# DOKUMENTATION ISG-kernel

# Funktionsbeschreibung
# Universelle Kinematik

Kurzbezeichnung:
FCT-C27

© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
Alle Rechte vorbehalten
www.isg-stuttgart.de
support@isg-stuttgart.de

Dokumentation Version: 1.181
12.11.2024

# Vorwort

## Rechtliche Hinweise

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte und der Funktionsumfang werden jedoch ständig weiterentwickelt. Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

## Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen, der zugehörigen Dokumentation und der Aufgabenstellung vertraut ist.

Zur Installation und Inbetriebnahme ist die Beachtung der Dokumentation, der nachfolgenden Hinweise und Erklärungen unbedingt notwendig. Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zum betreffenden Zeitpunkt veröffentlichte Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbarer Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

## Weiterführende Informationen

Unter den Links (DE)

https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

bzw. (EN)

https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

finden Sie neben der aktuellen Dokumentation weiterführende Informationen zu Meldungen aus dem NC-Kern, Onlinehilfen, SPS-Bibliotheken, Tools usw.

## Haftungsausschluss

Änderungen der Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig.

## Marken und Patente

Der Name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® und entsprechende Logos sind eingetragene und lizenzierte Marken der ISG Industrielle Steuerungstechnik GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltene Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

## Copyright

© ISG Industrielle Steuerungstechnik GmbH, Stuttgart, Deutschland.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster oder Geschmacksmustereintragung vorbehalten.

# Allgemeine- und Sicherheitshinweise

## Verwendete Symbole und ihre Bedeutung

In der vorliegenden Dokumentation werden die folgenden Symbole mit nebenstehendem Sicherheitshinweis und Text verwendet. Die (Sicherheits-) Hinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

## Symbole im Erklärtext

➢ Gibt eine Aktion an.

⇨ Gibt eine Handlungsanweisung an.

**⚠ GEFAHR**

**Akute Verletzungsgefahr!**

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

**⚠ VORSICHT**

**Schädigung von Personen und Maschinen!**

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen und Maschinen geschädigt werden!

**Achtung**

**Einschränkung oder Fehler**

Dieses Symbol beschreibt Einschränkungen oder warnt vor Fehlern.

**Hinweis**

**Tipps und weitere Hinweise**

Dieses Symbol kennzeichnet Informationen, die zum grundsätzlichen Verständnis beitragen oder zusätzliche Hinweise geben.

**Beispiel**

**Allgemeines Beispiel**

Beispiel zu einem erklärten Sachverhalt.


**Programmierbeispiel**

**NC-Programmierbeispiel**

Programmierbeispiel (komplettes NC-Programm oder Programmsequenz) der beschriebenen Funktionalität bzw. des entsprechenden NC-Befehls.


**Versionshinweis**

**Spezifischer Versionshinweis**

Optionale, ggf. auch eingeschränkte Funktionalität. Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

# Inhaltsverzeichnis

**Vorwort** ..... **2**

**Allgemeine- und Sicherheitshinweise** ..... **3**

**1 Übersicht** ..... **6**

**2 Beschreibung** ..... **7**

**3 Konfiguration der Kinematik** ..... **8**

3.1 Beschreibung von Flansch und Werkzeug ..... 8
3.2 Programmierung bzw. Beschreibung der kinematischen Kette ..... 10
3.3 Einstellung des Programmiermodus ..... 12
3.4 Einstellung der Winkeltransformation ..... 13
3.5 Hinweise zur Parametrierung ..... 14

**4 Kinematische Kette** ..... **16**

4.1 Achsreihenfolge in der kinematischen Kette ..... 16
4.1.1 1. Beispiel anhand einer CA-Maschine ..... 17
4.1.2 2. Beispiel anhand einer AC-Maschine ..... 19
4.1.3 3. Beispiel anhand einer AC-Maschine mit Kardan-Tisch ..... 21
4.2 Einschränkungen in der kinematischen Kette ..... 24
4.3 Testen der Konfiguration in der Simulation ..... 25

**5 Programmiermodi** ..... **28**

5.1 Punkt-Vektor-Programmierung ..... 28
5.2 Winkelprogrammierung ..... 29
5.2.1 Klassische Programmiermodi ..... 29
5.2.2 Konforme und Direkte Programmierung ..... 30

**6 Korrektur von Achsfehlstellungen** ..... **31**

**7 Posenwechsel vermeiden** ..... **32**

**8 Bestehende Kinematiken nachbilden** ..... **33**

**9 Transformation zw. Achswerten und kartesischen Koordinaten** ..... **35**

**10 Weitere Beispiele** ..... **38**

10.1 Realisierung CB Kopfkinematik ..... 38

**11 Parameter** ..... **40**

11.1 Übersicht ..... 40
11.2 Beschreibung ..... 41

**12 Anhang** ..... **48**

12.1 Anregungen, Korrekturen und neueste Dokumentation ..... 48

**Stichwortverzeichnis** ..... **49**

# Abbildungsverzeichnis

Abb. 1: Achsreihenfolge bei der CA-Maschine ..... 17
Abb. 2: Kinematische Struktur der 5-achsigen Maschine mit AC Werkstücktisch ..... 19
Abb. 3: Kinematische Struktur der 5-achsigen Maschine mit Kardantisch ..... 21
Abb. 4: Detailansicht des Werkzeugtisches ..... 22
Abb. 5: CB Kopfkinematik ..... 38

# 1 Übersicht

## Aufgabe

Die "Universelle Kinematik" hat die **ID91** und ist eine kinematische Transformation.

Die freie Konfigurierbarkeit ermöglicht es, neue kinematische Transformationen zu erstellen, ohne die ISG Kinematikbibliothek ([KITRA]) erweitern zu müssen.

## Eigenschaften

Die Kinematik orientiert sich nicht an einer speziellen Maschine, sondern ermöglicht durch freie Konfigurierbarkeit die Nachbildung von Maschinen, die durch eine kinematische Kette beschrieben werden können (serielle Kinematiken).

Darunter fallen die klassischen 5-Achs-Maschinen (CA-Maschine, 45°-BA-Maschine usw.), aber auch 3- und 4-Achs-Bearbeitung ist möglich.

## Parametrierung

Die Parametrierung ist vom Aufbau der Maschine abhängig und daher sehr individuell. Sie wird im Kapitel „Konfiguration der Kinematik“ [▶ 8] näher beschrieben.

Die Einstellung des Programmiermodus wird über den Kanalparameter P-CHAN-00112 oder über die Kinematik selbst vorgenommen. Eine Einstellung an der Kinematik hat Priorität über P-CHAN-00112.

## Programmierung

Es werden 2 Programmiermodi unterschieden:

1. Punkt-Vektor-Programmierung
2. Winkelprogrammierung (klassisch, konform und direkt)

**Hinweis**

**Transformationen sind eine lizenzpflichtige Zusatzoption.**

## *Obligatorischer Hinweis zu Verweisen auf andere Dokumente*

Zwecks Übersichtlichkeit wird eine verkürzte Darstellung der Verweise (Links) auf andere Dokumente bzw. Parameter gewählt, z.B. [PROG] für Programmieranleitung oder P-AXIS-00001 für einen Achsparameter.

Technisch bedingt funktionieren diese Verweise nur in der Online-Hilfe (HTML5, CHM), allerdings nicht in PDF-Dateien, da PDF keine dokumentenübergreifenden Verlinkungen unterstützt.

# 2 Beschreibung

## Konzepte

Die Kinematik unterstützt 2 verschiedene Programmiermodi:

1. Punkt-Vektor-Modus (einfacher). Der Anwender programmiert Orientierung und Position des Werkzeuges direkt.
2. In allen anderen Modi werden Winkel programmiert. Die Kinematik interpretiert diese Winkel anhand des gegebenen Programmiermodus und berechnet zusammen mit den kartesischen Werten die Orientierung und Position des Werkzeuges. Der Anwender kann nun entscheiden, auf welche Weise die Achswinkel bestimmt werden sollen:

| Achswinkelberechnung | Beschreibung |
|---|---|
| RTCP (Rotation Tool Center Point) / unvollständig | Die programmierten Winkel werden einfach als Achswinkel übernommen |
| Vollständig | Achswinkel werden wie beim Punkt-Vektor-Modus aus der berechneten Orientierung bestimmt. |

## Anwendungen

Da für jede Achse Orientierung und Stützpunkt explizit angegeben werden, können auch Maschinen mit ungewöhnlichen Achsstellungen abgebildet werden (45°-Winkel, 30°-Winkel usw.).

Aus demselben Grund ist es möglich, maschinenbedingte Fehlstellungen der Achsen zu kompensieren. Konfiguriert man die Kinematik entsprechend, dann berechnet die Kinematik korrigierende Achswerte.

Da auch die Reihenfolge der Achsen in der kinematischen Kette frei konfigurierbar ist, können z.B. Rundachsen im Werkzeugtisch abgebildet werden.

# 3 Konfiguration der Kinematik

## Übersicht

Die Kinematik wird so konfiguriert, dass sie eine klassische CA-Maschine mit den Achsen XYZ-CA nachbildet. Die Konfiguration der Kinematik erfolgt in drei Schritten:

**Schritt 1: Kinematische Kette**

- Nullstellung des Werkzeugs
- Beschreibung der beteiligten Achsen
- Reihenfolge der Achsen in der kinematischen Kette

**Schritt 2: Programmiermodus**

- Punkt-Vektor, CA, BA, …

**Schritt 3: Winkeltransformation**

- Unvollständige (RTCP = Rotation Tool Center Point, d.h. Drehwinkel werden durch die Transformation nicht abgebildet) oder
- vollständige Transformation.

**Hinweis**

Jede der Komponenten kann über Kanal-Listendateien (z.B. default_sda.lis) oder über Variablen im NC-Code eingestellt werden.

In den Beispielen dieser Dokumentation werden Listen verwendet.

## 3.1 Beschreibung von Flansch und Werkzeug

Mit den Parametern *zero_position* (P-CHAN-00286) und *zero_orientation* (P-CHAN-00285) können die Nullstellung bzw. die Richtung des Werkzeugs zur Nullstellung der Maschine angegeben werden.

### Nullstellung der Maschine

Über P-CHAN-00286 kann die Position des Flanschs in der Nullstellung der Maschine angegeben werden.

**Beispiel**

**Parametrierung der Nullposition des Flanschs**

```
# Flansch ruht im Punkt (12000, -3200, 500)
kinematik[91].zero_position[0]    12000
kinematik[91].zero_position[1]    -3200
kinematik[91].zero_position[2]    500
```

## Richtung des Werkzeugs zur Nullstellung

Über P-CHAN-00285 kann die Richtung des Werkzeuges in der Nullstellung der Maschine angegeben werden.

Der Parameter *zero_orientation* ist nur dann wirkt nur dann auf die Kinematik, wenn eine Werkzeuglänge benutzt wird. In der Nullstellung der Maschine wird die Position des TCP wie folgt berechnet.

```
TCP = zero_position - Werkzeuglänge * zero_orientation
```

**Beispiel**

**Null-Orientierung des Werkzeuges**

```
# Werkzeug zeigt in Z-Richtung
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1
```

**Beispiel**

**Null-Orientierung des Werkzeuges mit 45 Grad Winkel**

```
# Werkzeug steht im 45°-Winkel zu Y- und Z-Achse
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    1
kinematik[91].zero_orientation[2]    1
```

Der Parameter *zero_orientation* (P-CHAN-00285) muss nicht als Einheitsvektor der Länge 1 angegeben werden. Er wird beim Einlesen automatisch normiert. D.h., das vorangegangene Beispiel hat die gleich Null-Orientierung wie das nachfolgende Beispiel.

**Beispiel**

**Null-Orientierung des Werkzeuges mit 45 Grad Winkel- ohne Normierung**

```
# Werkzeug steht im 45°-Winkel zu Y- und Z-Achse
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0.707
kinematik[91].zero_orientation[2]    0.707
```

## 3.2 Programmierung bzw. Beschreibung der kinematischen Kette

### Hinweis

Alle Komponenten der Kinematik werden im Programmierkoordinatensystem (PCS) beschrieben.


