Werte dieses Parameters sind nur von Bedeutung, wenn im Parameter "K-0-0070, Bandanzahl der Kinematik" mindestens der Wert 4 steht.

Der Aufbau des Parameters entspricht dem Aufbau des Parameters "K-0-0071, Kopplungsfaktoren Band 1".

14.3.40 K-0-0080, Joggen, Distanz kurz im WC-System

Funktion In diesem Parameter ist die kurze Distanz für das inkrementelle Joggen im Weltkoordinatensystem hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (z.B. x, y, z, alpha, beta, gamma) ein Eintrag,
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achsen (siehe K-0-0008).
Die Einheiten sind im K-0-0050 hinterlegt.
Beispiel:
für x, y, z in mm bzw. in inch (siehe K-0-0054),
für alpha, beta, gamma in Grad.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 1.0 mm bzw. inch bzw. Grad.

Verwendung Dieser Parameter wird von der Robot-Control für das inkrementelle Joggen der Achse im Weltkoordinatensystem verwendet.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das inkrementelle Joggen im Weltkoordinatensystem (WC) mit kurzer Distanz aktivieren:

Z.B. Inkrementelles Joggen in WC mit kurzer Distanz der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys:=KIN_CORD_SYS_ORIGIN_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SMALL;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die kurze Distanz für das Joggen der Achse steht in K-0-0080.
- Die maximale Geschwindigkeit steht in K-0-0083 (für inkrementelles Joggen immer schnell).
- Die maximale Beschleunigung steht in K-0-0084 und
- die maximale Bremsverzögerung steht in K-0-0085.
- Der maximal Ruck für die Beschleunigung steht in K-0-0086 und
- der maximal Ruck für die Bremsverzögerung steht in K-0-0087.

Ein vorzeitiges Beenden der Jog-Bewegung ist durch Setzen von enJogMode = KIN_JOG_STOP möglich.

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.3.41 K-0-0081, Joggen, Distanz lang im WC-System

Funktion In diesem Parameter ist die lange Distanz für das inkrementelle Joggen im Weltkoordinatensystem hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (z.B. x, y, z, alpha, beta, gamma) ein Eintrag,
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0050 hinterlegt.
Beispiel:
für x, y, z in mm bzw. in inch (siehe K-0-0054),
für alpha, beta, gamma in Grad.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 10.0 mm bzw. inch bzw. Grad.

Verwendung Dieser Parameter wird von der Robot-Control für das inkrementelle joggen der Achse im Weltkoordinatensystem verwendet.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das inkrementelle Joggen im Weltkoordinatensystem (WC) mit langer Distanz aktivieren:

Z.B. Inkrementelles Joggen in WC mit langer Distanz der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys:=KIN_CORD_SYS_ORIGIN_COORD;  
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_LARGE;  
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die lange Distanz für das Joggen der Achse steht in K-0-0081.
- Die maximale Geschwindigkeit steht in K-0-0083 (für inkrementelles Joggen immer schnell).
- Die maximale Beschleunigung steht in K-0-0084 und

RCL-Parameter

- die maximale Bremsverzögerung steht in K-0-0085.
- Der maximal Ruck für die Beschleunigung steht in K-0-0086 und
- der maximal Ruck für die Bremsverzögerung steht in K-0-0087.

Ein vorzeitiges Beenden der Jog-Bewegung ist durch Setzen von enJogMode = KIN_JOG_STOP möglich.

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

14.3.42 K-0-0082, Joggen, Geschwindigkeit langsam im WC-System

Funktion In diesem Parameter ist die langsame Geschwindigkeit für das kontinuierliche Joggen im Weltkoordinatensystem hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag,
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0051 hinterlegt.
Beispiel:
für x, y, z in mm/s bzw. inch/s (siehe K-0-0054),
für alpha, beta, gamma in Grad/s.
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 10.0 mm/s bzw. inch/s bzw. Grad/s.

Verwendung Dieser Parameter wird von der Robot-Control für das kontinuierliche Joggen der Achse im Weltkoordinatensystem verwendet.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das kontinuierliche Joggen im Weltkoordinatensystem (WC) mit langsamer Geschwindigkeit aktivieren:

Z.B. kontinuierliches Joggen in WC mit langsamer Geschwindigkeit der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys:=KIN_CORD_SYS_ORIGIN_COORD;
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_SLOW;
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die langsame Geschwindigkeit steht in K-0-0082,
- die maximale Beschleunigung steht in K-0-0084 und
- die maximale Bremsverzögerung steht in K-0-0085.
- Der maximal Ruck für die Beschleunigung steht in K-0-0086 und
- der maximal Ruck für die Bremsverzögerung steht in K-0-0087.

Das kontinuierliche Joggen kann auf 2 Arten beendet werden:

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

und / oder

Programm:

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

14.3.43 K-0-0083, Joggen, Geschwindigkeit schnell im WC-System

Funktion In diesem Parameter ist die schnelle Geschwindigkeit für das kontinuierliche Joggen im Weltkoordinatensystem hinterlegt.



Dieser Parameter ist auch in Phase 4 beschreibbar. Der neue Wert wird mit dem nächsten Jog-Befehl wirksam. Eine Änderung des Parameters für den aktiven Jog-Befehl ist nicht möglich.

Aufbau

- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag.
- Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0051 hinterlegt.
Beispiel:
für x, y, z in mm/s bzw. inch/s (siehe K-0-0054),
für alpha, beta, gamma in Grad/s.
- Wertebereich: > 0.0 (Fließkommazahl)
- Default: 25.0 mm/s bzw. inch/s bzw. Grad/s.

Verwendung

Dieser Parameter wird von der Robot-Control für das kontinuierliche Joggen der Achse im Weltkoordinatensystem verwendet.

Mit den Jog-Bits (KinData[1].stJogCmd_q) kann das SPS-Programm das kontinuierliche Joggen im Weltkoordinatensystem (WC) mit schneller Geschwindigkeit aktivieren:

Z.B. kontinuierliches Joggen in WC mit schneller Geschwindigkeit der 3. Achse in positive Richtung:

Programm:

```
KinData[1].stJogCmd_q.enJogCmdCoordSys:=KIN_CORD_SYS_ORIGIN_COORD;  
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STEP_FAST;  
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)  
KinData[1].stJogCmd_q.wJogCmdPositive := 4; (*3. Bit für 3.Achse der Kinematik bzw. Gruppe*)
```

- Die schnelle Geschwindigkeit steht in K-0-0083,
- die maximal Beschleunigung steht in K-0-0084 und
- die maximal Bremsverzögerung steht in K-0-0085.
- Der maximal Ruck für die Beschleunigung steht in K-0-0086 und
- der maximal Ruck für die Bremsverzögerung steht in K-0-0087.

Das kontinuierliche Joggen kann auf 2 Arten beendet werden:

RCL-Parameter

Programm:

```
KinData[1].stJogCmd_q.enJogMode := KIN_JOG_STOP;
```

und / oder

Programm:

```
KinData[1].stJogCmd_q.wJogCmdNegative := 0; (*keine Achse in neg.*)
KinData[1].stJogCmd_q.wJogCmdPositive := 0; (*keine Achse in pos.*)
```

14.3.44 K-0-0084, Joggen, Beschleunigung im WC-System

Funktion In diesem Parameter ist die Beschleunigung für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem hinterlegt.



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag,
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse K-0-0008.
Die Einheiten sind im K-0-0052 hinterlegt.
Beispiel:
für x, y, z in mm/s² bzw. inch/s² (siehe K-0-0054),
für alpha, beta, gamma in Grad/s².
 - Wertebereich: > 0.0 (Fließkommazahl)
 - Default: 370 mm/s² bzw. inch/s² bzw. Grad/s².

Verwendung Dieser Parameter wird von der Robot-Control für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem verwendet.

- Die maximale Bremsverzögerung steht in K-0-0085,
- der maximale Ruck für die Beschleunigung steht in K-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in K-0-0087.

Für das Joggen der Achsen (JC) gelten für die Beschleunigung A-0-0084 und für die Bremsverzögerung A-0-0085.

14.3.45 K-0-0085, Joggen, Bremsverzögerung im WC-System

Funktion In diesem Parameter ist die Bremsverzögerung für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem hinterlegt.



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag.
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0052 hinterlegt.
Beispiel:

für x, y, z in mm/s² bzw. inch/s² (siehe K-0-0054),
für alpha, beta, gamma in Grad/s².

- Wertebereich: ≥ 0.0 (Fließkommazahl)
- Default: 370 mm/s² bzw. inch/s² bzw. Grad/s².

Verwendung Dieser Parameter wird von der Robot-Control für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem verwendet.

- Die maximale Beschleunigung steht in K-0-0084,
- der maximale Ruck für die Beschleunigung steht in K-0-0086 und
- der maximale Ruck für die Bremsverzögerung steht in K-0-0087.

Für das Joggen der Achsen (JC) gelten für die Beschleunigung A-0-0084 und für die Bremsverzögerung A-0-0085.

14.3.46 K-0-0086, Joggen, Beschleunigungsruck im WC-System

Funktion In diesem Parameter ist der max. Ruck bei Beschleunigung für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem hinterlegt (indirekte Definition der Beschleunigungsänderungszeit).



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0053 hinterlegt.
Beispiel:
für x, y, z in mm/s³ bzw. inch/s³ (siehe K-0-0054),
für alpha, beta, gamma in Grad/s³.
 - Wertebereich: ≥ 0.0 (Fließkommazahl)
 - Default: 0.0 (keine Ruckbegrenzung).

Verwendung Dieser Parameter wird von der Robot-Control für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem verwendet.

$$\Delta\alpha = j * t \quad \text{Aus} \quad t = \frac{\Delta\alpha}{j} \quad \text{bzw.} \quad t = \frac{K-0-0084}{K-0-0086} \quad \text{errechnen.}$$

Der maximale Ruck bei der Bremsverzögerung im Raumkoordinatensystem wird in K-0-0087 angegeben.

Die Ruck-Begrenzungswerte für das achsweise Joggen werden in A-0-0086 und A-0-0087 angegeben.

14.3.47 K-0-0087, Joggen, Verzögerungsruck im WC-System

Funktion In diesem Parameter ist der maximale Ruck bei Bremsverzögerung für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem hinterlegt (indirekte Definition der Verzögerungsänderungszeit).

RCL-Parameter



Mit MLC 04VRS ist der Parameter nur in Phase 2 beschreibbar.

- Aufbau**
- Listenparameter:
für jede Koordinatenachse (x, y, z, alpha, beta, gamma) ein Eintrag.
 - Einheit:
abhängig vom Trafo-Typ K-0-0031 und der verwendeten Achse (siehe K-0-0008).
Die Einheiten sind im K-0-0053 hinterlegt.
Beispiel:
für x, y, z in mm/s³ bzw. inch/s³ (siehe K-0-0054),
für alpha, beta, gamma in Grad/s³.
 - Wertebereich: ≥ 0.0 (Fließkommazahl)
 - Default: 0.0 (keine Ruckbegrenzung).

Verwendung Dieser Parameter wird von der Robot-Control für das Joggen (inkrementell und kontinuierlich) im Weltkoordinatensystem verwendet.

$$\Delta\alpha = j * t \quad \text{kann man die Verzögerungszeit} \quad t = \frac{\Delta\alpha}{j} \quad \text{bzw.}$$

$$t = \frac{K-0-0085}{K-0-0087} \quad \text{errechnen.}$$

Der maximale Ruck bei der Beschleunigung im Raumkoordinatensystem wird in K-0-0086 angegeben.

Die Ruck-Begrenzungswerte für das achsweise Joggen werden in A-0-0086 und A-0-0087 angegeben.

14.3.48 K-0-0100, Aktuelle Position, Kartesische Originalkoordinaten

- Funktion** Der Parameter zeigt die aktuelle Position der Kinematik im Original-Koordinatensystem an.
- Die aktuelle Position besteht je nach Kinematiktyp aus einer Liste von kartesischen Raumkoordinaten, der Orientierung des Werkzeugs und den Bandzählern.

14.3.49 K-0-0102, Aktuelle Bahngeschwindigkeit

- Funktion** Dieser Parameter liefert die aktuelle Bahngeschwindigkeit der Kinematik am TCP.
- Es werden nur die Positionierungsachsen berücksichtigt, nicht die Orientierungs- oder Bandachsen.
- Die Einheit ist mm/s bzw inch/s (siehe K-0-0054).

14.3.50 K-0-0103, Aktuelle Bahnbeschleunigung

- Funktion** Dieser Parameter liefert die aktuelle Bahnbeschleunigung der Kinematik am TCP.
- Es werden nur die Positionierungsachsen berücksichtigt, nicht die Orientierungs- oder Bandachsen.
- Die Einheit ist mm/s² bzw inch/s².

14.3.51 K-0-0104, Aktueller Bahnruck

- Funktion** Dieser Parameter liefert den aktuellen Bahnruck der Kinematik am TCP.
Es werden nur die Positionierungsachsen berücksichtigt, nicht die Orientierungs- oder Bandachsen.
Die Einheit ist mm/s³ bzw inch/s³.

14.3.52 K-0-0105, Aktuelle Position, Kartesische Werkstückkoordinaten

- Funktion** Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.53 K-0-0106, Koordinatenwerte vom Werkzeug

- Funktion** Der Parameter zeigt die Koordinatenwerte des aktiven Werkzeuges an, die drei Translationen X, Y Z und die drei Rotationen O1, O2, O3.

14.3.54 K-0-0107, Koordinatenwerte vom Werkstück

- Funktion** Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.55 K-0-0121 bis K-0-0136, Koordinatennamen

Die Parameter K-0-0121 ... 0136 enthalten die Namen der einzelnen Koordinaten. Diese Namen werden beim Anlegen eines RCL-Programms in der IndraWorks-Umgebung in die Compiler-Anweisung

```
;;WC_NAMES
```

übernommen.

Die einheitliche detaillierte Beschreibung entnehmen Sie bitte "K-0-0121, Koordinatenname K1".

14.3.56 K-0-0121, Koordinatennamen K1

- Funktion** Dieser Parameter enthält den Namen der ersten Koordinate des Raumkoordinatensystems der Kinematik.



Folgende Einschränkungen gelten:

- maximal 12 Zeichen
- A...Z, a...z, 0..9, '_'
- erstes Zeichen muss ein Buchstabe sein
- "K_1" identisch "k_1", keine Unterscheidung von Groß- und Kleinschreibung.

14.3.57 K-0-0140, Werkzeugbezeichnung

- Funktion** Der Parameter zeigt den Namen des aktiven Werkzeug-Koordinatensystems an. Nach dem Hochlauf wird "none active" angezeigt.

14.3.58 K-0-0141, Werkstückbezeichnung

- Funktion** Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.59 K-0-0150, Interpolierte Sollposition

- Funktion** Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.60 K-0-0152, Interpolierte Bahngeschwindigkeit

- Funktion** Dieser Parameter ist reserviert und besitzt keine Funktionalität.

RCL-Parameter

14.3.61 K-0-0153, Interpolierte Bahnbeschleunigung

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.62 K-0-0154, Interpolierter Bahnruddruck

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.63 K-0-0155, Schutzzonen ein/aus

Funktion Dieser Parameter gibt an, welche der 32 Schutzzonen aktiv oder inaktiv sind.

Aufbau

Bit	Inhalt
0	Aktivierung der 1. Schutzzone
1	Aktivierung der 2. Schutzzone
...	...
31	Aktivierung der 32. Schutzzone

Abb. 14-22: K-0-0155 Schutzzonen ein/aus

Verwendung Der Parameter K-0-0155 kann direkt in P4 (BB RUN) verändert werden. Ein Umschalten nach P2 und zurück nach BB ist nicht notwendig.

Da die Prüfung der Schutzzonen in der Satzvorbereitung durchgeführt wird, muss bei einem laufenden RCL-Programm jedoch berücksichtigt werden, dass die Aktivierung/Deaktivierung der Schutzzonen verzögert ausgeführt werden kann.

Beispiel:

K-0-0155: 0000 0000 0000 0000 0000 0000 0000 0100 ; Die 3. Schutzzone ist aktiviert.

14.3.64 K-0-0156, Schutzzone1 min. Punkt

Funktion Dieser Parameter legt den minimalen Punkt der 1. Schutzzone fest.

Mit dem maximalen Punkt aus K-0-0157 wird ein Quader als Schutzzone festgelegt.

Aufbau Der Parameter ist ein Listenparameter mit folgender Struktur:

- Element 1: Position der 1. Achse
- Element 2: Position der 2. Achse
- Element 3: Position der 3. Achse

Verwendung Der Parameter kann direkt in P4 (BB RUN) verändert werden. Ein Umschalten nach P2 ist nicht notwendig.

Da die Prüfung der Schutzzonen in der Satzvorbereitung durchgeführt wird, muss bei einem laufenden RCL-Programm jedoch berücksichtigt werden, dass die Änderung der Schutzzone verzögert ausgeführt werden kann.

Außerdem muss der maximale Punkt immer größer sein als der minimale Punkt, da sonst die Prüfung der Schutzzone inaktiv ist.

14.3.65 K-0-0157, Schutzzone1 max. Punkt

Funktion Dieser Parameter legt den maximalen Punkt der 1. Schutzzone fest.

Mit dem minimalen Punkt aus K-0-0156 wird ein Quader als Schutzzone festgelegt.

Aufbau Der Parameter ist ein Listenparameter mit folgender Struktur:

- Element 1: Position der 1. Achse

- Element 2: Position der 2. Achse
- Element 3: Position der 3. Achse

Verwendung

Der Parameter kann direkt in P4 (BB RUN) verändert werden. Ein Umschalten nach P2 ist nicht notwendig.

Da die Prüfung der Schutzzone in der Satzvorbereitung durchgeführt wird, muss bei einem laufenden RCL-Programm jedoch berücksichtigt werden, dass die Änderung der Schutzzone verzögert ausgeführt werden kann.

Außerdem muss der maximale Punkt immer größer sein als der minimale Punkt, da sonst die Prüfung der Schutzzone inaktiv ist.

14.3.66 K-0-0156/K-0-0157 Schutzzone1 min./max. Punkt bis K-0-0218/ K-0-0219 Schutzzone32 min./max. Punkt

Funktion: Durch Schutzzoneen können Bereiche definiert werden, die für die Roboterbewegung gesperrt werden. Es stehen 32 Schutzzoneen pro Kinematik zur Verfügung.

Funktionsbeschreibung: Ein Bewegungsbefehl im RCL-Programm besteht aus einem Anfangspunkt A und dem programmierten Endpunkt B.

In der Satzvorbereitung wird aus den Punkten A und B eine lineare Strecke AB interpoliert. Außerdem wird geprüft, ob die Strecke AB einen Schnittpunkt oder Berührungspunkt mit einer Seite einer Schutzzone gemeinsam hat. Diese Prüfung wird für jeden Bewegungsbefehl und alle der Kinematik zugeordneten Arbeitsräume durchgeführt.

Erkennt die Satzvorbereitung eine Verletzung einer Schutzzone im nächsten Satz, so wird die Kinematik bis zum Startpunkt des fehlerhaften Satzes verfahren und dann das RCL-Programm abgebrochen. Außerdem wird die Fehlermeldung „F12C0433 Arbeitsraumverletzung“ ausgegeben.

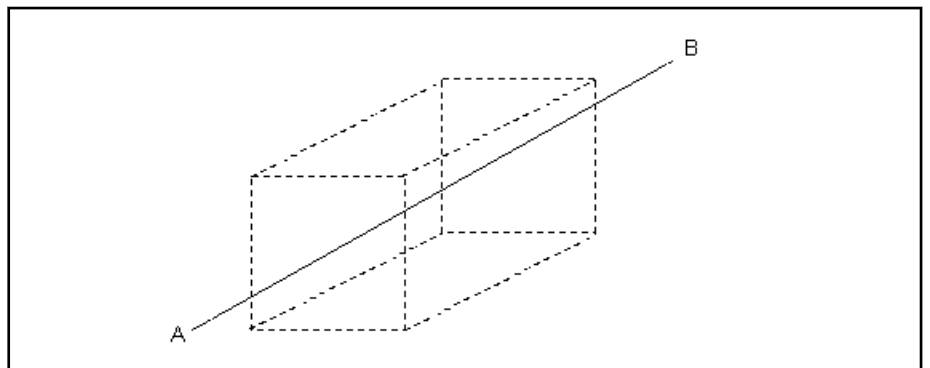


Abb. 14-23: Schutzzone mit min. Punkt (A) und max. Punkt (B)

Bedingungen: Bei der Programmierung der Schutzzoneen müssen die folgenden Punkte berücksichtigt werden:

- Jede Kinematik berücksichtigt immer nur die eigenen Schutzzoneen.
- Die Schutzzoneen berücksichtigen nur den TCP des Roboters, nicht die Mechanik.
- Bei PTP-Interpolation ist die Überwachung der Schutzzoneen stark eingeschränkt, da der zu verfahrenende Weg nicht definiert ist. Abweichungen von der linearen Strecke AB werden nicht erkannt.
- Bei CIRCULAR-Interpolation ist die Überwachung der Schutzzoneen stark eingeschränkt. Die Prüfung der Schutzzoneen beschränkt sich hier auch auf die lineare Strecke AB. Die Kreisbahn und der programmierte Zwischenpunkt werden nicht geprüft.

RCL-Parameter

- Beim Überschleifen mit VIA oder APPROX werden Abweichungen von der linearen Strecke AB nicht erkannt. Abweichungen durch Überschleifradien werden nicht berücksichtigt.
- Die Schutzzonenüberwachung ist nur im Automatikbetrieb wirksam. Im Handbetrieb erfolgt keine Überwachung der Schutzzonen.
- Bei einer bandsynchronen Verfahrbewegung wird der bandsynchrone Bewegungsanteil nicht für die Prüfung der Schutzzonen berücksichtigt.
- Da die Prüfung der Schutzzonen in der Satzvorbereitung durchgeführt wird ist es nicht möglich Schutzzonen dynamisch anzupassen. Eine Änderung der Schutzzonen wird erst aktiv, wenn der Satzpuffer abgearbeitet wurde. Daher kann sich eine Änderung oder Aktivierung / Deaktivierung einer Schutzzone bei einem laufenden RCL-Programm verzögern.
- Der maximale Punkt muss immer größer sein als der minimale Punkt da sonst die Prüfung der Schutzzone inaktiv ist.

Verwendung: Die Aktivierung und Anpassung der Schutzzonen erfolgt über die Parameter K-0-0155 bis K-0-0219.

Die K-Parameter können über die entsprechenden SPS-Funktionsbausteine oder den Parametereditor direkt in P4 (BB RUN) verändert werden.

Ein Umschalten in P2 ist nicht notwendig.

Über den Parameter K-0-0155 wird die Prüfung der Schutzzonen aktiviert (Bit 1 für die 1. Schutzzone bis Bit 32 für die 32. Schutzzone).

Beispiel: K-0-0155 0000 0000 0000 0000 0000 0000 0100 $\equiv$ dezimal: 4 $\Rightarrow$ Schutzzone 3 ist aktiv.

Über die nachfolgenden Parameter K-0-0156 bis K-0-0219 werden die minimalen und maximalen Punkte angegeben, durch die Quader als Schutzzonen definiert werden.

K-0-0155 0000 0000 0000 0000 0000 0000 0100 $\equiv$ dezimal: 4 $\Rightarrow$ Schutzzone 3 ist aktiv.
 K-0-0156 minimaler Punkt Schutzzone 1 (0, 0, 0),
 K-0-0157 maximaler Punkt Schutzzone 1 (20, 10, 10),
 K-0-0158 minimaler Punkt Schutzzone 2 (... , ... , ...),
 K-0-0159 maximaler Punkt Schutzzone 2 (... , ... , ...),

 K-0-0219 maximaler Punkt Schutzzone 32 (... , ... , ...).

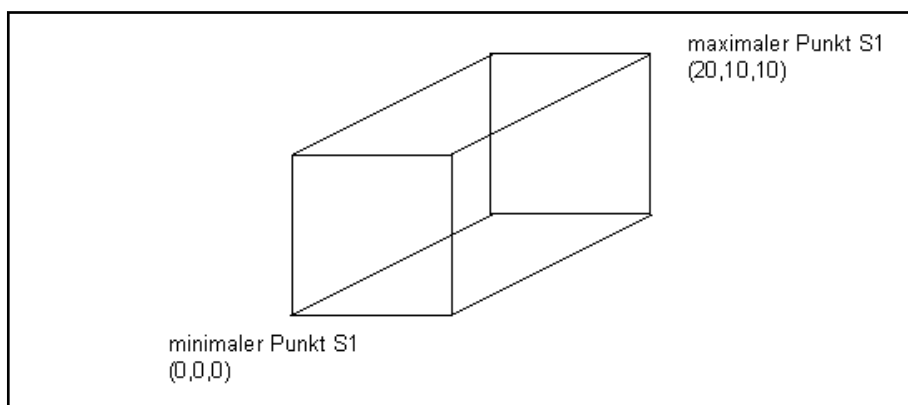


Abb. 14-24: Max. und min. Punkt der Schutzzone 1

14.3.67 K-0-0220, Dateitransfername

Funktion In diesem Parameter steht der Name des RCL Programms, das mit den Kommandoparametern K-0-1060 und K-0-1061 kopiert werden kann.

Wird als Endung * angegeben, werden alle Dateien mit gleichem Programmnamen, unabhängig von der Endung beim Ausführen eines der genannten Kommandoparameter kopiert.

Wird als Programmname *.* angegeben, werden alle Dateien beim Ausführen eines der genannten Kommandoparameter kopiert.



Folgende Einschränkungen gelten:

- maximal 12 Zeichen
- A...Z, a...z, 0..9, ' _ ' ! ' * '
- erstes Zeichen muss ein Buchstabe sein
- "Robi 1" identisch "ROBI 1"

14.3.68 K-0-0221, Name der zu kompilierenden Datei

Funktion In diesem Parameter steht der Name der zu kompilierenden QLL-Datei.
Eine Endung, z.B. .qll, wird automatisch abgeschnitten. Intern wird immer die Endung .QLL angefügt.

Wird über den Kommandoparametern K-0-1062 ein Kompilervorgang gestartet, wird die im K-0-0221 angegebene QLL-Datei kompiliert.



Folgende Einschränkungen gelten:

- maximal 12 Zeichen mit Endung
- maximal 8 Zeichen ohne Endung
- nur A...Z, a...z, 0..9, ' _ ' ' zulässig
- das erste Zeichen muss ein Buchstabe sein
- keine Unterscheidung zwischen Groß- und Kleinschreibung, "Robi 1" identisch "ROBI 1"

14.3.69 K-0-0240, Feedrate-Override

Funktion Mit diesem Parameter kann die Sollgeschwindigkeit prozentual zur Laufzeit geändert werden.

Ein Wert von 100% bedeutet, dass mit der vorgegebenen (programmierten) Geschwindigkeit die Achsen verfahren werden.

Der Prozentwert kann zwischen 0% und 600% eingestellt werden.



Bei 0% bleiben die Achsen stehen, ohne den Zustand der Kinetik-Zustandsmaschine zu ändern.

Solange im RCL-Programm der Befehl VFIX aktiv ist, bleibt dieser Parameter ohne Wirkung

14.3.70 K-0-0241, Feedrate-Override-Steuerwort

Funktion Mit diesem Parameter wird der RoCoFeedrate-Override gesteuert.

Aufbau

Bit	Bezeichnung/ Funktion
0	reserviert
1	Bit 1 = 0: RoCoFeed-Override im K-0-0240 ist wirksam, Bit 1 = 1: Vorschub Halt, der Wert in K-0-0240 bleibt unverändert.
15..2	reserviert

RCL-Parameter

Verwendung Bit 1:

Mit Setzen des Bit 1 wird der Vorschub Halt wirksam.

14.3.71 K-0-0242, Acceleration-Override

Funktion Mit diesem Parameter kann die Beschleunigung prozentual zur Laufzeit geändert werden.

Ein Wert von 100% bedeutet, dass mit der vorgegebenen (programmierten) Beschleunigung die Achsen verfahren werden.

Der Prozentwert kann zwischen 1% und 600% eingestellt werden.



Das Setzen von K-0-0242 wirkt in der Satzvorverarbeitung, also nicht sofort.

Solange im RCL-Programm der Befehl AFIX aktiv ist, bleibt dieser Parameter ohne Wirkung

14.3.72 K-0-0243, Deceleration-Override

Funktion Mit diesem Parameter kann die Bremsverzögerung prozentual zur Laufzeit geändert werden.

Ein Wert von 100% bedeutet, dass mit der vorgegebenen (programmierten) Bremsverzögerung die Achsen verfahren werden.

Der Prozentwert kann zwischen 1% und 600% eingestellt werden.



Das Setzen von K-0-0243 wirkt in der Satzvorverarbeitung, also nicht sofort.

Kleine Werte in K-0-0243 führen zu sehr langen Bremswegen.

Ist im RCL-Programm die Überschleiffunktion PROGR_SLOPE aktiv, kann dies dazu führen, dass bei zu kurzem Bremsweg ruckartig angehalten wird.

14.3.73 K-0-1060, Kommando: Lade Datei von CF

Funktion Dieser Kommando-Parameter kopiert die im Parameter K-0-0220 angegebene(n) Datei(en) von der Compact Flash-Karte in den inneren Speicher der Steuerung.

Das Quell-Verzeichnis "<User-Partition> / RC_Files" der Dateien auf der Compact Flash ist fest vorgegeben.

- Verwendung**
- Eingabe des Binärwertes "0000000000000011" (=0x0003) in diesen Parameter. Dies löst den Kopiervorgang aus.
 - Nach Abschluss des Kopiervorganges setzt die Steuerung automatisch den Parameter zurück auf "0000000000000000".

14.3.74 K-0-1061, Kommando: Speichere Datei auf CF

Funktion Dieser Kommando-Parameter kopiert die im Parameter K-0-0220 angegebene(n) Datei(en) vom inneren Speicher der Steuerung auf die Compact Flash-Karte.

Das Ziel-Verzeichnis "<User-Partition> / RC_Files" der Dateien auf der Compact Flash ist fest vorgegeben.

- Verwendung**
- Eingabe des Binärwertes "0000000000000011" (=0x0003) in diesen Parameter. Dies löst den Kopiervorgang aus.
 - Nach Abschluss des Kopiervorganges setzt die Steuerung automatisch den Parameter zurück auf "0000000000000000".

14.3.75 K-0-1062, Kommando: Compiliere Datei auf MLC

Funktion Dieser Parameter kompiliert die im Parameter K-0-0221 angegebene QLL-Datei.



Die zu kompilierende QLL-Datei muss im SRAM/RAM der Steuerung abliegen.

Über den Parameter K-0-1060 kann die QLL-Datei ins SRAM/RAM der Steuerung geladen werden.

- Verwendung**
- Eingabe des Binärwertes "0000000000000011" (=0x0003) in diesen Parameter. Dies löst den Kompiliervorgang aus.
 - Nach Abschluss des Kompilierens setzt die MLC automatisch den Parameter zurück auf "0000000000000000".

14.3.76 K-0-2200, Programmierte Position 1 (aktiv)

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.77 K-0-2201, Programmierte Position 2 (aktiv)

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.78 K-0-2202, Typ der aktiven programmierten Position

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.79 K-0-2203, Interpolationstyp

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.80 K-0-2204, Aktuell aktiver QLL-Dateiname

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.81 K-0-2205, Aktuell aktive Zeilennummer

Funktion Dieser Parameter ist reserviert und besitzt keine Funktionalität.

14.3.82 K-0-2300, SPS-Bit-Eingang (RCL)

Funktion Der Parameter zeigt die Bit-Eingänge an, die von RCL in Richtung SPS übertragen werden.

14.3.83 K-0-2301, SPS-Ganzzahl-Eingang (RCL)

Funktion Der Parameter zeigt die Ganzzahl-Eingänge an, die von RCL in Richtung SPS übertragen werden.

14.3.84 K-0-2302, SPS-Gleitkommazahl-Eingang (RCL)

Funktion Der Parameter zeigt die Gleitkommazahl-Eingänge an, die von RCL in Richtung SPS übertragen werden.

15 Kinematik-Benutzeroberfläche

15.1 Allgemeines

IndraWorks bietet für die Realisierung einer Roboter-Anwendung entsprechende Benutzerschnittstellen. Diese lassen sich in folgende Komponenten gliedern:

- [Robot Control Menüstruktur, Seite 208,](#)
Die Robot Control Menüstruktur befindet sich im MLC-Projektbaum. Sämtliche Icons der Werkzeugleisten sind mit Hilfetexten versehen, die angezeigt werden, sobald der Mauszeiger eine kurze Weile auf dem Icon verweilt.
Zusätzlich werden Informationen zu den jeweiligen Funktionen in der Statusleiste eingeblendet.
- **Kinematik-Wizard**
Das Anlegen einer Kinematik erfolgt über den Kinematik-Wizard. Hierüber können Kinematikeigenschaften konfiguriert werden.
- [RCL-Editor, Seite 209,](#)
Mit dem RCL-Editor werden Anwenderprogramme in RCL (Robot-Control-Language) erstellt und geprüft.
- [Compiler, Seite 214,](#)
Der Compiler übersetzt den Quelltext in einen maschinenlesbaren Code. Die Ausgabetexte und die Fehlermeldungen werden in ein Ausgabefenster umgeleitet und angezeigt. Über einen Doppelklick kann zur Position im Programm gesprungen werden.
- [Debugger, Seite 215,](#)
Der Debugger erleichtert das Auffinden von Programmfehlern. Im Debug-Modus können die Werte zuvor deklarierter Variablen angezeigt und während der Programmausführung überwacht werden.
- **Punkte-Editor**
Der Punkte-Editor dient zum Eingeben, Löschen und Verändern von Punkten. Die Punkte werden aus der Punktedatei (Online oder Offline) gelesen und tabellarisch dargestellt. Punkte können auch als Punktefelder deklariert werden. Die Deklaration erfolgt mit Hilfe des DEF-Befehls (siehe [Kap. 7.3 "RCL-Variablendeklaration" auf Seite 57](#)).
- **Datei-Transfer**
Es können Dateien zur Steuerung übertragen sowie auf der Steuerung gelöscht werden.
- **Diagnose-Übersicht**
Über den Dialog "Status" der entsprechenden Kinematik werden alle relevanten Statusinformationen angezeigt.

15.2 Robot Control Menüstruktur

15.2.1 Menüpunkt Robot

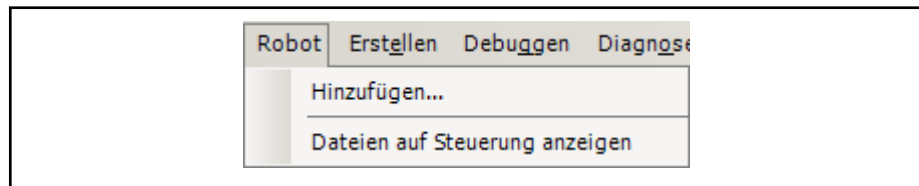


Abb. 15-1: Menüpunkt Robot

Durch Selektion des Knotens "Robot" im Projekt wird der Menüpunkt **Robot** im IndraWorks-Hauptmenü aktiviert. Mit **Hinzufügen...** wird der Wizard zum Konfigurieren einer neuen Kinematik gestartet. Über **Dateien auf Steuerung anzeigen** können die RCL-Programme, die sich auf der Steuerung befinden, angezeigt werden. Die zuvor beschriebenen Funktionen sind auch über das Kontextmenü des Knotens "Robot" zu erreichen.

