DOCUMENTATION ISG-kernel

Functional description
Universal Kinematic

Short Description:
FCT-C27

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de

Documentation version: 1.18
12/12/2023

Preface

Legal information

This documentation was produced with utmost care. The products and scope of functions described are under continuous development. We reserve the right to revise and amend the documentation at any time and without prior notice.

No claims may be made for products which have already been delivered if such claims are based on the specifications, figures and descriptions contained in this documentation.

Personnel qualifications

This description is solely intended for skilled technicians who were trained in control, automation and drive systems and who are familiar with the applicable standards, the relevant documentation and the machining application.

It is absolutely vital to refer to this documentation, the instructions below and the explanations to carry out installation and commissioning work. Skilled technicians are under the obligation to use the documentation duly published for every installation and commissioning operation.

Skilled technicians must ensure that the application or use of the products described fulfil all safety requirements including all applicable laws, regulations, provisions and standards.

Further information

Links below (DE)

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

or (EN)

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

contains further information on messages generated in the NC kernel, online help, PLC libraries, tools, etc. in addition to the current documentation.

Disclaimer

It is forbidden to make any changes to the software configuration which are not contained in the options described in this documentation.

Trade marks and patents

The name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® and the associated logos are registered and licensed trade marks of ISG Industrielle Steuerungstechnik GmbH.

The use of other trade marks or logos contained in this documentation by third parties may result in a violation of the rights of the respective trade mark owners.

Copyright

General and safety instructions

Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

Icons in explanatory text

➢ Indicates an action.

⇨ Indicates an action statement.

⚠ DANGER

Acute danger to life!

If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

⚠ CAUTION

Personal injury and damage to machines!

If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

Attention

Restriction or error

This icon describes restrictions or warns of errors.

Notice

Tips and other notes

This icon indicates information to assist in general understanding or to provide additional information.

Example

General example

Example that clarifies the text.

Programing Example

NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

Release Note

Specific version information

Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

Table of contents

List of figures

1 Overview

Task

“Universal kinematics” has the **ID91** and is a kinematic transformation.

Free configurability makes it possible to create new kinematic transformations without having to extend the ISG kinematics library ([KITRA]).

Properties

The kinematic is not based on a special machine. However, the free configurability function permits the simulation of machines that are describable by a kinematic chain (serial kinematics).

They include conventional 5-axis machines (CA machine, 45° BA machine, etc.), but 3 and 4-axis machines are also possible.

Parameterisation

Parameterisation is dependent on machine structure and is therefore highly individual. This is described in detail in the section “Configuring the kinematics” [▸ 8].

The programming mode is set via the channel parameter P-CHAN-00112 or via the kinematic itself. A setting on the kinematic has priority over P-CHAN-00112.

Programming

A distinction is made between 2 programming modes:

1. Point-vector programming
2. Angle programming (conventional, compliant and direct)

Notice

Transformations are additional options and subject to the purchase of a license.

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 Description

Concepts

Kinematics supports two different programming modes:

1. Point-vector mode (simpler). The user programs tool orientation and position directly.
2. Angles are programmed in all the other modes. The kinematics interprets these angles based on the given programming mode and calculates tool orientation and position as well as Cartesian values. Users can then decide on the method to determine axis angles:

Axis angle calculation	Description
RTCP (Rotation Tool Centre Point) / incomplete	Programmed angles are simply adopted as axis angles.
Complete	As in the case of point-vector mode, axis angles are determined from the calculated orientation.

Applications

Since the orientation and interpolation point must be specified explicitly for each axis, machines can be mapped even if they have unusual axis positions (45° angles, 30° angles, etc.).

For the same reason, incorrect machine-related axis positions can also be compensated. If the kinematics is configured accordingly, it calculates compensatory axis values.

Since the sequence of axes in the kinematic chain is freely configurable, rotary axes can be mapped in the tool table, for example.

3 Configuring the kinematics

Overview

The kinematics should be configured to simulate a classic CA machine with the axes XYZCA. The kinematics is configured in three steps:

Step 1: Kinematic chain

- Tool zero position
- Description of participating axes
- Sequence of axes in the kinematic chain

Step 2: Programming mode

- Point vector, CA, BA, etc.

Step 3: Angle transformation

- Incomplete (RTCP = Rotation Tool Centre Point, i.e. rotation angles are not mapped by the transformation) or
- Complete transformation.

Notice

Each of the components can be set by channel list files (e.g. default_sda.lis) or by variables in the NC code.

This documentation uses lists in the examples.

3.1 Description of flange and tool

The parameters *zero_position* (P-CHAN-00286) and *zero_orientation* (P-CHAN-00285) specify the zero position and the zero orientation of the machine.

Zero orientation of the machine

The parameter P-CHAN-00286 specifies the position of the flange in the zero orientation of the machine.

Example

Parameter definition of the flange zero position

```
# Flange rests at point (12000, -3200, 500)
kinematik[91].zero_position[0]    12000
kinematik[91].zero_position[1]    -3200
kinematik[91].zero_position[2]    500
```

Tool orientation in relation to zero position

The parameter P-CHAN-00285 specifies the position of the tool in the zero orientation of the machine.

The parameter *zero_orientation* then only acts on the kinematic if a tool length is used. In the zero orientation of the machine, the position of the TCP is calculated as follows:

```
TCP = zero_position - tool length * zero_orientation
```

Example

Zero orientation of the tool

```
# Tool points in Z direction
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1
```

Example

Zero orientation of the tool at an angle of 45 degrees

```
# Tool is at an angle of 45° to the Y and Z axes
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    1
kinematik[91].zero_orientation[2]    1
```

The parameter *zero_orientation* (P-CHAN-00285) must not be specified as a unit vector of length 1. It is automatically normalised when read. This means, the previous examples has the same zero orientation as the example below.

Example

Zero orientation of the tool at an angle of 45 degrees without normalisation

```
# Tool is at an angle of 45° to the Y and Z axes
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0,707
kinematik[91].zero_orientation[2]    0,707
```

3.2 Programming or description of the kinematic chain

Notice

All components of the kinematics are described in the same coordinate system (PCS).

Release Note

The syntax of the CNC Builds V2.11.20xx or V2.11.28xx is used in list files in this documentation.

This syntax changed for transformation configurations as of Build V3.00.

For CNC Builds > V3.00 P-CHAN-00262 [▸ 43] must be absolutely assigned the transformation ID 91.

Old syntax: for CNC Builds V2.11.20xx and V2.11.28xx	New syntax: for CNC Builds as of V3.00 and higher:
kinematik[91].zero_orientation[0] 0 kinematik[91].zero_orientation[1] 0 …	trafo[0].id 91 trafo[0].zero_orientation[0] 0 trafo[0].zero_orientation[1] 0 …

Number of axes

The number of axes is specified by:

```
# typical CA machine: XYZCA
kinematik[91].number_of_axes          5
```

axes

Each of the axes is defined by the following characteristics.