### Versionshinweis

In dieser Dokumentation wird in Listendateien die Syntax der CNC-Versionen V2.11.20xx und V2.11.28xx verwendet.

Diese Syntax hat sich für Transformationskonfigurationen ab V3.00 geändert.

Für CNC-Version > V3.00 muss P-CHAN-00262 [▶ 43] zwingend der Transformations-ID 91 belegt werden.

| Alte Syntax: für CNC Versionen V2.11.20xx und V2.11.28xx | Neue Syntax: für CNC Versionen ab V3.00 |
|---|---|
| kinematik[91].zero_orientation[0] 0<br>kinematik[91].zero_orientation[1] 0<br>… | trafo[0].id 91<br>trafo[0].zero_orientation[0] 0<br>trafo[0].zero_orientation[1] 0<br>… |

### Achsanzahl

Die Anzahl der Achsen wird angegeben mit:

```
# typische CA-Maschine: XYZCA
kinematik[91].number_of_axes          5
```

### Achsen

Jede der Achsen ist durch folgende Kenngrößen definiert:

| Feld | Beschreibung |
|---|---|
| Typ | Linearachse (1) oder Rundachse (2), siehe P-AXIS-00018 |
| Orientierung | Richtungsvektor der Achse, nicht Nullvektor |
| Punkt | Stützpunkt, nur relevant für Rundachsen |

## Die einzelnen Achsen werden angegeben mit:

```
# X-Achse definieren
kinematik[91].axis[0].type              1
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0

# Y-Achse definieren
…
# Z-Achse definieren
…
# C-Achse definieren
# zeigt in Z-Richtung und läuft durch
# den Punkt (800, 1200, 0)
kinematik[91].axis[3].type              2
kinematik[91].axis[3].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    1
kinematik[91].axis[3].point[0]          800
kinematik[91].axis[3].point[1]          1200
kinematik[91].axis[3].point[2]          0

# A-Achse definieren
…
```

## Achsreihenfolge

Die Reihenfolge der Achsen in der kinematischen Kette muss angegeben werden. Diese Reihenfolge kann, aber muss nicht, mit der Reihenfolge der Achsdefinitionen übereinstimmen. Damit ist es z.B. möglich, Rundachsen an den Anfang der kinematischen Kette zu stellen und so eine Drehachse im Werkstücktisch nachzubilden. Siehe Kapitel Einstellung des Programmiermodus [▶ 12].

```
kinematik[91].chain[0]                  0
kinematik[91].chain[1]                  1
kinematik[91].chain[2]                  2
kinematik[91].chain[3]                  3
kinematik[91].chain[4]                  4
```

Dabei bedeutet „chain[i] = j“, dass die i-te Position in der kinematischen Kette mit der j-ten Achse belegt ist.

**Achtung**

Die Universelle Kinematik (ID91) darf nur eingeschaltet sein, wenn alle beteiligten Achsen im Kanal vorhanden sind, ansonsten wird ein Fehler ausgegeben.

# 3.3 Einstellung des Programmiermodus

## Erklärung

Der Programmiermodus gibt an, wie die Orientierung des Werkzeuges aus den programmierten Werten bestimmt wird.

- Im Punkt-Vektor-Modus programmiert der Anwender direkt Position und Orientierung des Werkzeuges.
- In allen anderen Modi (Winkelprogrammiermodus) programmiert der Anwender Winkel, die dann direkt als Achswinkel weitergegeben (RTCP = Rotation Tool Center Point) oder zur Berechnung der Orientierung des Werkzeuges herangezogen werden.

## Programmiermodus

Die Einstellung des Programmiermodus wird über den Kanalparameter P-CHAN-00112 oder über die Kinematik (P-CHAN-00288 [▶ 45]) selbst vorgenommen. Eine Einstellung an der Kinematik hat Priorität über P-CHAN-00112.

Es wird der klassische Programmiermodus für CA-Winkel verwendet.

```
# Programmiermodus CA einstellen über P-CHAN-00112
ori_rotation_angle                      17

# Programmiermodus CA an Kinematik einstellen über P-CHAN-00288 [▶ 45]
kinematik[91].programming_mode          17
```

# 3.4 Einstellung der Winkeltransformation

## Beschreibung

In einem Winkelprogrammiermodus (alle außer Punkt-Vektor) können die programmierten Winkel auf 2 Arten behandelt werden:

1. Bei der RTCP-Transformation (unvollständig) werden die programmierten Winkel direkt weitergegeben, ohne dass eine Umrechnung stattfindet.
   Dieser RTCP-Modus ist bei den meisten Transformationen der Kinematikbibliothek ([KITRA]) Standard.
2. Bei der vollständigen Transformation interpretiert die Kinematik die programmierten Winkel anhand des Programmiermodus und bestimmt daraus die Orientierung des Werkzeuges. Aus dieser Orientierung werden dann die Achswinkel berechnet.

Die Einstellung kann über P-CHAN-00287 [▶ 44] erfolgen.

**Hinweis**

Im RTCP-Modus werden die programmierten Winkel an die Maschine weitergegeben.

Bei der vollständigen Transformation werden Winkel im Bereich (-π, π] an die Maschine weitergegeben.

**Achtung**

Bei der vollständigen Transformation können die Achswinkel sehr stark von den programmierten Winkeln abweichen. Dies gilt insbesondere, wenn Rundachsen nicht am Ende der kinematischen Kette sondern z.B. im Werkstück liegen.

## Winkeltransformation

Es wird der RTCP-Modus verwendet.

```
# RTCP-Modus einschalten
kinematik[91].rtcp                    1
```

## 3.5 Hinweise zur Parametrierung

Die Parametrierung der Universalkinematik kann über

- Listen-Parameter: kinematik[91].param[i] bzw. trafo[j].param[i]
- V.G.-Variablen: V.G.KIN[91].PARAM[i] / V.KIN[91].ZERO…
- Werkzeugversätze, Werkzeugdatenbank

erfolgen.

**Achtung**

**Alle unterschiedlichen Parametriermöglichkeiten nutzen dieselbe Speicherstelle. Bei Schreib- und Lesezugriffen ist dies zwingend zu beachten.**

Nachfolgend ist exemplarisch eine Gegenüberstellung der beiden Parametriermöglichkeiten über V.G.-Variablen.

```
V.G.KIN[91].ZERO_ORIENTATION[0]       =    V.G.KIN[91].PARAM[0]
V.G.KIN[91].ZERO_ORIENTATION[1]       =    V.G.KIN[91].PARAM[1]
V.G.KIN[91].ZERO_ORIENTATION[2]       =    V.G.KIN[91].PARAM[2]
V.G.KIN[91].ZERO_POSITION[0]          =    V.G.KIN[91].PARAM[3]
V.G.KIN[91].ZERO_POSITION[1]          =    V.G.KIN[91].PARAM[4]
V.G.KIN[91].ZERO_POSITION[2]          =    V.G.KIN[91].PARAM[5]
V.G.KIN[91].NUMBER_OF_AXES            =    V.G.KIN[91].PARAM[6]
V.G.KIN[91].AXIS[0].TYPE              =    V.G.KIN[91].PARAM[7]
V.G.KIN[91].AXIS[0].ORIENTATION[0]    =    V.G.KIN[91].PARAM[8]
V.G.KIN[91].AXIS[0].ORIENTATION[1]    =    V.G.KIN[91].PARAM[9]
V.G.KIN[91].AXIS[0].ORIENTATION[2]    =    V.G.KIN[91].PARAM[10]
V.G.KIN[91].AXIS[0].POINT[0]          =    V.G.KIN[91].PARAM[11]
V.G.KIN[91].AXIS[0].POINT[1]          =    V.G.KIN[91].PARAM[12]
V.G.KIN[91].AXIS[0].POINT[2]          =    V.G.KIN[91].PARAM[13]
V.G.KIN[91].AXIS[1].TYPE              =    V.G.KIN[91].PARAM[14]
V.G.KIN[91].AXIS[1].ORIENTATION[0]    =    V.G.KIN[91].PARAM[15]
V.G.KIN[91].AXIS[1].ORIENTATION[1]    =    V.G.KIN[91].PARAM[16]
V.G.KIN[91].AXIS[1].ORIENTATION[2]    =    V.G.KIN[91].PARAM[17]
V.G.KIN[91].AXIS[1].POINT[0]          =    V.G.KIN[91].PARAM[18]
V.G.KIN[91].AXIS[1].POINT[1]          =    V.G.KIN[91].PARAM[19]
V.G.KIN[91].AXIS[1].POINT[2]          =    V.G.KIN[91].PARAM[20]
V.G.KIN[91].AXIS[2].TYPE              =    V.G.KIN[91].PARAM[21]
V.G.KIN[91].AXIS[2].ORIENTATION[0]    =    V.G.KIN[91].PARAM[22]
V.G.KIN[91].AXIS[2].ORIENTATION[1]    =    V.G.KIN[91].PARAM[23]
V.G.KIN[91].AXIS[2].ORIENTATION[2]    =    V.G.KIN[91].PARAM[24]
V.G.KIN[91].AXIS[2].POINT[0]          =    V.G.KIN[91].PARAM[25]
V.G.KIN[91].AXIS[2].POINT[1]          =    V.G.KIN[91].PARAM[26]
V.G.KIN[91].AXIS[2].POINT[2]          =    V.G.KIN[91].PARAM[27]
V.G.KIN[91].AXIS[3].TYPE              =    V.G.KIN[91].PARAM[28]
V.G.KIN[91].AXIS[3].ORIENTATION[0]    =    V.G.KIN[91].PARAM[29]
V.G.KIN[91].AXIS[3].ORIENTATION[1]    =    V.G.KIN[91].PARAM[30]
V.G.KIN[91].AXIS[3].ORIENTATION[2]    =    V.G.KIN[91].PARAM[31]
V.G.KIN[91].AXIS[3].POINT[0]          =    V.G.KIN[91].PARAM[32]
V.G.KIN[91].AXIS[3].POINT[1]          =    V.G.KIN[91].PARAM[33]
V.G.KIN[91].AXIS[3].POINT[2]          =    V.G.KIN[91].PARAM[34]
V.G.KIN[91].AXIS[4].TYPE              =    V.G.KIN[91].PARAM[35]
V.G.KIN[91].AXIS[4].ORIENTATION[0]    =    V.G.KIN[91].PARAM[36]
V.G.KIN[91].AXIS[4].ORIENTATION[1]    =    V.G.KIN[91].PARAM[37]
V.G.KIN[91].AXIS[4].ORIENTATION[2]    =    V.G.KIN[91].PARAM[38]
V.G.KIN[91].AXIS[4].POINT[0]          =    V.G.KIN[91].PARAM[39]
V.G.KIN[91].AXIS[4].POINT[1]          =    V.G.KIN[91].PARAM[40]
V.G.KIN[91].AXIS[4].POINT[2]          =    V.G.KIN[91].PARAM[41]
V.G.KIN[91].AXIS[5].TYPE              =    V.G.KIN[91].PARAM[42]
V.G.KIN[91].AXIS[5].ORIENTATION[0]    =    V.G.KIN[91].PARAM[43]
V.G.KIN[91].AXIS[5].ORIENTATION[1]    =    V.G.KIN[91].PARAM[44]
V.G.KIN[91].AXIS[5].ORIENTATION[2]    =    V.G.KIN[91].PARAM[45]
V.G.KIN[91].AXIS[5].POINT[0]          =    V.G.KIN[91].PARAM[46]
V.G.KIN[91].AXIS[5].POINT[1]          =    V.G.KIN[91].PARAM[47]
V.G.KIN[91].AXIS[5].POINT[2]          =    V.G.KIN[91].PARAM[48]
V.G.KIN[91].CHAIN[0]                  =    V.G.KIN[91].PARAM[49]
V.G.KIN[91].CHAIN[1]                  =    V.G.KIN[91].PARAM[50]
V.G.KIN[91].CHAIN[2]                  =    V.G.KIN[91].PARAM[51]
V.G.KIN[91].CHAIN[3]                  =    V.G.KIN[91].PARAM[52]
V.G.KIN[91].CHAIN[4]                  =    V.G.KIN[91].PARAM[53]
V.G.KIN[91].CHAIN[5]                  =    V.G.KIN[91].PARAM[54]
V.G.KIN[91].PROGRAMMING_MODE          =    V.G.KIN[91].PARAM[55]
V.G.KIN[91].RTCP                      =    V.G.KIN[91].PARAM[56]
```

# 4 Kinematische Kette

## 4.1 Achsreihenfolge in der kinematischen Kette

Die meisten Kenngrößen der kinematischen Kette sind leicht zu bestimmen. Die einzige Herausforderung besteht ggf. in der Bestimmung der korrekten Achsreihenfolge, welche in dem Feld kinematik[91].chain definiert wird.