15.2.2 Menüpunkt Kinematik

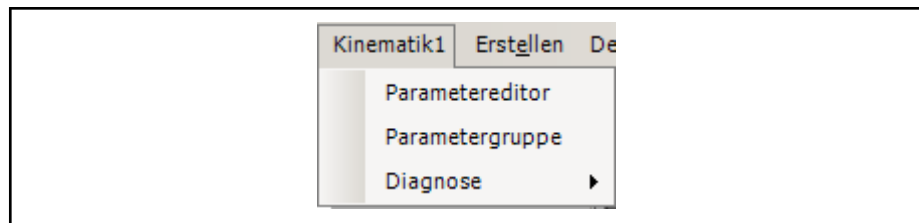


Abb. 15-2: Menüpunkt Kinematik

Der Kinematik-Knoten wird nach Auswahl des Kinematiknamens aktiv. Hier-nach wird der Menüpunkt im IndraWorks-Hauptmenü aktiviert.

15.2.3 Menüpunkt Programme

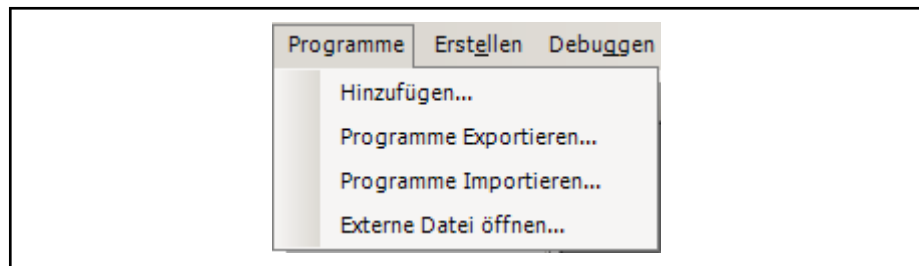


Abb. 15-3: Menüpunkt Programme

Durch Selektion des Knotens "Programme" im Projekt wird der Menüpunkt **Programme** im IndraWorks-Hauptmenü aktiviert.

Mit **Hinzufügen...** wird ein neues RCL-Programm im Editor angelegt.

Mit **Programme Exportieren...** können alle RCL-Programme außerhalb des Projektes abgelegt werden.

Mit **Programme Importieren...** können alle RCL-Programme, die sich außerhalb des Projektes befinden, ins Projekt importiert werden.

Mit **Externe Datei öffnen...** kann ein RCL-Programm, das sich außerhalb des Projektes befindet, im Editor geöffnet werden.

15.2.4 Menüpunkt Prog.QLL

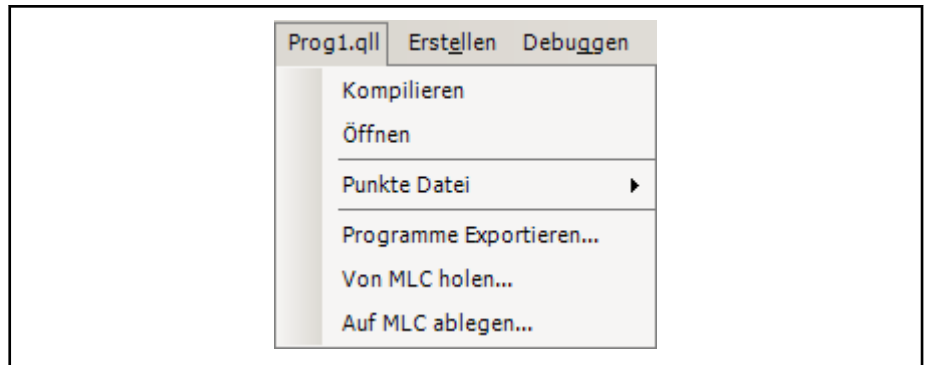


Abb. 15-4: Menüpunkt Prog.QLL

Durch Selektion des Knotens "Prog.QLL" im Projekt wird der Menüpunkt **Prog.QLL** im IndraWorks-Hauptmenü aktiviert. Als Name im Menüpunkt bzw. des Knotens wird jeweils der Name des angelegten RCL-Programms angezeigt.

Mit **Kompilieren** wird aus dem Quelltext (.QLL) ein lauffähiges Roboterprogramm erzeugt.

Mit **Öffnen** wird das selektierte RCL-Programm im Editor geöffnet.

Mit **Punkte Datei** kann über die Untermenüs eine Punktedatei geöffnet bzw. abgespeichert werden.

Mit **Programme Exportieren ...** kann das selektierte RCL-Programm (QLL) außerhalb des Projektes abgelegt werden.

Mit **Von MLC holen...** kann das selektierte RCL-Programm aus der Steuerung ins Projekt geladen werden.

Mit **Auf MLC ablegen...** kann das selektierte RCL-Programm in die Steuerung übertragen werden.

15.3 RCL-Editor

15.3.1 Allgemeines

Mit dem Texteditor werden Anwenderprogramme in der Programmiersprache RCL (**R**obot- **C**ontrol-**L**anguage) erstellt, geprüft und in Betrieb genommen.

Mittels Optionen kann das Verhalten, Aussehen, Farben, Zeilennummerierung, Tabulatorbreiten, Einrückungen etc. eingestellt werden.

15.3.2 Merkmale

Besondere Merkmale des Editors sind:

- Syntax-Highlighting (über Optionsdialog frei konfigurierbar)
- Kopieren, Einfügen, Suchen, Ersetzen, usw.
- Lesezeichenverwaltung
- Zeilennummerierung
- Tooltip
- Automatische Schlüsselwortvorgabe
- Code Falten
- Intellisense
- Code Schablonen (Templates)

- Debugging

15.3.3 Optionen

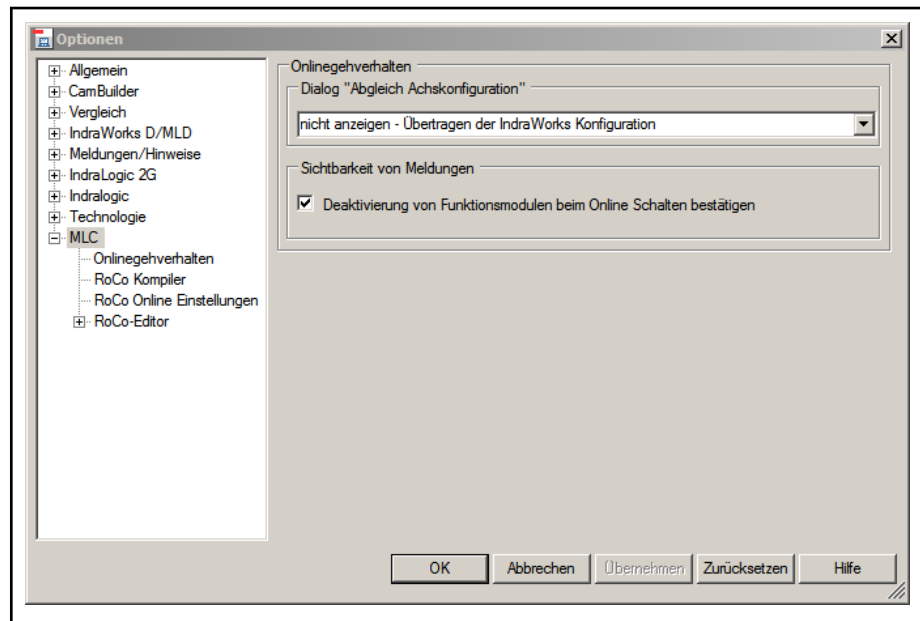


Abb. 15-5: Dialog RoCo-Editor Optionen

Unter dem IndraWorks Menüpunkt **Extras ► Optionen** unter dem Baummenü "RoCo-Editor" sind Funktionen zur Unterstützung der Quelltexteingabe standardmäßig voreingestellt. Sie können durch den Anwender geändert werden.

15.3.4 Neues RCL-Programm erstellen

Über folgende Menüs kann ein neues RCL-Programm erstellt werden:

- Im IndraWorks-Hauptmenü über **Programme ► Hinzufügen**
- oder -
- über den Programm-Kontextmenüpunkt **Hinzufügen...** der Kinematik

Der per Default vorgegebene Programmname kann durch den Anwender überschrieben werden. Wird die Eingabe mit OK quittiert, wird der RCL-Editor gestartet, welcher den Programm-Rumpf des neuen RCL-Programms enthält. Das Programm erscheint nach dem ersten Speichern im Menüpunkt.

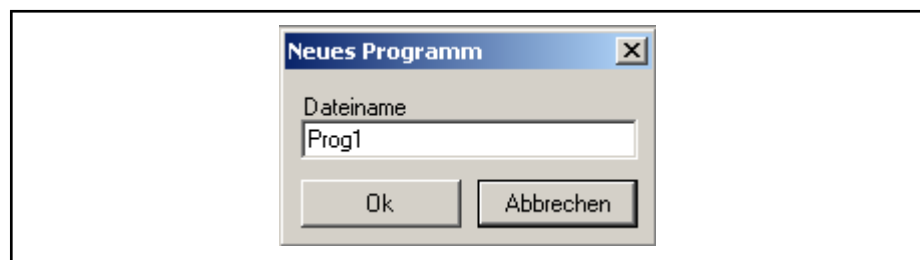


Abb. 15-6: Dialog "Neues Programm hinzufügen"

15.3.5 Arbeiten mit dem Editor

Allgemeine Editorfunktionen

Der RoCo-Editor hat nützliche Techniken üblicher Windows-Editoren integriert:

- Suchen
- Suchen nach Dateien

- Ersetzen
- Blockmarkierung
- Groß-/Kleinförmierung
- Drag & Drop

Suchen und Ersetzen

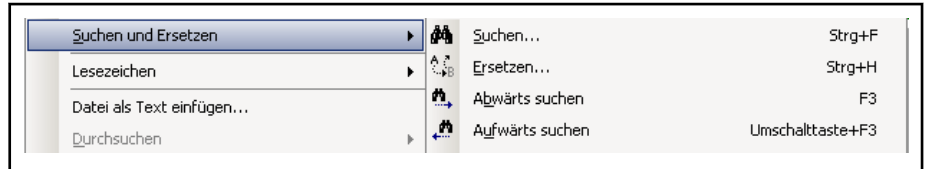


Abb. 15-7: Menüpunkt "Suchen und Ersetzen"

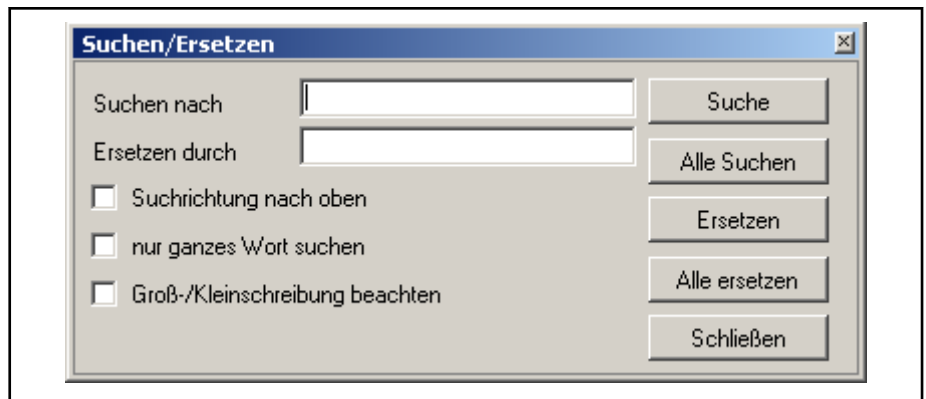


Abb. 15-8: Dialog "Suchen/Ersetzen"

Lesezeichen

Der Texteditor stellt eine Lesezeichenverwaltung bereit. Lesezeichen können Zeilen hinzugefügt werden, um das Navigieren in langen Programmen zu erleichtern. Die im folgenden beschriebenen Befehle ermöglichen das Setzen von und Navigieren mit den Lesezeichen. Sie sind standardmäßig im Untermenü 'Lesezeichen' im "Bearbeiten"-Menü enthalten. Außerdem sind sie über die Symbolleiste "Lesezeichen" zu erreichen.

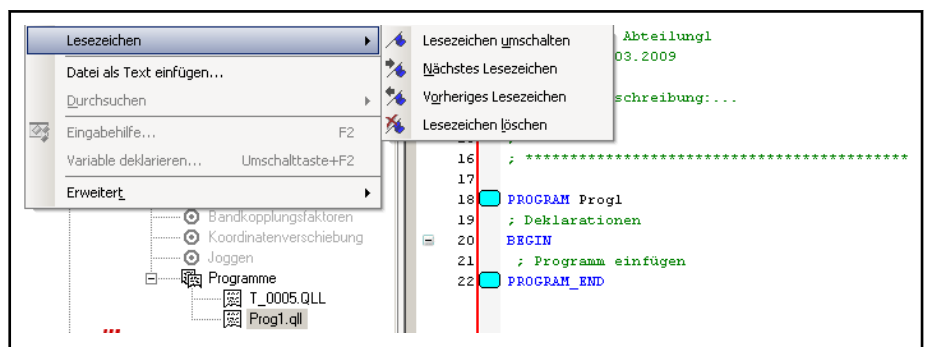


Abb. 15-9: Menüpunkt "Lesezeichen"

Kommentare

Es gibt die Möglichkeit, Kommentare einzufügen. Der Kommentar beginnt mit ";" und endet am Ende der Zeile.

Ausschneiden, Kopieren, Einfügen

Der Anwender markiert mit der linken Maustaste oder über eine Umschalt-Cursortastenkombination den gewünschten Bereich im Editor. Die Komman-

Kinematik-Benutzeroberfläche

dos "Ausschneiden", "Kopieren", "Einfügen" können auf diesen Bereich entsprechend angewendet werden.

Die Kommandos sind im "Bearbeiten"-Menü und über das Kontextmenü des Editors erreichbar.

Unterstützung der Quelltexteingabe

- Parameterinformation**

Wird der Mauszeiger über ein Unterprogramm geführt, werden die Übergabeparameter angezeigt.

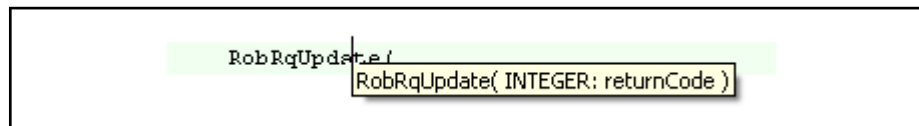


Abb. 15-10: Parameterinfo

Werden der Name eines Unterprogramms und eine öffnende Klammer eingegeben, werden die Übergabeparameter ebenfalls angezeigt.

- Codevervollständigung**

Diese Funktion dient zur sinnvollen Ergänzung von Benutzereingaben, wobei sich die Eingabemöglichkeit äquivalent zum Fortschritt und ggf. Semantik der Benutzereingabe verkleinert. Durch Eingabe des ersten Zeichens bzw. durch die Tastenkombination <Strg>+<Leertaste> wird eine Auswahl-Liste eingeblendet, die den RoCo-Befehlssatz und bereits deklarierte Variablen enthält. Mit der Taste <Return> wird das in der Auswahl-Liste selektierte Objekt übernommen.

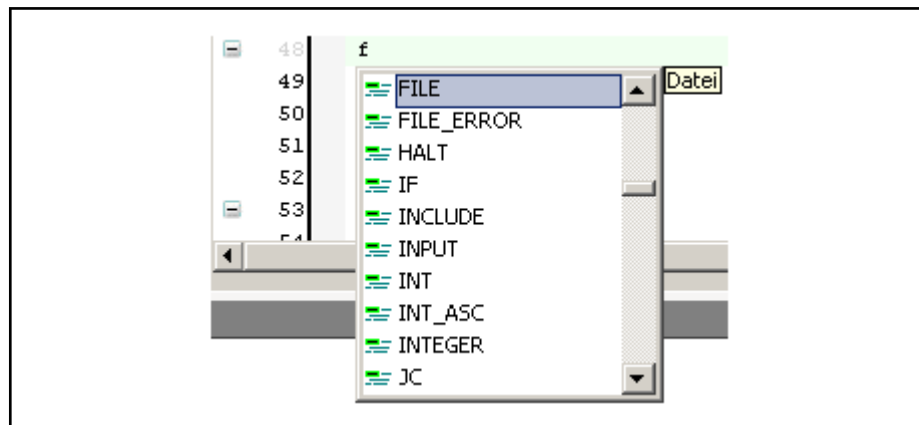


Abb. 15-11: Codevervollständigung

- Elementauflistung**

Bei Eingabe einer Verbundvariablen und eines anschließenden Punkts werden die Elemente der Verbundvariablen zur Auswahl angezeigt. Über die Pfeiltasten kann das gewünschte Element ausgewählt werden. Mit <Return> wird die Auswahl übernommen.

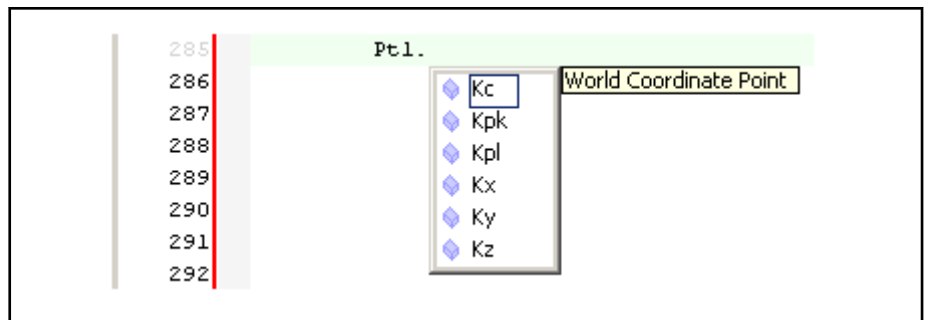


Abb. 15-12: Elementauflistung

- **Typinformation**

Wenn Sie den Mauszeiger über eine Variable führen, wird die Typinformation in Form eines Infotextes angezeigt.

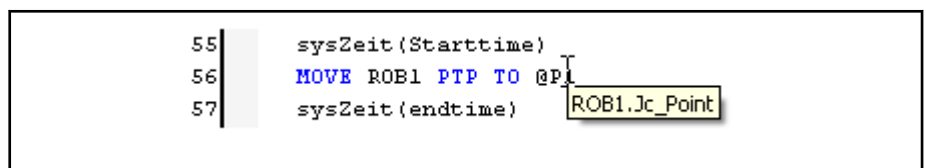


Abb. 15-13: Typinformation

- **Cursor automatisch einrücken**

Es wird beim Schreiben eines Quelltextes der Einzug der vorherigen Zeile automatisch in der nächsten Zeile übernommen.

- **Tabulatoren**

Die Länge der Tabulatorenspünge kann über "Optionen" eingestellt werden.

- **Syntaxfärbung**

Die Schlüsselbegriffe im Quelltext sind andersfarbig dargestellt. Die Initialvorgaben können über "Optionen" durch den Anwender verändert werden.

- **Spezialfunktionen/Codeschablonen**

Die Eingabe von Spezialfunktionen wird im Rahmen der Codevervollständigung unterstützt.

Durch Eingabe des RoCo-Befehls für Spezialfunktionen wird dieser in der Drop-Down Liste selektiert.

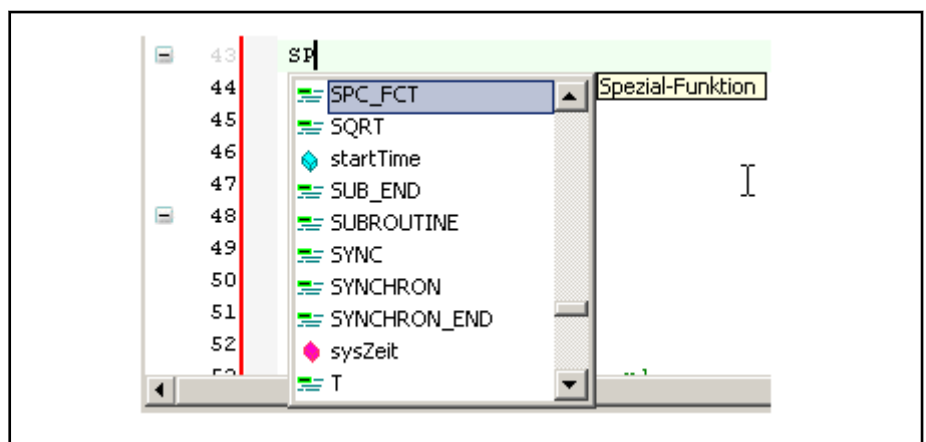


Abb. 15-14: Auswahl Befehl "Spezial-Funktion"

Kinematik-Benutzeroberfläche

Mit der Taste <Return> wird das in der Auswahl-Liste selektierte Objekt übernommen. Es werden in einer weiteren Drop-Down Liste die verfügbaren Spezialfunktionen zur Auswahl dargestellt.

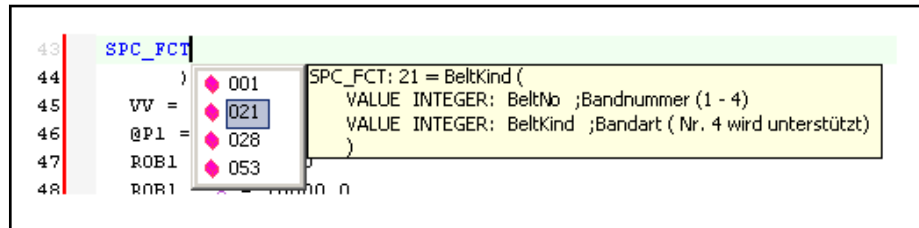


Abb. 15-15: Auswahl der Spezial-Funktion

Mit der Taste <Return> wird die in der Auswahl-Liste selektierte Spezialfunktion übernommen. Die per Default angelegten Namen der Übergabeparameter können vom Anwender geändert werden.

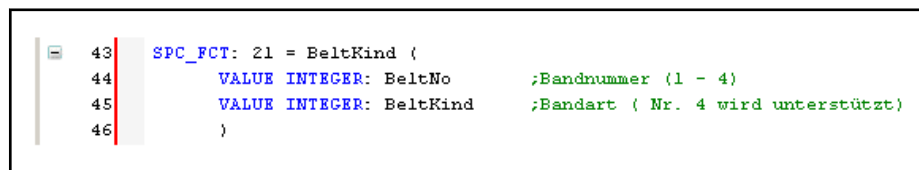


Abb. 15-16: Eingefügte Spezial-Funktion

15.4 Compiler

Der Compiler erzeugt aus dem Quelltext das Programm in maschinenlesbarer Form und überprüft jede einzelne Zeile auf korrekte Befehlssyntax. Bei selektiertem RCL-Programm im Editor kann das Kompilieren des Quelltextes durchgeführt werden.

Das Kompilieren kann wie folgt gestartet werden:

- über das Hauptmenü

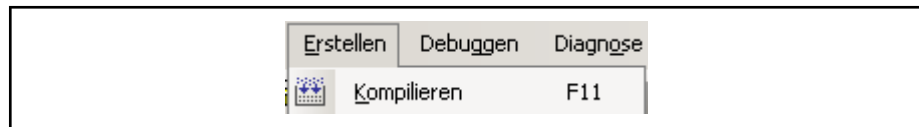


Abb. 15-17: Auswahl des Compilers

- über das Kontextmenü im Editor

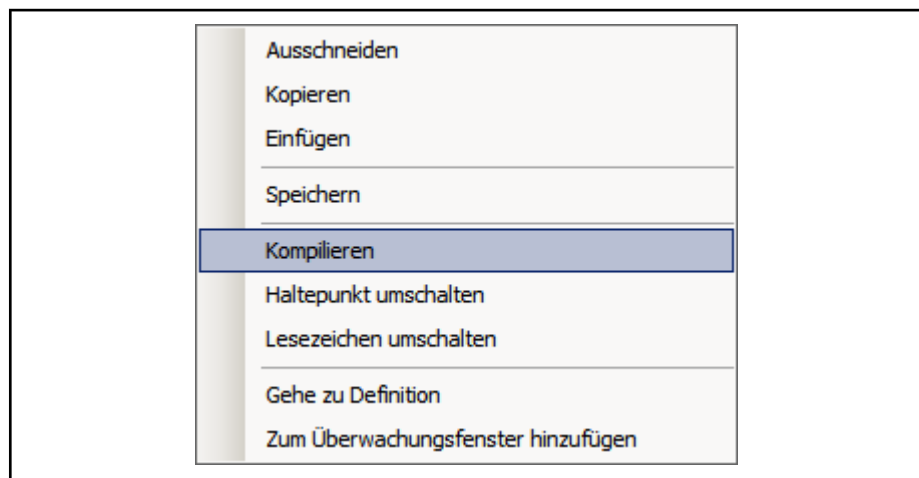


Abb. 15-18: Auswahl des Compilers

- über die Funktionstaste <F11>

Kinematik-Benutzeroberfläche

Die ausgewählte Datei wird in die Maschinensprache übersetzt und im Projekt gespeichert. Über die Option "RoCo Compiler" kann zusätzlich das sofortige Speichern in der Steuerung aktiviert werden.

Die Ausgabetexte sowie Fehlermeldungen werden in das andockbare Fenster "Ausgabe" umgeleitet.

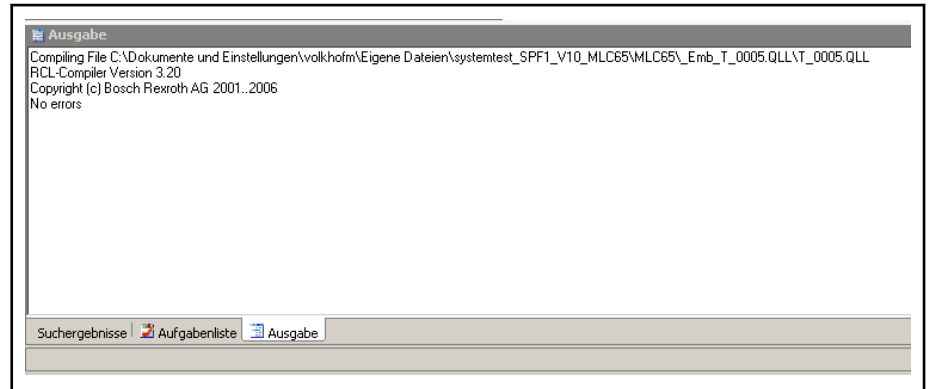


Abb. 15-19: Ausgabefenster des Compilers

Werden Fehlermeldungen im Ausgabebereich angezeigt, kann über einen Doppelklick auf eine Fehlermeldung im Fenster **Aufgabenliste** direkt zu der Position im Programm gesprungen werden, an der der Fehler entdeckt wurde.

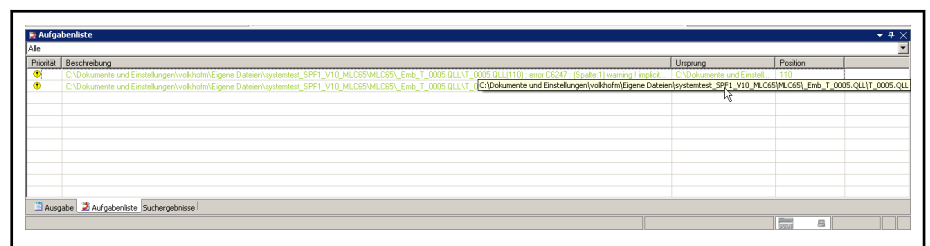


Abb. 15-20: Ausgabe von Compiler-Fehlermeldungen

15.5 Debugger

15.5.1 Allgemeines

Sie können das Ausführen eines Programms im Debug-Modus auf logische Programmfehler untersuchen und den Inhalt von Variablen im andockbaren Fenster "Überwachung" anzeigen.

15.5.2 Setzen und Löschen von Haltepunkten

Sie haben die Möglichkeit Haltepunkte zu setzen, um den Programmablauf an einer festgelegten Stelle zu unterbrechen.

Setzen Sie den Cursor an die Stelle im Programm, an die Sie den Haltepunkt setzen möchten.

Kinematik-Benutzeroberfläche

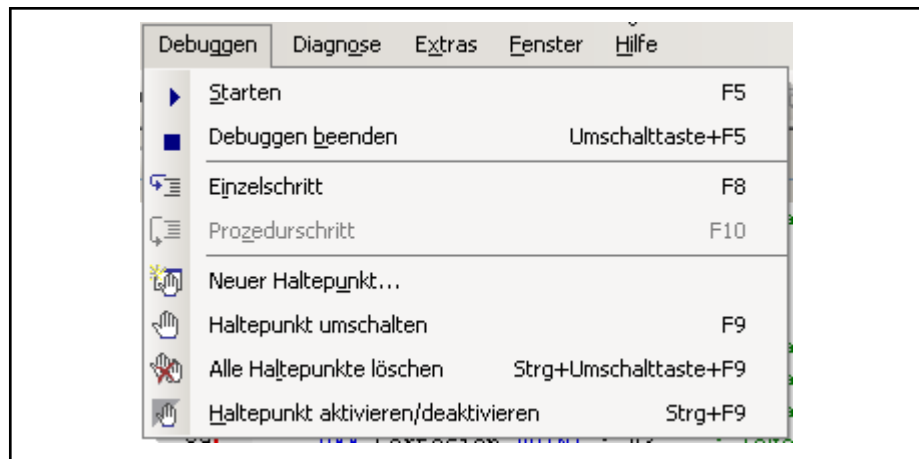


Abb. 15-21: Menü "Debuggen"

Klicken Sie die Funktion **Neuer Haltepunkt...** im Menü **Debuggen** an, um den Haltepunkt zu setzen bzw. wieder zu löschen. Über die Funktion **Haltepunkt umschalten** bzw. über die Funktionstaste <F9> kann ebenfalls ein Haltepunkt gesetzt und wieder gelöscht werden. Sie können Haltepunkte auch durch Klicken in die Indikatorleiste (linker grauer Rand des Quelltextfensters) setzen bzw. wieder löschen.

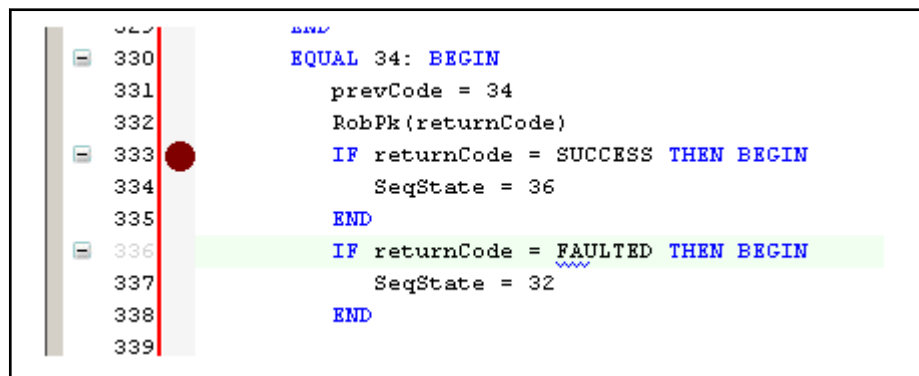


Abb. 15-22: Gesetzter Haltepunkt

Über die Funktion **Alle Haltepunkte löschen** im Menü **Debuggen** oder die Tastenkombination <Strg>+<Umschalttaste>+<F9> können alle Haltepunkte gelöscht werden.

15.5.3 Programm debuggen

Über den Menüpunkt **Debuggen ▶ Starten** bzw. die Funktionstaste <F5> wird das Programm im Debug-Modus ausgeführt.

⚠ WARNUNG

Mit dem Befehl **Debuggen ▶ Starten** wird das Programm direkt gestartet.

Treffen Sie Vorkehrungen, damit dies gefahrlos möglich ist. So können Sie z.B. auf den ersten Befehl im RCL Programm einen Haltepunkt setzen.

Vor dem Start wird geprüft, ob das Programm auf dem PC oder auf der Steuerung geändert wurde. Werden Änderungen, z.B. der Punktedatei erkannt, so erfolgt ein entsprechender Hinweis und Sie können wählen, ob die Punktedatei auf die Steuerung übertragen oder der Debugger beendet werden soll.



Voraussetzung für die korrekte Funktionsweise dieser Überwachung ist, dass Datum und Uhrzeit von PC und Steuerung übereinstimmen.

Siehe Menüpunkt **Diagnose ► Gerätestatus** im Kontextmenü der Steuerung im IndraWorks Projektexplorer.



Über das Icon in der RoCo-Werkzeugleiste kann der Debugger auf ein aktives Programm aufgeschaltet werden.

Der Programmablauf wird am ersten Haltepunkt unterbrochen.

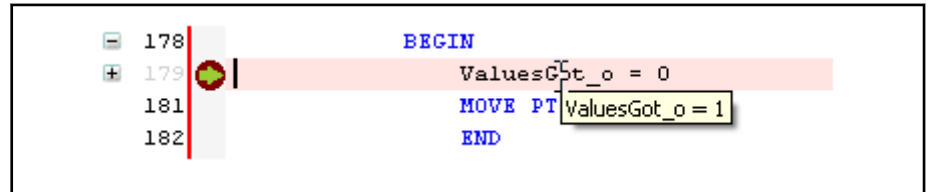


Abb. 15-23: Programmablauf hält am ersten Haltepunkt an

Über die Funktionstaste <F5> kann das Programm nach dem Halt an einem Haltepunkt wieder fortgesetzt werden. Die Funktionstaste <F8> ermöglicht es, das Programm im Einzelschrittverfahren zu testen. Der Debugger kann mit <Umschalttaste>+<F9> wieder beendet werden.

Wird der Cursor auf eine Variable gestellt, so wird der aktuelle Wert der Variable im Tooltip angezeigt.



Die Variablenwerte werden nur angezeigt, wenn das Programm auf einem Haltepunkt steht.

15.5.4 Variablen überwachen

Im Debug-Modus besteht die Möglichkeit, Variablen ins Überwachungsfenster zu übernehmen. Die Variablen können hierzu direkt ins Fenster eingegeben oder mittels Drag & Drop dort abgelegt werden.



Über das Icon in der RoCo-Werkzeugleiste wird das Überwachungsfenster geöffnet.



Die Variablenwerte werden nur angezeigt, wenn das Programm auf einem Haltepunkt steht.

16 Anwendung der RoCo-Multikinematik-Technik

16.1 Allgemeines

- Bis zu 16 Kinematiken (Roboter) können durch eine MLC-Steuerung unterstützt werden.
- Unterschiedliche Robotertypen sind nebeneinander in einer MLC ansteuerbar (pro Kinematik ein Robotertyp, wie z.B. Karteser, SCARA, Delta, ...).
- Bis zu 16 Achsen können pro Kinematik unterstützt werden (inkl. Bandachsen).
- Bis zu 4 Bandachsen können pro Kinematik unterstützt werden.
- Mehrere Roboter können an einem Band synchron arbeiten (Bandachsen können mehreren Kinematiken zugeordnet werden).
- Der Betriebsmodus ist für jede Kinematik einzeln wählbar (Programmed Motion, Manual Motion, Discrete-/Synchronized Motion).
- Für jede Kinematik existiert ein eigenes PLC-Shared-Memory-Interface (KinData).

16.2 Anlegen von Kinematiken

16.2.1 Allgemeines

Im Parameter C-0-2484, Kinematikkonfigurationsliste, werden im Offline-Zustand von IndraWorks durch **Robot ► Hinzufügen** die Kinematiken angelegt.