Field	Description
Type	Linear axis (1) or rotary axis (2); see P-AXIS-00018
Orientation	Direction vector of axis, not zero vector
Point	Interpolation point, not relevant for rotary axes

The individual axes are specified by:

```
# Define X axis
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

# Define Y axis
…
# Define Z axis
…
# Define C axis
# points in Z direction and crosses
# point (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

# Define A axis
…
```

Axis sequence

The sequence of the axes in the kinematic chain must be specified. This sequence can, but need not, match the sequence of axis definitions. For example, it is then possible to place rotary axes at the start of the kinematic chain to simulate a rotary axis in the workpiece table. See Section Setting the programming mode [▸ 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
```

where "chain[i] = j" means that the i-th position in the kinematic chain is occupied by the j-th axis.

Attention

Universal Kinematics (ID91) may only be activated if all participating axes exist in the channel. Otherwise, an error is output.

3.3 Setting the programming mode

Explanation

The programming mode specifies how tool orientation is determined from the programmed values.

- In the point-vector mode, the user programs tool position and orientation directly.
- In all other modes (angle programming modes), the user programs angles that are then passed on directly as axis angles (RTCP = Rotation Tool Centre Point) or are used to calculate tool orientation.

Programming mode

The programming mode is set via the channel parameter P-CHAN-00112 or via the kinematic itself (P-CHAN-00288 [▸ 45]). A setting on the kinematic has priority over P-CHAN-00112.

The classic programming mode for CA angles is used.

```
# Set CA programming mode via P-CHAN-00112
ori_rotation_angle                      17

# Set CA programming mode at kinematic via P-CHAN-00288 [▸ 45]
kinematik[91].programming_mode          17
```

3.4 Setting angle transformation

Description

In an angle programming mode (all except point-vector), programmed angles can be treated in two ways.

1. With RTCP transformation (incomplete), programmed angles are passed on directly without conversion.
   The RTCP mode is the default for most transformations of the ISG kinematics library ([KITRA]).
2. With complete transformation, the kinematics interprets the programmed angles based on the programming mode and then determines tool orientation. The axis angles are calculated from this orientation.

The setting can be made using P-CHAN-00287 [▸ 44].

Notice

In RTCP mode, the programmed angles are passed on to the machine.

With complete transformation, angles in the range (-π, π] are passed on to the machine.

Attention

With complete transformation, axis angles may deviate considerably from the programmed angles. This applies in particular if rotary axes are not located at the end of the kinematic chain but in the workpiece, for example.

Angle transformation

RTCP mode is used.

```
# Activate RTCP mode
kinematik[91].rtcp                    1
```

3.5 Notes on parameterisation

The Universal Kinematic can be parameterised by

- list parameter: kinematik[91].param[i] and trafo[j].param[i]
- V.G. variables: V.G.KIN[91].PARAM[i] / V.KIN[91].ZERO…
- Tool offsets, tool database

.

Attention

All the various parameterisation options use the same memory location. This must be observed for read and write access.

Below is an example of a comparison of the two parameterisation options via V.G. variables.

```
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 Kinematic chain

4.1 Sequence of axes in the kinematic chain

Most characteristics in the kinematic chain are easy to determine. The only challenge may consist in determining the correct axis sequence defined in the kinematic chain field kinematik[91].chain.

Notice

With Universal Kinematics the axis sequence is different from the axis sequence of the replaced kinematics.

Universal transformation **always** goes from workpiece to tool.

This must be considered when configuring the sequence of the participating axes.

General rules

Procedure for finding the axis sequence in the kinematic chain

- Imagine walking from the workpiece to the machine origin and from there on to the tool tip.
- Notice the sequence of axes in the order of their occurrence. Enter the applicable axis indices in this order in the kinematik[91].chain field.
- The orientation vector must be inverted for all axes that are located on the workpiece side (i.e. occur before you reach the machine origin).

4.1.1 1. Example of a CA machine

Example

CA machine (ID09)

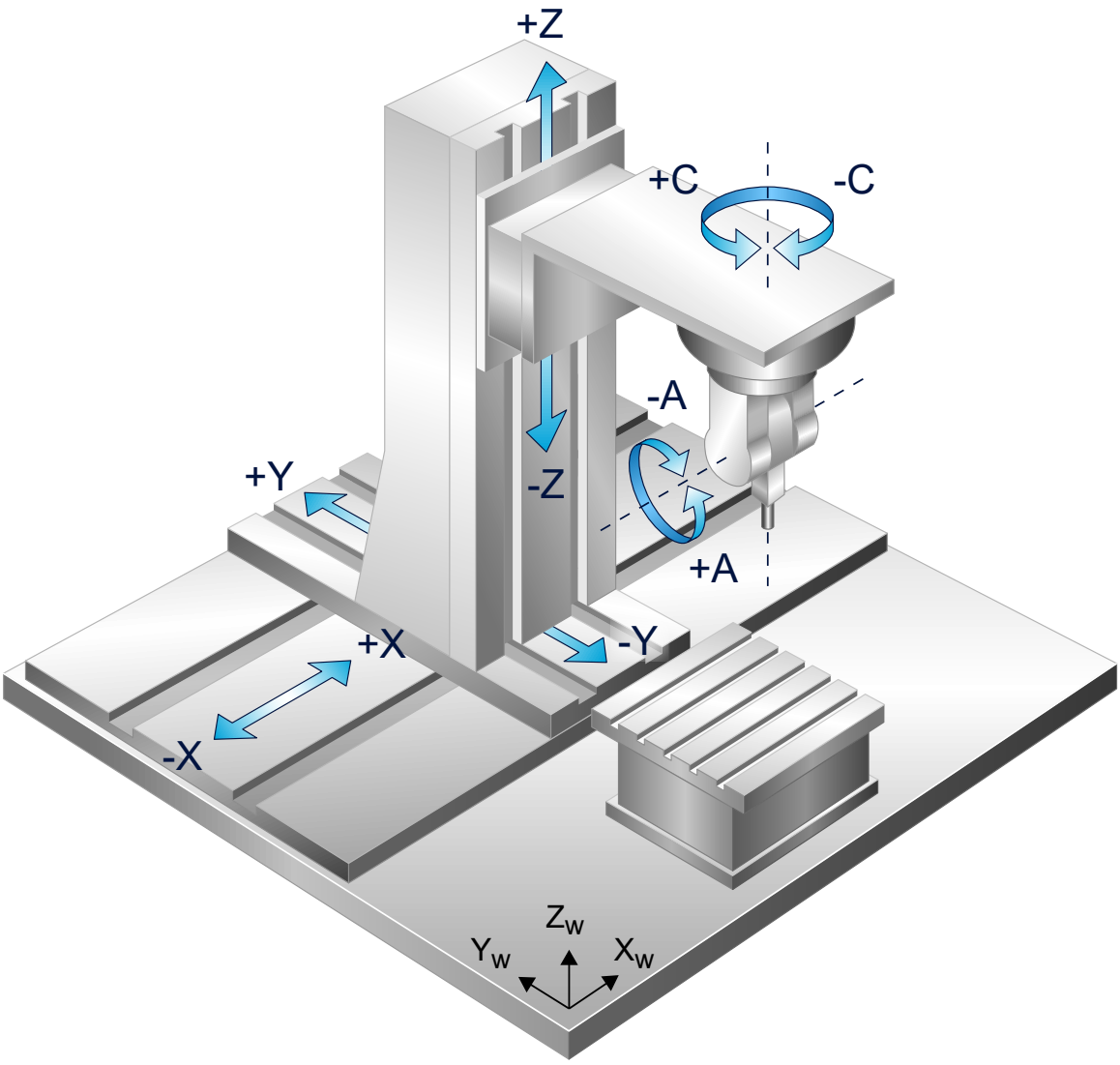


Fig. 1: Axis sequence with a CA machine

With a CA machine, all axes are on the tool side and none of them on the workpiece side. If you imagine walking from the machine origin (MNP) to the tool tip, the axes will occur

X, Y , Z, C, A

in this order. This is the axis sequence for the kinematic chain. A (simplified) configuration of the CA machine may also look like the following:

Configuration of a CA machine

```
# Zero orientation of the tool
# Tool points in Z direction
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Zero position of the tool
# Tool rests at point (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# Define X axis (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
…
# Define Y axis (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
…

# Define Z axis (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
…
# define C axis (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
…
# Define A axis (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
…
# Sequence in kin. chain: XYZCA
kinematik[91].chain[0]                  0 # X axis
kinematik[91].chain[1]                  1 # Y axis
kinematik[91].chain[2]                  2 # Z axis
kinematik[91].chain[3]                  3 # C axis
kinematik[91].chain[4]                  4 # A axis
```

4.1.2 2. Example of an AC machine

Example

AC machine (ID58)

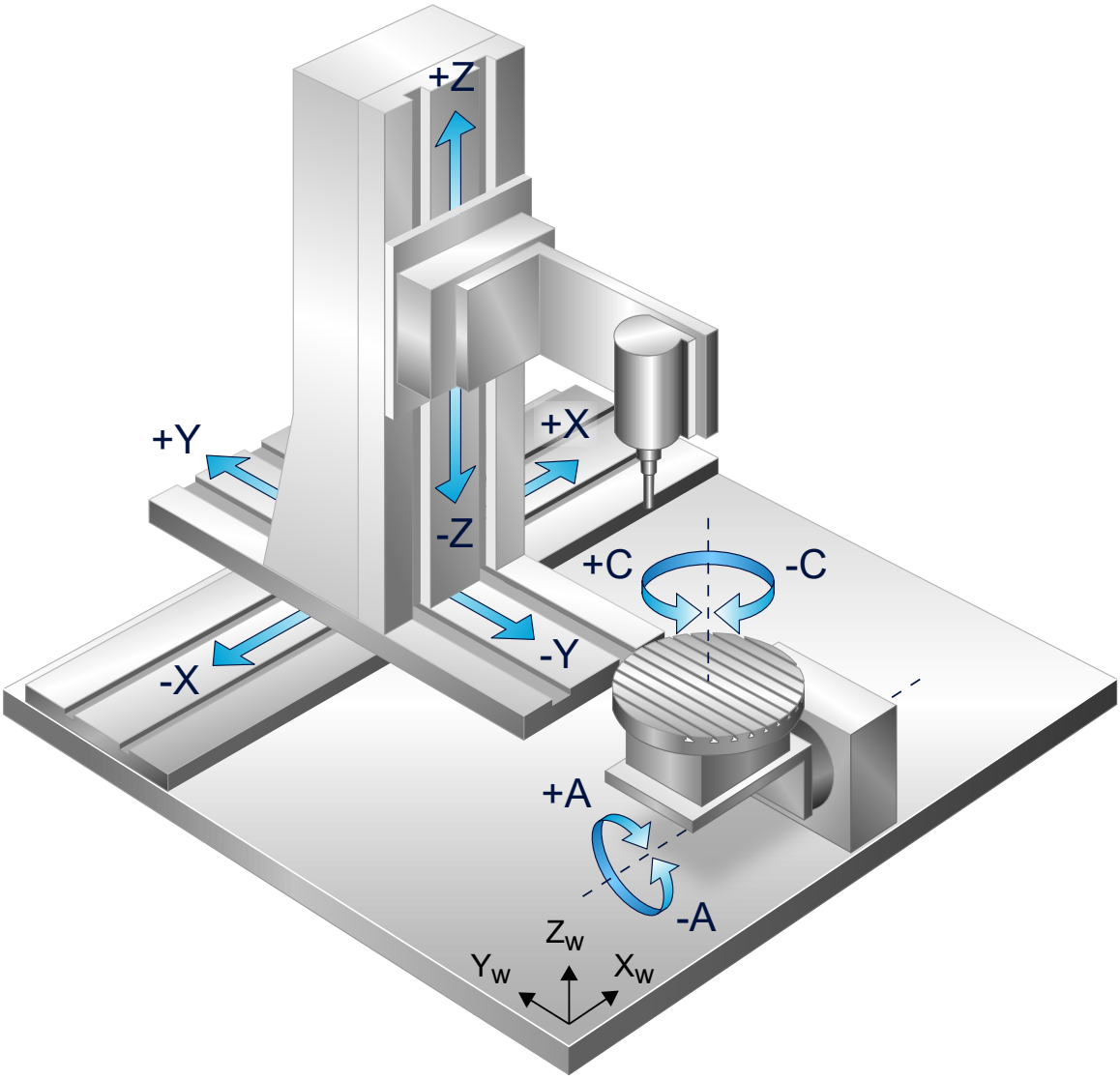


Fig. 2: Kinematics of 5-axis machine with AC workpiece table

With an AC machine, the linear axes are on the tool side and the rotary axes are on the workpiece side (as rotators in the tool table). If you imagine walking from the tool tip to the machine origin, the axes occur in this sequence

C, A, X, Y, Z

in this order. This is the axis sequence in the kinematic chain. However, it must be noted that the orientation vector must be multiplied by -1 for every axis that is located on the workpiece side. Here is a possible configuration of an AC machine:

Configuration of an AC machine

```
# Zero orientation of the tool
# Tool points in Z direction
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Zero position of the tool
# Tool rests at point (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# Define X axis (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
…
# Define Y axis (index 1)
kinematik[91].axis[1].type              1
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[1].orientation[1]    1
kinematik[91].axis[1].orientation[2]    0
…
# Define Z axis (index 2)
kinematik[91].axis[2].type              1
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[2].orientation[1]    0
kinematik[91].axis[2].orientation[2]    1
…


# define C axis (index 3)
kinematik[91].axis[3].type              2
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    -1 # inverted
…
# Define A axis (index 4)
kinematik[91].axis[4].type              2
kinematik[91].axis[4].orientation[0]    -1 # inverted
kinematik[91].axis[4].orientation[1]    0
kinematik[91].axis[4].orientation[2]    0
…
# Sequence in kin. chain: CAXYZ
kinematik[91].chain[0]                  3 # C axis
kinematik[91].chain[1]                  4 # A axis
kinematik[91].chain[2]                  0 # X axis
kinematik[91].chain[3]                  1 # Y axis
kinematik[91].chain[4] 2 # Z axis
```

4.1.3 3. Example of an AC machine with cardanic table

Example

AC machine with workpiece table

Universal Kinematics (ID91) can also be used to map kinematic transformations which are not contained in the kinematics library.