**Hinweis**

Bei der Universellen Kinematik unterscheidet sich die Achsreihenfolge von der Achsreihenfolge der zu ersetzenden Kinematik.

Bei der universellen Transformation wird **immer** vom Werkstück zum Werkzeug gegangen!

Dies muss bei der Konfiguration der Reihenfolge der beteiligten Achsen berücksichtigt werden.

### Allgemeine Regel

**Vorgehen zum Auffinden der Achsreihenfolge in der kinematischen Kette**

- Gedanklich vom Werkstück zum Maschinennullpunkt (MNP) und von dort weiter zur Werkzeugspitze gehen.
- Die Achsen in der Reihenfolge ihres Auftretens notieren. Das Feld kinematik[91].chain mit den entsprechenden Achsindizes in dieser Reihenfolge parametrieren.
- Für alle Achsen, die auf Werkstückseite liegen (also auftreten, bevor man den MNP erreicht), muss der Orientierungsvektor invertiert (mit -1 multipliziert) werden.

### 4.1.1 1. Beispiel anhand einer CA-Maschine

**Beispiel**

**CA-Maschine (ID09)**

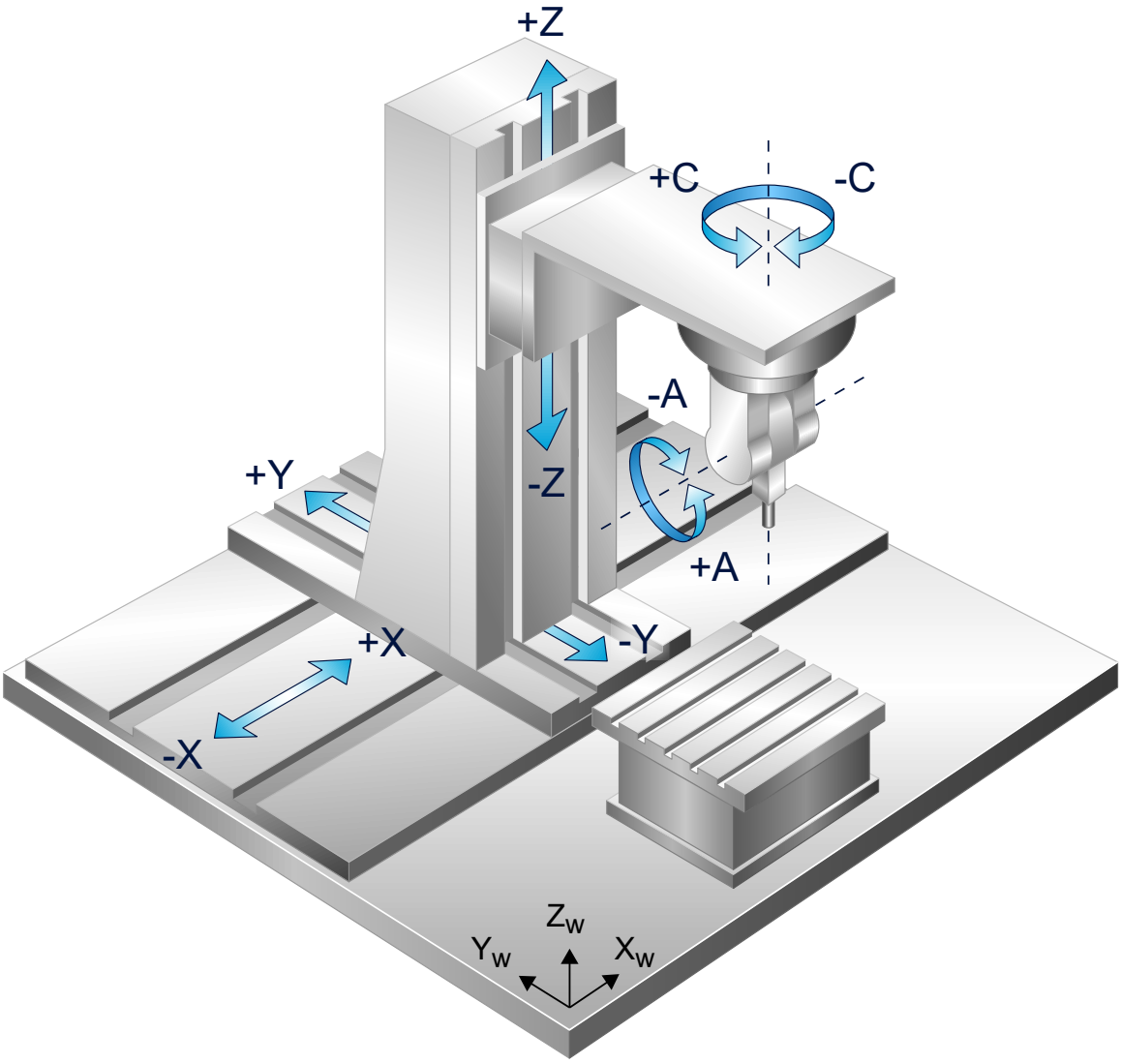



**Abb. 1: Achsreihenfolge bei der CA-Maschine**

Bei der CA-Maschine liegen alle Achsen auf Werkzeugseite, keine auf Werkstückseite. Läuft man gedanklich vom Maschinennullpunkt (MNP) zur Werkzeugspitze, begegnet man den Achsen

X, Y , Z, C, A

in dieser Reihenfolge. Dies ist die Achsreihenfolge für die kinematische Kette. Eine (vereinfachte) Konfiguration der CA-Maschine könnte wie folgt aussehen:

## Konfiguration einer CA-Maschine:

```
# Null-Orientierung des Werkzeuges
# Werkzeug zeigt in Z-Richtung
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Null-Position des Werkzeuges
# Werkzeug ruht im Punkt (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# X-Achse definieren (Index 0)
kinematik[91].axis[0].type              1
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0
…
# Y-Achse definieren (Index 1)
kinematik[91].axis[1].type              1
kinematik[91].axis[1].orientation[0]    0
kinematik[91].axis[1].orientation[1]    1
kinematik[91].axis[1].orientation[2]    0
…

# Z-Achse definieren (Index 2)
kinematik[91].axis[2].type              1
kinematik[91].axis[2].orientation[0]    0
kinematik[91].axis[2].orientation[1]    0
kinematik[91].axis[2].orientation[2]    1
…
# C-Achse definieren (Index 3)
kinematik[91].axis[3].type              2
kinematik[91].axis[3].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    1
…
# A-Achse definieren (Index 4)
kinematik[91].axis[4].type              2
kinematik[91].axis[4].orientation[0]    1
kinematik[91].axis[4].orientation[1]    0
kinematik[91].axis[4].orientation[2]    0
…
# Reihenfolge in kin. Kette: XYZCA
kinematik[91].chain[0]                  0 # X-Achse
kinematik[91].chain[1]                  1 # Y-Achse
kinematik[91].chain[2]                  2 # Z-Achse
kinematik[91].chain[3]                  3 # C-Achse
kinematik[91].chain[4]                  4 # A-Achse
```

### 4.1.2 2. Beispiel anhand einer AC-Maschine

**Beispiel**

**AC-Maschine (ID58)**

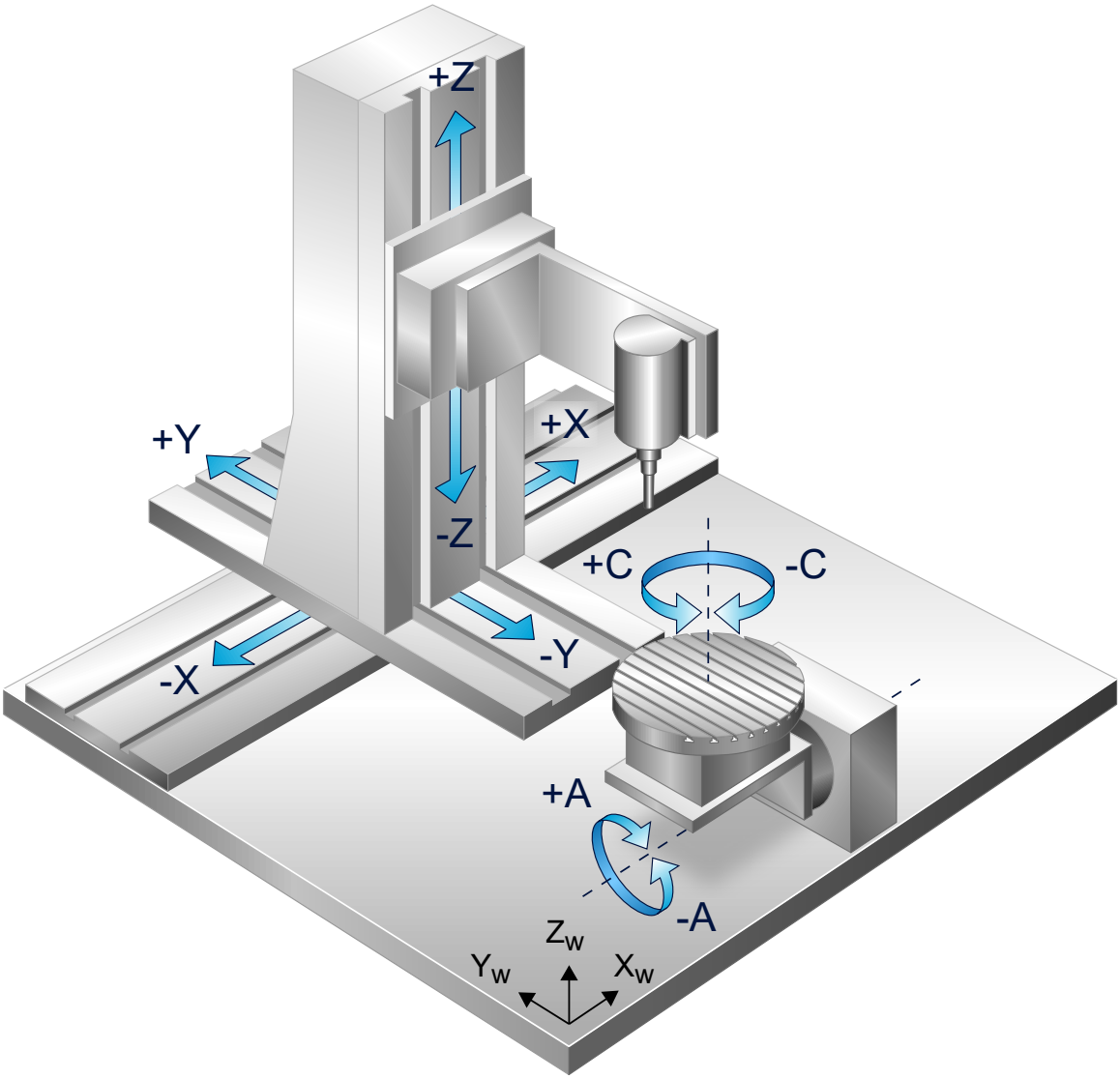



**Abb. 2: Kinematische Struktur der 5-achsigen Maschine mit AC Werkstücktisch**

Bei der AC-Maschine liegen die Linearachsen auf Werkzeugseite und die Rundachsen auf Werkstückseite (als Rotatoren im Werkzeugtisch). Läuft man gedanklich vom Werkstück zum MNP und von dort zur Werkzeugspitze, begegnet man den Achsen

