SIEMENS

SINUMERIK

SINUMERIK 840D sl Transformations

Function Manual

Valid for:

Control system
SINUMERIK 840D sl / 840DE sl

Software
CNC software version 4.92

06/2019
A5E47435470B AA

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: A5E47435470B AA
℗ 06/2019 Subject to change

Preface

SINUMERIK documentation

The SINUMERIK documentation is organized into the following categories:

- General documentation/catalogs
- User documentation
- Manufacturer/service documentation

Additional information

You can find information on the following topics at the following address (https://support.industry.siemens.com/cs/de/en/view/108464614):

- Ordering documentation/overview of documentation
- Additional links to download documents
- Using documentation online (find and search in manuals/information)

If you have any questions regarding the technical documentation (e.g. suggestions, corrections), please send an e-mail to the following address (mailto:docu.motioncontrol@siemens.com).

mySupport/Documentation

At the following address (https://support.industry.siemens.com/My/ww/en/documentation), you can find information on how to create your own individual documentation based on Siemens' content, and adapt it for your own machine documentation.

Training

At the following address (http://www.siemens.com/sitrain), you can find information about SITRAIN (Siemens training on products, systems and solutions for automation and drives).

FAQs

You can find Frequently Asked Questions in the Service&Support pages under Product Support (https://support.industry.siemens.com/cs/de/en/ps/faq).

SINUMERIK

You can find information about SINUMERIK at the following address (http://www.siemens.com/sinumerik).

Target group

This publication is intended for:

- Project engineers
- Technologists (from machine manufacturers)
- System startup engineers (Systems/Machines)
- Programmers

Benefits

The function manual describes the functions so that the target group knows them and can select them. It provides the target group with the information required to implement the functions.

Standard version

This documentation only describes the functionality of the standard version. Extensions or changes made by the machine tool manufacturer are documented by the machine tool manufacturer.

Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

Further, for the sake of simplicity, this documentation does not contain all detailed information about all types of the product and cannot cover every conceivable case of installation, operation or maintenance.

Note regarding the General Data Protection Regulation

Siemens observes standard data protection principles, in particular the principle of privacy by design. That means that

this product does not process / store any personal data, only technical functional data (e.g. time stamps). If a user links this data with other data (e.g. a shift schedule) or stores personal data on the same storage medium (e.g. hard drive) and thus establishes a link to a person or persons, then the user is responsible for ensuring compliance with the relevant data protection regulations.

Technical Support

Country-specific telephone numbers for technical support are provided in the Internet at the following address (https://support.industry.siemens.com/sc/ww/en/sc/2090) in the "Contact" area.

Information on the structure and contents

Structure

This Function Manual is structured as follows:

- Inner title (page 3) with the title of the Function Manual, the SINUMERIK controls as well as the software and the version for which this version of the Function Manual is applicable and the overview of the individual functional descriptions.
- Description of the functions in alphabetical order (e.g. A2, A3, B1, etc.)
- Appendix with:
  - List of abbreviations
  - Documentation overview
- Index of terms

Note

For detailed descriptions of data and alarms see:

- For machine and setting data:
  Detailed machine data description
- For NC/PLC interface signals:
  NC Variables and Interface Signals List Manual
- For alarms:
  Diagnostics Manual

Notation of system data

The following notation is applicable for system data in this documentation:

Signal/Data	Notation	Example
NC/PLC interface signals	... NC/PLC interface signal: <signal address> (<signal name>)	When the new gear stage is engaged, the following NC/PLC interface signals are set by the PLC program: DB31, ... DBX16.0-2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)
Machine data	... machine data: <Type><Number> <Complete Designator> (<Meaning>)	Master spindle is the spindle stored in the machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (position of deletion of the master spindle in the channel)
Setting data	... setting data: <Type><Number> <Complete Designator> (<Meaning>)	The logical master spindle is contained in the setting data: SD42800 $SC_SPIND_ASSIGN_TAB[0] (spindle number converter)

Quantity structure

Explanations concerning the NC/PLC interface are based on the absolute maximum number of the following components:

- Mode groups (DB11)
- Channels (DB21, etc.)
- Axes/spindles (DB31, etc.)

Data types

The control provides the following data types that can be used for programming in part programs:

Type	Meaning	Range of values
INT	Signed integers	-2.147.483.648 ... +2.147.483.647
REAL	Numbers with decimal point	≈ ±5.0*10^{-324} … ≈ ±1.7*10^{+308}
BOOL	Boolean values	TRUE (≠0) , FALSE (0)
CHAR	ASCII characters and bytes	0 ... 255 or -128 ... 127
STRING	Character string, null-terminated	Maximum of 400 characters + /0 (no special characters)
AXIS	Axis names	All axis names available in the control system
FRAME	Geometrical parameters for moving, rotating, scaling, and mirroring	---

Arrays

Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.

Example: `DEF INT ARRAY[2, 3, 4]`

Number systems

The following number systems are available:

- Decimal: `DEF INT number = 1234` or `DEF REAL number = 1234.56`
- Hexadecimal: `DEF INT number = 'H123ABC'`
- Binary: `DEF INT number = 'B10001010010'`

Querying REAL variables

We recommend that querying REAL or DOUBLE variables in NC programs and synchronized actions is programmed as limit value evaluation.

Example: Querying the actual value of an axis for a specific value

```
Program code                                                                    Comment
DEF REAL AXPOS = 123.456
IF ($VA_IM[<axis>] - 1ex-6) <= AXPOS <= ($VA_IM[<axis>] + 1ex-6)                ; actual position
   ...                                                                          == AXPOS
```

Program code	Comment
ELSE	
...	<> AXPOS
ENDIF	

Table of contents

Fundamental safety instructions

1

1.1 General safety instructions

⚠ WARNING

Danger to life if the safety instructions and residual risks are not observed

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠ WARNING

Malfunctions of the machine as a result of incorrect or changed parameter settings

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

1.2 Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. Application examples do not represent specific customer solutions, but are only intended to provide support for typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

1.3 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Products and solutions from Siemens constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. using firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that can be implemented, please visit:

Industrial security (https://www.siemens.com/industrialsecurity)

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they become available, and that only the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at:

Industrial security (https://www.siemens.com/industrialsecurity)

Further information is provided on the Internet:

Industrial Security Configuration Manual (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ WARNING

Unsafe operating states resulting from software manipulation

Software manipulations, e.g. viruses, Trojans, or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

- Keep the software up to date.
- Incorporate the automation and drive components into a holistic, state-of-the-art industrial security concept for the installation or machine.
- Make sure that you include all installed products into the holistic industrial security concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- On completion of commissioning, check all security-related settings.
- Protect the drive against unauthorized changes by activating the "Know-how protection" converter function.

M1: Kinematics transformation

2

2.1 TRANSMIT face end transformation (option)

2.1.1 Function

2.1.1.1 Introduction

Note

The "TRANSMIT and peripheral surface transformation" option that is under license is required for the function "End face transformation (TRANSMIT)."

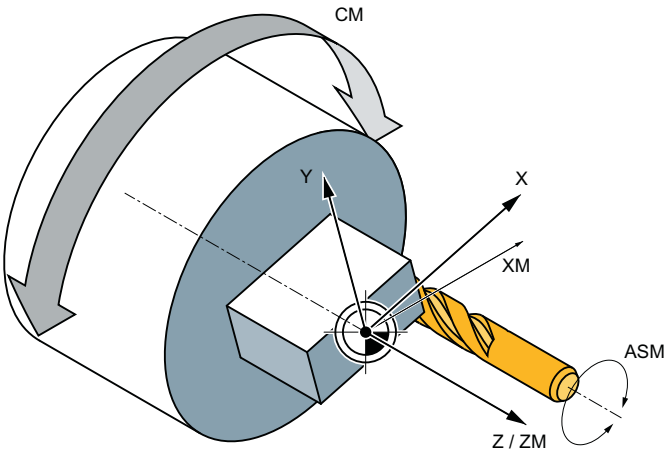
The TRANSMIT transformation permits end face machining (drill holes, contours) on turning machines.

A Cartesian coordinate system can be used to program these machining operations.

The controller transforms the programmed traversing movements of the Cartesian coordinate system to the traversing movements of the real machine axes.

Standard case:

- Rotary axis
- Infeed axis, perpendicular to rotary axis
- Longitudinal axis, parallel to rotary axis
- The linear axes are perpendicular to one another.



X, Y, Z	Geometry axes
CM	Machine axis: Rotary axis
XM	Machine axis: Linear axis, perpendicular to rotary axis
ZM	Machine axis: Linear axis, parallel to rotary axis
ASM	Machine axis: Main spindle

Other options:

- A tool center offset relative to the turning center is permitted.
- The tool center point path can pass through the turning center point of the rotary axis.
- The rotary axis does not need to be a modulo axis.

Note

For active transformation, the names of the involved machine, channel and geometry axes are different:

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (geometry axis name)

2.1.1.2 Machining options

The TRANSMIT transformation has a pole at the zero point of the TRANSMIT plane. The pole is at the intersection of the radial linear axis and the rotary axis. In the vicinity of the pole, small positional changes in the geometry axes generally result in large changes in position in the machine rotary axis. The only exceptions are linear motions into or through the pole.

A tool center point path through the pole does not cause the parts program to be aborted. There are no restrictions with respect to programmable traversing commands or active tool radius compensations. Nevertheless, workpiece processing operations close to the pole are not

recommended since these may require sharp feedrate reductions to prevent overloading of the rotary axis.

New features

A pole is said to exist if the line described by the tool center point intersects the turning center of the rotary axis.

The following cases are covered:

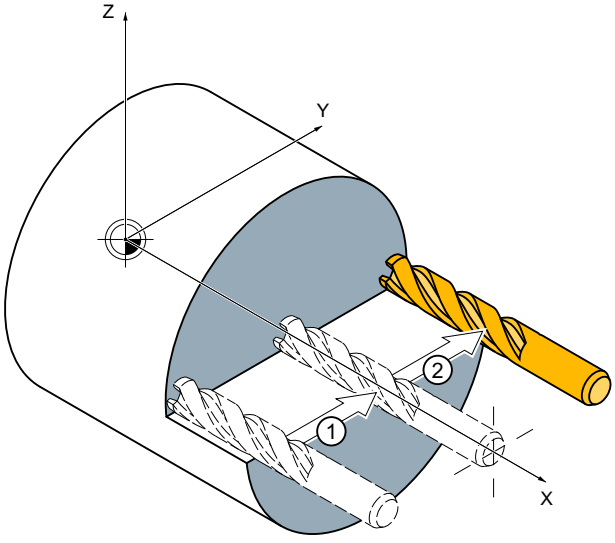
- Under what conditions and by what methods the pole can be traversed
- The response in pole vicinity
- The response with respect to working area limitations
- Monitoring of rotary axis rotations over 360°.

Pole crossing

The pole can be traversed by two methods:

- Traversal along linear axis
- Traversal into pole with rotation of rotary axis in pole

Traversal along linear axis

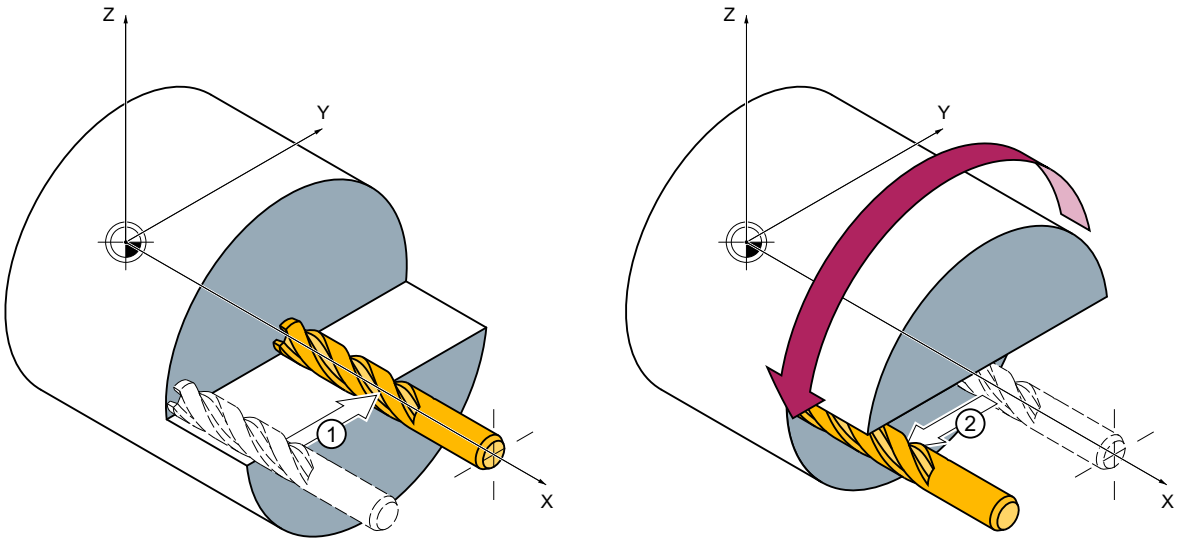


① Travel into the pole

② Travel out of the pole

Figure 2-1 Traversal of x axis through pole

Rotation in pole



① Travel into the pole
② Travel out of the pole

Figure 2-2 Traversal of x axis into pole (a), rotation (b), exit from pole (c)

Selection of method

The method must be selected according to the capabilities of the machine and the requirements of the part to be machined. The method is selected by machine data:

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2

The first MD applies to the first TRANSMIT transformation in the channel and the second MD correspondingly to the second TRANSMIT transformation.

VALUE	Meaning
0	Pole crossing The tool center point path (linear axis) must cross the pole on a continuous path.
1	Rotation around the pole. The tool center point path must be restricted to a positive traversing range of the linear axis (in front of turning center).
2	Rotation around the pole. The tool center point path must be restricted to a negative traversing range of the linear axis. (behind turning center).

Special features relating to pole traversal

The method of pole traversal along the linear axis may be applied in the AUTOMATIC and JOG modes.

Behavior:

Table 2-1 Traversal of pole along the linear axis

Mode	Status	Response
AUTOMATIC	All axes involved in the transformation are moved synchronously. TRANSMIT active.	High-speed pole traversal
	Not all of the axes participating in the transformation are moved synchronously (e.g. positioning axis). TRANSMIT not active	Traversal of pole at creep speed
	An applied DRF (external zero offset) does not interfere with the operation. Servo errors may occur close to the pole during application of a DRF.	Abortion of processing, alarm
JOG	-	Traversal of pole at creep speed

Special features relating to rotation in pole

Precondition: This method is only effective in the AUTOMATIC mode.

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

Value: 1 Linear axis remains within positive traversing range

Value: 2 Linear axis remains within negative traversing range

In the case of a contour that would require the pole to be traversed along the tool center point path, the following three steps are taken to prevent the linear axis from traversing in ranges beyond the turning center:

Step	Action
1	Linear axis traverses into pole
2	Rotary axis turns through 180°, the other axes involved in the transformation remain stationary.
3	Execution of remaining block. The linear axis now exits from the pole again.

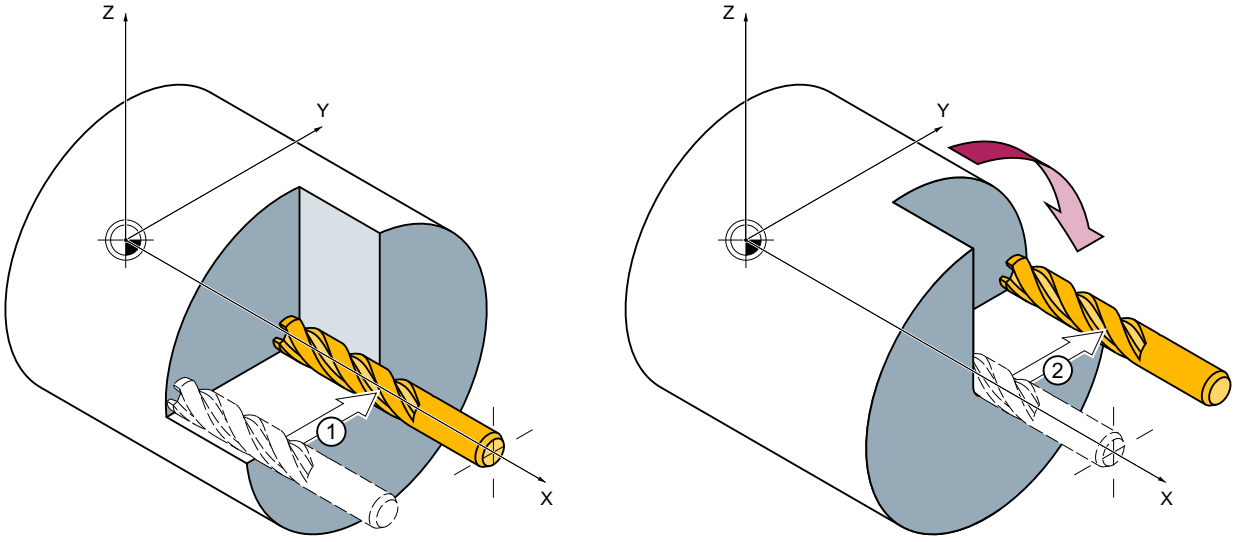
In JOG mode, the motion stops in the pole. In this mode, the axis may exit from the pole only along the path tangent on which it approached the pole. All other motion instructions would require a step change in the rotary axis position or a large machine motion in the cases of minimum motion instructions. They are rejected with alarm 21619.

Traversal close to pole

If a tool center point traverses past the pole, the control system automatically reduces the feedrate and path acceleration rate such that the settings of the machine axes (MD32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the pole, the greater the reduction in the feedrate.

Tool center point path with corner in pole

A tool center point path which includes a corner in the pole will not only cause a step change in axis velocities, but also a step change in the rotary axis position. These cannot be reduced by decelerating.



① Travel into the pole

② Travel out of the pole

Figure 2-3 Pole traversal

Requirements:

AUTOMATIC mode,

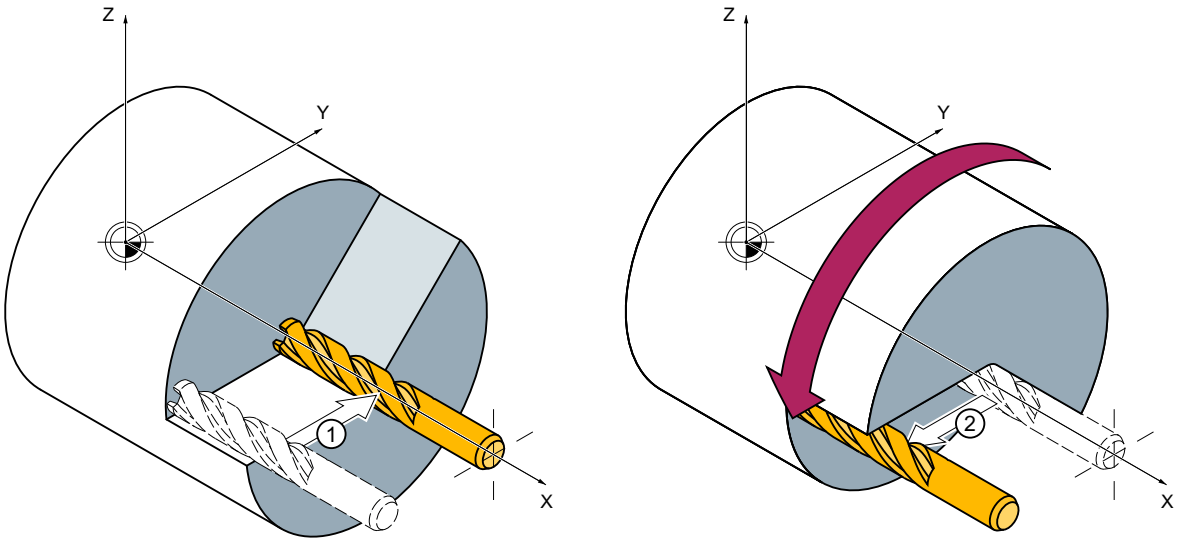
MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 0

The control system inserts a traversing block at the step change point. This block generates the **smallest possible rotation** to allow processing of the contour to continue.

Corner without pole traversal



① Travel into the pole
② Travel out of the pole

Figure 2-4 Processing on one pole side

Requirements:

AUTOMATIC mode,

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

The control system inserts a traversing block at the step change point. This block generates the **necessary rotation** so that processing of the contour can continue on the same side of the pole.

Transformation selection in pole

If the processing operation must continue from a position on the tool center path which corresponds to the pole of the activated transformation, then an exit from the pole is specified for the new transformation.

Actual

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1= 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2= 0

is set (pole transition), then a rotation **as small as possible** is generated at the beginning of the block originating in the pole. Depending on this rotation, the axis then traverses either in front of or behind the turning center.

For

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1= 1

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2= 1

processing is done **before** the rotational center point (linear axis in positive traversing range), for

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1= 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2= 2

behind the rotational center point (linear axis in negative traversing range).

Transformation selection outside pole

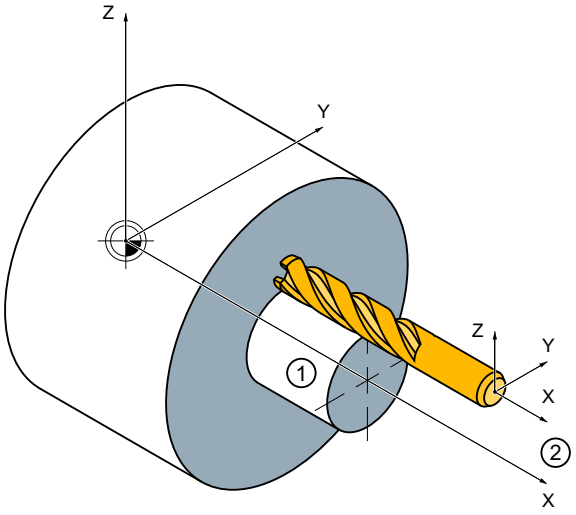
The controller traverses the axes participating in the transformation without evaluating the machine data MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_<t>. <t> = 1 marks the first TRANSMIT transformation and <t> = 2 marks the second TRANSMIT transformation in the channel.

2.1.1.3 Working area limitations

Starting point

When TRANSMIT is active, the pole is replaced by a working area limitation if the tool center point cannot be positioned at the turning center of the rotary axis involved in the transformation. This is the case when the axis perpendicular to the rotary axis (allowing for tool offset) is not positioned on the same radial plane as the rotary axis or if both axes are positioned mutually at an oblique angle. The distance between the two axes defines a cylindrical space in the BCS in which the tool cannot be positioned.

The illegal range cannot be protected by the software limit switch monitoring function since the traversing range of the machine axes is not affected.



① Unmachinable cylinder - working area limitation

② Axis distance

Figure 2-5 Working area limitation based on offset linear axis

Traverse into working area limitation

Any motion that leads into the working area limitation is rejected with alarm 21619. Any corresponding parts program block is not processed. The control system stops processing at the end of the preceding block.

If the motion cannot be foreseen promptly enough (JOG modes, positioning axes), then the control stops at the edge of the working area limitation.

Response close to working area limitation

If a tool center point traverses past the prohibited range, the control system automatically reduces the feedrate and path acceleration rate such that the settings in the machine axes (MD32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the working area limitation, the greater the reduction in the feedrate may be.

2.1.1.4 Overlaid motions with TRANSMIT

The control system cannot predict overlaid motions. However, these do not interfere with the function provided that they are very small (e.g. fine tool offset) in relation to the current distance from the pole (or from working area limitation). With respect to axes that are relevant for the transformation, the transformation monitors the overlaid motion and signals any critical quantity by alarm 21618. This alarm indicates that the block-related velocity planning function no longer adequately corresponds to the actual conditions on the machine. When the alarm is output, the conventional, non-optimized online velocity monitor is therefore activated. The preprocessing routine is re-synchronized with the main run by a REORG generated internally in the control.

Alarm 21618 should be avoided whenever possible since it indicates a state that can lead to axis overload and thus abortion of parts program processing.

2.1.1.5 Monitoring of rotary axis rotations over 360º

The positions of the rotary axis are ambiguous with respect to the number of rotations. The control breaks down blocks containing several rotations around the pole into sub-blocks.

This subdivision must be noted with respect to parallel actions (e.g. output of auxiliary functions, block-synchronized positioning axis motions) since the programmed block end is no longer relevant for synchronization, but the end of the first sub-block.

Further information
Function Manual Basic Functions; Auxiliary function outputs to PLC
Function Manual Synchronized actions

In single block operation the control system machines individual blocks explicitly. Otherwise the sub-blocks are traversed with Look Ahead just like a single block. A limitation of the rotary axis setting range is monitored by the software limit switch monitoring function.

2.1.2 Parameterization

2.1.2.1 Overview

Machine data: Transformation data in general

The following machine data is used to define transformation data sets in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<n> (definition of the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_AXES_IN_<n> (axis assignment for the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_GEOAX_ASSIGN_TAB_<n> (assignment of geometry axes to channel axes for transformation <n>)
- MD2xxxx $MC_TRAFO_INCLUDES_TOOL_<n> (tool handling with active nth transformation)

where <n> = 1, 2, 3, ... max. number of transformation data sets

For TRANSMIT (type 256 or 257), no more than two transformation data sets may be parameterized in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<x> = <TRANSMIT type>
- MD2xxxx $MC_TRAFO_TYPE_<y> = <TRANSMIT type>

Machine data: TRANSMIT transformation

A TRANSMIT transformation is parameterized using the following machine data:

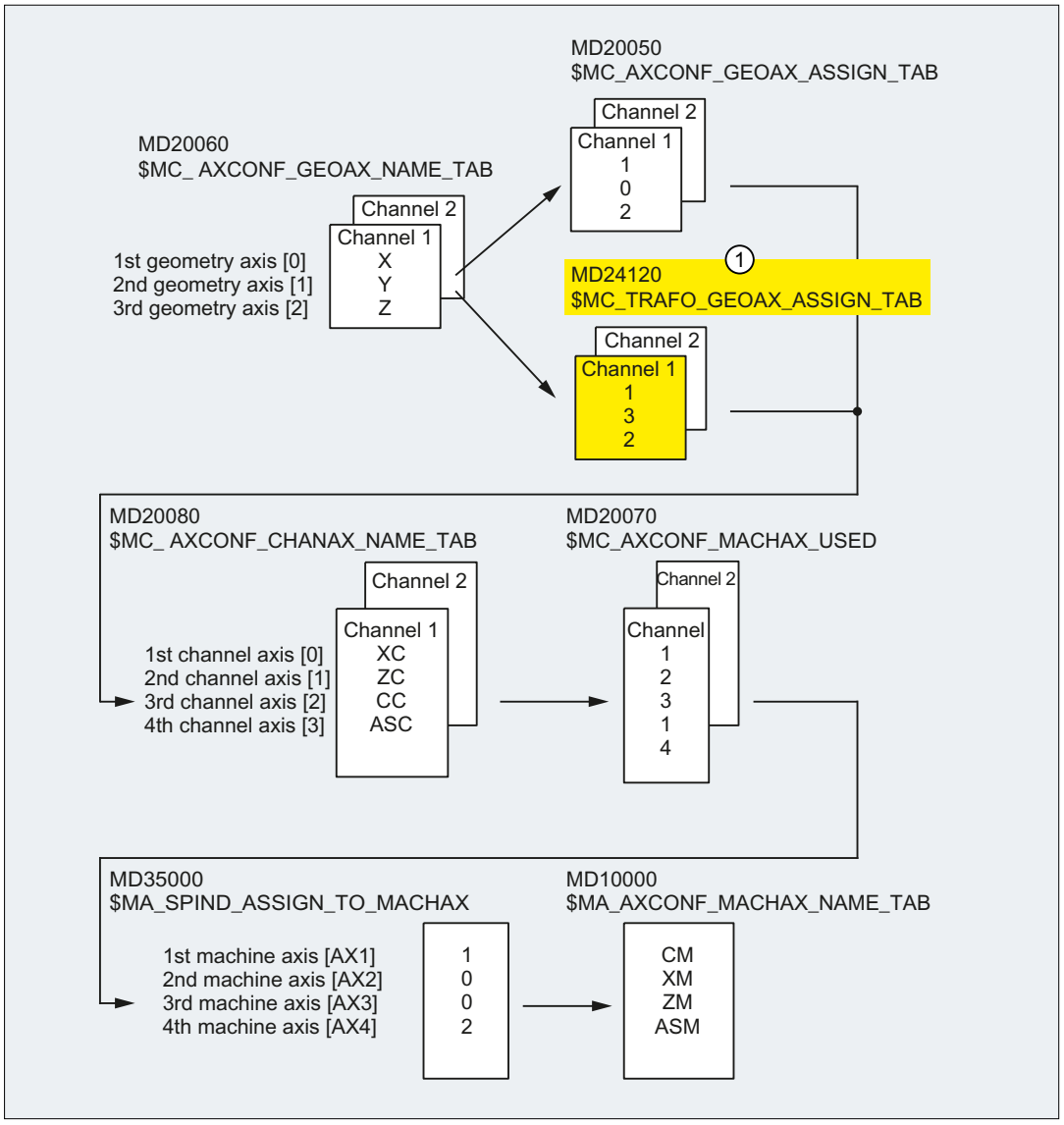
- MD2xxxx $MC_TRANSMIT_ROT_AX_OFFSET_<n> (offset of the rotary axis)
- MD2xxxx $MC_TRANSMIT_ROT_AX_FRAME_<n> (rotary axis offset)
- MD2xxxx $MC_TRANSMIT_ROT_SIGN_IS_PLUS_<n> (sign of the rotary axis)

- MD2xxxx $MC_TRANSMIT_BASE_TOOL_<n> (vector of the base tool)
- MD2xxxx $MC_TRANSMIT_POLE_SIDE_FIX_<n> (limitation of the working area in front of/ behind the pole)

where <n> = 1, 2 (TRANSMIT data set number)

2.1.2.2 Axis configuration

The following shows an axis configuration that is typical of TRANSMIT.



① Effective if TRANSMIT is active.

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "XM"

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "ZM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ASM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "XC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "CC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "ASC"

Assignment of geometry axes to channel axes

TRANSMIT **not** active

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geometry axis → 1st channel axis XC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2 (3rd geometry axis → 2nd channel axis ZC)

TRANSMIT active

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1 (1st transformation geometry axis → 1st channel axis XC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 3 (2nd transformation geometry axis → 3rd channel axis CC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 2 (3rd transformation geometry axis → 2nd channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 2 (1st channel axis → 2nd machine axis XM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 3 (2nd channel axis → 3rd machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 1 (3rd channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 4 (4th channel axis → 4th machine axis ASM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 2 (spindle)

2.1.2.3 Specific settings

One rotary and one linear axis: TRAFO_TYPE = 256

The transformation type 256 must be set for TRANSMIT with a rotary and a linear axis:

$MC_TRAFO_TYPE_<n> = 256
where <n> = 1, 2, ... max. number of transformations

Input axes of the transformation: $MC_TRAFO_AXES_IN_<n>

Those channel axes are specified in the machine data on which the axes of the transformation Cartesian coordinate system are to be mapped:

$MC_TRAFO_AXES_IN_<n>[<index>] = <channel axis number>
where <n> = 1, 2, ... max. number of transformations

<Index>	Meaning
0	Linear axis, perpendicular to rotary axis
1	Rotary axis
2	Linear axis, parallel to rotary axis

The channel axis numbers must relate to the axis sequence defined with $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>.

One rotary and two linear axes: TRAFO_TYPE = 257

The transformation type 257 must be set for TRANSMIT with a rotary and two linear axes:

$MC_TRAFO_TYPE_<n> = 257
where <n> = 1, 2, ... max. number of transformations

The second linear axis must be oriented perpendicular to the plane clamped by the rotary and the linear axis. The second linear axis for TRANSMIT is used exclusively for tool correction.

Input axes of the transformation: $MC_TRAFO_AXES_IN_<n>

Those channel axes are specified in the machine data on which the axes of the transformation Cartesian coordinate system are to be mapped:

$MC_TRAFO_AXES_IN_<n>[<index>] = <channel axis number>
where <n> = 1, 2, ... max. number of transformations

<Index>	Meaning
0	Linear axis, perpendicular to rotary axis
1	Rotary axis
2	Linear axis, parallel to rotary axis
3	Linear axis perpendicular to the axes from index 0 and 1

The channel axis numbers must relate to the axis sequence defined with $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>.

Rotary axis offset: TRANSMIT_ROT_AX_OFFSET

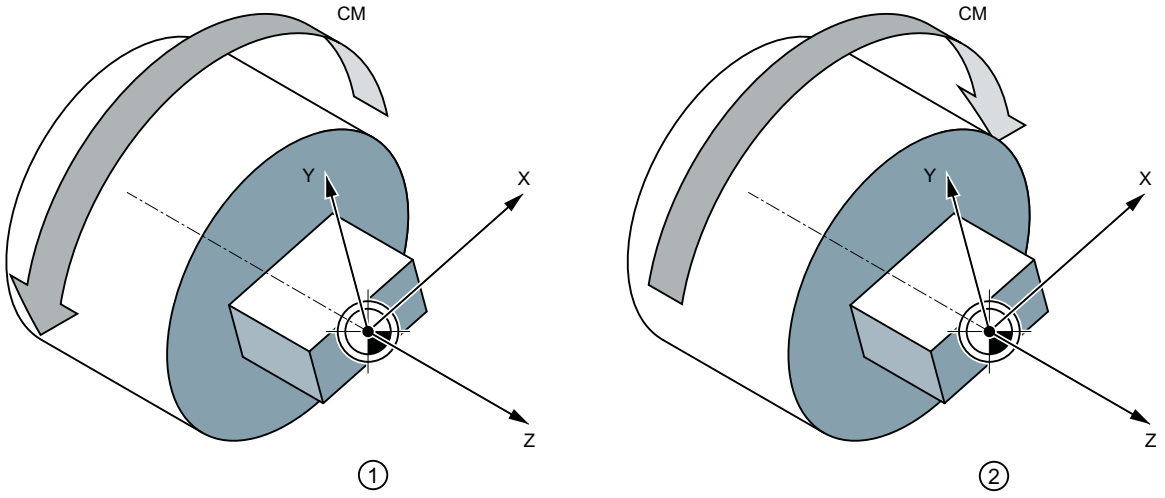
If the rotary axis zero point does not match the rotary axis zero position when the TRANSMIT transformation is active, the angular difference must be entered as an offset in the machine data:

$MC_TRANSMIT_ROT_AX_OFFSET_<t> = <angular difference>
where <t> = 1, 2

Direction of rotation: TRANSMIT_ROT_SIGN_IS_PLUS

TRANSMIT must be informed of the rotary axis direction of rotation with the following machine data:

- The direction of rotation of the rotary axis is positive with regard to TRANSMIT if the rotary axis rotates counterclockwise (in relation to the X/Y plane looking at the Z axis), when traversing in the positive direction.
- The rotary axis direction of rotation with regard to TRANSMIT is negative if it rotates clockwise when traversing in the positive direction



① Positive direction of rotation
② Negative direction of rotation

Figure 2-6 Rotary axis direction of rotation

$MC_TRANSMIT_ROT_SIGN_IS_PLUS_<t> = <direction of rotation>
where <t> = 1, 2

<Direction of rotation>	Meaning
0	Negative rotary axis direction of rotation ⇒ internal sign reversal
1	Positive rotary axis direction of rotation ⇒ no internal sign reversal

Position of the tool zero: TRANSMIT_BASE_TOOL

The position of the tool zero is specified in relation to the origin of the effective Cartesian coordinate system for TRANSMIT:

- MD24920 $MC_TRANSMIT_BASE_TOOL_<t>[0] = <Offset in X>
- MD24920 $MC_TRANSMIT_BASE_TOOL_<t>[1] = <Offset in Y>
- MD24920 $MC_TRANSMIT_BASE_TOOL_<t>[2] = <Offset in Z>

with <t> = 1, 2

Switchable geometry axes

When the `GEOAX()` geometry axes are switched, the parameterized M function is output to the NC/PLC interface:

- MD22534 $MC_TRAFO_CHANGE_M_CODE = <M function>

Note

The values 0 to 6, 17 and 30 are **not** output.

Further information:

Function Manual Basic Functions; Coordinate systems, axis types, axis configurations, workpiece-related actual-value system, external zero offset

2.1.3 Programming

The front face transformation (TRANSMIT) is activated in the part program or synchronized action using the `TRANSMIT` statement.

Syntax

```
TRANSMIT
TRANSMIT(<n>)
```

Meaning

`TRANSMIT`:	Activate TRANSMIT with the first TRANSMIT data set
`TRANSMIT(n)`:	Activate TRANSMIT with the nth TRANSMIT data set

Note

A TRANSMIT transformation active in the channel is activated with:

- Deactivate transformation: `TRAFOOF`
- Activation of another transformation: E.g. `TRACYL`, `TRAANG`, `TRAORI`

2.1.4 Constraints

Look Ahead

All functions requiring Look Ahead (traversal through pole, Look Ahead) work satisfactorily only if the relevant axis motions can be calculated exactly in advance. With TRANSMIT, this applies to the rotary axis and the linear axis perpendicular to it. If one of these axes is the positioning axis, then the Look Ahead function is deactivated by alarm 10912 and the conventional online velocity check activated instead.

Selection of method

The **user is responsible** for making the optimum choice of "Traversal through pole" or "Rotation around pole".

Several pole traversals

A block can traverse the pole any number of times (e.g. programming of a helix with several turns). The part program block is subdivided into a corresponding number of sub-blocks. Analogously, blocks which rotate several times around the pole are likewise divided into sub-blocks.

Rotary axis as modulo axis

The rotary axis can be a modulo rotary axis. However, this is not a mandatory requirement as was the case in SW 2 and 3. The relevant restrictions applying in SW 2 and 3 have been eliminated.

Rotary axis as spindle

If the rotary axis without transformation is used as a spindle, it must be switched to position-controlled mode with SPOS before the transformation is selected.

TRANSMIT with supplementary linear axis

With active TRANSMIT, the channel name of posBCS[ax[3]] must have another name in the part program, like the geometry axes. If posBCS[ax[3]] is written only outside the TRANSMIT transformation, this restriction does not apply if the axis has been assigned to a geometry axis. With active TRANSMIT, no contour information is processed via ax[3].

REPOS

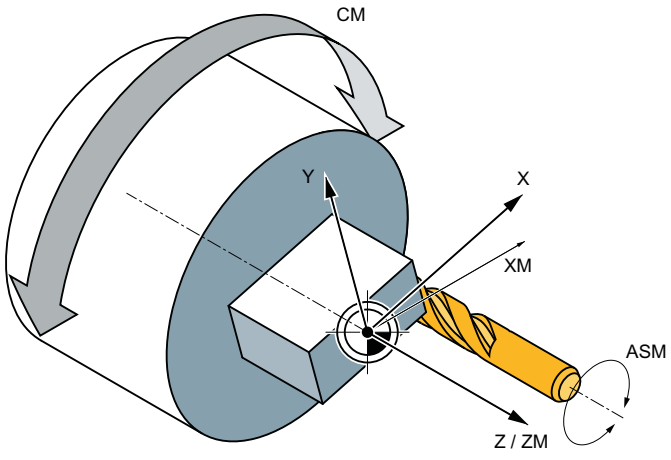
It is possible to reposition on the sub-blocks produced as a result of the extended TRANSMIT function in SW 4. In this case, the control uses the first sub-block that is closest to the repositioning point in the BCS.

Block search

In the case of block search with calculation, the block end point (of the last sub-block) is approached in cases where intermediate blocks have been generated as the result of the extended functionality in SW 4.

2.1.5 Example

The example refers to the axis configuration shown in the following figure.



X, Y, Z	Geometry axes
CM	1st machine axis: Rotary axis
XM	2nd machine axis: Linear axis, perpendicular to rotary axis
ZM	3rd machine axis: Linear axis, parallel to rotary axis
ASM	4th machine axis: Main spindle

Parameter assignment

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM" (1st machine axis: spindle/ rotary axis)
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "XM" (2nd machine axis: linear axis)
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "ZM" (3rd machine axis: linear axis)
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ASM" (4th machine axis: spindle)

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "XC" (linear axis)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "ZC" (linear axis)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "CC" (spindle/rotary axis)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "ASC" (spindle)

Assignment of geometry axes to channel axes

TRANSMIT **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geo. axis → 1st channel axis XC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2 (3rd geo. axis → 2nd channel axis ZC)

TRANSMIT active:

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1 (1st TrafoGeoAxis → 1st channel axis XC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 3 (2nd TrafoGeoAcxis → 3rd channel axis CC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 2 (3rd TrafoGeoAxis → 2nd channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 2 (1st channel axis → 2nd machine axis XM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 3 (2nd channel axis → 3rd machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 1 (3rd channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 4 (4th channel axis → 4th machine axis ASM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (1st machine axis: spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (2nd machine axis: axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (3rd machine axis: axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 2 (4th machine axis: spindle)

Transformation type

- MD24100 $MC_TRAFO_TYPE_1 = 256 (transformation TRANSMIT with a rotary or linear axis)

Offset relative to the zero position of the rotary axis

- MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 = 0

Sign of rotary axis

- MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 = 0

Basic offset of the tool zero relative to the geometry axes while TRANSMIT is active

- MD24920 $MC_TRANSMIT_BASE_TOOL_1 [0] = 0.0 (offset relative to 1st TrafoGeoAxis)
- MD24920 $MC_TRANSMIT_BASE_TOOL_1 [1] = 0.0 (offset relative to 2nd TrafoGeoAxis)
- MD24920 $MC_TRANSMIT_BASE_TOOL_1 [2] = 0.0 (offset relative to 3rd TrafoGeoAxis)

TRANSMIT input axes

- MD24110 $MC_TRAFO_AXES_IN_1[0] = 1 (1st channel axis XC, perpendicular to the rotary axis)
- MD24110 $MC_TRAFO_AXES_IN_1[1] = 3 (3rd channel axis CC, rotary axis)
- MD24110 $MC_TRAFO_AXES_IN_1[2] = 2 (2nd channel axis ZC, parallel to the rotary axis)

Modulo conversion for rotary axis

- MD30300 $MA_ROT_IS_MODULO[0] = 0 (1st machine axis: spindle/rotary axis)
- MD30300 $MA_ROT_IS_MODULO[1] = 0 (2nd machine axis: linear axis)
- MD30300 $MA_ROT_IS_MODULO[2] = 0 (3rd machine axis: linear axis)
- MD30300 $MA_ROT_IS_MODULO[3] = 1 (4th machine axis: spindle)

Programming example

Program code	Comment
N10 T1 D1 G54 G17 G90 F5000 G94	; tool selection
	; 1st Settable work offset
	; XY plane
	; absolute dimensions
	; linear feedrate F in mm/min
N20 G0 X20 Z10 SPOS=45	; approach the start position
	; spindle positioning 45
N30 TRANSMIT	; TRANSMIT with first
	; data record from MD24100 ON
N40 ROT RPL=-45	; frame: Rotation in XY plane by Z
N50 ATRANS X-2 Y10	; frame: Additive offset
N60 G1 X10 Y-10 G41 OFFN=1	; square roughing; 1 mm tolerance (OFFN)
N70 X-10	

Program code	Comment
N80 Y10	
N90 X10	
N100 Y-10	
N110 G0 Z20 G40 OFFN=0	; tool radius compensation OFF,
	; 0 mm tolerance
N120 T2 D1 X15 Y-15	; tool change
N130 Z10 G41	; tool radius compensation left of contour
N140 G1 X10 Y-10	; square part finishing
N150 X-10	
N160 Y10	
N170 X10	
N180 Y-10	
N190 Z20 G40	
N200 TRANS	; deselect frame
N210 TRAFOOF	; TRANSMIT OFF
N220 G0 X20 Z10 SPOS=45	; approach the start position
N230 M30	

2.2 TRACYL cylinder surface transformation (option)

2.2.1 Function

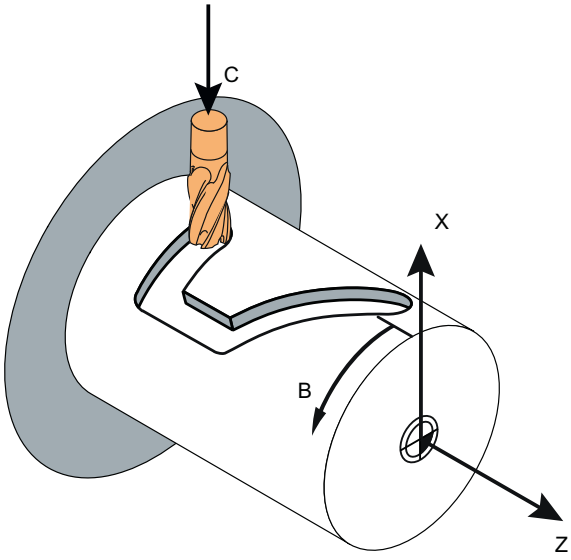
Note

The licensed "TRANSMIT and peripheral surface transformation" option is required for the function "Cylinder surface transformation (TRACYL)."

The TRACYL transformation permits the machining of cylinder jacket curves (grooves) on turning machines.

The path of the grooves is programmed with reference to the unwrapped, level surface of the cylinder.

The controller transforms the programmed traversing movements of the cylinder coordinate system to the traversing movements of the real machine axes.



The machine kinematics must correspond to the cylinder coordinate system:

- One, two or three linear axes and one rotary axis
- The linear axes must be oriented perpendicular to each other
- The rotary axis must be oriented parallel to one of the linear axes

Note

For active transformation, the names of the involved machine, channel and geometry axes are different:

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (geometry axis name)

Transformation Types

The TRACYL transformation exists in three variants:

- without groove side offset (transformation type 512):
- with groove side offset (transformation type 513):
- programmable with or without groove side offset (transformation type 514)

TRACYL without groove wall offset

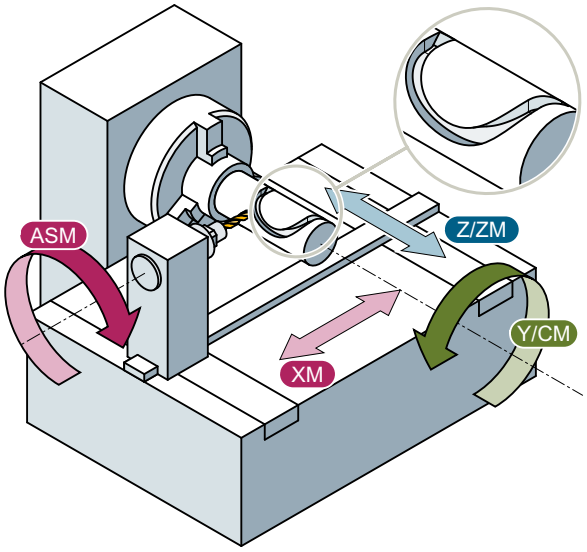
The cylinder surface transformation without groove side offset is used in machine kinematics with one or two linear axes (**axis configuration 1**).

One linear axis

For a machine kinematic with only one linear axis (X), only grooves parallel to the periphery of the cylinder can be generated (transverse grooves).

Two linear axes

For a machine kinematic with two linear axes (X and Z), grooves of any form can be generated on the cylinder.