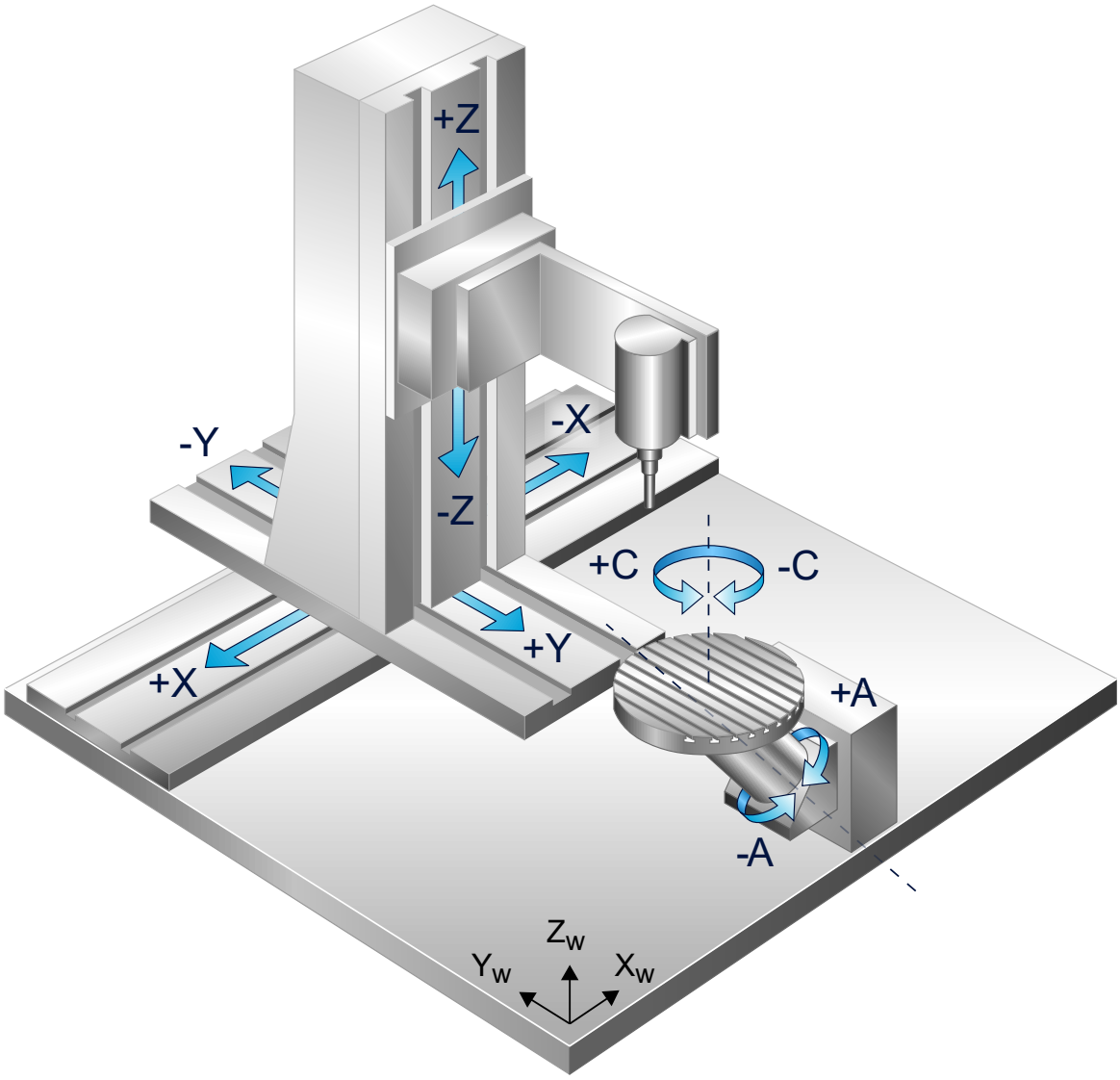


Fig. 3: Kinematics of 5-axis machine with AC workpiece table

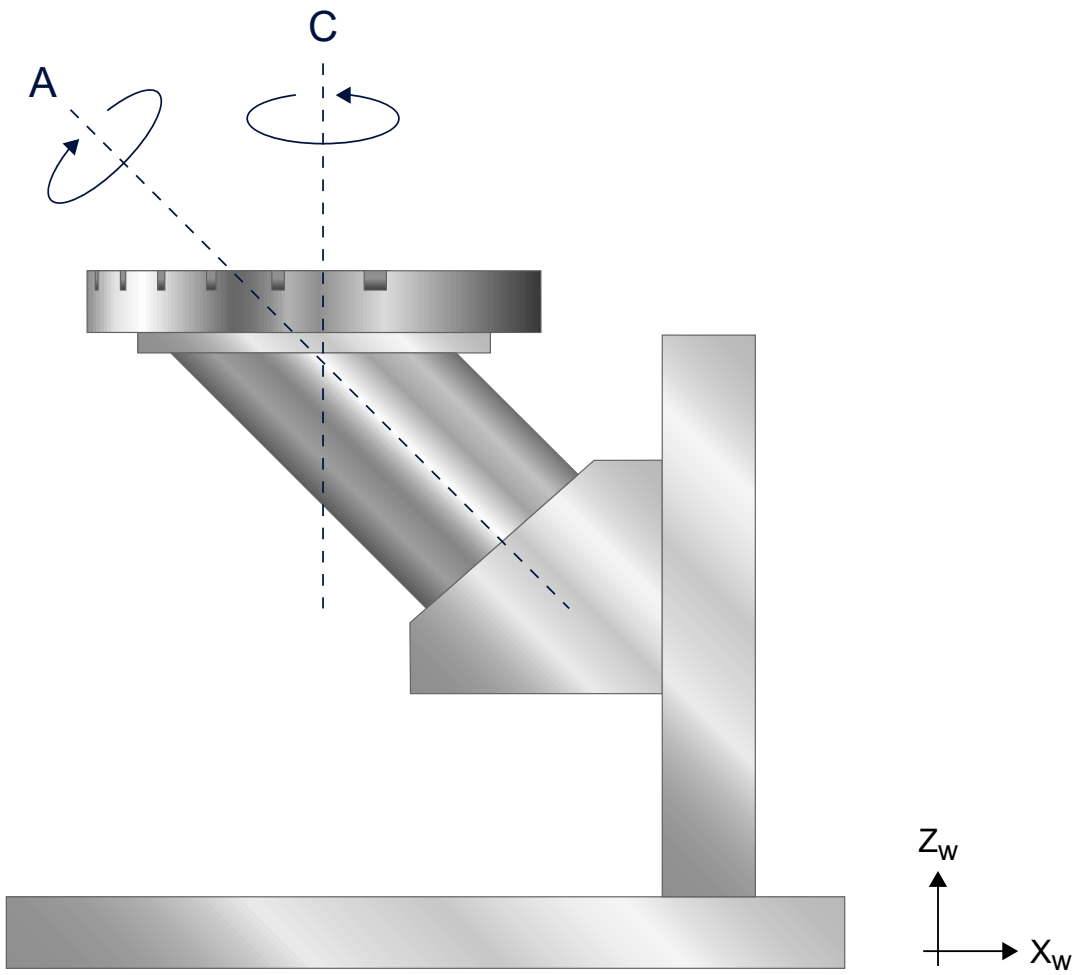


Fig. 4: Detailed view of the workpiece table

The depicted AC machine with workpiece table shows the linear axes on the tool side and the rotary axes on the workpiece side (as rotators in the tool table). The axes are located in the sequence along the path from the workpiece to the machine origin and from there to

C, A, X, Y, Z

the tool tip. This is the axis sequence in the kinematic chain. However, it must be noted that the orientation vector must be multiplied by -1 for every axis that is located on the workpiece side. This is a possible configuration of the cardanic AC machine:

Configuration of an AC machine with workpiece table:

```
# Zero orientation of the tool
# Tool points in Z direction
kinematik[91].zero_orientation[0]    0
kinematik[91].zero_orientation[1]    0
kinematik[91].zero_orientation[2]    1

# Zero position of the tool
# Tool rests at point (12000, -3200, 500)
kinematik[91].zero_position[0]       12000
kinematik[91].zero_position[1]       -3200
kinematik[91].zero_position[2]         500


# CA programming mode
kinematik[91].programming_mode          17

# Activate RTCP mode
kinematik[91].rtcp                      1

# Define X axis (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
#…

# Define Y axis (index 1)
kinematik[91].axis[1].type              1
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[1].orientation[1]    1
kinematik[91].axis[1].orientation[2]    0
#…

# Define Z axis (index 2)
kinematik[91].axis[2].type              1
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[2].orientation[1]    0
kinematik[91].axis[2].orientation[2]    1
#…

# define C axis (index 3)
kinematik[91].axis[3].type              2
kinematik[91].axis[0].orientation[0]    0
kinematik[91].axis[3].orientation[1]    0
kinematik[91].axis[3].orientation[2]    -1 # inverted
#…

# Define A axis (index 4) cardanic angle 45 degrees
kinematik[91].axis[4].type              2
kinematik[91].axis[4].orientation[0]    -1 # inverted
kinematik[91].axis[4].orientation[1]    0
kinematik[91].axis[4].orientation[2]    -1 # inverted
#…

# Sequence in kin. chain: CAXYZ
kinematik[91].chain[0]                  3 # C axis
kinematik[91].chain[1]                  4 # A axis
kinematik[91].chain[2]                  0 # X axis
kinematik[91].chain[3]                  1 # Y axis
kinematik[91].chain[4] 2 # Z axis
```

4.2 Sequence of axes in the kinematic chain

Attention

When a kinematic chain is created, there are restrictions resulting from kernel architecture, memory space limitations and algorithmic limits.

Number of axes

The following restrictions apply to the number of axes:

- The kinematics must have precisely 3 linear axes.
- The kinematics may have a maximum of 3 rotary axes.
- For complete transformations, the kinematics may have maximum 2 rotary axes.

This permits the simulation of 3-, 4-, 5- and 6-axis machines.

Axis indexing

The following restrictions apply to axis definition:

- The 3 linear axes must be defined at indices 0, 1 and 2.
- In RTCP mode, the rotary axes must be defined at the same indices where their counterparts are defined in the axis groups of the channel configuration. See P-CHAN-00006 and the **#SET AX** command. If there is an entry in the channel configuration, the C axis must be defined at index 3 in the axis definition of the kinematic chain. See code example:

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

4.3 Testing the configuration in the simulation

In a simulation, it is easy to test the configuration of the universal kinematics using an NC program if you have an approximate idea of the axis motion to be expected when moving with a transformation.

The NC program below can be used as a template which may have to be slightly modified.

In the NC program, the axis positions are logged to an output file by the NC command #MSG. (See Writing messages to a file (#MSG SAVE) and Definition of file names (#FILE NAME))

The logged axis positions must then be compared with the expected axis positions.

Programing Example

Type and source of risk

```
;-----------------------------------------------------------
;
;   First configure the universal kinematics in the NC
;   program instead of in the channel list.
;   This simplifies the test since then the lists need
;   not be reloaded every time.
;
;-----------------------------------------------------------
; Tool points in Z direction
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

; Zero position of the flange
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 axes XYZCA
N00210  V.G.KIN[91].NUMBER_OF_AXES              = 5

; X axis, translatory
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 axis, translatory
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 axis, translatory
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 axis
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 axis
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

; Axis sequence in the kinematic chain: 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

; Programming mode 12 or 13, P-CHAN-00112
N00670  V.G.KIN[91].PROGRAMMING_MODE            = 13

; Set RTCP mode
N00700  V.G.KIN[91].RTCP                        = 1

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

;-----------------------------------------------------------
;
;   Now the machine can be moved, various
;   tool lengths tried out, etc.
;
;    #CHANNEL INIT fetches the axis position
;   and outputs it with #MSG and V.A.ACS.ABS to a
;   text file (normally "message.txt").
;
;-----------------------------------------------------------

; Select transformation
N00870  #KIN ID [91]
N00880  #TRAFO ON

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

; Deselect transformation
N01020  #TRAFO OFF

; Change tool length
N01050  V.G.WZ_AKT.L = 99
```

```
; Select transformation
N01080  #KIN ID [91]
N01090  #TRAFO ON

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

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

; Deselect transformation
N01340  #TRAFO OFF

; end
N01370  M30
```

5 Programming modes

configuration

The programming mode can be set by:

- channel parameter P-CHAN-00112
- Variable V.G.KIN[91].PROGRAMMING_MODE
- Channel parameter kinematik[91].programming_mode.

Specifying multiple entries results in the following priorities:

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

5.1 Point-vector programming

Introduction

With point-vector programming, tool position and orientation are programmed directly.

- Position is programmed by the letters X, Y, Z.
- Orientation is programmed by the letters U, V, W.

Axis definition

Six axes must be defined in the channel parameters. These axes must be indexed in the order X, Y, Z, U, V, W.

Notice

No angles are programmed in point-vector programming, i.e. the corresponding RTCP flag is ignored. Angle transformation is complete, i.e. the axis angles are calculated from tool orientation.

5.2 Angle programming

Angles are programmed in all other programming modes. These angles are adopted as axis angles in RTCP mode (Rotation Tool Centre Point mode).

With complete transformation the programmed angles are used to determine tool orientation from which the axis angles are then calculated. Therefore, the way in which tool orientation is calculated must be specified from the programmed angles.

This also occurs with a kinematic chain which typically corresponds to the kinematic chain of the machine but without axis offsets. For example, the CA machine ID09 can be programmed in CA programming mode which includes the kinematic chain XYZCA. Universal Kinematics is capable of considering axis offsets to determine axis values.

5.2.1 Classic programming modes

CA mode (17)

Tool position and orientation are determined as a forward transformation of the following kinematic chain.

- Tool zero position is [0, 0, 0].
- Tool zero orientation is [0, 0, 1] (Z direction).
- Axis sequence XYZCA
- All axes pass through the zero point [0, 0, 0].

Other modes

By analogy, there are other modes AC, AB, BA, BC, CB. Tool zero orientation is always the orientation of the first rotary axis mentioned.

For more information, see P-CHAN-00112.

5.2.2 Compliance and direct programming

Compliance

Compliance programming extends the idea of the classic programming modes. The kinematic chain used is a copy of the defined kinematic chain of the machine where the axis offsets are removed.

For example, this permits the programming of a machine with an unusual axis position. The machine with transformation ID11 has a B axis tilted by 45°. If the kinematic chain for this machine is defined accordingly and if the compliance programming mode is selected, the programmed B angles are actually interpreted as angles in this tilted kinematics.

Notice

With compliance programming, the linear axes XYZ must lie on the the first three indices of the kinematik[].axis[] array.

Direct

With direct programming, the kinematic chain of the machine is used to calculate orientation without removing axis offsets.

For more information, see P-CHAN-00112.

6 Compensating axis position errors

Axis position error

On a real machine, axes are not normally in the assumed ideal position.

For example, the X axis may have a disrupted orientation vector, i.e. may not point towards (1, 0, 0) in the machine coordinate system but towards (0.99953, 0.0002, -0.0004). This value is obtainable by a compensation calculation in a measurement cycle, for example.

These deviations can be mapped in the kinematics. Instead of

```
# Define X axis
kinematik[91].axis[0].orientation[0]    1
kinematik[91].axis[0].orientation[1]    0
kinematik[91].axis[0].orientation[2]    0
…
```

set the following values:

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

Compensating axis position errors

Generally:

- Errors in linear axes can be compensated by correcting the Cartesian axis values.
- Errors in rotary axes can be compensated by correcting the Cartesian and rotary axis values.

Correcting axis values

When axis values are calculated, the kinematics considers and compensates for axis position errors. It determines axis values that then initiate correct tool positioning and orientation on the defective machine.