C, A, X, Y, Z

in dieser Reihenfolge. Dies ist die Achsreihenfolge in der kinematischen Kette. Allerdings ist zu beachten, dass bei jeder Achse, die auf Werkstückseite liegt, der Orientierungsvektor mit -1 multipliziert werden muss. Hier eine mögliche Konfiguration der AC-Maschine:

## Konfiguration einer AC-Maschine:

```
# Null-Orientierung des Werkzeuges
# Werkzeug zeigt in Z-Richtung
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Null-Position des Werkzeuges
# Werkzeug ruht im Punkt (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# X-Achse definieren (Index 0)
kinematik[91].axis[0].type              1
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0
…
# Y-Achse definieren (Index 1)
kinematik[91].axis[1].type              1
kinematik[91].axis[1].orientation[0]    0
kinematik[91].axis[1].orientation[1]    1
kinematik[91].axis[1].orientation[2]    0
…
# Z-Achse definieren (Index 2)
kinematik[91].axis[2].type              1
kinematik[91].axis[2].orientation[0]    0
kinematik[91].axis[2].orientation[1]    0
kinematik[91].axis[2].orientation[2]    1
…


# C-Achse definieren (Index 3)
kinematik[91].axis[3].type              2
kinematik[91].axis[3].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    -1 # invertiert
…
# A-Achse definieren (Index 4)
kinematik[91].axis[4].type              2
kinematik[91].axis[4].orientation[0]    -1 # invertiert
kinematik[91].axis[4].orientation[1]    0
kinematik[91].axis[4].orientation[2]    0
…
# Reihenfolge in kin. Kette: CAXYZ
kinematik[91].chain[0]                  3 # C-Achse
kinematik[91].chain[1]                  4 # A-Achse
kinematik[91].chain[2]                  0 # X-Achse
kinematik[91].chain[3]                  1 # Y-Achse
kinematik[91].chain[4]                  2 # Z-Achse
```

### 4.1.3 3. Beispiel anhand einer AC-Maschine mit Kardan-Tisch

**Beispiel**

**AC-Maschine mit Kardan-Tisch**

Weiterhin können mit der Universellen Kinematik (ID91) kinematische Transformationen, die in der Kinematikbibliothek nicht vorhanden sind, nachgebildet werden.

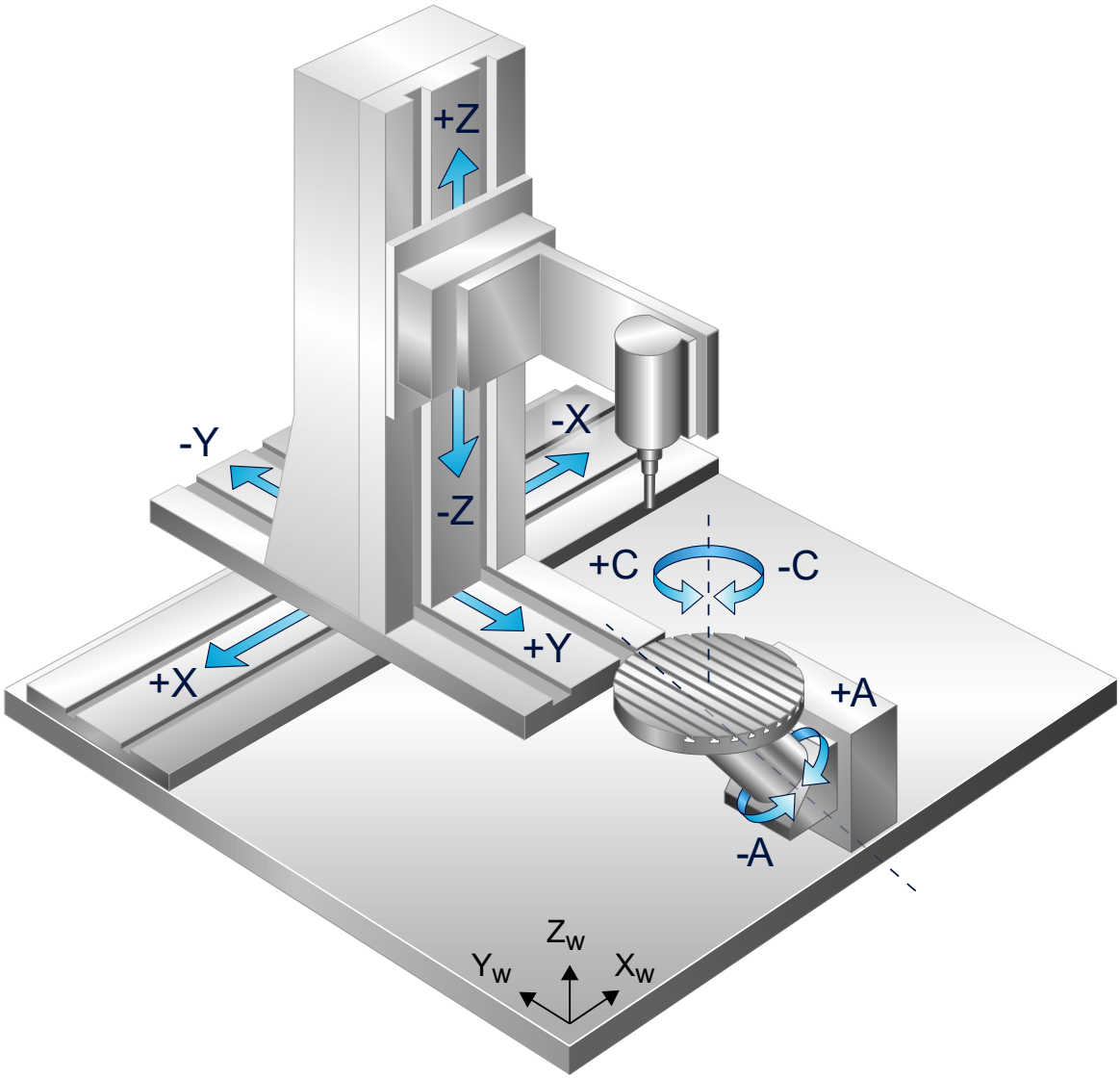



**Abb. 3: Kinematische Struktur der 5-achsigen Maschine mit Kardantisch**

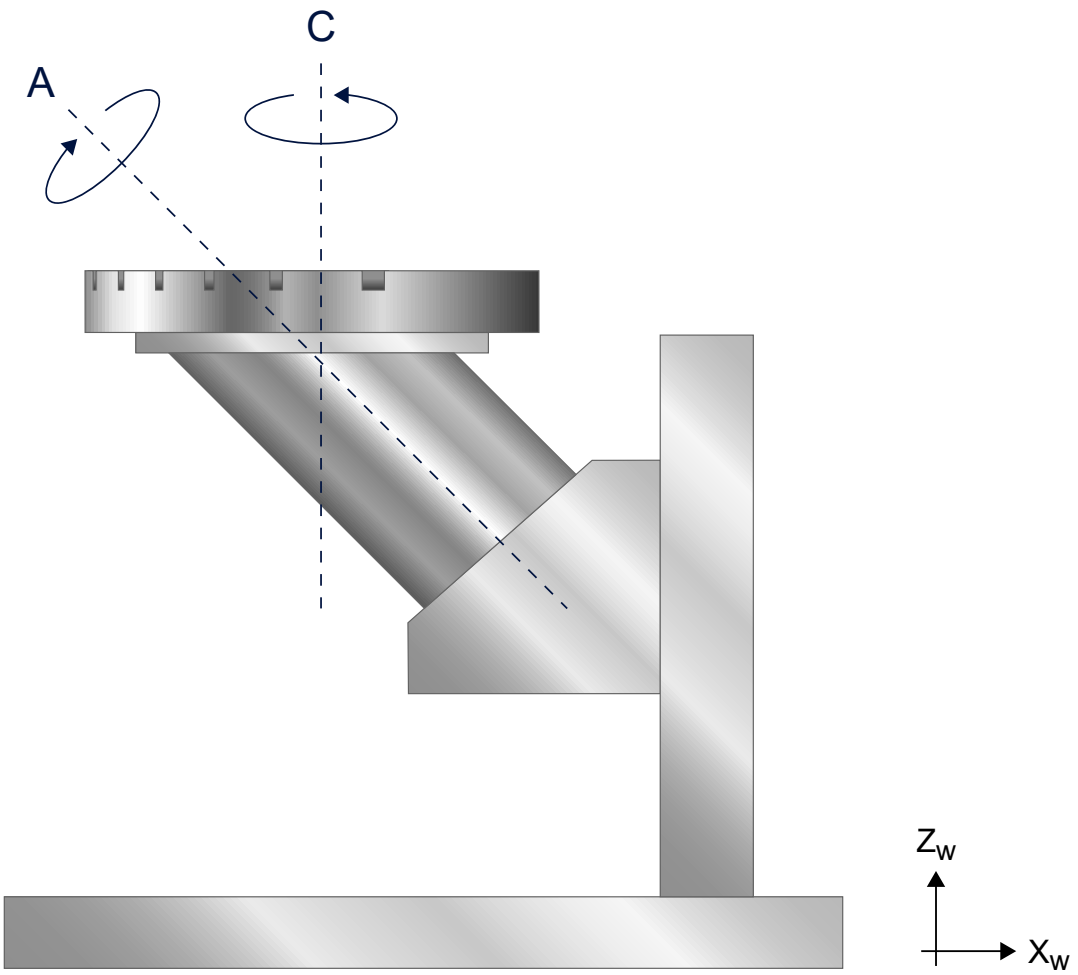



**Abb. 4: Detailansicht des Werkzeugtisches**

Bei der abgebildeten AC-Maschine mit Kardan-Tisch liegen die Linearachsen auf Werkzeugseite und die Rotationsachsen auf Werkstückseite (als Rotatoren im Werkzeugtisch). Auf dem Weg vom Werkstück zum MNP und von dort zur Werkzeugspitze, liegen die Achsen in der Reihenfolge

C, A, X, Y, Z

vor. Dies ist die Achsreihenfolge in der kinematischen Kette. Allerdings ist zu beachten, dass bei jeder Achse, die auf Werkstückseite liegt, der Orientierungsvektor mit -1 multipliziert werden muss. Hier eine mögliche Konfiguration der Kardan AC-Maschine:

## Konfiguration einer AC-Maschine mit Kardan-Tisch:

```
# Null-Orientierung des Werkzeuges
# Werkzeug zeigt in Z-Richtung
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Null-Position des Werkzeuges
# Werkzeug ruht im Punkt (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# Programmiermodus CA
kinematik[91].programming_mode          17

# RTCP-Modus einschalten
kinematik[91].rtcp                      1

# X-Achse definieren (Index 0)
kinematik[91].axis[0].type              1
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0
#…

# Y-Achse definieren (Index 1)
kinematik[91].axis[1].type              1
kinematik[91].axis[1].orientation[0]    0
kinematik[91].axis[1].orientation[1]    1
kinematik[91].axis[1].orientation[2]    0
#…

# Z-Achse definieren (Index 2)
kinematik[91].axis[2].type              1
kinematik[91].axis[2].orientation[0]    0
kinematik[91].axis[2].orientation[1]    0
kinematik[91].axis[2].orientation[2]    1
#…

# C-Achse definieren (Index 3)
kinematik[91].axis[3].type              2
kinematik[91].axis[3].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    -1 # invertiert
#…

# A-Achse definieren (Index 4) Kardanwinkel 45 Grad
kinematik[91].axis[4].type              2
kinematik[91].axis[4].orientation[0]    -1 # invertiert
kinematik[91].axis[4].orientation[1]    0
kinematik[91].axis[4].orientation[2]    -1 # invertiert
#…

# Reihenfolge in kin. Kette: CAXYZ
kinematik[91].chain[0]                  3 # C-Achse
kinematik[91].chain[1]                  4 # A-Achse
kinematik[91].chain[2]                  0 # X-Achse
kinematik[91].chain[3]                  1 # Y-Achse
kinematik[91].chain[4]                  2 # Z-Achse
```

## 4.2 Einschränkungen in der kinematischen Kette

**Achtung**

Bei der Erstellung einer kinematischen Kette gelten Einschränkungen, die aus der Architektur des Kerns, Speicherplatzbeschränkungen und algorithmischen Grenzen folgen.