Anwendung der RoCo-Multikinematik-Technik

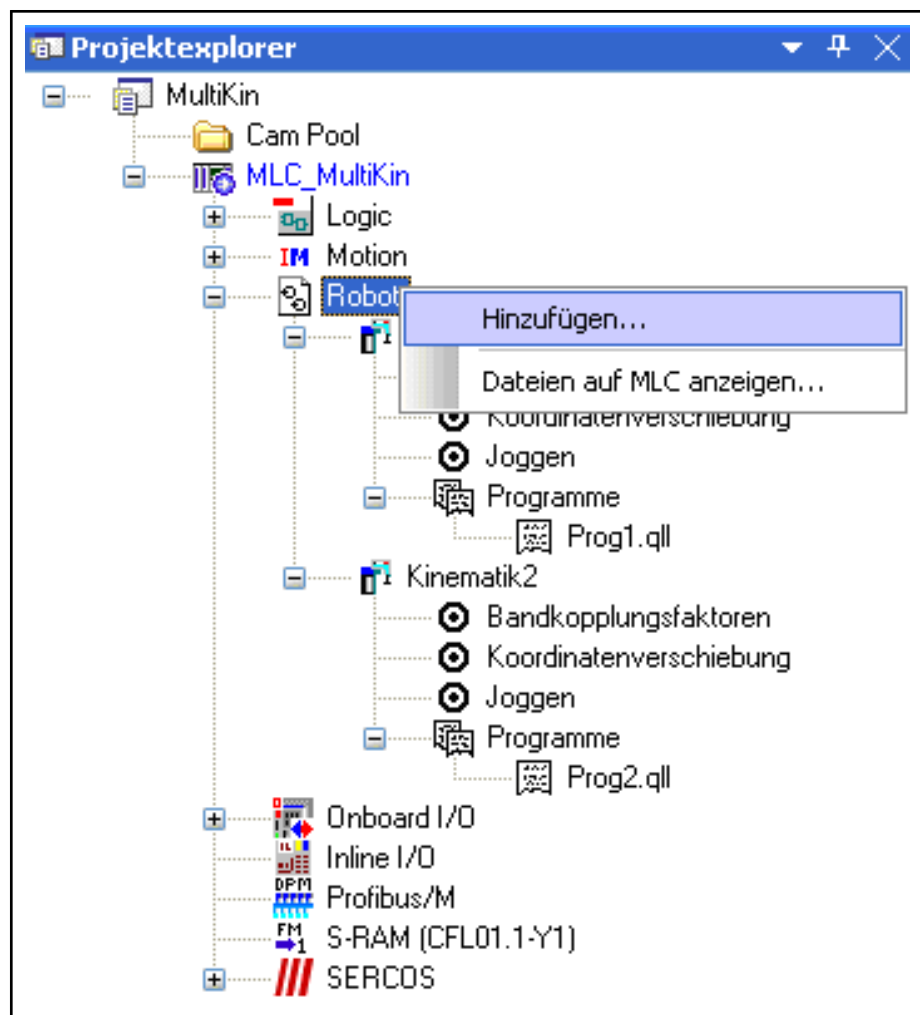


Abb. 16-1: Anlegen einer Kinematik

Beim Anlegen einer Kinematik wird dieser automatisch eine fortlaufende Nummer vergeben. Im Kinematik-Wizard, der automatisch beim Hinzufügen gestartet wird, muss der Anwender der Kinematik einen eindeutigen symbolischen Bezeichner (Gerätename) geben.

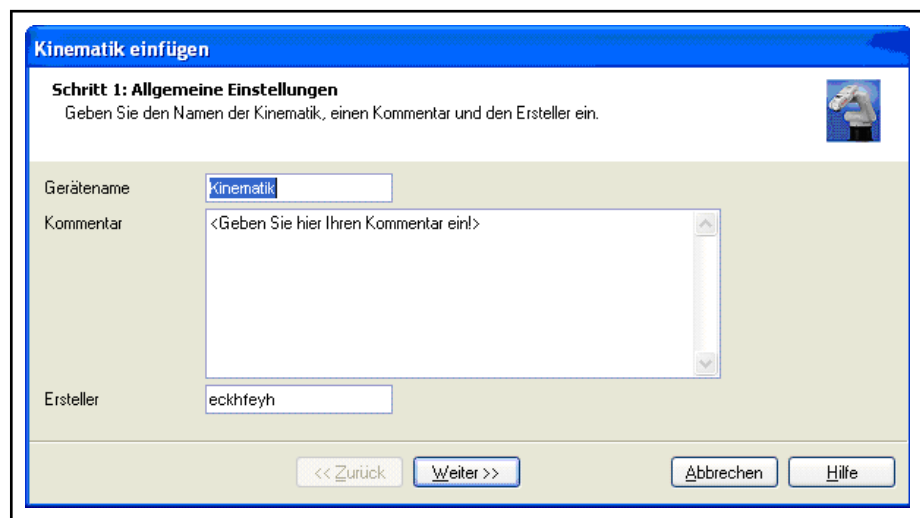


Abb. 16-2: Eingabe des Kinematik-Namens

Anwendung der RoCo-Multikinematik-Technik

Über den Namen der Kinematik kann das PLC-Programm unabhängig von der vergebenen Nummer auf die Kinematik zugreifen. Desweiteren werden beim Anlegen eines RCL-Programms die Nummer und der Name der Kinematik im Kopf des RCL-Programms automatisch eingetragen. Mit diesem Eintrag kann später sichergestellt werden, dass nur Programme, die für diese Kinematik geschrieben wurden, auf dieser Kinematik ausgeführt werden können.

16.2.2 Festlegung des Kinematiktyps (Robotertyp, K-0-0031)

Mit **Weiter** wird dann der Typ der Kinematik festgelegt.

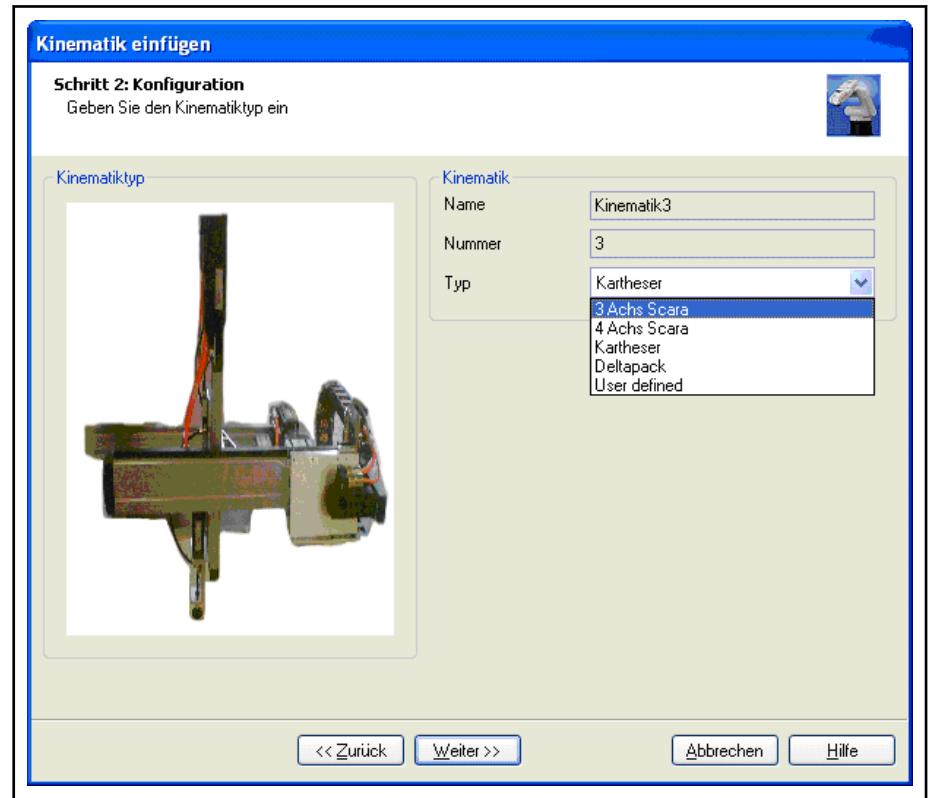


Abb. 16-3: Festlegen des Kinematik-Typs

Hier stehen immer alle Typen (SCARA, Karteser, Delta, ...) zur Verfügung. Es können also beliebige Robotertypen in einer MLC gemischt betrieben werden.

16.2.3 Festlegung der Achszuordnung (K-0-0008)

Eine Steuerung MLC kann bis zu 16 Roboter (Kinematiken) mit jeweils bis zu 16 Achsen gleichzeitig ansteuern. Im Vollausbau wären dies theoretisch 256 Achsen. Diese theoretische Achsanzahl wird aber durch die maximale Achsanzahl der Steuerung begrenzt.

- Auf der L40.2 sind maximal 32 Achsen möglich, also maximal 16 Kinematiken mit je 2 Achsen.
- Auf der L65 sind maximal 64 Achsen möglich, also maximal 16 Kinematiken mit je 4 Achsen.

Die in der Steuerung verfügbaren Achsen stehen im Parameter C-0-0484, Achskonfigurationsliste. Diese Achsen können den jeweiligen Kinematiken im Parameter K-0-0008, Achskonfigurationsliste, zugeordnet werden. Der Kinematik-Wizard, der beim Anlegen einer Kinematik automatisch gestartet wird,

Anwendung der RoCo-Multikinematik-Technik

bietet für Positionier- und Orientierungsachsen nur diejenigen Achsen aus dem C-0-0484 an, die noch keiner Kinematik zugeordnet wurden.

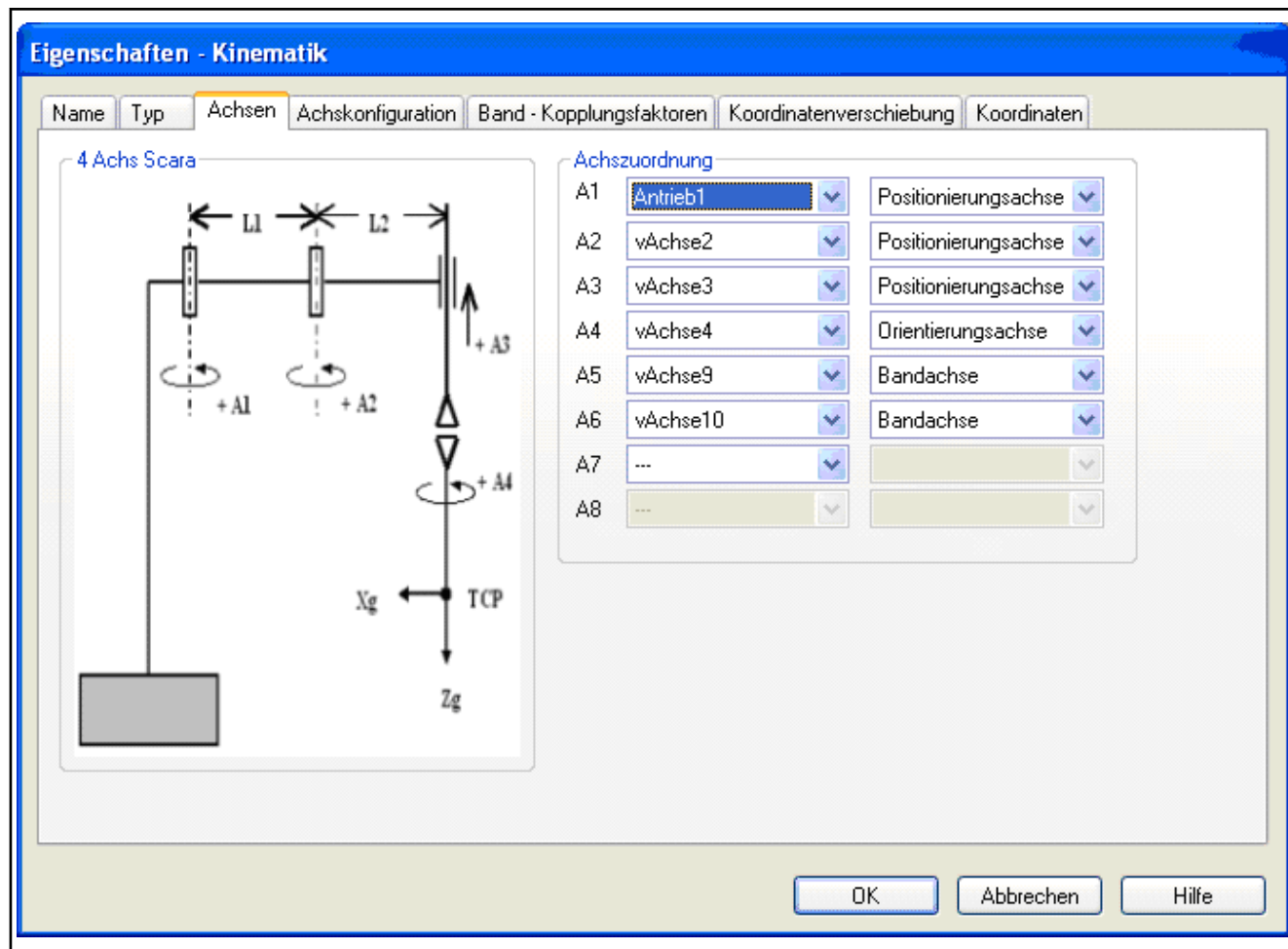


Abb. 16-4: Achszuordnung zu unterschiedlichen Kinematiken (hier 4Achs-Scara)

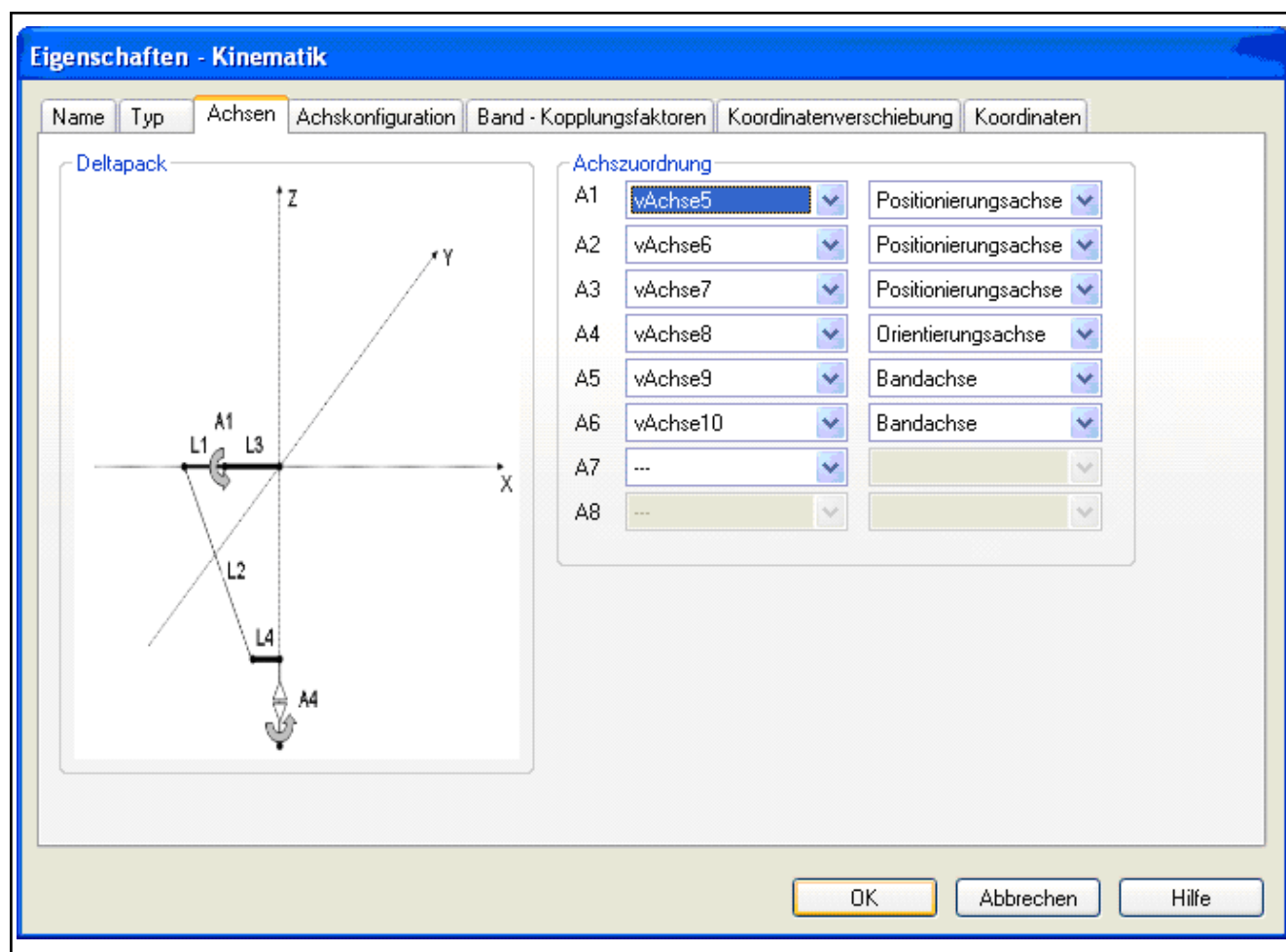


Abb. 16-5: Achszuordnung zu unterschiedlichen Kinematiken (hier Deltapack)

Damit ist sichergestellt, dass keine Achse (außer Bandachsen) mehreren Kinematiken zuordenbar ist.

Beim Umschalten von Parameter-Mode (P2) nach Operation-Mode (BB bzw. P4) wird nochmals die Achsbelegungen in den K-0-0008 Parametern überprüft und ggf. der Fehler F0160040, Fehler im Cross-check K-Parameter, gemeldet. Im Fehlerfall findet man dann im K-0-0011, Liste der ungültigen K-Parameter, den Parameter K-0-0008.

Bandachsen können mehreren Kinematiken gleichzeitig zugeordnet werden. Mit diesem Feature können sich mehrere, auch unterschiedliche Robotertypen, unabhängig an einem oder mehreren Bändern, synchron bewegen. Unabhängig bedeutet, dass jede Kinematik eigene Bandzähler verwaltet.

Dies hat den Vorteil, dass der Bandüberlauf je Kinematik getrennt erfolgen kann und die Kinematiken dazu nicht angehalten werden müssen. Es hat aber auch den Nachteil, dass bei der Erfassung der Produktposition immer alle Bandzähler der Kinematiken gelatched werden müssen – es sei denn, es steht schon zu diesem Zeitpunkt fest, von welchem Roboter das Produkt abgegriffen werden soll.

16.2.4 Löschen von Kinematiken

Eine Kinematik kann auch aus der Sollkonfiguration des IndraWorks-Projekts gelöscht werden.

Anwendung der RoCo-Multikinematik-Technik

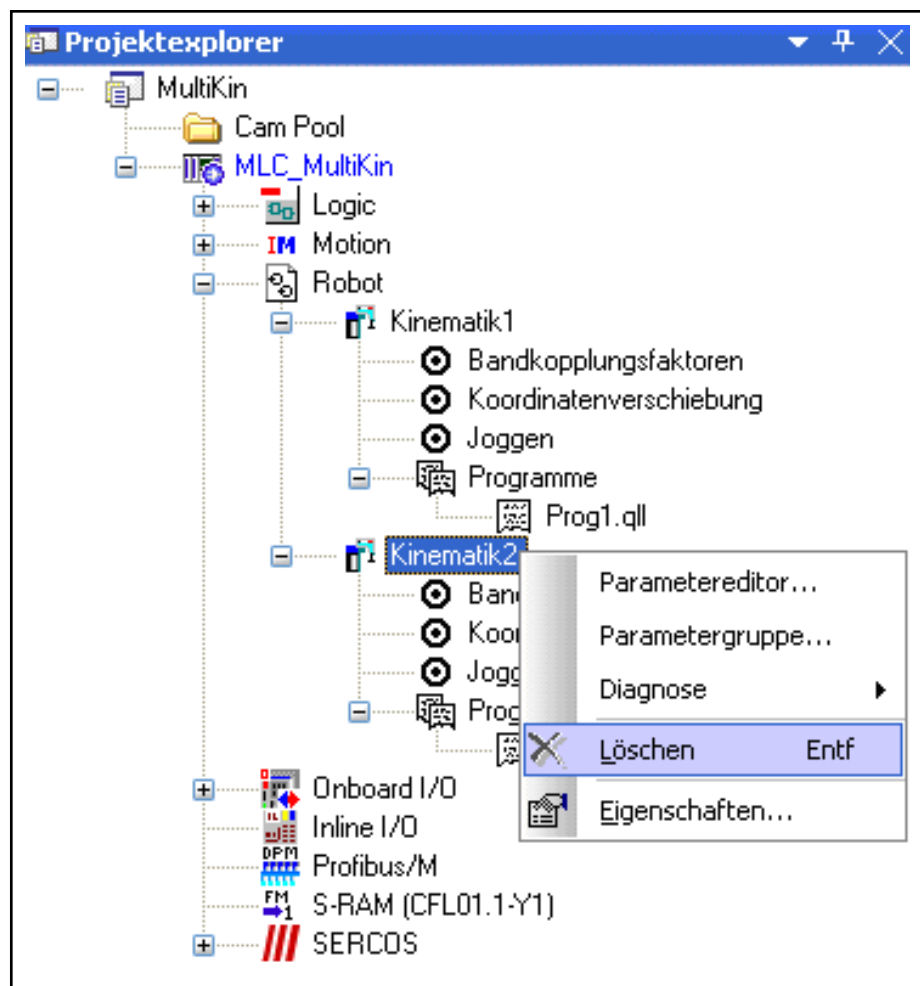


Abb. 16-6: Löschen einer Kinematik



Die Liste im C-0-2484 darf, im Gegensatz zur Achskonfigurationsliste C-0-0484, keine Lücken enthalten.

Dadurch wird erreicht, dass die Adressierung der Kinematiken immer lückenlos von 1..n erfolgt (n= Anzahl der verwendeten Kinematiken). Dies vereinfacht sowohl die Arbeit der Firmware als auch das PLC-Programm erheblich.

Will man eine Kinematik aus der Sollkonfiguration in IndraWorks löschen, müssen die zuvor gesicherten Parameter neu geladen und den verbleibenden Kinematiken entsprechend zugeordnet werden.

16.3 Bewegungssteuerung der Kinematiken

Die Achsen eines Roboters können

- "direkt" vom PLC-Programm über die PLCOpen-Funktionsbausteine z.B. MOVE_LINEAR. **Die PLCOpen-FBs sind derzeit noch in der Entwicklung.**
- durch Abarbeitung eines RCL-Programms (ein RCL-Programm je Kinematik),
- durch Setzen der JOG-Bits im jeweiligen KinData-Interface (manuelle Bewegung).

Anwendung der RoCo-Multikinematik-Technik

bewegt werden.

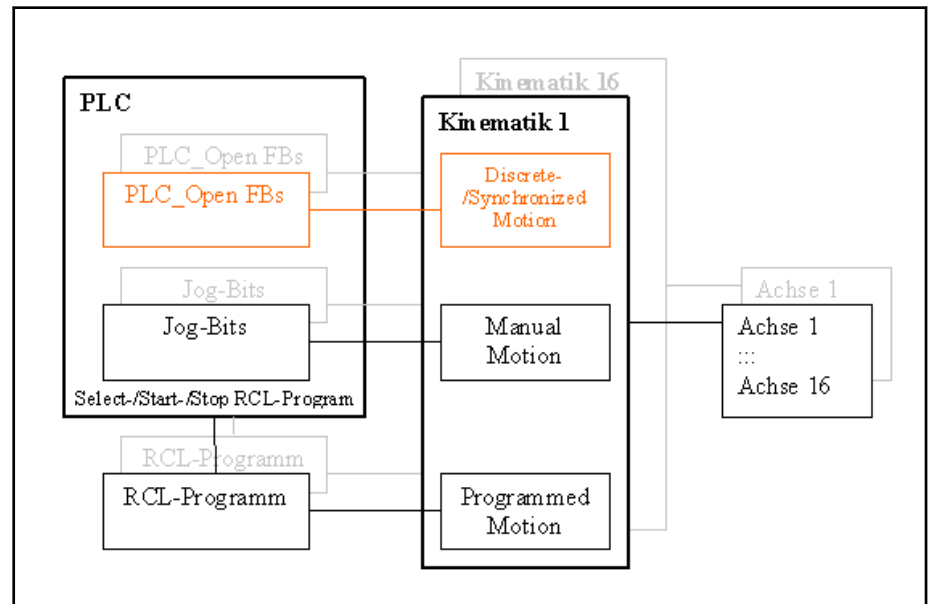


Abb. 16-7: Steuerungsmöglichkeiten für Achsen in Kinematiken

Alle Kinematiken laufen zum größten Teil unabhängig voneinander. D.h. die Kinematik 1 kann in Manual-Motion, die Kinematik 2 im Programmed-Motion und Kinematik 3 kann in Discrete-Motion sein.

Mit anderen Worten: während der eine Roboter ein RCL-Programm abfährt, der andere direkt vom PLC-Programm über die PLCOpen-FBs bewegt wird, kann mit dem dritten Roboter manuell verfahren werden (Jog, Vertippen).

16.4 RCL-Programme der Kinematiken



Ein RCL-Programm kann genau eine Kinematik bewegen. In der RoCo wird folgende Philosophie vertreten:

1. das RCL-Programm ist für die Bewegung des Roboters zuständig,
2. das PLC-Programm erledigt alle anderen Aufgaben (Kommunikation, Koordination, Anlagen-/Prozesssteuerung, ...).

Wird über IndraWorks ein neues RCL-Programm angelegt (**Programme ► Hinzufügen...**), werden automatisch der Programmkopf und der Rumpf des RCL-Programms erstellt.

Anwendung der RoCo-Multikinematik-Technik

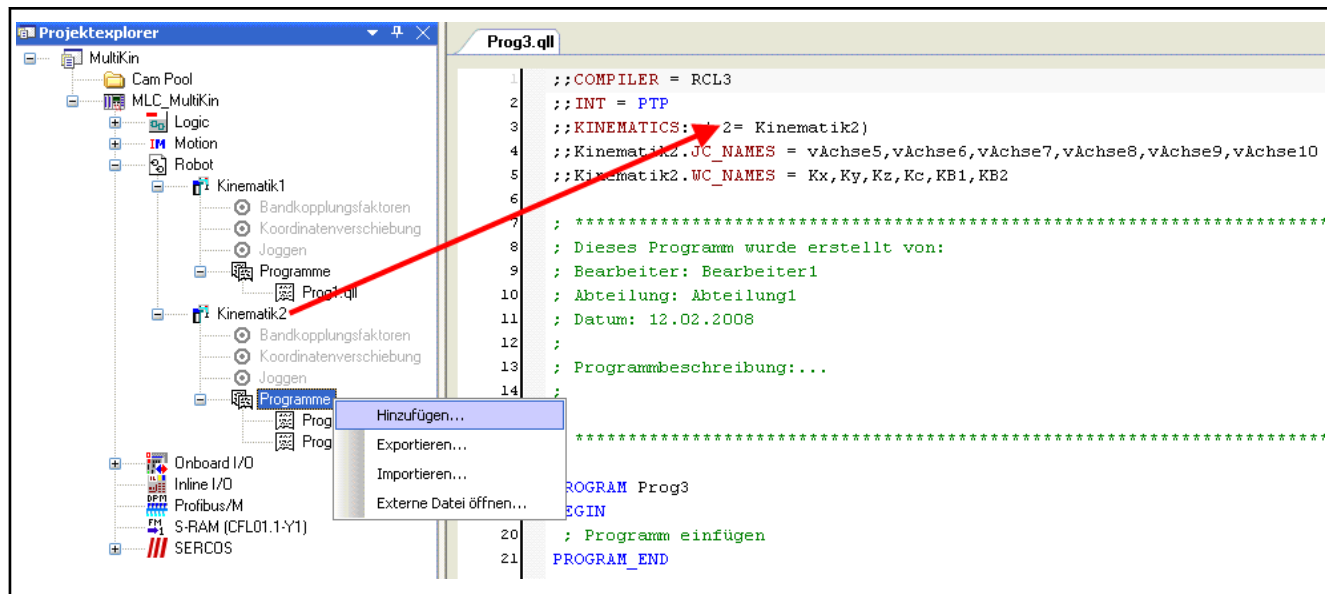


Abb. 16-8: Neues RCL-Programm anlegen

Der Eintrag KINEMATICS in der RCL-Compiler-Anweisung z.B.

```
;;KINEMATICS: ( 2= Kinematik2)
```

legt fest, für welche Kinematik dieses Programm geschrieben ist.

Wird später in der PLC versucht ein RCL-Programm zu starten, das nicht für diese Kinematik geschrieben wurde, wird der Fehler F12C0072 gemeldet, und die Kinematics-State-Machine geht in den Zustand KIS_Error. Bei flüchtigem Hinsehen scheint diese Reaktion korrekt zu sein.

Aber unter anderem verhindert dieses Verhalten die Möglichkeit, ein RCL-Programm auf mehreren Robotern gleichzeitig ablaufen zu lassen. Hat man z.B. mehrere gleiche Roboter in einer Linie, könnten theoretisch alle diese Roboter das gleiche RCL-Programm haben. Dies kann nur dadurch erreicht werden, dass für jede Kinematik ein Programm erstellt und der Quellcode entsprechend kopiert wird. Ansonsten kann nicht sichergestellt werden, dass das RCL-Programm auf der Kinematik auch wirklich lauffähig ist.

Ein RCL-Programm kann im MLC-RoCo-System nur einmalig zu einer Zeit gestartet werden. Dies liegt hauptsächlich an der Verwaltung der Variablen, die bei mehrfachem Start überschrieben werden würden.

16.5 Punktetabellen der Kinematiken

16.5.1 Allgemeines

Zu einem RCL-Programm kann es eine Punktetabelle geben.

Aus Sicht der Satzvorbereitung ist eine Punktetabelle nur ein anderer Speicherort für Punkt-Variablen. Ein anderes RCL-Programm kann nicht mit einer fremden Punktetabelle verknüpft werden. Die Namen der Punktvariablen legt der Anwender fest; sie sind somit fest an ein RCL-Programm gebunden.

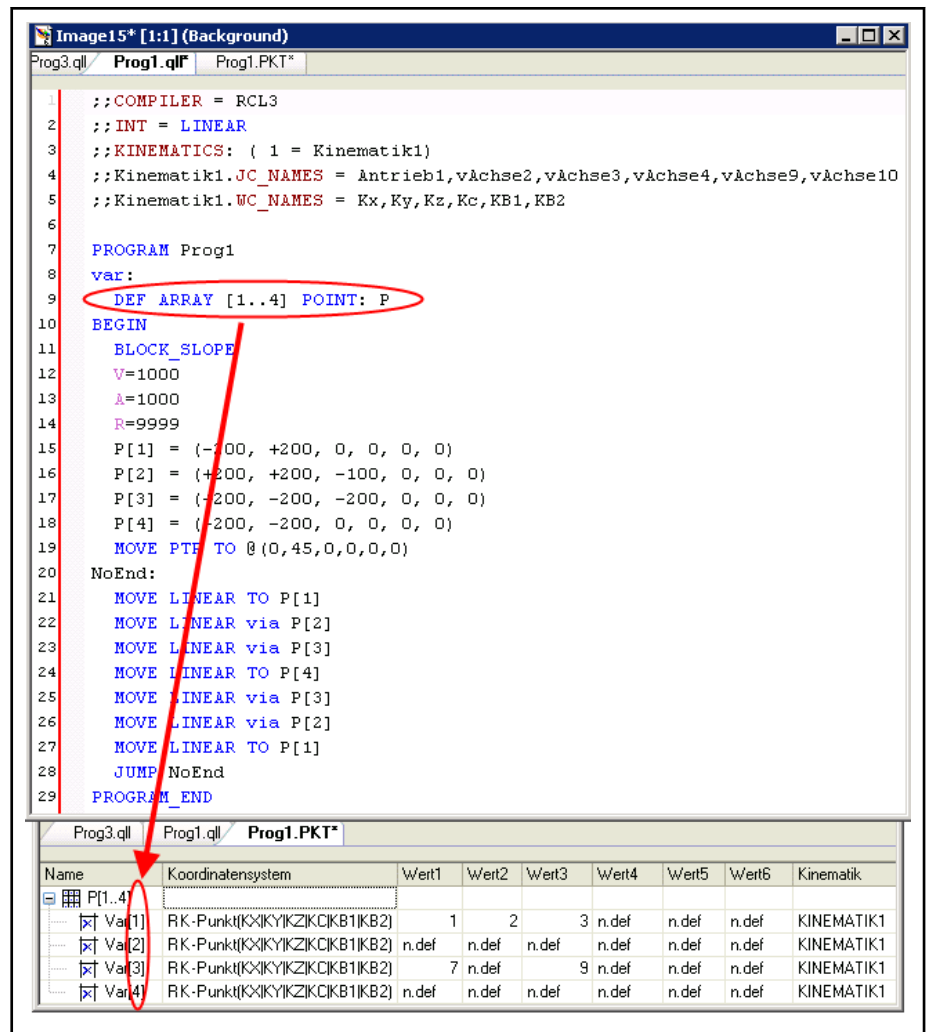


Abb. 16-9: Zuordnung Kinematik und Punktetabelle

Desweiteren kann die Definition eines Punktes in jeder Kinematik unterschiedlich sein (Anzahl der Freiheitsgrade = Achsanzahl). Mit den Compiler-Anweisungen JC_NAMES und WC_NAMES wird beim Anlegen eines RCL-Programms die Freiheitsgrade der Punkte definiert, also abhängig von der Achskonfiguration im Parameter K-0-0008.

16.5.2 Punktetabellen, Zuordnung und Verhalten

Somit ergibt sich folgende strikte Zuordnung

- eine Punktetabelle ist immer genau einem RCL-Programm fest zugeordnet.
- Ein RCL-Programm kann eine (und nicht mehrere!) Punktetabelle haben. Der RCL-Compiler erstellt/ändert die Punktetabelle und stellt sicher, dass das IRD-Programm die Punkt-Variablen aus der Punktetabelle lesen bzw. beschreiben kann.
- Ein RCL-Programm kann zur Laufzeit keine neuen Punkte in der Punktetabelle generieren.
- Ein RCL-Programm ist genau einer Kinematik zugeordnet (siehe Abschnitt [RCL-Programme, Seite 225](#)).
- Eine Kinematik kann mehrere RCL-Programme haben.

Anwendung der RoCo-Multikinematik-Technik

- Der Name des RCL-Programms muss im gesamten MLC-RoCo-System einzigartig sein, d.h. das Programm "Prog1.qll" darf es nicht mehrmals geben – auch nicht in unterschiedlichen Kinematiken.

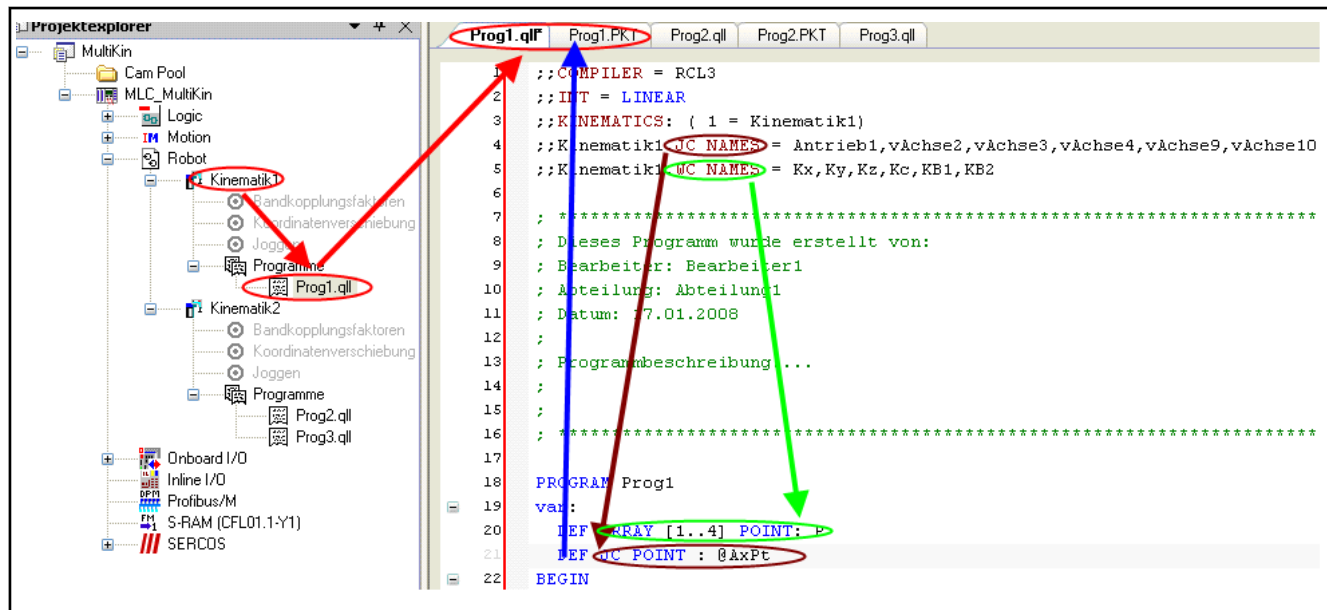


Abb. 16-10: Zuordnung Kinematik, Punktetabelle und Koordinaten

Der Datentyp "POINT" wird durch die Compiler-Anweisung WC_NAMES definiert (=Struktur mit den Elementen Kx, Ky, Kz, ..., wobei die Elemente jeweils vom Datentyp "REAL" sind).