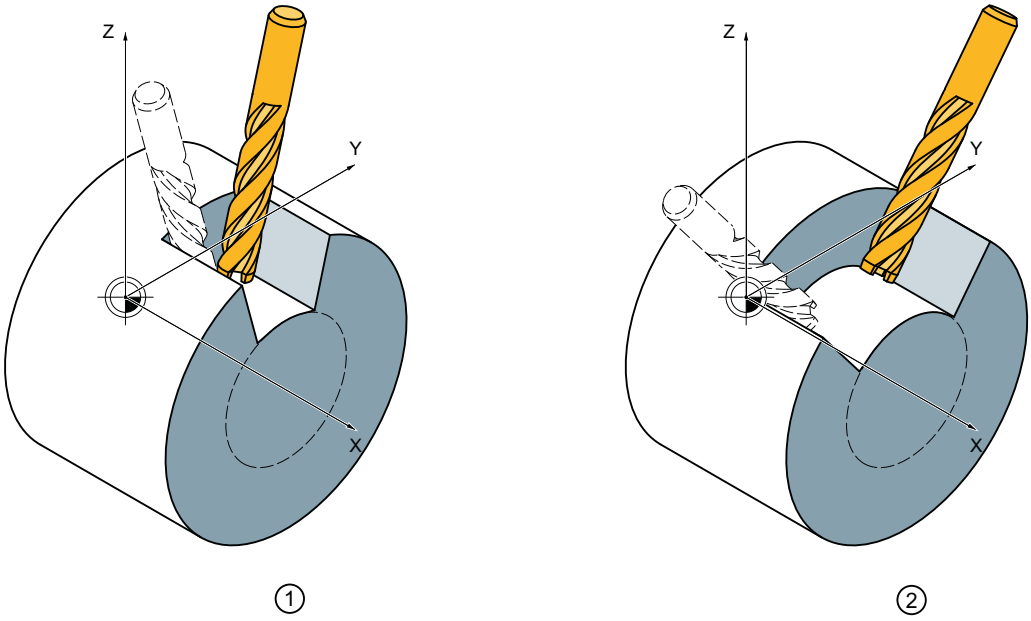
XM	Infeed axis perpendicular to the turning center
ZM	Linear axis parallel to the turning center
Y / CM	Transformatory Y axis / rotary axis
ASM	Main spindle

Figure 2-7 Machine kinematics with two linear axes

Groove edges

For a cylinder surface transformation without groove wall correction, the edges of the groove longitudinal to the rotary axis (longitudinal grooves) are only parallel if the groove width corresponds to the tool diameter. For groove widths greater than the tool diameter, the groove edges are at an angle to each other (see ①).

The groove edges of grooves that extend parallel to the circumference (transverse grooves) are not parallel to each other (see ②). Only the start and the end of the groove edges are not parallel.



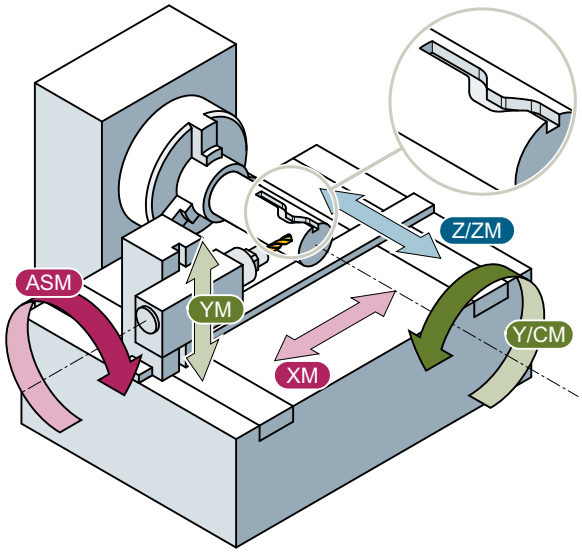
① ②

① Longitudinal groove
② Transverse groove

Figure 2-8 Groove edges with TRACYL without groove side offset

TRACYL with groove wall offset

The cylinder surface transformation with groove wall offset is used for machine kinematics with three linear axes (X, Y, and Z) (**axis configuration 2**).



XM	Infeed axis perpendicular to the turning center
YM	Supplementary axis perpendicular to the X-Z plane
ZM	Linear axis parallel to the turning center
Y / CM	Transformatory Y axis / rotary axis
ASM	Main spindle

Figure 2-9 Machine kinematics with three linear axes

For a machine kinematic with three linear axes (X, Y and Z), grooves of any form can be generated on the cylinder.

Groove edges

Because the Y axis is oriented perpendicular to the turning center, almost parallel groove edges can be generated even for groove widths larger than the tool diameter.

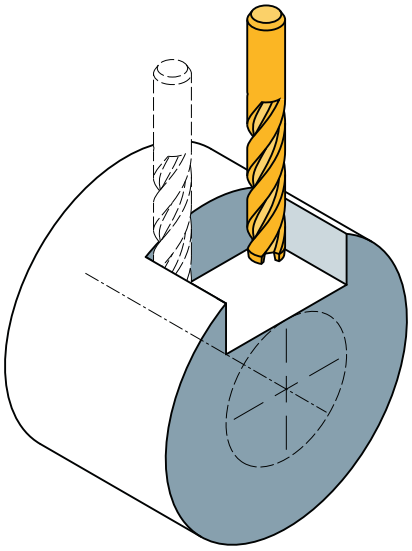


Figure 2-10 Parallel limited longitudinal groove with TRACYL with groove side offset

2.2.2 Parameterization

2.2.2.1 Overview

Machine data: Transformation data in general

The following machine data is used to define transformation data sets in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<n> (definition of the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_AXES_IN_<n> (axis assignment for the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_GEOAX_ASSIGN_TAB_<n> (assignment of geometry axes to channel axes for transformation <n>)
- MD2xxxx $MC_TRAFO_INCLUDES_TOOL_<n> (tool handling with active <n>th transformation)

where <n> = 1, 2, 3, ... max. number of transformation data sets

For TRACYL (type 512, 513, or 514), no more than two transformation data sets may be parameterized in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<x> = <TRACYL type>
- MD2xxxx $MC_TRAFO_TYPE_<y> = <TRACYL type>

Machine data: TRACYL transformation

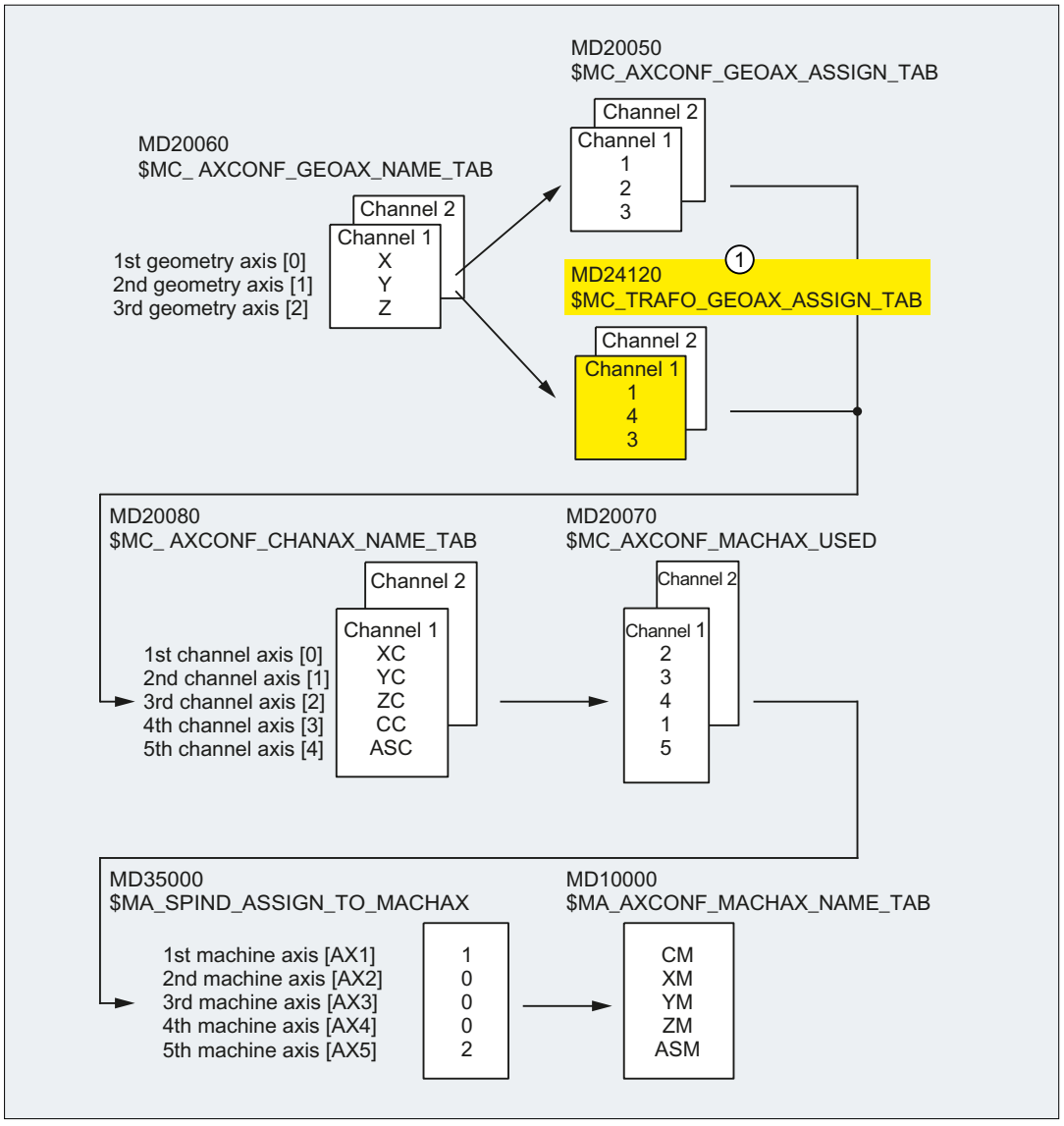
A TRACYL transformation is parameterized in the following machine data:

- MD2xxxx $MC_TRACYL_ROT_AX_OFFSET_<n> (offset of the rotary axis)
- MD2xxxx $MC_TRACYL_ROT_AX_FRAME_<n> (rotary axis offset)
- MD2xxxx $MC_TRACYL_DEFAULT_MODE_<n> (selection of TRACYL mode)
- MD2xxxx $MC_TRACYL_ROT_SIGN_IS_PLUS_<n> (sign of the rotary axis)
- MD2xxxx $MC_TRACYL_BASE_TOOL_<n> (vector of the base tool)

where <n> = 1, 2 (TRACYL data set number)

2.2.2.2 Axis configuration

The following shows an axis configuration that is typical of TRACYL.



① Effective if TRACYL is active.

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "XM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "YM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ZM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4] = "ASM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "XC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "YC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "CC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4] = "ASC"

Assignment of geometry axes to channel axes

TRACYL **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geometry axis → 1st channel axis XC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2 (2nd geometry axis → 2nd channel axis YC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3 (3rd geometry axis → 3rd channel axis ZC)

TRACYL active:

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1 (1st transformation geometry axis → 1st channel axis XC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 4 (2nd transformation geometry axis → 4th channel axis CC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3 (3rd transformation geometry axis → 3rd channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 2 (1st channel axis → 2nd machine axis XM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 3 (2nd channel axis → 3rd machine axis YM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 4 (3rd channel axis → 4th machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 1 (4th channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[4] = 5 (5th channel axis → 5th machine axis ASM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4] = 2 (spindle)

2.2.2.3 Specific settings

Setting the transformation type

The transformation type is set specifically for the transformation data set in:

$MC_TRAFO_TYPE_<n> = <transformation type>

with <n> = 1, 2, ... max. number of transformations

Axis configuration	<Transformation type>
1	512
2	513 or 514

The TRACYL transformation with programmed groove side offset (type 514) can be activated with or without groove side offset (see "Programming (Page 52)").

Transformation type 514 without groove wall offset

If the machine has another linear axis which is perpendicular to both the rotary axis and the first linear axis, transformation type 514 can be used to apply tool offsets with the real Y axis. In this case, it is assumed that the working area of the second linear axis is small and will not be used to execute the part program.

The previous settings apply for MD10000 $MC_TRAFO_GEOAX_ASSSIGN_TAB_<n>.

Slots with slot side offset

The required inclusion of the tool offset has already been taken into account for the TRACYL transformation with groove side offset.

Axis image

The following paragraph describes how the transformation axis image is specified.

Transformation geometry axes

For the transformation dataset <n>, three (or four) channel axis numbers must be specified for TRACYL:

- MD24110 $MC_TRAFO_AXES_IN_1[0]=channel axis number of the axis radial to the rotary axis.
- MD24110 $MC_TRAFO_AXES_IN_1[1]=channel axis number of the rotary axis.
- MD24110 $MC_TRAFO_AXES_IN_1[2]=channel axis number of the axis parallel to the rotary axis.
- MD24110 $MC_TRAFO_AXES_IN_1[3]=channel axis number of the supplementary axis parallel to the cylinder surface and perpendicular to the rotary axis (provided axis configuration 2 is present).

The axis numbers must refer to the channel axis sequences defined by the following machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_<n>

Slots without slot side offset

For transformation type 514 the following indices apply to MD24110 $MC_TRAFO_AXES_IN_<n>[ ].

Meaning of indices in relation to base coordinate system (BCS):

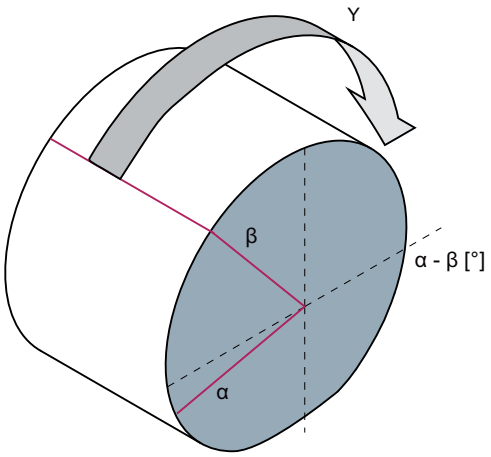
- [0]: Cartesian axis radial to rotary axis (if configured)
- [1]: Axis in generated cylinder surface perpendicular to rotary axis
- [2]: Cartesian axis parallel to rotary axis
- [3]: Linear axis parallel to index 2 in initial position of machine

Meaning of indices in relation to machine coordinate system (MCS):

- [0]: Linear axis radial to rotary axis (if configured)
- [1]: Rotary axis
- [2]: Linear axis, parallel to rotary axis
- [3]: Linear axis, perpendicular to the axes of indices [0] and [1]

Rotational position

The rotational position of the axis on the cylinder peripheral surface perpendicular to the rotary axis must be defined as follows:



α Rotational position of the rotary axis for C=0

β Position of Y = 0

Figure 2-11 Center of axis rotation in the peripheral cylinder surface

MD24800 TRACYL_ROT_AX_OFFSET_<t>

The rotational position of the peripheral surface in relation to the defined zero position of the rotary axis is specified with:

MD24800 $MC_TRACYL_ROT_AX_OFFSET_<t> = ...°

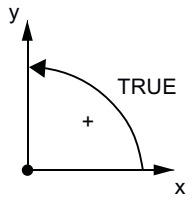
During this, <t> is replaced by the number of TRACYL transformations (<t> can be a maximum of 2) agreed upon in the transformation datasets.

Direction of rotation

The direction of rotation of the rotary axis is specified by machine data as described in the following paragraph.

TRACYL_ROT_SIGN_IS_PLUS_<t>

If the direction of rotation of the rotary axis on the x-y plane is counterclockwise when viewed against the z axis, then the machine data must be set to TRUE, otherwise to FALSE.



MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_<t>=TRUE

In this case, "t" is substituted by the number of TRACYL transformations declared in the transformation data blocks (t may not be greater than 2).

Switchable geometry axes

The PLC is informed when a geometry axis has been replaced using GEOAX( ) through the optional output of an M code that can be set in machine data.

- MD22534 $MC_TRAFO_CHANGE_M_CODE
  Number of the M code that is output at the VDI interface in the case of transformation changeover.

  Note

  If this machine data is set to one of the values 0 to 6, 17, 30, then no M code is output.

Further information

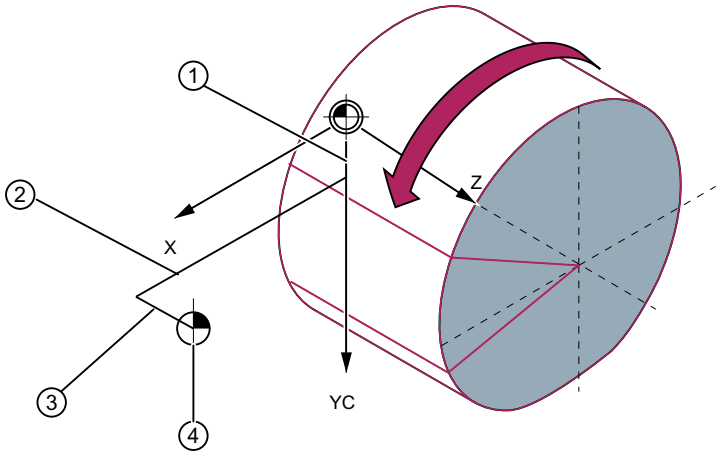
Function Manual Basic Functions; Coordinate systems, axis types, axis configurations, workpiece-related actual-value system, external zero offset

Position of tool zero

The position of the tool zero point in relation to the origin of the Cartesian coordinate system is specified by machine data as described in the following paragraph.

MD24820 TRACYL_BASE_TOOL_<t>

This machine data is used to inform the control of the tool zero point position in relation to the origin of the cylinder coordinate system declared for TRACYL. The machine data has three components for the axes X, Y, Z of the machine coordinate system.



① ty
② tx
③ tz
④ Tool zero

Figure 2-12 Position of tool zero in relation to machine zero

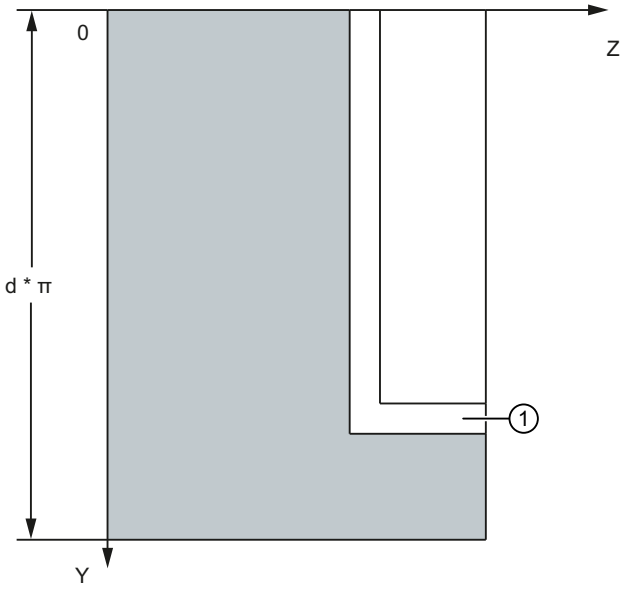
Example

MD24820 $MC_TRACYL_BASE_TOOL_<t>[0] = tx

MD24820 $MC_TRACYL_BASE_TOOL_<t>[1] = ty

MD24820 $MC_TRACYL_BASE_TOOL_<t>[2] = tz

with <t> = the number of TRACYL transformations agreed upon in the transformation datasets



① Slot (example)

Figure 2-13 Cylinder coordinate system

2.2.3 Programming

2.2.3.1 Activate cylinder surface transformation (TRACYL)

The cylinder surface transformation (TRACYL) is activated in the part program or synchronized action using the `TRACYL` statement.

Syntax

```
TRACYL(<d>)
TRACYL(<d>,<n>)
TRACYL(<d>,<n>,<k>)
```

Meaning

`TRACYL(<d>)`:	Activate TRACYL with the first TRACYL data set and working diameter <d>	
`TRACYL (<d>,<n>)`:	Activate TRACYL with the <n>th TRACYL data set and working diameter <d>	
`<d>`:	Reference or working diameter The value must be greater than 1.	
`<n>`:	TRACYL data set number (optional)	
	Range of values:	1, 2

<k>:	The parameter <k> is only relevant for transformation type 514	
	k = 0:	without groove side correction
	k = 1:	with groove side correction
	If the parameter is not specified, then the parameterized basic position applies: $MC_TRACYL_DEFAULT_MODE_<n> With <n> = TRACYL data set number	

Note

A TRACYL transformation active in the channel is switched-off with:

- Deactivate transformation: `TRAFOOF`
- Activation of another transformation: E.g. `TRAANG`, `TRANSMIT`, `TRAORI`

Example

Program code	Comment
...	
N40 TRACYL(40.)	; Activate TRACYL with the first TRACYL data set and working diameter 40 mm.
...	

2.2.3.2 Activate cylinder surface transformation (TRACYL): Further information

Further information

Program structure

A part program for milling a groove with TRACYL transformation 513 (TRACYL with groove side offset) generally comprises the following steps:

1. Select tool.
2. Select TRACYL.
3. Select suitable coordinate offset (frame).
4. Positioning.
5. Program OFFN.
6. Select TRC.
7. Approach block (position TRC and approach groove side).
8. Groove center line contour.
9. Deselect TRC.
10. Retraction block (retract TRC and move away from groove side).
11. Positioning.

12.TRAFOOF.

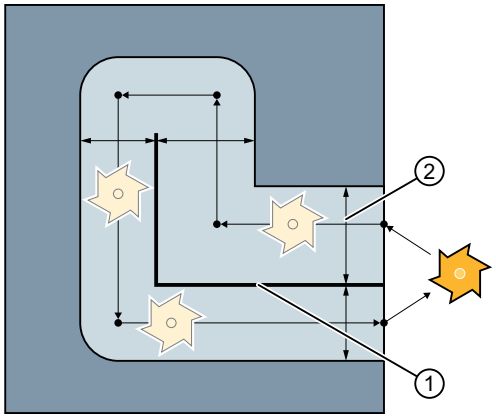
13.Reselect original coordinate shift (frame).

Contour offset (OFFN)

In order to mill grooves using TRACYL transformation 513, the center line of the groove and **half of the groove width** via the OFFN address are programmed in the part program.

To avoid damage to the groove side, OFFN acts only when the tool radius compensation is active.

It is possible to change OFFN within a part program. This allows the groove center line to be offset from the center:



① OFFN
② Programmed path

Note

OFFN should be at least as large as the tool radius to avoid damage occurring to the opposite side of the groove wall.

Note

OFFN acts differently with TRACYL than it does without TRACYL. Since, even without TRACYL, OFFN is included when TRC is active, OFFN should be reset to zero after TRAFOOF.

NOTICE
Effect of OFFN depends on the transformation type For TRACYL transformation 513 (TRACYL with groove side offset), half the groove width is programmed for OFFN. For TRACYL transformation 512 (TRACYL with groove side offset), the value of OFFN acts as an allowance for the TRC.

Tool radius compensation (TRC)

For TRACYL transformation 513, the TRC is not taken into account relative to the groove side, but to the programmed center of the groove. In order that the tool travels to the left of the groove

side, statement G42 must be programmed instead of G41 or the value of OFFN specified with a negative sign.

Tool diameter

With TRACYL and a tool whose diameter is less than the groove width, the same groove side geometry is not generated as with a tool whose diameter is the same as the groove width. To improve the precision, it is recommended that the tool diameter is selected to be only slightly less than the groove width.

Axis utilization

Note

The following axes cannot be used as a positioning axis or a reciprocating axis:

- The geometry axis in the peripheral direction of the cylinder peripheral surface (Y axis).
- The additional linear axis for groove side compensation (Z axis).

2.2.4 Boundary conditions

Selection/deselection

- An intermediate motion block is not inserted (phases/radii).
- A series of spline blocks must be concluded.
- Tool radius compensation must be deselected.
- The frame which was active prior to `TRACYL` is deselected by the control system (analog `G500`).
- An active working area limit is deselected for axes affected by the transformation (analog `WALIMOF`).
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- The Y axis used for the compensation should be set to zero for selection and active groove side compensation.

Tool change

Tools can be changed only when the tool radius compensation function is deselected.

Frame

A frame change with `G91` (incremental dimension) is not specially treated for active transformation. The path to be traversed is evaluated in the workpiece coordinate system of the new frame - regardless of which frame was active in the previous block.

A rotary axis offset can, for example, be entered by compensating the inclined position of a workpiece can be considered using a frame or as offset of the rotary axis.

The following setting is required for the axial complete frame of the rotary axis to act in the transformation:

$MC_TRACYL_ROT_AX_FRAME_<t> = 1

Note

Changes in the axis assignments are converted every time the transformation is selected or deselected.

Further information

Function Manual Basic Functions; Coordinate systems, frames

Function restriction for transformation axes

The axes involved on the cylinder surface transformation must not be used for the following functions:

- Positioning axis
- Oscillating axis
- Preset axis
- Fixed point approach `G75`
- Reference point approach `G74`

Manual traversal in JOG

When generated cylinder surface transformation with groove side compensation ($MC_TRAFO_TYPE = 513) is active in JOG mode, it must be noted that the axes are traversed depending on the preceding status in `AUTOMATIC`. For active slide side compensation, the axes so move differently than for deselected compensation. The part program can therefore be continued (`REPOS`) after a part program interruption.

Interrupt part program

- Mode change from AUTOMATIC to JOG
  If a part program processing is interrupted for active transformation and traversed manually in the JOG operating mode, ensure for continuation of the part program in the AUTOMATIC operating mode that the transformation is already active in the restart block from the current position to the interruption location.

  ⚠ WARNING

  Risk of collision

  No monitoring for collisions takes place.

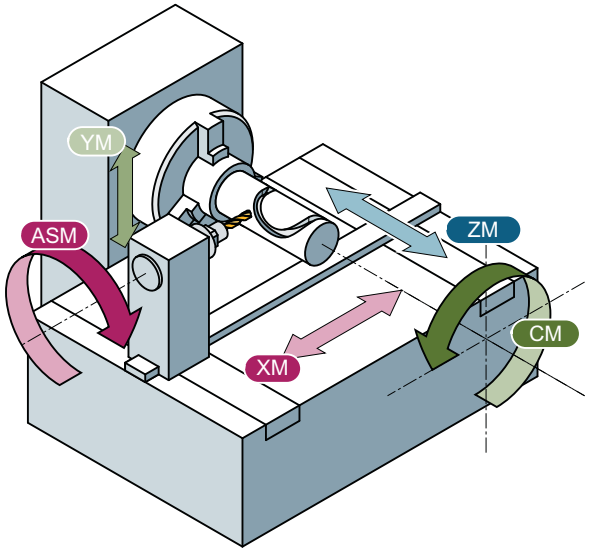
  The operator is responsible for ensuring that the tool can be re-positioned without any difficulties.

- START after RESET
  If a part program processing is interrupted with `RESET` and restarted with `START`, ensure that at part program begin all axes travel with a linear block (G0 or G1) to a defined position, and the remaining part program is traversed reproducably. A tool which was active on `RESET` may no longer be taken into account by the control (settable via machine data).

2.2.5 Examples

2.2.5.1 Machining grooves on a cylinder surface with X-Y-Z-C kinematics

The example refers to the turning machine with an additional Y axis drawn in the following figure.



XM	Infeed axis, perpendicular to rotary axis
YM	Additional axis
ZM	Axis is parallel to rotary axis
CM	Rotary axis
ASM	Main spindle

Parameter assignment

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "XM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "YM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ZM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4] = "ASM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "XC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "YC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "CC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4] = "ASC"

Assignment of geometry axes to channel axes

TRACYL **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geometry axis → 1st channel axis XC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2 (2nd geometry axis → 2nd channel axis YC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3 (3rd geometry axis → 3rd channel axis ZC)

TRACYL active:

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1 (1st transformation geometry axis → 1st channel axis XC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 4 (2nd transformation geometry axis → 4th channel axis CC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 3 (3rd transformation geometry axis → 3rd channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 2 (1st channel axis → 2nd machine axis XM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 3 (2nd channel axis → 3rd machine axis YM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 4 (3rd channel axis → 4th machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 1 (4th channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[4] = 5 (5th channel axis → 5th machine axis ASM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4] = 2 (spindle)

Transformation type

- MD24100 $MC_TRAFO_TYPE_1 = 513 (TRACYL with groove side offset)

Offset relative to the zero position of the rotary axis

- MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 = 0

Sign of rotary axis

- MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 = FALSE

Basic offset of the tool zero relative to the geometry axes while TRACYL is active

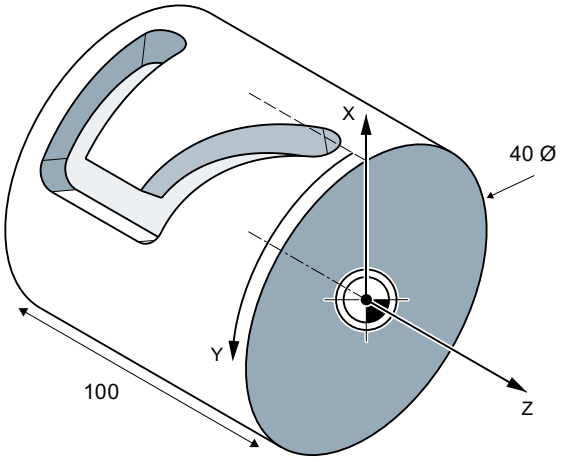
- MD24820 $MC_TRACYL_BASE_TOOL_1 [0] = 0.0 (offset relative to 1st transformation geometry axis)
- MD24820 $MC_TRACYL_BASE_TOOL_1 [1] = 0.0 (offset relative to 2nd transformation geometry axis)
- MD24820 $MC_TRACYL_BASE_TOOL_1 [2] = 0.0 (offset relative to 3rd transformation geometry axis)

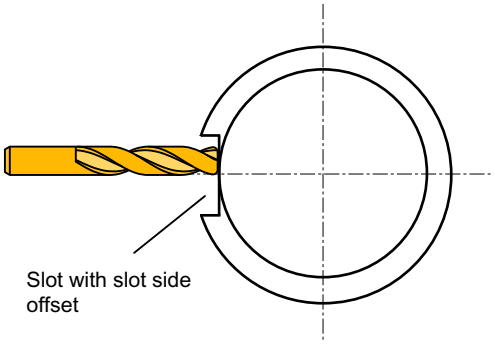
TRACYL input axes

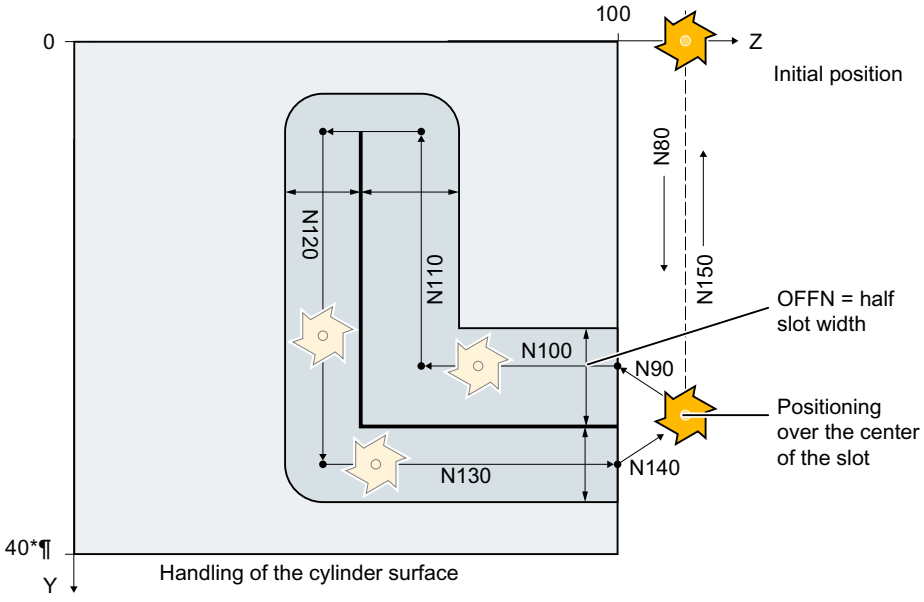
- MD24110 $MC_TRAFO_AXES_IN_1[0] = 1 (1st channel axis XC, perpendicular to the rotary axis)
- MD24110 $MC_TRAFO_AXES_IN_1[1] = 4 (4th channel axis CC, rotary axis)
- MD24110 $MC_TRAFO_AXES_IN_1[2] = 3 (3rd channel axis ZC, parallel to the rotary axis)
- MD24110 $MC_TRAFO_AXES_IN_1[3] = 2 (2nd channel axis YC, special axis)

Programming

Producing a hook-shaped groove with groove side offset (TRACYL transformation type 513)







Tool definition

Program code	Comment
; Tool parameters	
$TC_DP1[1,1]=120	; Tool type: Milling tool
$TC_DP2[1,1] = 0	; Cutting edge position: For turning tools only

Program code	Comment
; Geometry: Length compensation	
$TC_DP3[1,1]=8.	; Length compensation vector: Calculation acc. to type

Program code	Comment
$TC_DP4[1,1]=9.	; Length compensation vector: Calculation according to plane
$TC_DP5[1,1]=7.	

Program code	Comment
; Geometry: Radius	
$TC_DP6[1,1]=6.	; Radius
$TC_DP7[1,1]=0	; Groove width b for slotting saw, rounding radius for milling tools
$TC_DP8[1,1]=0	; Projection k: For slotting saw only
$TC_DP9[1,1]=0	
$TC_DP10[1,1]=0	
$TC_DP11[1,1]=0	; Angle for tapered milling tools

Program code	Comment
; Wear: Length and radius compensation	
$TC_DP12[1,1] =0	; Remaining parameters to $TC_DP24=0 (basis dimension/adapter)

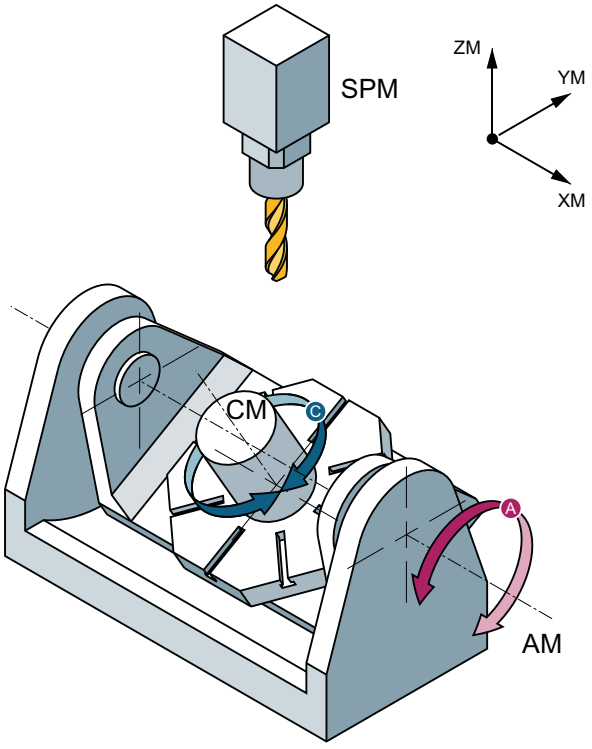
Groove machining on the cylinder surface

Program code	Comment
N10 T1 D1 G54 G90 F5000 G94	; Tool selection, clamping compensation
N20 SPOS=0	; Switchover of the spindle to rotary axis operation, positioning at 0 degrees
N30 G0 X25 Z110	; Approach the start position
N40 TRACYL(40.)	; Activating TRACYL with the first TRACYL data set and working diameter 40 mm
N50 G19	; Machining plane Y/Z (cylinder surface)
N60 Y70	; Positioning over the groove center
N70 G1 X15	; Infeed tool to groove base
N80 OFFN=10	; Define 10 mm groove side spacing relative to groove center line
N90 Z100 G42	; TRC selection and approach to right groove wall
	; G42: TRC right of the progr. contour (≙ groove center)
N100 Z50	; Path I: Producing the groove section parallel with the cylinder axis
N110 Y10	; Path I: Producing the groove section parallel with the circumference
N120 Y70	; Path II: Producing the groove section parallel with the circumference
N130 Z100	; Path II: Producing the groove section parallel with the cylinder axis

Program code	Comment
N140 Z110 G40	; Travel away from the groove wall and TRC dese-lection
N150 G0 X25 Y0	; Return to the initial position
N170 TRAFOOF	; Deactivate transformation
N180 M30	; End of program

2.2.5.2 Machining grooves on a cylinder surface with X-Y-Z-A-C kinematics

The example refers to the 5-axis milling machine with an A- and a C-axis in the following figure.



XM	1. axis of the machining plane
YM	2. axis of the machining plane
ZM	Infeed axis
AM	Rotary axis
CM	Rotary axis
SPM	Main spindle

Parameter assignment

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "XM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "YM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "ZM"

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "SPM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4] = "AM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[5] = "CM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "XC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "YC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "SPC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4] = "AC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[5] = "CC"

Assignment of geometry axes to channel axes

TRACYL **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1 (1st geometry axis → 1st channel axis XC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2 (2nd geometry axis → 2nd channel axis YC)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3 (3rd geometry axis → 3rd channel axis ZC)

TRACYL active:

- MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] = 6 (1st transformation geometry axis → 6th channel axis CC)
- MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] = 2 (2nd transformation geometry axis → 2nd channel axis YC)
- MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] = 3 (3rd transformation geometry axis → 3rd channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 1 (1st channel axis → 1st machine axis XM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 2 (2nd channel axis → 2nd machine axis YM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 3 (3rd channel axis → 3rd machine axis ZM)

- MD20070 $MC_AXCONF_MACHAX_USED[3] = 4 (4th channel axis → 4th machine axis SPM)
- MD20070 $MC_AXCONF_MACHAX_USED[4] = 5 (5th channel axis → 5th machine axis AM)
- MD20070 $MC_AXCONF_MACHAX_USED[5] = 6 (6th channel axis → 6th machine axis CM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4] = 0 (axis)

Transformation type

- MD24200 $MC_TRAFO_TYPE_2 = 513 (TRACYL with groove side offset)

Offset relative to the zero position of the rotary axis

- MD24850 $MC_TRACYL_ROT_AX_OFFSET_2 = 0

Sign of rotary axis

- MD24860 $MC_TRACYL_ROT_SIGN_IS_PLUS_2 = FALSE

Basic offset of the tool zero relative to the geometry axes while TRACYL is active

- MD24870 $MC_TRACYL_BASE_TOOL_2 [0] = 0.0 (offset relative to 1st transformation geometry axis)
- MD24870 $MC_TRACYL_BASE_TOOL_2 [1] = 0.0 (offset relative to 2nd transformation geometry axis)
- MD24870 $MC_TRACYL_BASE_TOOL_2 [2] = 0.0 (offset relative to 3rd transformation geometry axis)

TRACYL input axes

- MD24210 $MC_TRAFO_AXES_IN_2[0] = 3 (3rd channel axis ZC, perpendicular to the rotary axis)
- MD24210 $MC_TRAFO_AXES_IN_2[1] = 6 (6th channel axis CC, rotary axis)
- MD24210 $MC_TRAFO_AXES_IN_2[2] = 2 (2nd channel axis YC, parallel with the rotary axis)
- MD24210 $MC_TRAFO_AXES_IN_2[3] = 1 (1st channel axis XC, special axis)

Programming

Program code	Comment
N10 WORKPIECE(,"",,"CYLINDER",0,0,-180,-80,179)	; Blank definition
N20 M3 S2000	; Setting spindle speed
N30 T="NUTFRAESER" M6 D1	; Tool selection
N40 G0 G54 X0 Y-20 Z105	; Positioning
N50 CYCLE800(0,"TABLE",100000,57,0,0,0,-90,0,0,0,0,0,-1,100,1)	; Rotate A-axis with swivel cycle
N60 G17 G90	; Setting the machining plane
N70 G0 Y-10 Z100 G40	; Positioning
N80 TRACYL(179, 2)	; Selecting the Tracyl data set 2 with groove wall offset
N90 OFFN=20	; Setting the offset (half groove width)
N100 G1 F500 X0 Z75 G42	; Setting the starting point and selecting the TRC
N110 Y30	; Groove center path
N120 X-60	; Groove center path
N130 X0	; Groove center path
N140 Y-10	; Groove center path
N150 Z105 G40	; Retraction and deselection of the TRC
N160 TRAFOOF	; Deselection of the transformation
N170 G0 X0 Y-20 Z115	; Positioning retract movement
N180 M5	; Spindle stop
N190 CYCLE800(0,"TABLE",100000,57,0,0,0,0,0,0,0,0,-1,100,1)	; Turn back A axis
N200 M30	; End of program

2.3 Oblique angle transformation (inclined axis) TRAANG (option)

2.3.1 Function

Note

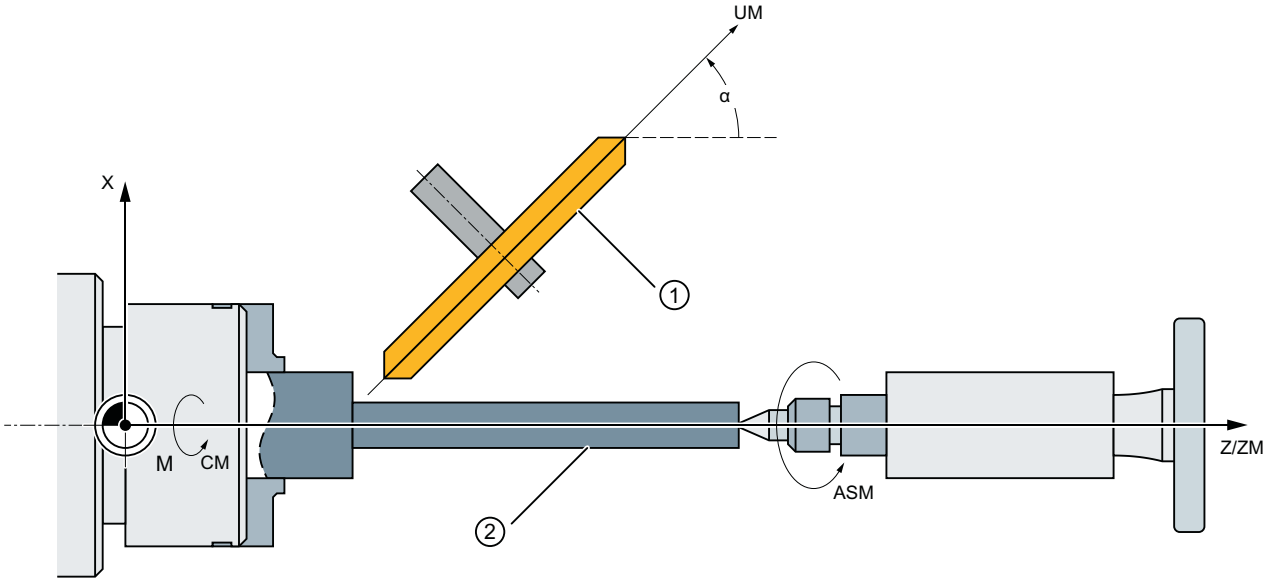
Option

For function "Oblique angle transformation (TRAANG) with **programmable** angle", option "**6FC5800-0AM28-0YB0**" is necessary which requires a license.

For function "Oblique angle transformation (TRAANG) with **fixed** angle", option "**6FC5800-0AS54-0YB0**" is necessary which requires a license.

The oblique angle transformation permits programming in the Cartesian workpiece coordinate system (WCS) on machines with obliquely arranged machine axes, which is a typical axis arrangement on grinding machines.

The controller transforms the programmed traversing movements of the Cartesian coordinate system to the traversing movements of the real machine axes.



①	Grinding disk
②	Workpiece
X	Geometry axis
Z	Geometry axis
ZM	Machine axis
UM	Machine axis
α	Angle of inclined axis
CM	Machine axis (spindle)
ASM	Machine axis (tailstock)

Figure 2-14 Grinding with inclined infeed axis

Versions

- Oblique angle transformation (TRAANG) with **programmable** angle
  For oblique angle transformation (TRAANG) with programmable angle, when activating the transformation, the angle can be specified.
  See Section "Activating an oblique angle transformation with programmable angle (TRAANG) (Page 72)"
- Oblique angle transformation (TRAANG) with **fixed** angle
  For oblique angle transformation (TRAANG) with fixed angle, when activating the transformation, an angle cannot be specified.
  See Section "Activate oblique angle transformation with fixed angle (TRAANG) (Page 73)"

Note

For active transformation, the names of the machine, channel and geometry axes involved must be different:

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (geometry axis name)

2.3.2 Parameterization

2.3.2.1 Overview

Machine data: Transformation data in general

The following machine data is used to define transformation data sets in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<n> (definition of the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_AXES_IN_<n> (axis assignment for the <n>th transformation in the channel)
- MD2xxxx $MC_TRAFO_GEOAX_ASSIGN_TAB_<n> (assignment of geometry axes to channel axes for transformation <n>)
- MD2xxxx $MC_TRAFO_INCLUDES_TOOL_<n> (tool handling with active <n>th transformation)

where <n> = 1, 2, 3, ... max. number of transformation data sets

For TRAANG (type 1024), no more than two transformation data sets may be parameterized in a channel:

- MD2xxxx $MC_TRAFO_TYPE_<x> = <TRAANG type>
- MD2xxxx $MC_TRAFO_TYPE_<y> = <TRAANG type>

Machine data: Transformation TRAANG

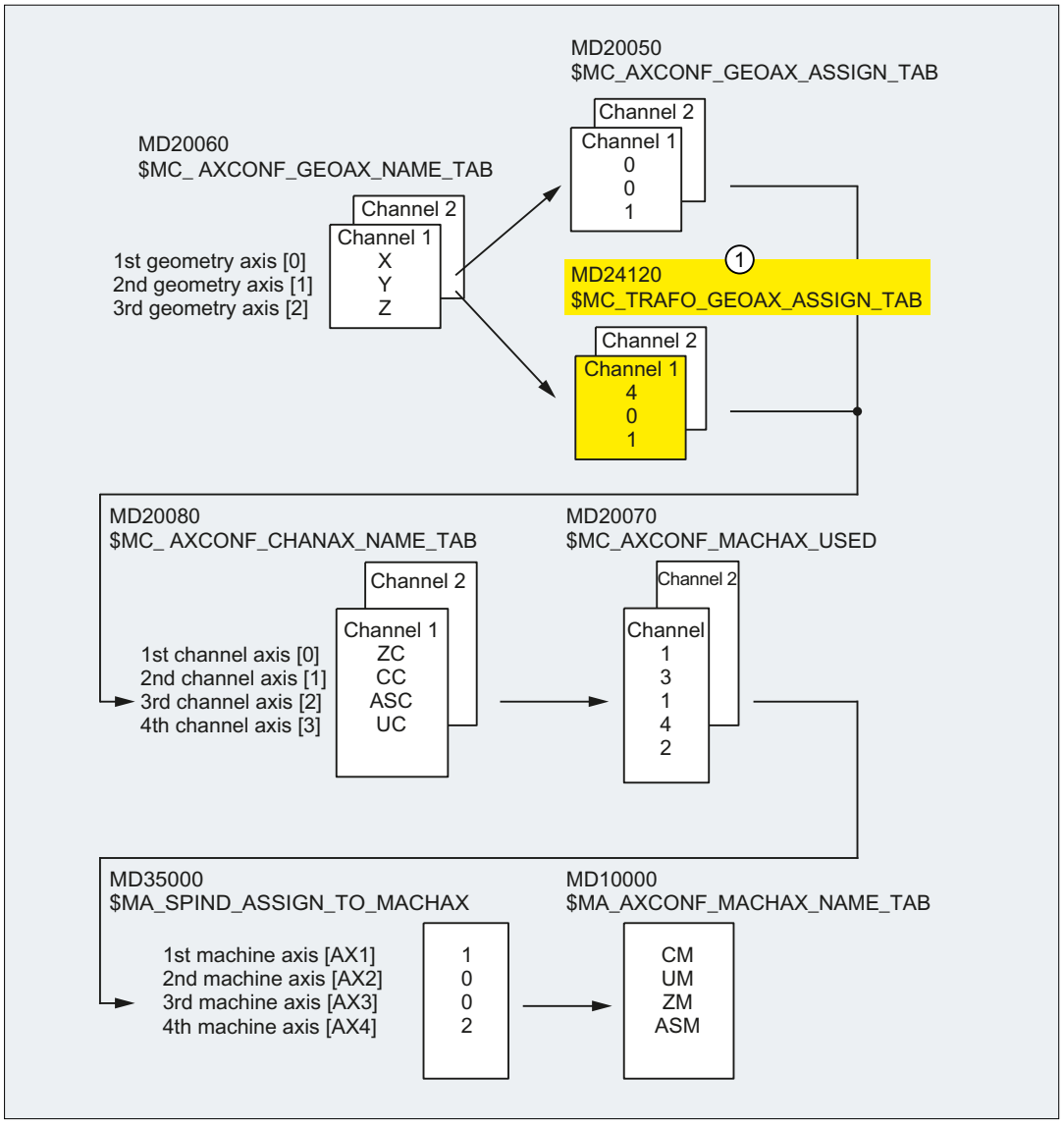
A TRAANG transformation is parameterized using the following machine data:

- MD2xxxx $MC_TRAANG_ANGLE_<n> (angle between longitudinal axis and inclined axis)
- MD2xxxx $MC_TRAANG_BASE_TOOL_<n> (basic offset of the tool zero)
- MD2xxxx $MC_TRAANG_PARALLEL_VELO_RES_<n> (velocity margin for the compensation movements of the longitudinal axis)
- MD2xxxx $MC_TRAANG_PARALLEL_ACCEL_RES_<n> (acceleration margin for the compensation movements of the longitudinal axis)

where <n> = 1, 2 (TRAANG data set number)

2.3.2.2 Axis configuration

The following shows an axis configuration that is typical of TRAANG.