Attention

Exception:

In RTCP mode the kinematics adopts the programmed axis angles. Axis angle correction is therefore not possible.

However, Cartesian axis values are still corrected.

7 Avoiding pose changes

Description of problem

For machines with two or more rotary axes, there are usually axis angle solutions for setting programmed tool orientation.

During motion along a path, it is usually undesirable to set one and the same orientation by means of different axis angles ("pose change") since the machine may execute unexpected compensation motions which would lead to the risk of collision.

Avoidance

When selected, the kinematics selects a pose. At all later points in time, the kinematics always selects the solution of axis angles belonging to the same pose.

Attention

This strategy is unnecessary on 4-axis machines since there is at most one axis angle solution only.

The above mentioned strategy is applied to all 5-axis machines that do not run in RTCP mode. At most, changing a pose is only possible when transformation is deselected. However, pose selection the next time the transformation is selected cannot be influenced externally.

8 Simulating existing kinematics

With Universal Kinematics (ID91), most kinematic transformations can be simulated from the ISG kinematics library. For example, this can compensate incorrect axis positions, see Section “Programming modes”. [▸ 28]

The programming example below shows how to simulate a CA head kinematic (KIN9) with the universal kinematics.

The syntax shown in the examples is available as of Version V3.00; for Builds V2.11.2xxx the syntax must be converted analogously (see Programming or description of the kinematic chain [▸ 10]).

Example

Simulation of the kinematic ID 9 with universal kinematics

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

# Tool points in Z direction
trafo[1].zero_orientation[0]          0
trafo[1].zero_orientation[1]          0
trafo[1].zero_orientation[2]          1

# Zero position of the tool
trafo[1].zero_position[0]          54000
trafo[1].zero_position[1]        -395000
trafo[1].zero_position[2]        -950000

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

# X axis
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 axis
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 axis
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
```

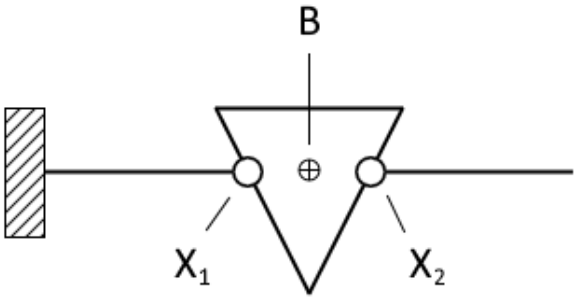
```
trafo[1].axis[2].point[2]              0

# C axis
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 axis
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

# Axis sequence in the kinematic chain
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

# CA programming mode
trafo[1].programming_mode             17

# RTCP mode, angles are directly programmed
trafo[1].rtcp                          1
```

9 Transformation between axis values and Cartesian coordinates

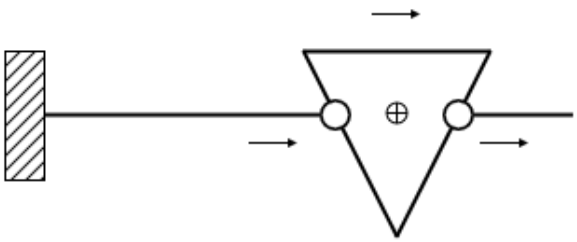
Normally, Universal Kinematics is used to describe a serial kinematic. Here, one axis includes all other axes that follow it in the kinematic chain.

However, it is possible to redirect the control of motions by using force-shaping mechanical components.

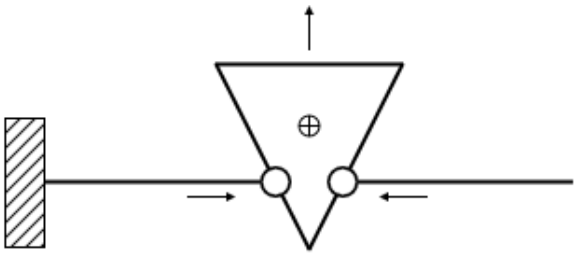
Example: Two parallel X axes generate an XZ movement across a spline. B is a reference point on the spline.



If both axes move the same distance to the right, then B does so too and the Z height of B is maintained.



f both axes move towards each other, the spline is pushed upwards.



Because of the symmetry of the spline, the Cartesian X coordinate of B is the average of both axis values plus an appropriate constant, i.e.

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

the Z coordinate of B, on the other hand, is dependent on the distance between the two axis values. Therefore, if appropriate constants are used for the rise of the spline leg and an optional offset, the following applies

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

If there is also a Y axis on the machine, the relationship between the Cartesian coordinates and axis values can be written in matrix form as follows:

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

The matrix and the offset vector can be specified in the channel parameter trafo[].linkage (P-CHAN-00295 [▸ 47]).

Example

Transformation between axis values and Cartesian coordinates

```
# Kinematic ID
trafo[0].id                    91
…

# Activate transformation
trafo[0].linkage_mode          1

# Matrix, first line
trafo[0].linkage[0][0]         0.5
trafo[0].linkage[0][1]         0.5
trafo[0].linkage[0][2]         0

# Matrix, second line
trafo[0].linkage[1][0]         0
trafo[0].linkage[1][1]         0
trafo[0].linkage[1][2]         1

# Matrix, third line, example m = 0.71
trafo[0].linkage[2][0]         0.71
trafo[0].linkage[2][1]         -0.71
trafo[0].inkage[2][2]          0

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

Notice

The parameters trafo[].axis[].orientation[] (P-CHAN-00292 [▶ 46]) describe the Cartesian system and are therefore retained.
In the example, the axis orientations for X1 and X2 are not defined in parallel. However, X1 points in the X direction and X2 in the Z direction.

10 Other examples

10.1 Implementation of CB head kinematic

Let's assume that the existing kinematic with ID 9, with a CA head kinematic, is to be converted to a CB head kinematic.