Der Datentyp "JC_POINT" wird durch die Compiler-Anweisung JC_NAMES definiert (=Struktur mit den Elementen Antrieb1, vAchse2, ... wobei die Elemente jeweils vom Datentyp "REAL" sind).

Die Anzahl der Elemente in POINT und JC_POINT ist immer gleich.

Das RCL-Programm "Prog1.qll" hat genau die eine Punktetabelle "prog1.pkt", die denselben Namen wie das Quellprogramm hat. Mit dem Schlüsselwort DEF hat der Programmierer festgelegt, dass diese Punkte in die Punktetabelle und nicht im ird-Programm abgelegt werden sollen. Das Programm Prog1 ist fest der Kinematik1 zugeordnet, und es existiert im gesamten MLC-Robot-System genau einmal.

16.6 PLC-Shared-Memory-Interfaces (KinData) der Kinematiken

Jede Kinematik hat ein eigenes KinData-Interface zur PLC.

IndraWorks legt in der Globalen Variablenliste "GlobaleMLCVariablen", analog zu den Achsen, für jede angelegte Kinematik unter deren Namen eine Variable vom Typ "AXIS_GROUP_REF" an.

Darin enthalten ist die GroupNo der Kinematik. Mit Hilfe dieser GroupNo kann das KinData-ARRAY indiziert werden. Damit wird sichergestellt, dass vom SPS-Programm aus immer die korrekte Kinematik angesprochen wird.

Anwendung der RoCo-Multikinematik-Technik

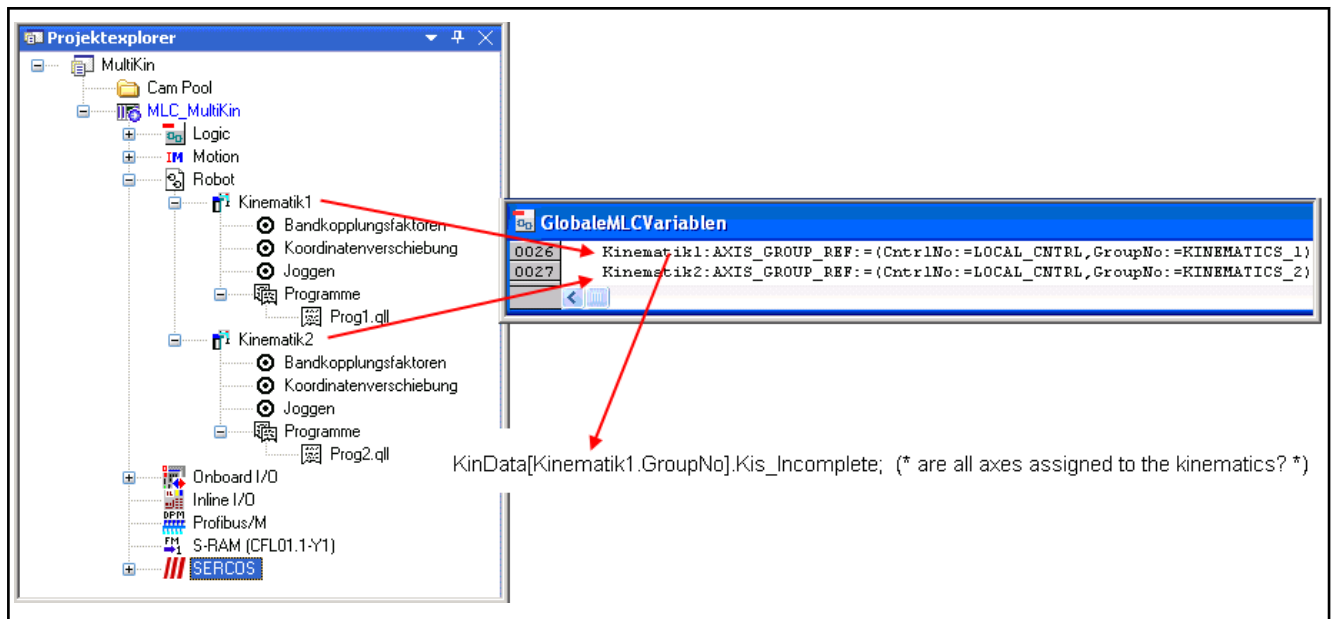


Abb. 16-11: Zugriff auf Daten im Shared Memory

17 Robot Control-Funktionalität, programmiert auf der SPS

Die Programmierung der Robot Control-Funktionalität kann vielfach sowohl in der SPS als auch im RCL-Programm erfolgen.

Es wird jedoch ausdrücklich empfohlen, nicht beide Zugriffsmöglichkeiten gleichzeitig einzusetzen.

SPS-Bibliotheken

- ML_Robot, Funktionsbausteine, Funktionen und Datentypen zur Realisierung von Robot Control-Funktionalität.
- ML_Base, Datentypen für ML_Robot und ML_PLCopen der Systeme IndraMotion XLC, MLC,
- RIL_CommonTypes, Datentypen für alle DCC-Systeme.

Die Bibliotheken sind gemeinsam beschrieben in

Rexroth IndraLogic XLC IndraMotion MLC 12VRS PLCopen-Bibliotheken, DOK-XLCMLC-FUNLIB**V12-LIxx-DE-P, R911334181.

Funktionsmodule

- S-RAM Modul (CFL01.1-Y1), Speichererweiterung für die IndraMotion MLC

Die Funktionsmodule sind gemeinsam beschrieben in

Rexroth IndraControl Lxx 12VRS Funktionsmodule, DOK-CONTRL-FM*LXX**V12-APxx-DE-P, R911333832.

Parameter (XLC, MLC)

- Überblick: Parameter (XLC, MLC)
- Überblick: RCL-Parameter,
- A-Parameter (Achsparemeter),
- C-Parameter (Steuerungsparameter),
- K-Parameter (Kinematik-Parameter).

Die Parameter sind gemeinsam beschrieben in

Rexroth IndraLogic XLC IndraMotion MLC 12VRS Parameter, DOK-XLCMLC-PARAM***V12-RExx-DE-P, R911334185.

Diagnosen (XLC, MLC)

- Überblick: Diagnosen (XLC, MLC)
- F1-Fehler, Achsgruppenfehler

Die Diagnosen sind gemeinsam beschrieben in

Rexroth IndraLogic XLC IndraMotion MLC 12VRS Diagnosen, DOK-XLCMLC-DIAG****V12-RExx-DE-P, R911334179.

...ein kleines Beispiel ist enthalten in:

- Erste Schritte - Robot-Control mit MLC

Rexroth IndraMotion MLC 12VRS Die ersten Schritte, DOK-MLC***-F*STEP**V12-COxx-DE-P, R911333845.

18 Transformationsvorschriften

18.1 Transformationsvorschriften, Überblick

Der vorliegende Abschnitt stellt die Sammlung der für die Programmiersprache RCL (Robot Control Language) verfügbaren Transformationsvorschriften dar.

- Robotertyp 00, Karteser, Seite 233,
- Robotertyp 07, 2-Achs-Delta mit optional 1 Handachse, Seite 237,
- Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse, Seite 240,
- Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse, Seite 242,
- Robotertyp 12, 2-Achs-Parallelogramm, Seite 245,
- Robotertyp 13, 2-Achs-Parallelogramm, Seite 247,
- Robotertyp 18, Umschlingungskinematik, Seite 249,
- Robotertyp 19, 2-Achs-Parallelogramm V1, Seite 250,
- Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse, Seite 252,
- Robotertyp 21, 2-Achs-Parallelogramm V2, Seite 257.
- Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen, Seite 259,
- Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Seite 265,
- Robotertyp 29, 2-Achs-Delta mit rotatorischer und translatorischer Achse, Seite 272,
- Robotertyp 30, Adabot, Seite 275,
- Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP, Seite 278,
- Robotertyp 100, SPS, Seite 279.

18.2 Robotertyp 00, Karteser

Beim Roboter-Typ 0 (Karteser) ist die Transformation abgeschaltet. Die Achskoordinaten (JC) sind mit den Weltkoordinaten (WC) identisch, d.h. MOVE PTP ist in Bezug auf die Bahnbewegung identisch mit MOVE LINEAR. Alle an der Bewegung beteiligten Achsen werden gleichzeitig gestartet und erreichen gleichzeitig ihren Zielpunkt.

Bei MOVE LINEAR und CIRCULAR wird die Pfadgeschwindigkeit für die Positionierachsen im RCL-Programm mit V programmiert, und die Geschwindigkeiten der Orientierungsachsen werden entsprechend so angepasst, dass alle Achsen zum selben Zeitpunkt ihre Zielposition erreichen.

Für MOVE PTP Bewegungen wird die Geschwindigkeit anhand der max. Achsgeschwindigkeiten multipliziert mit dem im RCL programmierten V_PTP ermittelt. Die Achse mit der zeitlich längsten Bewegung bestimmt die Gesamtzeit für die PTP Bewegung. Alle anderen Achsen werden auf diese Zeit angepasst.

Transformationsvorschriften

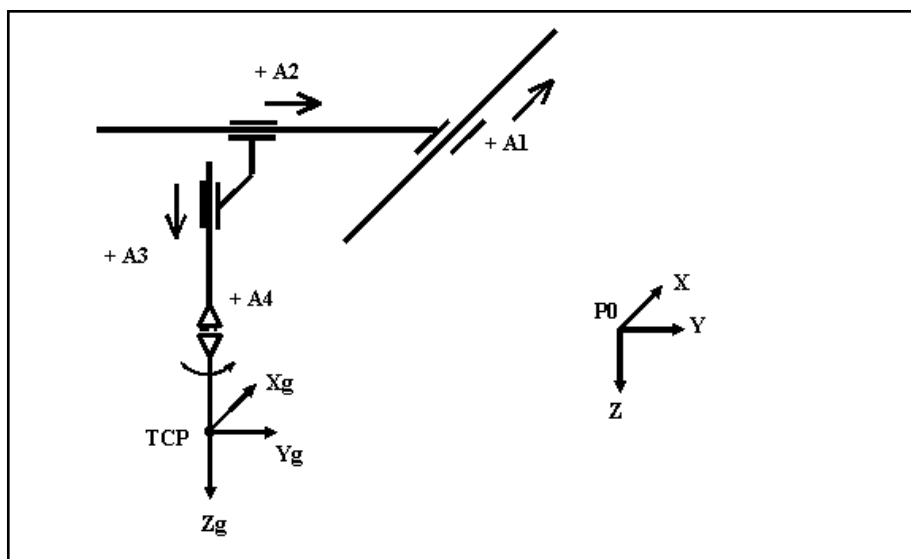


Abb. 18-1: Robotertyp 0, Karteser, 3D-Ansicht

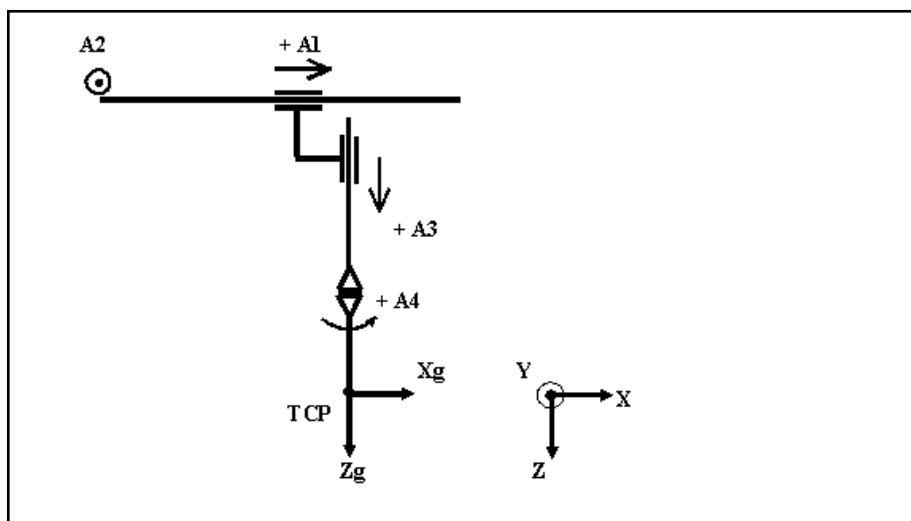


Abb. 18-2: Robotertyp 0, ein typisches X-Y-Z-C kartesisches System

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	translatorisch absolut	
A2:	Positionierungsachse (optional) Orientierungsachse (optional) Bandachse (optional)	translatorisch absolut rotatorisch absolut/modulo translatorisch absolut	optional
A3:	Positionierungsachse (optional) Orientierungsachse (optional) Bandachse (optional)	translatorisch absolut rotatorisch absolut/modulo translatorisch absolut	optional
A4:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A5:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional

Transformationsvorschriften

A6:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A7:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A8:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A9:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A10:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A11:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A12:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A13:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A14:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A15:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional
A16:	Orientierungsachse (optional) Bandachse (optional)	rotatorisch absolut/modulo translatorisch absolut	optional

Abb. 18-3: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 0 (Cartesian Robot)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1
- K-0-0040, Transformationslängen, Achskonfiguration, Achslängen nicht verwendet
- K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung nicht verwendet
- K-0-0121 .. K-0-0136, Koordinatennamen,
Koordinaten: der Roboter hat im kartesischen Koordinatensystem genau so viele Freiheitsgrade wie Achsen im K-0-0008 zugewiesen wurden (inkl. Bandachsen).

Allen Freiheitsgraden müssen Achsen zugeordnet werden.

Beispiel eines 3-achsigen kartesischem Systems mit einer Orientierungsachse und 2 Bandachsen

Name	Koordinate	Bemerkung
K-0-0121	Koordinate 1:	Kx

Transformationsvorschriften

K-0-0122	Koordinate 2:	Ky
K-0-0123	Koordinate 3:	Kz
K-0-0124	Koordinate 4:	Kc
K-0-0125	Koordinate 5:	Kb1
K-0-0126	Koordinate 6:	Kb2

*Abb. 18-4: K-0-0121 .. K-0-0126, Koordinatennamen, Koordinaten***Achs-Drehrichtungen und Nullstellungen**

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Justierung der Achswinkel,
- korrekte Drehrichtung der Achsen.

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Für Roboter-Typ 0 (Kartesiser) stehen folgende Ansichten zur Verfügung:

- [Abb. 18-1 "Robotertyp 0, Karteser , 3D-Ansicht " auf Seite 234](#) und
- [Abb. 18-2 "Robotertyp 0, ein typisches X-Y-Z-C kartesisches System" auf Seite 234.](#)
- Die Nullstellungen aller Achsen ist beim Karteser frei wählbar und kann auch außerhalb der Verfahrbereichsgrenzen liegen.
- Die Verfahrrichtung der bis zu 3 Positionierachsen bilden immer ein rechtshändiges kartesisches Koordinatensystem.
- Die Nullpunkte und die Verfahrrichtungen der Orientierungsachsen sind beliebig.

18.3 Robotertyp 07, 2-Achs-Delta mit optional 1 Handachse

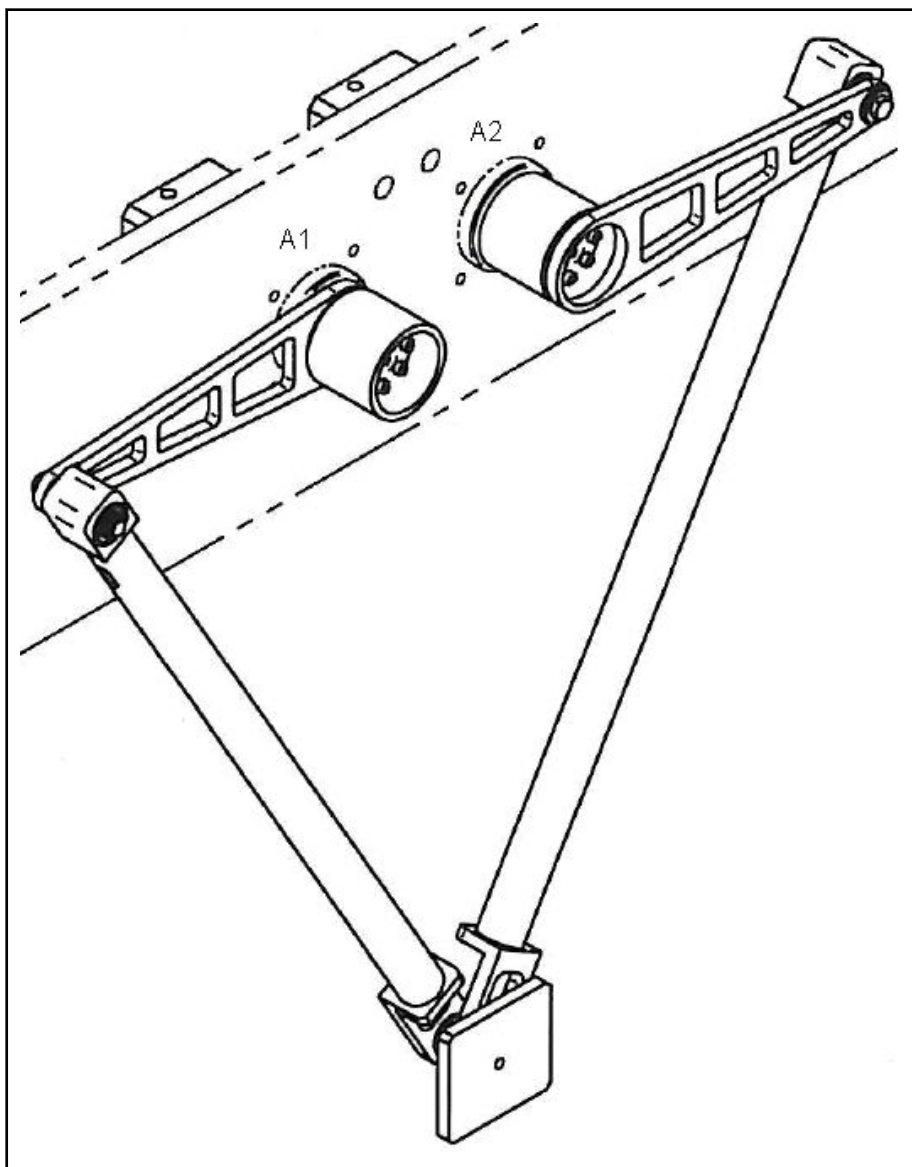


Abb. 18-5: Robotertyp 07, 2-Achs-Delta

Transformationsvorschriften

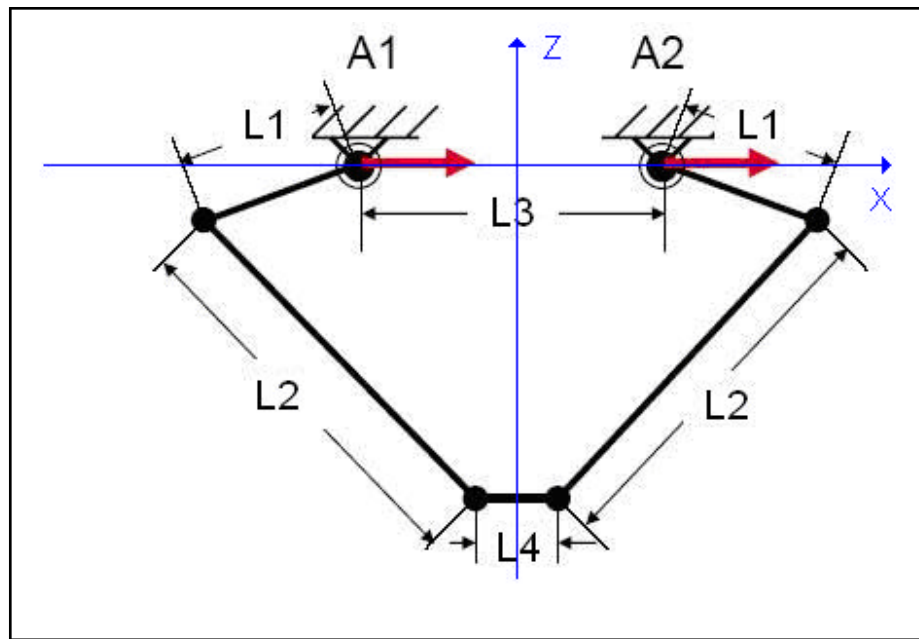


Abb. 18-6: Robotertyp 07, 2-Achs-Delta (X-Z Ansicht)

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+ 90 .. 270 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A3:	Orientierungsachse	rotatorisch abs./modulo	optional
A4:	Bandachse (optional)	translatorisch modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-7: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 7 (Delta 2 Achs, Bipod)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des Arms (Schenkel) am Motor	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des Stabs zur Werkzeugplatte	mm bzw. inch (siehe K-0-0054)	>0
L3:	Abstand der beiden Rotationspunkte der Arme	mm bzw. inch (siehe K-0-0054)	>0
L4:	Abstand der Stäbe an der Grundplatte	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-8: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung, ist nicht verwendet.

Transformationsvorschriften

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Kz)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kc)	optional
Koordinate 4:	Name der 4. Koordinate (z.B. Kbelt1)	optional
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt2)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kbelt3)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kbelt4)	optional

Abb. 18-9: K-0-0121 .. K-0-0136, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

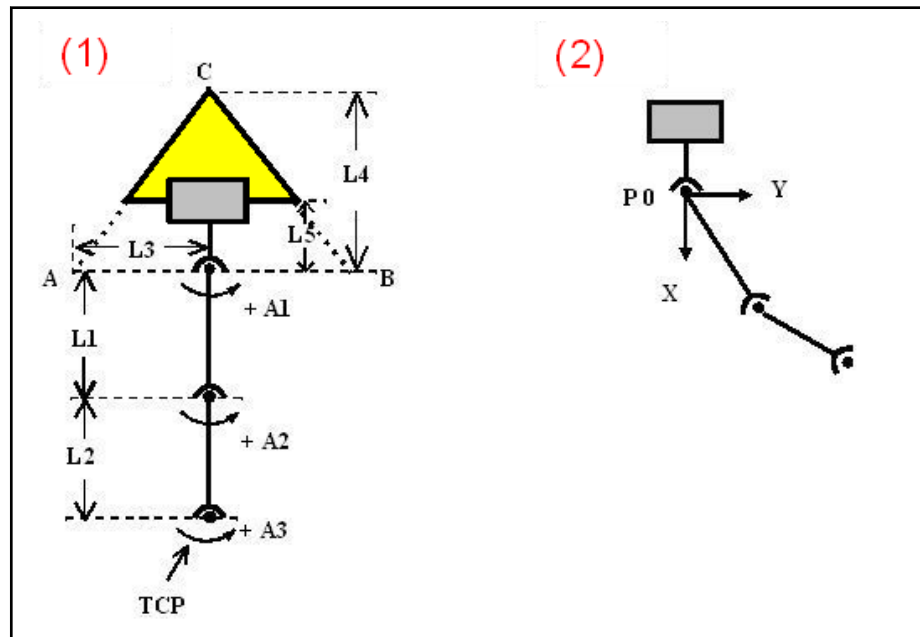
Abb. 18-5 "Robotertyp 07, 2-Achs-Delta" auf Seite 237 zeigt einen Bipod-Roboter in 3D-Darstellung mit den Achspositionen $A1=180^\circ$ und $A2 = 0^\circ$.

In der Nullstellung stehen beide Achsen 1 und 2 (Positionierachsen) waagrecht in Richtung der X-Achse (siehe rote Pfeile in Abb. 18-6 "Robotertyp 07, 2-Achs-Delta (X-Z Ansicht)" auf Seite 238).

- Der Nullpunkt der 3. Achse (Orientierungsachse) ist in positive Richtung der X-Achse
- Die Armlänge L_1 ist gemessen vom Rotationsmittelpunkt bis zum Mittelpunkt der Aufnahme der Stäbe.
- Die Stablänge L_2 ist gemessen von den beidseitigen Aufnahmepunkten.
- L_3 ist der Abstand der Rotationsmittelpunkte der beiden Arme.
- L_4 ist die Länge der Werkzeugplatte gemessen an den Lagerungen der Stäbe.
- Die Drehrichtung der Positionierachsen A1 und A2 ist positiv entgegen dem Uhrzeigersinn.
- Der kartesische Nullpunkt des Bipod-Roboters liegt in der Mitte der beiden Armdrehpunkte A1 und A2.

Transformationsvorschriften

18.4 Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse



(1) Draufsicht mit Nullstellungen aller Achsen, gelb: verbotener Bereich
(2) Nullpunkt-Bezug aller Achsen

Abb. 18-10: Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A3:	Orientierungsachse (optional)	rotatorisch	optional
A4:	Bandachse (optional)	translatorisch modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-11: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 09 (3 Achs Scara)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Abstand vom Nullpunkt bis zum Schnittpunkt der Gerade AC	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)

Transformationsvorschriften

L4:	Abstand vom Nullpunkt bis zur Spitze C des Dreiecks	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)
L5:	Abstand vom Nullpunkt zur Geraden AB	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)

Abb. 18-12: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Trafo-Faktor	Achskonfiguration	Maßeinheit	Achskopplung
K1:	mechanische Kopplung zwischen Achse 1, 2 und 3	Faktor	optional

Abb. 18-13: K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kc)	optional
Koordinate 4:	Name der 4. Koordinate (z.B. Kb1)	optional
Koordinate 5:	Name der 5. Koordinate (z.B. Kb2)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kb3)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kb4)	optional

Abb. 18-14: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- exakte Ermittlung der Kopplungsfaktoren (falls vorhanden)
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden. Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-10 "Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse" auf Seite 240: (1) zeigt in einer Draufsicht die Nullstellungen aller Achsen.

- Nimmt der Roboter die in den Abbildungen dargestellte Stellung ein, so müssen sämtliche Achsen in Maschinenkoordinaten Null anzeigen.

Dies ist durch eine entsprechende Einstellung der Referenzpunkt-Istwerte zu gewährleisten.

- Die Armlängen L_1 , L_2 und die positiven Drehrichtungen der Achsen sind ebenfalls in den Abbildungen dargestellt.

- Mit Hilfe der Längen L_3 , L_4 , L_5 lässt sich ein verbotener Bereich definieren, in den der Tool-Center-Point (TCP) nicht eindringen kann.

Dies dient der Kollisionsvermeidung, wenn die Verfahrbereiche der Achsen 1, 2 so groß sind, dass der Roboter gegen sein eigenes Stativ fahren könnte.

Dieser verbotene Bereich ist in (1) gelb dargestellt.

- Ist die Länge $L_3=0$, so ist keine Bereichsüberwachung aktiv.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv einzugeben, wobei L_1 , L_2 ungleich Null sein müssen.

Transformationsvorschriften

- Die Transformation kann eine mechanische Kopplung der Achsen 1, 2 mit der 3. Achse berücksichtigen.

Eine solche Kopplung liegt vor, wenn sich beim Verfahren der Achsen 1, 2 die 3. Achse mit bewegt, ohne dass das Meßsystem der 3. Achse diese Bewegung erfasst. Dies ist in der Regel dann der Fall, wenn der Motor der 3. Achse nicht direkt auf der Achse, sondern auf dem Stativ sitzt.

Diese Kopplung ist über den Parameter K-0-0041 einzustellen (siehe oben).

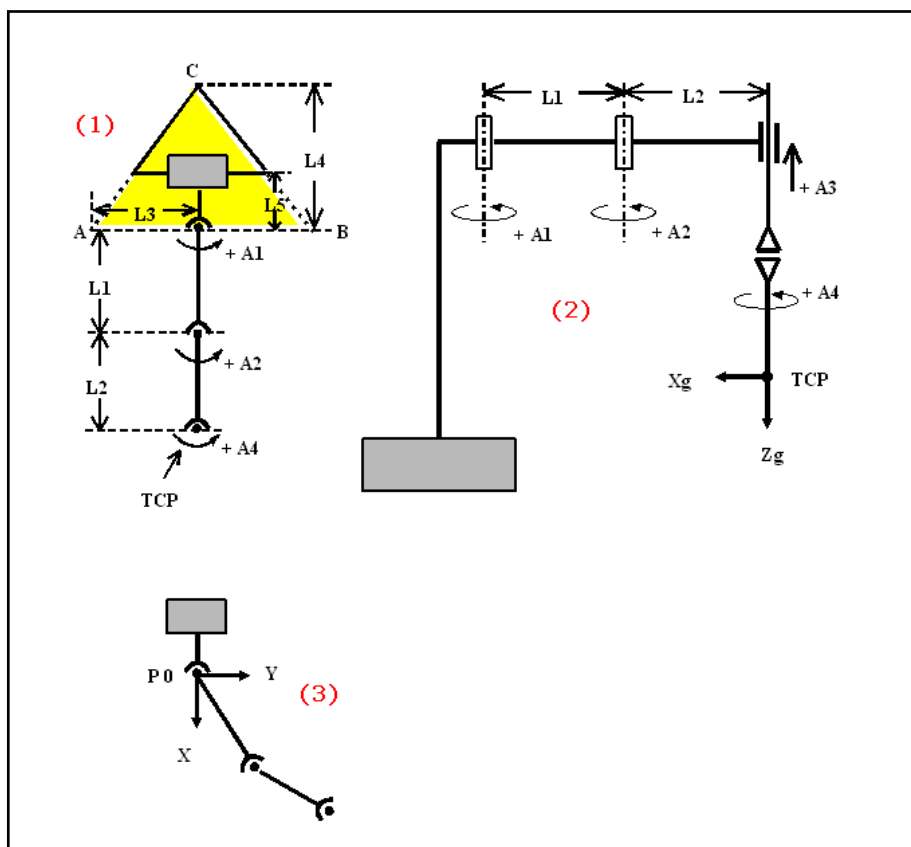


Der Verfahrbereich der Achsen 1 und 2 darf maximal ± 179 Grad betragen. Der Verfahrbereich der 3. Achse ist unbeschränkt.

Das zugehörige Raumkoordinaten-System ist in [Abb. 18-10 "Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse"](#) auf Seite 240 (2) durch die 2 Hauptachsen (x, y) gekennzeichnet.

Der Raumkoordinaten-Nullpunkt liegt im Fußpunkt P_0 des Roboters.

18.5 Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse



(1) Draufsicht mit Nullstellungen aller Achsen, gelb: verbotener Bereich
(2) Seitenansicht mit Nullstellungen aller Achsen
Abb. 18-15: Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	± 179 Grad

Transformationsvorschriften

A2:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A3:	Positionierungsachse (optional)	translatorisch	optional
A4:	Orientierungsachse (optional)	rotatorisch	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-16: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 10 (4 Achs Scara)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Abstand vom Nullpunkt bis zum Schnittpunkt der Geraden AC	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)
L4:	Abstand vom Nullpunkt bis zur Spitze C des Dreiecks	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)
L5:	Abstand vom Nullpunkt zur Geraden AB	mm bzw. inch (siehe K-0-0054)	optional, siehe (1)

Abb. 18-17: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Trafo-Faktor	Achskonfiguration	Maßeinheit	Achskopplung
K1:	mechanische Kopplung zwischen Achse 1, 2 und 4	Faktor	optional
K2:	mechanische Kopplung zwischen Motor 4 und 3 (Hub-/Drehspindel)	mm/Grad	optional

Abb. 18-18: K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kz)	optional
Koordinate 4:	Name der 4. Koordinate (z.B. Kc)	optional
Koordinate 5:	Name der 5. Koordinate (z.B. Kb1)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kb2)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kb3)	optional
Koordinate 8:	Name der 8. Koordinate (z.B. Kb4)	optional

Abb. 18-19: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinatentransformation sind:

- exakte Einstellung der Achs-Nullpunkte

Transformationsvorschriften

- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- exakte Ermittlung der Kopplungsfaktoren (falls vorhanden)
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-15 "Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse" auf Seite 242: (1) zeigt in einer Draufsicht und (2) in einer Seitenansicht die Nullstellungen aller Achsen.

- Nimmt der Roboter die in den Abbildungen dargestellte Stellung ein, so müssen sämtliche Achsen in Maschinenkoordinaten Null anzeigen.
Dies ist durch eine entsprechende Einstellung der Referenzpunkt-Istwerte zu gewährleisten.

- In der Nullstellung stehen die Achsen 1, 2, 4 ausgestreckt nach vorne.
- Der Nullpunkt der 3. Achse ist frei wählbar.
- Die Armlängen L_1 , L_2 und die positiven Drehrichtungen der Achsen sind ebenfalls dargestellt.
- Mit Hilfe der Längen L_3 , L_4 , L_5 läßt sich ein verbotener Bereich definieren, in den der Tool-Center-Point (TCP) nicht eindringen kann.
- Dies dient der Kollisionsvermeidung, wenn die Verfahrbereiche der Achsen 1, 2 so groß sind, dass der Roboter gegen sein eigenes Stativ fahren könnte.

Dieser verbotene Bereich ist in (1) gelb dargestellt.

- Ist die Länge $L_3=0$, so ist keine Bereichsüberwachung aktiv.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv einzugeben, wobei L_1 , L_2 ungleich Null sein müssen.
- Die Transformation kann eine mechanische Kopplung der Achsen 1, 2 mit der 4. Achse berücksichtigen.

Eine solche Kopplung liegt vor, wenn sich beim Verfahren der Achsen 1, 2 die 4. Achse mitbewegt, ohne daß das Meßsystem der 4. Achse diese Bewegung erfaßt. Dies ist in der Regel dann der Fall, wenn der Motor der 4. Achse nicht direkt auf der Achse, sondern auf dem Stativ sitzt.

Diese Kopplung ist über den Parameter K-0-0041 einzustellen (siehe oben).

- Desweiteren kann die Transformation eine mechanische Kopplung der 4. Achse mit der 3. Achse berücksichtigen.

Eine solche Kopplung liegt vor, wenn sich beim Verfahren der Achse 4 die 3. Raumkoordinate (Z-Achse) verändert, ohne daß das Meßsystem der 3. Achse diese Bewegung erfaßt.

Dies ist beispielsweise dann der Fall, wenn es sich bei den Achsen 3 und 4 um eine Hub-, Drehspindel handelt.

Der Kopplungsfaktor ist folgendermaßen definiert:

- $K_{43} = \Delta z / \Delta A_4$

Dabei bezeichnet ΔA_4 den vom Meßsystem erfaßten Achswinkel und Δz die Änderung der Raumkoordinate.

Transformationsvorschriften



Der Verfahrbereich der Achsen 1 und 2 darf maximal ± 179 Grad betragen.

Der Verfahrbereich der Achsen 3 und 4 ist unbeschränkt.

Das zugehörige Raumkoordinaten-System ist durch die 3 Hauptachsen (x, y, z) gekennzeichnet.

Der Raumkoordinaten-Nullpunkt liegt im Fußpunkt P_0 des Roboters.

Die Transformation kann auch für einen 2- bzw. 3- Achs-Scara verwendet werden.