### Achsanzahl

Für die Anzahl der Achsen gelten folgende Einschränkungen:

- Die Kinematik muss genau 3 Linearachsen haben.
- Die Kinematik darf höchstens 3 Rundachsen haben.
- Für vollständige Transformationen sind höchstens 2 Rundachsen möglich.

Damit ist die Nachbildung von 3-, 4-, 5- und 6-Achs-Maschinen möglich.

### Achsindizierung

Bei der Definition der Achsen gelten folgende Einschränkungen:

- Die 3 Linearachsen müssen auf den Indizes 0, 1, 2 definiert werden.
- Im RTCP-Modus müssen die Rundachsen auf denselben Indizes definiert werden, auf denen ihre Entsprechungen in den Achsgruppen der Kanalkonfiguration definiert sind, siehe P-CHAN-00006 und den **#SET AX**-Befehl. Wenn in der Kanalkonfiguration ein Eintrag existiert, dann muss auch in der Achsdefinition der kinematischen Kette die C-Achse auf Index 3 definiert werden. Siehe Codebeispiel:

```
gruppe[0].achse[3].bezeichnung    C
```

## 4.3 Testen der Konfiguration in der Simulation

In einer Simulation kann man die Konfiguration der Universalkinematik mithilfe eines NC-Programms leicht testen, wenn man eine ungefähre Vorstellung hat, welche Achsbewegung beim Fahren unter Trafo zu erwarten ist.

Als Vorlage kann das nachfolgende NC-Programm verwendet werden. das evtl. noch leicht angepasst werden muss.

Im NC-Programm werden die Achspositionen über den NC-Befehl #MSG in eine Ausgabedatei protokolliert.
(Siehe Schreiben von Meldungen in eine Datei (#MSG SAVE) und Definition von Dateinamen (#FILE NAME))

Die protokollierten Achspositionen müssen anschließend mit den erwarteten Achspositionen verglichen werden.


**Programmierbeispiel**

**Art und Quelle der Gefahr**

```
;------------------------------------------------------------
;
;   Zuerst wird die Universalkinematik im NC-Programm
;   konfiguriert, anstatt in der Kanal-Liste.
;   Das erleichtert den Test, da man nicht immer wieder
;   Listen neu laden muss.
;
;------------------------------------------------------------
; Werkzeug zeigt in Z-Richtung
N00110  V.G.KIN[91].ZERO_ORIENTATION[0]         = 0
N00120  V.G.KIN[91].ZERO_ORIENTATION[1]         = 0
N00130  V.G.KIN[91].ZERO_ORIENTATION[2]         = 1

; Nullposition des Flansches
N00160  V.G.KIN[91].ZERO_POSITION[0]            = 10000
N00170  V.G.KIN[91].ZERO_POSITION[1]            = 20000
N00180  V.G.KIN[91].ZERO_POSITION[2]            = 30000

; 5 Achsen XYZCA
N00210  V.G.KIN[91].NUMBER_OF_AXES              = 5

; X-Achse, translatorisch
N00240  V.G.KIN[91].AXIS[0].TYPE                = 1
N00250  V.G.KIN[91].AXIS[0].ORIENTATION[0]      = 1
N00260  V.G.KIN[91].AXIS[0].ORIENTATION[1]      = 0
N00270  V.G.KIN[91].AXIS[0].ORIENTATION[2]      = 0

; Y-Achse, translatorisch
N00300  V.G.KIN[91].AXIS[1].TYPE                = 1
N00310  V.G.KIN[91].AXIS[1].ORIENTATION[0]      = 0
N00320  V.G.KIN[91].AXIS[1].ORIENTATION[1]      = 1
N00330  V.G.KIN[91].AXIS[1].ORIENTATION[2]      = 0

; Z-Achse, translatorisch
N00360  V.G.KIN[91].AXIS[2].TYPE                = 1
N00370  V.G.KIN[91].AXIS[2].ORIENTATION[0]      = 0
N00380  V.G.KIN[91].AXIS[2].ORIENTATION[1]      = 0
N00390  V.G.KIN[91].AXIS[2].ORIENTATION[2]      = 1
```

```
; A-Achse
N00420  V.G.KIN[91].AXIS[3].TYPE                = 2
N00430  V.G.KIN[91].AXIS[3].ORIENTATION[0]      = 1
N00440  V.G.KIN[91].AXIS[3].ORIENTATION[1]      = 0
N00450  V.G.KIN[91].AXIS[3].ORIENTATION[2]      = 0
N00460  V.G.KIN[91].AXIS[3].POINT[0]            = 40000
N00470  V.G.KIN[91].AXIS[3].POINT[1]            = 50000
N00480  V.G.KIN[91].AXIS[3].POINT[2]            = 60000

; C-Achse
N00510  V.G.KIN[91].AXIS[4].TYPE                = 2
N00520  V.G.KIN[91].AXIS[4].ORIENTATION[0]      = 0
N00530  V.G.KIN[91].AXIS[4].ORIENTATION[1]      = 0
N00540  V.G.KIN[91].AXIS[4].ORIENTATION[2]      = 1
N00550  V.G.KIN[91].AXIS[4].POINT[0]            = 70000
N00560  V.G.KIN[91].AXIS[4].POINT[1]            = 80000
N00570  V.G.KIN[91].AXIS[4].POINT[2]            = 90000

; Achsreihenfolge in kinematischer Kette: XYZCA
N00600  V.G.KIN[91].CHAIN[0]                    = 0
N00610  V.G.KIN[91].CHAIN[1]                    = 1
N00620  V.G.KIN[91].CHAIN[2]                    = 2
N00630  V.G.KIN[91].CHAIN[3]                    = 4
N00640  V.G.KIN[91].CHAIN[4]                    = 3

; Programmiermodus 12 oder 13, P-CHAN-00112
N00670  V.G.KIN[91].PROGRAMMING_MODE            = 13

; RTCP-Modus einstellen
N00700  V.G.KIN[91].RTCP                        = 1

; Achsen holen
N00730  #SET AX [X, 1, 0] [Y, 2, 1] [Z, 3, 2] [A, 4, 3] [C, 5, 4]

;-----------------------------------------------------------
;
;   Nun kann man die Maschine bewegen, verschiedene
;   Werkzeuglängen ausprobieren usw.
;
;   Mit #CHANNEL INIT werden die Achsposition geholt
;   und mit #MSG und V.A.ACS.ABS in eine Textdatei
;   ausgegeben (normalerweise "message.txt").
;
;-----------------------------------------------------------

; Trafo anwaehlen
N00870  #KIN ID [91]
N00880  #TRAFO ON

; fahren
N00910  G00 X0 Y0 Z0 B0
N00920  #FLUSH WAIT
N00930  #CHANNEL INIT [CMDPOS]
N00940  #MSG SAVE EXCLUSIVE ["X-Sollwert: %f", V.A.ACS.ABS.X]
N00950  #MSG SAVE EXCLUSIVE ["Y-Sollwert: %f", V.A.ACS.ABS.Y]
N00960  #MSG SAVE EXCLUSIVE ["Z-Sollwert: %f", V.A.ACS.ABS.Z]
N00970  #MSG SAVE EXCLUSIVE ["A-Sollwert: %f", V.A.ACS.ABS.A]
N00980  #MSG SAVE EXCLUSIVE ["C-Sollwert: %f", V.A.ACS.ABS.C]
N00990  #MSG SAVE EXCLUSIVE [""]

; Trafo abwaehlen
N01020  #TRAFO OFF
```

```
; Werkzeuglaenge aendern
N01050  V.G.WZ_AKT.L = 99

; Trafo anwaehlen
N01080  #KIN ID [91]
N01090  #TRAFO ON

; fahren
N01120  G00 X0 Y0 Z0 B0
N01130  #FLUSH WAIT
N01140  #CHANNEL INIT [CMDPOS]
N01150  #MSG SAVE EXCLUSIVE ["X-Sollwert: %f", V.A.ACS.ABS.X]
N01160  #MSG SAVE EXCLUSIVE ["Y-Sollwert: %f", V.A.ACS.ABS.Y]
N01170  #MSG SAVE EXCLUSIVE ["Z-Sollwert: %f", V.A.ACS.ABS.Z]
N01180  #MSG SAVE EXCLUSIVE ["A-Sollwert: %f", V.A.ACS.ABS.A]
N01190  #MSG SAVE EXCLUSIVE ["C-Sollwert: %f", V.A.ACS.ABS.C]
N01200  #MSG SAVE EXCLUSIVE [""]

; fahren
N01230  G00 X0 Y0 Z0 B45
N01240  #FLUSH WAIT
N01250  #CHANNEL INIT [CMDPOS]
N01260  #MSG SAVE EXCLUSIVE ["X-Sollwert: %f", V.A.ACS.ABS.X]
N01270  #MSG SAVE EXCLUSIVE ["Y-Sollwert: %f", V.A.ACS.ABS.Y]
N01280  #MSG SAVE EXCLUSIVE ["Z-Sollwert: %f", V.A.ACS.ABS.Z]
N01290  #MSG SAVE EXCLUSIVE ["A-Sollwert: %f", V.A.ACS.ABS.A]
N01300  #MSG SAVE EXCLUSIVE ["C-Sollwert: %f", V.A.ACS.ABS.C]
N01310  #MSG SAVE EXCLUSIVE [""]

; Trafo abwaehlen
N01340  #TRAFO OFF

; beenden
N01370  M30
```

# 5 Programmiermodi

## Konfiguration

Der Programmiermodus kann eingestellt werden über:

- Kanalparameter P-CHAN-00112
- Variable V.G.KIN[91].PROGRAMMING_MODE
- Kanalparameter kinematik[91].programming_mode.

Dabei gilt bei Mehrfachangabe die folgende Priorisierung:

1. V.G.KIN[91].PROGRAMMING_MODE
2. Kinematik[91].programming_mode
3. P-CHAN-00112

# 5.1 Punkt-Vektor-Programmierung

## Einleitung

Bei der Punkt-Vektor-Programmierung werden die Position und die Orientierung des Werkzeuges direkt programmiert.

- Die Programmierung der Position erfolgt über die Buchstaben X, Y, Z.
- Die Programmierung der Orientierung erfolgt über die Buchstaben U, V, W.

## Achsdefinition

In den Kanalparametern müssen 6 Achsen definiert werden. Diese Achsen müssen in der Reihenfolge X, Y, Z, U, V, W indiziert sein.

**Hinweis**

In der Punkt-Vektor-Programmierung werden keine Winkel programmiert, d.h. das entsprechende RTCP-Flag wird ignoriert. Die Winkeltransformation ist vollständig, d.h., die Achswinkel werden aus der Werkzeugorientierung berechnet.

## 5.2 Winkelprogrammierung

In allen anderen Programmiermodi werden Winkel programmiert. Im RTCP-Modus (Rotation Tool Center Point-Modus) werden diese Winkel als Achswinkel übernommen.