① Effective if TRAANG is active.

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "UM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "ZM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ASM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "CC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "ASC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "UC"

Assignment of geometry axes to channel axes

TRAANG **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 1 (3rd geometry axis → 1st channel axis ZC)

TRAANG active:

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 4 (1st transformation geometry axis → 4th channel axis UC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 0 (-)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 1 (3rd transformation geometry axis → 1st channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 3 (1st channel axis → 3rd machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 1 (2nd channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 4 (3rd channel axis → 4th machine axis ASM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 2 (4th channel axis → 2nd machine axis UM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 2 (spindle)

2.3.2.3 Specific settings

Angle between longitudinal axis and inclined axis

- MD24700 $MC_TRAANG_ANGLE_<n> = <angle>

with -90° < angle < 90° , without 0°

The angle is counted positively in the clockwise direction starting at X (see Section "Function (Page 66)": Angle α).

Base offset of the tool zero

The base offset of the tool zero is specified for the valid geometry axes for the active transformation. The base offset is included with and without selection of the tool length compensation. Programmed tool length compensations are added to the base tool.

- $MC_TRAANG_BASE_TOOL_<n>[k] = <base offset>

with k = 0, 1, 2 (1st - 3rd geometry axis)

Note

The tool zero is not converted when the angle is changed.

Optimization of velocity control

The machine data used to optimize the velocity control in jog mode and in positioning and oscillation modes:

Speed margin

The machine data sets the speed margin for the compensation movements of the longitudinal axis:

MD24720 $MC_TRAANG_PARALLEL_VELO_RES_<n> = <value>

<value>	Meaning
0.0	The speed margin is determined by the NC depending on the angle of the inclined axis and the speed capability of the inclined and the longitudinal axis so that the same speed limitation results in the direction of the longitudinal axis and of the associated perpendicular (virtual) axis.
> 0.0	Speed margin = <value> * (MD32000 $MA_MAX_AX_VELO of the longitudinal axis)

Acceleration margin

The machine data sets the acceleration margin for the compensation movements of the longitudinal axis:

$MC_TRAANG_PARALLEL_ACCEL_RES_<n> = <value>

<value>	Meaning
0.0	The acceleration margin is determined by the NC depending on the angle of the inclined axis and the acceleration capability of the inclined and the longitudinal axis so that the same acceleration limitation results in the direction of the longitudinal axis and of the associated perpendicular (virtual) axis.
> 0.0	Acceleration margin = <value> * (MD32300 $MA_MAX_AX_ACCEL of the longitudinal axis)

M code for transformation change

The specified M code is output for switching the transformation:

- MD22534 $MC_TRAFO_CHANGE_M_CODE = <value>

⚠ WARNING

No M code is output.

The values 0 to 6, 17 and 30 are not output to the PLC.

2.3.3 Programming

2.3.3.1 Activating an oblique angle transformation with programmable angle (TRAANG)

The oblique angle transformation with programmable angle is activated in the part program or synchronized action using the `TRAANG` statement.

Syntax

```
TRAANG
TRAANG()
TRAANG(, <n>)
TRAANG(<α>)
TRAANG(<α>,<n>)
```

Meaning

`TRAANG`: `TRAANG()`:	Activate TRAANG with the first TRAANG data set and last valid angle <α>
`TRAANG(, <n>)`:	Activate TRAANG with the <n>th TRAANG data set and last valid angle <α>
`TRAANG(<α>)`:	Activate TRAANG with the first TRAANG data set and angle <α>
`TRAANG(<α>,<n>)`:	Activate TRAANG with the <n>th TRAANG data set and angle <α>

<α>:	Angle of the inclined axis (optional)	
	Range of values:	-90° < α < + 90°
	The initial state parameterized in the machine data is effective if an angle is not specified: MD2xxxx $MC_TRAANG_ANGLE_<n>	
<n>:	TRAANG data set number (optional)	
	Range of values:	1, 2

Note

Oblique angle transformation TRAANG active in the channel is deactivated using:

- Deactivate transformation: TRAFOOF
- Activation of another transformation: E.g. TRACYL, TRANSMIT, TRAORI

Example

Program code	Comment
N20 TRAANG(45)	; Activate TRAANG with the first TRAANG data set and angle 45°

2.3.3.2 Activate oblique angle transformation with fixed angle (TRAANG)

The oblique angle transformation (TRAANG) with a fixed angle (TRAANG) is activated in the part program or synchronized action using the TRAANG statement. When activated, it is not permissible that an angle is specified. The angle parameterized in the machine data of the particular TRAANG data set is active.

Syntax

```
TRAANG
TRAANG()
TRAANG(, <n>)
```

Meaning

TRAANG: TRAANG():	Activate TRAANG with the first TRAANG data set and angle <α> from machine data MD24700 $MC_TRAANG_ANGLE_1	
TRAANG(, <n>):	Activate TRAANG with <n>th TRAANG data set and angle <α> from machine data MD2xxxx $MC_TRAANG_ANGLE_<n>	
<n>:	TRAANG data set number (optional)	
	Range of values:	1, 2

Note

Oblique angle transformation TRAANG active in the channel is deactivated using:

- Deactivate transformation: TRAFOOF
- Activation of another transformation: E.g. TRACYL, TRANSMIT, TRAORI

Example

Program code	Comment
N20 TRAANG(,2)	; Activate TRAANG with second TRAANG data set
	; and angle from MD24750 $MC_TRAANG_ANGLE_2.

2.3.3.3 Oblique plunge-cutting on grinding machines (G5, G7)

The G commands G7 and G5 are used to simplify programming of oblique plunge-cutting on grinding machines with "inclined axis (TRAANG)", so that when plunge cutting, only the inclined axis is traversed.

Only the required end position of the plunge-cutting motion has to be programmed in X and Z. For G7, starting from the actual position of the X axis, the NC calculates and approaches the programmed end position and angle α of the inclined axis.

The starting position is calculated from the point where the two straight lines intersect:

- Straight line parallel to the Z axis, at a distance from the actual position of the X axis
- Straight line parallel to the inclined axis through the programmed end position

With the subsequent G5, the inclined axis is traversed to the programmed end position.

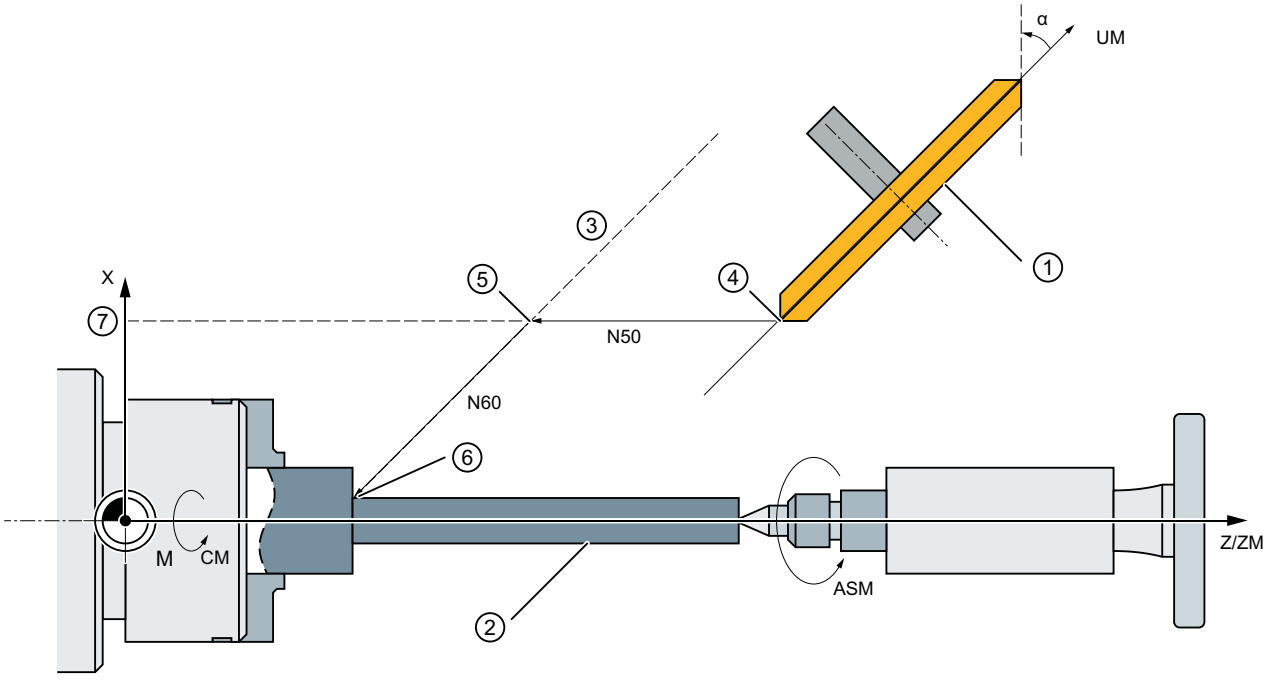
Syntax

```
G7 <Endpos_X> <Endpos_Z>
G5 <Endpos_X>
```

Meaning

G7:	Calculate the starting position for the oblique plunge-cutting and approach.
G5:	Traverse the inclined axis to the programmed end position
<Endpos_X>:	X axis end position
<Endpos_Z>:	End position of the Z axis

Example



① Grinding wheel
② Workpiece
③ Parallel to the inclined axis through the programmed end position
④ Starting position
⑤ Plunge-cutting: Starting position
⑥ Plunge-cutting: End position
⑦ Parallel to the Z axis, at a distance from the actual position of the X axis
X Geometry axis
Z Geometry axis
ZM Machine axis
UM Machine axis

Figure 2-15 Programming an inclined axis

Program code	Comment
N... G18	; Select XZ plane.
N40 TRAANG (45.0)	; Activate TRAANG transformation, angle = 45°
N50 G7 X40 Z70 F4000	; Calculate the starting position and approach
N60 G5 X40 F100	; Traverse inclined axis to the end position.
N70 ...	

2.3.4 Boundary conditions

The transformation can be selected and deselected via part program or MDA.

Selection and deselection

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- The current frame is deselected by the control system. Corresponds to programmed G500.
- An active working area limitation is deselected by the control for the axes affected by the transformation (corresponds to programmed `WALIMOF`).
- An activated tool length compensation is included in the transformation by the control.
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- All axes specified in machine data MD24110 $MC_TRAFO_AXES_IN_n must be synchronized on a block-related basis (e.g. no traversing instruction with POSA...).

Tool change

Tools may only be changed when the tool radius compensation function is deselected.

Frames

All instructions which refer to the workpiece coordinate system are permissible (FRAME, tool radius compensation). Unlike the procedure for inactive transformation, however, a frame change with G91 (incremental dimension) is not specially treated. The increment to be traversed is evaluated in the workpiece coordinate system of the new frame - regardless of which frame was effective in the previous block.

Extensions

When the oblique angle transformation (`TRAANG`) is selected and deselected, the assignment between geometry axes and channel axes can change. The user can apply these geometric contour sections to the axial frame as a translation, rotation, scaling and mirroring in relation to the x and z planes with respect to the inclined infeed axis.

Further information

Further detailed information on the frame corrections for transformations can be found in:

Function Manual Basic Functions; Axes, coordinate systems, frames

Unusable functions

The following functions cannot be used in channel axes involved in the transformation:

- Set actual value (`PRESETON`)
- Travel to fixed stop (`FXS`)
- Reference point approach (`G74` or manual reference point approach)

Velocity control

The velocity monitoring function for oblique angle transformation (TRAANG) is implemented by default during preprocessing. The velocity monitoring function and limitation in the main run takes place in the following operating states:

- AUTOMATIC mode: Programming of a positioning or oscillating axis that is involved in the transformation
- JOG mode: After a change of operating mode to JOG mode

After resynchronization of the preprocessing to the main run, e.g. after a change of operating mode from JOG to AUTOMATIC mode, velocity monitoring and limitation takes place in the preprocessing again.

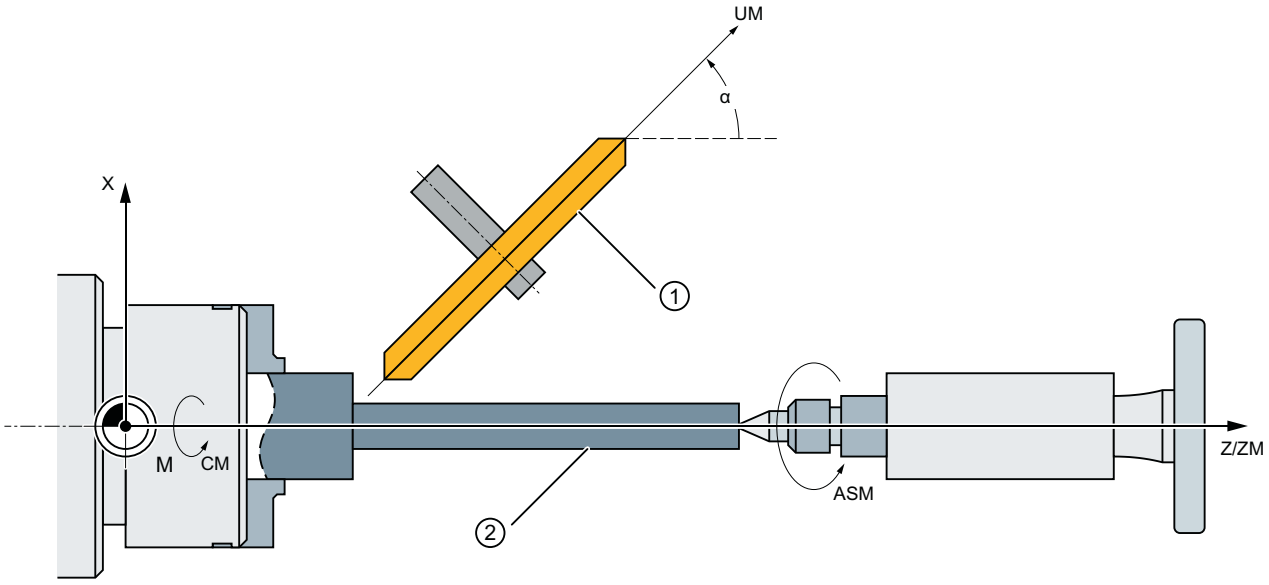
Note

Program operation: Behavior with axis-specific feedrate stop

- Velocity monitoring in main run
  Program execution is stopped with active velocity monitoring in the main run for one of the axes involved in the transformation of the axis-specific feedrate stop at (DBX31, ... DBX4.3 == 1).
- Velocity monitoring in the preprocessing
  The NC program execution continues with active velocity monitoring in the preprocessing for one of the axes involved in the transformation of the axis-specific feedrate stop at (DBX31, ... DBX4.3 == 1). The precondition for this is that no traversing motions are programmed for the relevant axis.

2.3.5 Example

The example refers to the axis configuration shown in the following figure.



①	Grinding disk
②	Workpiece
X	Geometry axis
Z	Geometry axis
ZM	Machine axis
UM	Machine axis
α	Angle of inclined axis
CM	Machine axis (spindle)
ASM	Machine axis (tailstock)

Figure 2-16 Inclined axis

Parameterization

Machine axis name

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0] = "CM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1] = "UM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2] = "ZM"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3] = "ASM"

Geometry axis names

- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0] = "X" (name of the 1st geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = "Y" (name of the 2nd geometry axis)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2] = "Z" (name of the 3rd geometry axis)

Channel axis names

- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "ZC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "CC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "ASC"
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "UC"

Assignment of geometry axes to channel axes

TRAANG **not** active:

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0 (-)
- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 1 (3rd GeoAxis → 1st channel axis ZC)

TRAANG active:

- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 4 (1st TrafoGeoAxis → 4th channel axis UC)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1] = 0 (-)
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2] = 1 (3. TrafoGeoAxis → 1st channel axis ZC)

Assignment of channel axes to machine axes

- MD20070 $MC_AXCONF_MACHAX_USED[0] = 3 (1st channel axis → 3rd machine axis ZM)
- MD20070 $MC_AXCONF_MACHAX_USED[1] = 1 (2nd channel axis → 1st machine axis CM)
- MD20070 $MC_AXCONF_MACHAX_USED[2] = 4 (3rd channel axis → 4th machine axis ASM)
- MD20070 $MC_AXCONF_MACHAX_USED[3] = 2 (4th channel axis → 2nd Machine axis UM)

Identification of spindles

- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0] = 1 (spindle)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2] = 0 (axis)
- MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3] = 2 (spindle)

Transformation type

- MD24100 $MC_TRAFO_TYPE_1 = 1024 (TRAANG)

Angle between Cartesian axis and real (oblique) axis

- MD24700 $MC_TRAANG_ANGLE_1 = 45.

Basic offset of the tool zero relative to the geometry axes while TRAANG is active

- MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0 (offset relative to 1st TrafoGeoAxis)
- MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0 (offset relative to 2nd TrafoGeoAxis)
- MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0 (offset relative to 3rd TrafoGeoAxis)

TRAANG input axes

- MD24110 $MC_TRAFO_AXES_IN_1[0] = 4 (4th channel axis UC, inclined axis)
- MD24110 $MC_TRAFO_AXES_IN_1[1] = 1 (1st channel axis ZC, parallel with the Z axis)
- MD24110 $MC_TRAFO_AXES_IN_1[2] = 0 (-)

Programming example

Program code	Comment
N10 G0 G90 Z0 UM=10 G54 F5000 G18 G64 T1 D1	; XZ planes, tool selection, clamping compensation
N20 TRAANG(45)	; Activate TRAANG with the first TRAANG data set and angle 45°
N30 G0 Z10 X5	; Approach the start position
N40 WAITP(Z)	; Wait for end of travel of the Z axis
N50 OSP1[Z]=10 OSP2[Z]=5 OST1[Z]=-2 OST2[Z]=-2 FA[Z]=5000	; Oscillating motion ; OSP1/OSP2: Left/right reversal point ; OST1/OST2: Stopping point at the left/right reversal point
N60 OS[Z]=1	; Activate oscillation
N70 POS[X]=4.5 FA[X]=50	; Position X axis in parallel
N80 OS[Z]=0	; Deactivate oscillation
N90 WAITP(Z)	; Wait for end of travel of the Z axis
N100 TRAFOOF	; Deactivate transformation
N110 G0 Z10 UM=10	; Retract
N120 M30	

2.4 Chained transformations

2.4.1 Function

2.4.1.1 Introduction

For a concatenated transformation, **two** transformations can be connected one after the other (chained). As a consequence, motion components of the axes can be taken from the first transformation and used as input data for the second transformation. The motion components of the axes from the second transformation then act on the machine axes.

- The following are possible as **first** transformation:
  - TRAORI orientation transformations, including universal milling head
    Further information
    F2: Multi-axis transformations (Page 137)
  - Face end transformation TRANSMIT
  - Cylinder surface transformation TRACYL
- Only the following is possible as **second** transformation:
  - Inclined axis TRAANG

Note

Exception, when commissioning

A concatenated transformation normally comprises **two** transformations chained one after the other. For test purposes, e.g. during commissioning, it is also permissible to enter just one transformation in the chaining list.

Note

Different axis identifiers

For active transformation, the names of the machine, channel and geometry axes involved must be different:

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (geometry axis name)

Application examples

- Grinding contours that are programmed as a side line of a cylinder (TRACYL) using an inclined grinding wheel, e.g. tool grinding.
- Finish cutting of a contour that is not round and was generated with TRANSMIT using an inclined grinding wheel.

Axis configuration

The following configuration measures are necessary for a chained transformation:

- Assignment of names to geometry axes
- Assignment of names to channel axes
- Assignment of geometry axes to channel axes
  - General situation (no transformation active)
- Assignment of channel axes to machine axis numbers
- Identification of spindle, rotation, modulo for axes
- Allocation of machine axis names
- Transformation-specific settings (for individual transformations **and** for **chained** transformations)
  - Transformation type
  - Axes going into the transformation
  - Assignment of geometry axes to channel axes during active transformation
  - Depending on transformation, rotational position of the coordinate system, direction of rotation, tool zero point in relation to the original coordinate system, angle of the inclined axis, etc.

Note

Supplementary conditions and special cases

The supplementary conditions and special cases specified in the individual transformation descriptions must also be observed when used within a chained transformation.

Boundary conditions

Transformation data blocks for each channel

Several transformation data blocks can be defined for each channel. The names of the corresponding machine data of the transformations begin with $MC_TRAFO... and end with..._n, where n = 1, 2, 3, ...

Number of chained transformations

Within the transformations of a channel, a maximum of **two chained** transformations may be defined.

Chaining direction

The first transformation of the chained transformations has the basic coordinate system (BCS) as input.

The second transformation of the chained transformations has the machine coordinate system (MCS) as output.

Tool data

A tool is always assigned the **first** transformation of chained transformations. The subsequent transformation then behaves as if the active tool length were zero.

Only the basic tool lengths set in the machine data (_BASE_TOOL_) are valid for the **first** transformation in the chain.

2.4.1.2 System variables

The following transformation-specific system variables are available:

System variable	Meaning
$AA_ITR	Current setpoint value at output of the nth transformation
$AA_IBC	Current setpoint value of a cartesian axis
$VA_ITR	Current actual value at output of the nth transformation
$VA_IBC	Current cartesian BCS encoder position of an axis
$VA_IW	Current WCS actual value of an axis
$VA_IB	Current BCS encoder position of an axis

Boundary conditions

- Warm restart
  The encoder position has the value 0 for non-referenced axes. For the $VA variables, the encoder actual values are appropriately inverse-transformed.
- Channel reset
  The active transformation can change after a channel reset. This can have a direct influence on the system variable values.
  An active transformation, which is active again after a channel reset, is briefly deactivated and then reactivated. This influences the position variables. The values of variables can change.
  Via the variable:
  $AC_STAT == 0
  this status can be queried in synchronous actions.

$AA_ITR: Actual setpoint value at output of the nth transformation

System variable $AA_ITR[<axis>, <transformation layer>] determines the setpoint position of an axis at the output of the n th cascaded transformation.

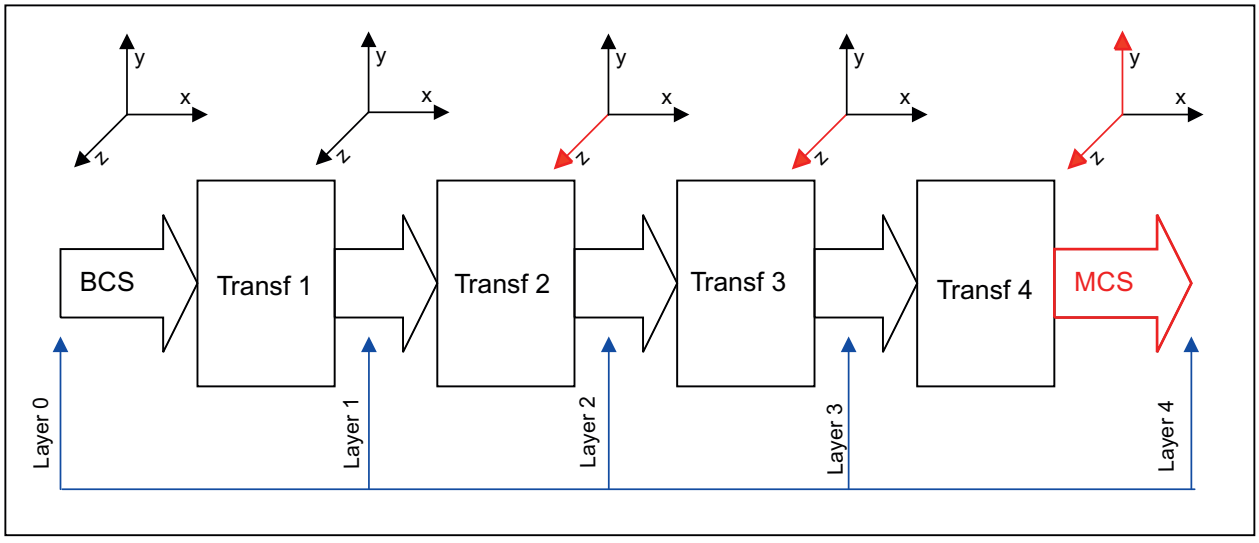


Figure 2-17 Transformer layer

Axis

As 1st index of the system variable, either a geometry, a channel or a machine axis name is permissible. When programming geometry axis names, in each transformer layer, the assignment of the channel axes to the geometry axes corresponding to transform layer 0 is effective. This means that the geometry axis assignment on the BCS side is active in all planes.

Using geometry axis names is meaningful only if the geometry axes are not switched over. Otherwise, it is always better to use channel axis names.

Transformer layer

The 2nd index of the system variable corresponds to the transformation layer from which the positions are taken:

- Transformer layer 0: The positions correspond to BCS positions, $AA_ITR[x,0] == $AA_IB[x]
- Transformer layer 1: Setpoint positions at the output of the 1st transformation
- Transformer layer 2: Setpoint positions at the output of the 2nd transformation
- Transformer layer 3: Setpoint positions at the output of the 3rd transformation
- Transformer layer 4: Setpoint positions at the output of the 4th transformation, $AA_ITR[x,4] == $AA_IM[x]

If one or more transformations of the transformation chain are missing, the highest layers continue to deliver the same values. If, for example, transformation 3 and transformation 4 are missing, then the following applies:

$AA_ITR[x,2] = $AA_ITR[x,3] = $AA_ITR[x,4] = $AA_IM[x]

If the transformations are deactivated using `TRAFOOF` - or for a channel reset - then layers 0 to 4 merge. The system variable then supplies the BCS value (layer 0).

$AA_IBC: Actual setpoint of a cartesian axis

System variable $AA_IBC[<axis>] determines the setpoint position of a cartesian axis lying between BCS and MCS. If an axis is cartesian at the output of the nth transformation, then this output value is delivered. If the corresponding axis at the output of all transformations is not cartesian, then the BCS value including all BCS offsets of the axis are determined.

If TRACON responds to an axis as cartesian, then its MCS value is delivered. The used axis name can be a geometry, channel or a machine axis name.

$VA_ITR: Actual value at the output of the nth transformation

System variable $VA_ITR[<axis>, <transformation layer>] determines the setpoint position of an axis at the output of the n th chained transformation.

$VA_IBC: Actual cartesian BCS encoder position of an axis

System variable $VA_IBC[<axis>] determines the encoder position of a cartesian axis lying between BCS and MCS. The used axis name can be a geometry, channel or a machine axis name.

If an axis at the output of the nth transformation is cartesian, then this output value is delivered. If the corresponding axis at the output of all transformations is not cartesian, then the BCS value of the axis is determined.

$VA_IW: Current WCS actual value of an axis

System variable $VA_IW[<axis>] determines the encoder position of an inverse-transformed axis in WCS. The WCS value contains all axis-related superimposition portions (DRF, AA_OFF, ext. zero offset etc.) and offset values (CEC, etc.).

$VA_IB: Actual BCS encoder position of an axis

System variable $VA_IB[<axis>] determines the inverse-transformed encoder position of an axis in BCS. The BCS value contains all axis-related superimposition portions (DRF, AA_OFF, ext. zero offset etc.) and offset values (CEC, etc.).

Note

VA_ITR, VA_IBC, $VA_IW, $VA_IB

The value of the variable does not change while reading the variable within an IPO cycle, although the actual value could have changed.

In active transformations, one must consider that the transformation of the actual values into BCS in the IPO cycle can be very time-consuming. In this case one must set an adequate IPO cycle.

2.4.2 Programming

A configured concatenated (chained) transformation is activated in the part program or synchronized action using the TRACON statement.

Syntax

```
TRACON(<Trafo_No>,<Par_1>,...,<Par_n>,<Par_n+1>)
...
TRAFOOF
```

Meaning

TRACON:	Activate concatenated transformation If another transformation was previously activated, it is implicitly disabled by means of TRACON().		
<Trafo_No>:	Number of the concatenated transformation:		
	Type:	INT	
	Range of values:	0 ... 2	
	Value:	0, 1	First/only concatenated transformation
		2	Second concatenated transformation
		Not specified	Same meaning as with 0 or 1
		Note: Values not equal to 0, 1, 2 generate an error alarm.	
<Par_1>, ..., <Par_n>, <Par_n+1>:	Parameters for the concatenated transformations		
	<Par_1>, ..., <Par_n>	Parameters of the first transformation in the chain The actual number depends on the transformation type: • TRAORI: 2 parameters • TRACYL: 1 to 2 parameters	
	<Par_n+1>	Parameters of the second transformation in the chain (TRAANG) ≙ angle of the inclined axis	
	If parameters are not set, the defaults or the parameters last used take effect. Commas must be used to ensure that the specified parameters are evaluated in the sequence in which they are expected, if default settings are to be effective for previous parameters. In particular, a comma is required before at least one parameter, even though it is not necessary to specify <Trafo_No> - for example TRACON(, 3.7).		
TRAFOOF:	Deactivate the last activated (concatenated) transformation		

Example

```
Program code                Comment
...
N230 TRACON(1,45.)          ; Activate first concatenated transformation.
                            ; The previously active transformation is automatically de-
                            selected.
                            ; The angle for the inclined axis is 45°.
...
N330 TRACON(2,40.)          ; Activate second concatenated transformation.
                            ; The angle for the inclined axis is 40°.
...
N380 TRAFOOF                ; Deactivate second concatenated transformation.
...
```

2.4.3 Examples

2.4.3.1 Application example of chained transformations

The following transformations are to be defined in the channel:

Data set	Transformation
1	TRAORI 5-axis transformation with rotatable tool and axis sequence AB (trafo type 16)
2	TRANSMIT Face end transformation (trafo type 256)
3	TRAANG Oblique angle transformation (trafo type 1024)
4	TRAORI + TRAANG First chained transformation (trafo type 8192)
5	TRANSMIT + TRAANG Second chained transformation (trafo type 8192)

Parameter assignment

General channel configuration

```
CHANDATA(1)                                ; Channel data in channel 1
MD20070 $MC_AXCONF_MACHAX_USED[0]=1
MD20070 $MC_AXCONF_MACHAX_USED[1]=2
MD20070 $MC_AXCONF_MACHAX_USED[2] = 3
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4
MD20070 $MC_AXCONF_MACHAX_USED[4]=5
MD20070 $MC_AXCONF_MACHAX_USED[5]=6
```

```
MD20070 $MC_AXCONF_MACHAX_USED[6]=7
MD20070 $MC_AXCONF_MACHAX_USED[7] = 0
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="A"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="B"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[5]="C"
MD36902 $MA_IS_ROT_AX[ AX4 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX5 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX6 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX7 ] = TRUE
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX5]= 0
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX7] = 1
MD35000 $MA_ROT_IS_MODULO[AX7] = TRUE
```

Single transformations

```
; 1. TRAORI
MD24470 $MC_TRAFO_TYPE_1= 16              ; TRAORI: A-B kinematics
MD24410 $MC_TRAFO_AXES_IN_1[0]=1
MD24410 $MC_TRAFO_AXES_IN_1[1]=2
MD24410 $MC_TRAFO_AXES_IN_1[2]=3
MD24410 $MC_TRAFO_AXES_IN_1[3]=4
MD24410 $MC_TRAFO_AXES_IN_1[4]=5
MD24410 $MC_TRAFO_AXES_IN_1[5]=0
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3
MD24550$MC_TRAFO5_BASE_TOOL_1[0]=0
MD24550$MC_TRAFO5_BASE_TOOL_1[1]=0
MD24550$MC_TRAFO5_BASE_TOOL_1[2]=0
```

```
; 2. TRANSMIT
MD24200 $MC_TRAFO_TYPE_2 = 256            ; TRANSMIT
MD24210 $MC_TRAFO_AXES_IN_2[0] = 1
MD24210 $MC_TRAFO_AXES_IN_2[1] = 6
MD24210 $MC_TRAFO_AXES_IN_2[2]=3
MD24210 $MC_TRAFO_AXES_IN_2[3]=0
MD24210 $MC_TRAFO_AXES_IN_2[4] = 0
MD24210 $MC_TRAFO_AXES_IN_2[5] = 0
MD24210 $MC_TRAFO_AXES_IN_2[6]=0
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =6
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3
```

```
; 3. TRAANG
MD24300 $MC_TRAFO_TYPE_3 = 1024           ; TRAANG
```

```
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 3
MD24310 $MC_TRAFO_AXES_IN_3[2] = 2
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =2
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0
```

Chained transformations

```
; 4. TRACON (TRAORI + TRAANG chaining)
MD24400 $MC_TRAFO_TYPE_4 = 8192
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0


; 5. TRACON (TRANSMIT + TRAANG chaining)
MD24430 $MC_TRAFO_TYPE_5 = 8192
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =6
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2] = 0
```

Programming example

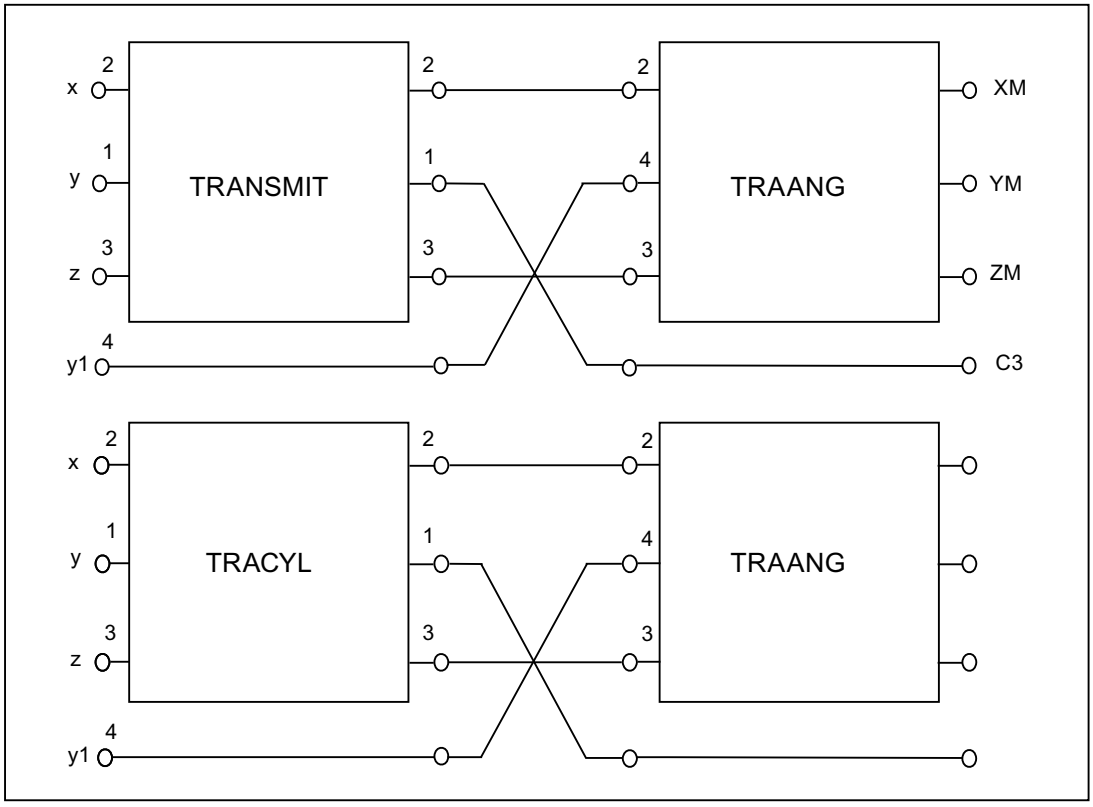
The following programming example assumes that the angle of the "inclined axis" can be set on the machine and is set to 0° when the single transformations are activated.

Program code	Comment
; Tool specification	
$TC_DP1[1,1]=120	;Tool type
$TC_DP3[1,1]=10	; Tool length
N2 X0 Y0 Z0 A0 B0 F20000 T1 D1	
N4 X20	

Program code	Comment
; Call single transformations	
N30 TRANSMIT	; Activate TRANSMIT
N40 X0 Y20	
N50 X-20 Y0	
N60 X0 Y-20	
N70 X20 Y0	
N80 TRAFOOF	; Deactivate TRANSMIT
N130 TRAANG(45.)	; Activate inclined axis transformation
	; Parameter: Angle 45°
N140 X0 Y0 Z20	
N150 X-20 Z0	
N160 X0 Z-20	
N170 X20 Z0	
...	
; Activate chained transformations	
N230 TRACON(1,45.)	; First concatenated transformation
	; Activate (TRAORI + TRAANG).
	; The previously active transformation
	; is automatically deselected.
	; The parameter for the inclined axis is 45°.
N240 X10 Y0 Z0 A3=-1 C3=1 ORIWKS	
N250 X10 Y20 B3=1 C3=1	
...	
N330 TRACON(2,40.)	; Activate second concatenated
	; transformation (TRANSMIT + TRAANG).
	; The parameter for the inclined axis
	; is 40°.
N335 X20 Y0 Z0	
N340 X0 Y20 Z10	
N350 X-20 Y0 Z0	
N360 X0 Y-20 Z0	
N370 X20 Y0 Z0	
N380 TRAFOOF	; Deactivate second concatenated
	; transformation.
...	
N1000 M30	

2.4.3.2 Determining the axis positions in the transformation chain

Two chained transformations are configured in the following example, and the system variables for determining the axis positions in the synchronous action are read cyclically in the part program.



Parameter assignment

```
CHANDATA(1)
; TRANSMIT
MD24100 $MC_TRAFO_TYPE_1=256
MD24110 $MC_TRAFO_AXES_IN_1[0] = 2
MD24110 $MC_TRAFO_AXES_IN_1[1] = 1
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3
; TRACYL
MD24200 $MC_TRAFO_TYPE_2=512
MD24210 $MC_TRAFO_AXES_IN_2[0]=2
MD24210 $MC_TRAFO_AXES_IN_2[1]=1
MD24210 $MC_TRAFO_AXES_IN_2[2]=3
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =2
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3
```

```
; TRAANG
MD24300 $MC_TRAFO_TYPE_3=1024
MD24310 $MC_TRAFO_AXES_IN_3[0] = 2
MD24310 $MC_TRAFO_AXES_IN_3[1]=4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =2
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0
; 1. TRANSMIT/TRAANG chaining
MD24400 $MC_TRAFO_TYPE_4=8192   ; TRACON (1)
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24995 $MC_TRACON_CHAIN_1[3] = 0
MD24410 $MC_TRAFO_AXES_IN_4[0]=1
MD24410 $MC_TRAFO_AXES_IN_4[1]=2
MD24410 $MC_TRAFO_AXES_IN_4[2]=3
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
; 2. TRACYL/TRAANG chaining
MD24430 $MC_TRAFO_TYPE_5=8192   ; TRACON (2)
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2]=0
MD24996 $MC_TRACON_CHAIN_2[3]=0
MD24432 $MC_TRAFO_AXES_IN_5[0]=1
MD24432 $MC_TRAFO_AXES_IN_5[1]=2
MD24432 $MC_TRAFO_AXES_IN_5[2]=3
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =2
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
M17
```

Programming example

Program code	Comment
N10 $TC_DP1[1,1]=120	
N20 $TC_DP3[1,1]=20	
N30 $TC_DP4[1,1]=0	
N40 $TC_DP5[1,1]=0	
N50	
N60 X0 Y0 Z0 F20000 T1 D1	
N70	
N80	; Cyclic reading in the synchronized actions
N90 ID=1 WHENEVER TRUE DO $R0=$AA_ITR[X,0] $R1=$AA_ITR[X,1] $R2=$AA_ITR[X,2]	

```
Program code                Comment
N100 ID=2 WHENEVER TRUE DO $R3=$AA_IBC[X] $R4=$AA_IBC[Y] $R5=$AA_IBC[Z]
N110 ID=3 WHENEVER TRUE DO $R6=$VA_IW[X]-$AA_IW[X]
N120 ID=4 WHENEVER TRUE DO $R7=$VA_IB[X]-$AA_IB[X]
N130 ID=5 WHENEVER TRUE DO $R8=$VA_IBC[X]-$AA_IBC[X]
N140 ID=6 WHENEVER TRUE DO $R9=$VA_ITR[X,1]-$AA_ITR[X,1]
N150
N160
N170 TRACON(1,)           ; 1. TRANSMIT/TRAANG chaining
N180 X20 Y0 Z0
N190 X0 Y20 Z10
N200 X-20 Y0 Z0
N210 X0 Y-20 Z0
N220 X20 Y0 Z0
N230 TRAFOOF
N240
N250
N260 TRACON(2, 40.)       ; 2. TRACYL/TRAANG chaining
N270 X20 Y0 Z0
N280 X0 Y20 Z10
N290 X-20 Y0 Z0
N300 X0 Y-20 Z0
N310 X20 Y0 Z0
N320 TRAFOOF
N330
N340 M30
```

2.5 Persistent transformation

Function

A persistent transformation is always active and has a relative effect to the other explicitly selected transformations.

Selected transformations are computed as the first chained transformation in relation to the persistent transformation.

Transformations such as e.g. TRANSMIT, which are selected relative to the persistent transformation, are chained in the NC program with persistent transformation using TRACON. The chained transformation, e.g. TRANSMIT, is then programmed in the NC program.

Further information

Programming Manual Advanced; Transformations > Activate concatenated transformation (TRACON)

Selection and deselection

Persistent transformation is selected via the following machine data:

- MD20144 $MC_TRAFO_MODE_MASK, bit 0 = 1
- MD20144 $MC_TRAFO_RESET_VALUE defines persistent transformation.
- MD20140 $MC_TRAFO_RESET_VALUE=Number of the transformation data set of the persistent transformation

Other machine data

- MD20110 $MC_RESET_MODE_MASK
  - Bit 0 = 1 (Bit 7 is evaluated)
  - Bit 7 =0 (MD20140 $MC_TRAFO_RESET_VALUE determines the transformation data set)
- MD20112 $MC_START_MODE_MASK (MD20140 $MC_TRAFO_RESET_VALUE)
- MD20118 $MC_GEOAX_CHANGE_RESET= TRUE (i.e. geometry axes are reset).

With `TRAFOOF` the active, chained transformation `TRACON` is deselected and the persistent transformation is automatically selected.

Effects on HMI operation

As a transformation is always active with the persistent transformation, the HMI user interface is adapted accordingly for the selection and deselection of transformations:

Chained transformation `TRACON`

The HMI operator interface does **not** display `TRACON`, but the 1st chain transformation of `TRACON`, e.g. `TRANSMIT`. Accordingly, the transformation type of the 1st chained transformation is returned by the system variable, i.e. $P_TRAFO or $AC_TRAFO. Cycles that `TRANSMIT` uses can then be directly used.

Deactivating the `TRAFOOF` transformation

In accordance with the `TRAFOOF` programming instruction **no** transformation is displayed in the G command list at the user interface. System variable $P_TRAFO as well as $AC_TRAFO then supply a value of 0. The persistent transformation is active and the BCS and MCS coordinate systems differ. The displayed MCS position always refers to the actual machine axes.

System variables

System variable to read transformation types of the active or chained transformations.

- $AC_TRAFO (active transformation)
- $AC_TRAFO_CHAIN[0] (active chained transformation)
- $P_TRAFO (programmed transformation)
- $P_TRAFO_CHAIN[0] (programmed chained transformation)

Frames

Frame adjustments for selection and deselection of the TRACON are carried out as if there was only the first chained transformation. Transformations on the virtual axis cease to be effective when TRAANG is selected.

JOG

The persistent transformation remains in effect when traversing with JOG.

Boundary conditions

The persistent transformation does not change the principle operating sequences in the NC. All restrictions applying to an active transformation also continue to apply:

- For RESET, an existing transformation is completely deselected - and the persistent transformation is selected again.
- When referencing, the transformation is deselected and then requires a channel reset or NC start in order to again select the persistent transformation.

Example

For a lathe with an inclined additional Y axis, the transformation of the inclined axis should be part of the machine configuration. As a consequence, it does not have to be taken into consideration by the programmer. With TRACYL or TRANSMIT, the transformations are selected, which must then include TRAANG.

When deactivating the programmed transformations, TRAANG is automatically reactivated. TRACYL or TRANSMIT are correspondingly displayed on the user interface.

Machine constellation: Lathe with a Y1 axis, inclined to X1 and perpendicular to Z1

Machine data parameterization

Program code

```
CANDATA (1)
; Kinematic without transformations
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3
; Data for TRAANG
MD24100 $MC_TRAFO_TYP_1 = 1024 ; TRAANG, Y1 axis inclined to X1,
perpendicular to Z1
MD24110 $MC_TRAFO_AXES_IN_1[0]=2
MD24110 $MC_TRAFO_AXES_IN_1[1]=1
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
MD24110 $MC_TRAFO_AXES_IN_1[3] = 0
MD24110 $MC_TRAFO_AXES_IN_1[4] = 0
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0] = 1
```

Program code

```
; Definition of persistent transformation
MD20144 $MC_TRAFO_MODE_MASK = 1
MD20140 $MC_TRAFO_RESET_VALVUE= 1
MD20110 $MC_RESET_MODE_MASK = 'H01'
MD20112 $MC_START_MODE_MASK = 'H80'
MD20140 $MC_TRAFO_RESET_VALUE
MD20118 $MC_GEOAX_CHANGE_RESET= TRUE
; Data for TRANSMIT, TRACYL
MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 ; also 2, causes alarm 21617
MD24200 $MC_TRAFO_TYP_2 = 257 ; TRANSMIT
MD24210 $MC_TRAFO_AXES_IN_2[0] = 1
MD24210 $MC_TRAFO_AXES_IN_2[1] = 4
MD24210 $MC_TRAFO_AXES_IN_2[2] = 3
MD24210 $MC_TRAFO_AXES_IN_2[3] = 0
MD24210 $MC_TRAFO_AXES_IN_2[4] = 0
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =4
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3

MD24300 $MC_TRAFO_TYP_3 = 514 ; TRACYL
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
; Data for TRACON
; TRACON chaining TRANSMIT 257/TRAANG (Y1 axis inclined in relation to X1)
MD24400 $MC_TRAFO_TYP_4 = 8192 ; TRACON
MD24995 $MC_TRACON_CHAIN_1[0] = 3
MD24995 $MC_TRACON_CHAIN_1[1] = 1
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =4
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
; TRACON chaining TRANSMIT 257/TRAANG (Y1 axis inclined in relation to X1)
MD24430 $MC_TRAFO_TYP_5 = 8192 ; TRACON
MD24996 $MC-TRACON_CHAIN_2[0] = 2
MD24996 $MC-TRACON_CHAIN_2[1] = 1
MD24996 $MC_TRACON_CHAIN_2[2] = 0
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =4
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
M17
```

NC program

Program code

```
$TC_DP1[1,1]=120 ; tool type
$TC_DP2[1,1] = 0
$TC_DP3[1,1]=3 ; length compensation vector
$TC_DP4[1,1]=25
$TC_DP5[1.1] =5
$TC_DP6[1,1]= 2; Radius; tool radius
; Transformation changeover
N1000 G0 X0 Y=0 Z0 A80 G603 SOFT G64
N1010 X10 Y20 Z30 ; TRAANG(,1) not required, as automatically selected
N1110 TRANSMIT(1)
N1120 X10 Y20 Z30
N1130 Y2=0        ; TRACON(2) not required, as automatically converted
N1210 TRAFOOF     ; TRAANG(,1) not required, as automatically converted
N1220 X10 Y20 Z30
M30
```

2.6 Cartesian PTP travel

2.6.1 Function

Precondition

Cartesian point-to-point or PTP travel can only be activated if one of the following transformations is active:

- TRAORI (orientation transformation)
- TRANSMIT (face side transformation)
- TRACON (chained transformation)
  Precondition: The first chained transformation supports PTP travel!
- RCTRA (transformation for 2 to 4-axis handling/robot kinematics)
  Precondition: Compile cycle "RMCC/RCTRA Transformation Handling" is loaded and active!
- ROBX (transformation for 4 to 6-axis robot kinematics)
  Precondition: Compile cycle "RMCC/ROBX Transformation Extended Robotics" is loaded and active!