18.6 Robotertyp 12, 2-Achs-Parallelogramm

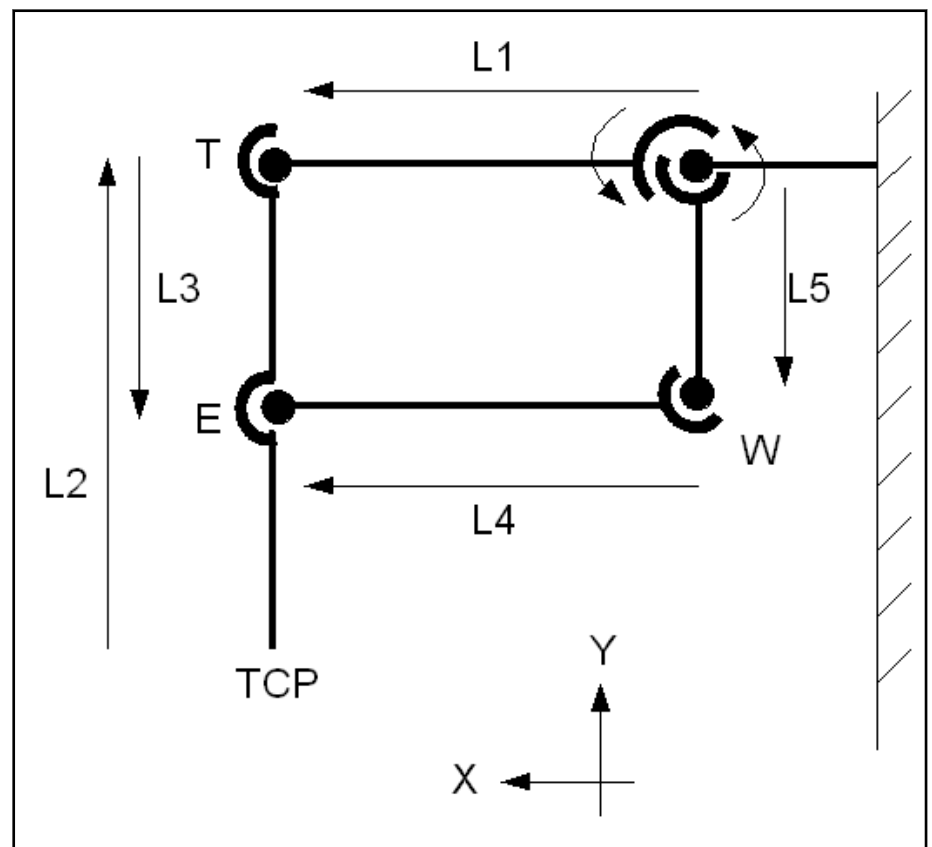


Abb. 18-20: Robotertyp 12, 2-Achs-Parallelogramm

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A3:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-21: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 12 (2-Achs-Parallelogramm)
- K-0-0032, Referenzieren aller Achsen = 0

Transformationsvorschriften

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des 3. Arms	mm bzw. inch (siehe K-0-0054)	>0
L4:	Länge des 4. Arms	mm bzw. inch (siehe K-0-0054)	>0
L5:	Länge des 5. Arms	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-22: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	

Abb. 18-23: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Bestimmung der Vorzeichen der einzelnen Längen
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-20 "Robotertyp 12, 2-Achs-Parallelogramm" auf Seite 245 zeigt

- Die schematische Darstellung der Kinematik, bei Nullstellung aller Achsen.
- Die Armlängen L_1 , L_2 , L_3 , L_4 , L_5 und die positiven Drehrichtungen der Achsen.
- Der Verfahrbereich der Achsen ist mechanisch so zu begrenzen, dass der Winkel zwischen den beiden Armen mit der Längenbezeichnung L_1 und L_5 nie gleich 180° beträgt. Ansonsten gibt es je nach Mechanik, 2 unterschiedliche mathematische Lösungen, was zu ungewünschten Bewegungen führen kann.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv einzugeben und müssen immer ungleich Null sein.
- Das zugehörige Raumkoordinaten-System ist durch die 2 Hauptachsen (x, y) gekennzeichnet.
- Der Raumkoordinaten-Nullpunkt liegt im Schittpunkt der beiden Längenangaben L_1 und L_5 .

18.7 Robotertyp 13, 2-Achs-Parallelogramm

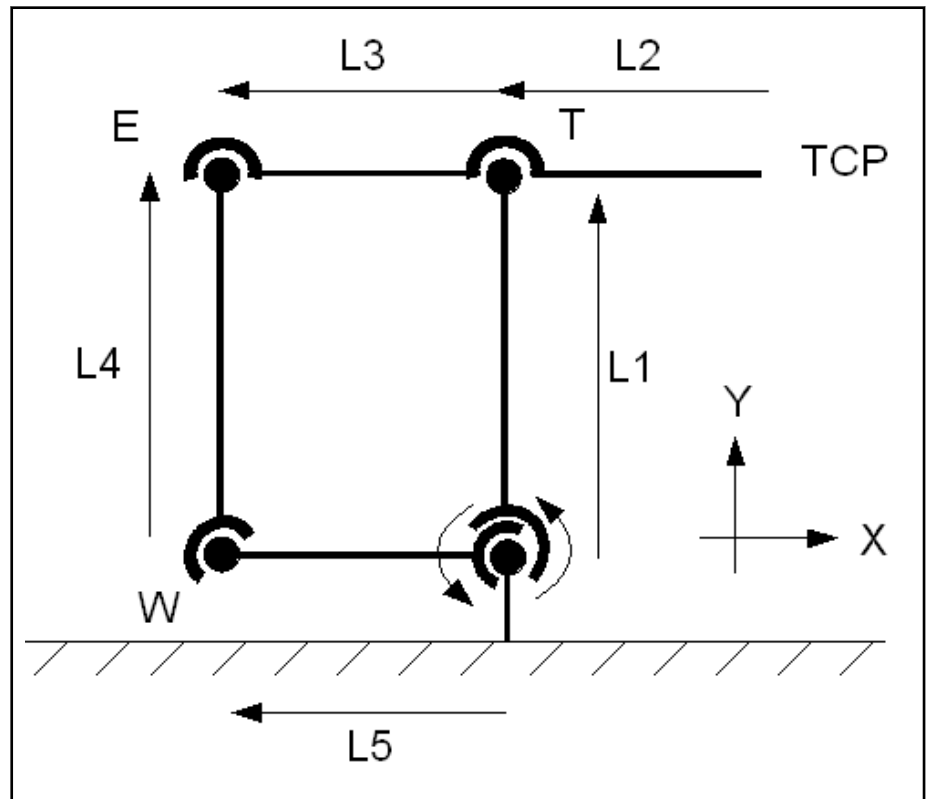


Abb. 18-24: Robotertyp 13, 2-Achs-Parallelogramm

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A3:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-25: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 13 (2-Achs-Parallelogramm)
- K-0-0032, Referenzieren aller Achsen = 0

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des 3. Arms	mm bzw. inch (siehe K-0-0054)	>0
L4:	Länge des 4. Arms	mm bzw. inch (siehe K-0-0054)	>0
L5:	Länge des 5. Arms	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-26: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Transformationsvorschriften

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	

Achs-Drehrichtungen und Nullstellungen

Abb. 18-27: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Bestimmung der Vorzeichen der einzelnen Längen
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

[Abb. 18-24 "Robotertyp 13, 2-Achs-Parallelogramm" auf Seite 247](#) zeigt

- Die schematische Darstellung der Kinematik, bei Nullstellung aller Achsen.
- Die Armlängen L_1 , L_2 , L_3 , L_4 , L_5 und die positiven Drehrichtungen der Achsen.
- Der Verfahrbereich der Achsen ist mechanisch so zu begrenzen, dass der Winkel zwischen den beiden Armen mit der Längenbezeichnung L_1 und L_5 nie gleich 180° beträgt. Ansonsten gibt es je nach Mechanik, 2 unterschiedliche mathematische Lösungen, was zu ungewünschten Bewegungen führen kann.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv und ungleich Null einzugeben.
- Das zugehörige Raumkoordinaten-System ist durch die 2 Hauptachsen (x, y) gekennzeichnet.
- Der Raumkoordinaten-Nullpunkt liegt im Schittpunkt der beiden Längenangaben L_1 und L_5 .

18.8 Robotertyp 18, Umschlingungskinematik

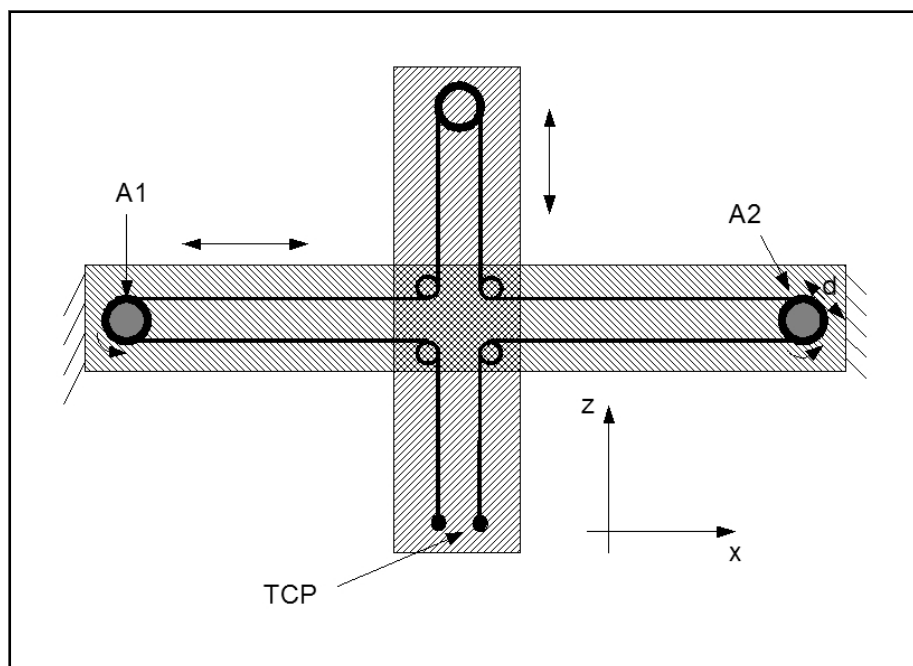


Abb. 18-28: Robotertyp 18, Umschlingungskinematik

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	Getriebe: lastseitig ist der Radius ($d/2$) einzugeben
A2:	Positionierungsachse	rotatorisch absolut	Getriebe: lastseitig ist der Radius ($d/2$) einzugeben

Abb. 18-29: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 18 (Umschlingungskinematik).
- K-0-0032, Referenzieren aller Achsen = 0.
- K-0-0040, Transformationslängen, Achskonfiguration, Achslängen werden nicht gesetzt.

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Kz)	

Abb. 18-30: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung des Durchmessers d der antreibenden Achsen
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Transformationsvorschriften

Abb. 18-28 "Robotertyp 18, Umschlingungskinematik" auf Seite 249 zeigt die schematische Darstellung der Kinematik, bei Nullstellung aller Achsen. Für die Anwendung bedeutet dies, dass die Achsen in ihre gewünschte Nullstellung zu fahren sind, dann muss referenziert werden.

- Die positiven Drehrichtungen der Achsen sind dargestellt.
- Die Verfahrbereichsgrenzen der Achsen sind so einzustellen, dass die mechanischen Verfahrbereiche nicht überschritten werden.
- Das zugehörige Raumkoordinaten-System ist durch die 2 Hauptachsen (x, z) gekennzeichnet.
- Das Getriebe der beiden Achsen ist so einzustellen, dass lastseitig der Radius ($d/2$) in mm eingetragen wird ($S-0-0121 = 1$, $S-0-0122 = d/2$).

18.9 Robotertyp 19, 2-Achs-Parallelogramm V1

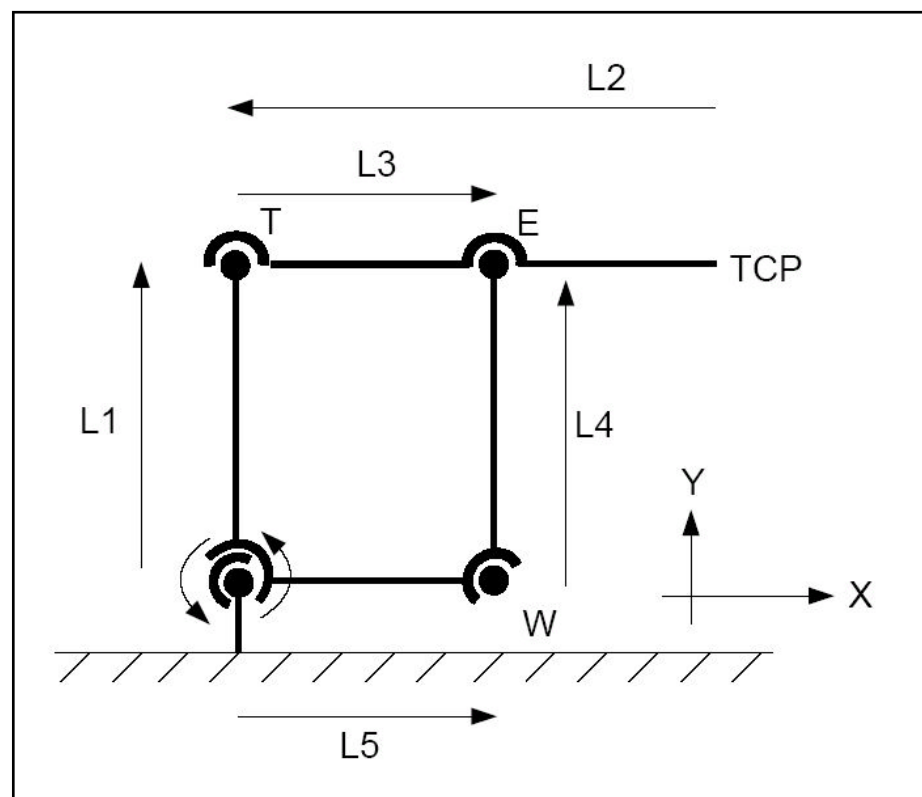


Abb. 18-31: Robotertyp 19, 2-Achs-Parallelogramm V1

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A3:	Positionierungsachse	translatorisch absolut	optional
A4:	Orientierungsachse	rotatorisch mod. / abs.	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional

Transformationsvorschriften

A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-32: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 19 (2-Achs-Parallelogramm)
- K-0-0032, Referenzieren aller Achsen = 0

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des 3. Arms	mm bzw. inch (siehe K-0-0054)	>0
L4:	Länge des 4. Arms	mm bzw. inch (siehe K-0-0054)	>0
L5:	Länge des 5. Arms	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-33: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kz)	
Koordinate 4:	Name der 4. Koordinate (z.B. Kb)	
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt)	

Abb. 18-34: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Bestimmung der Vorzeichen der einzelnen Längen
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-31 "Robotertyp 19, 2-Achs-Parallelogramm V1" auf Seite 250 zeigt

- Die schematische Darstellung der Kinematik, bei Nullstellung aller Achsen.
- Die Armlängen L_1 , L_2 , L_3 , L_4 , L_5 und die positiven Drehrichtungen der Achsen.
- Der Verfahrbereich der Achsen ist mechanisch so zu begrenzen, dass der Winkel zwischen den beiden Armen mit der Längenbezeichnung L_1 und L_5 nie gleich 180° beträgt. Ansonsten gibt es je nach Mechanik, 2 unterschiedliche mathematische Lösungen, was zu ungewünschten Bewegungen führen kann.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv und ungleich Null einzugeben.

Transformationsvorschriften

- Das zugehörige Raumkoordinaten-System ist durch die 2 Hauptachsen (x, y) gekennzeichnet.
- Der Raumkoordinaten-Nullpunkt liegt im Schnittpunkt der beiden Längenangaben L_1 und L_5 .

18.10 Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse

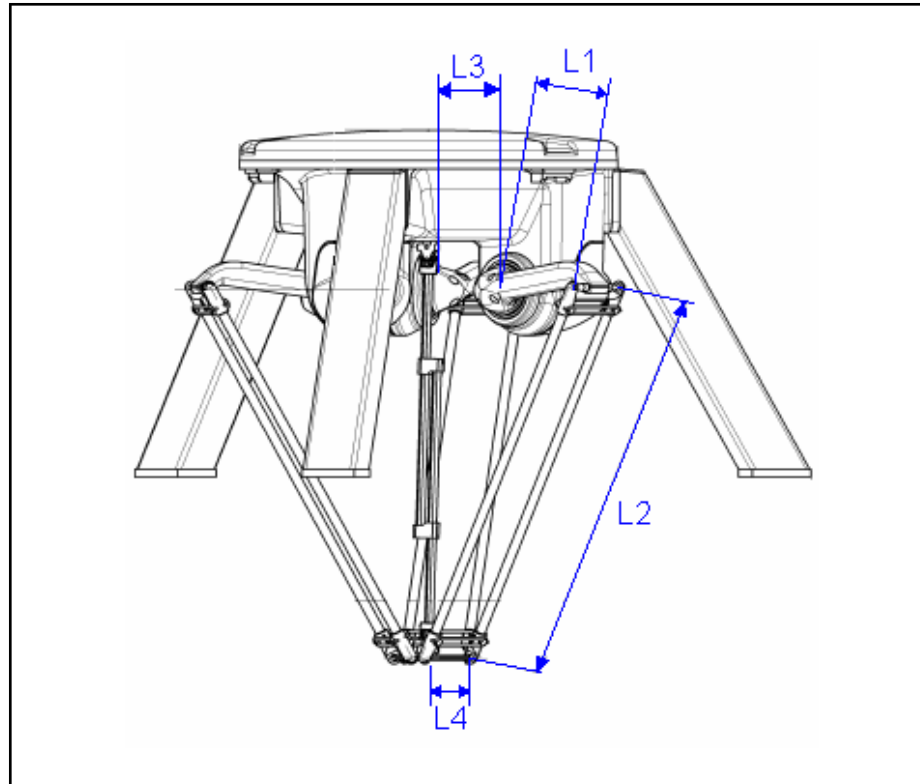
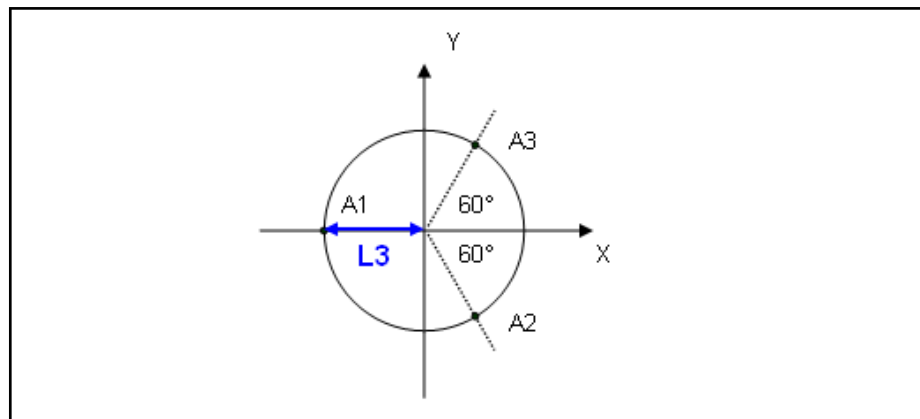


Abb. 18-35: Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse

Abb. 18-36: Radius $L3$ der Motorhalterung und Koordinatensystem (X-Y Ansicht)
Die drei Achsen A1 bis A3 sind um jeweils 120° versetzt.

Transformationsvorschriften

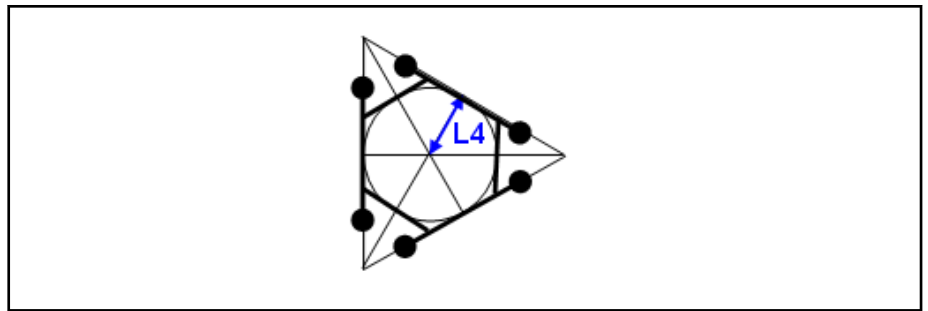
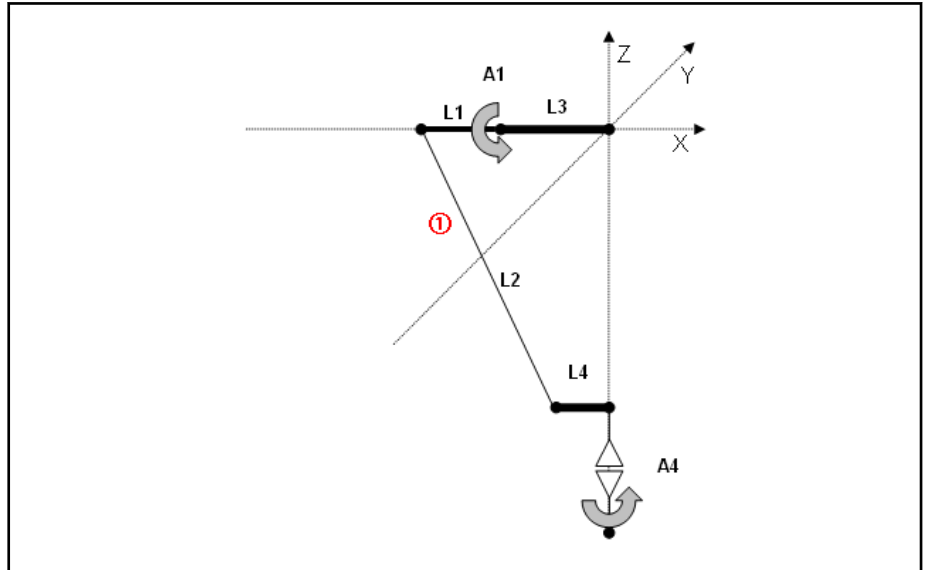


Abb. 18-37: Radius L4 der Werkzeugplatte



① Verbindungsstange

Abb. 18-38: Darstellung der Nullstellung der ersten Achse

Nullstellung der Orientierungsachse A4 sollte die positive X-Richtung sein.

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A3:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A4:	Orientierungsachse (optional)	rotatorisch abs./modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-39: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 20 (Delta 3 Achs)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1

Transformationsvorschriften

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des Arms (Schenkel) am Motor	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des Stabs zur Werkzeugplatte	mm bzw. inch (siehe K-0-0054)	>0
L3:	Radius des durch die 3 Rotationspunkte der Arme definierten Kreises	mm bzw. inch (siehe K-0-0054)	>0
L4:	Radius der Grundplatte	mm bzw. inch (siehe K-0-0054)	>0
L5:	muss 0 sein		=0
L6:	muss 0 sein		=0

Abb. 18-40: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen



wenn L5 und L6 $\neq$ 0 sind werden translatorisch gewichtete Achsen verwendet. Siehe [Translatorisch gewichtete Positionierachsen](#), Seite 255.

K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung, ist nicht verwendet.

Name	Koordinate	Bemerkung
K-0-0121	Koordinate 1:	Kx
K-0-0122	Koordinate 2:	Ky
K-0-0123	Koordinate 3:	Kz
K-0-0124	Koordinate 4:	Kc
K-0-0125	Koordinate 5:	Kb1
K-0-0126	Koordinate 6:	Kb2
K-0-0127	Koordinate 7:	Kb3
K-0-0128	Koordinate 8:	Kb4

Abb. 18-41: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

[Abb. 18-35 "Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse" auf Seite 252](#) zeigt einen Delta-Roboter in Nullstellung aller Achsen, d.h. nimmt der Roboter die in den Abbildungen dargestellte Stellung ein, so müssen sämtliche Achsen in Maschinenkoordinaten Null anzeigen. Dies ist durch eine entsprechende Einstellung der Referenzpunkt-Istwerte zu gewährleisten.

In der Nullstellung stehen die Achsen 1, 2 und 3 (Positionierachsen) waagrecht.

Transformationsvorschriften

Der Nullpunkt der 4. Achse (Orientierungsachse) ist in positive Richtung der X-Achse.

Die Armlänge L_1 ist gemessen vom Rotationsmittelpunkt bis zum Mittelpunkt der Aufnahme der Stäbe.

Die Stablänge L_2 ist gemessen von den beidseitigen Aufnahmepunkten.

L_3 ist der Radius des durch die 3 Motorspindeln (bzw. Getriebe) gebildeten Umkreises, gemessen an den Rotationsmittelpunkten der Arme (siehe Abb. 18-36 "Radius L_3 der Motorhalterung und Koordinatensystem (X-Y Ansicht)" auf Seite 252).

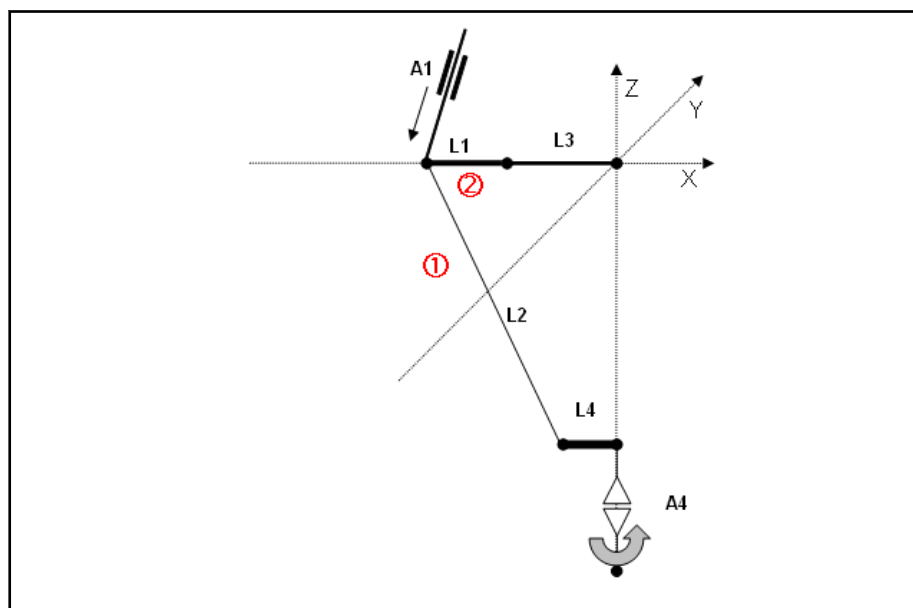
L_4 ist der Radius des Innenkreises des gleichseitigen Dreiecks auf der Werkzeugplatte. Das gleichseitige Dreieck wird durch die Aufnahmepunkte der Stäbe definiert. (siehe Abb. 18-37 "Radius L_4 der Werkzeugplatte" auf Seite 253).

Die Drehrichtung der Positionierachsen A1 bis A3 ist positiv, wenn die Arme in Z-Richtung nach unten bewegt werden. Die Orientierungsachse A4 (Drehung um die Z-Achse) dreht positiv entgegen dem Uhrzeigersinn.

Der kartesische Nullpunkt des Delta-Roboters liegt im Mittelpunkt des Umkreises der durch die 3 Motorspindeln definiert ist (siehe auch Abb. 18-36 "Radius L_3 der Motorhalterung und Koordinatensystem (X-Y Ansicht)" auf Seite 252).

Translatorisch gewichtete Positionierachsen

Die Transformation kann auch mit translatorisch gewichteten Positionierachsen betrieben werden.



- ① Verbindungsstange
- ② Arm

Abb. 18-42: Darstellung der Nullstellung bei translatorisch gewichteten Positionierachsen

Bei waagerechter Stellung der Arme müssen die Positionen der Achsen A1 bis A3 Null sein. Die positive Bewegung der Positionierachsen drückt den entsprechenden Arm nach unten.

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	translatorisch absolut	

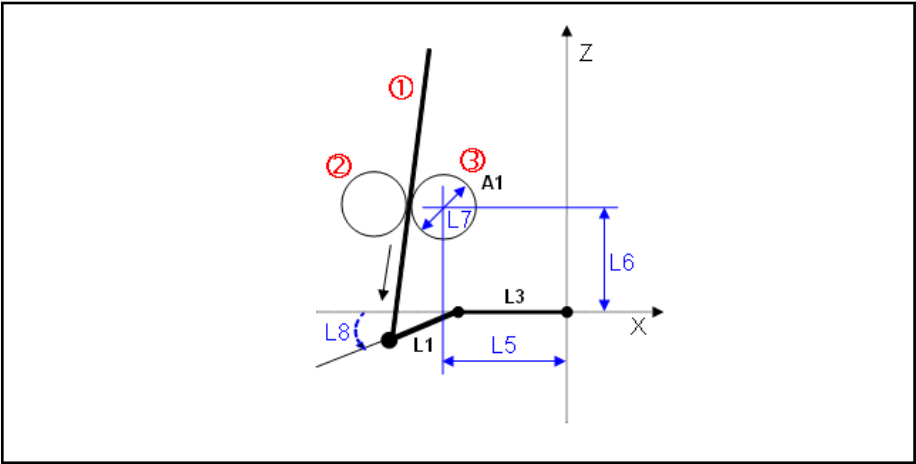
Transformationsvorschriften

A2:	Positionierungsachse	translatorisch absolut	
A3:	Positionierungsachse	translatorisch absolut	

Abb. 18-43: K-0-0008 Achskonfigurationsliste, Achszuordnung

Die Positionierachsen A1 bis A3 können auch translatorisch sein.

- K-0-0040 Transformationslängen, Achskonfiguration,
Achslängen zu den bereits oben beschriebenen K-0-0040 Einträgen L1 bis L4 müssen noch L5 bis L8 gesetzt werden:



- ① Zahnstange
- ② Andruckrolle
- ③ Antriebsrolle

Abb. 18-44: Weitere Einstellungen bei translatorisch gewichteten Positionierachse

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L5:	Horizontale Distanz von der Zahnstangen-Antriebsrolle bis zum Ursprung des Koordinatensystems	mm bzw. inch (siehe K-0-0054)	>0
L6:	Vertikaler Abstand der Antriebsrolle zur x-y-Ebene	mm bzw. inch (siehe K-0-0054)	>0
L7:	Durchmesser der Antriebsrolle	mm bzw. inch (siehe K-0-0054)	>0
L8:	Winkelkorrektur	Grad	-45 ... +45

Abb. 18-45: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

 wenn L5 und L6 ≠ 0 sind, werden translatorisch gewichtete Achsen verwendet.

18.11 Robotertyp 21, 2-Achs-Parallelogramm V2

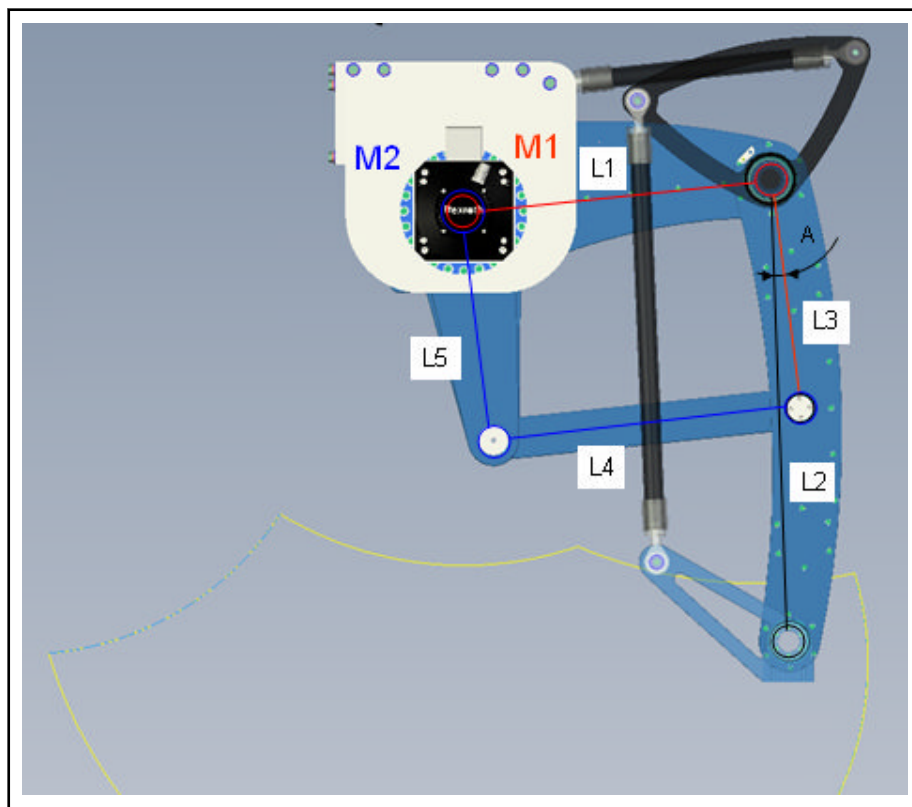


Abb. 18-46: Robotertyp 21, 2-Achs-Parallelogramm V2

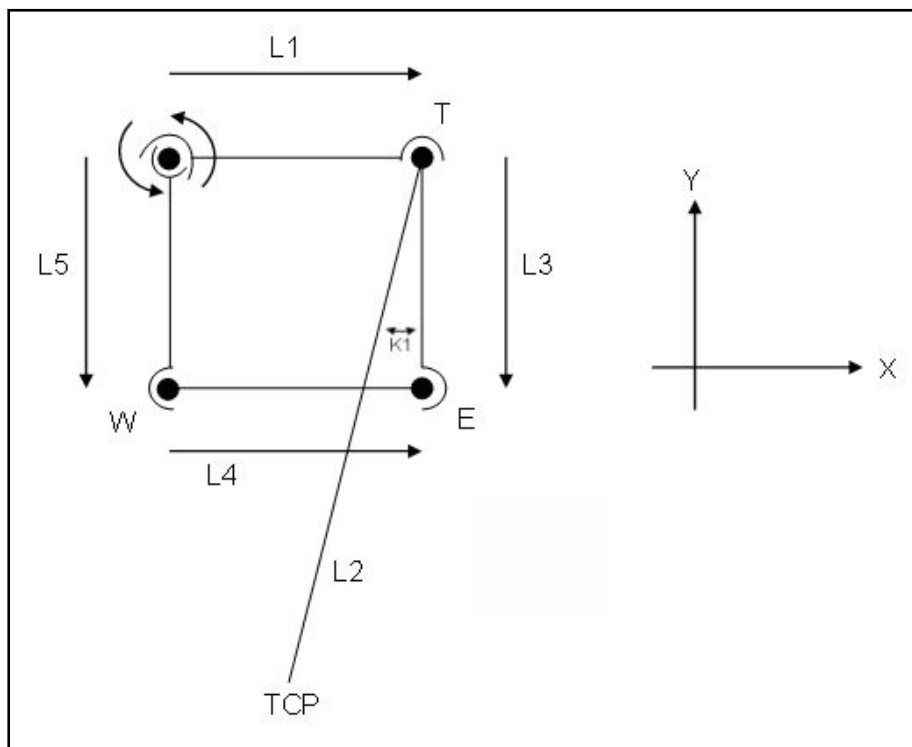


Abb. 18-47: Robotertyp 21, 2-Achs-Parallelogramm V2, in Nullstellung der Achsen

Parametereinstellungen

Transformationsvorschriften

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 89 Grad
A3:	Positionierungsachse	translatorisch absolut	optional
A4:	Orientierungsachse	rotatorisch abs./mod.	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-48: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 21 (2-Achs-Parallelogramm)
- K-0-0032, Referenzieren aller Achsen = 0

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des 1. Arms	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des 2. Arms	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des 3. Arms	mm bzw. inch (siehe K-0-0054)	>0
L4:	Länge des 4. Arms	mm bzw. inch (siehe K-0-0054)	>0
L5:	Länge des 5. Arms	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-49: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Trafo-Faktor	Konfiguration	Maßeinheit	Bemerkung
K1:	Winkel zwischen L ₂ und L ₃	Grad	

Abb. 18-50: K-0-0041, Transformationsfaktoren

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kz)	
Koordinate 4:	Name der 4. Koordinate (z.B. Kb)	
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt)	

Abb. 18-51: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Bestimmung der Vorzeichen der einzelnen Längen
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Transformationsvorschriften

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-47 "Robotertyp 21, 2-Achs-Parallelogramm V2, in Nullstellung der Achsen" auf Seite 257 zeigt die schematische Darstellung der Kinematik, bei Nullstellung aller Achsen.