Bei der vollständigen Transformation dienen die programmierten Winkel dazu, eine Werkzeugorientierung zu bestimmen, aus welcher die Achswinkel berechnet werden. Daher muss angegeben werden, auf welche Art und Weise aus den programmierten Winkeln eine Werkzeugorientierung zu berechnen ist.

Dies geschieht ebenfalls mit einer kinematischen Kette, welche im typischen Fall der kinematischen Kette der Maschine entspricht, jedoch ohne Achsversätze. Die CA-Maschine ID09 kann im CA-Programmiermodus programmiert werden, hinter dem sich die kinematische Kette XYZCA verbirgt. Die Leistung der Universellen Kinematik besteht dann darin, Achsversätze bei der Ermittlung der Achswerte zu berücksichtigen.

### 5.2.1 Klassische Programmiermodi

#### CA-Modus (17)

Position und Orientierung des Werkzeuges werden als Vorwärtstransformation der folgenden kinematischen Kette bestimmt.

- Nullposition des Werkzeuges ist [0, 0, 0]
- Nullorientierung des Werkzeuges ist [0, 0, 1] (Z-Richtung)
- Achsreihenfolge XYZCA
- Alle Achsen gehen durch den Nullpunkt [0, 0, 0]

#### Weitere Modi

Analog gibt es weitere Modi AC, AB, BA, BC, CB. Die Nullorientierung des Werkzeuges ist immer die Orientierung der erstgenannten Rundachse.

Weitere Informationen dazu siehe P-CHAN-00112.

## 5.2.2 Konforme und Direkte Programmierung

### Konform

Die Konforme Programmierung erweitert die Idee der klassischen Programmiermodi. Die verwendete kinematische Kette ist dabei eine Kopie der definierten kinematischen Kette der tatsächlichen Maschine, bei der die Achsversätze entfernt wurden.

Damit ist es z.B. möglich, eine Maschine mit ungewöhnlicher Achsstellung zu programmieren. Die Maschine mit der Transformation ID11 besitzt eine um 45° geneigte B-Achse. Definiert man die kinematische Kette für diese Maschine entsprechend und verwendet den konformen Programmiermodus, dann werden die programmierten B-Winkel tatsächlich als Winkel dieser Schief-Kinematik interpretiert.

**Hinweis**

**Bei konformer Programmierung müssen die Linear-Achsen XYZ auf den ersten drei Indizes des kinematik[].axis[]-Arrays liegen.**

### Direkt

Bei der Direkten Programmierung wird die kinematische Kette der Maschine zur Orientierungsberechnung verwendet, ohne die Achsversätze zu entfernen.

Weitere Informationen dazu siehe P-CHAN-00112.

# 6 Korrektur von Achsfehlstellungen

## Achsfehlstellung

Bei der realen Maschine sind die Achsen normalerweise nicht in der angenommenen idealen Stellung.

Die X-Achse kann einen gestörten Orientierungsvektor haben, also z.B. im MKS nicht nach (1, 0, 0) zeigen, sondern nach (0.99953, 0.0002, -0.0004). Dieser Wert kann durch einen Messzyklus mittels Ausgleichsrechnung gewonnen werden.

Diese Abweichungen können in der Kinematik abgebildet werden. Statt

```
# X-Achse definieren
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0
…
```

stellt man die folgenden Werte ein:

```
# X-Achse definieren
kinematik[91].axis[0].orientation[0]    0.99953
kinematik[91].axis[0].orientation[1]    0.0002
kinematik[91].axis[0].orientation[2]   -0.0004
…
```

## Korrektur von Achsfehlstellungen

Allgemein gilt:

- Fehler in Linearachsen können durch Korrektur der kartesischen Achswerte kompensiert werden.
- Fehler in Rundachsen können durch Korrektur der kartesischen und rotatorischen Achswerte kompensiert werden.

## Korrektur von Achswerten

Bei der Berechnung der Achswerte berücksichtigt und kompensiert die Kinematik die Achsfehlstellungen. Sie ermittelt Achswerte, die dann auf der fehlerhaften Maschine zu einer korrekten Positionierung und Orientierung des Werkzeuges führen.

**Achtung**

**Ausnahme:**

Im RTCP-Modus übernimmt die Kinematik die programmierten Achswinkel, daher ist eine Korrektur der Achswinkel nicht möglich.

Eine Korrektur der kartesischen Achswerte findet aber trotzdem statt.

# 7 Posenwechsel vermeiden

## Problembeschreibung

Für Maschinen mit 2 oder mehr rotatorischen Achsen gibt es im Allgemeinen mehrere Achswinkellösungen für das Einstellen der programmierten Werkzeugorientierung.

Während der Fahrt entlang einer Kontur ist es meistens unerwünscht, dass ein und dieselbe Orientierung mittels unterschiedlicher Achswinkel eingestellt wird („Posenwechsel“), da die Maschine ggf. unerwartete Ausgleichsbewegungen vollführt und damit Kollisionsgefahr besteht.

## Vermeidung

Beim Anwählen wählt die Kinematik eine Pose aus. Zu allen späteren Zeitpunkten wählt die Kinematik dann immer die Achswinkel als Lösung aus, die zu derselben Pose gehören.

**Achtung**

Für 4-Achsmaschinen ist eine solche Strategie nicht nötig, da höchstens eine Achswinkellösung existiert.

Die genannte Strategie wird für alle 5-Achs-Maschinen angewendet, die nicht im RTCP-Modus laufen. Der Wechsel einer Pose ist höchstens möglich, wenn die Transformation abgewählt ist. Die Wahl der Pose beim nächsten Anwählen der Transformation kann von außen aber nicht beeinflusst werden.

# 8 Bestehende Kinematiken nachbilden

Mit der Universellen Kinematik (ID91) ist es möglich, die meisten der kinematischen Transformationen aus der ISG Kinematikbibliothek nachzubilden. Dadurch können zum Beispiel Achsfehlstellungen kompensiert werden, siehe dazu Kapitel „Programmiermodi“. [▶ 28]

Das folgende Parametrierbeispiel zeigt, wie man mit der Universalkinematik eine CA-Kopf-Kinematik (KIN9) nachbilden kann.

Die dargestellte Syntax des Beispiels ist ab der Version V3.00 verfügbar, für Versionen V2.11.2xxx muss die Syntax analog umgesetzt werden (siehe Programmierung bzw. Beschreibung der kinematischen Kette [▶ 10]).

## Beispiel

**Nachbildung der Kinematik ID 9 mit universeller Kinematik**

```
# Kinematik ID
trafo[1].id 91

# Werkzeug zeigt in Z-Richtung
trafo[1].zero_orientation[0]          0
trafo[1].zero_orientation[1]          0
trafo[1].zero_orientation[2]          1

# Nullposition des Werkzeugs
trafo[1].zero_position[0]         54000
trafo[1].zero_position[1]       -395000
trafo[1].zero_position[2]       -950000

# 5 Achsen (XYZCA)
trafo[1].number_of_axes   5

# X-Achse
trafo[1].axis[0].type                 1
trafo[1].axis[0].orientation[0]       1
trafo[1].axis[0].orientation[1]       0
trafo[1].axis[0].orientation[2]       0
trafo[1].axis[0].point[0]             0
trafo[1].axis[0].point[1]             0
trafo[1].axis[0].point[2]             0

# Y-Achse
trafo[1].axis[1].type                 1
trafo[1].axis[1].orientation[0]       0
trafo[1].axis[1].orientation[1]       1
trafo[1].axis[1].orientation[2]       0
trafo[1].axis[1].point[0]             0
trafo[1].axis[1].point[1]             0
trafo[1].axis[1].point[2]             0

# Z-Achse
trafo[1].axis[2].type                 1
trafo[1].axis[2].orientation[0]       0
trafo[1].axis[2].orientation[1]       0
trafo[1].axis[2].orientation[2]       1
trafo[1].axis[2].point[0]             0
trafo[1].axis[2].point[1]             0
```

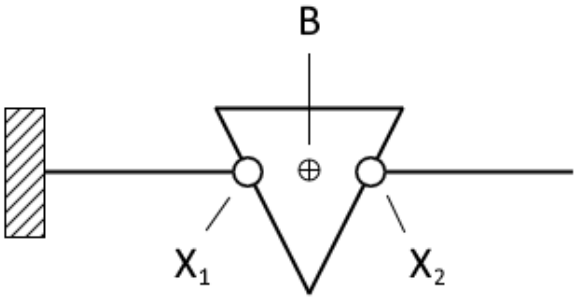
```
trafo[1].axis[2].point[2]              0

# C-Achse
trafo[1].axis[3].type                  2
trafo[1].axis[3].orientation[0]        0
trafo[1].axis[3].orientation[1]        0
trafo[1].axis[3].orientation[2]        1
trafo[1].axis[3].point[0]         610000
trafo[1].axis[3].point[1]         195000
trafo[1].axis[3].point[2]              0

# A-Achse
trafo[1].axis[4].type                  2
trafo[1].axis[4].orientation[0]        1
trafo[1].axis[4].orientation[1]        0
trafo[1].axis[4].orientation[2]        0
trafo[1].axis[4].point[0]              0
trafo[1].axis[4].point[1]         165000
trafo[1].axis[4].point[2]        -700000

# Achsreihenfolge in kinematischer Kette
trafo[1].chain[0]                      0
trafo[1].chain[1]                      1
trafo[1].chain[2]                      2
trafo[1].chain[3]                      3
trafo[1].chain[4]                      4

# Programmiermodus CA
trafo[1].programming_mode             17

# RTCP-Modus, Winkel werden direkt programmiert
trafo[1].rtcp                          1
```

# 9 Transformation zw. Achswerten und kartesischen Koordinaten

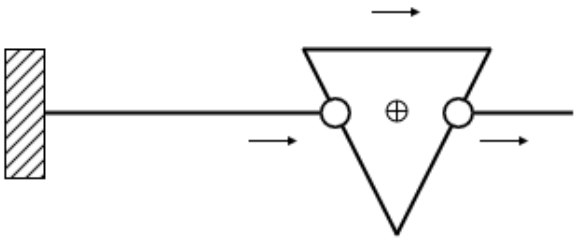
Im Normalfall wird mit der Universellen Kinematik eine serielle Kinematik abgebildet. Dabei trägt eine Achse alle anderen Achsen, die in der kinematischen Kette nach ihr folgen.

Durch kraftumformende mechanische Bauteile es jedoch möglich, Bewegungsführungen umzuleiten.

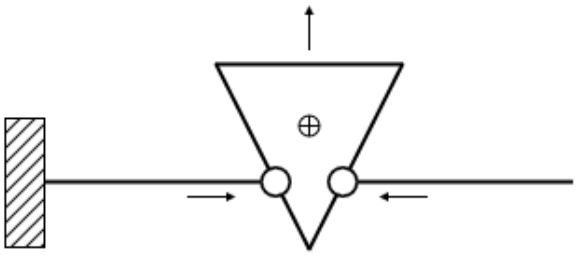
Beispiel: Zwei parallele X-Achsen erzeugen über einen Keil eine XZ-Bewegung. B ist hier ein Bezugspunkt auf dem Keil.




Fahren beide Achsen um die gleiche Distanz nach rechts, dann B ebenfalls und die Z-Höhe von B bleibt erhalten.



Fahren beide Achsen aufeinander zu, wird der Keil nach oben gedrückt.



Wegen der Symmetrie des Keils ist die kartesische X-Koordinate von B der Mittelwert beider Achswerte plus einer geeigneten Konstanten, also

$$X_{kart} = \frac{(X_1 + X_2)}{2} + c$$

Die Z-Koordinate von B hängt hingegen vom Abstand beider Achswerte ab. Mit geeigneten Konstanten für den Anstieg des Keil-Schenkels und einem optionalen Offset gilt also

$$Z_{kart} = m(X_1 - X_2) + D$$

Wenn es an der Maschine noch eine Y-Achse gibt, dann lässt sich der Zusammenhang zw. kartesischen Koordinaten und Achswerten wie folgt in Matrix-Form schreiben:

$$\begin{bmatrix} X_{kart} \\ Y_{kart} \\ Z_{kart} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \\ m & -m & 0 \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ Y \end{bmatrix} + \begin{bmatrix} C \\ 0 \\ D \end{bmatrix}$$

Die Matrix und der Offset-Vektor können im Kanalparameter trafo[].linkage (P-CHAN-00295 [▶ 47]) angegeben werden.