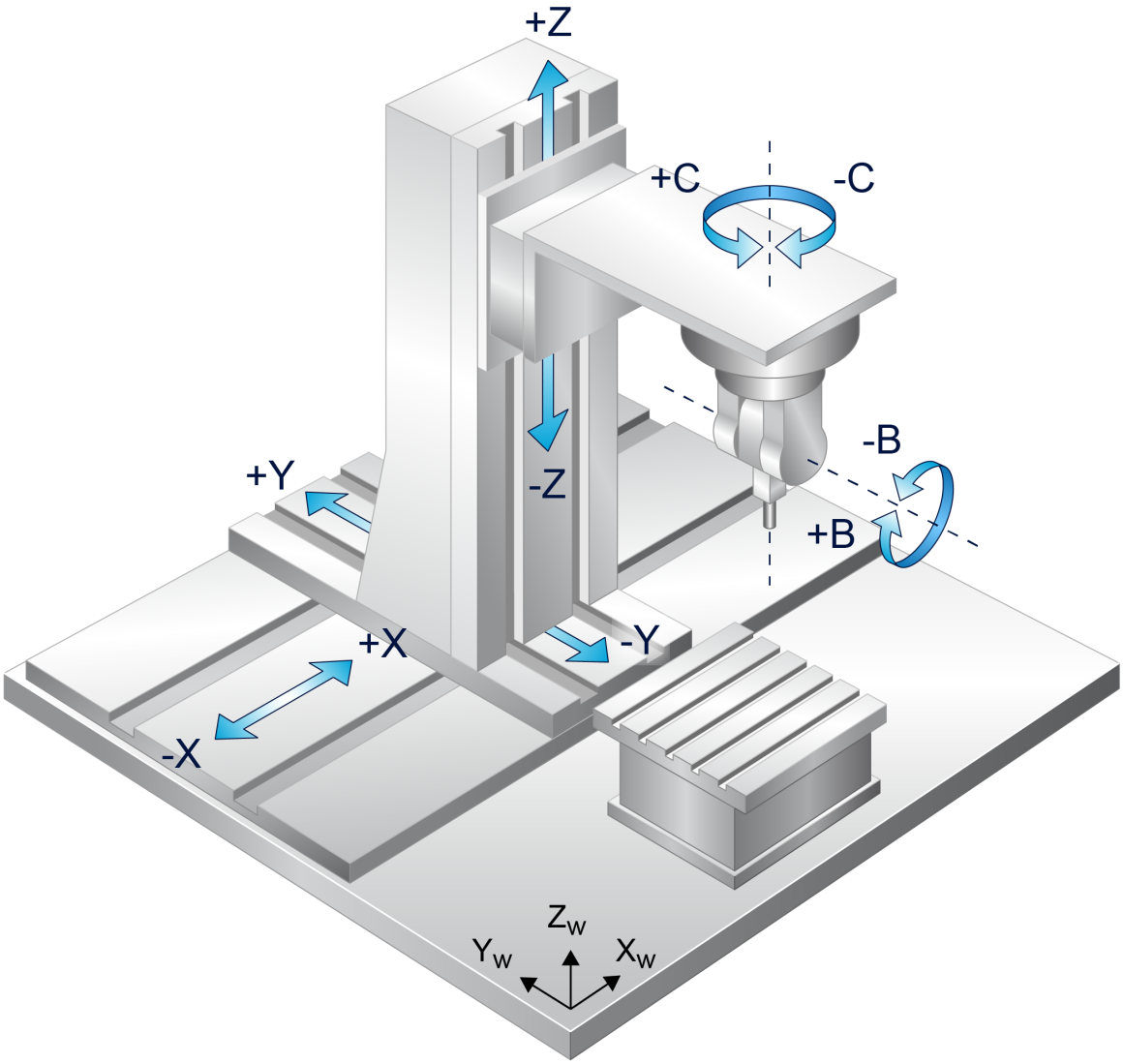


Fig. 5: CB head kinematic

Configuration of a CB machine

```
# CB machine: XYZCB
#
kinematik[91].rtcp                        1
# Programming mode CB->19
kinematik[91].programming_mode            19
kinematik[91].number_of_axes               5
#WZ Zero orientation of machine axes
kinematik[91].zero_orientation[0]         0
kinematik[91].zero_orientation[1]         0
kinematik[91].zero_orientation[2]         1
#WZ Zero position of machine axes
kinematik[91].zero_position[0]            0
kinematik[91].zero_position[1]            0
kinematik[91].zero_position[2]            0
#
# X axis type, position (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 axis type, position (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 axis type, position (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 axis type, position (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 axis type, position (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
#
# Sequence of kinematic chain: YXZCB
kinematik[91].chain[0]    0    #X axis
kinematik[91].chain[1]    1    # Y axis
kinematik[91].chain[2]    2    # Z axis
kinematik[91].chain[3]    3    # C axis
kinematik[91].chain[4]    4    # B axis
#
```

The machine parameters can also be defined in the NC program similar to the example of the CA machine [▸ 33].

11 Parameter

11.1 Overview

ID	Description
P-CHAN-00112	Orientation angle mode
P-CHAN-00262	Definition of kinematic ID = 91
P-CHAN-00285	Zero orientation of the tool
P-CHAN-00286	Zero position of the tool
P-CHAN-00287	Angle transformation
P-CHAN-00288	Programming mode
P-CHAN-00289	Number of axes
P-CHAN-00290	Axis sequence
P-CHAN-00291	Axis-specific data - Axis type
P-CHAN-00292	Axis-specific data - Axis orientation
P-CHAN-00293	Axis-specific data - Interpolation point on the axis
P-CHAN-00295	Transformation between axis values and Cartesian coordinates

11.2 Description

P-CHAN-00112	Mode of orientation angle programming for kinematic transformations
Description	With a complete transformation, orientation about the coordinate system axes can be programmed either by using an orientation vector with the three components U, V, W or by using three rotation angles A, B, C depending on the transformation type. Due to the additional degree of freedom provided by manual orientation, programming the rotation angles A, B, C is often the property of robot structures. The sequence of the three rotations about the assigned rotary axes X, Y, Z leads to the required target orientation or to the target effector coordinate system. If not otherwise defined, single rotations are executed in a mathematically positive direction about the coordinate system axes which are to be reset. The starting point is an axis sequence with the Cartesian axes X, Y, Z and the rotary axes A, B, C. The default assignment of rotations about the coordinate system axes is A -> X, B -> Y, C -> Z. This may deviate with special angle modes. Some kinematics use special sequences of rotation which are not listed here. In this case, switching using P-CHAN-00112 is not possible. P-CHAN-00112 has no measuring with standard five-axis kinematics. Special values can be entered in P-CHAN-00112 for Universal Kinematics (KIN-ID91).
Parameter	ori_rotation_angle
Data type	SGN16
Data range	-1: Programmed orientation axes are forwarded to kinematic transformation without any changes. Any Cartesian transformation which may be active with an active rotation has no influence on these orientation axes **Complete kinematics transformations** **3 degrees of freedom for orientation** 0: YPR (Yaw Pitch Role) sequence of rotation: 1st rotation about Z (C), 2nd rotation negative about Y´ (B), 3rd rotation about X´´ (A) (default) 1: Euler, order of rotation: 1. rotation about Z (C), 2nd rotation about Y´ (B), 3. rotation about Z´´ (C) 2: CBA, similar to YPR with positive B rotation and different axis assignment. Rotation about Z (A), 2nd rotation about Y´ (B), 3rd rotation about X´´ (C). -> A15 B-90 C20 (CBA) is identical to A20 B90 C15 (YPR). 3: CAB rotation sequence, 1st rotation about Z (C), 2nd rotation about X' (A), 3rd rotation about Y'' (B) (as of Build V3.1.3079.35) 4: CBA_STD, corresponds to CBA with a different axis assignment Rotation sequence: 1. rotation about Z(C), 2nd rotation about Y'(B), 3rd rotation about X''(A) -> A15 B90 C20 (CBA_STD) is identical to A20 B90 C15 (CBA) and A15 B-90 C20 (YPR). (as of V3.1.3079.9) 5: ABC rotation sequence, 1st rotation about X (A), 2nd rotation about Y´ (B), 3rd rotation about Z´´ (C). (as of Build V3.1.3079.35) **2 degrees of freedom for orientation (cf. KIN-ID 91)** 14: AB Rotation sequence: 1. rotation about X(A), 2nd rotation about Y' (B) (as of Build V3.1.3079.30) 15: BA Rotation sequence: 1. rotation about Y(B), 2nd rotation about X'(A) (as of Build V3.1.3079.30) 2nd **Universal kinematic transformations (KIN-ID 91):** 10: Point-vector programming. Tool orientation is programmed by the axes U, V, W. The vector [U, V, W] need not be normalised but neither may it be the zero vector. 11: Free programming. Not currently supported.