- Die Armlängen L_1 , L_2 , L_3 , L_4 , L_5 sind ebenfalls in den Abbildungen dargestellt. Die positiven Drehrichtungen der Achsen sind gegen den Uhrzeigersinn.
- Der Verfahrbereich der Achsen ist mechanisch so zu begrenzen, dass der Winkel zwischen den beiden Armen mit der Längenbezeichnung L_1 und L_5 nie gleich 180° beträgt. Ansonsten gibt es je nach Mechanik, 2 unterschiedliche mathematische Lösungen, was zu ungewünschten Bewegungen führen kann.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 sind immer positiv und ungleich Null einzugeben.
- Das zugehörige Raumkoordinaten-System ist durch die 2 Hauptachsen (x, y) gekennzeichnet.
- Der Raumkoordinaten-Nullpunkt liegt im Schnittpunkt der beiden Längenangaben L_1 und L_5 .

18.12 Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen

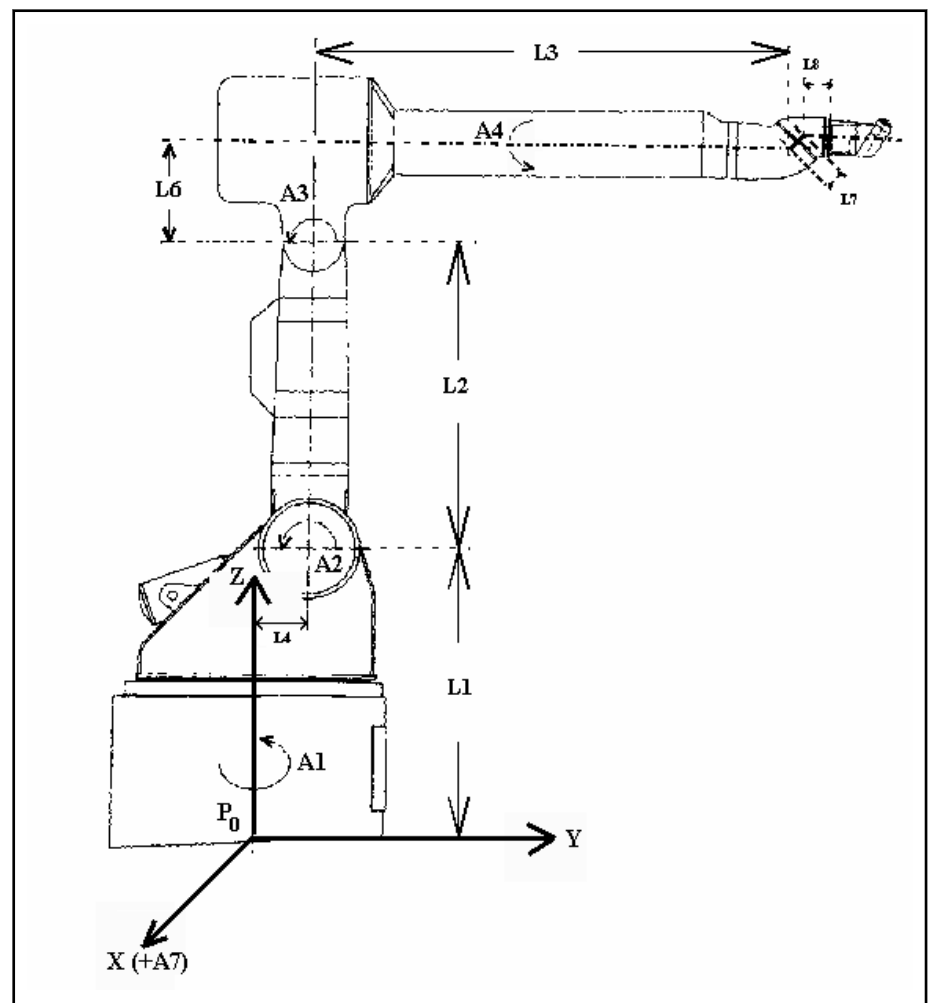


Abb. 18-52: Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen, Nullstellung

Transformationsvorschriften

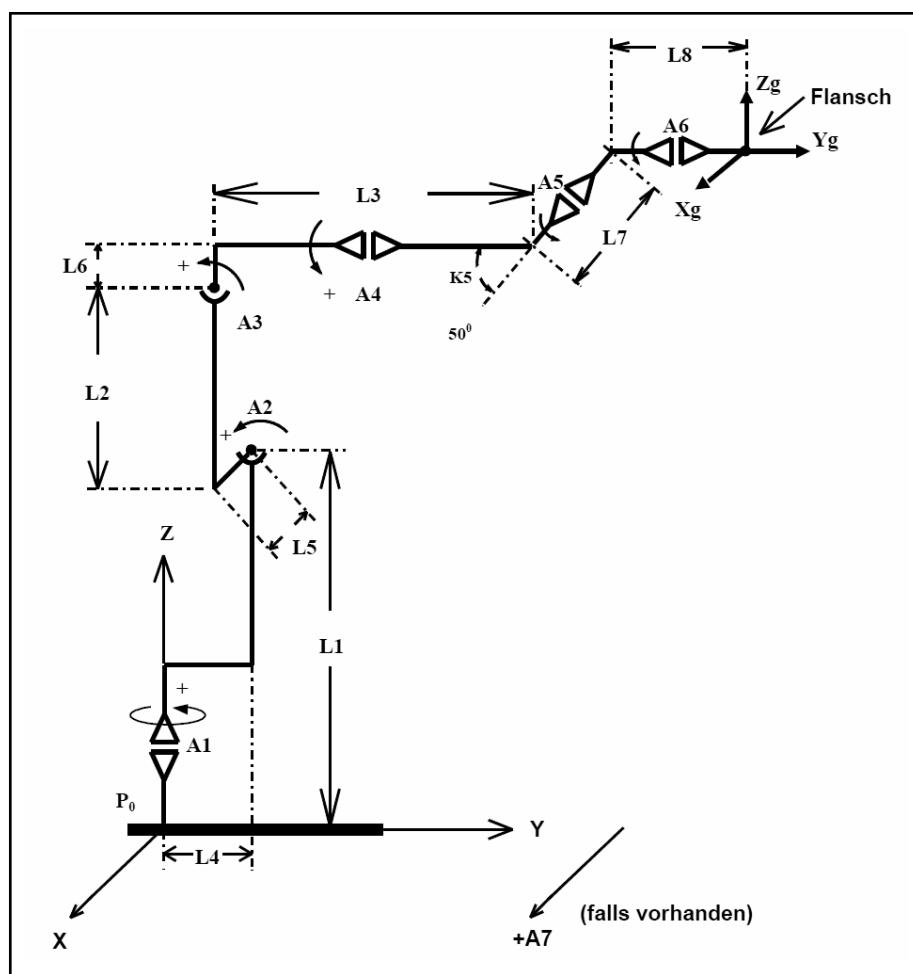


Abb. 18-53: Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen, Nullstellung Schematisch

Transformationsvorschriften

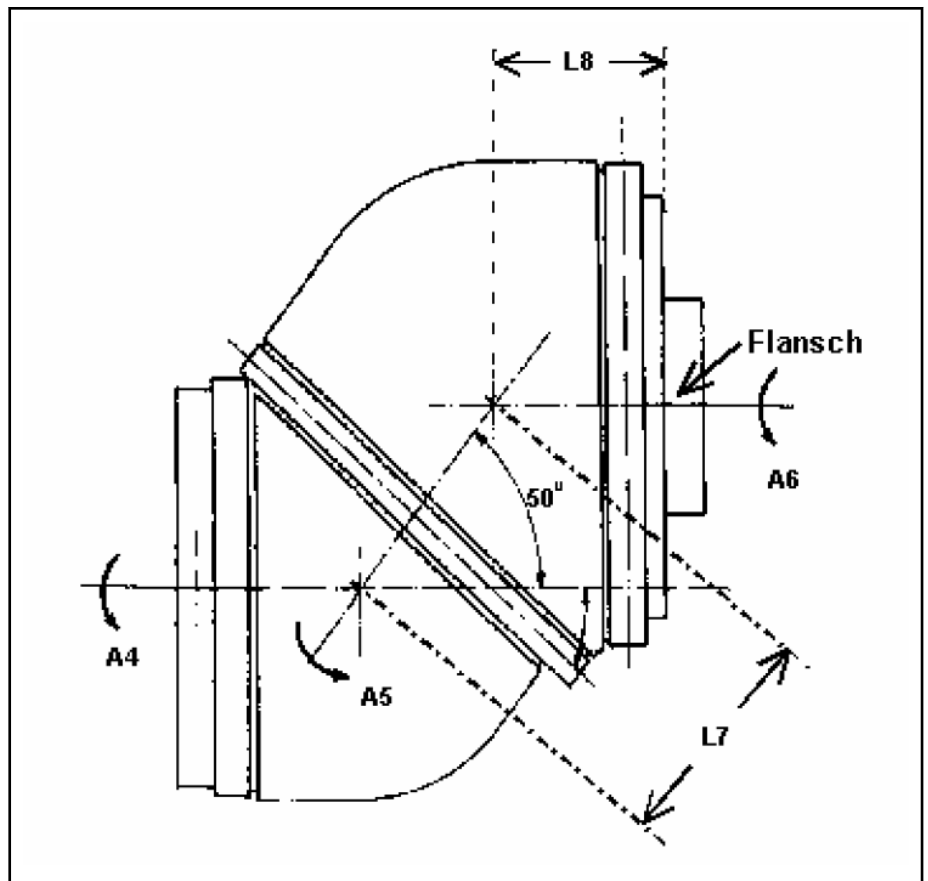


Abb. 18-54: Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen, Handgelenk

Nullstellung

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Ermittlung der Kopplungsfaktoren (falls vorhanden)
- korrekte Drehrichtungen der Achsen

⚠ VORSICHT

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden. Dies kann zu falschen und unerwarteten Bewegungen des Roboters führen.

Das zugehörige Raumkoordinaten-System ist durch die 3 Hauptachsen (x, y, z) in Abb. 18-53 "Robotertyp 26, Winkelhand-Roboter mit 6 (7) Achsen, Nullstellung Schematisch" auf Seite 260 gekennzeichnet. Der Nullpunkt liegt im Fußpunkt P0 des Roboters.



Bei Verwendung dieser Transformation in Verbindung mit PROGR_SLOPE wird die Bahn für das räumliche Überschleifen nicht von der Steuerung berechnet. Der Anwender muss selbst tangentielle Übergänge beim Satzwechsel programmieren.

Einstellungen in der Achskonfigurationsliste, Achszuordnung (K-0-0008)

Folgende Einstellungen sind in der Kinematik und Achskonfiguration vorzunehmen:

Transformationsvorschriften

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A3:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A4:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A5:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A6:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A7:	Orientierungsachse	translatorisch absolut	parallel zum Band (optional)

Abb. 18-55: K-0-0008 Achskonfigurationsliste, Achszuordnung

Transformationsschema, Typ
(K-0-0031)

Als Transformationsschema muss 26 eingestellt werden.

Referenzieren aller Achsen
(K-0-0032)

Als Wert für diesen Parameter muss 1 gewählt werden.

IPOS; Aktualisierung bei Koordinatenwechsel (K-0-0033)

Als Wert für diesen Parameter muss 1 gewählt werden.

Transformationslängen, Achskonfiguration, Achslängen (K-0-0040)

Alle Daten sind [Abb. 18-53 "Robotertyp 26, Winkelhand-Roboter mit 6 \(7\) Achsen, Nullstellung Schematisch" auf Seite 260](#) zu entnehmen:

Trafo-Länge	Bedeutung	Maßeinheit	Einschränkung
L1:	Armlänge L_1	mm bzw. inch (siehe K-0-0054)	≥ 0
L2:	Armlänge L_2	mm bzw. inch (siehe K-0-0054)	> 0
L3:	Armlänge L_3	mm bzw. inch (siehe K-0-0054)	> 0
L4:	Außermittigkeit L_4	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L5:	Außermittigkeit L_5	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L6:	Außermittigkeit L_6	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L7:	Armlänge L_7	mm bzw. inch (siehe K-0-0054)	≥ 0
L8:	Armlänge L_8	mm bzw. inch (siehe K-0-0054)	≥ 0

Abb. 18-56: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Die Außermittigkeiten L_4 , L_5 , L_6 können sowohl positiv als auch negativ sein:

- $L_4 > 0$: Außermittigkeit in (+y)-Richtung
- $L_4 < 0$: Außermittigkeit in (-y)-Richtung
- $L_5 > 0$: Außermittigkeit in (+x)-Richtung
- $L_5 < 0$: Außermittigkeit in (-x)-Richtung
- $L_6 > 0$: Außermittigkeit in (+z)-Richtung
- $L_6 < 0$: Außermittigkeit in (-z)-Richtung

Transformationsfaktoren, Achskonfiguration, Achskopplung
(K-0-0041)

Transformationsvorschriften

Kopplungsfaktor	Bedeutung	Einschränkung
K1:	K ₄₅ mechanische Kopplung zwischen Achse 4 / 5	ohne Einschränkung
K2:	K ₄₆ mechanische Kopplung zwischen Achse 4 / 6	ohne Einschränkung
K3:	K ₅₆ mechanische Kopplung zwischen Achse 5 / 6	ohne Einschränkung
K4:	K ₂₃ mechanische Kopplung zwischen Achse 2 / 3	ohne Einschränkung
K5:	Knickwinkel der 5. Achse (50°)	ohne Einschränkung
K6:	max. Summe der Handachsen (1080°)	ohne Einschränkung
K7:	Abbruchgenauigkeit für Iteration	>= 0
K8:	max. Anzahl von Iterationsschritten + Auswahl der Orientierungsdefinition + Aktivierung der Tensor-Interpolation	>= 0

Abb. 18-57: K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung

Kopplungsfaktor K1 bis K4

Falls mechanische Achskopplungen vorhanden sind, müssen die entsprechenden Kopplungsfaktoren ermittelt werden. Sie sind folgendermaßen definiert:

$$K_{23} = \Delta G_3 / \Delta A_2$$

$$K_{45} = \Delta G_5 / \Delta A_4$$

$$K_{46} = \Delta G_6 / \Delta A_4$$

$$K_{56} = \Delta G_6 / \Delta A_5$$

Dabei bezeichnen A_i die vom Messsystem erfaßten Achswinkel und G_i die Gelenkwinkel. D.h. der Kopplungsfaktor K_{ij} gibt an, wie sich eine Änderung der Achse i auf das Gelenk j auswirkt, ohne dass das Messsystem j diese Änderung mitbekommt.

Kopplungsfaktor K5

Winkel, um den die Roboterhand abgelenkt ist (50°), [Abb. 18-53 "Roboter-typ 26, Winkelhand-Roboter mit 6 \(7\) Achsen, Nullstellung Schematisch" auf Seite 260](#)

Kopplungsfaktor K6

Maximale Summe der 3 Handachsen (1080°). Wird der Wert überschritten, so erfolgt ein Bewegungsabbruch mit der Meldung "Handachs-Überdrehung"

Kopplungsfaktor K7

Abbruchgenauigkeit in Grad für die Iteration der Rückwärtstransformation. Ist der Wert kleiner 1 (z.B. 0.01), so wird für Positions- (Achse 1, 2, 3) und Orientierungsachsen (Achse 4, 5, 6) die gleiche Abbruchgenauigkeit appliziert. Mit einem Wert größer 1 besteht die Möglichkeit, unterschiedliche Abbruchkriterien für Positions- und Orientierungsachsen zu applizieren: Der Nachkommawert gibt die Abbruchgenauigkeit der Positionachsen an; die Abbruchgenauigkeit der Orientierungsachsen ergibt sich aus Nachkommawert multipliziert mit dem Vorkommawert (z.B. 20.01 => Positionsgenauigkeit = 0.01° Orientierungsgenauigkeit = 20 x 0.01° = 0.2°)

Kopplungsfaktor K8

Die erste und zweite Stelle von K8 gibt die maximal zulässige Anzahl von Iterationen an (max. 99, Empfehlung 8). Ist nach der eingestellten Anzahl von Iterationen noch keine Konvergenz erzielt worden, so erfolgt ein Bewegungsabbruch mit der Meldung "Iteration divergent". Die dritte und vierte Stelle von K8 (Rk-Orientierungstyp) legt die Definition der Orientierungswinkel (Raumkoord. 4, 5, 6) fest. Ist die fünfte Stelle von K8 gleich 1, so ist die Tensor-Interpolation aktiviert.

Beispiel:

Transformationsvorschriften

- $K8 = 8$
 - maximal 8 Iterationen
 - Rk-Orientierungstyp 0
 - keine Tensor-Interpolation
- $K8 = 312$
 - maximal 12 Iterationen
 - Rk-Orientierungstyp 3
 - keine Tensor-Interpolation
- $K8 = 10807$
 - maximal 7 Iterationen
 - Rk-Orientierungstyp 8
 - mit Tensor-Interpolation

Bei Rk-Orientierungstyp = 0, 4 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste z -Achse
- $Rk[5]$ = Drehung um die sich ergebende y'-Achse
- $Rk[6]$ = Drehung um die sich ergebende x''-Achse

Bei Rk-Orientierungstyp = 1, 5 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste z-Achse
- $Rk[6]$ = Drehung um die sich ergebende x'-Achse
- $Rk[5]$ = Drehung um die sich ergebende z''-Achse

Bei Rk-Orientierungstyp = 2, 6 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste z-Achse
- $Rk[5]$ = Drehung um die sich ergebende x'-Achse
- $Rk[6]$ = Drehung um die sich ergebende y''-Achse

Bei Rk-Orientierungstyp = 3, 7 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste y-Achse
- $Rk[5]$ = Drehung um die sich ergebende x'-Achse
- $Rk[6]$ = Drehung um die sich ergebende z''-Achse

Bei Rk-Orientierungstyp = 8 gilt folgende Definition (Eulerwinkel):

- $Rk[4]$ = Drehung um die raumfeste x-Achse
- $Rk[5]$ = Drehung um die sich ergebende y'-Achse
- $Rk[6]$ = Drehung um die sich ergebende z''-Achse

Bei den Rk-Orientierungstypen 0, 1, 2, 3 sind die Raumkoordinaten 4, 5, 6 unbeschränkt, bei den Typen 4, 5, 6, 7 gelten jeweils folgende Begrenzungen:

- $-90^\circ < Rk[4] < 90^\circ$
- $-180^\circ < Rk[5] < 180^\circ$
- $-180^\circ < Rk[6] < 180^\circ$

Für die "klassischen" Eulerwinkel (Typ 8) gelten folgende Begrenzungen:

- $-180^\circ < Rk[4] < 180^\circ$
- $0^\circ < Rk[5] < 180^\circ$

- $-180^\circ < Rk[6] < 180^\circ$

Über ein 5-stelliges K_8 wird die Tensor-Interpolation aktiviert; d.h.: Im Automatikbetrieb werden die Raumorientierungen ($RK[4]$, $RK[5]$, $RK[6]$) bei Linearbewegungen über einen Drehtensor interpoliert. Bei der Tensor-Interpolation muss eine der beschränkten Rk -Orientierungstypen über den Kopplungsfaktor K_8 gewählt werden (d.h. RK -Orientierungstyp = 4, 5, 6, 7, 8).

18.13 Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen

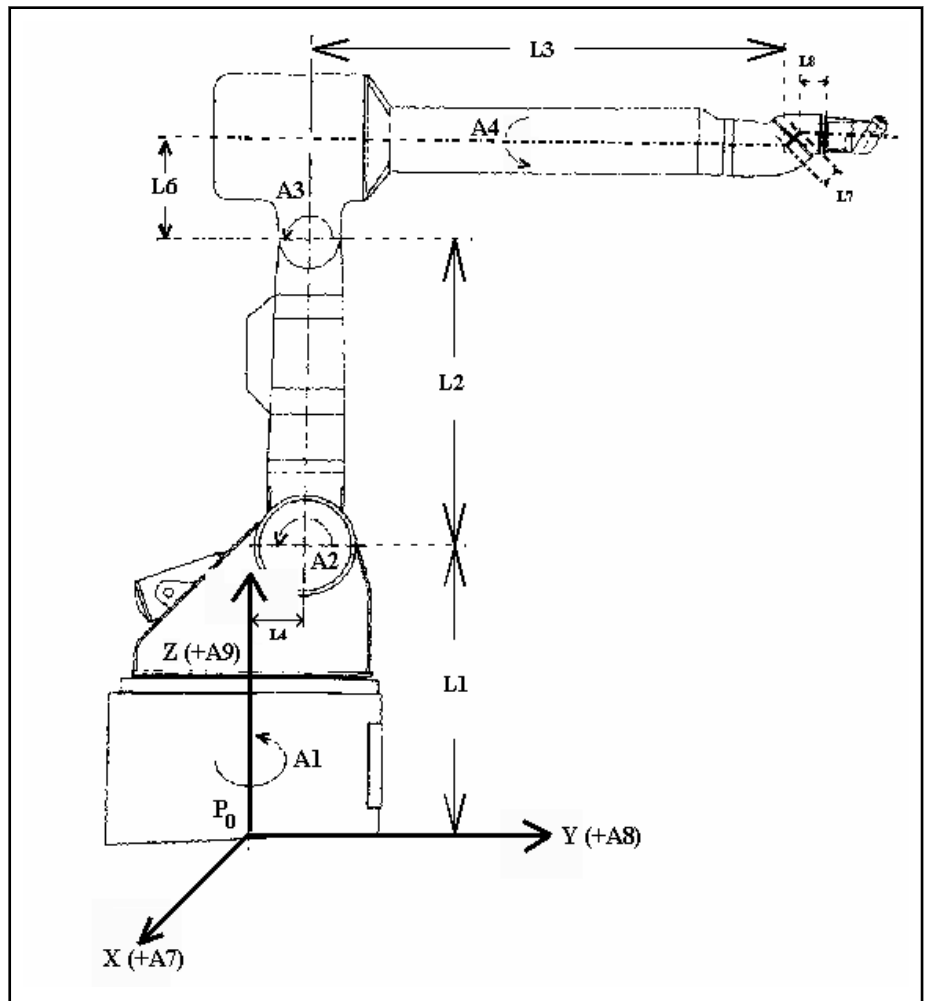


Abb. 18-58: Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Nullstellung

Transformationsvorschriften

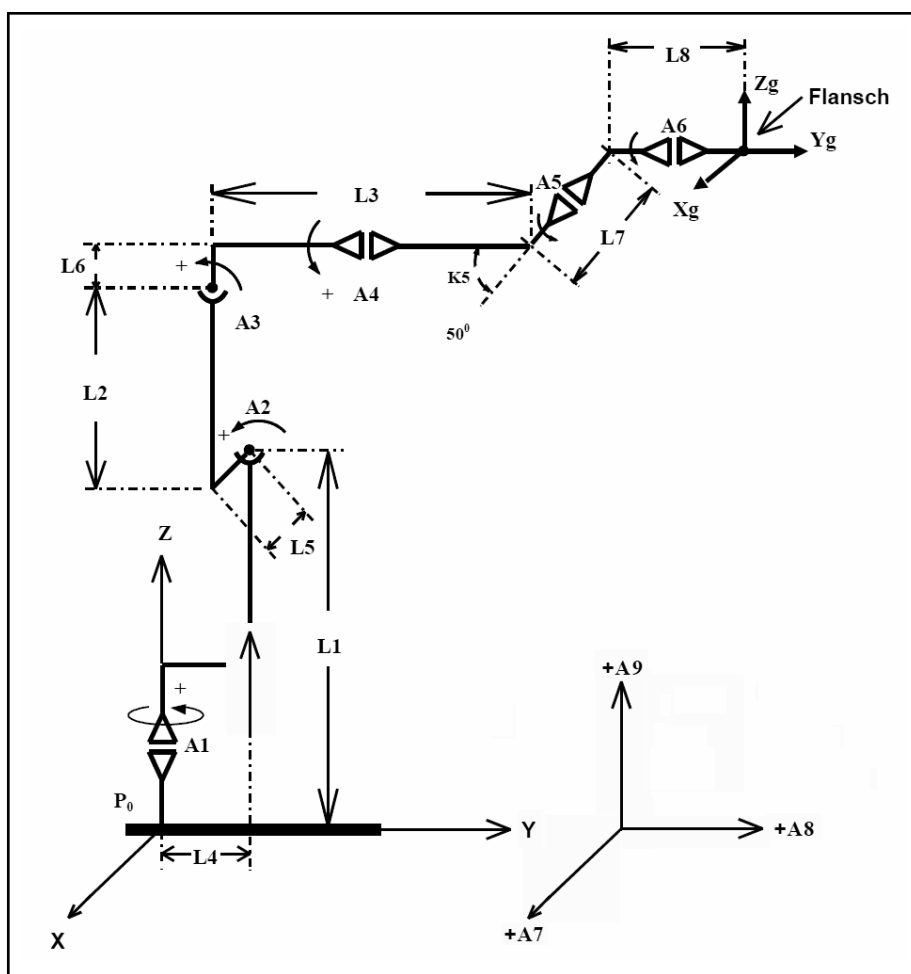


Abb. 18-59: Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Nullstellung Schematisch

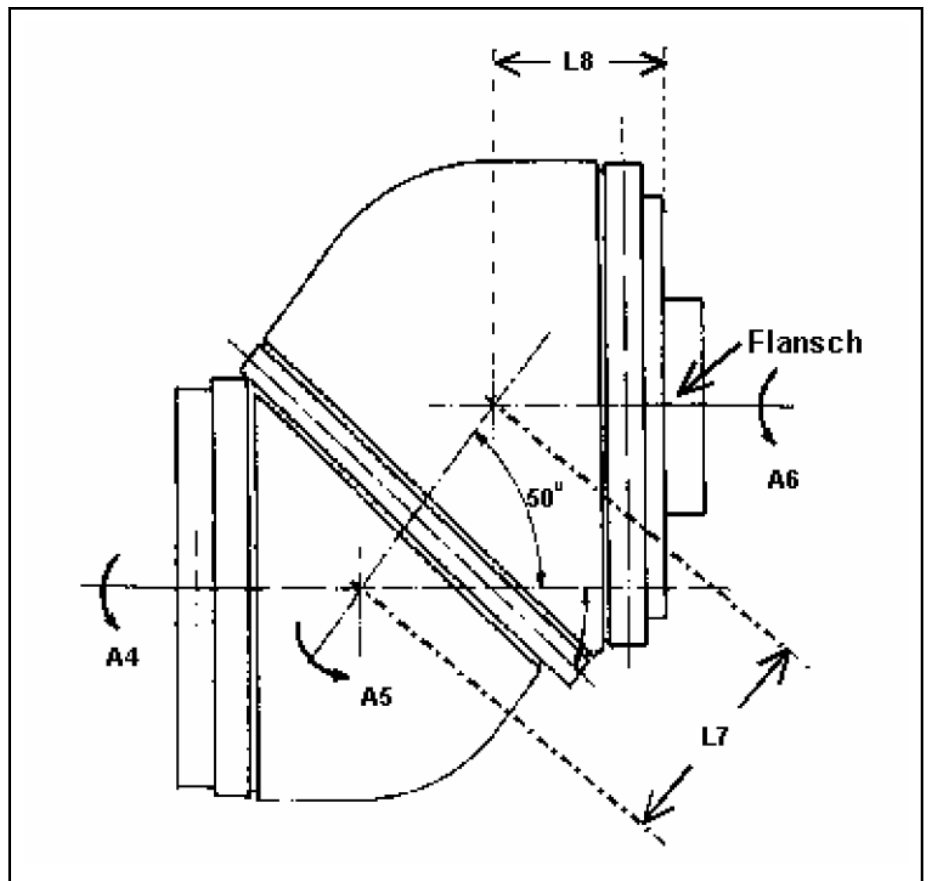


Abb. 18-60: Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Handgelenk

Nullstellung

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Ermittlung der Kopplungsfaktoren (falls vorhanden)
- korrekte Drehrichtungen der Achsen

⚠ VORSICHT

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden. Dies kann zu falschen und unerwarteten Bewegungen des Roboters führen.

Das zugehörige Raumkoordinaten-System ist durch die 3 Hauptachsen (x, y, z) in [Abb. 18-59 "Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Nullstellung Schematisch"](#) auf Seite 266 gekennzeichnet. Der Nullpunkt liegt im Fußpunkt P0 des Roboters.



Bei Verwendung dieser Transformation in Verbindung mit `PROGR_SLOPE` wird die Bahn für das räumliche Überschleifen nicht von der Steuerung berechnet. Der Anwender muss selbst tangentielle Übergänge beim Satzwechsel programmieren.

Einstellungen in der Achskonfigurationsliste, Achszuordnung (K-0-0008)

Folgende Einstellungen sind in der Kinematik und Achskonfiguration vorzunehmen:

Transformationsvorschriften

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 179 Grad
A3:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A4:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A5:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A6:	Orientierungsachse	rotatorisch absolut	unbeschränkt
A7:	Orientierungsachse	translatorisch absolut	parallel zur x-Achse
A8:	Orientierungsachse	translatorisch absolut	parallel zur y-Achse
A9:	Orientierungsachse	translatorisch absolut	parallel zur z-Achse

Abb. 18-61: K-0-0008 Achskonfigurationsliste, Achszuordnung

Transformationsschema, Typ
(K-0-0031)

Als Transformationsschema muss 27 eingestellt werden.

Referenzieren aller Achsen
(K-0-0032)

Als Wert für diesen Parameter muss 1 gewählt werden.

IPOS; Aktualisierung bei Koordinatenwechsel (K-0-0033)

Als Wert für diesen Parameter muss 1 gewählt werden.

Transformationslängen, Achskonfiguration, Achslängen (K-0-0040)

Alle Daten sind [Abb. 18-59 "Robotertyp 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Nullstellung Schematisch" auf Seite 266](#) zu entnehmen:

Trafo-Länge	Bedeutung	Maßeinheit	Einschränkung
L1:	Armlänge L_1	mm bzw. inch (siehe K-0-0054)	≥ 0
L2:	Armlänge L_2	mm bzw. inch (siehe K-0-0054)	> 0
L3:	Armlänge L_3	mm bzw. inch (siehe K-0-0054)	> 0
L4:	Außermittigkeit L_4	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L5:	Außermittigkeit L_5	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L6:	Außermittigkeit L_6	mm bzw. inch (siehe K-0-0054)	ohne Einschränkung
L7:	Armlänge L_7	mm bzw. inch (siehe K-0-0054)	≥ 0
L8:	Armlänge L_8	mm bzw. inch (siehe K-0-0054)	≥ 0

Abb. 18-62: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Die Außermittigkeiten L_4 , L_5 , L_6 können sowohl positiv als auch negativ sein:

- $L_4 > 0$: Außermittigkeit in (+y)-Richtung
- $L_4 < 0$: Außermittigkeit in (-y)-Richtung
- $L_5 > 0$: Außermittigkeit in (+x)-Richtung
- $L_5 < 0$: Außermittigkeit in (-x)-Richtung
- $L_6 > 0$: Außermittigkeit in (+z)-Richtung
- $L_6 < 0$: Außermittigkeit in (-z)-Richtung

Transformationsfaktoren, Achskonfiguration, Achskopplung
(K-0-0041)

Transformationsvorschriften

Kopplungsfaktor	Bedeutung	Einschränkung
K1:	K ₄₅ mechanische Kopplung zwischen Achse 4 / 5	ohne Einschränkung
K2:	K ₄₆ mechanische Kopplung zwischen Achse 4 / 6	ohne Einschränkung
K3:	K ₅₆ mechanische Kopplung zwischen Achse 5 / 6	ohne Einschränkung
K4:	K ₂₃ mechanische Kopplung zwischen Achse 2 / 3	ohne Einschränkung
K5:	Knickwinkel der 5. Achse (50°)	ohne Einschränkung
K6:	Variante der linearen Zusatzachsen	ohne Einschränkung
K7:	Abbruchgenauigkeit für Iteration	>=0
K8:	max. Anzahl von Iterationsschritten + Auswahl der Orientierungsdefinition + Aktivierung der Tensor-Interpolation	>=0

Abb. 18-63: K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung

Kopplungsfaktor K1 bis K4

Falls mechanische Achskopplungen vorhanden sind, müssen die entsprechenden Kopplungsfaktoren ermittelt werden. Sie sind folgendermaßen definiert:

$$K_{23} = \Delta G_3 / \Delta A_2$$

$$K_{45} = \Delta G_5 / \Delta A_4$$

$$K_{46} = \Delta G_6 / \Delta A_4$$

$$K_{56} = \Delta G_6 / \Delta A_5$$

Dabei bezeichnen A_i die vom Messsystem erfaßten Achswinkel und G_i die Gelenkwinkel. D.h. der Kopplungsfaktor K_{ij} gibt an, wie sich eine Änderung der Achse i auf das Gelenk j auswirkt, ohne dass das Messsystem j diese Änderung mitbekommt.

Kopplungsfaktor K5

Winkel, um den die Roboterhand abgelenkt ist (50°), [Abb. 18-59 "Roboter-typ 27, Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen, Nullstellung Schematisch" auf Seite 266](#)

Kopplungsfaktor K6

Varianten der linearen Zusatzachsen

Eintrag im K6	Achsanzahl	Lineare Zusatzachsen
0	6	ohne lineare Zusatzachse
-11	7	lineare Zusatzachse in x-Richtung
-12	7	lineare Zusatzachse in y-Richtung
-13	7	lineare Zusatzachse in z-Richtung
-21	8	lineare Zusatzachsen in x- und y-Richtung
-22	8	lineare Zusatzachsen in x- und z-Richtung
-23	8	lineare Zusatzachsen in y- und z-Richtung
10	9	lineare Zusatzachse in x-, y- und z-Richtung sind virtuelle Achsen
11	9	lineare Zusatzachse in x-Richtung, y- und z-Richtung sind virtuelle Achsen
12	9	lineare Zusatzachse in y-Richtung, x- und z-Richtung sind virtuelle Achsen

Transformationsvorschriften

13	9	lineare Zusatzachse in z-Richtung, x- und y-Richtung sind virtuelle Achsen
21	9	lineare Zusatzachsen in x- und y-Richtung, z-Richtung ist virtuelle Achse
22	9	lineare Zusatzachsen in x- und z-Richtung, y-Richtung ist virtuelle Achse
23	9	lineare Zusatzachsen in y- und z-Richtung, x-Richtung ist virtuelle Achse
31	9	lineare Zusatzachsen in x-, y- und z-Richtung
Sonst	6	keine Zusatzachse vorhanden

Abb. 18-64: Varianten der linearen Zusatzachsen

Virtuelle Achsen müssen formal angelegt werden. Bei Punktangaben müssen sie als Koordinaten mit dem Wert 0 mitgeführt werden.

Kopplungsfaktor K7

Abbruchgenauigkeit in Grad für die Iteration der Rückwärtstransformation. Ist der Wert kleiner 1 (z.B. 0.01), so wird für Positions- (Achse 1, 2, 3) und Orientierungsachsen (Achse 4, 5, 6) die gleiche Abbruchgenauigkeit appliziert. Mit einem Wert größer 1 besteht die Möglichkeit, unterschiedliche Abbruchkriterien für Positions- und Orientierungsachsen zu applizieren: Der Nachkommawert gibt die Abbruchgenauigkeit der Positionsachsen an; die Abbruchgenauigkeit der Orientierungsachsen ergibt sich aus Nachkommawert multipliziert mit dem Vorkommawert (z.B. 20.01 => Positionsgenauigkeit = 0.01° Orientierungsgenauigkeit = 20 x 0.01° = 0.2°)

Kopplungsfaktor K8

Die erste und zweite Stelle von K8 gibt die maximal zulässige Anzahl von Iterationen an (max. 99, Empfehlung 8). Ist nach der eingestellten Anzahl von Iterationen noch keine Konvergenz erzielt worden, so erfolgt ein Bewegungsabbruch mit der Meldung "Iteration divergent". Die dritte und vierte Stelle von K8 (Rk-Orientierungstyp) legt die Definition der Orientierungswinkel (Raumkoord. 4, 5, 6) fest. Ist die fünfte Stelle von K8 gleich 1, so ist die Tensor-Interpolation aktiviert.