## Beispiel

**Transformation zwischen Achswerten und kartesischen Koordinaten**

```
# Kinematik-ID
trafo[0].id                    91
…

# Transformation aktivieren
trafo[0].linkage_mode          1

# Matrix, erste Zeile
trafo[0].linkage[0][0]         0.5
trafo[0].linkage[0][1]         0.5
trafo[0].linkage[0][2]         0

# Matrix, zweite Zeile
trafo[0].linkage[1][0]         0
trafo[0].linkage[1][1]         0
trafo[0].linkage[1][2]         1

# Matrix, dritte Zeile, Beispiel m = 0.71
trafo[0].linkage[2][0]         0.71
trafo[0].linkage[2][1]         -0.71
trafo[0].inkage[2][2]          0

# Offset-Vektor, Beispiel C = 1120000, D = -750000
trafo[0].linkage[0][3]         1120000
trafo[0].linkage[1][3]         0
trafo[0].linkage[2][3]         -750000
```

## Hinweis

**Die Parameter trafo[].axis[].orientation[] (P-CHAN-00292 [▶ 46]) beschreiben das kartesische System und bleiben deswegen erhalten.**
**Im Beispiel werden die Achsorientierungen für X1 und X2 nicht parallel definiert, sondern X1 zeigt in X-Richtung und X2 in Z-Richtung.**

# 10 Weitere Beispiele

## 10.1 Realisierung CB Kopfkinematik

Die existierende Kinematik mit der ID 9, mit der CA Kopfkinematik, soll zu einer CB Kopfkinematik umgewandelt werden.

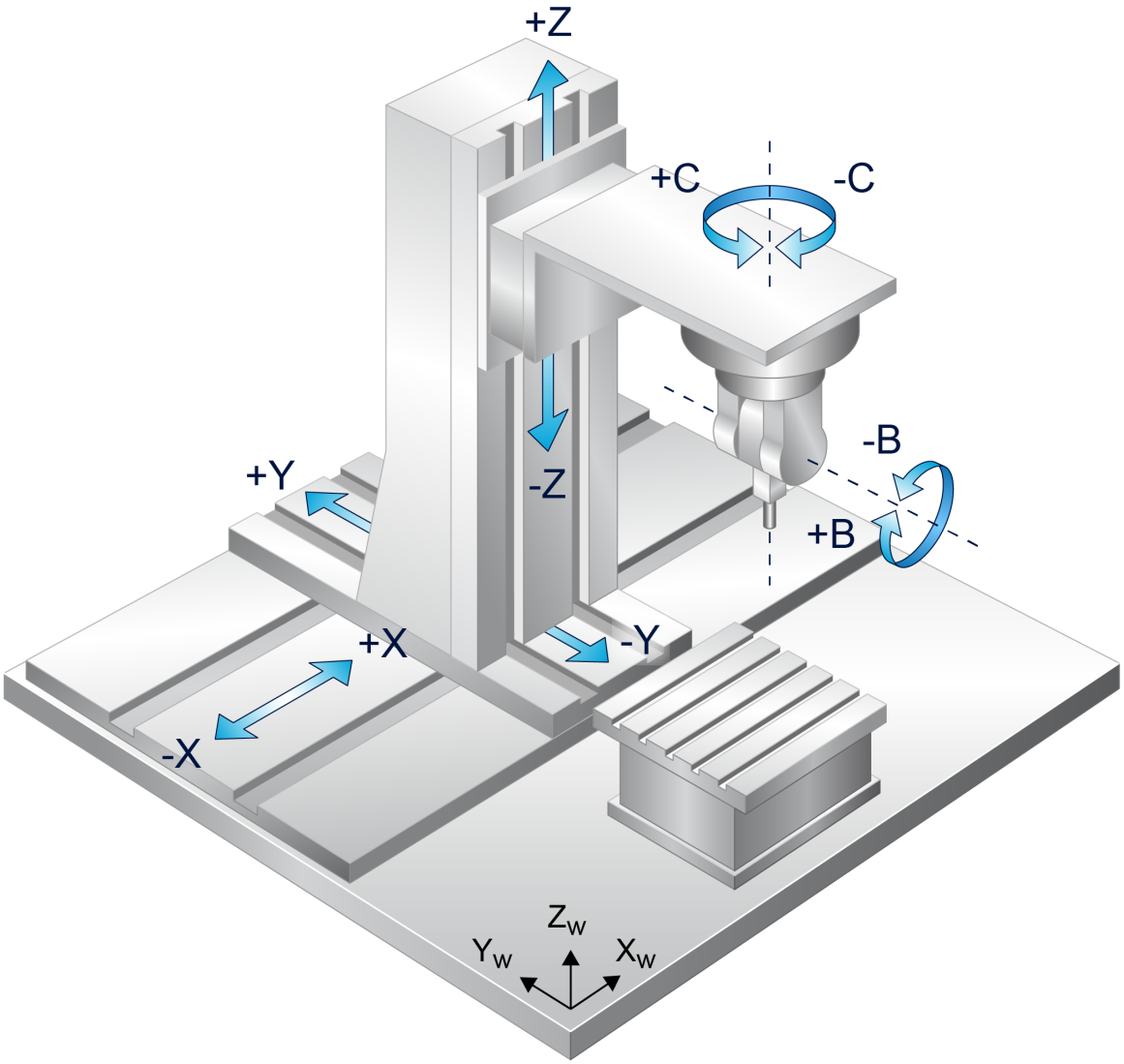



**Abb. 5: CB Kopfkinematik**

## Konfiguration einer CB-Maschine

```
# CB-Maschine: XYZCB
#
kinematik[91].rtcp                          1
# Programiermodus CB->19
kinematik[91].programming_mode              19
kinematik[91].number_of_axes                 5
#WZ Orientierung in Nullstellung der Maschinenachsen
kinematik[91].zero_orientation[0]           0
kinematik[91].zero_orientation[1]           0
kinematik[91].zero_orientation[2]           1
#WZ Aufnahme in Nullstellung der Maschinenachsen
kinematik[91].zero_position[0]              0
kinematik[91].zero_position[1]              0
kinematik[91].zero_position[2]              0
#
# X-Achse Typ , Lage (Index 0)
kinematik[91].axis[0].type                  1    #lin
kinematik[91].axis[0].orientation[0]        1
kinematik[91].axis[0].orientation[1]        0
kinematik[91].axis[0].orientation[2]        0
#
# Y-Achse Typ , Lage (Index 1)
kinematik[91].axis[1].type                  1    #lin
kinematik[91].axis[1].orientation[0]        0
kinematik[91].axis[1].orientation[1]        1
kinematik[91].axis[1].orientation[2]        0
#
# Z-Achse Typ , Lage (Index 2)
kinematik[91].axis[2].type                  1    #lin
kinematik[91].axis[2].orientation[0]        0
kinematik[91].axis[2].orientation[1]        0
kinematik[91].axis[2].orientation[2]        1
#
# C-Achse Typ , Lage (Index 3)
kinematik[91].axis[3].type                  2    #rot
kinematik[91].axis[3].orientation[0]        0
kinematik[91].axis[3].orientation[1]        0
kinematik[91].axis[3].orientation[2]        1
#
# B-Achse Typ , Lage (Index 4)
kinematik[91].axis[4].type                  2    #rot
kinematik[91].axis[4].orientation[0]        0
kinematik[91].axis[4].orientation[1]        1
kinematik[91].axis[4].orientation[2]        0
kinematik[91].axis[4].point[0]              0
kinematik[91].axis[4].point[1]              0
kinematik[91].axis[4].point[2]              1000000
#
# Reihenfolge der kinematischen Kette: YXZCB
kinematik[91].chain[0]    0    #X-Achse
kinematik[91].chain[1]    1    # Y-Achse
kinematik[91].chain[2]    2    # Z-Achse
kinematik[91].chain[3]    3    # C-Achse
kinematik[91].chain[4]    4    # B-Achse
#
```

Die Parametrierung der Maschine kann analog zum Beispiel der CA-Maschine [▶ 33] ebenfalls im NC-Programm festgelegt werden.

# 11 Parameter

## 11.1 Übersicht

| ID | Beschreibung |
|---|---|
| **P-CHAN-00112** | Orientierungswinkelmodus |
| **P-CHAN-00262** | Angabe der Kinematik ID = 91 |
| **P-CHAN-00285** | Null-Orientierung des Werkzeugs |
| **P-CHAN-00286** | Null-Position des Werkzeugs |
| **P-CHAN-00287** | Winkeltransformation |
| **P-CHAN-00288** | Programmiermodus |
| **P-CHAN-00289** | Achsanzahl |
| **P-CHAN-00290** | Achsreihenfolge |
| **P-CHAN-00291** | Achsspezifische Daten - Achstyp |
| **P-CHAN-00292** | Achsspezifische Daten – Achsorientierung |
| **P-CHAN-00293** | Achsspezifische Daten – Stützpunkt auf Achse |
| **P-CHAN-00295** | Transformation zw. Achswerten und kartesischen Koordinaten |

## 11.2 Beschreibung

| P-CHAN-00112 | Modus der Orientierungswinkelprogrammierung bei kinematischen Transformationen |
|---|---|
| Beschreibung | Bei einer vollständigen Transformation kann die Orientierung entweder über einen Orientierungsvektor mit den drei Komponenten U, V, W oder über drei Drehwinkel A, B, C um die Koordinatensystemachsen angegeben werden. Dies ist abhängig vom Transformationstyp.<br>Wegen des zusätzlichen Freiheitgrades in der Handorientierung ist die Programmierung der Drehwinkel A, B, C häufig die Eigenschaft von Roboterstrukturen.<br>Die Reihenfolge der drei Drehungen um die zugeordneten Rotationsachsen X, Y, Z führt zur gewünschten Zielorientierung bzw. zum Zieleffektorkoordinatensystem. Die Einzeldrehungen werden dabei, sofern nicht anders vermerkt, jeweils mathematisch positiv um die sich neu einstellenden Koordinatensystemachsen durchgeführt.<br>Ausgangspunkt ist eine Achssequenz mit den kartesischen Achsen X, Y, Z und den Drehachsen A, B, C. Die Standardzuordnung der Drehungen um die Koordinatensystemachsen ist A -> X, B -> Y, C -> Z. Diese kann bei speziellen Winkelmodi davon abweichen.<br>Manche Kinematiken verwenden spezielle Drehsequenzen, die hier nicht aufgeführt sind. Eine Umschaltung über P-CHAN-00112 ist in diesem Fall nicht möglich. Bei Standard-Fünfachskinematiken ist P-CHAN-00112 ohne Bedeutung.<br>Für die Universelle Kinematik (KIN-ID91) können in P-CHAN-00112 spezielle Werte eingetragen werden. |
| Parameter | ori_rotation_angle |
| Datentyp | SGN16 |
| Datenbereich | -1: Die programmierten Orientierungsachsen werden ohne Änderungen an die kinematische Transformation weitergereicht. Eine evtl. aktive kartesische Transformation mit einer aktiven Drehung hat keinen Einfluss auf diese Orientierungsachsen<br>**Vollständige kinematische Transformationen**<br>**3 Freiheitsgrade in der Orientierung**<br>0: YPR (Yaw Pitch Roll) Rotationsreihenfolge, 1. Drehung um Z (C), 2. Drehung negativ um Y´ (B), 3. Drehung um X´´ (A) (Standard)<br>1: Euler, Rotationsreihenfolge: 1. Drehung um Z (C), 2. Drehung um Y´(B), 3. Drehung um Z ´´ (C)<br>2: CBA, ähnlich YPR mit positiver B Drehung und anderer Achszuordnung. Drehung um Z (A), 2. Drehung um Y´ (B), 3. Drehung um X´´ (C).<br>-> A15 B-90 C20 (CBA) ist identisch zu A20 B90 C15 (YPR).<br>3: CAB Rotationsreihenfolge, 1. Drehung um Z (C), 2. Drehung um X´ (A), 3. Drehung um Y ´´ (B). (ab V3.1.3079.35)<br>4: CBA_STD, entspricht CBA mit anderer Achszuordnung. Rotationsreihenfolge: 1. Drehung um Z(C), 2. Drehung um Y‘(B), 3. Drehung um X‘‘(A)<br>-> A15 B90 C20 (CBA_STD) ist identisch zu A20 B90 C15 (CBA) und A15 B-90 C20 (YPR). (ab V3.1.3079.9)<br>5: ABC Rotationsreihenfolge, 1. Drehung um X (A), 2. Drehung um Y´ (B), 3. Drehung um Z ´´ (C). (ab V3.1.3079.35)<br>**2 Freiheitgrade in der Orientierung (vgl. KIN-ID 91)**<br>14: AB, Rotationsreihenfolge: 1. Drehung um X(A), 2. Drehung um Y‘(B) (ab V3.1.3079.30)<br>15: BA, Rotationsreihenfolge: 1. Drehung um Y(B), 2. Drehung um X‘(A) (ab V3.1.3079.30)<br>**Universelle kinematische Transformationen (KIN-ID 91):**<br>10: Punkt-Vektor-Programmierung. Die Orientierung des Werkzeuges wird über die Achsen U, V, W programmiert. Der Vektor [U, V, W] muss nicht normiert sein, darf aber nicht der Null-Vektor sein.<br>11: Freie Programmierung. Momentan nicht unterstützt. |