Without active transformation, alarm 14146 "CP or PTP motion without transformation not permitted" is output.

Note

The axes of the transformation must not simultaneously be positioning axes.

Function

With the Cartesian PTP travel, in G0 and G1 sets it is possible to approach a point programmed as a Cartesian destination point with a synchronous axis movement.

Regarding the traversing movement, the Cartesian PTP travel acts as though no transformation is active The position data in the part program continue to be in the Cartesian workpiece coordinate system. Programmed frames (coordinate movements and rotations) remain valid. The display is also in the Cartesian workpiece coordinate system.

The movement in the Cartesian PTP travel is described by a starting and end position of the axes. The path of the movement described by the individual axes from the starting to the end position is not specified by the controller by means of path interpolation points and can therefore not be predicted. The axes only have to reach their specified target position. Interaction between the axis movements only exists in regard to time, i.e. all axis start simultaneously and are stopped at the same time when the target position has been reached. The slowest axis determines the total traverse time.

One sensible usage option is to travel by a singularity. If Cartesian positions are available, which are provided by a CAD/CAM system for example, this has the benefit of not having to convert them into machine axis values. Furthermore, direct traversing of the machine axes is time-optimized compared to Cartesian traversing with active transformation and programmed feed. The result of this would either be a corresponding reduction of the feedrate when traversing through the singularity (speed planning in preprocessing) or an overload of the axis if no speed planning is carried out in preprocessing.

Note

Cartesian PTP travel is only permissible in conjunction with the interpolation types G0 and G1.

Versions

Three versions are available for Cartesian PTP travel:

- **PTP**
  The programmed Cartesian position in G0 and G1 blocks is approached with synchronous axis motion.
- **PTPG0**
  Only in G0 blocks is the programmed Cartesian position approached with synchronous axis motion. In G1 blocks, it is switched to the type of motion **CP** (CP = Continuous Path) and the position is approached with a Cartesian path motion.
- **PTPWOC** (only for orientation transformation)
  Just the same as PTP, however, without any compensatory motion, which is caused by motion of rotary axes and orientation axes.
  Since the transformation, which compensates the movement of the tool tip (TCP) during orientation changes, is not active, the position of the TCP in the workpiece coordinate system must generally also be changed if no linear axes involved in the transformation are programmed.
  Programmed linear axis movements are carried out in the workpiece coordinate system that applies at the beginning of the block and overlays the simultaneously programmed rotary axis movements if applicable. The block end point that is reached in the workpiece coordinate system is therefore generally **not** the same as the programmed end point.

The following table lists which version can be practically used for which transformation:

transformation	PTP	PTPG0	PTPWOC	CP
TRAORI	+	+	+	+
TRANSMIT	+	+	-	+
RCTRA	+	+	-	+
ROBX	+	+	-	+

NOTICE

Risk of collision

For PTP travel it must be observed that in part there are significantly different tool movements than with CP!

This must especially be considered for PTPG0, because subprograms can be created with it independently of the active transformation. With active front-end transformation (TRANSMIT), however, entirely different collision risks may occur than with an orientation transformation (TRAORI).

For PTP travel with active front-end transformation (TRANSMIT), machine axes basically travel the shortest path. Minor displacements of the block end point can cause the rotary axis to rotate by +179.99° instead of + -179.99°. This results in a completely different motion, even though the block end point has hardly changed.

PTP/CP switchover in the NC program

The switchover between the Cartesian path movement (CP) and the Cartesian PTP travel in the NC program is done via the commands of the G group 49 (see "Activating/deactivating Cartesian PTP travel (PTP, PTPG0, PTPWOC, CP) (Page 103)").

PTP/CP switchover in JOG mode

You can also switch between the Cartesian path movement (CP) and the Cartesian PTP travel in JOG mode.

Switchover is carried out using the NC/PLC interface signal:

DB21, ... DBX29.4 (activate PTP travel)

Note

DB21, ... DBX29.4 is only effective in JOG mode when the transformation is active.

The active type of motion is reported via the following NC/PLC interface signal:

DB21, ... DBX317.6 (PTP travel active)

2.6.2 Commissioning

2.6.2.1 Response after POWER ON

After Power On, traversing mode CP is automatically effective for axis traversal with transformation.

The default can be switched over to Cartesian PTP travel via the following machine data:

MD20150 $MC_GCODE_RESET_VALUES[48] (initial setting of G group 49)

Value	Meaning
1	CP (default setting)
2	PTP
3	PTPG0
4	PTPWOC

2.6.2.2 Response after reset/part program end

Which command from G group 49 takes effect after a reset/part program end depends on the setting in the machine data:

MD20152 $MC_GCODE_RESET_MODE[48] (reset behavior of G group 49)

Value	Meaning
0	The default command in MD20150 $MC_GCODE_RESET_VALUES[48] becomes effective (default setting).
1	The command that was active before the reset/part program end remains effective.

2.6.2.3 Consideration of the SW limits during PTP travel

Function

With Cartesian PTP travel, a rotary axis is normally traversed according to the strategy "shortest path" if no position information has been programmed via the operation TU (Page 108). To prevent a software limit switch being overtraveled when the target position is behind the software limit switch, the user can use the "Consideration of the SW limits during PTP travel" function. The rotary axis is then traversed with the strategy "long path" instead of "shortest path" if overtravel of the software limit switch would happen, i.e. the opposite traversing direction as with the "shortest path" strategy.

Requirement

The function can only be used when the relevant axis has been referenced.

Activation

The "Consideration of the SW limits during PTP travel" function can be activated separately for each axis.

This setting is made via bit 14 in the axial machine data:

MD30455 $MA_MISC_FUNCTION_MASK

Bit 14	= 0	With Cartesian PTP travel, the strategy "shortest path" is retained for software limit switch overtravel of a rotary axis (basic setting).
	= 1	The strategy "long path" is used when during Cartesian PTP travel a rotary axis with the strategy "shortest path" would overtravel the software limit switch.

Supplementary conditions

Modulo rotary axes

With modulo rotary axes, the strategy "long path" can only be used when the software limit switch monitoring has been selected for the relevant axis:

DB31, ... DBX12.4 (modulo rotary axis: Activate traversing range limits) **= 1**

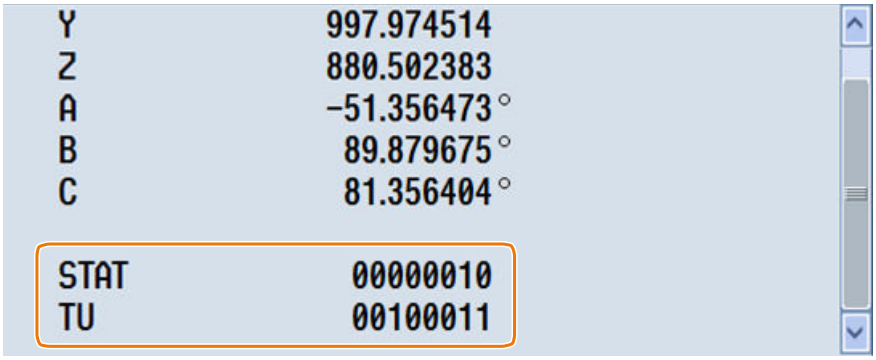
Example

Axis 6 is to traverse from +150° t +240°. The software limit switch is at +200°.

If for axis 6, bit 14 in MD30455 $MA_MISC_FUNCTION_MASK is set to "1", axis 6 is traversed to -120°.

2.6.2.4 Displaying STAT and TU

In the SINUMERIK Operate user interface, in the "Machine" operating area, users can track the actual values of STAT (Page 104) and TU (Page 108):



Precondition: The display of STAT and TU is activated when commissioning SINUMERIK Operate.

Further information

Commissioning Manual SINUMERIK Operate; Adapt operating area of machine > Display STAT and TU

2.6.3 Programming

2.6.3.1 Activating/deactivating Cartesian PTP travel (PTP, PTPG0, PTPWOC, CP)

The Cartesian point-to-point or PTP travel is activated/deactivated in the NC program using G group 49 commands.

The commands are modal. The default setting is travel with Cartesian path motion (CP).

Contrary to CP, for active PTP travel, only the Cartesian target point is transformed, and the machine axes are traversed in synchronism.

In order that the Cartesian target point can be uniquely converted into machine axis values, in addition to position and angular data, information is also necessary that identifies the axis positions. This data is retrieved from the adjustable addresses STAT (Page 104) and TU (Page 108).

Requirement

Transformation TRANSMIT, RCTRA or ROBX is active.

Syntax

```
PTP / PTPG0 / PTPWOC
...
CP
```

Meaning

PTP:	Activating point-to-point motion PTP The programmed Cartesian position in G0 and G1 blocks is approached with synchronous axis motion.
PTPG0:	Activating point-to-point motion PTPG0 Only in G0 blocks is the programmed Cartesian position approached with synchronous axis motion. In G1 blocks, a switchover is made to CP path motion.
PTPWOC:	Activate point-to-point movement PTPWOC (only possible if orientation transformation is active) Just the same as PTP, however, without any compensatory motion, which is caused by motion of rotary axes and orientation axes.
CP:	Deactivating point-to-point motion and activating path motion CP Cartesian path motion is executed with CP.

Note

PTPWOC

It does not make any sense to use PTPWOC in combination with a RCTRA or ROBX transformation!

Examples

See:

- Example 3: PTPG0 and TRANSMIT (Page 112)

2.6.3.2 Activating/deactivating Cartesian PTP travel (PTP, PTPG0, PTPWOC, CP)

The Cartesian point-to-point or PTP travel is activated/deactivated in the NC program using G group 49 commands.

The commands are modal. The default setting is travel with Cartesian path motion (CP).

Contrary to CP, for active PTP travel, only the Cartesian target point is transformed, and the machine axes are traversed in synchronism.

In order that the Cartesian target point can be uniquely converted into machine axis values, in addition to position and angular data, information is also necessary that identifies the axis positions. This data is retrieved from the adjustable addresses STAT (Page 104) andTU (Page 108).

Precondition

Transformation TRAORI, TRANSMIT, RCTRA or ROBX is active.

Syntax

```
PTP / PTPG0 / PTPWOC
...
CP
```

Meaning

PTP:	Activating point-to-point motion PTP The programmed Cartesian position in G0 and G1 blocks is approached with synchronous axis motion.
PTPG0:	Activating point-to-point motion PTPG0 Only in G0 blocks is the programmed Cartesian position approached with synchronous axis motion. In G1 blocks, a switchover is made to CP path motion.
PTPWOC:	Activate point-to-point movement PTPWOC (only possible if orientation transformation is active) Just the same as PTP, however, without any compensatory motion, which is caused by motion of rotary axes and orientation axes.
CP:	Deactivating point-to-point motion and activating path motion CP Cartesian path motion is executed with CP.

Note

PTPWOC

It does not make any sense to use PTPWOC in combination with a RCTRA or ROBX transformation!

Examples

See:

- Example 1: PTP travel of a 6-axis robot with ROBX transformation (Page 111)
- Example 2: PTP travel for generic 5-axis transformation (Page 112)
- Example 3: PTPG0 and TRANSMIT (Page 112)

2.6.3.3 Specify the position of the joints (STAT)

Position data with Cartesian coordinates and specification of the tool orientation are not sufficient to uniquely identify the machine position, as several joint positions are possible for the same tool orientation. Depending on the kinematics involved, there can be as many as 8 different joint positions. These different joint positions are transformation-specific.

In an order to avoid any ambiguity, the joint positions are specified under the STAT address.

Note

The control takes into account programmed STAT values only for PTP motions. They are ignored with CP motions because a change of position is not normally possible while traversing with an active transformation. When traversing with active CP, the position for the target point is taken from the starting point.

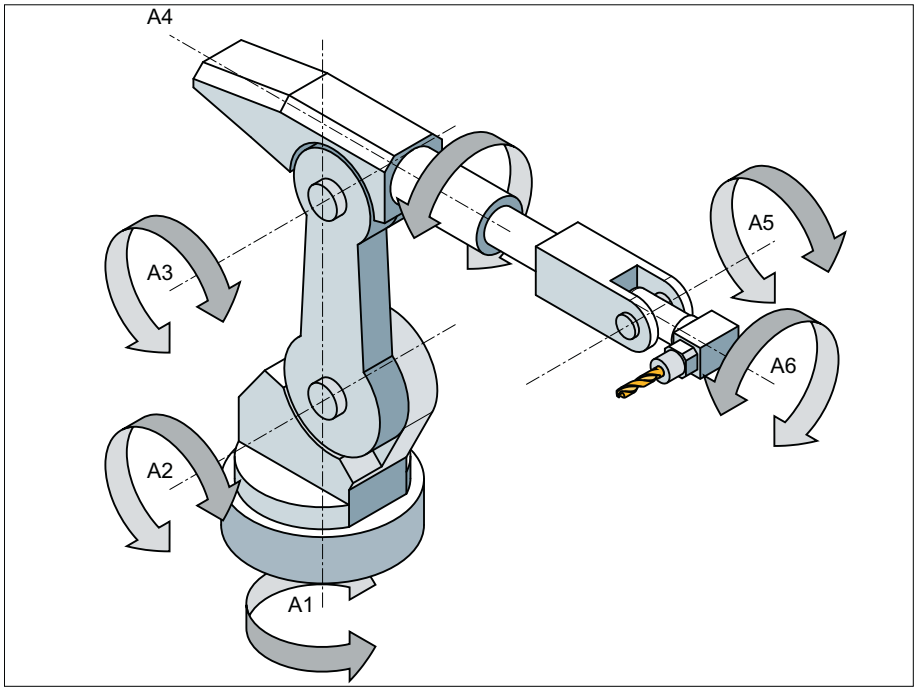
Syntax

```
STAT=<Value>
```

Meaning

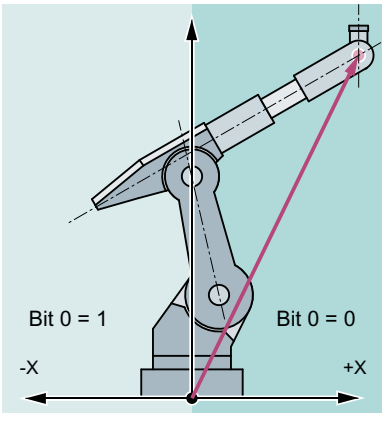
`STAT`:	Adjustable address to specify joint positions
`<Value>`:	Binary or decimal value Contains one bit for each possible position. The significance of the bits is defined by the particular transformation.

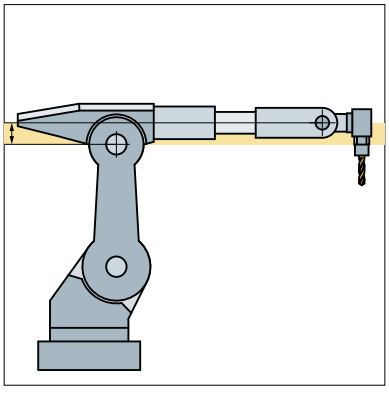
The use of STAT is to be illustrated by the example of a 6-axis articulated robot with milling spindle. The kinematic transformation is to be realized using the ROBX robot transformation (precondition: Compile cycle "RMCC/ROBX Transformation Extended Robotics" is loaded and active).



Axes A1, A2 and A3 are the main axes of the articulated robot. The axes A4, A5 and A6, which are also designated as head or hand/wrist axes, are positioned in the working area with the main axes. The additional motion options of the hand/wrist axes enable the milling spindle to be orientated in space as required for the particular machining task. Various articulated joint positions are possible to achieve the same tool orientation.

The articulated joint positions required for machining are selected by programming bit 0 ... 2 of the adjustable STAT address:

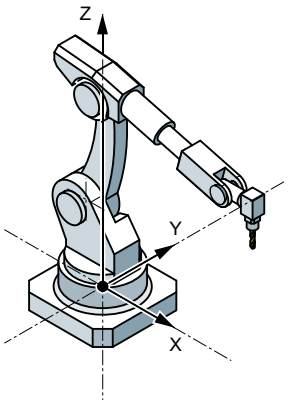
Bit 0	Position of the intersection points of the hand/wrist axes (A4, A5, A6)		
	= 0	Basic range (**shoulder right**) The robot is in the basic range if the X value of the intersection point of the hand/wrist axes is positive in relation to to the A1 coordinate system.	 Example: The intersection point of the hand/wrist axes lies in the basic range
	= 1	Overhead range (**shoulder left**) The robot is in the overhead range if the X value of the intersection point of the hand/wrist axes is negative in relation to the A1 coordinate system.	

Bit 1	Position of axis 3 The angle at which the value of bit 1 changes depends on the particular robot type. The following applies to robots whose axes 3 and 4 intersect:	
	= 0	A3 <0° (**elbow down**)
	= 1	A3 ≥0° (**elbow up**)
	Note: For robots with an offset between axes 3 and 4, the angle at which the value of bit 1 changes depends on the magnitude of this offset.	 Offset between A3 and A4
Bit 2	Position of axis 5	
	= 0	A5 ≥0° (**no handflip**)
	= 1	A5 <0° (**handflip**)

Program example:

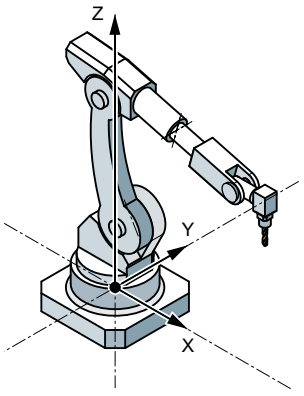
Program code	**Comment**
...	
N14 T="T8MILLD20" D1	; $TC_DP3[1,1]=132.95
N16 ORIMKS	
N17 G1 PTP X1665.67 Y0 Z1377.405 A=0 B=0 C=0 **STAT=...** F2000	; The STAT value defines the articulated joint positions (see below)
...	

STAT=1 ('B001') → Shoulder left
→ Elbow down
→ No handflip



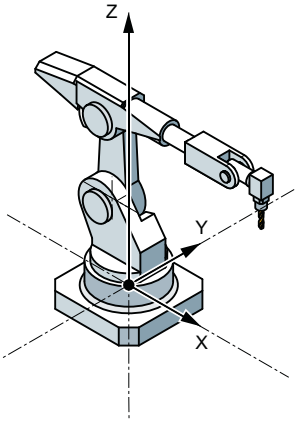
`STAT=2 ('B010')`

→ Shoulder right
→ Elbow up
→ No handflip



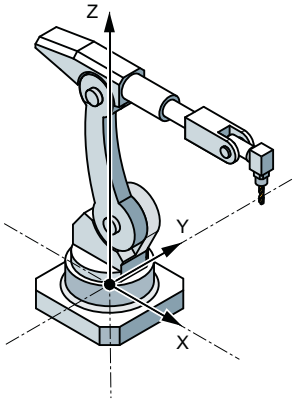
`STAT=5 ('B101')`

→ Shoulder left
→ Elbow down
→ Handflip



`STAT=6 ('B110')`

→ Shoulder right
→ Elbow up
→ Handflip



TRANSMIT

For TRANSMIT, the STAT address is used to initiate the equivocality regarding the pole.

The following applies if the rotary axis must rotate through 180º or the contour for CP would go through the pole:

Bit 0	Only relevant for $MC_TRANSMIT_POLE_SIDE_FIX_1/2 = 1 or 2:	
	= 0	Rotary axis traverses through +180º or rotates clockwise.
	= 1	Rotary axis rotates through -180º or rotates counterclockwise.
Bit 1	Only relevant for $MC_TRANSMIT_POLE_SIDE_FIX_1/2 = 0:	
	= 0	The axis traverses through the pole. The rotary axis does not rotate.
	= 1	The axis rotates around the pole. Bit 0 of STAT is relevant.

2.6.3.4 Specify the sign of the axis angle (TU)

In order that rotary axes can also approach axis angles exceeding +180° or less than -180° without requiring a special traversing strategy (e.g. intermediate point), the sign of the axis angle must be specified under the adjustable address TU.

Note

The control only takes into account programmed TU values for PTP motion. CP motion is ignored.

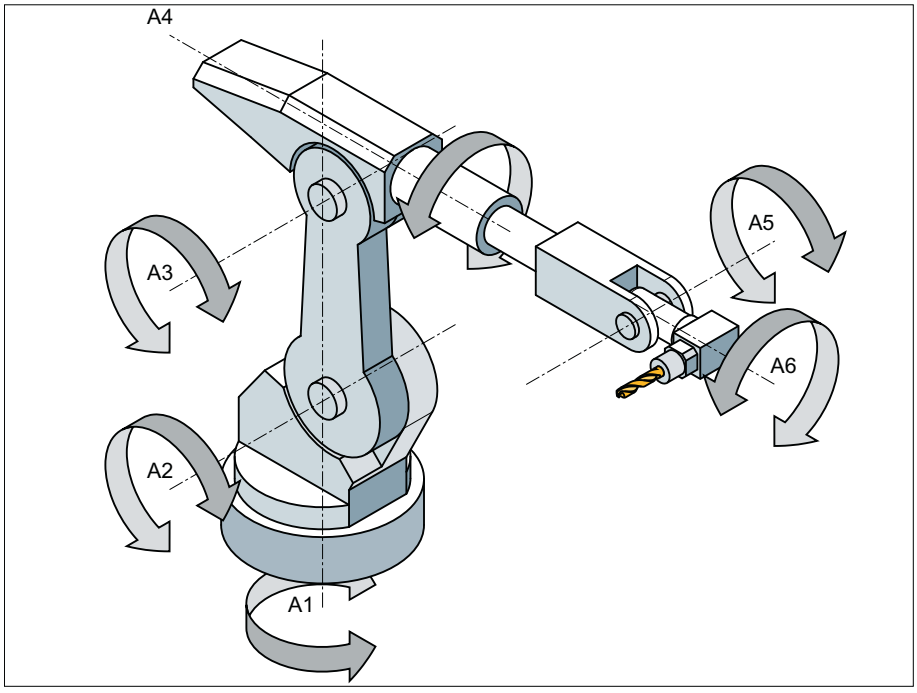
Syntax

```
TU=<Value>
```

Meaning

`TU`:	Adjustable address to specify axis angle signs			
`<Value>`:	Binary or decimal value For each axis that is involved in the transformation, there is a bit that indicates the sign of the axis angle (θ), and therefore the traversing direction.			
	Bit	= 0	Axis angle sign: +	Axis angular range: 0° ≤ θ < 360°
		= 1	Axis angle sign: -	Axis angular range: - 360° < θ < 0°

Example: 6-axis articulated robot



Bit	Meaning	Value	Axis angle sign	Axis angle
Bit 0 [1]	Sign for the axis angle of A1	= 0	+	≥ 0°
		= 1	-	< 0°
Bit 1 [1]	Sign for the axis angle of A2	= 0	+	≥ 0°
		= 1	-	< 0°
Bit 2 [1]	Sign for the axis angle of A3	= 0	+	≥ 0°
		= 1	-	< 0°
Bit 3 [1]	Sign for the axis angle of A4	= 0	+	≥ 0°
		= 1	-	< 0°
Bit 4 [1]	Sign for the axis angle of A5	= 0	+	≥ 0°
		= 1	-	< 0°
Bit 5 [1]	Sign for the axis angle of A6	= 0	+	≥ 0°
		= 1	-	< 0°

[1] The actual TU bit numbers obtained from the channel axis numbers of the robot axes! In the example, robot axes (A1 to A6) are the first six axes in the channel; as a consequence, TU bits 0 ... 5 are used. For another channel axis assignment of the robot axes, the TU bit numbers of the robot axes would correspondingly change (e.g.: robot axes are the 3rd to 8th channel axis, i.e. TU bits 2 ... 7 are used for the robot axes).

TU=19 (corresponds to TU='B010011) would therefore signify:

Bit	Value		Axis angle
0	= 1	①	$\theta_{A1} < 0°$
1	= 1	①	$\theta_{A2} < 0°$

2	= 0	①	$\theta_{A3} \geq 0°$
3	= 0	①	$\theta_{A4} \geq 0°$
4	= 1	①	$\theta_{A5} < 0°$
5	= 0	①	$\theta_{A6} \geq 0°$

Note

In the case of axes with a traversing range > ±360°, the axis always moves along the shortest path because the axis position cannot be specified uniquely by the TU information.

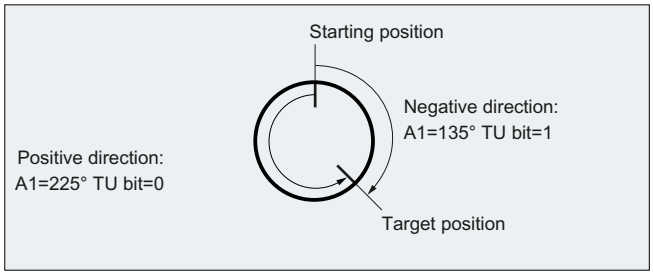
If no TU is programmed for a position, then depending on MD30455 $MA_MISC_FUNCTION_MASK, the shorter or longer path is traversed (see Chapter "Taking into account the software limits for PTP travel (Page 100)" in the Extended Functions Function Manual).

TRANSMIT

For PTP travel with TRANSMIT active, the address of TU has no meaning!

Example

The rotary axis position shown in the following diagram can be approached in the negative or positive direction. The angular position is programmed under address A1. The traversing direction is only absolutely clear when TU is specified.



2.6.3.5 Example 1: PTP travel of a 6-axis robot with ROBX transformation

In the following application example, Cartesian PTP travel and the associated NC commands are shown in the form of an example.

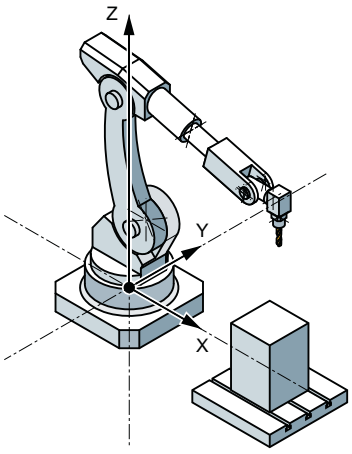


Figure 2-18 6-axis articulated robot with milling spindle

```
N1 G90
N2 T="T8MILLD20" D1 M6
N3 TRAORI
;$P_UIFR[1]=CTRANS(X,1500,Y,0,Z,400):CROT(X,0,Y,0,Z,-90)
N4 G54
N5 M3 S20000
N6 ORIWKS
N7 ORIVIRT1
N8 CYCLE832(0.01,_FINISH,1)
;HOME
N9 TRAFOOF
N10 G0 RA1=0.0000 RA2=-90.0000 RA3=90.0000 A=0.0000 B=90.0000 C=0.0000
N11 TRAORI
N12 G54
N13 G0 PTP X1369.2426 Y956.7528 Z502.5517 A=135.5761 B=-33.2223
C=161.1435 STAT='B010' TU='B001011'
N14 G0 X1355.1242 Y1014.9394 Z424.9695 A=135.8491 B=-33.1439
C=160.9941 STAT='B010' TU='B001011'
N15 G1 CP X1354.8361 Y1016.1269 Z423.3862 A=136.0635 B=-33.0819 C=160.8770
F1000
N16 G1 X1336.4283 Y1016.1269 Z426.6311 A=136.0484 B=-32.2151 C=160.9643
F2000
N17 G1 X1317.9831 Y1016.1269 Z429.6730 A=136.0175 B=-31.3394 C=161.0655
;HOME
N18 TRAFOOF
N19 G0 RA1=0.0000 RA2=-90.0000 RA3=90.0000 A=0.0000 B=90.0000 C=0.0000
N20 M30
```

2.6.3.6 Example 2: PTP travel for generic 5-axis transformation

Assumption: Right-angled CA kinematics used as basis.

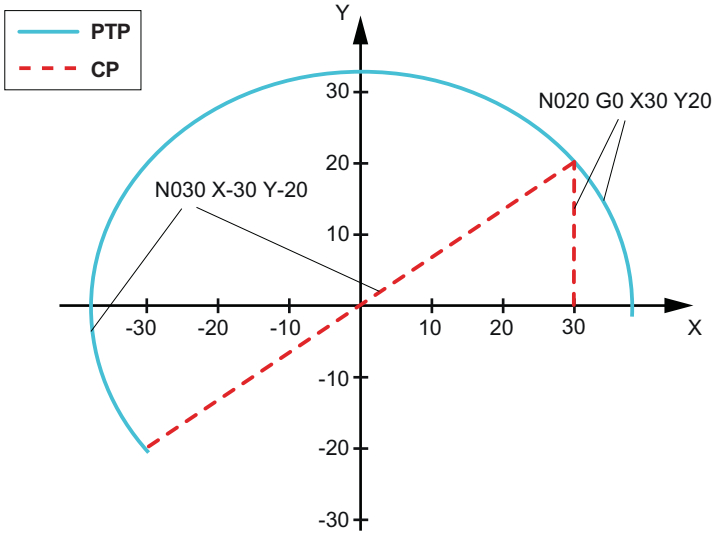
Program code	Comment
TRAORI	;Transformation CA kinematics ON
PTP	; Activate PTP traversal
N10 A3=0 B3=0 C3=1	; rotary axis positions C=0 A=0
N20 A3=1 B3=0 C3=1	; rotary axis positions C=90 A=45
N30 A3=1 B3=0 C3=0	; rotary axis positions C=90 A=90
N40 A3=1 B3=0 C3=1 **STAT=1**	; rotary axis positions C=270 A=-45

Select clear approach position of rotary axis position:

In block N40, the rotary axes – as a result of the programming of **STAT=1** – travel the longer distance from their start point (C=90, A=90) to the end point (C=270, A=–45). On the other hand, with **STAT=0**, the rotary axes would travel along the shortest path to the end point (C=90, A=45).

2.6.3.7 Example 3: PTPG0 and TRANSMIT

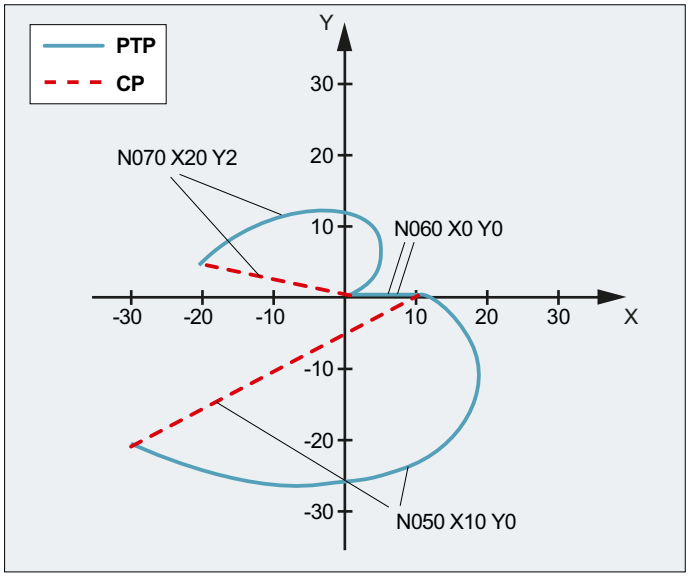
Traversing around the pole with PTPG0 and TRANSMIT



Program code	Comment
N001 G0 X30 Z0 F10000 T1 D1 G90	;Initial setting absolute dimension
N002 SPOS=0	
N003 **TRANSMIT**	;TRANSMIT transformation
N010 **PTPG0**	; for each G0 block, automatically PTP – and then CP again.
N020 G0 X30 Y20	
N030 X-30 Y-20	

Program code	Comment
N120 G1 X30 Y20	
N110 X30 Y0	
M30	

Traversing from the pole with PTPG0 and TRANSMIT



Programming	Comment
N001 G0 X90 Z0 F10000 T1 D1 G90	;Initial setting
N002 SPOS=0	
N003 **TRANSMIT**	;TRANSMIT transformation
N010 **PTPG0**	; for each G0 block, automatically PTP – and then CP again.
N020 G0 X90 Y60	
N030 X-90 Y-60	
N040 X-30 Y-20	
N050 X10 Y0	
N060 X0 Y0	
N070 X-20 Y2	
N170 G1 X0 Y0	
N160 X10 Y0	
N150 X-30 Y-20	
M30	

2.6.4 Supplementary conditions

Tool radius compensation (TRC)

With PTP/PTPWOC, no tool radius compensation (TRC) can be active, because PTP/PTPWOC creates contours other than those calculated by the TRC.

The response is different for PTPG0. Here, with active TRC, an internal switchover to CP takes place so that the TRC is performed correctly.

Smooth approach and retraction (SAR)

If PTP/PTPWOC is active, SAR cannot be used, because SAR needs a contour to design the approach and retraction and to be able to approach or retract tangentially.

With PTPG0, on the other hand, SAR is possible because the blocks required for SAR are traversed with CP. The G0 blocks up to the actual approach contour are executed with PTP travel and therefore quickly. The same applies to the retract blocks.

Chipping cycles

With active PTP/PTPWOC, chipping cycles such as CONTPRON or CONTDCON cannot be applied because the chipping require a contour to be able to construct the cut segmentation.

With PTPG0, on the other hand, chipping cycles are possible because the blocks required for the chipping cycles are traversed with CP.

Chamfer and rounding

Chamfer and rounding are ignored during PTP travel.

Axis overlay

An axis superimposing in the interpolation may not change during the PTP sequence. This applies, for example, to LIFTFAST, fine tool offset, coupled motion TRAILON and tangential follow-up TANGON.

NC block compression

In the PTP blocks, an active compressor function (COMPON, COMPCURV, COMPCAD or COMPSURF) is automatically deselected.

Blending

Blending with G643 is not possible with the PTP travel. In the PTP blocks with an active G643, there is an automatic switchover to G642.

In addition, the following special features should be noted:

- Blending is always done in the machine coordinate system and it thus high-performance in terms of time. The criteria for G642 is always interpreted in the machine coordinate system and not as usual in the Cartesian basic coordinate system.
- Smoothing G641 is determined as a function of the fictitious path calculated from the machine axis coordinates.

Path feedrate

An F value input with G1 refers to the fictitious path calculated from the machine axis coordinates.

Mode change

Cartesian PTP travel only makes sense in the modes AUTOMATIC and MDI.

When changing the mode to JOG, the current setting is retained:

- If PTP travel is set, the axes will traverse in MCS.
- If CP is set, the axes will traverse in WCS.

After returning to the AUTOMATIC or MDI mode, the status that was active before the switch to JOG mode is restored.

Block search

If the face end transformation (TRANSMIT) is active, different machine axis positions for the same Cartesian position can result if a program section is executed with block search.

REPOS

The setting for Cartesian PTP travel is not altered during repositioning. If PTP was set in the interruption block, then repositioning also takes place with PTP.

2.7 Cartesian manual traversing (option)

Note

The "Handling transformation package" option is necessary for the "Cartesian manual travel" function.

Function

The "Cartesian manual travel" function, as a reference system for JOG mode, allows axes to be set independently of each other in the following Cartesian coordinate systems:

- Basic coordinate system (BCS)
- Workpiece coordinate system (WCS)
- Tool coordinate system (TCS)

Adjustment and activation is done using machine data:

MD21106 $MC_CART_JOG_SYSTEM (coordinate systems for Cartesian JOG)

Bit	Meaning
0	Basic coordinate system
1	Workpiece coordinate system
2	Tool coordinate system

Note

The workpiece coordinate system has been shifted and rotated compared to the basic coordinate system via frames.

Further information:
Function Manual Basic Functions; Axes, coordinate systems, frames

Representation of the reference points in the coordinate system:

	Workpiece zero		Tool carrier reference point

Selecting reference systems

For JOG motion, one of the three reference systems can be specified separately not only for the **translation** (coarse offset) with geometry axes, but also for the **orientation** with orientation axes via the following setting data:

SD42650 $SC_CART_JOG_MODE

If more than one bit is set for the translation or orientation reference system, or when an attempt is made to set a reference system which was not released by the MD21106 $MC_CART_JOG_SYSTEM, the alarm 14148 "Reference system for Cartesian manual travel not allowed" will be generated.

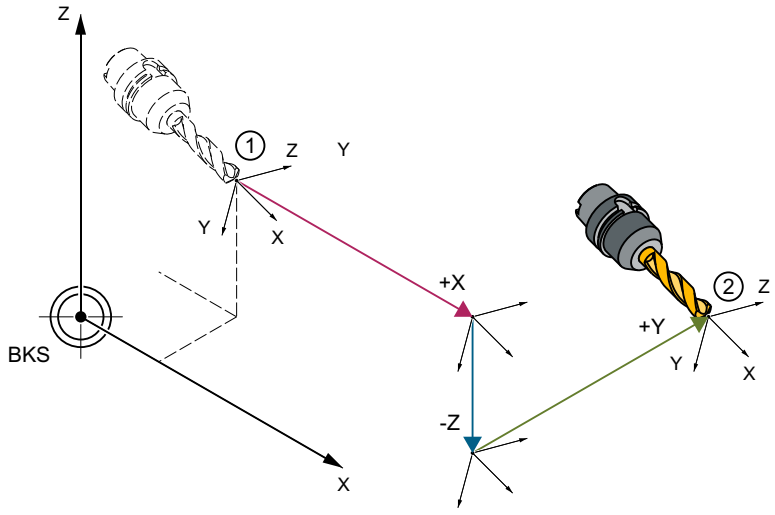
Translation

A translation movement can be used to move the tool center point (TCP) in parallel and 3-dimensional to the axes of the reference system. The traversing movement is made via the VDI signals of the geometry axes.

Machine data MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_x[n] is used to assign the geometry axes. Simultaneous traversing in more than one direction permits the execution of movements that lie parallel to the directions of the reference system.

Translation in the BCS

The basic coordinate system (BCS) describes the Cartesian zero of the machine.

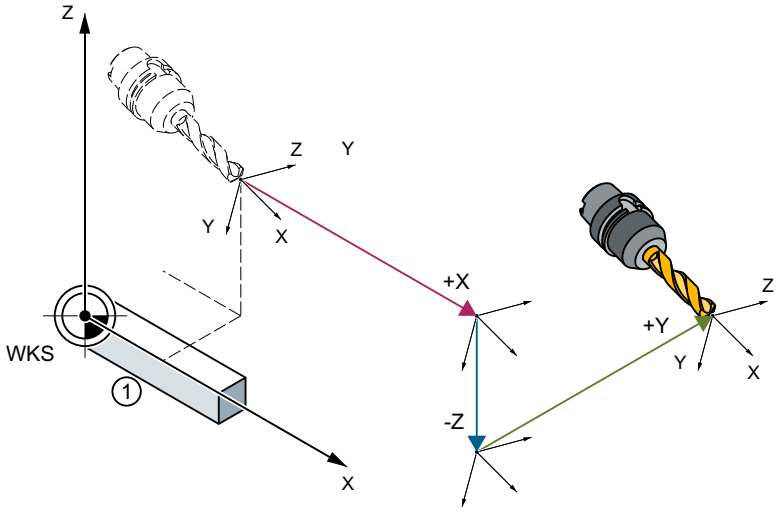


① Starting position
② End position

Figure 2-19 Cartesian manual travel in the basic coordinate system (translation)

Translation in the WCS

The workpiece coordinate system (WCS) lies in the workpiece zero. The workpiece coordinate system can be shifted and rotated relative to the basic coordinate system via frames. As long as no frame rotation is active, traversing motion corresponds to the translation of motion in the basic coordinate system.

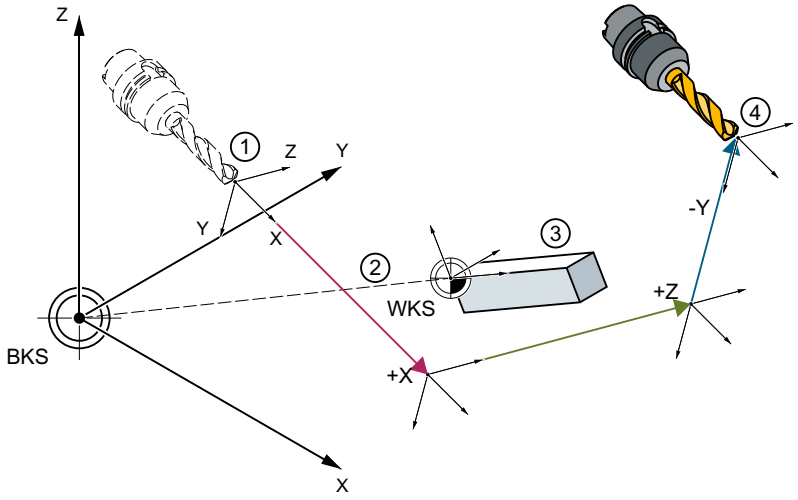


① Workpiece

Figure 2-20 Cartesian manual travel in the workpiece coordinate system (translation)

Translation in the TCS

The tool coordinate system (TCS) is at the tool center point. Its direction depends on the current setting of the machine, since the tool coordinate system moves during the motion.



① Starting position
② Frame
③ Workpiece
④ End position

Figure 2-21 Cartesian manual travel in the tool coordinate system (translation)

Translation and orientation simultaneously in the TCS

If translation and orientation movements are executed at the same time, the translation is always traversed corresponding to the current orientation of the tool. This permits infeed movements that are made directly in the tool direction or movements that run perpendicular to tool direction.

Orientation

The tool can be aligned to the component surface via an orientation movement. The orientation movement is given control from the PLC via the VDI signals of the orientation axes (DB21, ... DBB321).

Several orientation axes can be traversed simultaneously. The virtual orientation axes execute rotations around the fixed axes of the relevant reference system.

The **rotations** are identified according to the RPY angles.

- A angle: Rotation around the Z axis
- B angle: Rotation around the Y axis
- C angle: Rotation around the X axis

Programming rotations:

The user can define how rotations are to be executed using the current G commands of group 50 for orientation definition

Specifying `ORIEULER`, `ORIRPY`, `ORIVIRT1` and `ORIVIRT2`.

With ORIVIRT1, rotation is executed according to MD21120 $MC_ORIAX_TURN_TAB_1. The orientation axes are assigned to the channel axes via machine data: MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1.

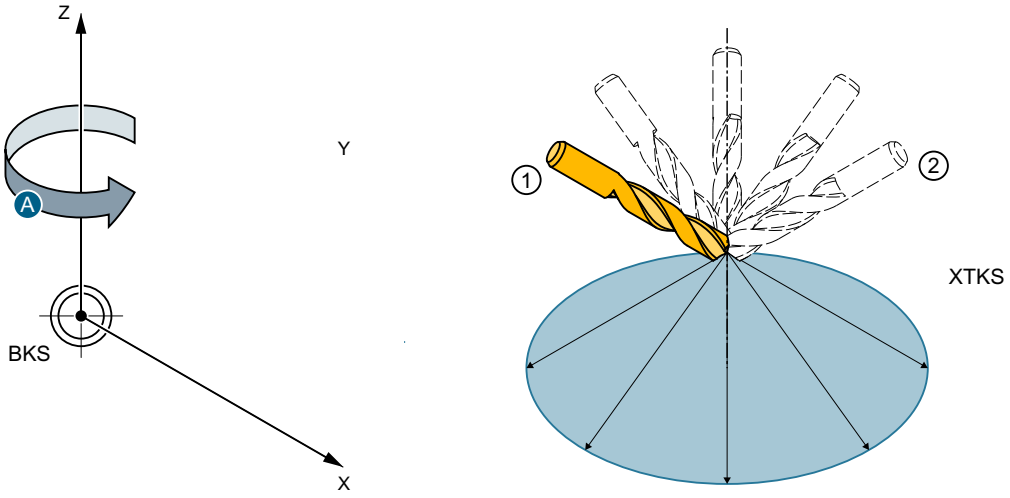
The **direction of rotation** is determined according to the "right hand rule". The thumb points in the direction of the rotary axis. The finger stipulates the positive direction of rotation.

Orientation in the WCS

The rotations are made around the defined directions of the workpiece coordinate system. If frame rotation is active, the movements correspond to the rotations in the basic coordinate system.

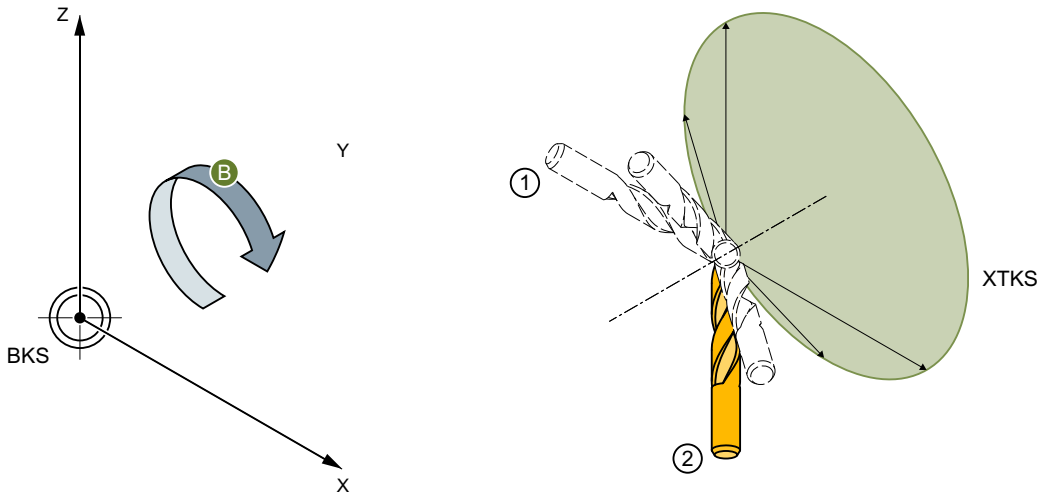
Orientation in the BCS

The rotations are made around the defined directions of the basic coordinate system.



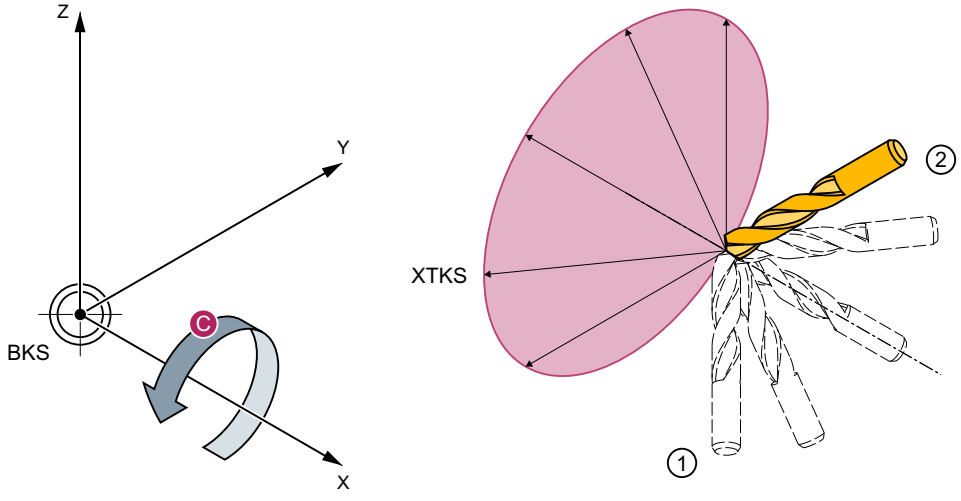
① Start orientation
② End orientation

Figure 2-22 Cartesian manual travel in the basic coordinate system, orientation angle A



① Start orientation
② End orientation

Figure 2-23 Cartesian manual travel in the basic coordinate system, orientation angle B



① Start orientation
② End orientation

Figure 2-24 Cartesian manual travel in the basic coordinate system, orientation angle C

Orientation in the TCS

The rotations are around the moving directions in the tool coordinate system. The current homing directions of the tool are always used as rotary axes.