Beispiel:

- K8 = 8
 - maximal 8 Iterationen
 - Rk-Orientierungstyp 0
 - keine Tensor-Interpolation
- K8 = 312
 - maximal 12 Iterationen
 - Rk-Orientierungstyp 3
 - keine Tensor-Interpolation
- K8 = 10807
 - maximal 7 Iterationen
 - Rk-Orientierungstyp 8
 - mit Tensor-Interpolation

Bei Rk-Orientierungstyp = 0, 4 gilt folgende Definition:

- Rk[4] = Drehung um die raumfeste z -Achse
- Rk[5] = Drehung um die sich ergebende y'-Achse
- Rk[6] = Drehung um die sich ergebende x''-Achse

Bei Rk-Orientierungstyp = 1, 5 gilt folgende Definition:

- Rk[4] = Drehung um die raumfeste z-Achse
- Rk[6] = Drehung um die sich ergebende x'-Achse

Transformationsvorschriften

- $Rk[5]$ = Drehung um die sich ergebende z'' -Achse

Bei Rk -Orientierungstyp = 2, 6 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste z -Achse
- $Rk[5]$ = Drehung um die sich ergebende x' -Achse
- $Rk[6]$ = Drehung um die sich ergebende y'' -Achse

Bei Rk -Orientierungstyp = 3, 7 gilt folgende Definition:

- $Rk[4]$ = Drehung um die raumfeste y -Achse
- $Rk[5]$ = Drehung um die sich ergebende x' -Achse
- $Rk[6]$ = Drehung um die sich ergebende z'' -Achse

Bei Rk -Orientierungstyp = 8 gilt folgende Definition (Eulerwinkel):

- $Rk[4]$ = Drehung um die raumfeste x -Achse
- $Rk[5]$ = Drehung um die sich ergebende y' -Achse
- $Rk[6]$ = Drehung um die sich ergebende z'' -Achse

Bei den Rk -Orientierungstypen 0, 1, 2, 3 sind die Raumkoordinaten 4, 5, 6 unbeschränkt, bei den Typen 4, 5, 6, 7 gelten jeweils folgende Begrenzungen:

- $-90^\circ < Rk[4] < 90^\circ$
- $-180^\circ < Rk[5] < 180^\circ$
- $-180^\circ < Rk[6] < 180^\circ$

Für die "klassischen" Eulerwinkel (Typ 8) gelten folgende Begrenzungen:

- $-180^\circ < Rk[4] < 180^\circ$
- $0^\circ < Rk[5] < 180^\circ$
- $-180^\circ < Rk[6] < 180^\circ$

Über ein 5-stelliges K_8 wird die Tensor-Interpolation aktiviert; d.h.: Im Automatikbetrieb werden die Raumorientierungen ($RK[4]$, $RK[5]$, $RK[6]$) bei Linearbewegungen über einen Drehtensor interpoliert. Bei der Tensor-Interpolation muss eine der beschränkten Rk -Orientierungstypen über den Kopplungsfaktor K_8 gewählt werden (d.h. RK -Orientierungstyp = 4, 5, 6, 7, 8).

Transformationsvorschriften

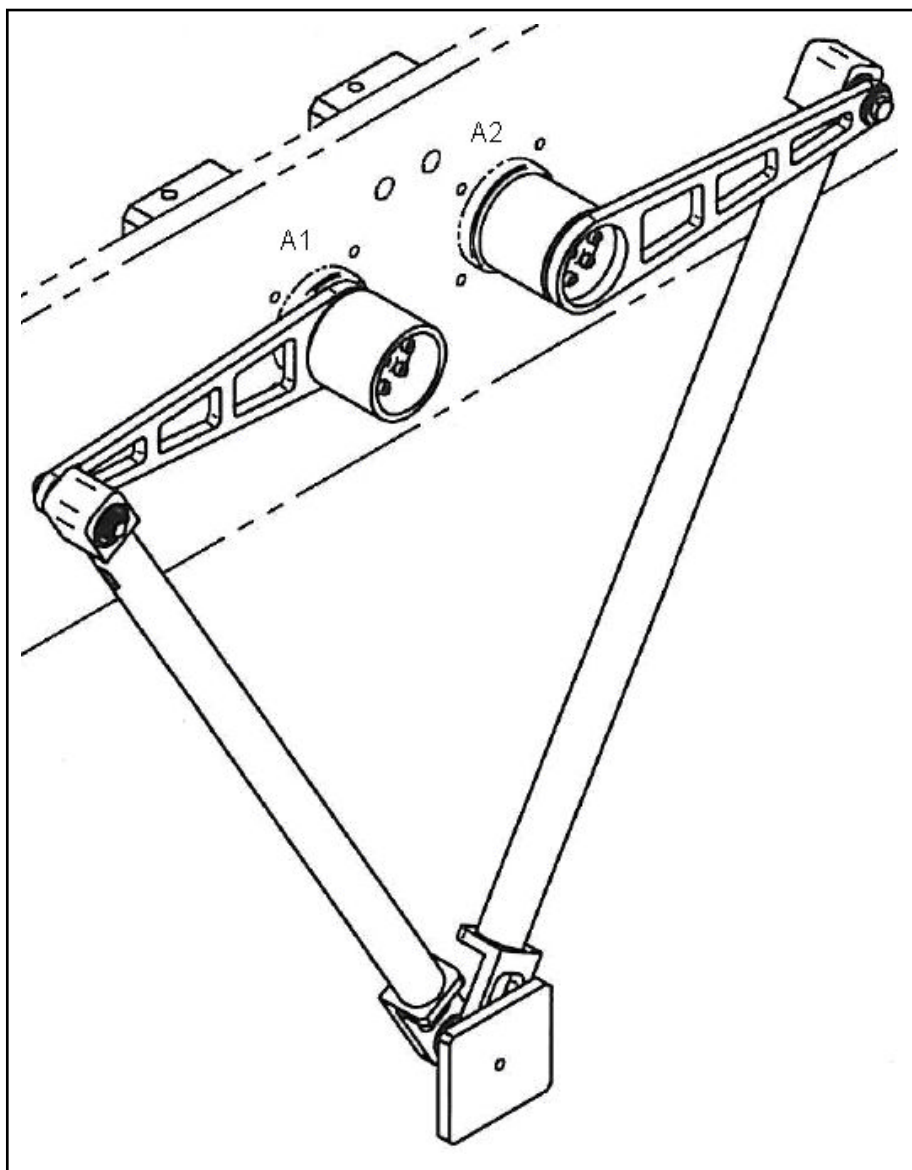
18.14 Robotertyp 29, 2-Achs-Delta mit rotatorischer und translatorischer Achse

Abb. 18-65: Robotertyp 29, 2-Achs-Delta

Transformationsvorschriften

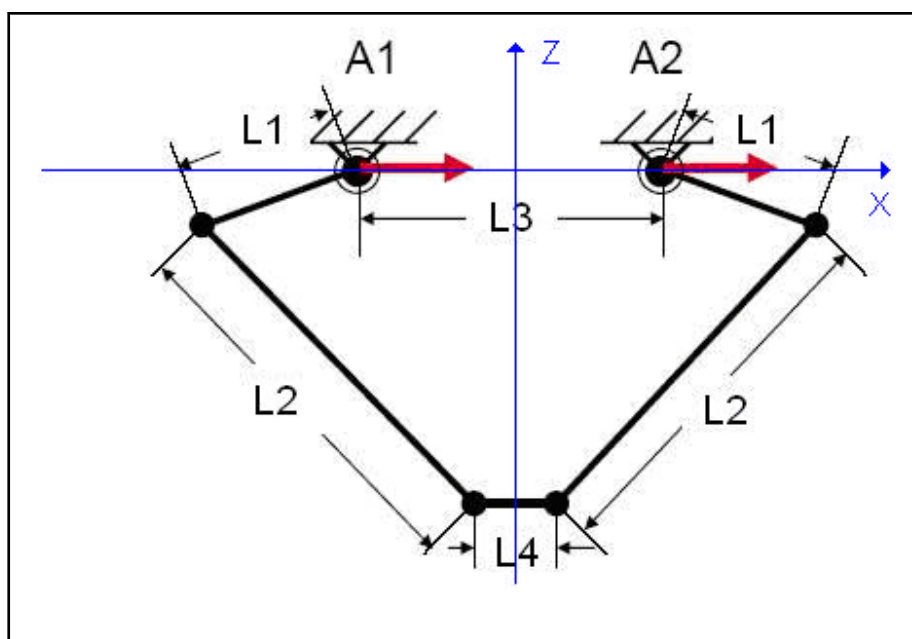


Abb. 18-66: Robotertyp 29, 2-Achs-Delta (X-Z Ansicht)

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+ 90 .. 270 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A3:	Positionierungsachse	translatorisch absolut	optional
A4:	Orientierungsachse	rotatorisch abs./modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-67: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 29 (Delta 2 Achs, Bipod)
- K-0-0032, Referenzieren alle Achsen = 0
- K-0-0033, IPOS-Aufnahme bei Koordinatenwechsel = 0
- K-0-0036, RK-Achsüberwachung = 1

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des Arms (Schenkel) am Motor	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des Stabs zur Werkzeugplatte	mm bzw. inch (siehe K-0-0054)	>0
L3:	Abstand der beiden Rotationspunkte der Arme	mm bzw. inch (siehe K-0-0054)	>0
L4:	Abstand der Stäbe an der Grundplatte	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-68: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Transformationsvorschriften

K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung, ist nicht verwendet.

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Kz)	
Koordinate 3:	Name der 3. Koordinate (z.B. Ky)	
Koordinate 4:	Name der 4. Koordinate (z.B. Kc)	
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt1)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kbelt2)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kbelt3)	optional
Koordinate 8:	Name der 8. Koordinate (z.B. Kbelt4)	optional

Abb. 18-69: K-0-0121 .. K-0-0128, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-65 "Robotertyp 29, 2-Achs-Delta" auf Seite 272 zeigt einen Bipod-Roboter in 3D-Darstellung mit den Achspositionen A1=180° und A2 = 0°.

In der Nullstellung stehen beide Achsen 1 und 2 (Positionierachsen) waage-recht in Richtung der X-Achse (siehe rote Pfeile in Abb. 18-66 "Robotertyp 29, 2-Achs-Delta (X-Z Ansicht)" auf Seite 273).

- Die Armlänge L_1 ist gemessen vom Rotationsmittelpunkt bis zum Mittelpunkt der Aufnahme der Stäbe.
- Die Stablänge L_2 ist gemessen von den beidseitigen Aufnahmepunkten.
- L_3 ist der Abstand der Rotationsmittelpunkte der beiden Arme.
- L_4 ist die Länge der Werkzeugplatte gemessen an den Lagerungen der Stäbe.
- Die Drehrichtung der Positionierachsen A1 und A2 ist positiv entgegen dem Uhrzeigersinn.
- Der kartesische Nullpunkt des Bipod-Roboters liegt in der Mitte der beiden Armdrehpunkte A1 und A2.

18.15 Robotertyp 30, Adabot

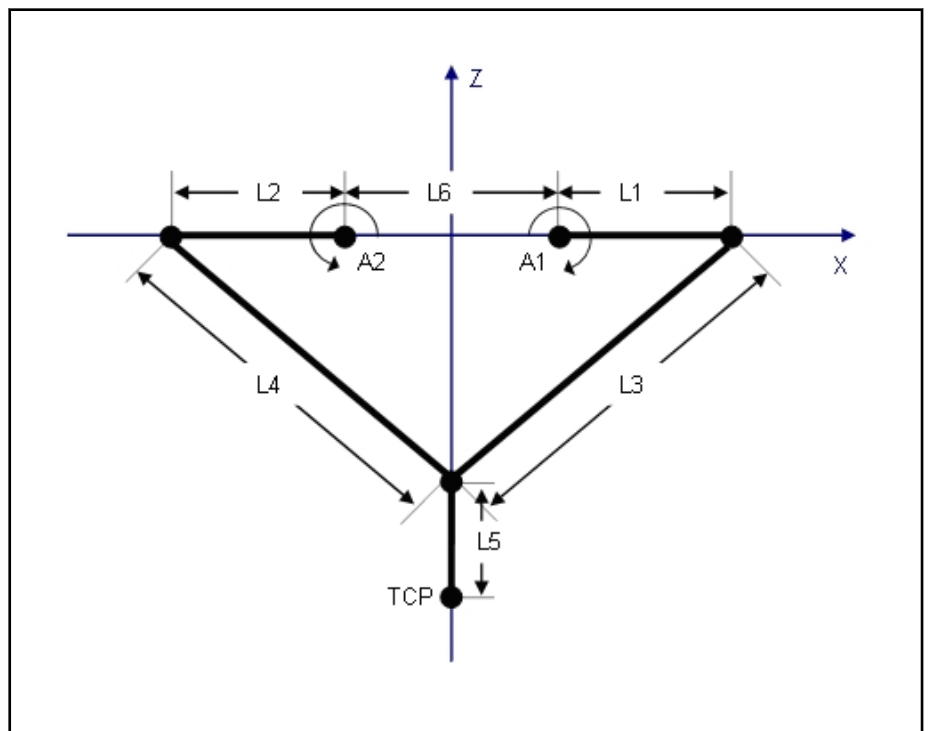


Abb. 18-70: Robotertyp 30, Adabot (X-Z Ansicht, Achsnullstellung)

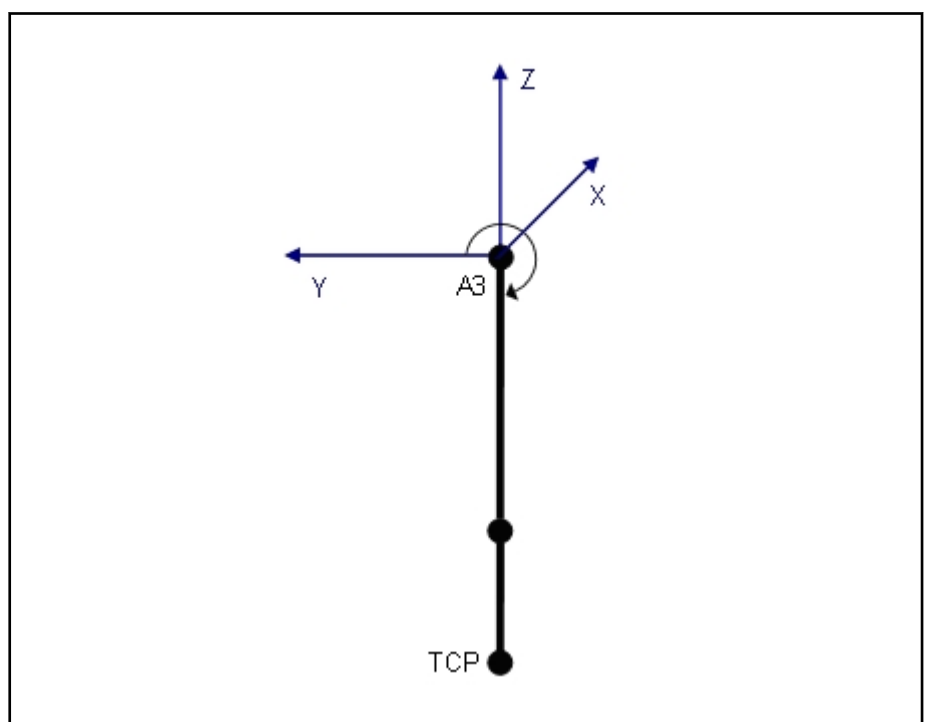


Abb. 18-71: Robotertyp 30, Adabot (Y-Z Ansicht, Achsnullstellung)

Transformationsvorschriften

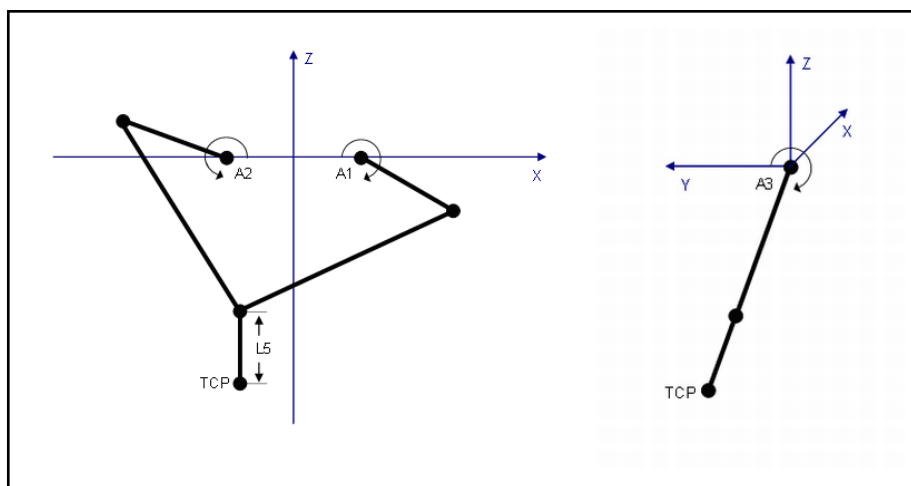


Abb. 18-72: Robotertyp 30, Adabot (Verhalten des Stabs L5)

Parametereinstellungen

Achsname	Achstype	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A3:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A4:	Orientierungsachse	rotatorisch abs./modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional
A8:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-73: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 30 (Adabot)
- K-0-0032, Referenzieren alle Achsen = 0

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des Arms (Schenkel) am Motor (rechts)	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des Arms (Schenkel) am Motor (links)	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des Stabs an Arm L1	mm bzw. inch (siehe K-0-0054)	>0
L4:	Länge des Stabs an Arm L2	mm bzw. inch (siehe K-0-0054)	>0
L5:	Stab zu TCP	mm bzw. inch (siehe K-0-0054)	>=0
L6:	Abstand der beiden Rotationspunkte der Arme	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-74: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung, ist nicht verwendet.

Transformationsvorschriften

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Ky)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kz)	
Koordinate 4:	Name der 4. Koordinate (z.B. Kc)	optional
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt1)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kbelt2)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kbelt3)	optional
Koordinate 8:	Name der 8. Koordinate (z.B. Kbelt4)	optional

Abb. 18-75: K-0-0121 .. K-0-0136, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-70 "Robotertyp 30, Adabot (X-Z Ansicht, Achsnullstellung)" auf Seite 275 und Abb. 18-71 "Robotertyp 30, Adabot (Y-Z Ansicht, Achsnullstellung)" auf Seite 275 zeigen die schematische Darstellung der Adabot-Kinematik, bei Nullstellung aller Achsen.

- Die Längen L_1 , L_2 , L_3 , L_4 , L_5 , L_6 sind ebenfalls in den Abbildungen dargestellt.
- Die positiven Drehrichtungen der Achsen A1, A2, A3 sind ebenfalls in den Abbildungen dargestellt.
- Die Längen L_1 , L_2 , L_3 , L_4 , L_6 sind immer positiv und ungleich Null einzugeben. L_5 ist immer positiv oder Null einzugeben.

Der Stab zum TCP (L_5) wird durch eine Mechanik (nicht in Abb. dargestellt) immer senkrecht zur X-Achse gehalten. Der Stab zum TCP (L_5) liegt immer in der Ebene, die durch die Stäbe L_1 , L_2 , L_3 , L_4 aufgespannt wird (siehe Abb. 18-72 "Robotertyp 30, Adabot (Verhalten des Stabs L_5)" auf Seite 276).

Der kartesische Nullpunkt der Adabot-Kinematik liegt in der Mitte der beiden Armdrehpunkte A1 und A2 (siehe Abb. 18-70 "Robotertyp 30, Adabot (X-Z Ansicht, Achsnullstellung)" auf Seite 275).

Transformationsvorschriften

18.16 Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP

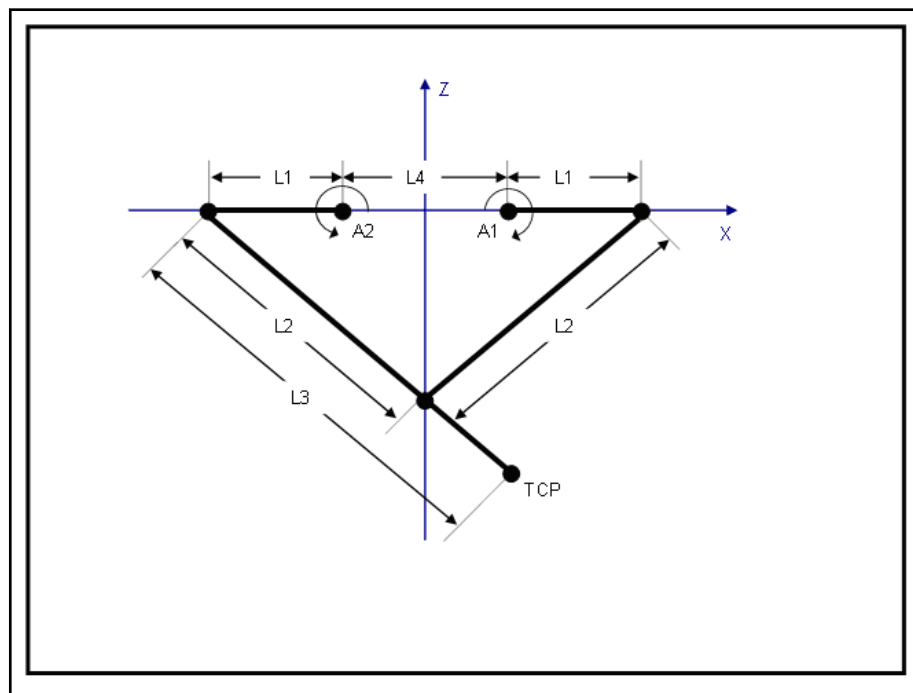


Abb. 18-76: Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP (X-Z Ansicht, Achsnullstellung)

Parametereinstellungen

Achsname	Achstyp	Wichtung	Bemerkung
A1:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A2:	Positionierungsachse	rotatorisch absolut	+/- 90 Grad
A3:	Orientierungsachse	rotatorisch abs. / modulo	optional
A4:	Bandachse (optional)	translatorisch modulo	optional
A5:	Bandachse (optional)	translatorisch modulo	optional
A6:	Bandachse (optional)	translatorisch modulo	optional
A7:	Bandachse (optional)	translatorisch modulo	optional

Abb. 18-77: K-0-0008 Achskonfigurationsliste, Achszuordnung

- K-0-0031, Transformationsschema, Typ = 32
- K-0-0032, Referenzieren alle Achsen = 0

Trafo-Länge	Achskonfiguration	Maßeinheit	Achslänge
L1:	Länge des Arms (Schenkel) am Motor	mm bzw. inch (siehe K-0-0054)	>0
L2:	Länge des kurzen Stabs	mm bzw. inch (siehe K-0-0054)	>0
L3:	Länge des langen Stabs zum TCP	mm bzw. inch (siehe K-0-0054)	>L2
L4:	Abstand der beiden Rotationspunkte der Arme	mm bzw. inch (siehe K-0-0054)	>0

Abb. 18-78: K-0-0040, Transformationslängen, Achskonfiguration, Achslängen

Transformationsvorschriften

K-0-0041, Transformationsfaktoren, Achskonfiguration, Achskopplung, ist nicht verwendet.

Name	Koordinate	Bemerkung
Koordinate 1:	Name der 1. Koordinate (z.B. Kx)	
Koordinate 2:	Name der 2. Koordinate (z.B. Kz)	
Koordinate 3:	Name der 3. Koordinate (z.B. Kc)	optional
Koordinate 4:	Name der 4. Koordinate (z.B. Kbelt1)	optional
Koordinate 5:	Name der 5. Koordinate (z.B. Kbelt2)	optional
Koordinate 6:	Name der 6. Koordinate (z.B. Kbelt3)	optional
Koordinate 7:	Name der 7. Koordinate (z.B. Kbelt4)	optional

Abb. 18-79: K-0-0121 .. K-0-0136, Koordinatennamen, Koordinaten

Achs-Drehrichtungen und Nullstellungen

Die Grundvoraussetzungen für eine korrekte Arbeitsweise der Koordinaten-Transformation sind:

- exakte Einstellung der Achs-Nullpunkte
- exakte Vermessung der Armlängen bzw. Außermittigkeiten
- korrekte Drehrichtung der Achsen

Ist eine dieser Voraussetzungen nicht 100%-ig erfüllt, so kann keine exakte Linearbewegung garantiert werden.

Dies kann zu falschen bzw. unerwarteten Bewegungen führen.

Abb. 18-76 "Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP (X-Z Ansicht, Achsnullstellung)" auf Seite 278 zeigt die schematische Darstellung der 2-Achs-Delta-Kinematik mit asymmetrischem TCP, bei Nullstellung aller Achsen.

- Die Längen L_1 , L_2 , L_3 , L_4 sind ebenfalls in der Abbildung dargestellt.
- Die positiven Drehrichtungen der Achsen A1, A2 sind ebenfalls in der Abbildung dargestellt.
- Die Längen L_1 , L_2 , L_3 , L_4 sind immer positiv und ungleich Null einzugeben. Außerdem muss L_3 immer einen größeren Wert enthalten als L_2 .
- Der kartesische Nullpunkt liegt in der Mitte der beiden Armdrehpunkte A1 und A2.

18.17 Robotertyp 100, SPS

Funktion

In der Applikation kommt es häufig vor, dass die Transformationen, die das System bereit stellt, nicht zu der eigenen Kinematik passen. Hierfür bietet das System die Möglichkeit des nutzerdefinierten Transformationstyps mit der Nummer 1100. Bei diesem Typ wird die Umrechnungsvorschriften von Welt- in Achskoordinaten und umgekehrt in der SPS implementiert.

Um solch eine Transformation zu definieren, muss das Interface MC_KIN_REF von einem Funktionsbaustein (hier Karteser) implementiert werden.

Im zweiten Schritt muss die Transformation am System angemeldet werden. Dies geschieht mit dem Funktionsbaustein MC_SetKinTransform.

Transformationsvorschriften

Diesem müssen die Funktionszeiger auf die einzelnen Funktionen des Interfaces MC_KIN_REF über die Struktur ML_KIN_AXES_CONFIG_DATA / ML_KIN_AXIS_CONFIG wie folgt übergeben werden.

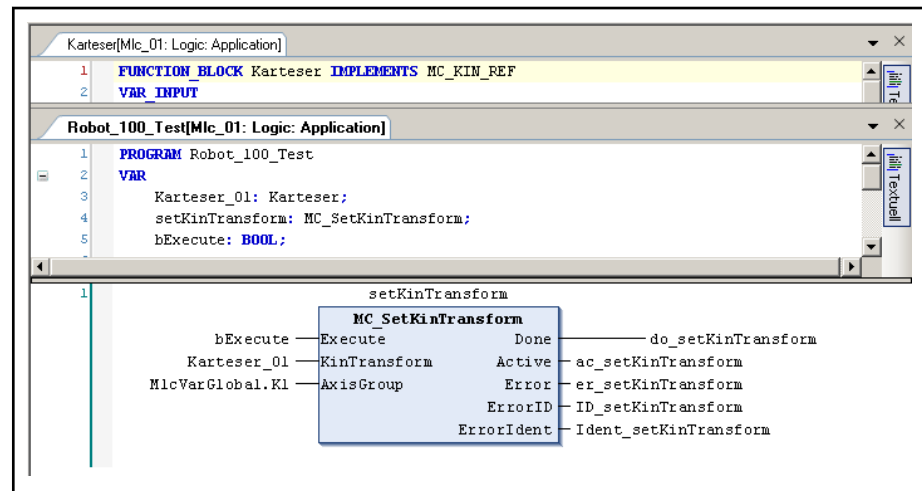


Abb. 18-80: Anmelden einer Transformation

Nachdem die Transformation am System angemeldet wurde, kann die Transformation vom System benutzt werden.

Details

Den benötigten Funktionsbaustein, das Interface und die Strukturen sind Bestandteile der Bibliothek ML_Robot_xx.library.

Die Beschreibung dieser Bibliothek ist Bestandteil der Dokumentation:

Rexroth IndraLogic XLC IndraMotion MLC 12VRS PLCopen-Bibliotheken, DOK-XLCMLC-FUNLIB**V12-L1xx-DE-P, R911334181.

Parametereinstellungen

- K-0-0031, Transformationsschema, Typ = 1100
- K-0-0040, Transformation, Längen der Mechanik
- K-0-0041, Transformation, Faktoren

19 RCL-Schlüsselworte

Nachfolgend sind alle für RCL z.Z. reservierten Sprachsymbole aufgelistet. Die aufgelisteten Sprachsymbole dürfen in einem RCL-Programm nicht als Variable, Dateiname oder Unterprogrammname verwendet werden.

Schlüsselwort	Bemerkung
@	Maschinenkoordinaten, joint coordinate points. Siehe JC_POINT in Zusammengesetzte RCL-Datentypen , Seite 55.
@MPOS	reserviert
@POS	aktuelle Position in Maschinen-Koordinaten, siehe auch Lesen der Ist-Position POS , @POS, Seite 61.
A	Bahnbeschleunigung, siehe auch A , Seite 125.
ABS	Funktion liefert den Betrag einer REAL- oder INTEGER-Zahl, ABS-Funktion , Seite 91.
AFACTOR	Bahnbeschleunigungsfaktor, siehe auch AFACTOR , Seite 126.
AFIX	AFIX , Seite 123.
ALSO	reserviert
AND	Bitweise logische UND-Verknüpfung, siehe auch Logische Ausdrücke , Seite 85.
APPROX	siehe MOVE_REL APPROX... , Seite 114.
ARRAY	Arbeit mit Feldvariablen (ARRAYs) , Seite 62.
ASC_INT	Datenkonvertierung CHAR nach INTEGER, ASC_INT-Funktion , Seite 93.
ASSIGN	Dateizugriff, ASSIGN-Funktion , Seite 92.
ATAN	ATAN-Funktion , Seite 90.
BEGIN	Beginn des Programmkodes, siehe auch Anweisungsteil , Seite 40.
BELT	Bandsynchronisation , Seite 129., Kommunikationskanäle zur SPS , Allgemeines, Seite 64.
BINARY	Übersicht , Seite 53.
BLOCK_SLOPE	SATZ_SLOPE Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , Seite 131.
BRN_FILE	reserviert
BREAK	reserviert
CASE	Fallentscheidung, CASE-Anweisung , Seite 82.
CASE_END	
CHAR	RCL-Datentypen , Seite 53.
CHR	Datenkonvertierung, berechnet einen CHAR aus INTEGER MOD 256, CHR-Funktion , Seite 91.

RCL-Schlüsselworte

Schlüsselwort	Bemerkung
CIRCULAR	Interpolationsart , Seite 116.
CLOSE	siehe auch ASSIGN-Funktion , Seite 92.
CLS	reserviert
COMPILER	Arbeitsweise des Compilers , Seite 35.
CONDITION	reserviert
CONST	RCL-Konstantenvereinbarung , Seite 51.
CONT_SLOPE	Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , Seite 131.
CONTROL	reserviert
COS	COS-Funktion , Seite 89.
DEBUGINFO	DEBUGINFO+, DEBUGINFO-, Übersetzungsanweisung, siehe auch Compileranweisungen , Seite 36.
DEF	Punktvariable in Punktetabelle, RCL-Variablendeklaration , Seite 57.
DEFAULT	CASE-Anweisung , Seite 82.
DFACTOR	Bremsverzögerungsfaktor, siehe auch DFACTOR , Seite 127.
ELSE	IF-THEN-(ELSE)-Anweisung , Seite 79.
END	Programmvereinbarung , Seite 41.
END_OF_FILE	EOF-Funktion , Seite 92.
EQUAL	CASE-Anweisung , Seite 82.
ERROR	Maximale Wartezeit , Seite 73.
EVERY	reserviert
EXACT	siehe MOVE_REL EXACT... , Seite 115.
EXCLUSIVE	reserviert
EXCLUSIVE_END	reserviert
EXTERNAL	Hauptprogrammaufruf im Hauptprogramm , Seite 42.
FILE	Dateiarbeit, siehe FILE , Seite 55.
HALT	HALT-Anweisung , Seite 76.
IF	IF-THEN-(ELSE)-Anweisung , Seite 79.
INCLUDE	Einfügen von Dateien , Seite 37.
INPUT	Interface zur SPS, Ausgänge der SPS - Eingänge der Ro-Co, RCL-Variablendeklaration , Seite 57.
INT	Anweisungsübergreifende Interpolationsart , Seite 118.
INT_ASC	Funktion konvertiert einen ganzzahligen Wert in ein Feld von CHAR, INT_ASC-Funktion , Seite 92.
INTEGER	RCL-Datentypen , Seite 53.
J	Bahnruck, siehe auch A , Seite 128.