| | |
|---|---|
| | 12: Direkte Programmierung. Die konfigurierte kinematische Kette wird benutzt, um aus programmierten kartesischen Koordinaten und Winkeln die Position und Orientierung des Werkzeuges zu berechnen.<br>13: Konforme Programmierung. Wie Direkte Programmierung, aber ohne Achsversätze, Verschiebungen und Richtungsflags. Ermöglicht z.B. die Programmierung von 45°-Achsstellungen.<br>14: AB-Programmierung. 15: BA-Programmierung. 16: AC-Programmierung.<br>17: CA-Programmierung. 18: BC-Programmierung. 19: CB-Programmierung. |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| **P-CHAN-00262** | **Definition der Kinematik-ID für Transformationen** |
|---|---|
| Beschreibung | Die Kinematik-ID identifiziert als Element des Datensatzes der Kinematikparameter die zugehörige Transformation.<br>Die Definition kann sowohl für ein- als auch für mehrstufige Transformationen sowie für PCS-Transformationen erfolgen. |
| Parameter | trafo[j].id<br>kin_step[i].trafo[j].id (mehrstufige Transformationen)<br>trafo_pcs[i].id (PCS-Transformation *) |
| Datentyp | UNS16 |
| Datenbereich | 1 ... MAX(UNS16) |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | Parametersyntax ab V300<br>*Funktionalität PCS-Transformation ist verfügbar ab V3.1.3110. |

| **P-CHAN-00285** | **Null-Orientierung des Werkzeuges (Universelle Kinematik)** |
|---|---|
| Beschreibung | Parameter dient zur Angabe der Orientierung des Werkzeuges in der Nullstellung (Vektor X, Y, Z, Werkzeugrichtung). |
| Parameter | trafo[j].zero_orientation[k] mit k = 0, 1, 2<br>kin_step[i].trafo[j].zero_orientation[k] (mehrstufige Transformationen)<br>kinematik[91].zero_orientation[k] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | ---- |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00286 | Null-Position des Werkzeuges (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter dient zur Angabe der Position des Werkzeuges in der Nullstellung (Position X, Y, Z, Ruhepunkt). |
| Parameter | trafo[j].zero_position[k] mit k = 0, 1, 2<br>kin_step[i].trafo[j].zero_position[k] (mehrstufige Transformationen)<br>kinematik[91].zero_position[k] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | ---- |
| Dimension | 0.1µm bzw. 0.0001inch |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00287 | Winkeltransformation (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter bestimmt bei aktivem Winkelprogrammiermodus (P-CHAN-00288) die Art der Behandlung der programmierten Winkel. |
| Parameter | trafo[j].rtcp<br>kin_step[i].trafo[j].rtcp (mehrstufige Transformationen)<br>kinematik[91].rtcp (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | 0: Vollständige Transformation: Umrechnung der Winkel in den Bereich (-π, π] und Weitergabe an die Maschine (Standard).<br>1: RTCP-Transformation: Programmierte Winkel werden direkt an die Maschine weitergegeben. |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00288 | Programmiermodus (Universelle Kinematik) |
|---|---|
| Beschreibung | Der Programmiermodus gibt an, wie die Orientierung des Werkzeuges aus den programmierten Werten bestimmt wird (Punkt-Vektor-Modus oder Winkelmodi). Alternativ kann der Modus auch über den Kanalparameter P-CHAN-00112 gesetzt werden.<br>Eine Einstellung in der Kinematik hat Priorität vor P-CHAN-00112. |
| Parameter | trafo[j].programming_mode<br>kin_step[i].trafo[j].programming_mode (mehrstufige Transformationen)<br>kinematik[91].programming_mode (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | 10: Punkt-Vektor-Programmierung. Die Orientierung des Werkzeuges wird über die Achsen U, V, W programmiert. Der Vektor [U, V, W] muss nicht normiert sein, darf aber nicht der Null-Vektor sein.<br>11: Freie Programmierung. Momentan nicht unterstützt.<br>12: Direkte Programmierung. Die konfigurierte kinematische Kette wird benutzt, um aus programmierten kartesischen Koordinaten und Winkeln die Position und Orientierung des Werkzeuges zu berechnen.<br>13: Konforme Programmierung. Wie Direkte Programmierung, aber ohne Achsversätze, Verschiebungen und Richtungsflags. Ermöglicht z.B. die Programmierung von 45°-Achsstellungen.<br>14: AB-Programmierung.<br>15: BA-Programmierung.<br>16: AC-Programmierung.<br>17: CA-Programmierung.<br>18: BC-Programmierung.<br>19: BC-Programmierung. |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00289 | Achsanzahl (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter beschreibt die Anzahl der Achsen der kinematischen Kette. |
| Parameter | trafo[j].number_of_axes<br>kin_step[i].trafo[j].number_of_axes (mehrstufige Transformationen)<br>kinematik[91].number_of_axes (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | 3 ... 6 |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | Der Parameter ist ebenfalls in mehrstufigen Transformationen verfügbar. Der Zugriff auf den Parameter lautet:<br>kin_step[i].trafo[j].number_of_axes |

| P-CHAN-00290 | Achsreihenfolge (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter beschreibt die Reihenfolge der Achsen in der kinematischen Kette. |
| Parameter | trafo[j].chain[k] mit k = 0 ... P-CHAN-00289 – 1<br>kin_step[i].trafo[j].chain[k] (mehrstufige Transformationen)<br>kinematik[91].chain[k] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | ---- |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00291 | Achstyp (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter legt den Typ der Achse fest. |
| Parameter | trafo[j].axis[k].type<br>kin_step[i].trafo[j].axis[k].type (mehrstufige Transformationen)<br>kinematik[91].axis[k].type (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | 1: Translator<br>2: Rotator |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00292 | Achsorientierung (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter definiert den Richtungsvektor (X, Y, Z, kein Nullvektor) der Achse. |
| Parameter | trafo[j].axis[k].orientation[i] mit i = 0, 1, 2<br>kin_step[i].trafo[j].axis[k].orientation[i] (mehrstufige Transformationen)<br>kinematik[91].axis[k].orientation[i] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | ---- |
| Dimension | ---- |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00293 | Stützpunkt auf der Achse (Universelle Kinematik) |
|---|---|
| Beschreibung | Parameter definiert einen Stützpunkt auf der Achse (Position X, Y, Z, nur relevant bei Rundachsen). |
| Parameter | trafo[j].axis[k].point[l] mit l = 0, 1, 2<br>kin_step[i].trafo[j].axis[k].point[l] (mehrstufige Transformationen)<br>kinematik[91].axis[k].point[l] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | ---- |
| Dimension | 0.1µm |
| Standardwert | 0 |
| Anmerkungen | |

| P-CHAN-00295 | Transformation zw. Achswerten und kartesischen Koordinaten (Universelle Kinematik) |
|---|---|
| Beschreibung | Definition einer Matrix und eines Offset-Vektors, um eine lineare Transformation zwischen Achswerten der Linearachsen und kartesischen Koordinaten zu beschreiben.<br>$\begin{bmatrix} X_{kart} \\ Y_{kart} \\ Z_{kart} \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$<br>Rundachsen gehen in die Berechnung nicht ein. Die Buchstaben X, Y, Z stehen hier für die drei Linearachsen der Universellen Kinematik, in der Reihenfolge ihrer Definition im Parameter trafo[].axis[] (P-CHAN-00293 [▶ 47]).<br>Die Matrix wird in trafo[].linkage[0-2][0-2] definiert. Der erste Index gibt die Zeilennummer an, der zweite Index die Spaltennummer, beide 0-basiert.<br>Der Offset-Vektor wird in trafo[].linkage[0-2][3] definiert.<br>Ist die Matrix nicht invertierbar, so wird der Fehler ID 292010 ausgegeben.<br>Konfigurationsbeispiel siehe [FCT-C27// Transformation zw. Achswerten und kartesischen Koordinaten [▶ 35]]. |
| Parameter | trafo[j].linkage[k][l] mit k = 0, 1, 2 und l = 0, 1, 2, 3<br>kin_step[i].trafo[j].linkage[k][l] (mehrstufige Transformationen)<br>kinematik[91].linkage[k][l] (bis Version V2.11.28xx) |
| Datentyp | REAL64 |
| Datenbereich | |
| Dimension | für die Matrix: ----<br>für den Offset-Vektor: 0.1µm |
| Standardwert | 0 |
| Anmerkungen | |

# 12 Anhang

## 12.1 Anregungen, Korrekturen und neueste Dokumentation

Sie finden Fehler, haben Anregungen oder konstruktive Kritik? Gerne können Sie uns unter documentation@isg-stuttgart.de kontaktieren. Die aktuellste Dokumentation finden Sie in unserer Onlinehilfe (DE/EN):

**QR-Code Link:** https://www.isg-stuttgart.de/documentation-kernel/

**Der o.g. Link ist eine Weiterleitung zu:**

https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html

**Hinweis**

**Mögliche Änderung von Favoritenlinks im Browser:**

Technische Änderungen der Webseitenstruktur betreffend der Ordnerpfade oder ein Wechsel des HTML-Frameworks und damit der Linkstruktur können nie ausgeschlossen werden.

Wir empfehlen, den o.g. „QR-Code Link“ als primären Favoritenlink zu speichern.

**PDFs zum Download:**

DE:
https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

EN:
https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

**E-Mail:** documentation@isg-stuttgart.de

# Stichwortverzeichnis

## P

P-CHAN-00112 ..... 42
P-CHAN-00262 ..... 43
P-CHAN-00285 ..... 43
P-CHAN-00286 ..... 44
P-CHAN-00287 ..... 44
P-CHAN-00288 ..... 45
P-CHAN-00289 ..... 45
P-CHAN-00290 ..... 46
P-CHAN-00291 ..... 46
P-CHAN-00292 ..... 46
P-CHAN-00293 ..... 47
P-CHAN-00295 ..... 47


© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
Alle Rechte vorbehalten
www.isg-stuttgart.de
support@isg-stuttgart.de