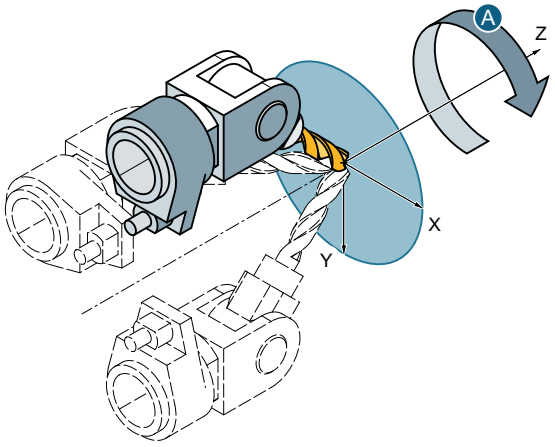


Figure 2-25 Cartesian manual travel in the tool coordinate system, orientation angle A

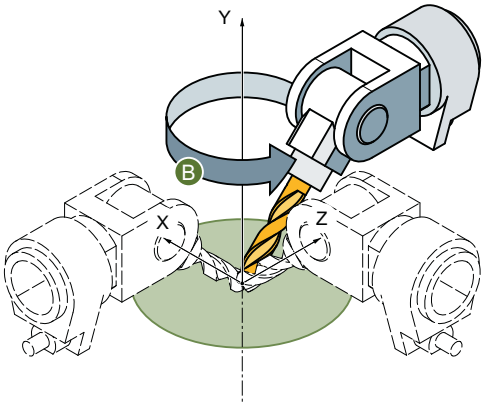


Figure 2-26 Cartesian manual travel in the tool coordinate system, orientation angle B

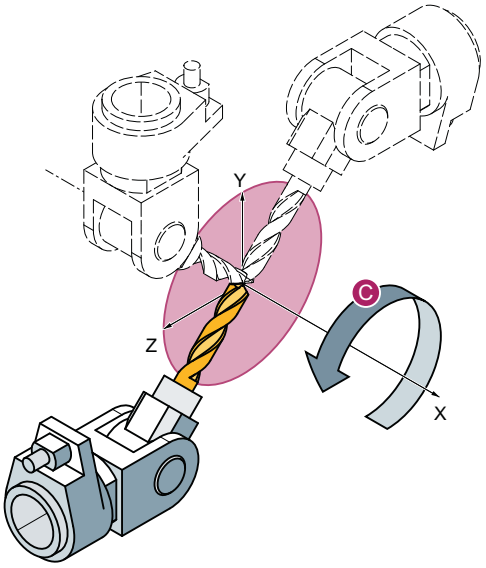


Figure 2-27 Cartesian manual travel in the tool coordinate system, orientation angle C

Boundary conditions

The "Cartesian manual travel" function can only be executed if the transformation is active in the NC: DB21, ... DBX33.6 == 1 ("transformation active")

The following supplementary conditions must be observed:

- "Handling transformation package" option with 5-axis or 6-axis transformation is set
- Virtual orientation axes must be defined via the following machine data:
  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[n]
- DB21, ... DBX29.4 == 0 (activate PTP travel)
- MD21106 $MC_CART_JOG_SYSTEM > 0

Table 2-2 Conditions for Cartesian manual travel

Transformation in program active (TRAORI..)	Prog. traversing type	DB21, ... DBX29.4 "Activate PTP travel"	DB21, ... DBX33.6 "Transformation active"
FALSE	Not active	Not active	0
TRUE	CP	0	1
TRUE	CP	1	0

Transformation in program active (TRAORI..)	Prog. traversing type	DB21, ... DBX29.4 "Activate PTP travel"	DB21, ... DBX33.6 "Transformation active"
TRUE	PTP	0	1
TRUE	PTP	1	0

The G command PTP/CP currently active in the program does not affect Cartesian manual travel. The NC/PLC interface signals are interpreted in the channel DB for geometry and orientation axes.

Activation

The reference system for Cartesian manual travel is set as follows:

- The Cartesian manual travel function is activated with the following machine data:
  MD21106 $MC_CART_JOG_SYSTEM > 0
  The BCS, WCS or TCS reference systems are enabled via MD21106 $MC_CART_JOG_SYSTEM.
- JOG traverse motion via SD42650 SC_CART_JOG_MODE
  Standard behavior as before: Bits 0 to 2 = 0, bits 8 to 10 = 0.
  Reference system for translation via bits 0-2 and the reference system for orientation via bits 8-10.
  If not all of the bits are set to 0, the process uses the new function. The reference systems for translation and orientation may be set independently.

SD42650 $SC_CART_JOG_MODE (only set one bit):

SD42650 $SC_CART_JOG_MODE							
Bit 11 - bit 15	Bit 10	Bit 9	Bit 8	Bit 7 - bit 3	Bit 2	Bit 1	Bit 0
Reserved	Orientation in the TCS	Orientation in the WCS	Orientation in the BCS	Reserved	Translation in the TCS	Translation in the WCS	Translation in the BCS

Combining reference systems

The table below shows all the combination options for reference systems.

SD42650 $SC_CART_JOG_MODE						Reference system for	
Bit 10	Bit 9	Bit 8	Bit 2	Bit 1	Bit 0	Orientation	Translation
0	0	0	0/1	0/1	0/1	Standard	Standard
Standard	Standard	Standard	0	0	0	Standard	Standard
0	0	1	0	0	1	BCS	BCS
0	0	1	0	1	0	BCS	WCS
0	0	1	1	0	0	BCS	TCS
0	1	0	0	0	1	WCS	BCS
0	1	0	0	1	0	WCS	WCS
0	1	0	1	0	0	WCS	TCS
1	0	0	0	0	1	TCS	BCS

SD42650 $SC_CART_JOG_MODE						Reference system for	
1	0	0	0	1	0	TCS	WCS
1	0	0	1	0	0	TCS	TCS

2.8 Activating transformation machine data via part program/softkey

2.8.1 Function

Transformation MD can now be activated by means of a program command softkey, i.e. these can, for example, be written from the parts program, thus altering the transformation configuration completely.

Up to ten different transformations can be set in the control system. The transformation type is set in the following machine data:

MD24100 $MC_TRAFO_TYPE_1

up to

MD24460 $MC_TRAFO_TYPE_10.

Characteristics

The transformation MDs are activated using the "Setting machine data effective".

The protection level is now 7/7 (KEYSWITCH_0), which means that data can be modified from the NC program without any particular authorization.

If the machine data are activated (regardless whether via the NEWCONF NC program command, the HMI or implicitly following reset or end of program) - and no transformation has been selected (activated) - then the machine data listed above can be altered without restriction and then activated.

Of particular relevance is that new transformations can be configured or existing transformations replaced by one of a different type or deleted, since the modification options are not restricted to re-parameterization of existing transformations.

2.8.2 Constraints

Changing machine data

The machine data which involve an active transformation may not be altered; any attempt to do so will generate an alarm.

Generally, these are all machine data, which are assigned a transformation via the associated transformation data group. Machine data, which although contained in the group of an active transformation, but are not used, can be changed (however, this could hardly be considered an operation that makes sense). For example, it would be possible to change machine data

MD24564 $MC_TRAFO5_NUTATOR_AX_ANGLE_n for an active transformation with MD24100 $MC_$MC_TRAFO_TYPE = 16 (5-axis transformation with rotatable tool and two mutually perpendicular rotary axes A and B) since this particular machine data is not involved in the transformation.

Please note that machine data MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE may not be altered for an active orientation transformation.

Note

For a program interruption (Repos, delete distance to go, ASUPs etc.) to restart, several blocks are also required in the control, which were already executed. That it is prohibited to change machine data of an active transformation also refers to these blocks.

Example:

Two orientation transformations are set via machine data, e.g. MD24100 $MC_TRAFO_TYPE_1 = 16, MD24200 $MC_TRAFO_TYPE_2 = 18.

When executing "Activate machine data", it is assumed that the second transformation is active. In this case, all machine data that relate only to the first transformation may be changed, e.g.:

MD24500 $MC_TRAFO5_PART_OFFSET_1

but not, for instance:

MD24650 $MC_TRAFO5_BASE_TOOL_2

or

MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE

Furthermore, another transformation (TRANSMIT) can be set, for example with MD24300 $MC_TRAFO_TYPE_3 = 256 and can be parameterized with additional machine data.

Defining geometry axes

Geometry axes must be defined **before** the control system powers up using the following machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_X[n]

or

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[n]

Changing the assignment

The assignment of a transformation data set to a transformation is determined by the sequence of entries in MD24100 $MC_TRAFO_TYPE_X. The first entry in the table is assigned to the first transformation data set, and accordingly the second entry to the second data set. It is not permissible (and not possible) to change this assignment for an active transformation.

Example:

Three transformations are set. two orientation transformations and one transmit transformation, e.g.

MD24100 $MC_TRAFO_TYPE_1 = 16

; orientation transformation 1: Orientation transformation data set

MD24200 $MC_TRAFO_TYPE_2 = 256 : Transmit transformations

MD24300 $MC_TRAFO_TYPE_3 = 18

; orientation transformation 2: Orientation transformation data set

The first data set for orientation transformations is assigned to the first transformation (at the same time, the first orientation transformation) - and the second transformation data set to the third transformation (at the same time, the the second orientation transformation).

If the third transformation is active when executing the "Activate machine data" function, then it is not permissible to change the first transformation into a transformation of another group (e.g. TRACYL) since, in this case, the third transformation would then not become the second orientation transformation, but the first.

In the above example, however, it is permissible to set another orientation transformation for the first transformation (e.g. using MD24100 $MC_TRAFO_TYPE_1 = 32) or a transformation from another group as the first transformation (e.g. using $MD24100 $MC_TRAFO_TYPE_1 = 1024, TRAANG), if the second transformation is changed into an orientation transformation at the same time, e.g. with MD24200 $MC_TRAFO_TYPE_2 = 48.

2.8.3 Control response to power ON, mode change, RESET, block search, REPOS

With the aid of the following machine data it is possible to select a transformation automatically in response to `RESET` (i.e. at end of program as well) and/or on program start:

MD20110 $MC_RESET_MODE_MASK

MD20112 $MC_START_MODE_MASK

and

MD20140 $MC_TRAFO_RESET_VALUE

This may result in the generation of an alarm, for example, at the end or start of a program, if the machine data of an active transformation has been altered.

To avoid this problem when re-configuring transformations via an NC program, we therefore recommend that NC programs are structured as follows:

Program code	**Comment**
`N10 TRAFOOF()`	`; Select a possibly still active transformation`
`N20$MC_TRAFO5_BASE_TOOL_1[0]=0`	`; Enter machine data`
`N30$MC_TRAFO5_BASE_TOOL_1[0]=3`	
`N40$MC_TRAFO5_BASE_TOOL_1[0]=200`	
`N130 NEWCONF`	`; Newly entered machine data`
	`; Transfer`
`N140 M30`	

2.8.4 List of machine data affected

The machine data that can be activated are listed below.

All transformations

Machine data which are relevant for all transformations:

- MD24100 $MC_TRAFO_TYPE_1 to MD24480 $MC_TRAFO_TYPE_10
- MD24110 $MC_TRAFO_AXES_IN_1 to MD24482 $MC_TRAFO_AXES_IN_10
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 to MD24484 $MC_TRAFO_GEOAX_ASSIGN_TAB_10

Orientation transformations

Machine data that are relevant for orientation transformations:

- MD24550 $MC_TRAFO5_BASE_TOOL_1 and MD24650 $MC_TRAFO5_BASE_TOOL_2
- MD24558 $MC_TRAFO5_JOINT_OFFSET_1 and MD24658 $MC_TRAFO5_JOINT_OFFSET_2
- MD24500 $MC_TRAFO5_PART_OFFSET_1 and MD24600 $MC_TRAFO5_PART_OFFSET_2
- MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1 and MD24610 $MC_TRAFO5_ROT_AX_OFFSET_2
- MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1 and MD24620 $MC_TRAFO5_ROT_SIGN_IS_PLUS_2
- MD 24530: TRAFO5_NON_POLE_LIMIT_1 and MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2
- MD24540 $MC_TRAFO5_POLE_LIMIT_1 and MD24640 $MC_TRAFO5_POLE_LIMIT_2
- MD24570 $MC_TRAFO5_AXIS1_1 and MD24670 $MC_TRAFO5_AXIS1_2
- MD24572 $MC_RAFO5_AXIS2_1 and MD24672 $MC_TRAFO5_AXIS2_2
- MD24574 $MC_TRAFO5_BASE_ORIENT_1 and MD24674 $MC_TRAFO5_BASE_ORIENT_2
- MD24562 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_1 and MD24662 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_2
- MD24564 $MC_TRAFO5_NUTATOR_AX_ANGLE_1 and MD24664 $MC_TRAFO5_NUTATOR_AX_ANGLE_2
- MD24566 $MC_TRAFO5_NUTATOR_VIRT_ORIAX_1 and MD24666 $MC_TRAFO5_NUTATOR_VIRT_ORIAX_2

Transmit transformations

Machine data that are relevant for transmit transformations:

- MD24920 $MC_TRANSMIT_BASE_TOOL_1 and
  MD24970 $MC_TRANSMIT_BASE_TOOL_2
- MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 and
  MD24950 $MC_TRANSMIT_ROT_AX_OFFSET_2
- MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 and
  MD24960 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_2
- MD24911 MC_RANSMIT_POLE_SIDE_FIX_1 and
  MD24961 $MC_TRANSMIT_POLE_SIDE_FIX_2

Tracyl transformations

Machine data that are relevant for Tracyl transformations:

- MD24820 $MC_TRACYL_BASE_TOOL_1 and
  MD24870 $MC_TRACYL_BASE_TOOL_2
- MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 and
  MD24850 $MC_TRACYL_ROT_AX_OFFSET_2
- MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 and
  MD24870 $MC_TRACYL_ROT_SIGN_IS_PLUS_2
- MD24808 $MC_TRACYL_DEFAULT_MODE_1 and
  MD24858 $MC_TRACYL_DEFAULT_MODE_2

Inclined axis transformations

Machine data that are relevant for inclined axis transformations:

- MD24710 $MC_TRAANG_BASE_TOOL_1 and
  MD24760 $MC_TRAANG_BASE_TOOL_2
- MD24700 $MC_TRAANG_ANGLE_1 and
  MD24750 $MC_TRAANG_ANGLE_2
- MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 and
  MD24770 $MC_TRAANG_PARALLEL_VELO_RES_2
- MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 and
  MD24771 $MC_TRAANG_PARALLEL_ACCEL_RES_2

Chained transformations

Machine data that are relevant for chained transformations:

- MD24995 $MC_TRACON_CHAIN_1 and
  MD24996 $MC_TRACON_CHAIN_2
- MD24997 $MC_TRACON_CHAIN_3 and
  MD24998 $MC_TRACON_CHAIN_4

Persistent transformation

Machine data that are relevant for persistent transformations:

- MD20144 $MC_TRAFO_MODE_MASK
- MD20140 $MC_TRAFO_RESET_VALUE
- MD20110 $MC_RESET_MODE_MASK and
  MD20112 $MC_START_MODE_MASK

Not transformation-specific

Machine data that are not transformation-specific. They are not uniquely assigned to a specific transformation data set - or are also of significance outside an active transformation:

- MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE
- MD21090 $MC_ MAX_LEAD_ANGLE
- MD21092 $MC_ MAX_TILT_ANGLE
- MD21100 $MC_ORIENTATION_IS_EULER

2.8.5 Example

It would be permissible in the following example to reconfigure (write) a machine data affecting the second transformation (e.g. MD24650 $MC_TRAFO5_BASE_TOOL_2[2]) in block N90, since writing a machine data alone does not activate it. However, if the program remained otherwise unchanged, an alarm would occur in block N130, because an attempt would then be made to modify an active transformation.

Example program:

```
Program code                                  Comment
N40 TRAORI(2)                                 ; Select 2nd orientation transformation
N50 X0 Y0 Z0 F20000 T1 T1
N60 A50 B50
N70 A0 B0
N80 X10
N90 $MC_TRAFO5_BASE_TOOL_1[2] = 50            ; Overwrite a machine data item of the
                                                1st orientation transformation
N100 A20
N110 X20
N120 X0
N130 NEWCONF                                  ; Accept new machine data
N140 TRAORI(1)                                ; Selection of the 1st orientation
                                                transformation MD is effective
N150 G19 X0 Y0 Z0
N160 A50 B50
N170 A0 B0
N180 TRAFOOF
```

Program code	Comment
N190 M30	

2.9 Data lists

2.9.1 Machine data

2.9.1.1 TRANSMIT

Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Geo-axis assignment for 6th transformation
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation

Number	Identifier: $MC_	Description
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24900	TRANSMIT_ROT_AX_OFFSET_1	Deviation of rotary axis from zero position in degrees (1st TRANSMIT)
24910	TRANSMIT_ROT_SIGN_IS_PLUS_1	Sign of rotary axis for TRANSMIT (1st TRANSMIT)
24911	TRANSMIT_POLE_SIDE_FIX_1	Limitation of working range in front of/behind pole, 1st transformation
24920	TRANSMIT_BASE_TOOL_1	Distance of tool zero point from origin of geo-axes (1st TRANSMIT)
24950	TRANSMIT_ROT_AX_OFFSET_2	Deviation of rotary axis from zero position in degrees (2nd TRANSMIT)
24960	TRANSMIT_ROT_SIGN_IS_PLUS_2	Sign of rotary axis for TRANSMIT (2nd TRANSMIT)
24961	TRANSMIT_POLE_SIDE_FIX_2	Limitation of working range in front of/behind pole, 2nd transformation
24970	TRANSMIT_BASE_TOOL_2	Distance of tool zero point from origin of geo-axes (2nd TRANSMIT)

2.9.1.2 TRACYL

Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
20144	TRAFO_MODE_MASK	Selection of the kinematic transformation function
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24130	TRAFO_INCLUDES_TOOL_1	Tool handling with active transformation 1.
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24230	TRAFO_INCLUDES_TOOL_2	Tool handling with active transformation 2.
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24330	TRAFO_INCLUDES_TOOL_3	Tool handling with active transformation 3.
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24426	TRAFO_INCLUDES_TOOL_4	Tool handling with active transformation 4.
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel

Number	Identifier: $MC_	Description
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24436	TRAFO_INCLUDES_TOOL_5	Tool handling with active transformation 5.
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Assignment geometry axes for 6th transformation
24446	TRAFO_INCLUDES_TOOL_6	Tool handling with active transformation 6.
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24456	TRAFO_INCLUDES_TOOL_7	Tool handling with active transformation 7.
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24466	TRAFO_INCLUDES_TOOL_8	Tool handling with active transformation 8.
24470	TRAFO_TYPE_9	Definition of the 9th transformation in channel
24472	TRAFO_AXES_IN_9	Axis assignment for the 9th transformation
24474	TRAFO_GEOAX_ASSIGN_TAB_9	Geo-axis assignment for 9th transformation
24476	TRAFO_INCLUDES_TOOL_9	Tool handling with active transformation 9.
24480	TRAFO_TYPE_10	Definition of the 10th transformation in channel
24482	TRAFO_AXES_IN_10	Axis assignment for the 10th transformation
24484	TRAFO_GEOAX_ASSIGN_TAB_10	Geo-axis assignment for 10th transformation
24486	TRAFO_INCLUDES_TOOL_10	Tool handling with active transformation 10.
24800	TRACYL_ROT_AX_OFFSET_1	Deviation of rotary axis from zero position in degrees (1st TRACYL)
24808	TRACYL_DEFAULT_MODE_1	Selection of TRACYL mode (1st TRACYL)
24810	TRACYL_ROT_SIGN_IS_PLUS_1	Sign of rotary axis for TRACYL (1st TRACYL)
24820	TRACYL_BASE_TOOL_1	Distance of tool zero point from origin of geo-axes (1st TRACYL)
24850	TRACYL_ROT_AX_OFFSET_2	Deviation of rotary axis from zero position in degrees (2nd TRACYL)
24858	TRACYL_DEFAULT_MODE_2	Selection of TRACYL mode (2nd TRACYL)
24860	TRACYL_ROT_SIGN_IS_PLUS_2	Sign of rotary axis for TRACYL (2nd TRACYL)
24870	TRACYL_BASE_TOOL_2	Distance of tool zero point from origin of geo-axes (2nd TRACYL)
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover

2.9.1.3 TRAANG

Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
20144	RAFO_MODE_MASK	Selection of the kinematic transformation function
20534	TRAFO_CHANGE_M_CODE	M code for transformation changeover
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Geo-axis assignment for 6th transformation
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24700	TRAANG_ANGLE_1	Angle of inclined axis in degrees (1st TRAANG)
24710	TRAANG_BASE_TOOL_1	Distance of tool zero point from origin of geometry axes (1st TRAANG)
24720	TRAANG_PARALLEL_VELO_RES_1	Velocity reserve of parallel axis for compensatory motion (1st TRAANG)
24721	TRAANG_PARALLEL_VELO_RES_2	Velocity reserve of parallel axis for compensatory motion (2nd TRAANG)
24750	TRAANG_ANGLE_2	Angle of inclined axis in degrees (2nd TRAANG)

Number	Identifier: $MC_	Description
24760	TRAANG_BASE_TOOL_2	Distance of tool zero point from origin of geometry axes (2nd TRAANG)
24770	TRAANG_PARALLEL_ACCEL_RES_1	Axis acceleration reserve of parallel axis for compensatory motion (1st TRAANG)
24771	TRAANG_PARALLEL_ACCEL_RES_2	Axis acceleration reserve of parallel axis for compensatory motion (2nd TRAANG)

2.9.1.4 Chained transformations

Channelspecific machine data

Number	Identifier: $MC_	Description
24995	TRACON_CHAIN_1	Transformation chain of the first chained transformation
24996	TRACON_CHAIN_2	Transformation chain of the second chained transformation
24997	TRACON_CHAIN_3	Transformation chain of the third chained transformation
24998	TRACON_CHAIN_4	Transformation chain of the fourth chained transformation

2.9.1.5 Non transformation-specific machine data

Channelspecific machine data

Number	Identifier: $MC_	Description
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21090	MAX_LEAD_ANGLE	Maximum permissible lead angle for orientation programming
21092	MAX_TILT_ANGLE	Maximum permissible side angle for orientation programming
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming

F2: Multi-axis transformations 3

3.1 Brief description

3.1.1 General specifications

Transformation data set

A transformation is defined with a specific number of machine data. All machine data of a transformation data set are marked using the same prefix <x>. With <x> = 1, 2, 3, ... maximum number of possible transformations in the channel.

Example based on the transformation type of a transformation: MD24100 $MC_TRAFO_TYPE_<x>

- Transformation type of the first transformation: MD24100 $MC_TRAFO_TYPE_**1**
- Transformation type of the second transformation: MD24100 $MC_TRAFO_TYPE_**2**
- ...

Axis names

While a transformation is active, the channel, geometry and machine axis names within a channel must be different.

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)
- MD20080 $MC_AXCONF_CHANAX_NAME_TAB (channel axis name)
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB (geometry axis name)

3.1.2 5-axis Transformation

Function

The "5-Axis Transformation" machining package is designed for machining sculptured surfaces that have two rotary axes in addition to the three linear axes X, Y, and Z. This package thus allows an axially symmetrical tool (milling cutter, laser beam) to be oriented in any desired relation to the workpiece in the machining space.

The path and path velocity are programmed in the same way as for 3-axis tools. The tool orientation is programmed additionally in the traversing blocks.

The real-time transformation performs the calculation of the resulting motion of all 5 axes. The generated machining programs are therefore not machine specific. Kinematic-specific post-processors are not used for the 5-axis machining operation.

A selection of various transformations is available for adapting the control to various machine kinematics. Part program commands can be issued in operation to switch over between two transformations parameterized during start-up.

This package therefore covers the three possible basic machine configurations which differ in terms of tool and workpiece orientation:

- Orientation of tool with two-axis swivel head (machine type 1)
- Orientation of workpiece with two-axis rotary table (machine type 2)
- Orientation of workpiece and tool with single-axis rotary table and swivel head (machine type 3)

The calculation also includes tool length compensation.

Since the orientation in relation to the workpiece surface is stored in a separate FRAME, a tool retraction operation with vertical orientation to the workpiece is also possible.

Tool orientation

Tool orientation can be specified in two ways:

- **Machine-related orientation**
  The machine-related orientation is dependent on the machine kinematics.
- **Workpiece-related orientation**
  The workpiece-related orientation is not dependent on the machine kinematics.
  It is programmed by means of:
  - Euler angles
  - RPY angles
  - Vector components

  The direction of the tool is described in the workpiece coordinate system with the part orientation. It is possible to program a specific component of the tool in its orientation to the workpiece. In most cases, this will be a longitudinal axis of the tool with the tool tip (Tool Center Point, TCP), which is also referred to as TCP-programming.

System variables for orientation

Part programs and synchronized actions can access the system variables that provide information on the following, in read only mode:

- End orientation of block (run-in value)
- Setpoint orientation
- Actual value orientation
- Switching between setpoint and actual value orientation
- Status for variables of actual value orientation

Special cases of 5-Axis transformation

The following transformations are to be entered as special cases of the general 5-Axis transformation:

- **3-axis and 4-axis transformation**
  There are 2 or 3 linear axes and a rotary axis.
- **Swivelling linear axis**
  One of the rotary axis rotates the 3rd linear axis.
- **Universal milling head**
  The two rotary axes are positioned at a projectable angle in relation to one another.

Knowledge of the general 5-axis transformation is a prerequisite for all of these transformations.

3.1.3 3-axis and 4-axis transformation

Function

The 3- and 4-Axis transformations are distinguished by the following characteristics:

Transformation	Features
3-axis Transformation	2 linear axes 1 rotary axis
4-Axis transformation	3 linear axes 1 rotary axis

Both types of transformation belong to the orientation transformations. Orientation of the tool must be programmed explicitly. The orientation of the tool is executed in a plane perpendicular to the rotary axis.

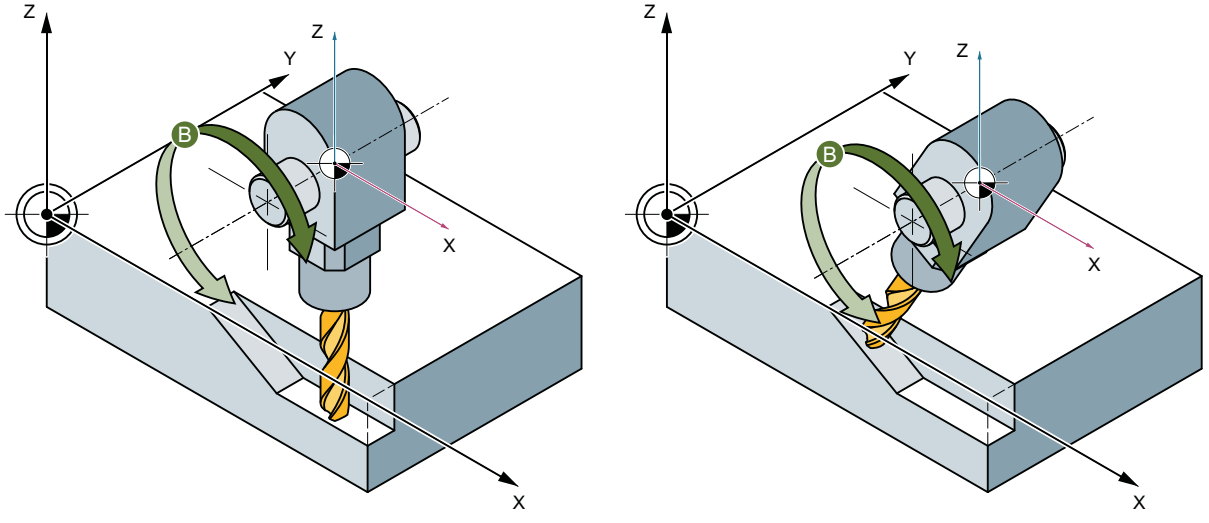


Figure 3-1 Schematic diagram of 3-axis transformation

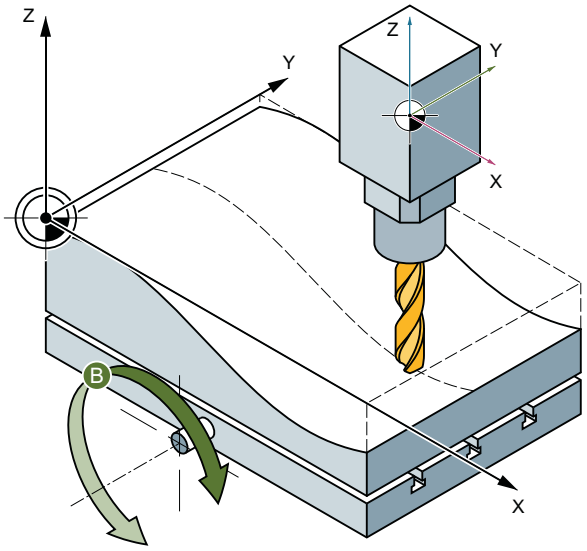


Figure 3-2 Schematic diagram of a 4-axis transformation with moveable workpiece

3.1.4 Orientation transformation with a swiveling linear axis.

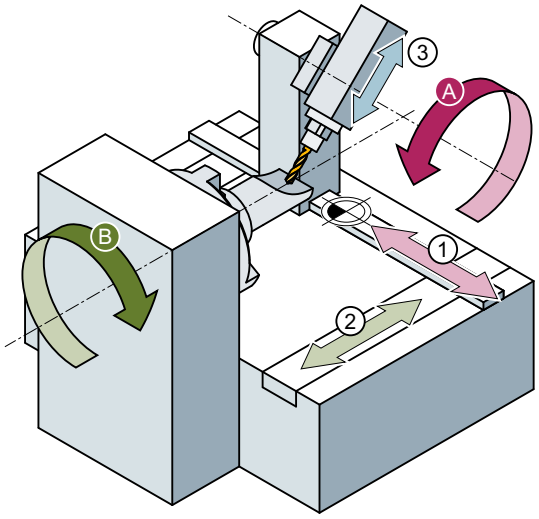
Function

The orientation transformation with swiveling linear axis is similar to the 5-axis transformation of Machine Type 3, though the 3rd linear axis is not always perpendicular to the plane defined by the other two linear axes.

Features of kinematics

- Kinematics with three linear axes and two orthogonal rotary axes.
- Rotary axes are parallel to two of the three linear axes.
- The first rotary axis is moved by two Cartesian linear axes. It rotates the third linear axis, which moves the tool. The tool is aligned parallel to the third linear axis.
- The second rotary axis rotates the workpiece.
- The kinematics comprise a moved workpiece and a moved tool.

The following figure shows the interrelations for one of the possible axis sequences, for which transformation is possible.



① Linear axis 1 (X)
② Linear axis 2 (Y)
③ Swiveling linear axis 3 (Z)

Figure 3-3 Schematic diagram of a machine with swiveling linear axis

3.1.5 Universal milling head

Function

A machine tool with a universal milling head has got at least 5 axes:

- 3 linear axes
  - for linear movement [X, Y, Z]
  - move the machining point to any random position in the working area
- 2 rotary swivelling axes
  - are arranged under a configurable angle (mostly 45 Degree)
  - enable the tool to define orientations in space
    (are limited to a hemisphere in a 45 degree arrangement)

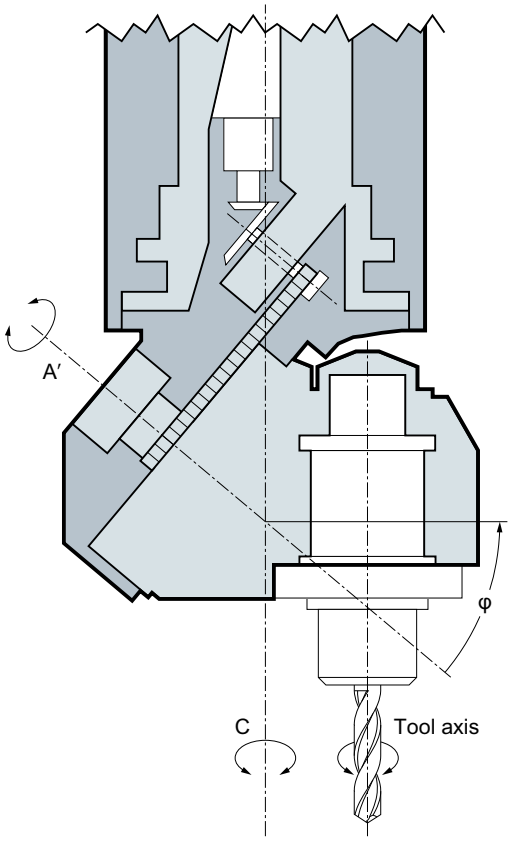


Figure 3-4 Schematic diagram of a machine tool with universal milling head

3.1.6 Orientation axes

Model for describing change in orientation

There is no such simple correlation between axis motion and change in orientation in case of robots, hexapodes or nutator kinamatics, as in the case of conventional 5-axes machines.

For this reason, the change in orientation is defined by a model that is created independently of the actual machine. This model defines three virtual orientation axes which can be visualized as rotations about the coordinate axes of a rectangular coordinate system.

For the purpose of 6-axis transformation, a third degree of freedom for orientation, describing the rotation of the tool about itself, has been introduced.

Real-time transformation

The Cartesian coordinates are converted from basic to machine coordinate system by means of a real-time transformation process.

These Cartesian coordinates comprise:

- Geometry axes
  Geometry axes describe the machining point.
- Orientation axes
  Orientation axes describe the orientation of a tool in space.

Tool orientation

You can define the orientation of the tool in space as follows using linear interpolation, large circle interpolation and by means of orientation vectors:

- Direct programming of rotary axis positions A, B, C
  5-axis transformation by programming:
  - The Euler- or RPY angle in degrees through `A2,B2,C2`
    or
  - The direction vector over `A3,B3,C3`
- Programming using lead angle `LEAD` and tilt angle `TILT`

3.1.7 Cartesian manual travel

Function

The "Cartesian manual travel" function allows the reference system for JOG motion to be selected for one of the following coordinate systems (for both translation and orientation as):

- Basic coordinate system (BCS)
- Workpiece coordinate system (WCS)
- Tool coordinate system (TCS)

3.1.8 Cartesian PTP travel

Function

The "Cartesian PTP Travel" [PTP = Point-to-point movement (**P**oint **t**o **P**oint)] function can be used to program a position in a cartesian coordinate system (workpiece coordinate system). The machine however moves in its machine coordinates.

The function can be used, for example, to traverse a singularity. Cartesian positions, supplied by a CAD system, need not been converted to machine axis values.

It must also be noted that axes take longer to traverse in the Cartesian coordinate system with active transformation and programmed feedrate than when they are traversed directly.

3.1.9 Generic 5-axis transformation

Function

The generic 5-axis transformation function differs from earlier 5-axis transformation versions insofar as it is no longer restricted with respect to the directions of rotary axes.

The basic orientation of the tool is no longer predefined in machine data as was the case in earlier versions of orientation transformations, but can now be programmed freely.

3.1.10 Online tool length offset

Function

The system variable $AA_TOFF[ ] can be used to overlay the effective tool lengths in 3-D in runtime. For an active orientation transformation (`TRAORI`) or for an active tool carrier that can be oriented, these offsets are effective in the particular tool axes.

If the tool orientation changes, the tool length offsets that apply are rotated so that the pivot point for the orientation movement always refers to the corrected tool tip.

3.1.11 Activation via parts/program/softkey

The machine data relevant to the kinematic transformation has thus far been activated mostly through POWER ON.

Transformations MDs can also be activated via the part program / softkey and it is not necessary to boot the control system.

References:
Function Manual, Extended Functions; Kinematic Transformations (M1),
Section: Cartesian PTP travel

See also

Cartesian PTP travel (Page 97)

3.1.12 Orientation compression

During the execution of NC programs containing blocks with relatively short traverse paths, the interpolator clock cycle can lead to a reduction in tool path velocity and a corresponding increase in machining time.

COMPON, COMPCURV, COMPCAD

You can run NC programs with short traverse paths without reducing the tool path velocity by activating "compressors" COMPON, COMPCURV or COMPCAD. The compressor also smoothes the programmed movements and consequently tool path velocity.

Programming of direction vectors

Programming of tool orientation that is independent of the kinematics, can be achieved through programming of direction vectors. NC programs with such direction vectors can be executed with compressors COMPON, COMPCURV and COMPCAD.

3.2 5-axis transformation

3.2.1 Kinematic transformation

Task of orientation transformation

The task of orientation transformation is to compensate movements of the tool nose, which result from changes in orientation, by means of appropriate compensating movements of the geometry axes. The orientation movement is therefore decoupled from the movement on the workpiece contour. Various machine kinematics each require their own orientation transformation.

Fields of application

The "5-axis transformation" machining package is provided for machine tools, which have two additional rotary axes (rotation about the linear axes) in addition to three linear axes X, Y and Z: This package thus allows an axially symmetrical tool (milling cutter, laser beam) to be oriented in any desired relation to the workpiece in every point of the machining space.

The workpiece is always programmed in the rectangular workpiece coordinate system; any programmed or set frames rotate and shift this system in relation to the basic system. The kinematic transformation then converts this information into motion commands of the real machine axes.

The kinematic transformation requires information about the design (kinematics) of the machine, which are stored in machine data.

The kinematic transformation does not act on positioning axes.

3.2.2 Machine types for 5-axis transformation

Depending on the orientation capability of tool and workpiece, machines are classified according to machine types 1, 2 and 3.

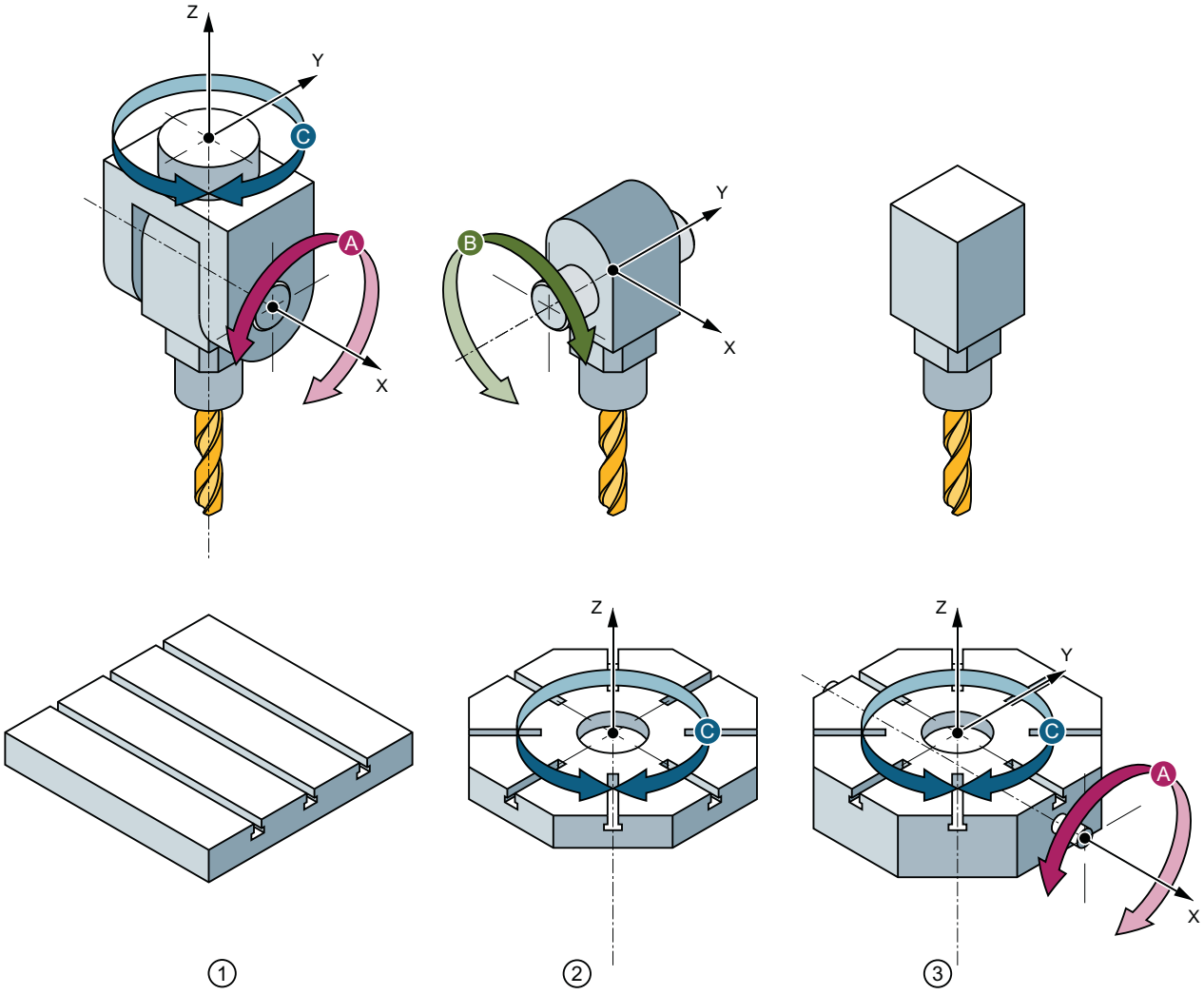


Figure 3-5 Machine types 5-axis transformation

Machine types

Number	Swivel head	workpiece table	Machine type	Tool
①	Rotary axes A and C	fixed	Two-axis swivel head with the axis arrangement CA	Orientable
②	Rotary axis B	Rotary axis C	Single-axis swivel head and single-axis rotary table with the axis sequence BC	Orientable tool and workpiece
③	fixed	Rotary axes A and C	Two-axis rotary table with the axis arrangement AC	Workpiece that can be oriented

Common properties of machine types 1, 2 and 3

- The three linear axes form a right-handed, Cartesian coordinate system.
- The rotary axes are orientated parallel to the linear axes
- The orientation of the rotary axes are vertical, on one another.
- The tool initial state is in the negative Z direction.

Other properties of machine types 1 and 2

- Rotary axis 1 is the 4th machine axis of the transformation
- Rotary axis 1 changes the orientation of rotary axis 2
- Rotary axis 2 is the 5th machine axis of the transformation
- Rotary axis 2 does not change the orientation of rotary axis 1

Other properties of machine type 3

- Rotary axis 1 is the 4th machine axis of the transformation
- Rotary axis 1 rotates the tool
- Rotary axis 2 is the 5th machine axis of the transformation
- Rotary axis 2 rotates the workpiece

Note

Transformations not in accordance with the conditions mentioned here are described in their own sections

3.2.3 Configuration of a machine for 5-axis transformation

Maximum number of 5-axis transformations per channel

Currently, no more than four 5-axis transformations can be parameterized per channel

Transformation type

The transformation type that is dependent on the machine type is set in machine data:

MD24100 $MC_TRAFO_TYPE_<x> = <transformation type>

where x = 1, 2, ... maximum number of transformations

Axis sequence of the machine	Machine type 1 (tool that can be swiveled/rotated)	Machine type 2 (workpiece that can be swiveled/rotated)	Machine type 3 (tool/workpiece that can be swiveled/rotated)
	Transformation type	Transformation type	Transformation type
AB	16	32	48
AC	--- [1]	33	49
BA	18	34	50
BC	--- [1]	35	51
CA	20	--- [1]	--- [1]
CB	21	--- [1]	--- [1]

[1] Not a practical combination, as a rotation of the C axis would only correspond to a rotation of the tool around its own longitudinal axis (symmetrical axis).

Identifying the axis sequence

The axis sequence is identified in the following way:

- AB means: A is 4th axis, B is 5th axis of the transformation
- For machine type 3, the swivel axis of the tool is the 4th axis of the transformation and the rotary axis of the workpiece is the 5th axis of the transformation.

Axis assignment

In the machine data, the axis assignment at the input of the 5-axis transformation defines which axis will be mapped by the transformation to which channel axis:

MD24110 $MC_TRAFO_AXES_IN_<x> (axis assignment for transformation x)

where x = 1, 2, ... maximum number of transformations

Geometry information

Information concerning machine geometry is required so that the 5-axis transformation can calculate axis values: This information is stored in the machine data (in this case, for the first transformation in the channel):

- $MC_TRAFO5_PART_OFFSET_<x> (workpiece-oriented offset)
  where x = 1, 2, ... maximum number of transformations in the channel
  - Machine type 1:
    Vector from machine reference point to table zero point (generally, the zero vector)
  - Machine type 2:
    Vector components from the last rotating joint of the table to the zero point of the table ①.