RCL-Schlüsselworte

Schlüsselwort	Bemerkung
JC (..)	Koordinatentransformation, Umrechnung von Raum- in Maschinen-Koordinaten, JC-Standardfunktion , Seite 95.
JC_NAMES	Übersetzungsanweisungen, Achsnamen (JC name definition) , Seite 37.
JC_POINT	Punktvariablen , Seite 59.
JUMP	JUMP-Anweisung , Seite 77.
KINEMATICS	Übersetzungsanweisung, Kinematikdefinition , Seite 36.
LIMIT_MAX	Arbeitsraumbegrenzungen, LIMIT_MIN , LIMIT_MAX , LIMIT_OFF -Funktionen, Seite 95.
LIMIT_MIN	
LIMIT_OFF	
LINEAR	Interpolationsart , Seite 116.
MAX_TIME	Maximale Wartezeit , Seite 73, siehe auch WAIT.
MCP	reserviert
MOD	Modulo-Division, Arithmetische Ausdrücke , Seite 83.
MOVE	Bewegungsbefehl, MOVE , Seite 112.
MOVE TO	Bewegungsbefehl, MOVE TO ... , Seite 113.
MOVE VIA	Bewegungsbefehl, MOVE VIA ... (Überschleifen), Seite 113.
MOVE VIA..TO	Bewegungsbefehl, MOVE VIA ... TO ... , Seite 113.
MOVE_REL	Bewegungsbefehl, MOVE_REL , Seite 114.
MOVE_REL APPROX	Bewegungsbefehl, MOVE_REL APPROX... , Seite 114.
MOVE_REL EXACT...	Bewegungsbefehl, MOVE_REL EXACT... , Seite 115.
MOVE_REL APPROX.. EXACT	Bewegungsbefehl, MOVE_REL APPROX ... EXACT ... , Seite 115.
NOT	Bitweise Negation, Logische Ausdrücke , Seite 85.
OR	Bitweise ODER-Verknüpfung, Logische Ausdrücke , Seite 85.
ORD	Datenkonvertierung, INTEGER-Wert einer Variablen vom Typ CHAR, ORD-Funktion , Seite 91.
OUTPUT	Interface zur SPS, Ausgänge der RoCo - Eingänge der SPS, RCL-Variablen Deklaration , Seite 57.
PARALLEL	reserviert
PARALLEL_END	reserviert
PAUSE	PAUSE-Anweisung , Seite 75.
PERMANENT	reserviert
PLC	reserviert
PLC_PROCESS	reserviert
PLC_TIME	reserviert

RCL-Schlüsselworte

Schlüsselwort	Bemerkung
POINT	Punktvariablen, Seite 59.
POS	Aktuelle Position in Raum-Koordinaten, Lesen der Ist-Position POS, @POS, Seite 61.
POSE	reserviert
PRIO	reserviert
PROCESS_KIND	reserviert
PROGR_SLOPE	Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE) , Seite 131.
PROGRAM	Programmvereinbarung, Seite 41.
PROGRAM_END	
PTP	Interpolationsart , Seite 116.
PUBLIC	reserviert
R	R, Seite 137.
R_PTP	R_PTP, Seite 137.
READ	siehe auch ASSIGN-Funktion, Seite 92. , FILE, Seite 55.
READ_BEGIN	siehe auch ASSIGN-Funktion, Seite 92.
REAL	Übersicht, Seite 53.
RECORD	Verbundtypen, RECORDs, Seite 56.
RECORD_END	
REF_PNT	reserviert
REPEAT	n: Anzahl der Wiederholungen REPEAT-Anweisung, Seite 76.
REPEAT n TIMES	
REPEAT_END	
RETURN	Unterprogrammvereinbarung, Seite 45.
RHO_FCT	reserviert
SEMAPHORE	reserviert
SIN	SIN-Funktion, Seite 89.
SIZEOF	SIZEOF-Funktion, Seite 94.
SPC_FCT	RCL-Spezialfunktionen, tabellarische Auflistung, Seite 98.
SQRT	SQRT-Wurzel-Funktion, Seite 90.
START	reserviert
STOP	reserviert
SUB_END	Unterprogrammvereinbarung, Seite 45.
SUBROUTINE	
SYNC	SYNC, Seite 129,
SYNCHRON	SYNCHRON, Seite 129,

RCL-Schlüsselworte

Schlüsselwort	Bemerkung
SYNCHRON_END	SYNCHRON_END , Seite 129,
TEXT	Übersicht, Seite 53.
THEN	IF-THEN-(ELSE)-Anweisung , Seite 79.
TIMES	siehe REPEAT-Anweisung , Seite 76.
TO	siehe MOVE TO ... , Seite 113.
TOOL	reserviert
TRUNC	Datenkonvertierung, wandelt das Argument vom Typ REAL durch Abschneiden in einen Wert vom Typ INTEGER, TRUNC-Funktion , Seite 91.
TTY	reserviert
TYPE	Benutzerdefinierte RCL-Typen , Seite 55.
UNTIL	siehe WAIT UNTIL , Abfrage von Kanälen/ Signalen , Seite 68.
V	V , Seite 122.
V_PTP	V_PTP , Seite 122.
VALUE	Unterprogrammaufruf, Call by value, Programmablauf , Seite 46.
VAR	Beginn der Variablendefinition, siehe Einfache RCL-Datentypen , Seite 54.
VERSION	reserviert
VFACTOR	VFACTOR , Seite 124.
VFIX	VFIX , Seite 123.
VIA	siehe MOVE VIA ... (Überschleifen), Seite 113.
WAIT	WAIT-Anweisung , Seite 71.
WAIT UNTIL	
WARNING	reserviert
WC (..)	Koordinatentransformation, Umrechnung von Maschinen- in Raum-Koordinaten, WC-Standardfunktion , Seite 95.
WC_CALCULATION	reserviert
WC_FRAME	reserviert
WC_NAMES	Übersetzungsanweisung, Welt-Koordinatennamen (WC name definition) , Seite 36.
WC_SYSTEM	reserviert
WC_UR	siehe WC_UR , Seite 51.
WITH	Anweisungsbezogene Geschwindigkeitsvorgabe (WITH) , Seite 123.
WRITE	siehe auch ASSIGN-Funktion , Seite 92., FILE , Seite 55.

RCL-Schlüsselworte

Schlüsselwort	Bemerkung
WRITE_BEGIN	siehe auch ASSIGN-Funktion , Seite 92.
WRITE_END	siehe auch ASSIGN-Funktion , Seite 92.

Abb. 19-1: Schlüsselworte für RCL-Programme

RCL-Übersetzeranweisungen

RCL-Übersetzeranweisungen beginnen mit ";;" gefolgt von der eigentlichen Steueranweisung.

Steueranweisung	Bemerkung
CONTROL	reserviert
COMPILER	Allgemeines , Seite 36.
DEBUGINFO	Testinformation , Seite 38.
FILE_ERROR	reserviert
INCLUDE	Einfügen von Dateien , Seite 37.
INT	Anweisungsübergreifende Interpolationsart , Seite 118.
JC_NAMES	Achsnamen (JC name definition) , Seite 37.
KINEMATICS	Kinematikdefinition , Seite 36.
WARNING	reserviert
WC_NAMES	Welt-Koordinatennamen (WC name definition) , Seite 36.

Abb. 19-2: Steueranweisungen für den RCL-Übersetzer

RCL-Standard-Funktionen

Standard-Funktionen	Bemerkung
ABS	Funktion liefert den Betrag einer REAL- oder INTEGER-Zahl, ABS-Funktion , Seite 91.
AND	Bitweise logische UND-Verknüpfung, siehe auch Logische Ausdrücke , Seite 85.
ASC_INT	Datenkonvertierung CHAR nach INTEGER, ASC_INT-Funktion , Seite 93.
ASSIGN	Dateizugriff, ASSIGN-Funktion , Seite 92.
ATAN	ATAN-Funktion , Seite 90.
BNR_FILE	reserviert
CHR	Datenkonvertierung, berechnet einen CHAR aus INTEGER MOD 256, CHR-Funktion , Seite 91.
CONDITION	reserviert
COS	COS-Funktion , Seite 89.
END_OF_FILE	Abfrage, ob das Dateiende beim Lesen einer dat-Datei erreicht ist, EOF-Funktion , Seite 92.
INT_ASC	Funktion konvertiert einen ganzzahligen Wert in ein Feld von CHAR, INT_ASC-Funktion , Seite 92.
JC	Koordinatentransformation, Umrechnung von Raum- in Maschinen-Koordinaten, JC-Standardfunktion , Seite 95.
MOD	Modulo-Division, Arithmetische Ausdrücke , Seite 83.

RCL-Schlüsselworte

Standard-Funktionen	Bemerkung
NOT	Bitweise logische Negation, siehe auch Logische Ausdrücke , Seite 85.
OR	Bitweise logische ODER-Verknüpfung, siehe auch Logische Ausdrücke , Seite 85.
ORD	Datenkonvertierung, INTEGER-Wert einer Variablen vom Typ CHAR, ORD-Funktion , Seite 91.
PLC_PROCESS	reserviert
PLC_TIME	reserviert
ROUND	Wandelt REAL in INTEGER. Rundung erfolgt zur am nächsten liegenden ganzen Zahl. ROUND-Funktion , Seite 91.
SIN	SIN-Funktion , Seite 89.
SIZEOF	Ermittlung des Speicherplatzbedarfs für beliebige RCL-Variablen bzw. Typen, SIZEOF-Funktion , Seite 94.
SQRT	SQRT-Wurzel-Funktion , Seite 90.
TRUNC	Wandelt REAL durch Abschneiden in INTEGER. TRUNC-Funktion , Seite 91.
WC	Koordinatentransformation, Umrechnung von Maschinen- in Raum-Koordinaten, WC-Standardfunktion , Seite 95.
WC_CALCULATION	reserviert

Abb. 19-3: RCL-Standard-Funktionen

RCL-Spezialfunktionen

SPC_FCT	Funktionskurzbeschreibung
1	IOL , Seite 100, (Positiongenaues Schalten digitaler Ausgänge auf der Bahn)
2	PPO , Seite 101, (Positiongenaues Schalten von Prozessdaten auf der Bahn)
21	belt_kind , Seite 102, (Bandart)
23	DateTime , Seite 103, (Systemdatum und Uhrzeit)
24	SysClk , Seite 103, (Systemzähler, realtime counter)
28	set_belt , Seite 104, (Bandzähler Rücksetzwert setzen) siehe SYNC
53	belt_area , Seite 105, (Bandbereich bei Bandart 4)
59	getBlendMode , Seite 108, (Überschleifmodus abfragen)
60	setBlendMode , Seite 107, (Überschleifmodus setzen)

Abb. 19-4: RCL-Spezialfunktionen

RCL-Standard-Konstanten

Standardkonstanten werden zur Zeit nur eingeschränkt verwendet.

RCL-Schlüsselworte

Standard-Konstante	Bemerkung
CLS	reserviert
VERSION	reserviert
WC_UR	siehe WC_UR , Seite 51.

Abb. 19-5: RCL-Standard-Konstanten

RCL-Standard-Variablen

Standard-Variablen	Bemerkung
@	Maschinenkoordinaten, joint coordinate points Siehe JC_POINT in Zusammengesetzte RCL-Datentypen , Seite 55.
@MPOS	reserviert
@POS	aktuelle Position in Maschinen-Koordinaten Lesen der Ist-Position POS, @POS , Seite 61.
A	A , Seite 125.
AFACTOR	AFACTOR , Seite 126.
AFIX	AFIX , Seite 123.
DFACTOR	DFACTOR , Seite 127
J	Bahnrudder, siehe auch A , Seite 128.
LIMIT_MAX	Arbeitsraumbegrenzungen, LIMIT_MIN , LIMIT_MAX , LIMIT_OFF -Funktionen, Seite 95.
LIMIT_MIN	
LIMIT_OFF	
MCP	reserviert
PLC	reserviert
POS	aktuelle Position in Raum-Koordinaten Lesen der Ist-Position POS, @POS , Seite 61.
R	R , Seite 137.
R_PTP	R_PTP , Seite 137.
TTY	reserviert
V	V , Seite 122.
V_PTP	V_PTP , Seite 122.
VFACTOR	VFACTOR , Seite 124.
VFIX	VFIX , Seite 123.
WC_SYSTEM	reserviert

Abb. 19-6: RCL-Standard-Variablen

RCL-Standard-Datentypen

Informationen zu den RCL-Standard-Datentypen finden Sie unter [Übersicht](#), Seite 53.

Standard-Datentypen	Bemerkung
Einfache Datentypen	
BINARY	0, 1

RCL-Schlüsselworte

Standard-Datentypen	Bemerkung
CHAR	Der ASCII (American Standard Code for Information Interchange)-Zeichensatz (7-Bit) wird verwendet, um Buchstaben und Symbole darzustellen, die auf einer US-Standard-Tastatur verwendet werden. Der ASCII-Zeichensatz entspricht den ersten 128 Zeichen (0-127) des ANSI-Zeichensatzes.
INTEGER	Variablen vom Datentyp INTEGER (lange Ganzzahl) werden als 32-Bit-Zahlen (4 Bytes) mit Vorzeichen im Bereich von -2.147.483.648 bis 2.147.483.647 gespeichert.
REAL	Variablen vom Datentyp REAL werden als 32-Bit-Zahlen (4 Bytes) -3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte gespeichert.
Strukturierte Datentypen	
ARRAY	Arbeit mit Feldvariablen (ARRAYs), Seite 62.
BELT	reserviert
FILE	Zusammengesetzte RCL-Datentypen, Seite 55.
JC_POINT	Punktvariablen, Seite 59.
POINT	Punktvariablen, Seite 59.
RECORD RECORD_END	Verbundtypen, RECORDs, Seite 56.
TEXT	Arbeit mit Textvariablen, Seite 62.
WC_FRAME	reserviert

Abb. 19-7: Standard-Datentypen

20 Service und Support

Für Ihre schnelle und optimale Unterstützung verfügen wir über ein dichtes weltweites Servicenetz. Unsere Experten stehen Ihnen mit Rat und Tat zur Seite. Sie erreichen uns täglich **rund um die Uhr - auch am Wochenende und an Feiertagen**.

Service Deutschland Unser technologieorientiertes Competence Center in Lohr deckt alle Belange rund um den Service für elektrische Antriebe und Steuerungen ab.

Die **Service Helpdesk & Hotline** erreichen Sie unter:

Telefon: **+49 (0) 9352 40 50 60**

Fax: **+49 (0) 9352 18 49 41**

E-Mail: service.svc@boschrexroth.de

Internet: <http://www.boschrexroth.com>

Auf unseren Internetseiten finden Sie ergänzende Hinweise zu Service, Reparatur (z.B. Anlieferadressen) und Training.

Service weltweit Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem Ansprechpartner auf. Die Hotline-Rufnummern entnehmen Sie bitte den Vertriebsadressen im Internet.

Vorbereitung der Informationen Wir können Ihnen schnell und effizient helfen, wenn Sie folgende Informationen bereithalten:

- detaillierte Beschreibung der Störung und der Umstände
- Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern
- Ihre Kontaktdaten (Telefon-, Faxnummer und E-Mail-Adresse)

Index

Symbole

@.....	55
@POS.....	61

A

A.....	125
A-0-0028, Fahrbereichsgrenzschalter.....	153
A-0-0029, Positionspolaritäten.....	153
A-0-0030, Positionsgrenzwert, positiv.....	154
A-0-0031, Positionsgrenzwert, negativ.....	154
A-0-0032, Geschwindigkeitsgrenzwert, positiv.....	155
A-0-0033, Geschwindigkeitsgrenzwert, negativ.....	155
A-0-0034, Beschleunigungsgrenzwert, bipolar.....	155
A-0-0036, Ruckgrenzwert, bipolar.....	155
A-0-0038, Drehmoment-/Kraftgrenzwert, bi- polar.....	156
A-0-0048, Geschwindigkeitspolaritäten.....	156
A-0-0049, Drehmoment-/Kraft-Polaritäten.....	157
A-0-0080, Joggen, kurze Distanz.....	158
A-0-0081, Joggen, lange Distanz.....	158
A-0-0082, Joggen, langsame Geschwindigkeit.....	159
A-0-0083, Joggen, schnelle Geschwindigkeit... ..	160
A-0-0084, Joggen, Beschleunigung.....	161
A-0-0085, Joggen, Bremsverzögerung.....	161
A-0-0086, Joggen, Beschleunigungsruck.....	162
A-0-0087, Joggen, Bremsruck.....	162
A-0-0224, Schnellhalt-Verzögerung.....	163
A-0-0225, Schnellhalt-Ruck.....	163
A-0-0230, Überschleifdistanz PTP.....	163
A-0-0240, Feedrate-Override.....	163
A-0-0241, Feedrate-Override-Steuerwort.....	164
A-0-0242, Lageregelung, Aufsynchroisier- beschleunigung.....	164
A-0-0243, Lageregelung, Aufsynchroisier- geschwindigkeit.....	165
Abkürzungen.....	14
ABS.....	91
AFACTOR.....	126
AFIX.....	123
ALIAS-Name.....	55
Allgemeines.....	146
AND.....	86
Anwendung der Koordinatensysteme.....	21
Arithmetische Ausdrücke.....	83
ARRAY.....	57, 62
ASC_INT.....	93
ASSIGN.....	92
ATAN.....	90

B

Bandsynchronisation.....	129
Basis-Koordinatensystem (OCS).....	16
Basis-Koordinatensystem relativ zum Welt- Koordinatensystem.....	21

B

BEGIN.....	41
BELT.....	129
Beschleunigung.....	124
Beschleunigung, wirksame.....	127
Beschleunigungsfaktor.....	126
Beschleunigungsvorgabe.....	124
anweisungsbezogene	125
anweisungsübergreifende	125
Bewegungsbeeinflussende Anweisungen.....	129
Bandsynchronisation	129
Look-Ahead über mehrere Sätze bei PROGR_SLOPE und CONT_SLOPE	136
Satzübergänge (BLOCK_SLOPE, PROGR_SLOPE, CONT_SLOPE)	131
Bewegungsbeeinflussender RCL-Anweisun- gen und Parameter.....	120
Bewegungsbefehle.....	112
Bewegungsoptimierung bei Orientierungs- achsen.....	122
BINARY.....	54
BLOCK_SLOPE.....	132
Bremsverzögerung, wirksam.....	127
Bremsverzögerungsfaktor.....	127

C

C-0-0123, Maximale Anzahl an Kinematiken....	166
C-0-2480, Konfigurierte Kinematikenanzahl....	166
C-0-2482, Liste der nicht verknüpften Achsen..	167
C-0-2483, Liste der konfigurierten Kinematiken	167
C-0-2484, Kinematikkonfigurationsliste.....	167
Call by VALUE.....	42
CASE.....	82
CHAR.....	55
CHR.....	91
CIRCULAR.....	116
CLOSE.....	55
Compileranweisungen.....	36
CONST.....	51
CONT_SLOPE.....	132
COS.....	89

D

Darstellung von Informationen	
Bezeichnungen und Abkürzungen	14
Sicherheitshinweise	13
Verwendete Symbole	14
DEBUGINFO.....	38
DEF.....	57
DEFAULT.....	82
Definition der Koordinatensysteme.....	21
DFACTOR.....	127
Direkte Bewegungsanweisungen.....	111
Bewegungsbefehle	112

Index

D

...Direkte Bewegungsanweisungen	
Interpolationsart	116
Zusatzinformationen	121

E

Einrichtebetrieb.....	149
ELSE.....	79
END_OF_FILE.....	92
EOF.....	92
EQUAL.....	82
err-Datei.....	36
Extension bei Einfügedateien.....	38
EXTERNAL.....	42
EXTERNAL, mit Parameterübergabe.....	43

F

Feldvariablen.....	62
FILE.....	55

G

Gelenk-Koordinatensystem (JCS).....	16
Geschwindigkeit.....	121
Geschwindigkeit, wirksame.....	124
Geschwindigkeitsfaktor	
VFACTOR	123
Geschwindigkeitsvorgabe	
anweisungsbezogen	123
anweisungsübergreifend	122

H

HALT.....	43, 76

I

IF.....	79, 85
INCLUDE.....	37
INPUT.....	57
INT.....	118
INT_ASC.....	92
INTEGER.....	54
Interpolationsart	116
anweisungsbezogene	117
anwendungsübergreifend	118
CIRCULAR	116
LINEAR	116
PTP	117
Ist-Positionsaufnahme.....	61

J

J.....	128
JC.....	61, 95
JC_NAMES.....	37
JC_POINT.....	55, 59
JUMP.....	77

K

K-0-0001, Logische Kinematiknummer.....	168
K-0-0002, Kinematikbezeichnung.....	168
K-0-0003, Maximale Bahngeschwindigkeit.....	168
K-0-0004, reserviert.....	169
K-0-0005, Software-Endschalter, positiv, WC... ..	169
K-0-0006, Software-Endschalter, negativ, WC.	169
K-0-0007, Achsanzahl der Kinematik.....	169
K-0-0008, Achskonfigurationsliste.....	169
K-0-0008 Achszuordnung festlegen.....	221
K-0-0009, Oszi: Liste aller oszilloskopierbaren K-Parameter.....	171
K-0-0010, Liste aller K-Parameter.....	172
K-0-0011, Liste aller ungültigen K-Parameter... ..	172
K-0-0012, Prüfsumme aller K-Parameter.....	172
K-0-0013, IDN-Liste der K-Prüfsummenparameter.....	172
K-0-0014, Bahnbeschleunigung, Änderungszeit.....	173
K-0-0015, Bahnbremsverzögerung, Änderungszeit.....	175
K-0-0016, Projektidentifikationsnummer.....	178
K-0-0020, Kinematikdiagnosemeldung.....	178
K-0-0021, Kinematikstatus.....	179
K-0-0022, Erweiterter Kinematikstatus.....	181
K-0-0023, Kinematikdiagnosenummer.....	183
K-0-0031, Transformationsschema.....	183
K-0-0031 Kinematiktyp festlegen.....	221
K-0-0032, Referenzieren der Achsen, Sollkonfiguration.....	184
K-0-0033, IPOS, Aktualisierung bei Koordinatenwechsel.....	184
K-0-0036, WC-Achsüberwachung.....	184
K-0-0040, Transformation, Längen der Mechanik.....	185
K-0-0041, Transformation, Faktoren.....	185
K-0-0042, Verschiebung des WC-Systems.....	186
K-0-0043, Werkstückkoordinatenzuordnungsliste.....	186
K-0-0050, Distanzeinheiten WC.....	187
K-0-0051, Geschwindigkeitseinheiten WC.....	188
K-0-0052, Beschleunigungseinheiten WC.....	189
K-0-0053, Ruckeinheiten WC.....	189
K-0-0054, Maßeinheit für Kinematikdaten.....	190
K-0-0070, Bandanzahl der Kinematik.....	190
K-0-0071, Kopplungsfaktoren Band 1.....	190
K-0-0072, Kopplungsfaktoren Band 2.....	191
K-0-0073, Kopplungsfaktoren Band 3.....	192
K-0-0074, Kopplungsfaktoren Band 4.....	192
K-0-0080, Joggen, Distanz kurz im WC-System.....	192
K-0-0081, Joggen, Distanz lang im WC-System.....	193
K-0-0082, Joggen, Geschwindigkeit langsam im WC-System.....	194
K-0-0083, Joggen, Geschwindigkeit schnell im WC-System.....	195

K

K-0-0084, Joggen, Beschleunigung im WC-System.....	196
K-0-0085, Joggen, Bremsverzögerung im WC-System.....	196
K-0-0086, Joggen, Beschleunigungsruck im WC-System.....	197
K-0-0087, Joggen, Verzögerungsruck im WC-System.....	197
K-0-0100, Aktuelle Position, Kartesische Originalkoordinaten.....	198
K-0-0102, Aktuelle Bahngeschwindigkeit.....	198
K-0-0103, Aktuelle Bahnbeschleunigung.....	198
K-0-0104, Aktueller Bahnruck.....	199
K-0-0105, Aktuelle Position, Kartesische Werkstückkoordinaten.....	199
K-0-0106, Koordinatenwerte vom Werkzeug....	199
K-0-0107, Koordinatenwerte vom Werkstück. .	199
K-0-0121, Koordinatennamen K1.....	199
K-0-0121 bis K-0-0136, Koordinatennamen . . .	199
K-0-0140, Werkzeugbezeichnung.....	199
K-0-0141, Werkstückbezeichnung.....	199
K-0-0150, Interpolierte Sollposition.....	199
K-0-0152, Interpolierte Bahngeschwindigkeit. .	199
K-0-0153, Interpolierte Bahnbeschleunigung....	200
K-0-0154, Interpolierter Bahnruck.....	200
K-0-0155, Schutzzonen ein/aus.....	200
K-0-0156, Schutzzone1 min. Punkt.....	200
K-0-0156/K-0-0157 Schutzzone1 min./max. Punkt bis K-0-0218/K-0-0219 Schutzzone32 min./max. Punkt	201
K-0-0157, Schutzzone1 max. Punkt.....	200
K-0-0220, Dateitransfername.....	202
K-0-0221, Name der zu kompilierenden Datei..	203
K-0-0240, Feedrate-Override.....	203
K-0-0241, Feedrate-Override-Steuerwort.....	203
K-0-0242, Acceleration-Override.....	204
K-0-0243, Deceleration-Override.....	204
K-0-1060, Kommando: Lade Datei von CF.....	204
K-0-1061, Kommando: Speichere Datei auf CF.....	204
K-0-1062, Kommando: Compiliere Datei auf MLC.....	205
K-0-2200, Programmierte Position 1 (aktiv).....	205
K-0-2201, Programmierte Position 2 (aktiv).....	205
K-0-2202, Typ der aktiven programmierten Position.....	205
K-0-2203, Interpolationstyp.....	205
K-0-2204, Aktuell aktiver QLL-Dateiname.....	205
K-0-2205, Aktuell aktive Zeilennummer.....	205
K-0-2300, SPS-Bit-Eingang (RCL).....	205
K-0-2301, SPS-Ganzzahl-Eingang (RCL).....	205
K-0-2302, SPS-Gleitkommazahl-Eingang (RCL).....	205
Kanalvereinbarung.....	67
KINEMATICS.....	36
Kinematik-Benutzeroberfläche.....	207

K

...Kinematik-Benutzeroberfläche	
Allgemeines	207
Compiler	214
Debugger	215
RCL-Editor	209
Robot Control Menüstruktur	208
Kommunikationskanäle zur SPS.....	64
Koordinaten.....	15
Koordinatensysteme.....	15

L

LIMIT_MAX.....	95
LIMIT_MIN.....	95
LIMIT_OFF.....	95
LINEAR.....	116
Linear-Überschleifen bei BLOCK_SLOPE.....	144
Linear-Überschleifen bei PROGR_SLOPE und CONT_SLOPE.....	140
Logische Ausdrücke.....	85
Look-Ahead über mehrere Sätze bei PROGR_SLOPE und CONT_SLOPE.....	136

M

Main program	
Anweisungsteil	40
Aufbau	39
Deklarationsteil	39
MAX_TIME.....	73
MOD.....	84
MOVE.....	112
MOVE_REL	114
MOVE_REL APPROX.....	114
MOVE_REL APPROX ... EXACT	115
MOVE_REL EXACT.....	115
MOVE TO	113
MOVE VIA	113
MOVE VIA ... TO	113

N

NOT.....	86

O

OPEN.....	55
Operatoren	
arithmetische	83
logische	85
OR.....	86
ORD.....	91
OUTPUT.....	57
Override.....	123
Override deaktivieren.....	123

P

PAUSE.....	75

Index

P

pkt-Datei.....	36
POINT.....	55, 59
POS.....	61
Positionsgenaues Schalten.....	99
Bandart	102
bandsynchroner Arbeitsbereich der	
Bandart 4	105
Bandzähler setzen	104
DateTime	103
digitaler Ausgänge auf der Bahn	100
Schalten von Prozessdaten auf der bahn ..	101
SysClk	103
PROGR_SLOPE.....	132
PROGRAM.....	41
PROGRAM_ END.....	41
PTP.....	117
PTP Überschleifen bei BLOCK_SLOPE.....	142
PTP-Überschleifen bei PROGR_SLOPE.....	138
Punkte und Punktedatei pkt.....	59
Punktvariable.....	59
Punktvariablen	
Vereinbarung	59
Vollständige Wertzuweisung	60
Wertzuweisung für Einzelkomponenten	60

Q

qll-Datei.....	35
Quelldatei.....	35

R

R.....	137
R_PTP.....	137
Rangfolge der Operationen.....	86
Räumliches Überschleifen.....	137
Überblick	137
Räumliches Überschleifen bei BLOCK_SLOPE.....	142
Räumliches Überschleifen bei	
PROGR_SLOPE und CONT_SLOPE.....	138
RCL	
Programmstruktur	35
RCL-Bewegungsanweisungen.....	111
bewegungsbeeinflussende	129
direkte	111
RCL-Bibliotheksfunktionen.....	26
RCL-Datentypen.....	53, 94
benutzerdefiniert	55
einfache	54
zusammengesetzte	55
RCL-Debugger.....	215
Programme debuggen	216
Setzen und Löschen von Haltepunkten	215
Variablen überwachen	217
RCL-Editor.....	209
Ausschneiden, Kopieren, Einfügen	211

R

...RCL-Editor	
Editorfunktionen	210
Kommentare	211
Lesezeichen	211
Merkmale	209
Neues Programm erstellen	210
Optionen	210
Suchen und Ersetzen	211
Unterstützung der Quelltexteingabe	212
RCL-Funktionen.....	89
ABS-Funktion	91
ASC_INT-Funktion	93
ASC-Funktion	92
ASSIGN-Funktion	92
ATAN-Funktion	90
CHR-Funktion	91
COS-Funktion	89
EOF-Funktion	92
JC-Funktion	95
LIMIT_MAX-Funktion	95
LIMIT_MIN-Funktion	95
LIMIT_OFF-Funktion	95
ORD-Funktion	91
ROUND-Funktion	91
SIN-Funktion	89
SIZEOF-Funktion	94
SQRT-Wurzel---Funktion	90
TRUNC-Funktion	91
WC-Funktion	95
RCL-Konstanten.....	51
Konstantenvereinbarungen	51
vordefiniert	51
RCL-Koordinatensysteme	
Basis-Koordinatensystem (OCS)	16
Gelenk-Koordinatensystem (JCS)	16
Joint Coordinate System (JCS)	16
Original Coordinate System (OCS)	16
Product Coordinate System (WpCS)	19
Welt-Koordinatensystem (WCS)	18
Werkstück-Koordinatensystem (WpCS)	19
World Coordinate System (WCS)	18
RCL-Parameter.....	153
achsbezogen	153
C-Paramter aller Kinematiken	166
Paramter einer Kinematik	168
RCL-Programmaufruf	
externe Programme	42
Hauptprogramm	42
Programmablauf	46
Unterprogramm	45
RCL-Programmierung	
Positionsgenaues Schalten	99
Überschleifmodus	107
RCL-Programmsteuerung.....	71
CASE-Anweisung	82
HALT-Anweisung	76

R

...RCL-Programmsteuerung	
IF-THEN--ELSE)-Anweisung	79
JUMP-Anweisung	77
PAUSE-Anweisung	75
REPEAT-Anweisung	76
WAIT-Anweisung	71
RCL-Programmstruktur	35
Compileranweisungen	36
Hauptprogramm-Aufbau	39
Hauptprogrammaufruf	42
Programmablauf	46
Programmvereinbarung	41
Übersetzer	35
Unterprogrammvereinbarung	45
RCL-Schlüsselworte	281
RCL-Spezialfunktionen	97
Aufruf	98
Positionsgenaueres Schalten	99
RCL-Variablen	53
Feldvariablen	62
Kommunikationskanäle zur SPS	64
Punkte und Punktedatei "pkt"	59
RCL-Datentypen	53
RCL-Punktevariablen	59
RCL-Variablendeklaration	57
Textvariablen	62
RCL-Variablendeklaration	57
READ	55
REAL	54
RECORD	56
RECORD_END	56
REPEAT	63, 76
REPEAT_END	48, 76
REPEAT n TIMES	48, 76
RETURN	45
Robot Control	
Compiler	214
Debugger	215
Editor	209
Menü Kinematik	208
Menü Prog.QLL	209
Menü Programme	208
Menü Robot	208
Menüstruktur	208
RoCo-Multikinematik	219
Anlegen von Kinematiken	219
Bewegungssteuerung	224
Festlegung der Achszuordnung	221
Festlegung des Kinematiktyps	221
Löschen von Kinematiken	223
PLC-Shared-Memory-Interfaces (KinData)	228
Punktetabellen	226
Punktetabellen, Zuordnung und Verhalten	227
RCL-Programme	225
Überblick	219
RoCo Programmierumgebung	207

R

ROUND	91
Ruck	128
Ruckvorgabe	128
anweisungsbezogene	128
anweisungsübergreifende	128

S

Satzübergänge (Slope-Arten)	131
SIN	89
SIZEOF	94
Späte Bewegungsbefehle	
Spätes Kommandieren	136
Spätes Kommandieren	136
Spätes Überschleifen	144
SPC_FCT	
tabellarische Auflistung	98
SPC_FCT: 1	
Positionsgenaueres Schalten digitaler Aus- gänge auf der Bahn	100
SPC_FCT: 2	
Positionsgenaueres Schalten von Prozess- daten auf der Bahn	101
SPC_FCT: 21	
Bandart	102
SPC_FCT: 23	
DateTime	103
SPC_FCT: 24	
SysClk	103
SPC_FCT: 28	
Bandzähler setzen	104
SPC_FCT: 53	
bandsynchroner Arbeitsbereich der Bandart 4	105
SPC_FCT: 59	
Überschleifmodus abfragen	108
SPC_FCT: 60	
Überschleifmodus setzen	107
Sprungziel	77
SQRT	90
SUB_END	45
SUBROUTINE	45
Support	
siehe Service-Hotline	291
sym-Datei	36
SYNC	129
SYNCHRON	129
SYNCHRON_END	129

T

TEXT	55
Textvariablen	62
THEN	79
Tippbetrieb	149
Tippen	
Achskoordinaten	150

Index

T

...Tippen	
Gelenk-Koordinaten	150
Weltkoordinaten	151, 152
Tippen in unterschiedlichen Koordinatensystemen.....	150
Transformationsvorschrift	
Robotertyp 00, Karteser	233
Robotertyp 07, 2-Achs-Delta mit optional 1 Handachse	237
Robotertyp 09, 3-Achs-Scara mit optional 1 Handachse	240
Robotertyp 10, 4-Achs-Scara mit optional 1 Handachse	242
Robotertyp 100, SPS	279
Robotertyp 12, 2-Achs-Parallelogramm	245
Robotertyp 13, 2-Achs-Parallelogramm	247
Robotertyp 18, Umschlingungskinematik	249
Robotertyp 19, 2-Achs-Parallelogramm V1	250
Robotertyp 20, 3-Achs-Delta mit optional 1 Handachse	252
Robotertyp 21, 2-Achs-Parallelogramm V2	257
Robotertyp 29, 2-Achs-Delta mit rotatorischer und translatorischer Achse	272
Robotertyp 30, Adabot	275
Robotertyp 32, 2-Achs-Delta mit asymmetrischem TCP	278
Überblick	233
Winkelhand-Roboter mit 6 (7) Achsen	259
Winkelhand-Roboter mit 6 Achsen + 3 Linearachsen	265
TRUNC.....	91
TYPE.....	55

U

Überblick	
Bewegungsbeeinflussender RCL-Anweisungen und Parameter	120
Verfahrensbereichsbegrenzungen	33
Überschleifen bei Wechsel der Interpolationsart.....	142, 144
Überschleifmodus.....	146
RCL-Programmierung	107
Übersetzer - Arbeitsweise.....	35
UNTIL.....	85

V

V.....	122
V_PTP.....	122
VALUE.....	46
VAR.....	54
Verbundtyp.....	56
Verfahrensbereichsbegrenzungen.....	33
Vergleichsoperationen.....	85
Verknüpfungen.....	83
Arithmetische Ausdrücke	83

V

...Verknüpfungen	
Logische Ausdrücke	85
Rangfolge der Operationen	86
Vergleichsoperationen	85
VFACTOR.....	124
VFIX.....	123
VFIX_PTP.....	123

W

WAIT.....	71
WC.....	61, 95
WC_FRAME.....	51
WC_NAMES.....	36
WC_SYSTEM.....	51
WC_UR.....	51
Welt-Koordinatensystem	
Basis-Koordinatensystem	21
Werkstück-Koordinatensystem	23
Welt-Koordinatensystem (WCS).....	18
Werkstückkoordinaten-Datei.....	25
Werkstückkoordinaten-Editor.....	25
Werkstück-Koordinatensystem (WpCS).....	19
Werkstück-Koordinatensystem relativ zum Welt-Koordinatensystem.....	23
Wertzuweisungen.....	83
WITH.....	123
WRITE.....	55

Z

Zielpunktbezeichnungen	116
Zu dieser Dokumentation.....	9
Darstellung von Informationen	13
Gültigkeit der Dokumentation	9
Zusatzinformationen.....	121
Anweisungsbezogene Beschleunigungsvorgabe (WITH)	125
Anweisungsbezogene Geschwindigkeitsvorgabe (WITH)	123
Anweisungsbezogene Ruckvorgabe (WITH)	128
Anweisungsübergreifende Beschleunigungsvorgabe (Globale Vorgabe)	125
Anweisungsübergreifende Geschwindigkeitsvorgabe	122
Anweisungsübergreifende Ruckvorgabe (Globale Vorgabe)	128
Beschleunigung	124
Beschleunigungsfaktor	126
Bewegungsoptimierung bei Orientierungsachsen	122
Bremsverzögerungsfaktor	127
Geschwindigkeit	121
Geschwindigkeitsfaktor	123
Override	123
Override deaktivieren	123

Index

Z

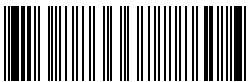
...Zusatzinformationen	
Ruck	128
Wirksame Beschleunigung	127
Wirksame Bremsverzögerung	127

Z

...Zusatzinformationen	
Wirksame Geschwindigkeit	124
Zwischenpunkt.....	116

Notizen

Bosch Rexroth AG
Electric Drives and Controls
Postfach 13 57
97803 Lohr, Deutschland
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Deutschland
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911333851

DOK-MLC***-RCL*PRO*V12-AP02-DE-P