	12: Direct programming. The configured kinematic chain is used to calculate the tool position and orientation from programmed Cartesian coordinates and angles. 13: Conformal programming. Same as direct programming but without any axis offsets, shifts or direction flags. For example, allows the programming of 45° axis positions. 14: AB programming. 15: BA programming. 16: AC programming. 17: CA programming. 18: BC programming. 19: CB programming.
Dimension	----
Default value	0
Remarks	

P-CHAN-00262	**Define kinematic ID for multi-step transformations**
Description	The kinematic ID identifies the related transformation as an element of the data set of the kinematic parameters.
Parameter	trafo[j].id kin_step[i].trafo[j].id (multistep transformations)
Data type	UNS16
Data range	1 ... MAX(UNS16)
Dimension	----
Default value	0
Remarks	Parameter syntax as of V300 and higher

P-CHAN-00285	**Zero orientation of the tool (Universal Kinematics)**
Description	This parameter defines the tool orientation in the zero setting (vector X, Y, Z, tool direction).
Parameter	trafo[j].zero_orientation[k] where k = 0, 1, 2 kin_step[i].trafo[j].zero_orientation[k] (multistep transformations) kinematik[91].zero_orientation[k] (up to Version V2.11.28xx)
Data type	REAL64
Data range	----
Dimension	----
Default value	0
Remarks	

P-CHAN-00286	**Zero position of the tool (Universal Kinematics)**
Description	This parameter defines the tool position in the zero position (position X, Y, Z, home position).
Parameter	trafo[j].zero_position[k] where k = 0, 1, 2 kin_step[i].trafo[j].zero_position[k] (multistep transformations) kinematik[91].zero_position[k] (up to Version V2.11.28xx)
Data type	REAL64
Data range	----
Dimension	0.1 µm or 0.0001 inch
Default value	0
Remarks	

P-CHAN-00287	**Angle transformation (Universal Kinematics)**
Description	When angle programming mode is active (P-CHAN-00288), this parameter defines how the programmed angle is treated.
Parameter	trafo[j].rtcp kin_step[i].trafo[j].rtcp (multistep transformations) kinematik[91].rtcp (up to Version V2.11.28xx)
Data type	REAL64
Data range	0: Complete transformation: transforming angle within the range (-p, p] and output to the machine (default). 1: RTCP transformation: programmed angles are output directly to the machine.
Dimension	----
Default value	0
Remarks	

P-CHAN-00288	Programming mode (Universal Kinematics)
Description	The programming mode defines the tool orientation from the programmed values (point-vector mode or angle modes). Alternatively the mode can also be set by the channel parameter P-CHAN-00112 . A setting of the kinematics has priority over P-CHAN-00112.
Parameter	trafo[j].programming_mode kin_step[i].trafo[j].programming_mode (multistep transformations) kinematik[91].programming_mode (up to Version V2.11.28xx)
Data type	REAL64
Data range	10: Point-vector programming. Tool orientation is programmed by the axes U, V, W. The vector [U, V, W] need not be normalised but neither may it be the zero vector. 11: Free programming. Not currently supported. 12: Direct programming. The configured kinematic chain is used to calculate the tool position and orientation from programmed Cartesian coordinates and angles. 13: Conformal programming. Same as direct programming but without any axis offsets, shifts or direction flags. For example, enables the programming of 45° axis positions. 14: AB programming. 15: BA programming. 16: AC programming. 17: CA programming. 18: BC programming. 19: BC programming.
Dimension	----
Default value	0
Remarks	

P-CHAN-00289	Number of axes (Universal Kinematics)
Description	Parameter defines the number of axes of the kinematic chain.
Parameter	trafo[j].number_of_axes kin_step[i].trafo[j].number_of_axes (multistep transformations) kinematik[91].number_of_axes (up to Version V2.11.28xx)
Data type	REAL64
Data range	3 ... 6
Dimension	----
Default value	0
Remarks	The parameter is also available in multi-step transformations. Access to the parameter is: kin_step[i].trafo[j].number_of_axes

P-CHAN-00290	**Axis sequence (Universal Kinematics)**
Description	Parameter defines the axis sequence in the kinematic chain.
Parameter	trafo[j].chain[k] where k = 0 ... P-CHAN-00289 – 1 kin_step[i].trafo[j].chain[k] (multistep transformations) kinematik[91].chain[k] (up to Version V2.11.28xx)
Data type	REAL64
Data range	----
Dimension	----
Default value	0
Remarks	

P-CHAN-00291	**Axis type (Universal Kinematics)**
Description	Parameter defines the axis type.
Parameter	trafo[j].axis[k].type kin_step[i].trafo[j].axis[k].type (multistep transformations) kinematik[91].axis[k].type (up to Version V2.11.28xx)
Data type	REAL64
Data range	1: Translator 2: Rotator
Dimension	----
Default value	0
Remarks	

P-CHAN-00292	**Axis orientation (Universal Kinematics)**
Description	Parameter defines the direction vector (X, Y, Z, no zero vector) of the axis.
Parameter	trafo[j].axis[k].orientation[i] where i = 0, 1, 2 kin_step[i].trafo[j].axis[k].orientation[i] (multistep transformations) kinematik[91].axis[k].orientation[i] (up to on V2.11.28xx)
Data type	REAL64
Data range	----
Dimension	----
Default value	0
Remarks	

P-CHAN-00293	**Interpolation point on the axis (Universal Kinematics)**
Description	Parameter defines an interpolation point on the axis (position X, Y, Z, only relevant for rotary axes).
Parameter	trafo[j].axis[k].point[l] where l = 0, 1, 2 kin_step[i].trafo[j].axis[k].point[l] (multistep transformations) kinematik[91].axis[k].point[l] (up to Version V2.11.28xx)
Data type	REAL64
Data range	----
Dimension	0.1µm
Default value	0
Remarks	

P-CHAN-00295	**Transformation between axis values and Cartesian coordinates (Universal Kinematics)**
Description	Definition of a matrix and an offset vector to describe a linear transformation between axis values of the linear axes and Cartesian coordinates. $\begin{bmatrix} X_{cart} \\ Y_{cart} \\ Z_{cart} \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$ Rotary axes are not included in the calculation. The letters X, Y, Z stand here for the three linear axes of the Universal Kinematics in the order of their definition in the parameter trafo[].axis[] (P-CHAN-00293 [▸ 47]) is inactive. The matrix is defined in trafo[].linkage[0-2][0-2]. The first index specifies the line number; the second index specifies the column number; both are 0-based. The offset vector is defined in trafo[].linkage[0-2][3]. If the matrix is not invertible, the error ID 292010 is output. For configuration example, see [FCT-C27// Transformation between axis values and Cartesian coordinates [▸ 35]].
Parameter	trafo[j].linkage[k][l] where k = 0, 1, 2 and l = 0, 1, 2, 3 kin_step[i].trafo[j].linkage[k][l] (multistep transformations) kinematik[91].linkage[k][l] (up to Build V2.11.28xx)
Data type	REAL64
Data range	
Dimension	for the matrix: ---- for the offset vector: 0.1 µm
Default value	0
Remarks	

12 Appendix

12.1 Suggestions, corrections and the latest documentation

Did you find any errors? Do you have any suggestions or constructive criticism? Then please contact us at documentation@isg-stuttgart.de. The latest documentation is posted in our Online Help (DE/EN):

QR code link: https://www.isg-stuttgart.de/documentation-kernel/

The link above forwards you to:

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

Notice

Change options for favourite links in your browser;

Technical changes to the website layout concerning folder paths or a change in the HTML framework and therefore the link structure cannot be excluded.

We recommend you to save the above "QR code link" as your primary favourite link.

PDFs for 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

Keyword index

P

www.isg-stuttgart.de
support@isg-stuttgart.de