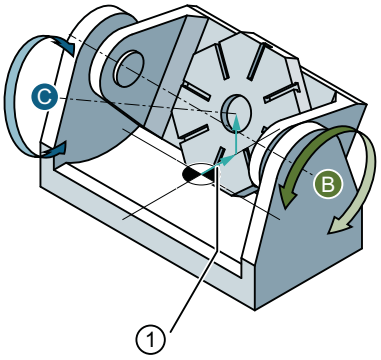
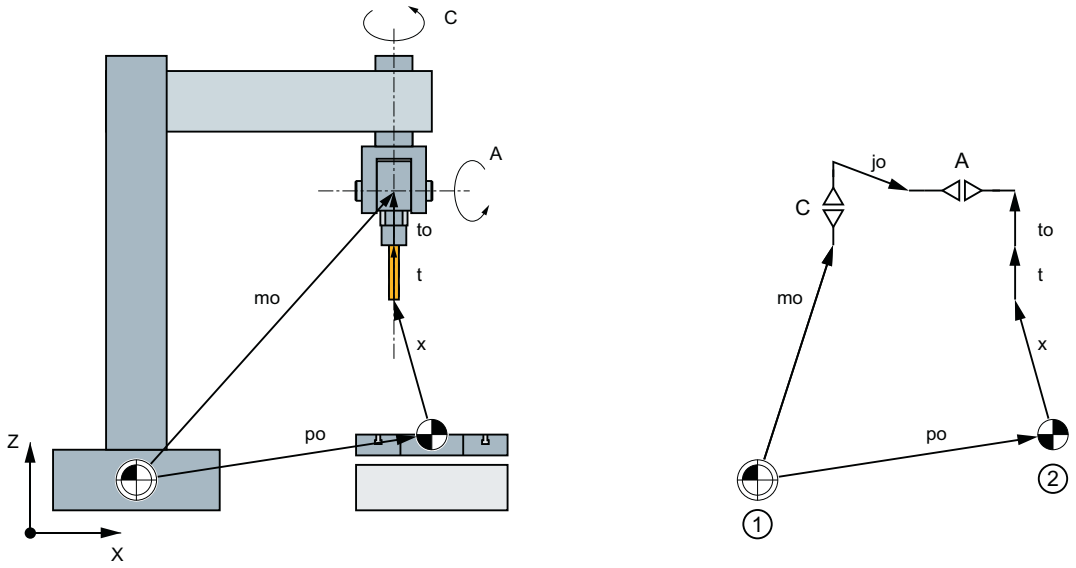


Figure 3-6 Machine data $MC_TRAFO5_PART_OFFSET_1 for machine type 2

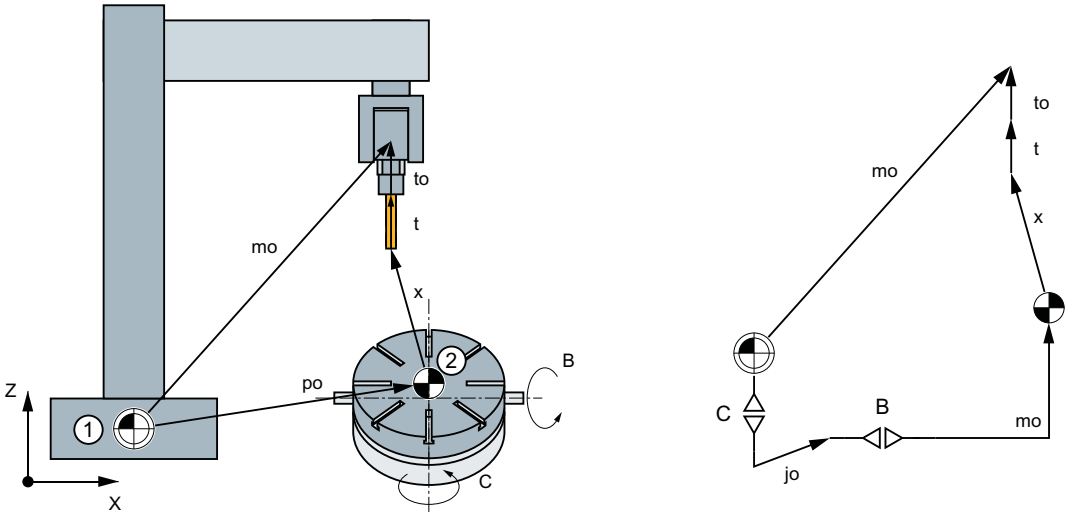
  - Machine type 3 (single-axis swivel head and single-axis rotary table)
    Vector from joint of rotary table to zero point of table
- $MC_TRAFO5_JOINT_OFFSET_<x> (vector of kinematic offset)
  where x = 1, 2, ... maximum number of transformations in the channel
  - Machine type 1 and 2:
    Vector from the first to the second swivel joint
  - Machine type 3:
    Vector from machine zero point to the swivel joint of the table
- $MC_TRAFO5_ROT_AX_OFFSET_<x> (position offset of rotary axes 1 / 2 / 3)
  where x = 1, 2, ... maximum number of transformations in the channel

Examples



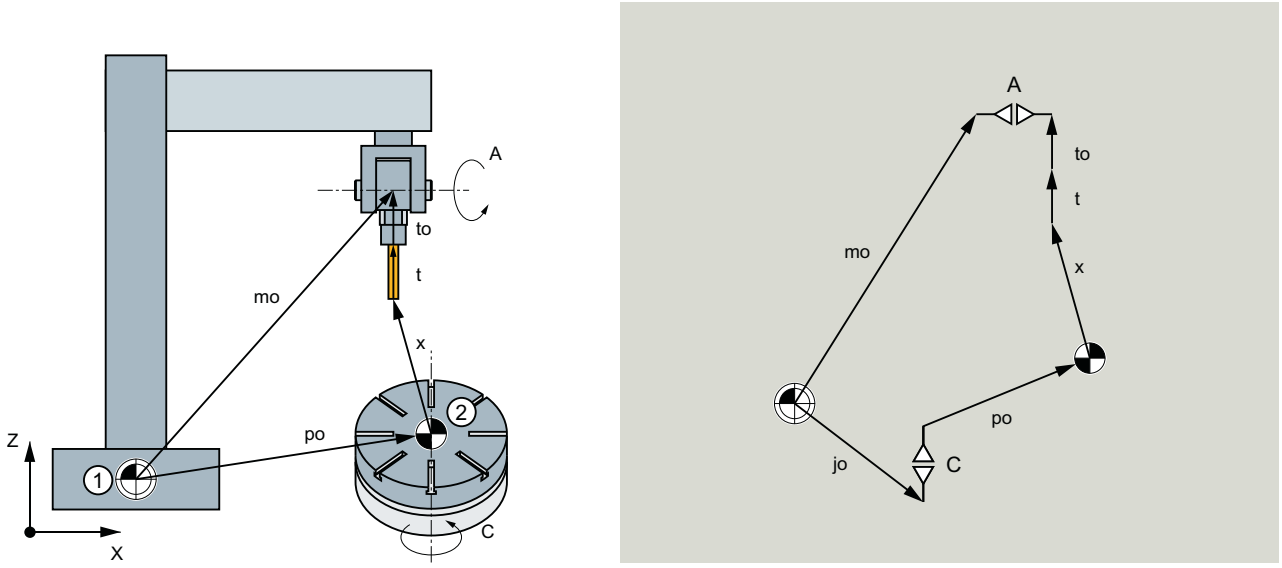
①	Machine zero
②	Zero point tool table
mo	Position vector in MCS
po	$MC_TRAFO5_PART_OFFSET_<x>[0 ..2]
x	Vector of programmed position in BCS
t	Tool correction vector
to	$MC_TRAFO5_BASE_TOOL_<x>[0 .. 2]
jo	MD24560 $MC_TRAFO5_JOINT_OFFSET_<x>[0 .. 2]

Figure 3-7 Schematic diagram of CA kinematics, moved tool



① Machine zero
② Zero point tool table
mo Position vector in MCS
po $MC_TRAFO5_PART_OFFSET_<x>[0 ..2]
x Vector of programmed position in BCS
t Tool correction vector
to $MC_TRAFO5_BASE_TOOL_<x>[0 .. 2]
jo $MC_TRAFO5_JOINT_OFFSET_<x>[0 .. 2]

Figure 3-8 Schematic diagram of CB kinematics, moved workpiece



① Machine zero
② Zero point tool table
mo Position vector in MCS
po $MC_TRAFO5_PART_OFFSET_<x>[0 ..2]
x Vector of programmed position in BCS
t Tool correction vector
to $MC_TRAFO5_BASE_TOOL_<x>[0 .. 2]
jo $MC_TRAFO5_JOINT_OFFSET_<x>[0 .. 2]

Figure 3-9 Schematic diagram of AC kinematics, moved tool, moved workpiece, moved tool

Sign handling for rotary axes

The sign handling of rotary axes involved in 5-axis transformation is set in the channel using machine data:

$MC_TRAFO5_ROT_SIGN_IS_PLUS_<x>[<n>] = <value>

where x = 1, 2, ... maximum number of transformations in the channel

<n>	Meaning
0	1st rotary axis
1	2nd rotary axis
2	3rd rotary axis

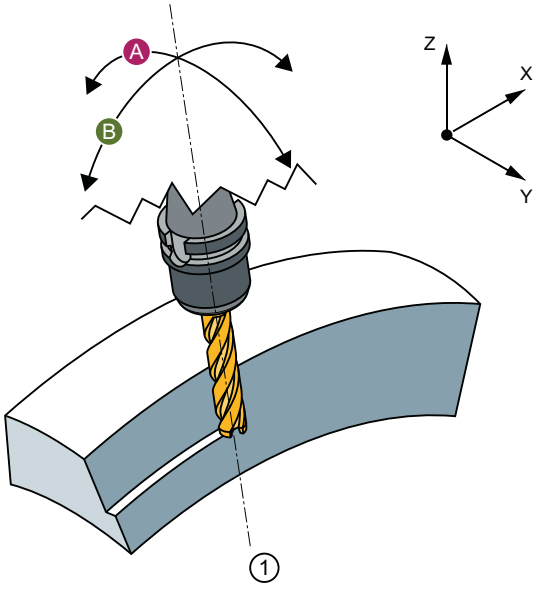
<Value>	Meaning
TRUE	The sign is not inverted. The traversing direction is as defined in MD32100 $MA_AX_MOTION_DIR.
FALSE	The sign is inverted.

The machine data is not used to define that the direction of rotation of the rotary axis involved is changed. Instead, it is specified as to whether the rotary axis, for traversing motion, rotates in the positive traversing direction in the mathematical positive (counterclockwise) or negative (clockwise) direction. The consequence when changing this machine data is therefore not a change in the direction of rotation, but a change in the compensating motion in the linear axes.

However, if a direction vector, and therefore implicitly compensation motion is specified, then the direction of rotation of the rotary axis involved changes.

This means that at a real machine, the machine data only has to be set to FALSE if, for traversing motion in the positive traversing direction, the rotary axis rotates in the mathematical positive direction (counterclockwise).

3.2.4 Tool orientation



① Tool axis

Figure 3-10 Processing of workpieces with 5-axis transformation

Programming

The orientation of the tool can be programmed in a block directly by specifying the rotary axes or indirectly by specifying the Euler angle, RPY angle and direction vector. The following options are available:

- directly as rotary axes A, B, C
- indirectly for 5-axis transformation:
  via Euler or RPY angles in degrees via A2, B2, C2
- Indirectly for 5-axis transformation via direction vector A3, B3, C3

The identifiers for Euler angles and direction vectors can be set in machine data:

Euler angles via:

MD10620 $MN_EULER_ANGLE_NAME_TAB (name of Euler angles)

Direction vector via:

MD10640 $MN_DIR_VECTOR_NAME_TAB (name of direction vectors)

The tool orientation can be located in any block. Above all, it can be programmed alone in a block, resulting in a change of orientation in relation to the tool tip which is fixed in its relationship to the workpiece.

Euler or RPY

The following machine data can be used to switch between Euler and RPY angles:

MD21100 $MC_ORIENTATION_IS_EULER (angle definition for orientation programming)

Orientation reference

A tool orientation at the start of a block can be transferred to the block end in two different ways:

- in the workpiece coordinate system with command ORIWKS
- in the machine coordinate system with command ORIMKS

ORIWKS command

The tool orientation is programmed in the workpiece coordinate system (WCS) and is thus not dependent on the machine kinematics.

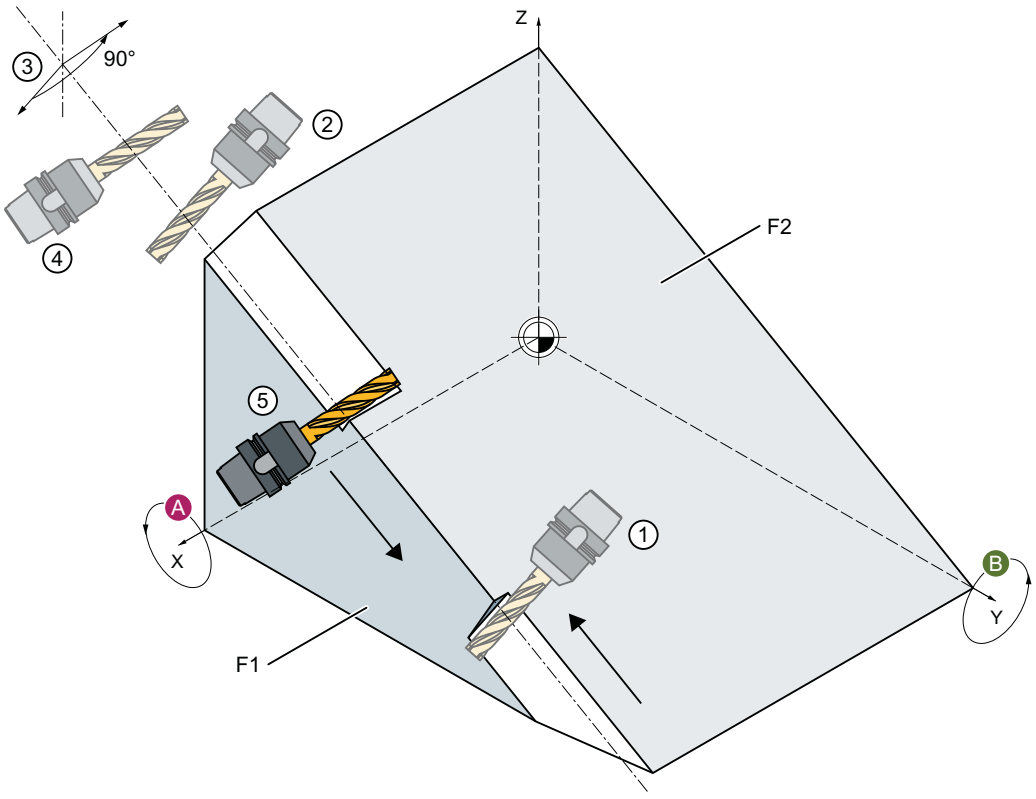
In the case of a change in orientation with the tool tip at a fixed point in space, the tool moves along a large arc on the plane stretching from the start vector to the end vector.

ORIMKS command

The tool orientation is programmed in the machine coordinate system and is thus dependent on the machine kinematics.

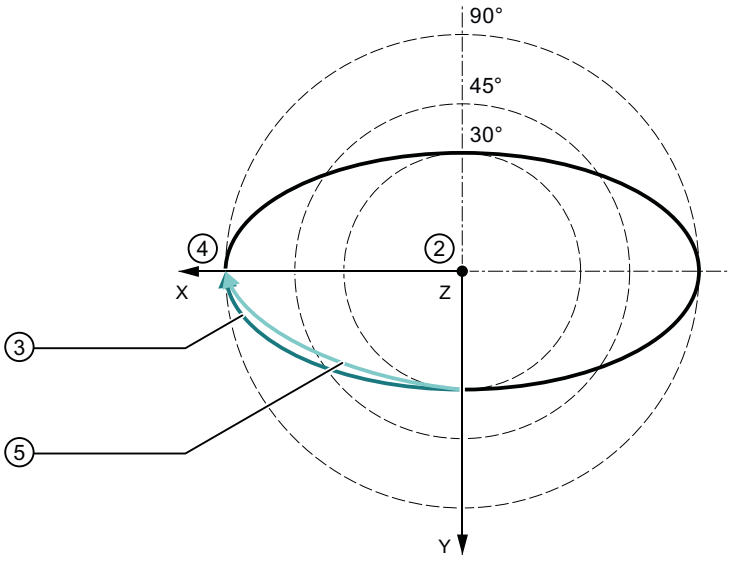
In the case of a change in orientation of a tool tip at a fixed point in space, linear interpolation takes place between the rotary axis positions.

The orientation is selected via NC language commands ORIWKS and ORIMKS.



① Process the surface F1 along the edge of F2
② Retraction
③ Orientation change B: +90° A: -30°
④ Approach of area F2
⑤ Process the surface F2 along the edge of F1

Figure 3-11 Change in cutter orientation while processing inclined edges



② Retraction
③ with ORIWKS (large circle)
④ Approach of area F2
⑤ With ORIMKS (linear interpolation between rotary axes)

Figure 3-12 Change in orientation while processing inclined edges

ORIMKS constitutes the basic setting

The basic setting can be changed via the following machine data:

MD20150 MC_GCODE_RESET_VALUES (RESET position of G groups)

MD20150 $MC_GCODE_RESET_VALUES [24] = 1 ⇒ ORIWKS is basic setting

MD20150 $MC_GCODE_RESET_VALUES [24] = 2 ⇒ ORIWKS is basic setting

Illegal tool orientation

If tool position is programmed in relation to the following functions, alarm 12130 "Illegal tool orientation" is output when Euler angles and direction vectors are selected and the NC program then stops (this alarm can also occur in connection with G331, G332 and G63).

- G04: Dwell time
- G33: Thread cutting with constant lead
- G74: Approaching a reference point
- G75: Approaching a fixed point
- REPOSL: Repositioning to the contour
- REPOSQ: Repositioning to the contour
- REPOSH: Repositioning to the contour

To remedy this situation, tool orientation can be programmed with axis end values.

Alarm 17630 or 17620 is output for G74 and G75 if a transformation is active and the axes to be traversed are involved in the transformation. This applies irrespective of orientation programming.

If the start and end vectors are inverse parallel when ORIWKS is active, then no unique plane is defined for the orientation programming, resulting in the output of alarm 14120.

If a transformation switch (switch On, switch Off or change transformation) is undertaken, alarm 14400 will be generated.

In the reverse situation, i.e. a tool radius offset is selected or deselected when a transformation is active, no alarm message is output.

Multiple input of tool orientation

According to DIN 66025, only one tool orientation may be programmed in a block, e.g. with direction vectors:

```
N50           A3=1 B3=1 C3=1
```

If tool orientation is entered multiply, i.e. with direction vectors and with Euler angles, error message 12240 "Channel X block Y tool orientation xx defined more than once" is displayed and the NC part program stops.

```
N60           A3=1 B3=1 C3=1                A2=0 B2=1 C2=3
```

Tool orientation using orientation vectors

Polynomials can also be programmed for the modification of the orientation vector.

This method produces an extremely smooth change in speed and acceleration at the block changes for rotary axes when the tool orientation has to be programmed over several blocks.

The interpolation of orientation vectors can be programmed with polynomials up to the 5th degree. Polynomial interpolation of orientation vectors is described in the "Polynomial Interpolation of Orientation Vectors" section.

Note

Further explanations of tool orientation using orientation vectors and their handling in machines are given in:

Further information
Function Manual Tools, tool offset, orientable tool carriers

3.2.5 Singular positions and handling

Extreme velocity increase

If the path runs in close vicinity to a pole (singularity), one or several axes may traverse at a very high velocity.

Alarm 10910 "Irregular velocity run in a path axis" is then triggered. The programmed velocity is then reduced to a value, which does not exceed the maximum axis velocity.

Behavior at pole

Unwanted behavior of fast compensating movements can be controlled by making an appropriate selection of the following machine data (see following Figure):

- MD24530 $MC_TRAFO5_NON_POLE_LIMIT_1 (definition of pole area for 5-axis transformation 1)
- MD24630 $MC_ TRAFO5_NON_POLE_LIMIT_2 (definition of pole area for 5-axis transformation 2)
- MD24540 $MC_TRAFO5_POLE_LIMIT_1 (closing angle tolerance for interpolation by pole for 5-axis transformation)
- MD24640 $MC_TRAFO5_POLE_LIMIT_2 (closing angle tolerance for interpolation by pole for 5-axis transformation)

Note

Singularities are dealt with differently in SW 5.2 and higher: Only one relevant machine data item exists, $MC_TRAFO5_POLE_LIMIT (see Section "Singularities of orientation (Page 189)" or "Programming Manual, Production Planning).

Definition of the pole range for 5-axis transformation

This machine data identifies a limit angle of the 5th axis of the first MD24530 $MC_TRAFO5_NON_POLE_LIMIT_1 or the second MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2 5-axis transformation with the following properties:

If the path runs past the pole at an angle lower than the value set here, it crosses through the pole.

With the 5-axis transformation, a coordinate system consisting of circles of longitude and latitude is spanned over a spherical surface by the two orientation axes of the tool.

If, as a result of orientation programming (i.e. the orientation vector is positioned on one plane), the path passes so close to the pole that the angle is less than the value defined in this machine data, then a deviation from the specified interpolation is made so that the interpolation passes through the pole.

End angle tolerance for interpolation by pole for 5-axis transformation

This machine data identifies a limit angle for the 5th axis of the first MD24540 $MC_TRAFO5_NON_POLE_LIMIT_1 or the second MD24640 $MC_TRAFO5_NON_POLE_LIMIT_2 5-axis transformation with the following properties:

With interpolation through the pole point, only the fifth axis moves; the fourth axis remains in its start position. If a movement is programmed which does not pass exactly through the pole point, but is to pass within the tolerance defined by the following machine data in the vicinity of

the pole, a deviation is made from the specified path because the interpolation runs exactly through the pole point.

- MD24530 $MC_TRAFO5_NON_POLE_LIMIT_1
- MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2

As a result, the position at the end point of the fourth axis (pole axis) deviates from the programmed value.

This machine data specifies the angle by which the pole axis may deviate from the programmed value with a 5-axis transformation if a switchover is made from the programmed interpolation to interpolation through the pole point. In the case of a greater deviation, an error message is output and the interpolation is not executed.

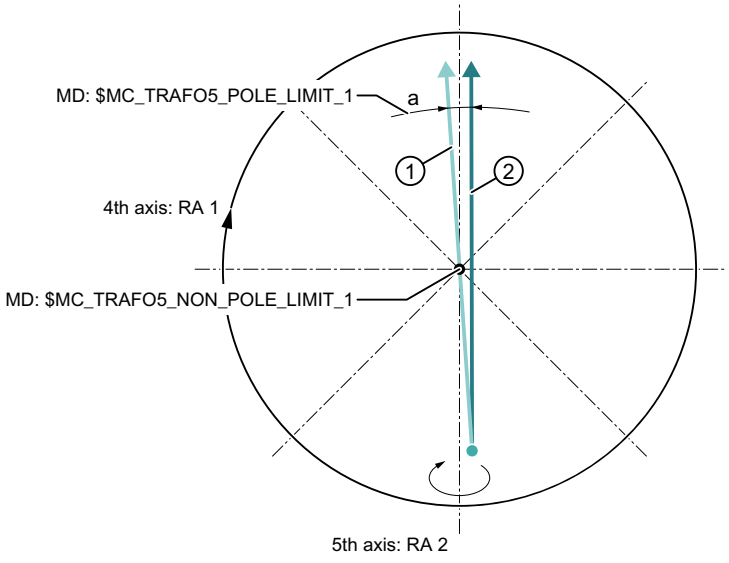


Figure 3-13 5-axis transformation; orientation path in pole vicinity. Example for machine type 1: 2-axis swivel head with rotary axis RA 1 (4th axis of transformation) and rotary axis RA 2 (5th axis of transformation)

Behavior during large circle interpolation at pole position

The following machine data can be used to set the response for large circle interpolation in pole position as follows:

MD21108 $MC_POLE_ORI_MODE

Does not define the treatment of changes in orientation during large circle interpolation unless the starting orientation is equal to the pole orientation or approximates to it and the end orientation of the block is outside the tolerance circle defined in the following machine data.

- MD24530 $MC_TRAFO5_NON_POLE_LIMIT_1
- MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2

The position of the polar axis is arbitrary in the polar position. For the large circle interpolation, however, a specified orientation is required for this axis.

The following machine data is coded decimally.

MD21108 $MC_POLE_ORI_MODE

The **units** define the behavior if start orientation coincides with pole position and the **decade** the behavior if start orientation is within the tolerances defined by the following machine data:

- MD24530 $MC_TRAFO5_NON_POLE_LIMIT_1
- MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2

All setting values are described in "Channel-specific Machine Data".

3.3 3-axis and 4-axis transformations

3 and 4-axis transformations are special types of 5-axis transformations, where the tool can only be orientated in the plane, perpendicular to the rotary axis. They support machine type 1 with movable tool - and type 2 with movable workpiece.

Versions of 3-axis and 4-axis transformation

Machine type	swiveling/ rotary	Rotary axis parallel to	orientation plane	Transformation type	Tool orientation in zero position
1	Tool	X	Y - Z	16	Z
		Y	X - Z	18	
		Z	X - Y	20	Y
		Z	X - Y	21	X
		any	Any	24 [1)]	Any
2	workpiece	X	Y - Z	32, 33	Z
		Y	X - Z	34, 35	
		Any	Any	40 [1)]	Any
1) The rotary axis and the orientation plane can be set, so that the orientation does not change in a plane, but on a tapered peripheral surface.					

Tool orientation in zero position

Tool orientation at the zero position is the position of the tool for active machining plane G17 **AND** position of the rotary axis 0 degrees

Axis assignments

The three linear axes included in the transformation are assigned to any channel axes via machine data $MC_TRAFO_GEOAX_ASSIGN_TAB_n[0..2] and $MC_TRAFO_AXES_IN_n[0..2]. The following must apply for the assignment of channel axes to geometry axes for the transformation:

$MC_TRAFO_GEOAX_ASSIGN_TAB_n[0] = $MC_TRAFO_AXES_IN_n[0]

$MC_TRAFO_GEOAX_ASSIGN_TAB_n[1] = $MC_TRAFO_AXES_IN_n[1]

$MC_TRAFO_GEOAX_ASSIGN_TAB_n[2] = $MC_TRAFO_AXES_IN_n[2]

The axes with corresponding index must be assigned to each other.

Parameter assignment procedure

- Enter the type of transformation according to the previous table as machine data: $MC_TRAFO_TYPE_n
- Assign channel axes to the geometry axes of the transformation.
- For a 3-axis transformation, set the values for the axis, which is not required:
  - $MC_TRAFO_GEOAX_ASSIGN_TAB_n[<geometry axis>] = 0
  - $MC_TRAFO_AXES_IN_n[<geometry axis>] = 0
    $MC_TRAFO_AXES_IN_n[4] = 0; → there is no 2nd rotary axis
- For a 4-axis transformation, set the following for the 3 linear axes
  - $MC_TRAFO_GEOAX_ASSIGN_TAB_n[<geometry axis>] = ...
  - $MC_TRAFO_AXES_IN_n[<geometry axis>] = ...
    $MC_TRAFO_AXES_IN_n[4] = 0; → there is no 2nd rotary axis

Complete examples of a 3-axis and 4-axis transformation can be found in "Example for 3- and 4-axis Transformation" section.

3.4 Transformation with swiveled linear axis

General information

The "transformation with swiveling linear axis forms" a transformation group of its own. It can be used when a kinematic as described in the Section "Orientation transformation with a swiveling linear axis. (Page 140)" is present:

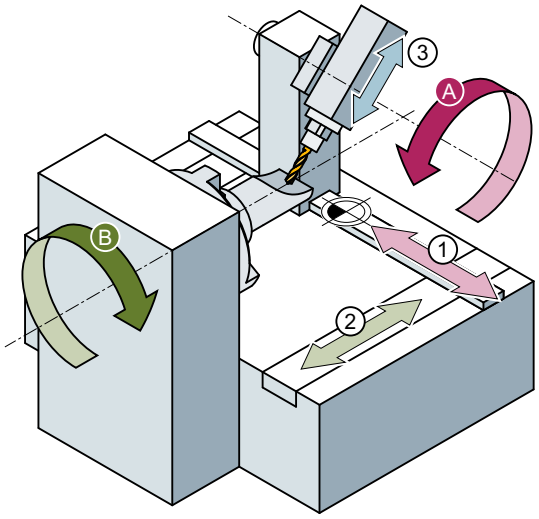
- Three Cartesian linear axes (X, Y, Z) and two orthogonal rotary axes (A, B).
- The rotary axes are parallel to two of the three linear axes.
- The first rotary axis (A) is moved by two Cartesian linear axes. It rotates the third linear axis (Z) that moves the **tool**.
- The tool is aligned parallel to the third linear axis (Z).
- The second rotary axis (B) rotates the **workpiece**.

Additional requirement:

- The first rotary axis (A) may only sweep a very small swivel range (swivel range << ± 90°).

Note

All the axis values used in the text relate to the designations of the example machine in the following figure "Machine with swiveling linear axis Z"



① Linear axis 1 (X)
② Linear axis 2 (Y)
③ Swiveling linear axis 3 (Z)

Figure 3-14 Example: Machine with swiveling linear axis Z

Pole

The transformation with swiveling linear axis has a pole for a tool orientation parallel to the second rotary axis (B). Singularity occurs in the pole position because the third linear axis (Z) is parallel to the plane of the first two linear axes (X, Y), thus excluding the possibility of compensating movements perpendicular to this plane.

Parameterization

Kinematic variants

The kinematic variant of the machine is set in the machine data:

MD24100, ... MD25190 $MC_TRAFO_TYP_n = <type>, with n = 1, 2, 3, ...

Kinematics			<type>
1st rotary axis	2nd rotary axis	swiveled linear axis	Bits 6 - 0
A	B	Z	10 00 000
A	C	Y	10 00 001
B	A	Z	10 00 010
B	C	X	10 00 011

Kinematics			<type>
1st rotary axis	2nd rotary axis	swiveled linear axis	Bits 6 - 0
C	A	Y	10 00 100
C	B	X	10 00 101

Machine kinematics

The machine kinematics is set for the 1st ($MC_TRAFO5 ... _1) and/or 2nd ($MC_TRAFO5 ... _2) 5-axis transformation in the channel set with the following machine data:

- Vector (**po**, see following figure) from the second rotary axis to workpiece table zero:
  - MD24500 $MC_TRAFO5_PART_OFFSET_1
  - MD24600 $MC_TRAFO5_PART_OFFSET_2
- Axis positions of the two rotary axes at the initial position of the machine:
  - MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1
  - MD24610 $MC_TRAFO5_ROT_AX_OFFSET_2
- Sign with which the rotary axis positions are included in the transformation:
  - MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1
  - MD24620 $MC_TRAFO5_ROT_SIGN_IS_PLUS_2
- Vector (**jo**) from machine zero to the second rotary axis:
  - MD24560 $MC_TRAFO5_JOINT_OFFSET_1
  - MD24660 $MC_TRAFO5_JOINT_OFFSET_2
- Vector (**to**) from the toolholder (flange) to the first rotary axis (measured at machine initial position):
  - MD24550 $MC_TRAFO5_BASE_TOOL_1
  - MD24650 $MC_TRAFO5_BASE_TOOL_2
- Vector (**ro**) from machine zero to the first rotary axis (measured at the machine initial position):
  - MD24562 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_1
  - MD24662 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_2

Determining the machine data values

As an aid for determining the values for the above-mentioned machine data, the following two sketches clarify the relationships between the vectors.

Note

Requirement

The machine has been traversed so that the tool holding flange aligns with the table zero (*). Is this is technically not possible, vector **to** must be corrected by the deviations.

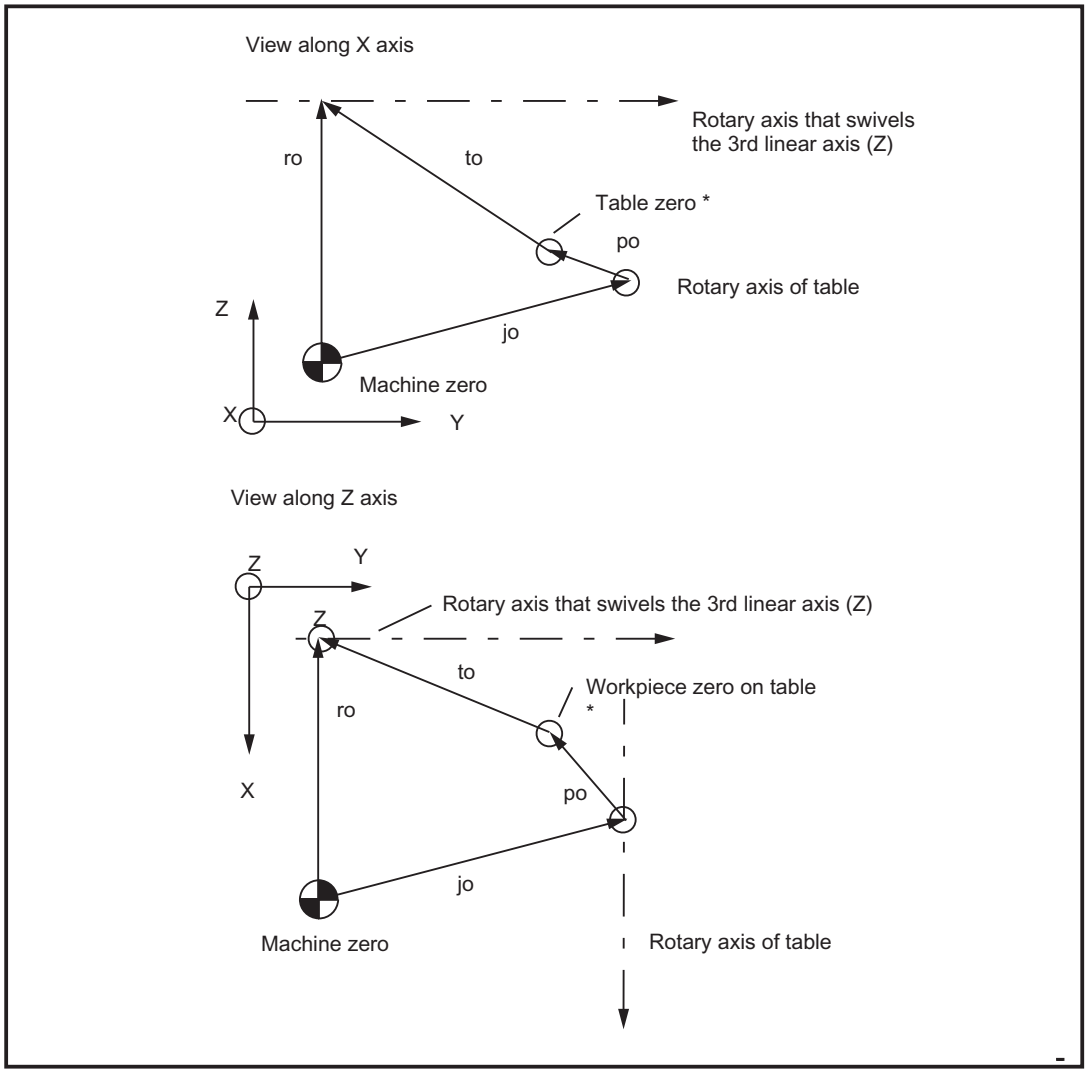
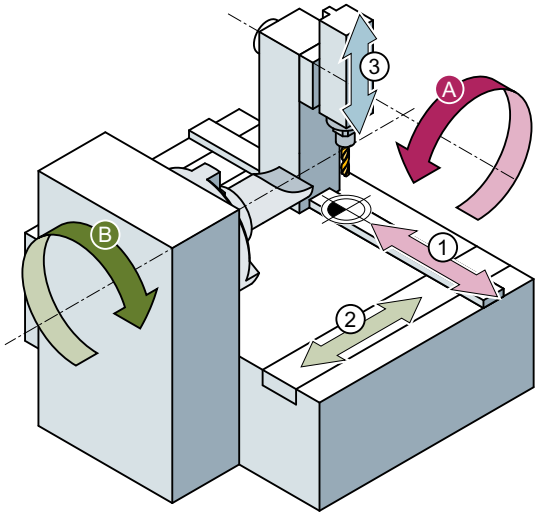


Figure 3-15 Projections of the vectors to be set in the machine data

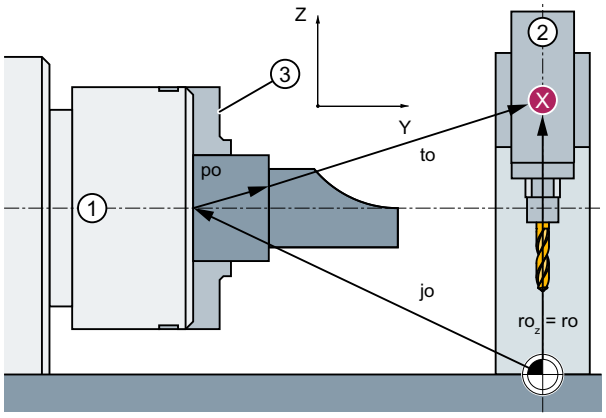
Note

A physically identical point on the 1st rotary axis (e.g. point of intersection between the tool axis and the 1st rotary axis) must be assumed for both views.



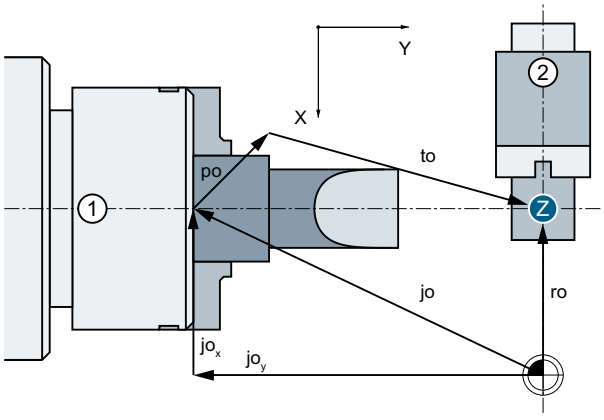
① Linear axis 1 (X)
② Linear axis 2 (Y)
③ Swiveling linear axis 3 (Z)

Figure 3-16 Machine in the zero position



① Stand
② Fixture
③ Axis of the tool

Figure 3-17 Front view: Vectors for machine in the zero position



① Stator with clamping device
② Swivel axis of the linear axis

Figure 3-18 Top view: Vectors for machine in the zero position

Determination of the machine data values

Perform the following operation:

1. Determine, as shown in the lower part for vector **jo** in the "Vectors for machine in zero position" figure, the X and Y components for all vectors.
2. As shown in the upper part for vector **ro** in the "Vectors for machine in zero position" figure, determine the Z component for all vectors.
3. Enter the X, Y and Z components of the vectors (**po**, **jo**, **to**, **ro**) in the relevant machine data.

The procedure can be used all settable kinematic variants.

Note

For the appropriate machine geometry or position of the machine zero, both individual components as well as complete vectors can become zero.

Programming

The switch on/off of the transformation in the part program or synchronized action is described in Section "Programming of the 3- to 5-axis transformation (Page 170)".

Tool orientation

For a transformation with swiveling linear axis, the same statements as for the 5-axis transformation with regard to the tool orientation apply similarly (see Section "Tool orientation (Page 153)").

3.5 Cardan milling head

3.5.1 Fundamentals of cardan milling head

Note

The following description of the cardan milling head transformation has been formulated on the assumption that the reader has already read and understood the general 5-axis transformation described in Section "5-axis transformation (Page 145)". Please note that where no specific statements relating to the cardan milling head are made in the following section, the statements relating to general 5-axis transformation apply.

Applications

A cardan milling head is used for machining contours of sculptured parts at high feedrates. An excellent degree of machining accuracy is achieved thanks to the rigidity of the head.

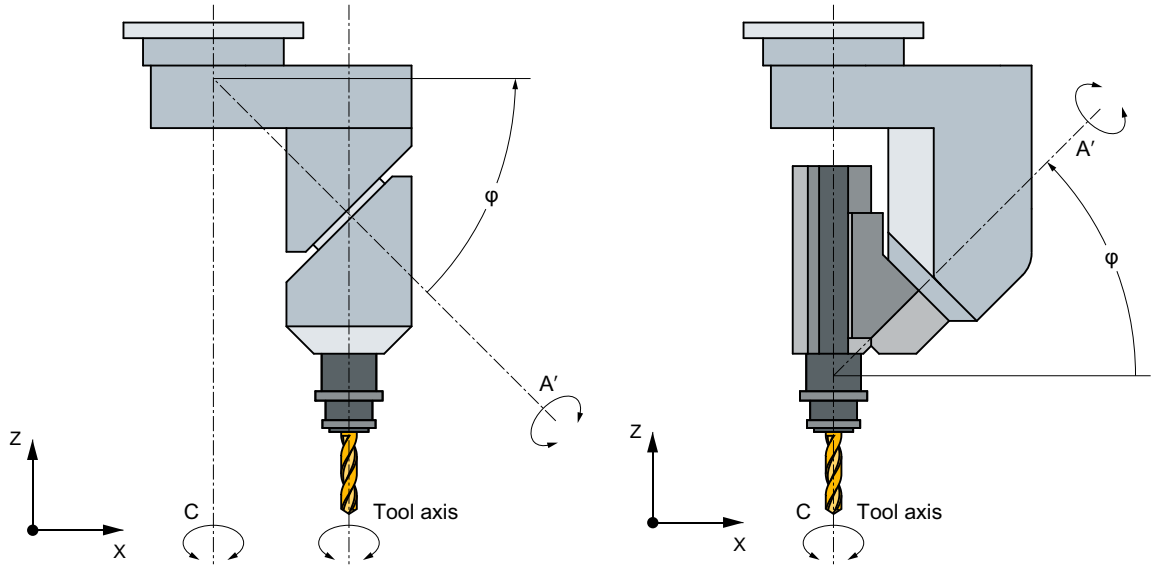


Figure 3-19 Schematic representation of the cardan milling head versions

Configuring the nutator angle φ

The angle of the inclined axis can be configured in a machine data:

$MC_TRAFO5_NUTATOR_AX_ANGLE_1: for the first orientation transformation

$MC_TRAFO5_NUTATOR_AX_ANGLE_2: for the second orientation transformation

The angle must lie within the range of 0 degrees to +89 degrees.

Tool orientation

Tool orientation at zero position can be specified as follows:

- parallel to the first rotary axis or
- perpendicular to it, and in the plane of the specified axis sequence

Types of kinematics

The axis sequence of the rotary axes and the orientation direction of the tool at zero position are set for the different types of kinematics using the following machine data:

$MC_TRAFO_TYPE_1 ... $MC_TRAFO_TYPE_10

Axis designation scheme

As for the other 5-axis transformations, the following applies:

the rotary axis ...

...A is parallel to X: A' is below the angle φ to the X axis

...B is parallel to Y: B' is below angle φ to the Y axis

...C is parallel to Z: C' is below angle φ to the Z axis

Angle definition

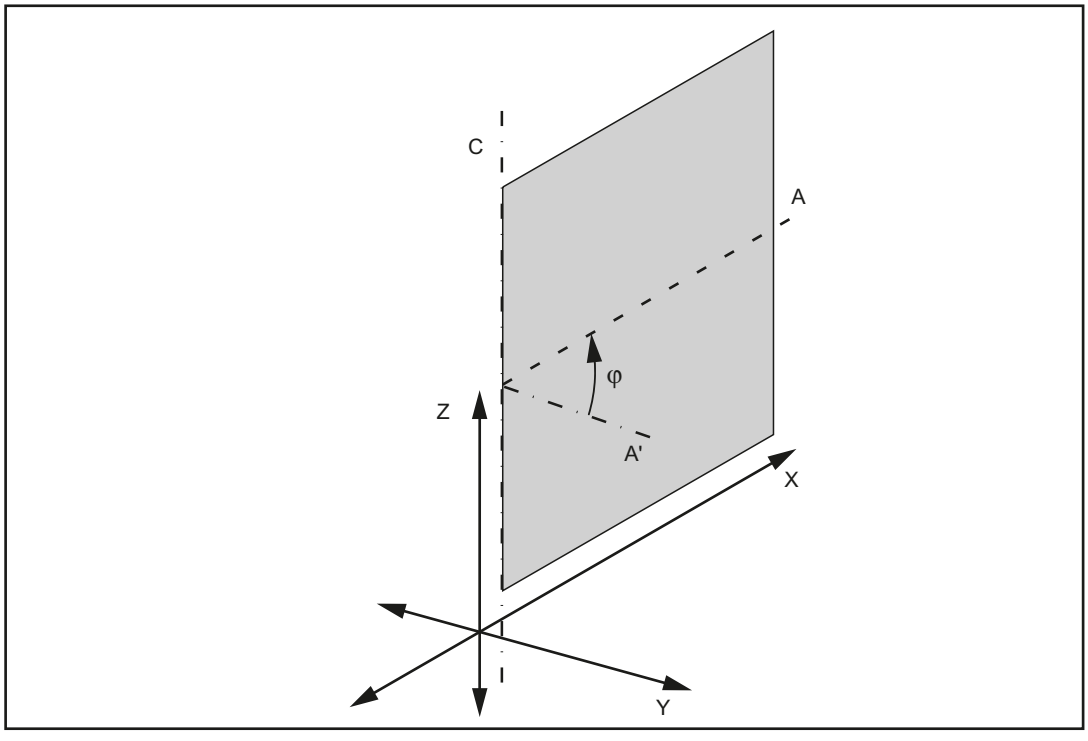


Figure 3-20 Position of axis A'

Axis A' is positioned in the plane spanned by the rectangular axes of the designated axis sequence. If, for example, the axis sequence is CA', then axis A' is positioned in plane Z-X. The angle φ then is the angle between axis A' and the X axis.

3.5.2 Parameterization

Setting the type of transformation

The transformation type is set with the machine data of the corresponding transformation data block:

MD24100, ... MD25190 $MC_TRAFO_TYP_n, with n = 1, 2, 3, ...

Bit	<value>	Description
7	1	Activation of the transformation for cardan milling head
6 - 5	Moving component	
	00	Movable **tool**
	01	Moving**workpiece**
	10	movable tool and workpiece
4 - 3	Direction of orientation of tool in position zero	
	00	X direction
	01	Y direction
	10	Z direction
2 - 0	Axis sequence	
	000	AB' or A'B
	001	AC' or A'C
	010	BA' or B'A
	011	BC' or B'C
	100	CA' or C'A
	101	CB' or C'B

The following transformation types can be set:

Axis sequence: Bits 0 - 2	Moving component: Bits 6 - 5								
	Tool 00			Workpiece 01			Tool/workpiece 10		
	Zero position [1]			Zero position [1]			Zero position [1]		
	X 00	Y 01	Z 10	X 00	Y 01	Z 10	X 00	Y 01	Z 10
AB' / A'B 000	x	x	-	-	-	-	-	-	-
AC' / A'C 001	x	-	x	-	-	-	-	-	-
BA' / B'A 010	x	x		-	-	-	-	x	x

Axis sequence: Bits 0 - 2	Moving component: Bits 6 - 5								
	Tool 00			Workpiece 01			Tool/workpiece 10		
	Zero position [1]			Zero position [1]			Zero position [1]		
	X 00	Y 01	Z 10	X 00	Y 01	Z 10	X 00	Y 01	Z 10
BC' / B'C 011	-	x	x	-	x	x	-	-	-
CA' / C'A 100	x	-	x	-	-	-	-	-	-
CB' / C'B 101	-	x	x	-	-	-	-	-	-

1) Orientation of the tool in the zero position: Bits 3 - 4

x: Transformation type can be set

-: Transformation type **cannot** be set

Active machining plane

The tool orientation at the zero position can be set not only in the Z direction. For this reason, ensure that the active working plane is set so that the tool length compensation acts in the tool orientation direction.

The active machining plane should always be the plane according to which the tool orientation is set in position zero.

Other settings

The geometry information used by the cardan milling head transformation for calculation of the axis values is set in the same way as that of the other 5-axis transformations.

3.5.3 Traverse of the cardan milling head in JOG mode

JOG

In JOG mode, the linear axes can be traversed normally. It is, however, difficult to set the orientation correctly by traversing these axes.

3.6 Programming of the 3- to 5-axis transformation

Switch on

The 3- to 5-axis transformations, including the transformations with swiveled linear axis and cardan milling head, are enabled with the `TRAORI(<transformation-no.>)` command. The enable of the transformation sets the NC/PLC interface signal:

DB21, ... DBX33.6 = 1 (transformation active)

Deactivation

With the `TRAFOOF` command disables the currently active 3- to 5-axis transformation. The disable of the transformation resets the NC/PLC interface signal:

DB21, ... DBX33.6 = 0 (transformation inactive)

Switch-over

If a transformation is already active in the channel, the `TRAORI(<transformation-no.>)` with a new transformation number command can be used to switch to another transformation.

Reset / program end

The control behavior after startup, program end or NC reset is set in the machine data:

MD20110 $MC_RESET_MODE_MASK, bit 7 = <value>

<Value>	Meaning
0	Initial setting for active transformation after reset / program end according to $MC_TRAFO_RESET_VALUE
1	The active transformation remains active over reset / program end

Option

The "5-axis transformation" function, together with its special forms, is an option.

Further information

A detailed description of the machine data can be found in:

Parameter Manual, Detailed Machine Data Description

See also

Dynamic orientation transformation TRAORI_DYN (Page 258)

3.7 Generic 5-axis transformation and variants

3.7.1 Functionality

Scope of functions

The scope of functions of generic 5-axis transformation covers implemented 5-axis transformations (see Section "5-axis transformation (Page 145)") for perpendicular rotary axes as well as transformations for the cardan milling head (one rotary axis parallel to a linear axis, the second rotary axis at any angle to it, see Section "Cardan milling head (Page 167)").

Applications

In certain cases, it may not be possible to compensate the conventional transformation machine accuracy, e.g. if:

- the rotary axes are not exactly mutually perpendicular or
- one of the two rotary axes is not positioned exactly parallel to the linear axes

In such cases, generic 5-axis transformation can produce better results.

Programming example

for generic 5-axis transformation is shown in Section "Example for Generic 5-axis Transformation".

Activation

Generic 5-axis transformation can also be activated like any other orientation transformation using the `TRAORI()` or `TRAORI(n)` command (where n is the number of the transformation). Furthermore, the basic transformation can be transferred in the call in three other parameters, e.g. `TRAORI(1, 1.1, 1.5, 8.9)`.

A transformation can be deselected implicitly by selecting another transformation or explicitly with `TRAFOOF`.

3.7.2 Description of machine kinematics

Machine types

Like the existing 5-axis transformations, there are three different variants of generic 5-axis transformation:

1. Machine type: Rotatable tool
   Both rotary axes change the orientation of the workpiece. The orientation of the workpiece is fixed.
2. Machine type: Rotatable workpiece
   Both rotary axes change the orientation of the workpiece. The orientation of the tool is fixed.
3. Machine type: Rotatable tool and rotatable workpiece - one rotary axis changes the tool orientation and the other the workpiece orientation.

Configurations

As previously, the machine configurations are defined in the following machine data (see Section "Configuration of a machine for 5-axis transformation (Page 147)"):

$MC_TRAFO_TYPE_1, ..., _8

Additional types have been introduced for generic 5-axis transformation:

Table 3-1 Overview of machine types for the generic 5-axis transformation

Machine type	1	2	3
Swivel/rotatable:	tool	workpiece	Tool/workpiece
Transformation types	24	40	56

Rotary axis direction

The direction of the rotary axis is defined by the following machine data:

$MC_TRAFO5_AXIS1_n (1st rotary axis) and

$MC_TRAFO5_AXIS2_n (2nd rotary axis)

where n is 1 or 2 for the first or second 5-axis transformation in the system respectively. The machine data specified above are fields with three values, which describe that axis direction vectorially (similar to the description of rotary axes for orientable toolholder). The absolute value of the vectors is insignificant; only the defined direction is relevant.

Example:

1. A-axis is the rotary axis (parallel to the x direction):

MD24570 $MC_TRAFO5_AXIS1_1[0] = 1.0 (direction first rotary axis)

MD24570 $MC_TRAFO5_AXIS1_1[1] = 0.0

MD24570 $MC_TRAFO5_AXIS1_1[2] = 0.0

2. B-axis is the rotary axis (parallel to the y direction):

MD24572 $MC_TRAFO5_AXIS2_1[0] = 0.0 (direction 2nd rotary axis)

MD24572 $MC_TRAFO5_AXIS2_1[1] = 1.0

MD24572 $MC_TRAFO5_AXIS2_1[2] = 0,0

3.7.3 Generic orientation transformation variants

Extension

Generic orientation transformation for 5-axis transformation has been extended with the following variants for 3-and 4-axis transformation:

Variant 1

4-axis transformations

A 4-axis transformation is characterized by the exclusive use of the first rotary axis as an entry axis of the transformation. The following applies:

MD24110 $MC_TRAFO_AXES_IN_1[4] = 0 (axis assignment for transformation 1) or

MD24210 $MC_TRAFO_AXES_IN_2[4] = 0 (axis assignment for transformation 2)

Variant 2

3-axis transformations

In a 3-axis transformation, one of the geometry axes is not present, by entering a zero in the field:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[n] (assignment between geometry axis and channel axis for transformation 1)

MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[n] (assignment between geometry axis and channel axis for transformation 2)

Transformation types

Both variants of generic 3- or 4-axis transformation are described by the following transformation types:

- 3- or 4-axis transformation with rotatable tool
  $MC_TRAFO_TYPE_n = 24
- 3- or 4-axis transformation with rotatable workpiece
  $MC_TRAFO_TYPE_n = 40

In conventional 3-axis or 4-axis transformations, the transformation type also defined the basic tool orientation in addition to the position of the rotary axis, which could then no longer be influenced.

Effects on orientations

Generic 3-axis or 4-axis transformation has the following effect on the various orientations:

The resulting tool orientation is defined according to the hierarchy specified for generic 5-axis transformation.

Priority:

- high: programmed orientation,
- medium: tool orientation and
- low: basic orientation

Allowance is made, in particular, for the following orientations:

- A programmed tool orientation
- A basic tool orientation, modified by orientable toolholders.

Note

Further information on programmable tool orientation and on basic tool orientation can be found in:

Further information
Function Manual Tools, tool offset, orientable tool carriers

Programming Manual, Fundamentals

Comparison

Besides the 3- and 4-axis transformations mentioned in Section "3- and 4-axis Transformations", the following differences should be noted:

- Position of the rotary axis:
  - can be arbitrary
  - need not be parallel to a linear axis
- Direction of the rotary axis
  - Must be defined by the following machine data:
    MD24570 $MC_TRAFO5_AXIS1_1[n] (direction 1st rotary axis) or
    MD24670 $MC_TRAFO5_AXIS1_2[n] (direction 1st rotary axis)
- Basic tool orientation
  - Must be defined by the following machine data:
    MD24574 $MC_TRAFO5_BASE_ORIENT_1[n] (workpiece orientation) or
    MD24674 $MC_TRAFO5_BASE_ORIENT_2[n] (workpiece orientation)
- Selection of a generic 3-/4-axis transformation
  - Optional tool orientation can be transferred as in the case of a generic 5-axis transformation.

3.7.4 Parameterization of orientable toolholder data

Application

Machine types for which the table or tool can be rotated, can either be operated as true 5-axis machines or as conventional machines with orientable toolholders. In both cases, machine kinematics is determined by the same data, which, due to different parameters, previously had to be entered twice - for toolholder via system variables and for transformations via machine data. The new transformation type 72 can be used to specify that these two machine types access identical data.

Transformation type 72

The following machine data can be used to define a generic 5-axis transformation for transformation type 72 with kinematic data read from the data for an orientable toolholder.

MD24100 $MC_TRAFO_TYPE_1 (definition of transformation 1 in the channel) or

MD24200 $MC_TRAFO_TYPE_2 (definition of transformation 2 in the channel)

From this number data is made available via machine data MD24582 $MC_TRAFO5_TCARR_NO_1 (TCARR-Number for the first 5-axis transformation) for the first or MD24682 $MC_TRAFO5_TCARR_NO_2 (TCARR-Number for the second 5-axis transformation) for the second orientation transformation. The corresponding transformation type can then be derived from the content of kinematic type with parameter $TC_CARR23 - see following table.

Table 3-2 Machine types for generic 5-axis transformation

Machine type	1	2	3	4
Swivel/ rotatable:	tool	workpiece	Tool/workpiece	Type 3 or orienta-ble toolholder
Kinematic type:	T	P	M	T, P, M
Transformation type:	24	40	56	72 from content of $TC_CARR23

Note

The transformation only takes place if the orientable toolholder concerned is available and the value of $TC_CARR23 contains a valid entry for type M, P or T kinematics in lower or upper case.

Transformation machine data for the first orientation transformation listed in the tables below are equally valid for the second orientation transformation. All other machine data that may affect the transformation characteristics and **do not** appear in the tables below, remain valid and effective:

MD24110/MD24210 $MC_TRAFO_AXES_IN_1/2 (axis assignment for transformation) or

MD24574/MD24674 $MC_TRAFO5_BASE_ORIENT_1/2 (basic tool orientation)

If in the tables below a second additive parameter appears in brackets for the parameters of the orientable toolholder (e.g. $TC_CARR24 (+ $TC_TCARR64)), the sum of both values will only be effective if the fine offset specified in setting data is active when the data is transferred from the orientable toolholder.

SD42974 $SC_TOCARR_FINE_CORRECTION = TRUE (fine offset TCARR on/off)

Activation

The most significant parameter values of an orientable toolholder for a transformation can be activated in the part program with NEWCONF. Alternatively, the machine data concerned for transformation type 72 can be activated via the HMI user interface.

Assignment for all types of transformation

The assignments between the toolholder data for writing the linear offsets and the corresponding machine data for kinematic transformations are determined by the transformation type. The following assignment of all other parameters is identical for all three possible types of transformation:

Assignment for all types of transformation together identical			
MD24100 $MC_TRAFO_TYPE_1 (definition of transformation 1 in the channel)	24	$TC_CARR23 =	T
	40		P
	56		M
MD24570 $MC_TRAFO5_AXIS1_1[0] (direction 1st rotary axis)		$TC_CARR7	
MD24570 $MC_TRAFO5_AXIS1_1[1]		$TC_CARR8	
MD24570 $MC_TRAFO5_AXIS1_1[2]		$TC_CARR9	
MD24572 $MC_TRAFO5_AXIS2_1[0] (direction 2nd rotary axis)		$TC_CARR10	
MD24572 $MC_TRAFO5_AXIS2_1[1]		$TC_CARR11	
MD24572 $MC_TRAFO5_AXIS2_1[2]		$TC_CARR12	
MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1[0] (position offset of rotary axes 1/2/3 for 5-axis transformation) 1)		$TC_CARR24 (+ $TC_TCARR64)	
MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1[1]		$TC_CARR25 (+ $TC_TCARR65)	

Assignment for all types of transformation together identical	
MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1[0] (sign of rotary axis 1/2/3 for 5-axis transformation 1)	TRUE*
MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1[1]	TRUE*

*) Machine data MD24520/MD24620 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1/2 are redundant. They are used to invert the direction of rotation of the assigned rotary axis. However, this can also be achieved by inverting the direction of axis vector $MC_TRAFO5_AXIS1/2_1/2. It is for this reason that there is no corresponding parameter for the orientable toolholder. For the purpose of absolute clarity, the following machine data must be ignored:

MD24520/MD24620 TRAFO5_ROT_SIGN_IS_PLUS_1/2

Assignments for transformation type 24

Toolholder data assignments dependent on transformation type 24

Transformation type "T" (in accordance with MD24100 $MC_TRAFO_TYPE_1 = 24)	
MD24500 $MC_TRAFO5_PART_OFFSET_1[0] (translation vector of 5-axis transformation 1)	$TC_CARR1 (+$TC_TCARR41)
MD24500 $MC_TRAFO5_PART_OFFSET_1[1]	$TC_CARR2 (+$TC_TCARR42)
MD24500 $MC_TRAFO5_PART_OFFSET_1[2]	$TC_CARR3 (+$TC_TCARR43)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[0] (vector of the kinematic offset of 5-axis transformation 1)	$TC_CARR4 (+$TC_TCARR44)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[1]	$TC_CARR5 (+$TC_TCARR45)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[2]	$TC_CARR6 (+$TC_TCARR46)
MD24550 $MC_TRAFO5_BASE_TOOL_1[0] (vector of basic tool when 5-axis transformation is active) 1)	$TC_CARR15 (+$TC_TCARR55)
MD24550 $MC_TRAFO5_BASE_TOOL_1[1]	$TC_CARR16 (+$TC_TCARR56)
MD24550 $MC TRAFO5_BASE_TOOL_1[2]	$TC_CARR17 (+$TC_TCARR57)

Assignments for transformation type 40

Toolholder data assignments dependent on transformation type 40

Transformation type "P" (in accordance with MD24100 $MC_TRAFO_TYPE_1 = 40)	
MD24550 $MC_TRAFO5_BASE_TOOL_1[0]	$TC_CARR4 (+$TC_TCARR44)
MD24550 $MC_TRAFO5_BASE_TOOL_1[1]	$TC_CARR5 (+$TC_TCARR45)
MD24550 $MC_TRAFO5_BASE_TOOL_1[2]	$TC_CARR6 (+$TC_TCARR46)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[0]	$TC_CARR15 (+$TC_TCARR55)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[1]	$TC_CARR16 (+$TC_TCARR56)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[2]	$TC_CARR17 (+$TC_TCARR57)
MD24500 $MC_TRAFO5_PART_OFFSET_1[0]	$TC_CARR18 (+$TC_TCARR58)

Transformation type "P" (in accordance with MD24100 $MC_TRAFO_TYPE_1 = 40)	
MD24500 $MC_TRAFO5_PART_OFFSET_1[1]	$TC_CARR19 (+$TC_TCARR59)
MD24500 $MC_TRAFO5_PART_OFFSET_1[2]	$TC_CARR20 (+$TC_TCARR60)

Assignments for transformation type 56

Toolholder data assignments dependent on transformation type 56

Transformation type "M" (in accordance with MD24100 $MC_TRAFO_TYPE_1 = 56)	
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[0] (vector of the kinematic offset of 5-axis transformation 1)	$TC_CARR1 (+$TC_TCARR41)
MD24560 $MC_TRAFO5_JOINT_OFFSET_1[1]	$TC_CARR2 (+$TC_TCARR42)
MD24560: TRAFO5_JOINT_OFFSET_1[2]	$TC_CARR3 (+$TC_TCARR43)
MD24550 $MC_TRAFO5_BASE_TOOL_1[0]	$TC_CARR4 (+$TC_TCARR44)
MD24550 $MC_TRAFO5_BASE_TOOL_1[1]	$TC_CARR5 (+$TC_TCARR45)
MD24550 $MC_TRAFO5_BASE_TOOL_1[2]	$TC_CARR6 (+$TC_TCARR46)
MD24558 $MC_TRAFO5_JOINT_OFFSET_PART_1[0]	$TC_CARR15 (+$TC_TCARR55)
MD24558 $MC_TRAFO5_JOINT_OFFSET_PART_1[1]	$TC_CARR16 (+$TC_TCARR56)
MD24558 $MC_TRAFO5_JOINT_OFFSET_PART_1[2]	$TC_CARR17 (+$TC_TCARR57)
MD24500 $MC_TRAFO5_PART_OFFSET_1[0]	$TC_CARR18 (+$TC_TCARR58)
MD24500 $MC_TRAFO5_PART_OFFSET_1[1]	$TC_CARR19 (+$TC_TCARR59)
MD24500 $MC_TRAFO5_PART_OFFSET_1[2]	$TC_CARR20 (+$TC_TCARR60)

Example of parameterization

The first 5-axis transformation is to obtain its data from machine data and the second, in contrast, is to be parameterized using the data from the 3rd orientable toolholder.

MD24100 $MC_TRAFO_TYPE_1 = 24	; first 5-axis transformation
MD24200 $MC_TRAFO_TYPE_2 = 72	; second 5-axis transformation
MD24682 $MC_TRAFO5_TCARR_NO_2 = 3;	; parameterize data of the third orientable ; toolholder

3.7.5 Extension of the generic transformation to 6 axes

6-axis transformation is an extension of the generic 5-axis transformation to include an additional rotary axis. This allows rotation of the tool around itself to be defined.

In addition to the machine data for 5-axis transformation, to parameterize 6-axis transformation, additional settings must be made in the subsequently listed machine data.

Parameter assignment: Transformation type

The transformation type is set on a channel-for-channel basis using:

$MC_TRAFO_TYPE_<x> = <transformation type>

The transformation type to be set is obtained from the machine type and/or the allocation of the three rotary axes regarding tool and workpiece orientation:

<Transformation type>	24 [1]	40 [1]	56 [2]	57 [3]
Machine type	1	2	3	4
Tool orientation	3 rotary axes	---	2 rotary axes	1 rotary axis
Workpieceorientation	---	3 rotary axes	1 rotary axis	2 rotary axes

1) The **first** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **workpiece**.
The **third** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **tool**.
When the **second** rotary axis traverses, then the third rotary axis is also moved.

2) The **second** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **workpiece**.
The **third** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **tool**.
When the **first** rotary axis traverses, then the third rotary axis is also moved.

3) The **first** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **tool**.
The **third** rotary axis of the transformation is the rotary axis that in the kinematic chain is located next for the **workpiece**.
When the **second** rotary axis traverses, then the third rotary axis is also moved.

Note

Using transformation types 24, 40, 56 and 57, only those kinematics are formed, where the three linear axes of the transformation form a right-handed Cartesian coordinate system - and, there is no rotary axis between the linear axes in the kinematic chain.

Parameter assignment: Offset vectors for transformation type 57

For transformation type 57, the same as for rotary axes, the offset vectors are to be specified starting from the tool to the workpiece zero:

Offset	Machine data
Tool tip → rotary axis 1	$MC_TRAFO5_BASE_TOOL_<x>[0 ... 2]
Rotary axis 1 → rotary axis 2	$MC_TRAFO5_JOINT_OFFSET_<x>[0 ... 2]
Rotary axis 2 → rotary axis 3	$MC_TRAFO6_JOINT_OFFSET_2_3_<x>[0 ... 2]
Rotary axis 3 → workpiece zero	$MC_TRAFO5_PART_OFFSET_<x>[0 ... 2]

Parameter assignment: Channel axis number of the 3rd rotary axis

The channel axis number of the third rotary axis is set in:

$MC_TRAFO_AXES_IN_<x>[**5**] = <channel axis number of the third rotary axis>

Parameter assignment: Direction vector of the 3rd rotary axis

The direction vector of the third rotary axis is set in:

$MC_TRAFO5_AXIS3_<x>[0 ... 2] = <direction vector component>

Parameter assignment: Tool normal vector

It is not permissible that the tool normal vector is in parallel or is antiparallel to the basic tool orientation vector (MD24574 $MC_TRAFO5_BASE_ORIENT_<x>). It is set in:

$MC_TRAFO6_BASE_ORIENT_NORMAL_<x>[0 ... 2] = <component of the tool normal vector>

Parameter assignment: Offset vector between the second and third rotary axes

The offset vector between the second and third rotary axes is set in:

$MC_TRAFO6_JOINT_OFFSET_2_3_<x>[0 ... 2] = <component of the offset vector>

Orientation

With the extension of the generic orientation transformation to six axes, all three degrees of freedom of the orientation can be freely selected. They can be uniquely defined through the position of a rectangular Cartesian coordinate system. The axis direction of the **third axis** defines the **orientation**.

Two degrees of freedom are required to specify this axis direction. The third degree of freedom is defined via a rotation around this axis direction, e.g. by specifying an angle THETA or a direction vector for one of the two other axes of the coordinate system (see Chapter "Rotations of orientation vector (Page 223)").

Addresses AN3, BN3, CN3 define the direction of the **second axis** of the coordinate system, i.e. the **orientation normal vector**. The orientation normal vector, programmed with addresses AN3, BN3, CN3, should be perpendicular to the orientation. If this is not the case, then the programmed orientation normal vector is internally modified to achieve this. Then, the modified orientation normal vector lies in the plane formed by the programmed orientation and orientation normal vector - and is perpendicular to the orientation vector. This orthogonalization is only possible if the two programmed vectors are neither parallel nor antiparallel. Alarm 4342 is output if this condition is not satisfied.

Direction vectors in the tool data

Orientation vector

Vector components (x,y,z) of the orientation vector of a tool can be defined in the tool data using $TC_DPV or $TC_DPV3 - $TC_DPV5.

Further information: Function Manual Tools; Tool offsets, sum and setup offsets

Orientation normal vector

Vector components (x,y,z) of the orientation normal vector of a tool can be defined in the tool data using $TC_DPVN3 - $TC_DPVN5. The significance of the vector components is analogous to the vector components of the orientation vector:

- $TC_DPVN3: Component in the direction of the tool length L1,
- $TC_DPVN4: Component in the direction of the tool length L2
- $TC_DPVN5: Component in the direction of the tool length L3.

The following settings must be made so that the tool data for the orientation normal vector can be used:

MD18114 $MN_MM_ENABLE_TOOL_ORIENT = 3 (assign tool cutting edges orientation)

The coordinate system is not rotated by rotating the tool with AN3, BN3, CN3 or THETA.

Default setting of the orientation normal vector

The initial state of the orientation normal vector is defined when **activating the transformation** in one of the following ways:

1. For a TRAORI function call, vector components (x,y,z) are transferred as parameters 8 - 10: TRAORI(p1, p2, p3, p4, p5, p6, p7, **p8**, **p9**, **p10**)
2. If, when calling the TRAORI function, **no** orientation normal vector is specified, and **one** tool is active, then vector components (x,y,z) are taken from the tool data: $TC_DPVN3, $TC_DPVN4, $TC_DPVN5
3. If, when calling the TRAORI function, **no** orientation normal vector is specified, and **no** tool is active, then vector components (x,y,z) are taken from the following machine data: MD24567 $MC_TRAFO6_BASE_ORIENT_NORMAL_<x>[0 ... 2] (tool normal vector)

The position of the orientation coordinate system of a standard tool depends on the **active plane** (G17, G18, G19) corresponding to the following table:

	G17	G18	G19
Direction of the orientation vector	Z	Y	X
Direction of the orientation normal vector	Y	X	Z

See also

Example of a generic 6-axis transformation (Page 245)

3.7.6 Extension of the generic transformation to 7 axes

Application

The generic 5-/6-axis transformation with transformation type 24 is extended by a 7th or 6th axis, which rotates the workpiece. The work space of the transformation can be expanded in this way.

Requirement

For generic 7-axis transformation there must be at least 6 or 7 axes.

Function

Another 7th axis is required in connection with the generic 6-axis transformation which rotates the workpiece. This 7th Axis is considered only along with transformation type 24 (generic 6-axis transformation having 3 rotary axes that move the tool).

The position of the 7th axis is specified according to a strategy of the CAD system and settled with the Cartesian position (X, Y, Z) by the generic transformation in such a way that the axes always approach the TCP position programmed with reference to the workpiece, independently of the position of the 7th axis. If ORIWKS is active, the end orientation programmed with reference to the workpiece is also rotated by the 7th axis. This way it is possible to program the orientation in relation to the workpiece.

The transformation uses the 7th axis as the observed input variable.

To configure the 7th axis, the channel machine data of the 5-/6-axis transformation is extended by one field containing the 3 components of the direction vector of the 7th axis and an axis offset.

This gives the following advantages:

- The contour and the orientation at the workpiece can be programmed in relation to the workpiece.
- The programmed feed is maintained in the contour, even if the 7th axis also moves.
- All the contour-related control functions can be used.
- The displayed WCS position corresponds to the programmed position.
- The transformation is configured as in generic 6-axis transformation. One can switch between a 6-axis and a 7-axis transformation smoothly.
- In case of large radius circular interpolation, the release of singularities incorporating the 7th axis.

Notations

Dedicated machine data exist for each general transformation and for each orientation transformation that are differentiated by the suffixes _1, _2 etc. (e.g. $MC_TRAFO_TYPE_1, $MC_TRAFO_TYPE_2 etc.). In the following, only the names for the first transformation are specified, i.e. those with the suffix _1. If a transformation other than the first is parameterized, the correspondingly modified names must be used.

Description of the kinematics

The 7-axis transformation builds on the generic 5-/6-axis transformation.

Note

The 7-axis transformation also covers kinematics in which the 6th axis is not available. In the following, we speak exclusively about a 7th axis or about a 7-axis transformation, even when it is actually the 6th axis in connection with a 5-axis kinematics.

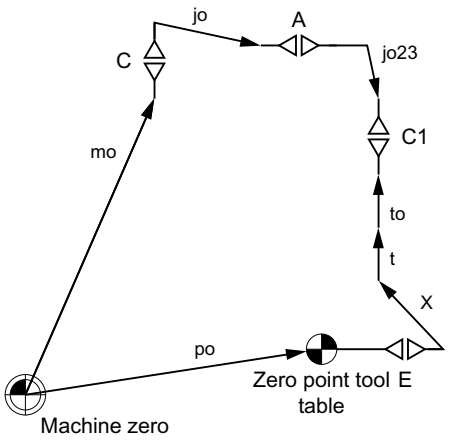
The 7-axis transformation types only cover those kinematics in which the three linear axes form a rectangular Cartesian coordinate system, i.e. no kinematics are covered in which at least one rotary axis lies between two linear axes in the kinematic chain.

There is only one machine kinematics for which a 7th axis can be configured. It is designated by the Transformation Type 24:

$MC_TRAFO_TYPE_1 = 24 Rotary tool: Three (or two) axes rotate the tool; the 7th axis rotates the workpiece.

The extensions of the following machine data are required to configure a generic 7-axis transformation:

Machine data	extension
$MC_TRAFO_AXES_IN_1[9]	The channel axis index of the 4th rotary axis is recorded here.
$MC_TRAFO_AXES_IN_1[10] and $MC_TRAFO_AXES_IN_1[11]	This machine data is to be assigned with default value of 0. (default setting)
$MC_TRAFO_AXES_IN_1[6..8]	This machine data is not evaluated by the generic 7-axis transformation.
$MC_TRAFO7_EXT_AXIS1_1[0..2]	The direction of the 4th rotary axis is specified here.
$MC_TRAFO7_EXT_AX_OFFSET_1[0..2]	A position offset of the 4th rotary axis is entered here.



mo:	Position vector in MCS
po:	$MC_TRAFO5_PART_OFFSET_n[0..2]
x.	Vector of programmed position in the WCS
t:	Tool correction vector
to:	$MC_TRAFO5_BASE_TOOL_n[0..2]
jo:	$MC_TRAFO5_JOINT_OFFSET_n[0..2]
jo23:	$MC_TRAFO6_JOINT_OFFSET_2_3_n[0..2]

Figure 3-21 Schematic diagram of 7-axis kinematics

Programming

1. **Programming the Cartesian position**
   The position of the 7th axis must be programmed in the workpiece coordination system in addition to the Cartesian position. The Cartesian position is thus programmed in relation to the constant workpiece. The 7-axis transformation converts the WCS position via the rotation of the 7th axis in the basic coordinate system. Possibly programmed or set frames are normally settled before the 7-axis transformation.
2. **Programming of orientation**
   All programming options of the generic 5/6-axis transformation are available while programming the orientation. The 7th axis must always be programmed additionally. Two different response types can be set in this context via the G command.
   - The position of the 7th axis does not influence the programmed orientation.
   - The programmed end orientation is rotated with the 7th axis.

Orientation

1. **Orientation with axis interpolation**
   If the 7th axis should have no influence on the programmed orientation, the G commands of groups 25 and 51 must be set accordingly:
   G group 25: ORIMKS
   G group 51: ORIAXES (if MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 is set).
   The programmed positions of the rotary axes are not changed by the position of the 7th axis in this case, but are approached directly. The orientation is programmed in relation to the machine.
   Example

```
Program code
TRAORI(1)
ORIAXES
ORIMKS
G1 X500 Y300 Z800 C15 A5 C1=10 E1=120
```

1. **Orientation and large circle interpolation**
   If traversing is to be done with large radius circular interpolation, the end orientation is rotated with the 7th axis.
   G command group 25: ORIWKS
   G command group 51: ORIVECT (if MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 is set).
   In this case the orientation must be programmed in relation to the workpiece. The programmed orientation is thus related to the fixed workpiece. The position of the 7th axis is thus not contained in the programmed orientation.
   Example

```
Program code
TRAORI(1)
ORIVECT
ORIWKS
G1 X500 Y800 Z100 A3=0 B3=1 C3=0 AN3=0 BN3=0 CN3=-1 E1=-90
```

Frames

The basic coordinate system sits on the 7th axis. It is also rotated when the 7th axis rotates. This way the workpiece coordinate system (WCS) does not remain stationary when the workpiece is rotated over the 7th axis. A workpiece position rotated to the zero position of the 7th axis can be compensated by an axial frame offset of the 7th axis.

Traversing with the 7th axis in JOG mode

Only the compensatory movements for the linear axes are created if the 7th axis is traversed in the JOG mode with active 7-axis transformation. The position at the workpiece is kept constant in this way. As the rotary axes go into the transformation only as input axes, they are not influenced by the 7th axis during the JOG travel. The orientation at the workpiece is thus kept variable.

3.7.7 Cartesian manual travel with generic transformation

Note
Option

The "Handling transformation package" option is necessary for the function.

Functionality

With function "Cartesian manual travel", the following Cartesian coordinate system can be set for traversing geometry and orientation axis in the JOG mode:

- Basic coordinate system (BCS)
- Workpiece coordinate system (WCS)
- Tool coordinate system (TCS)

Traversing motion of linear and rotary axes is realized via the NC/PLC interface signals of the geometry and/or orientation axes.

Note

For further information about the representation of the translations for the Cartesian manual travel in the corresponding coordinate systems, see:
Further information
M1: Kinematics transformation (Page 19)

Parameter assignment: Machine data

Coordinate systems

The following machine data not only activates the function, but also sets the permitted coordinate systems, which can be toggled between.

MD21106 $MC_CART_JOG_SYSTEM, bit n = <value>

Bit n	Value	Meaning
0	1	Function active: Basic coordinate system (BCS)
1	1	Function active: Workpiece coordinate system (WCS)
2	1	Function active: Tool coordinate system (TCS)

Parameter assignment: Setting data

Defining the virtual kinematics for the orientation axes

The active virtual kinematics for manually traversing the orientation axes is defined using the following setting data. Here, only kinematics can be set, where the rotary axes are perpendicular to one another.

SD42660 $SC_ORI_JOG_MODE = <value>

Value	Meaning
0	The virtual kinematics are defined using transformation
1	Rotation with Euler angles: Rotation sequence, ZX'Z" convention:
2	Rotation with Euler angles: XY'Z" convention (RPY angle)
3	Rotation with Euler angles: Rotation sequence ZY'X" convention (RPY angles)
4	Defining the rotation sequence using: MD21120 $MC_ORIAX_TURN_TAB_1
5	Defining the rotation sequence using: MD21130 $MC_ORIAX_TURN_TAB_2

For further explanations regarding orientation motion (see Chapter "Orientation (Page 191)" and "Orientation axes (Page 212)").

Note

For further information on programming the rotations, see:
Further information
Programming Manual Advanced; Transformations

3.8 Restrictions for kinematics and interpolation

3.8.1 Restrictions for kinematics and interpolation

For systems where there are less than six axes available for transformation, the following restrictions must be taken into account.

5-axis kinematics

For 5-axis kinematics there are two degrees of freedom for orientation. The assignment of orientation axes and tool vector direction must be selected so that there is no rotation around the tool vector. As a result, only two orientation angles are required to describe the orientation. If the axis is traversed using `ORIVECT`, the tool vector performs pure swiveling motion.

3-and 4-axis kinematics

For 3- and 4-axis kinematics, only one degree of freedom is available for orientation. The respective transformation determines the relevant orientation angle. In this case, it only makes sense to traverse the orientation axis using `ORIAXES`. In this case, the orientation axis is directly and linearly interpolated.

Interpolation of the tool orientation over several blocks by means of orientation vectors

If the orientation of a tool is programmed over several consecutive part program blocks by directly entering the appropriate rotary axis positions, then undesirable discontinuous changes of the orientation vector are obtained at the block transitions. This results in discontinuous velocity and acceleration changes of the rotary axes. This means that no continuous velocity and acceleration of the orientation axes over several blocks can be achieved using large circle interpolation.

Continuous block transitions

As long as only linear blocks (`G1`) are programmed, then the orientation axes also behave just like linear axes. In this case, motion with continuous acceleration is achieved through polynomial interpolation. Significantly better results can be achieved by programming the orientation in space using orientation vectors (see Section "Polynomial interpolation of orientation vectors (Page 219)").

3.8.2 Singularities of orientation

Description of problem

As described in Section "Singularities and how to treat them", singularities (poles) are constellations in which the tool is orientated becomes parallel to the first rotary axis. If the orientation is changed when the tool is in or close to a singularity (as is the case with large-circle interpolation ORIWKS), the rotary axis positions must change by large amounts to achieve small changes in orientation. In extreme cases, a jump in the rotary axis position would be needed.

Such a situation would be treated as follows:

There is only one relevant machine data, which circles the pole as usual:

MD24540 $MC_TRAFO5_POLE_LIMIT_1 (closing angle tolerance for interpolation by pole for 5-axis transformation)

or

MD24640 $MC_TRAFO5_POLE_LIMIT_2 (closing angle tolerance for interpolation by pole for 5-xis transformation)

For further information about the handling of singular positions, see:

Further information

Programming Manual Advanced; Transformations, Cartesian PTP travel

Example for machine type 1

Rotatable tool

Both rotary axes change the orientation of the workpiece. The orientation of the workpiece is fixed.

2-axis swivel head with rotary axis RA 1 (4th transformation axis) and rotary axis RA 2 (5th transformation axis)

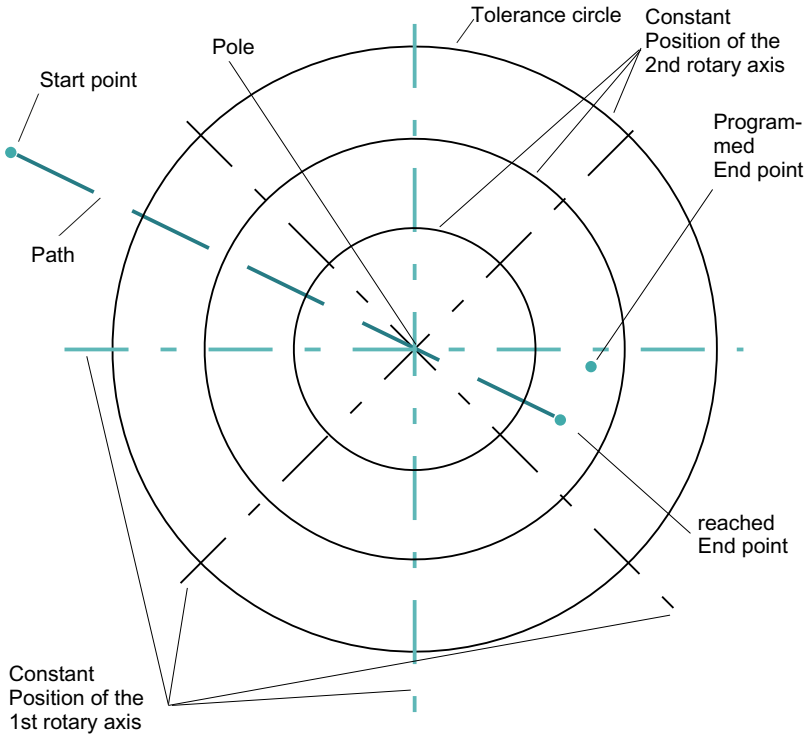


Figure 3-22 Generic 5-axis transformation; end point of orientation inside tolerance circle.

End point within the circle

If the end point is within the circle, the first axis comes to a standstill and the second axis moves until the difference between target and actual orientation is minimal. However, since the first rotary axis does not move, the orientation will generally deviate from the programmed value (see previous figure). However, the programmed orientation can at least be reached exactly if the first rotary axis happens to be positioned correctly.

Note

In the previous Figure the resulting path is a straight line because the position of the first rotary axis is constant on that path. This representation is always correct, irrespective of the angle between the two rotary axes. The orientation vector only moves in a plane, however, if the two rotary axes and the basic orientation are all mutually perpendicular. In all other cases, the orientation vector describes the outside of a cone.

End point outside the circle

If the orientation interpolation describes a path through the circle, while the end point is outside the circle, the end point is approached with axis interpolation. This applies in particular if the interpolation starting point is located inside the circle. Path deviations from the programmed setpoint orientation are thus unavoidable.

3.9 Orientation

3.9.1 Basic orientation

Differences to the previous 5-axis transformations

In the 5-axis transformations implemented to date, basic orientation of the tool was defined by the type of transformation.

Generic 5-axis transformation can be used to enable any basic tool orientation, i.e. space orientation of the tool is arbitrary, with axes in their initial positions.

If an orientation is programmed by means of Euler angles, RPY angles (A2, B2, C2) or vectors (A3, B3, C3), basic orientation is taken into consideration, i.e. the rotary axes are positioned so that a tool positioned in basic orientation is traversed to the programmed orientation.

If the rotary axes are programmed directly, basic orientation has no effect.

Definition

The basic orientation can be defined in three different ways:

- Definition by calling the transformation
- Definition by the orientation of the active tool
- Definition using a machine data

Definition by calling the transformation

When the transformation is called, the direction vector of the basic orientation can be specified in the call, e.g. `TRAORI(0, 0., 1., 5.)`. The direction vector is defined by parameters 2 to 4. This is the reason that in the example it has the value (0., 1., 5.).

The first parameter specifies the transformation number. The number can be omitted if the first transformation is to be activated. To enable the parameters to be identified correctly when specifying an orientation, a blank space has to be inserted instead of the transformation number, e.g. `TRAORI(, 0., 1., 5.)`.

Note

The orientation data is absolute. It is not modified by a frame that is possibly active.

The absolute value of the vector is insignificant, only the direction is relevant. Non-programmed vector elements can be set to zero.

Please note that if all three vector components are zero (because they have been set explicitly so or not specified at all), the basic orientation is not defined by data in the `TRAORI(...)` call, but by one of the two other options.

If a basic orientation is defined by calling the transformation, it cannot be altered while a transformation is active. The orientation can be changed only by selecting the transformation again.

Definition by the orientation of the active tool

The basic orientation is determined by the tool, if:

- it has not been defined by specifying a direction vector in the transformation call and
- a tool is already active.

The orientation of a tool is dependent on the selected plane. It is parallel to Z at G17, parallel to Y at G18 and parallel to X at G19.

It can be modified arbitrarily by orientable toolholders, see:

Further information
Function Manual Tools; Tool offset, orientable tool carriers

If the tool is changed when a transformation is active, the basic orientation is also updated. The same applies if the orientation of a tool changes as the result of a change in plane (plane changes are equivalent to tool changes, as they also alter the assignment between tool length components and individual axes).

If the tool is de-selected, thereby canceling the definition of tool orientation, the basic orientation programmed in machine data becomes operative.

Definition using a machine data

If the basic orientation is not defined by either of the two variants described above, it is specified with reference to the following machine data:

$MC_TRAFO5_BASE_ORIENT_n (basic tool orientation)

This machine data must not be set to a zero vector or else an alarm will be generated during control run-up when a transformation is active.

If a basic orientation is programmed in machine data $MC_TRAFO5_BASE_ORIENT_n when a transformation is active and a tool is subsequently activated, the basic orientation is re-defined by the tool.

Note

The range of settable orientations depends on the directions of the rotary axes involved and the basic orientation. The rotary axes must be mutually perpendicular if all possible orientations are to be used. If this condition is not met, "dead" ranges will occur.

Examples:

1. Extreme example: A machine with rotatable tool has a C axis as its first rotary axis and an A axis as its second. If the basic orientation is defined in parallel to the A axis, the orientation can only be changed in the X-Y plane (when the C axis is rotating), i.e. orientation with a Z component unequal to zero is not possible in this instance. The orientation does not change when the A axis rotates.
2. Realistic example: A machine with nutator kinematics (cardan head) with an axis inclined at less than 45° in a basic orientation parallel to the Z axis can only assume orientations within a semi-circle: The top semi-circle with basic orientation towards +Z and the bottom with basic orientation towards -Z.

3.9.2 Orientation movements with axis limits

Calculate rotary axis position

If the final orientation in a 5-axis transformation is programmed indirectly in an NC block by means of a Euler, RPY angle or direction vector, it is necessary to calculate the rotary axis positions that produce the desired orientation. This calculation has no unique result.

There are always at least two essentially different solutions. In addition, any number of solutions can result from a modification to the rotary axis positions by any multiple of 360 degrees.

The control system chooses the solution which represents the shortest distance from the current starting point, allowing for the programmed interpolation type.

Determining permissible axis limits

The control system attempts to define another permissible solution if the axis limits are violated, by approaching the desired axis position along the shortest path. The second solution is then verified, and if this solution also violates the axis limits, the axis positions for both solutions are modified by multiples of 360 until a valid position is found.

The following conditions must be met in order to monitor the axis limits of a rotary axis and modify the calculated end positions:

- A generic 5-axis transformation of type 24, 40 or 56 must be active.
- The axis must be referenced.
- The axis must not be a modulo rotary axis.
- The following machine data may not be equal to zero:
  MD21180 $MC_ROT_AX_SWL_CHECK_MODE (check software limits for orientation axes)

MD21180 $MC_ROT_AX_SWL_CHECK_MODE specifies the conditions under which the rotary axis positions may be modified:

Value	Meaning
0	No modification permitted (default, equivalent to previous behavior).
1	Modification is only permitted if axis interpolation is active (ORIAXES or ORIMKS).
2	Modification is always permitted, even if vector interpolation (large circle interpolation, conical interpolation, etc.) was active originally.

Switch-over to axis interpolation

If the axis positions have to be changed from the originally determined value, the system switches to rotary axis interpolation because the original interpolation path, e.g. large circle interpolation or conical interpolation, can no longer be maintained.

Example

An example for modifying the rotary axis motion of a 5-axis machine with a rotatable tool is shown in Chapter "Example of generic 5-axis transformation (Page 243)".

3.9.3 Orientation compression

Function

Using compressor functions COMPON, COMPCURV, COMPCAD and COMPSURF, NC programs, in which the orientation is programmed using direction vectors, can be compressed, but still maintaining a specifiable tolerance.

Requirement

An orientation motion is only compressed under the following conditions:

- Orientation transformation (TRAORI) is active.
- Large-radius circular interpolation is active.
  I.e. tool orientation is changed in the plane that is determined by start and end orientation.
  Large-radius circular interpolation is performed under the following conditions:
  - MD21104 $MC_ORI_IPO_WITH_G_CODE = 0
    + ORIWKS is active.
    + orientation is programmed using vectors (with A3, B3, C3 or A2, B2, C2).
  - MD21104 $MC_ORI_IPO_WITH_G_CODE = 1
    + ORIVECT or ORIPLANE is active.
    The tool orientation can be programmed either as a direction vector or with rotary axis positions. Large-radius circular interpolation is not executed if G command ORICONxx or ORICURVE is active or if polynomials for orientation angle (PO[PHI] and PO[PSI]) are programmed.

Parameterization

NC blocks can only be compressed if deviations are allowed between the programmed contour and interpolated contour or between the programmed orientation and interpolated orientation.

Compression tolerances can be used to set the maximum permissible deviation. The higher the tolerances, the more blocks can be compressed. However, the higher the tolerances, the more the interpolated contour or orientation can deviate from the programmed values.

Axis accuracy

For each programmed path, the compressor generates a spline curve such that, for the axes involved, the maximum tolerance set with the machine data is complied with in the endpoints:

MD33100 $MA_COMPRESS_POS_TOL = <maximum deviation with compression>

Contour accuracy

The maximum deviation from the programmed contour and tool orientation permitted during compression is set with the following setting data:

- SD42475 $SC_COMPRESS_CONTUR_TOL = <maximum contour deviation>
- SD42476 $SC_COMPRESS_ORI_TOL = <maximum angular deviation of the tool orientation>
  Only effective with orientation transformation.
- SD42477 $SC_COMPRESS_ORI_ROT_TOL = <maximum angular deviation of the tool orientation>
  Only effective with orientation transformation in conjunction with 6-axis transformation.

Note

A maximum deviation of tool orientation can only be specified if an orientation transformation is active (TRAORI).

Compression mode

With the machine data, the following decimal-coded functions are parameterized:

- Units digit: Taking into account tolerances
- Tens digit: Compression of blocks with programmed tool orientation and / or value assignments.
- Hundreds digit: Compression of blocks, except linear blocks (G1).
- Thousands digit: Compression for specific applications.

MD20482 $MC_COMPRESSOR_MODE = <value>

Value	Meaning	
Units digit (Effective tolerance specifications)		
xxx0	Geometry and orientation axes	MD33100 $MA_COMPRESS_POS_TOL
xxx1	Geometry axes	SD42475 $SC_COMPRESS_CONTUR_TOL
	Orientation axes	MD33100 $MA_COMPRESS_POS_TOL
xxx2	Geometry axes	MD33100 $MA_COMPRESS_POS_TOL
	Orientation axes	SD42476 $SC_COMPRESS_ORI_TOL SD42477 $SC_COMPRESS_ORI_ROT_TOL
xxx3	Geometry axes	SD42475 $SC_COMPRESS_CONTUR_TOL
	Orientation axes	SD42476 $SC_COMPRESS_ORI_TOL SD42477 $SC_COMPRESS_ORI_ROT_TOL
Tens digit (Compression of blocks with programmed tool orientation and / or value assignments)		
xx0x	Blocks with programmed tool orientation and/or value assignments are compressed.	
xx1x	Blocks with programmed tool orientation are compressed. Blocks with value assignments are **not** compressed.	

xx2x	Blocks with value assignments are compressed. Blocks with programmed tool orientation are **not** compressed.
xx3x	Blocks with programmed tool orientation and / or value assignments are **not** compressed.
Hundreds digit (Compression of blocks, except linear blocks (G1))	
x0xx	Circular blocks and G0 blocks are **not** compressed.
x1xx	Circular blocks are linearized and compressed by `COMPCAD`.
x2xx	G0 blocks [1] are compressed.
x3xx	Circular blocks and G0 blocks [1] are compressed.
Thousands digit (Compression for specific applications)	
0xxx	Optimization regarding the surface quality in tool and mold making.
1xxx	Optimization regarding smooth and quick traversal. Relevant for applications such as, for example, tape laying.
1) Another tolerance can be specified for compression with MD20560 $MC_G0_TOLERANCE_FACTOR or NC command `STOLF`.	

Programming

Tool orientation

If orientation transformation (TRAORI) is active, for 5-axis machines, tool orientation can be programmed in the following way (independent of the kinematics):

- Programming of the direction **vector** via:
  `A3=<...> B3=<...> C3=<...>`
- Programming of the Euler**angles** or RPY-**angles** via:
  `A2=<...> B2=<...> C2=<...>`

Rotation of the tool

For **6-axis** machines you can program the tool rotation in addition to the tool orientation.

The angle of rotation is programmed with:

```
THETA=<...>
```

Note

NC blocks, in which rotation is also programmed, can only be compressed if the angle of rotation changes **linearly**. I.e., an angle of rotation must not be programmed with a polynomial PO[THT]=(...).

General structure of a compressible NC block

The general structure of an NC block that can be compressed can therefore look like this:

```
N... X=<...> Y=<...> Z=<...> A3=<...> B3=<...> C3=<...> THETA=<...> F=<...>
```

or

```
N... X=<...> Y=<...> Z=<...> A2=<...> B2=<...> C2=<...> THETA=<...> F=<...>
```

Programming tool orientation using rotary axis positions

Tool orientation can be also specified using rotary axis positions, e.g. with the following structure:

```
N... X=<...> Y=<...> Z=<...> A=<...> B=<...> C=<...> THETA=<...> F=<...>
```

In this case, compression is executed in two different ways, depending on whether large-radius circular interpolation is executed. If no large-radius circular interpolation takes place, then the compressed change in orientation is represented in the usual way by axial polynomials for the rotary axes.

Activate compressor function

Compressor functions are activated by modal G commands `COMPON`, `COMPCURV`, `COMPCAD` or `COMPSURF`.

Deactivate compressor function

`COMPOF` terminates the compressor function.

Example

In the example program below, a circle approximated by a polygon definition is compressed. The tool orientation moves on the outside of the taper at the same time. Although the programmed orientation changes are executed one after the other, but discontinuously, the compressor function generates a smooth motion of the orientation.

```
Program code                                    Comment
DEF INT NUMBER=60
DEF REAL RADIUS=20
DEF INT COUNTER
DEF REAL ANGLE
N10 G1 X0 Y0 F5000 G64 //sort match
rate lower first


$SC_COMPRESS_CONTUR_TOL=0.05                    ; Maximum deviation of the contour = 0.05 mm
$SC_COMPRESS_ORI_TOL=5                          ; Maximum deviation of the orientation
                                                = 5 degrees


TRAORI
COMPCURV
                                                ; The movement describes a circle generated
                                                from polygons. The orientation moves on a
                                                taper around the Z axis with an opening an-
                                                gle of 45 degrees.
N100 X0 Y0 A3=0 B3=-1 C3=1
N110 FOR COUNTER=0 TO NUMBER
N120 ANGLE=360*COUNTER/NUMBER
N130 X=RADIUS*cos(angle) Y=RADIUS*sin(angle)
A3=sin(angle) B3=-cos(angle) C3=1
N140 ENDFOR
```

Further information

Function Manual Basic Functions; Continuous-path mode, exact stop, LookAhead > Compressor functions

3.9.4 Smoothing of the orientation characteristic

3.9.4.1 Function

With many of the NC programs for 5-axis machining created with CAD/CAM systems it happens that although the contour characteristic is sufficiently smooth in accordance with the underlying geometry the orientation characteristic contains fluctuations it to one extent or the other. These fluctuations in orientation result in very unsmooth running of the orientation axes with permanent acceleration and braking. The compensating motion that the linear axes then have to carry out requires that they also have to be continually accelerated and braked. Due to this unnecessary acceleration, the possible path velocity is significantly limited and consequently the machining time unnecessarily lengthened.

The "Smoothing the orientation characteristic" function can be used to smooth oscillations affecting orientation over several blocks. The aim is to achieve a smooth characteristic for both the orientation and the contour.

Preconditions

The following preconditions apply when using the function:

- The function is only available in systems with 5/6-axis transformation.
- It can only be used in conjunction with the COMPCAD compressor function.

3.9.4.2 Commissioning

Parameterization

Number of blocks

Smoothing of the orientation characteristic is carried out by means of an adjustable number of blocks:

MD28590 $MC_MM_ORISON_BLOCKS = <value>

For most applications, 10 blocks should be sufficient. The minimum value that should be entered is 4.

Note

If smoothing of the orientation characteristic is activated without sufficient block memory having been configured for it (MD28590 < 4), an alarm message will be output and the function cannot be executed.

Maximum block path length

The orientation characteristic is only smoothed in blocks whose traversing distance is shorter than the settable maximum block path length:

MD20178 $MC_ORISON_BLOCK_PATH_LIMIT

Blocks with longer traversing distances interrupt smoothing and are traversed as programmed.

Maximum tolerance

Smoothing of the orientation characteristic is carried out with the specified maximum tolerance being observed (maximum angular displacement of tool orientation in degrees):

SD42678 $SC_ORISON_TOL

Maximum path distance

The maximum distance over which smoothing is carried is the specified path distance:

SD42680 $SC_ORISON_DIST

3.9.4.3 Activating/deactivating the orientation characteristic (ORISON, ORISOF)

The "Smoothing of the orientation characteristic" is activated/deactivated in the part program using the commands of G group 61. The commands are modal.

Preconditions

- System with 5/6-axis transformation.
- Compressor function COMPCAD is active.

Syntax

```
ORISON
...
ORISOF
```

Meaning

ORISON:	Activating the orientation characteristic smoothing
ORISOF:	Deactivating the orientation characteristic smoothing

Example

```
Program code                    Comment
...
TRAORI()                        ; Activation of orientation transformation.
COMPCAD                         ; Activating the COMPCAD compressor function.
ORISON                          ; Activating orientation smoothing.
```

Program code	Comment
$SC_ORISON_TOL=1.0	; Maximum angular deviation of the tool orientation = 1.0 degrees.
G91	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
X10 A3=1 B3=0 C3=1	
X10 A3=-1 B3=0 C3=1	
...	
ORISOF	; Deactivation of orientation smoothing.
...	

The orientation is pivoted through 90 degrees on the XZ plane from -45 to +45 degrees. Due to the smoothing of the orientation characteristic the orientation is no longer able to reach the maximum angle values of -45 or +45 degrees.

3.9.5 Path-relative orientation (ORIPATH, ORIPATHS, ORIROTC)

Functionality

Irrespective of certain technological applications, the previous programming of tool orientation is improved in that the programmed relative orientation in relation to the total path is maintained. The required deviations from the ideal orientation path can be specified if, for example, a corner occurs in the contour.

Tool orientation can be modified not only via configurable machine data, but also via new language commands in the part program. In this way, it is possible to maintain the relative

orientation not only at the block end, but also throughout the entire trajectory. The desired orientation is achieved:

- By settable orientation methods with ORIPATH, specifying how interpolation is to be performed relative to the path.
- Whether the tool orientation should either always run continuously with specifiable deviations from the orientation relative to the path at a block transition, **or** whether the orientation jump should be smoothed in a dedicated, inserted intermediate block. In this case, path motion is stopped in the contour corner.
- There are two options of 6-axis transformations:
  - Like tool rotation, tool orientation is interpolated relative to the path (ORIPATH, ORIPATHS).
  - The orientation vector is programmed and interpolated in the usual manner. The rotation of the orientation vector is initiated relative to the path tangent using ORIROTC.

Note

Orientation relative to the path interpolation with ORIPATH or ORIPATHS and ORIROTC, **cannot** be used in conjunction with the function "Orientation smoothing". For this OSOF must be active in the part program. Otherwise alarm 10980 "Orientation smoothing not possible" is generated.

Deviation from the desired orientation

During the interpolation of the block, the orientation may deviate more or less from the desired relative orientation. The orientation achieved in the previous block is transferred to the programmed end orientation using large circular interpolation. The resulting deviation from the desired relative orientation has two main causes:

1. The **end orientation of the previous block** refers to the tangent and the surface normal vector at the end of the previous block. Both can differ from this at the start of the current block. Therefore, the start orientation in the current block does not have the same alignment with respect to the tangent and the surface normal vector as at the end of the previous block.
2. Not only the tangent, but also the surface normal vector can **change throughout the entire block**. This is the case, when circles, splines or polynomials are programmed for the geometry axes, or when not only a start, but also an end value is programmed for the surface normal vector. In this case, the tool orientation must change accordingly during the interpolation of the block, in order to have the same reference to the path tangent and to the surface normal vector in each path point.

Parameterization: Machine data

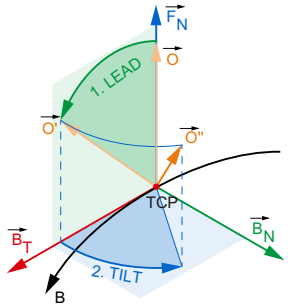
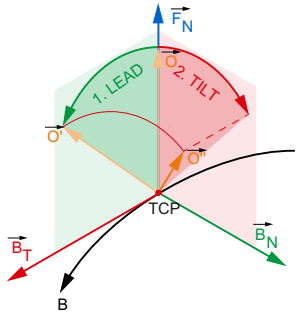
Setting for path relative orientation

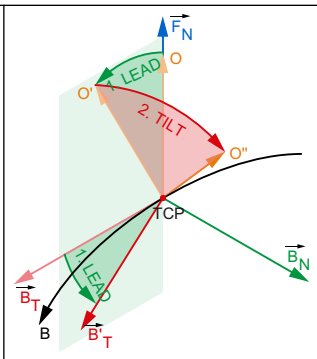
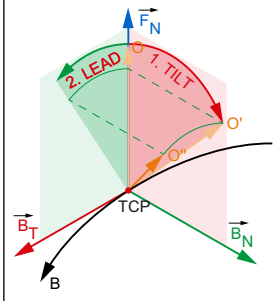
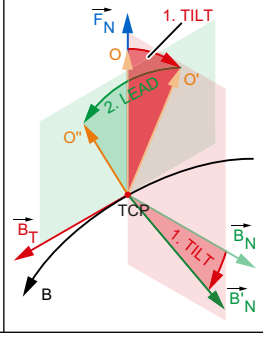
With the machine data, the following decimal-coded functions are parameterized:

- Units digit: Orientation relative to the path
- Tens digit: Interpretation of the angle of rotation LEAD and TILT

- Hundred digit: Activation and definition of the direction of the lift motion for reorientation during an active ORIPATH
- Thousands digit: Behavior of the path-relative orientation in the activation/deactivation blocks of the tool offset

MD21094 $MC_ORIPATH_MODE = <value>

<table>
<tr><th>Val-ue</th><th colspan="2">Meaning</th></tr>
<tr><td colspan="3">Units digit
(Orientation relative to the path)</td></tr>
<tr><td>xxx0</td><td colspan="2">The tool orientation only has the reference to the path tangent and to the surface normal vector programmed with LEAD and TILT, at the end of the block. During the block, the orientation does not follow the path tangent.</td></tr>
<tr><td>xxx1</td><td colspan="2">In the complete block, the tool orientation has the reference to the path tangent and surface normal vector programmed with LEAD and TILT.</td></tr>
<tr><td colspan="3">Tens digit
(Interpretation of the angles of rotation LEAD and TILT)
For path-relative orientation, the coordinate system is formed by the vectors path tangent T and programmed surface normal vector N.
Note 1: Programming the surface normal vector N
• at the block start: A4=… B4=… C4=…
• at the block end: A5=… B5=… C5=…</td></tr>
<tr><td>xx0x</td><td> 
</td><td>1st rotation LEAD: Rotation of the orientation vector O around the path normal vector $B_N \Rightarrow O'$
2nd rotation TILT: Rotation of the new orientation vector O' around the surface normal vector $F_N \Rightarrow O''$</td></tr>
<tr><td>xx1x</td><td> 
</td><td>1st rotation LEAD: Rotation of the orientation vector O around the path normal vector $B_N \Rightarrow O'$
2nd rotation TILT: Rotation of the new orientation vector O' around the path tangent vector $B_T \Rightarrow O''$</td></tr>
</table>

xx2x		1st rotation `LEAD`: Rotation of the orientation vector O and the path tangent vector B_T around the path normal vector B_N ⇒ O' and B'_T
		2nd rotation `TILT`: Rotation of the new orientation vector O' around the new path tangent vector B'_T ⇒ O''
xx3x		1st rotation `TILT`: Rotation of the new orientation vector O around the path tangent vector B_T ⇒ O'
		2nd rotation `LEAD`: Rotation of the new orientation vector O' around the path normal vector B_N ⇒ O''
xx4x		1st rotation `TILT`: Rotation of the orientation vector O and the path normal vector B_N around the path tangent vector B_T ⇒ O' and B'_N
		2nd rotation `LEAD`: Rotation of the new orientation vector O' around the new path normal vector B'_N ⇒ O''

Hundreds digit

(Activation and definition of the direction of the **lift motion** for reorientation during an active `ORIPATH`)

Note

A programmed retraction/lift motion is only executed, for:

- Once digit == 1 **AND**
- Length of the lift vector ≠ 0.0

x0xx	No lift motion is executed.
x1xx	Lift motion is executed in the direction of the programmed lift vector in the **tool**coordinate system. The lift vector refers to the coordinate system defined by the actual tool direction (z coordinate) and the orientation change (x coordinate).
x2xx	Lift motion is executed in the direction of the programmed lift vector in the **workpiece**coordinate system. The lift vector refers to the coordinate system that is defined by the active plane (z coordinate is the surface normal vector of the active plane) and the orientation change (x coordinate).

Thousands digit

(Behavior of the path-relative orientation in the activation/deactivation blocks of the tool offset)

0xxx	The path-relative orientation is also **maintained** in activation or deactivation blocks of the tool offset.
1xxx	The path-relative orientation is **not maintained** in activation or deactivation blocks of the tool offset. **Note** The tool orientation normally remains constant in the activation or deactivation blocks of the tool offset. However, in these blocks it is permitted to program a tool orientation, which is then traversed through in these blocks. However, programming the orientation in these blocks is only possible with vectors; it is not permitted to program rotary axis positions.

Address names for the lift vector components

The address names for the lift vector components are defined using machine data:

MD10624 $MN_ORIPATH_LIFT_VECTOR_TAB[<vector component>] = <name>

Example with **standard data**

Machine data

```
$MN_ORIPATH_LIFT_VECTOR_TAB[ 0 ] = "A8" ; x coordinate
$MN_ORIPATH_LIFT_VECTOR_TAB[ 1 ] = "B8" ; y coordinate
$MN_ORIPATH_LIFT_VECTOR_TAB[ 2 ] = "C8" ; z coordinate
```

Programming

```
ORIPATHS A8=X_KOORD B8=Y_KOORD C8=Z_KOORD
```

Note

Naming convention

The rules defined by MD20080 $MC_AXCONF_CHANAX_NAME_TAB should be complied with for axis identifiers.

Address names for the safety factor for orientation change

Normally the retracting movement is performed simultaneously to the orientation change. The address name for safety factor R can be defined using machine data:

MD10626 $MN_ORIPATH_LIFT_FACTOR_NAME = "<name>"

If safety factor R is greater than 0.0, then the orientation is only changed if the tool is traversed by safety clearance S in the direction of the lift vector. Safety clearance S is calculated as follows:

S = R * lift vector amount; definition range of factor R: 0 ≤ R < 1

Example with **standard data**

Machine data

```
$MN_ORIPATH_LIFT_FACTOR_NAME = "ORIPLF"
```

Programming

```
ORIPATHS A8=X_KOORD B8=Y_KOORD C8=Z_KOORD  ; lift vector
ORIPLF=0.1  ; safety clearance = 0.1 * lift vector amount (A8,
B8, C8)
```

Note

Naming convention

The rules defined by MD20080 $MC_AXCONF_CHANAX_NAME_TAB should be complied with for axis identifiers.

Parameterization: Setting data SD42670 (path section to the orientation smoothing)

Using the setting data, a path section is specified, within which, for a tool orientation step at a block transition, it is possible to deviate from the programmed orientation path in order to smooth the orientation path. If this path section is set too low, then it is possible that the path velocity must be significantly reduced.

SD42670 $SC_ORIPATH_SMOOTH_DIST = <path length>

Response for path length 0.0

If a value of 0.0 is set as path length, then a dedicated intermediate block is inserted if necessary to smooth the orientation path. The path is maintained until the orientation change was executed in the intermediate block. However, the orientation change is then only performed with **continuous acceleration** when `ORIPATHS` is active. Otherwise, from the start to the end orientation, the orientation change is realized using linear large circle interpolation.

The tool can be lifted while the orientation axes are traversed to execute the orientation change. The lift motion is activated via the hundreds digit of MD21094 $MC_ORIPATH_MODE (see the paragraph above: "Parameterization: Machine data" > "Setting for path-relevant orientation").

The direction and path length of the lift motion is defined by the programmed lift vector (see paragraph above: "Parameterization: Machine data" > "Address names for the lift vector components"). If the length of this vector is exactly zero, no retracting movement is executed.

Normally the retracting movement is performed simultaneously to the orientation change. However, a "Safety clearance" can also be programmed (see paragraph above: "Parameterization: Machine data" > "Address names for the safety factor for orientation change")

Parameterization: Setting data SD42672 (tolerance for orientation smoothing)

If just the path and the path velocity are continuous at a block transition but not the path acceleration (e.g. tangential transition, straight line/circle), then regarding the orientation, only the orientation path is continuous, not the orientation change. This results in a generally undesirable velocity step in the orientation axes. In the setting data, a tolerance window can be parameterized; within this tolerance window it is permissible that the traversed orientation path deviates from the programmed orientation path in order to smooth the orientation path.

SD42672 $SC_ORIPATH_SMOOTH_TOL = <tolerance>

This type of orientation smoothing is only executed if the following applies:

- `ORIPATHS` is active **AND**
- SD42672 $SC_ORIPATH_SMOOTH_TOL > 0.0

Path-relative interpolation of the rotation (ORIROTC)

For **6-axis transformations**, in addition to the path-relative interpolation of all of the components of the tool orientation (rotation around each of the three axes of rotation), using `ORIROTC` it is possible that only the rotation of the tool around the orientation vector relative to the path tangent is interpolated.

The orientation vector can still be programmed with direction components or via Euler or RPY angle - and the interpolation type defined using `ORIVECT`, `ORIAXES`, `ORICONxx`, `ORICURVE` (see Chapter "Rotations of orientation vector (Page 223)").

3.9.6 Programming of orientation polynominals

Function

Orientation polynomials and even axis polynomials can be programmed with different types of polynomials regardless of the type of polynomial interpolation currently active. This can be applied to:

- Linear interpolation with G command G01
- Polynomial interpolation with G command TERMINAL
- Circular interpolation with G command G02, G03 or CIP
- Involute interpolation with G command INVCW or INVCCW

This enables a number of polynomials to be programmed for one contour **at the same time**.

Note

For additional information about programming axis polynomials with PO[X], PO[Y], PO[Z] and orientation polynomials such as PO[PHI], PO[PSI], PO[THT] and PO[XH], PO[YH], PO[ZH], see:
Further information
Programming Manual Production Planning

Two different types of orientation polynomials are defined:

- Polynomials for angles with reference to the plane defined by the start and end orientation (type 1 orientation polynomials)
- Polynomials for coordinates in space of a reference point on the tool (type 2 orientation polynomials).

Type 1 polynomials

Orientation polynomials of type 1 are polynomials **for angles**

PO[PHI]:	Angle in the plane between start and end orientation
PO[PSI]:	Angle describing the tilt of the orientation from the plane between start and end orientation

Type 2 polynomials

Orientation polynomials of type 2 are polynomials **for coordinates**

PO[XH]:	x coordinate of the reference point on the tool
PO[YH]:	y coordinate of the reference point on the tool
PO[ZH]:	z coordinate of the reference point on the tool

Polynomials for angle of rotation and rotation vectors

For **6-axis transformations**, the rotation of the tool around itself can be programmed for tool orientation. This rotation of a third rotary axis is described either by an angle of rotation or by a rotation vector, which is perpendicular to the tool direction in the plane.

In addition, a polynomial **for rotation** with PO[THT} of the orientation vector can be programmed in these three cases. This is always possible if the kinematic transformation applied, supports rotary angles.

Angle of rotation with ORIPATH and ORIPATHS

With **orientation interpolation relative to the path** with ORIPATH or ORIPATHS, the additional rotation can be programmed with the angle `THETA=<...>`. In addition, for this angle of rotation, with `PO[THT]=(...)` As a maximum, 5th degree polynomial angles can be programmed.

The 3 possible angles, i.e. lead angle, tilt angle and angle of rotation, have the following meaning with respect to the rotation effect:

LEAD:	Angle relative to the surface normal vector in the plane put up by the path tangent and the surface normal vector.
TILT:	Rotation of orientation in the z direction or rotation about the path tangent.
THE-TA:	Rotation around the tool direction. This is only possible when the tool orientation has a total of 3 degrees of freedom (see Section "Extension of the generic transformation to 6 axes (Page 179)").

How the angles LEAD and TILT are to be interpreted, can be set with the following machine data:

MD21094 $MC_ORIPATH_MODE (setting for path relative orientation ORIPATH)

In addition to the constant angles programmed with `LEAD` and `TILT`, polynomials can be programmed for lead angle and tilt angle. The polynomials are programmed using the angles `PHI` and `PSI`:

`PO[PHI]=(a2,a3,a4,a5)`:	Polynomial for the LEAD angle
`PO[PSI]=(b2,b3,b4,b5)`:	Polynomial for the TILT angle

For both angles, as a maximum, 5th degree polynomial angles can be programmed. The angle values at the block end are programmed with the NC addresses `LEAD=<...>` or `TILT=<...>`.

The higher polynomial coefficients, which are zero, can be omitted when programming. For example **PO[PHI] = (a2)** programs a parabola for the lead angle `LEAD`.

Rotations of rotation vectors with ORIROTC

The rotation vector is interpolated relative to the path tangent with an offset that can be programmed using the THETA angle.

For the offset angle, with `PO[THT]=(c2,c3,c4,c5)`, as a maximum, a 5th degree polynomial angle can be programmed.

Note

If ORIAXES is active, i.e. the tool orientation is interpolated via axis interpolation, the orientation of the rotation vector relative to the path is only fulfilled at the end of the block.

For further information about programming, see:
Further information
Programming Manual Production Planning; Transformations, Interpolation type (ORIPATH, ORIPATHS)

Boundary conditions

It is only useful to program orientation polynomials for specific interpolation types, which affect both contour and orientation. A number of supplementary conditions must be met to avoid illegal programming settings:

Orientation polynomials cannot be programmed,

- if ASPLINE, BSPLINE, CSPLINE spline interpolations are active.
  Polynomials for type 1 orientation angles are possible for every type of interpolation except spline interpolation, i.e. linear interpolation with rapid traverse G00 or with feedrate G01 and polynomial POLY and circular/involute interpolation G02, G03, CIP, CT, INVCW and INVCCW.
  In contrast, type 2 orientation polynomials are only possible if linear interpolation with rapid traverse G00 or with feedrate G01 or polynomial interpolation POLY is active.
- if the orientation is interpolated using ORIAXES axis interpolation.
  In this case, polynomials can be programmed directly with PO[A] and PO[B] for orientation axes A and B.

If `ORICURVE` is active, the Cartesian components of the orientation vector are interpolated and only type 2 orientation polynomials are possible. However, type 1 orientation polynomials are not permitted.

Only type 1 orientation polynomials are possible for large circle interpolation and taper interpolation with ORIVECT, ORIPLANE, ORICONxxx. However, type 2 orientation polynomials are not permitted.

Alarms

An illegally programmed polynomial is signaled with the following alarms:

Alarm 14136: Orientation polynomial is generally not allowed.
Alarm 14137: Polynomials PO[PHI] and PO[PSI] are not permitted.
Alarm 14138: Polynomials PO[XH], PO[YH], PO[ZH] are not permitted.
Alarm 14139: Polynomial for angle of rotation PO[THT] is not permitted.

3.9.7 System variable for tool orientation

The tool orientation can be read in various coordinate systems (BCS, WCS, SZS) via system variables as well as also via OPI variables.

Tool orientation in BCS

System variable		Meaning	
$AC_TOOLO_ACT[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the actual reference orientation	
$AC_TOOLO_END[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the end orientation of the actual block	
$AC_TOOLO_DIFF		Residual angle in degrees, i.e. this is the angle between vectors $AC_TOOLO_END[<i>] and $AC_TOOLO_ACT[<i>]	
$VC_TOOLO[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the actual orientation	
$VC_TOOLO_DIFF		Angle in degrees between reference and actual value orientation	
$VC_TOOLO_STAT		Status variable for actual orientation Indicates whether the actual orientation can be calculated. The following values are possible:	
		0	Actual orientation can be calculated.
		-1	Actual orientation cannot be calculated, as the presently active transformation cannot calculate these values in real time.

These system variables can always be read by the part program as well as in synchronized actions. Write access operations are not permitted.

Note

The components of vectors $AC_TOOLO_ACT[<i>], $AC_TOOLO_END[<i>] and $VC_TOOLO[<i>] of the orientation are scaled so that the orientation vector has the absolute value of 1.

Rotation vector in the BCS

For a 6-axis kinematics, in addition to the orientation of the tool, there is also a rotation of the tool, which can be changed.

System variable		Meaning	
$P_TOOLROT[<i>]	; <i> = 1, 2, 3	i-th component of the actual rotation vector in the NC program	
$AC_TOOLR_ACT[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the actual setpoint of the orientation rotation	
$AC_TOOLR_END[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the setpoint of the rotation at the end of the actual block	
$AC_TOOLR_DIFF		Residual angle in degrees, i.e. this is the angle between vectors $AC_TOOLR_END[<i>] and $AC_TOOLR_ACT[<i>]	
$VC_TOOLR[<i>]	; <i> = 1, 2, 3	i-th component of the vector of the actual value of the orientation rotation	
$VC_TOOLR_DIFF		Angle in degrees between the setpoint and actual value of the orientation rotation	
$VC_TOOLR_STAT		Status variable for the actual value of the rotation vector Indicates whether the actual value of the rotation vector can be calculated. The following values are possible:	
		0	The actual value of the rotation vector can be calculated.
		-1	Actual value of the rotation vector cannot be calculated, as the presently active transformation cannot calculate these values in real time.

Orientation and rotation of the tool in the various coordinate systems (BCS, WCS, SZS)

Tool orientation

System variable		Meaning	
$P_TOOL_O[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the actual orientation vector in the NC program in the coordinate system <j>	
		<j> = 0:	BCS
		<j> = 1:	WCS
		<j> = 2:	SZS
$AC_TOOL_O_ACT[<i>,< j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the actual orientation vector in the coordinate system <j>	
$AC_TOOL_O_END[<i>,< j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the end orientation of the actual block in the coordinate system <j>	
$AC_TOOL_O_DIFF[<j>]	; <j> = 0, 1, 2	Residual angle of the orientation vector in degrees in various coordinate systems <j>	

System variable		Meaning
$VC_TOOL_O[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the vector of the actual orientation in various coordinate systems <j>
$VC_TOOL_O_DIFF[<j>]	; <j> = 0, 1, 2	Angle in degrees between reference and actual value orientation in various coordinate systems <j>

Rotation vector (only for 6-axis kinematics)

System variable			
$P_TOOL_R[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the actual rotation vector in the NC program in the coordinate system <j>	
		<j> = 0:	BCS
		<j> = 1:	WCS
		<j> = 2:	SZS
$AC_TOOL_R_ACT[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the actual rotation vector in the coordinate system <j>	
$AC_TOOL_R_END[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the rotation vector at the end of the actual block in the coordinate system <j>	
$AC_TOOL_R_DIFF[<j>]	; <j> = 0, 1, 2	Residual angle of the rotation vector in degrees in various coordinate systems <j>	
$VC_TOOL_R[<i>,<j>]	; <i> =1, 2, 3 ; <j> = 0, 1, 2	i-th component of the actual value of the rotation vector in various coordinate systems <j>	
$VC_TOOL_R_DIFF[<j>]	; <j> = 0, 1, 2	Angle in degrees between reference and actual value of the rotation vector in various coordinate systems <j>	

Boundary conditions

Not all transformations provide the actual value of the tool orientation in real time. In this case, variables $VC_TOOLO[<i>] or VC_TOOL_O[<i>] and $VC_TOOLO_DIFF or $VC_TOOL_O_DIFF cannot be calculated. The components of $VC_TOOLO[<i>] or $VC_TOOL_O[<i>] are all zero, and status variable $VC_TOOLO_STAT supplies the value "-1". The same is true for the actual values of the rotation vector $VC_TOOLR[<i>] or $VC_TOOL_R[<i>,<j>].

3.10 Orientation axes

3.10.1 Orientation axes

Directions of rotation

The directions around which axes are rotated are defined by the axes of the reference system. In turn, the reference system is defined by ORIMKS and ORIWKS commands:

- ORIMKS : Reference system = Basic coordinate system
- ORIWKS: Reference system = Workpiece coordinate system

Order of rotation

The order of rotation for the orientation axes is defined by the following machine data:

MD21120 $MC_ORIAX_TURN_TAB_1[0..2] (definition of reference axes for ORI axes)

1. First rotation around the axis of the reference system, defined in the following machine data: MD21120 $MC_ORIAX_TURN_TAB_1[0]
2. Second rotation around the axis of the reference system, defined in the following machine data: MD21120 $MC_ORIAX_TURN_TAB_1[1]
3. Third rotation around the axis of the reference system, defined in the following machine data: MD21120 $MC_ORIAX_TURN_TAB_1[2]

Direction of the tool vector

The direction of the tool vector in the initial machine setting is defined in the following machine data:

MD24580 $MC_TRAFO5_TOOL_VECTOR_1 (orientation vector direction) or

MD24680 $MC_TRAFO5_TOOL_VECTOR_2 (orientation vector direction)

Assignment to channel axes

Machine data MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[0..2] (ORI/channel assignment Transformation 1) are used to assign up to a total of 3 virtual orientation axes to the channel, which are set as input variables in machine data $MC_TRAFO_AXES_IN_n[4..6] (axis assignment for Transformation n).

For assigning channel axes to orientation axes, the following applies:

- $MC_TRAFO5_ORIAX_ASSIGN_TAB_n[0] = $MC_TRAFO_AXES_IN_n[4]
- $MC_TRAFO5_ORIAX_ASSIGN_TAB_n[1] = $MC_TRAFO_AXES_IN_n[5]
- $MC_TRAFO5_ORIAX_ASSIGN_TAB_n[2] = $MC_TRAFO_AXES_IN_n[6]

Orientation transformation 1:	
MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[n]	n = channel axis [0..2]
Orientation transformation 2:	
MD24685 $MC_TRAFO5_ORIAX_ASSIGN_TAB_2[n]	n = channel axis [0..2]
transformation [1..4]	
MD24110 $MC_TRAFO5_AXES_IN_1[n] (axis assignment for transformation)	n = channel axis [0..7]
to	
MD24410 $MC_TRAFO5_AXES_IN_4[n] (axis assignment for transformation 4)	
transformation [5..8]	
MD24432 $MC_TRAFO5_AXES_IN_5[n] (axis assignment for transformation 5)	n = channel axis [0..7]
to	
MD24462 MC_TRAFO5_AXES_IN_8[n] (axis assignment for transformation 8)	

Example

For orientation axes, see Chapter "Example for orientation axes (Page 240)".

3.10.2 JOG mode

Preconditions

Orientation axes can only be traversed in the jog operating mode when the following preconditions are satisfied:

- The orientation axis must be defined as such, that is, a value must be set in the following machine data:
  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB (ORI/channel axis assignment Transformation 1)
- A transformation must be active (TRAORI).

Traversing using traverse keys

When using the traverse keys to move an axis continuously or incrementally, it must be noted that only one orientation axis can be moved at a time.

Alarm 20062 "Channel 1 axis 2 already active" is output if more than one orientation axis is traversed.

Traversing using handwheels

More than one orientation axis can be moved simultaneously using handwheels.

Velocity

When orientation axes are traversed manually, the channel-specific feedrate override switch or the rapid traverse override switch in rapid traverse override applies.

Normally, velocities when traversing in the jog mode have always been derived from the machine axis velocities. However, for geometry and orientation axes, there is not necessarily a direct assignment to a particular machine axis. This is the reason that dedicated machine data exist for geometry axes and orientation axes, which allow velocities to be separately specified:

- MD21150 $MC_JOG_VELO_RAPID_ORI[n] (conventional rapid traverse for ORI axes)
- MD21155 $MC_JOG_VELO_ORI[n] (conventional ORI axis velocity)
- MD21160 $MC_JOG_VELO_RAPID_GEO[n] (conventional rapid traverse for GEO axes)
- MD21165 $MC_JOG_VELO_GEO[n] (conventional GEO axis velocity)

Acceleration

The acceleration for orientation axes is set in machine data:

MD21170 $MC_ACCEL_ORI[n] (acceleration for orientation axes)

3.10.3 Programming for orientation transformation

The values can only be programmed in conjunction with an orientation transformation.

Programming of orientation

Orientation axes are programmed by means of axis names **A2, B2** and **C2**.

Euler and RPY values are distinguished on the basis of G-group 50:

- `ORIEULER`:
  Orientation programming on the basis of Euler angles (default)
- `ORIRPY`:
  Orientation programming via RPY angles
- `ORIVIRT1`:
  Orientation programming on the basis of virtual orientation axes (definition 1)
- `ORIVIRT2`:
  Orientation programming on the basis of virtual orientation axes (definition 2)

The type of interpolation is distinguished on the basis of G-group 51:

- `ORIAXES`:
  Orientation programming of linear interpolation of orientation axes or machine axes
- `ORIVECT`:
  Orientation programming of large circle interpolation of orientation axes (interpolation of the orientation vector)

Machine data MD21102 $MC_ORI_DEF_WITH_G_CODE (definition of ORI axes via G command) is used to specify whether MD21100 $MC_ORIENTATION_IS_EULER (angle definition for orientation programming) is active (default) or G group 50.

The following four variants are available for programming orientation:

1. `A`, `B`, `C`:
   Machine axis parameter designation
2. `A2`, `B2`, `C2`:
   Angle programming of virtual axes
3. `A3`, `B3`, `C3`:
   Vector component designation
4. `LEAD`, `TILT`:
   Specification of lead and side angles with reference to path and surface

Further information
Programming Manual Fundamentals

Note

The four variants of orientation programming are mutually exclusive. If mixed values are programmed, alarm 14130 or alarm 14131 is generated.

Exception:

For 6 axis kinematics with a 3rd degree of freedom for orientation, C2 may also be programmed for variants 3 and 4. C2 in this case describes the rotation of the orientation vector about its own axis.

Example

For orientation axes for kinematics with 6 or 5 transformed axes, see Section "Example for orientation axes (Page 240)".

Interpolation type

The following machine data is used to specify which interpolation type is used:

MD21104 $MC_ORI_IPO_WITH_G_CODE (G command for orientation interpolation):

- ORIMKS or ORIWKS
- G group 51 with the commands `ORIAXES` or `ORIVECT`
  - `ORIAXES`:
    Linear interpolation of machine axes or orientation axes.
  - `ORIVECT`:
    Orientation is controlled by the orientation vector being swiveled in the plane spanned by the start and end vectors (large circle interpolation). With 6 transformation axes a rotation around the orientation vector is executed in addition to the swivel movement. With `ORIVECT` the orientation axes are always traversed on the shortest possible path.

Range of values

Value range for orientation axes:

- 180 degrees < A2 < 180 degrees
- 90 degrees < B2 < 90 degrees
- 180 degrees < C2 < 180 degrees

All possible rotations can be represented with this value range. Values outside the range are normalized by the control system to within the range specified above.

Feedrate when programming ORIAXES

Feedrate for an orientation axis can be limited via the FL[ ] instruction (feed limit).

3.10.4 Programmable offset for orientation axes

How the programmable offset works

The additional programmable offset for orientation axes acts in addition to the existing offset and is specified when transformation is activated. Once transformation has been activated, it is no longer possible to change this additive offset and no zero offset will be applied to the orientation axes in the event of an orientation transformation.

The programmable offset can be specified in two ways.

1. Direct programming of the offset with `TRAORI()` when transformation is activated.
2. Automatic transfer of the offset from the zero offset active for the orientation axes when transformation is activated. This automatic transfer is configured via machine data.

Programming offset directly

When transformation is activated, the offset can be programmed directly as TRAORI(n, x, y, z, a, b) `TRAORI(n, x, y, z, a, b)` . The following parameters are available as an option:

n:	Number of transformation n = 1 or 2
x, y, z	Components of the vector for the basic orientation of the tool (generic 5-axis transformation only).
a, b:	Offset for rotary axes

These optional parameters can be omitted. However, if they are used for programming purposes, the correct sequence must be observed. If for example only one rotary axis offset is to be entered, `TRAORI(,,,, a, b)` is to be programmed.

For further information about programming, please see:

Further information
Programming Manual Advanced; Transformations

Programming offset automatically

As the offset is transferred automatically from the currently active zero offset on the orientation axes, the effects of zero offset on rotary axes are always the same, both with and without active transformation. Automatic take-over of offset from zero offset is possible via machine data MD24590 $MC_TRAFO5_ROT_OFFSET_FROM_FR_1 = TRUE (Offset of rotary transformation axes from NPV) for the first machine data, or MD24690 $MC_TRAFO5_ROT_OFFSET_FROM_FR_2 = TRUE (Offset of rotary transformation axes from NPV) for the second transformation in the channel.

Note

There is no difference between a zero offset on the orientation axes programmed during active transformation and the previous offset.

If automatic transfer of offset has been activated and a rotary axis offset is programmed at the same time, the programmed offset value takes priority.

Orientable toolholder with additive offset

On an orientable toolholder, the offset for both rotary axes can be programmed with the system variables $TC_CARR24 and $TC_CARR25. This rotary axis offset can be transferred automatically from the zero offset effective at the time the orientable toolholder was activated.

Automatic transfer of offset from zero offset is made possible via the following machine data:

MD21186 $MC_TOCARR_ROT_OFFSET_FROM_FR = TRUE (offset of TOCARR rotary axes from NPV)

Note

For more information about orientable toolholders, please see:

Further information
Function Manual Tools; Tool offset

3.10.5 Orientation transformation and orientable tool holders

Note

Orientation transformation and orientable tool holders can becombined.

The resulting orientation of the tool is produced by linking the orientation transformation and the orientable tool holder.

3.10.6 Modulo display of orientation axes

Function

The positions of orientation axes can be displayed for the BCS and WCS display in a settable modulo area. Whether the machine axes are linear or rotary is not relevant in this context. This means that this display option can be enabled even for normal generic 5/6-axis transformation.

Requirements

- Orientation axes must be available. This is the case if an orientation transformation is active (e.g. generic 5-/6-axis transformation).
- The following machine data must also be set for OEM transformations:
  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[0..2]

Parameterization

The modulo display of orientation axes is activated as follows:
MD21132 $MC_ORI_DISP_IS_MODULO[0...2] = TRUE

The modulo range is defined with the help of the following machine data:

- MD21134 $MC_ORI_MODULO_RANGE[0...2]
  (Size of the modulo range for the display of the orientation axes)
- MD21136 $MC_ORI_MODULO_RANGE_START[0...2]
  (Starting position of the modulo range for the display of the orientation axes)

Please note the following:

- The machine data become effective with NEWCONF.
- The machine data does not have any influence or effects on:
  - any axis positions that can be programmed for these axes.
  - The traversing movements of these axes.
  - The display of the MCS values of these axes

3.11 Orientation vectors

3.11.1 Polynomial interpolation of orientation vectors

Programming of polynomials for axis motions

The rotary axes are normally subjected to linear interpolation in case of orientation changes with the help of rotary axis interpolation. However, it is also possible to program the polynomials as usual for the rotary axes. This allows a generally more homogeneous axis motion to be produced.

Note

Further information about programming polynomial interpolation with POLY and on interpolation of orientation vectors is given in:

Further information
Programming Manual; Advanced

A block with POLY is used to program polynomial interpolation. Whether the programmed polynomials are then interpolated as polynomial, depends on whether the G command POLY is active or not:

- The G command is **not active**: The programmed axis end points are traversed linearly.
- The G command is **active**: The programmed polynomials are interpolated as polynomials.

MD10674

Using machine data MD10674 $MN_PO_WITHOUT_POLY = **FALSE**
(polynomial can be programmed without G command POLY), it can be set as to whether the following programming is possible:

- PO[...] or PO(...) is possible only if POLY is active, **or**
- PO[ ] or PO( ) polynomials are also possible without active G command POLY.

As default, MD10674: PO_WITHOUT_POLY = FALSE set and with MD10674 $MN_PO_WITHOUT_POLY = **TRUE** the following programming is always possible:

- PO[...] = (...), regardless of whether POLY is active or not.

Orientation polynomials can be programmed in conjunction with different interpolation types and are described in Section "Programming of Orientation Polynomials".

POLYPATH:

In addition to the modal G command POLY, with the predefined subprogram POLYPATH(argument), polynomial interpolation can be selectively activated for various axis groups. The following arguments are allowed for the activation of polynomial interpolation

("AXES"):	For all path axes and supplementary axes
("VECT"):	For orientation axes
("AXES", "VECT"):	For path axes, supplementary axes and orientation axes
(without argument):	**deactivates** polynomial interpolation for all axis groups

Normally, the polynomial interpolation is activated for all axis groups.

Programming of orientation vectors

An orientation vector can be programmed in each block. If polynomials are programmed for the orientation, the interpolated orientation vector is generally turned not in the plane between start and end vector, but it can be rotated at random from this plane.

Orientation vectors can be programmed as follows:

1. Programming of rotary axis positions with A, B and C or with the actual rotary axis names.
2. Programming in Euler angle or RPY angle via A2, B2, C2
3. Programming of the direction vector via A3, B3, C3.
4. Programming using lead angle `LEAD` and tilt angle `TILT`.

Selection of type of interpolation

The type of interpolation of orientation axes is selected with G command of group 51 and is independent of the programming type of the end vector:

- `ORIAXES`: Linear interpolation of the machine axes or using polynomials for active POLY or
- `ORIVECT`: Interpolation of the orientation vector using large circle interpolation

If `ORIAXES` is active, the interpolation of the rotary axis can also take place using polynomials like polynomial interpolation of axes with `POLY` .

On the other hand, if `ORIVECT` is active, "normal" large circle interpolation is carried out through linear interpolation of the angle of the orientation vector in the plane that is defined by the start and end vector.

Polynomials for 2 angles

Additional programming of polynomials for 2 angles that span the start vector and end vector can also be programmed as complex changes in orientation with `ORIVECT` .

The two PHI and PSI angles are specified in degrees.

POLY	Activation of polynomial interpolation for all axis groups.
POLYPATH ( )	Activation of polynomial interpolation for all axis groups. "AXES" and "VECT" are possible groups.

The coefficients a_n and b_n are specified in degrees.

PO[PHI]=(a_2, a_3, a_4, a_5)	The angle PHI is interpolated according to PHI(u) = $a_0 + a_1*u + a_2*u^2 + a_3*u^3 + a_4*u^4 + a_5*u^5$.
PO[PSI]=(b_2, b_3, b_4, b_5)	The angle PHI is interpolated according to PSI(u) = $b_0 + b_1*u + b_2*u^2 + b_3*u^3 + b_4*u^4 + b_5*u^5$.
PL	Length of the parameter interval where polynomials are defined. The interval always starts at 0. **Theoretical value range** for PL: 0.0001 ... 99999.9999 The PL value applies to the block that contains it. PL = 1 is applied if no PL value is programmed.

Rotation of the orientation vector

Changes in orientation are possible with `ORIVECT` independent of the type of end vector programming. The following situations apply:

Example 1: Components of end vectors are programmed directly.

N... POLY A3=a B3=b C3=c PO[PHI] = (a2, a3, a4, a5) PO[PSI] = (b2, b3, b4, b5)

Example 2: The end vector is determined by the position of the rotary axes.

N... POLY Aa Bb Cc PO[PHI] = (a2, a3, a4, a5) PO[PSI] = (b2, b3, b4, b5)

The angle PHI describes the rotation of the orientation vector in the plane between start and end vectors (large circle interpolation, see figure below). The orientation is interpolated exactly as in Example 1.

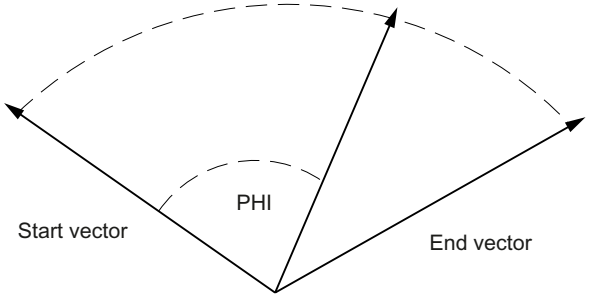


Figure 3-23 Rotation of the orientation vector in the plane between start and end vector

PHI and PSI angle

Programming of polynomials for the two angles PO[PHI] and PO[PSI] is always possible. Whether the programmed polynomials are actually interpolated for PHI and PSI depends on:

- `POLYPATH("VECT")` and `ORIVECT` are **active**, then the polynomials will be interpolated.
- If `POLYPATH("VECT")` and `ORIVECT` are **not active**, the programmed orientation vectors are traversed at the end of the block by a "normal" large circle interpolation. This means that the polynomials for the two angles PHI and PSI are ignored in this case.

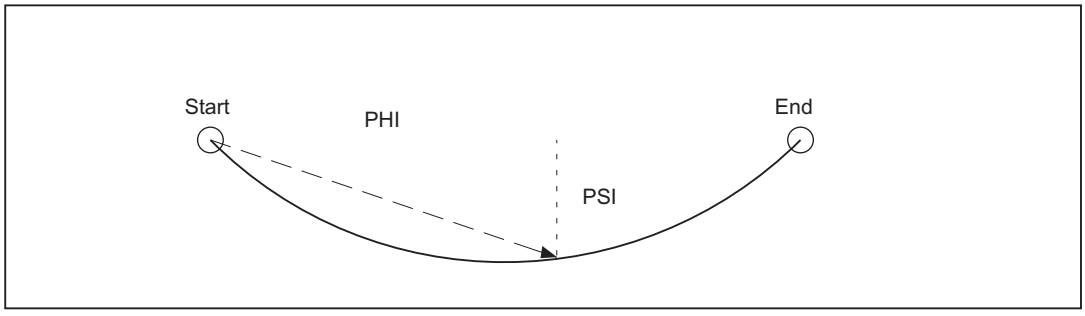


Figure 3-24 Movement of the orientation vector in plan view

The angle PSI can be used to generate movements of the orientation vector perpendicular to large circle interpolation plane (see previous figure).

Maximum polynomials of the 5th degree permitted

As a maximum, 5th degree polynomials can be programmed for the PHI and PSI angles. Here the constants and linear coefficient are defined by the initial value or end value of the orientation vector.

Higher degree coefficients can be omitted from the coefficient list (..., ...) if these are all equal to zero.

The length of the parameter interval in which the polynomials are defined can also be programmed with PL.

Special situations

If **no polynomial for angle PSI** is programmed, the orientation vector is always interpolated in the plane defined by the start and end vector.

The PHI angle in this plane is interpolated according to the programmed polynomial for PHI. As a result the orientation vector moves through a "normal" large circle interpolation in the plane between the start and end vector and the movement is more or less irregular depending on the programmed polynomial.

In this way, the velocity and acceleration curve of the orientation axes can be influenced within a block, for example.

Note

Further information on polynomial interpolation for axis motion and general programming is given in:

Further information
Programming Manual; Advanced

Boundary conditions

Polynomial interpolation of orientation vectors is only possible for control variants in which the following functions are included in the functional scope:

- Orientation transformation
- Polynomial interpolation

3.11.2 Rotations of orientation vector

Functionality

Changes in tool orientation are programmed by specifying an orientation vector in each block, which is to be reached at the end of the block. The end orientation of each block can be programmed in the following way:

1. Programming the vector directly, or
2. Programming the rotary axis positions.

The second option depends on the machine kinematics. Interpolation of the orientation vector between the start and end values can also be modified by programming polynomials.

Programming of orientation directions

The following options are available for programming tool orientation:

1. Direct programming of rotary axis positions (the orientation vector is derived from machine kinematics).
2. Programming in Euler angles via A2, B2, C2 (angle C2 is irrelevant).
3. Programming in RPY angles via A2, B2, C2.
4. Programming the direction vector via A3, B3, C3 (the length of the vector is irrelevant).

Switching between Euler and RPY angle programming can be selected via the following machine data or via the G commands `ORIEULER` and `ORIRPY` :

MD21100 $MC_ORIENTATION_IS_EULER (angle definition for orientation programming)

Programming of orientation direction and rotation

While the direction of rotation is already defined when you program the orientation with RPY angles, additional parameters are needed to specify the direction of rotation for the other orientations:

1. Direct programming of the rotary axes
   A supplementary rotary axis for direction of rotation has to be defined.
2. Programming in Euler angles via A2, B2, C2
   Angle C2 must be programmed additionally. The complete orientation is thus defined, including tool rotation.
3. Programming in RPY angles via A2, B2, C2
   Additional settings are not required.
4. Programming of the direction vector via A3, B3, C3
   The rotation angle is programmed with `THETA=<value>`.

Note

The following cases do not allow for a programmed rotation:

Multiple programming of the direction of rotation is not allowed and results in an alarm. If the Euler angle C2 and the angle of rotation `THETA` are programmed simultaneously, the programmed rotation is not executed.

If machine kinematics are such that the tool cannot be rotated, any programmed rotation is ignored. This is the case with a normal 5-axis machine, for example.

Rotation of the orientation vector

The following options are available for interpolating rotation of the orientation vector by programming the vector directly:

- Linear interpolation, i.e. the angle between the current rotation vector and the start vector is a linear function of the path parameter.
- Non-linear due to additional programming of a polynomial for the angle of rotation q of maximum 5th degree, in the form:
  PO[THT] = (d_2, d_3, d_4, d_5)

Interpolation of the angle of rotation

Higher degree coefficients can be omitted from the coefficient list (..., ...) if these are all equal to zero.

In such cases, the end value of the angle and the constant and linear coefficient d_n of the polynomial cannot be directly programmed.

Linear coefficient d_n is defined by the end angle q_e, and is specified in degrees.

End angle q_e is derived from the programming of the rotation vector.

Start angle q_s is defined from the start value of the rotation vector, resulting from the end value of the previous block. The constant coefficient of the polynomial is defined by the start angle of the polynomial.

The rotation vector is always perpendicular to the actual tool orientation and forms the angle THETA in conjunction with the basic rotation vector.

Note

When configuring the machine, the direction in space in which the rotation vector points at a specific angle of rotation can be defined, when the tool is in the basic orientation.

Formula

In general, the angle of rotation is interpolated with a 5th degree polynomial:

$$\theta u=\theta_s+d_1u+d_2u^2+d_3u^3+d_4u^4+d_5u^5 \quad (14)$$

For parameter interval 0 ... 1, this produces the following values for linear coefficients:

$$d_1=\theta_e-\theta_s-d_2-d_3-d_4-d_5 \quad (15)$$

Interpolation of the rotation vector

The programmed rotation vector can be interpolated in the following way, using modal G commands:

- ORIROTA (**ori**entation **rot**ation **a**bsolute):
  The angle of rotation THETA is interpreted with reference to an absolute direction in space. The basic direction of rotation is defined by machine data.
- ORIROTR (**ori**entation **rot**ation **r**elative):
  Angle of rotation THETA is interpreted relative to the plane defined by the start and end orientation.
- ORIROTT (**ori**entation **rot**ation **t**angential):
  Angle of rotation THETA is interpreted relative to the change in orientation. This means the rotation vector interpolation is tangential to the change in orientation for THETA=0.
  This is only different to ORIROTR, if the change in orientation does not take place in one plane. This is the case if, for the orientation, at least one polynomial was programmed for "tilt angle" PSI. An additionally programmed angle of rotation THETA can then be used to interpolate the rotation vector such that it always forms a specific angle referred to the change in orientation.

Activating rotation

A rotation of the orientation vector is programmed with identifier THETA. The following programming options are available:

THETA=<value>:	Programming an angle of rotation that is reached at the end of the block.
THETA = q_e	Programmed angle q_e can be interpreted as absolute angle (G90 is active) - as well as also relative angle (G91 is active incremental dimension).
THETA = AC(...)	Non-modal switchover to absolute dimensions

`THETA = IC(...)`	Non-modal switchover to incremental dimensions
`PO[THT] = (...)`	Programming a polynomial for rotation angle `THETA`.

Angle `THETA` is programmed in degrees.

Interpolation of the rotation vector is defined by modal G commands:

`ORIROTA`	Angle of rotation to an absolute direction of rotation
`ORIROTR`	Angle of rotation relative to the plane between the start and end orientation
`ORIROTT`	Angle of rotation relative to the change of the orientation vector tangential rotation vector to the orientation change
`ORIROTC`	Angle of rotation relative to the change of the orientation vector tangential rotation vector to the path tangent
`PL`	Length of the parameter interval on which the polynomials are defined. The interval always starts at 0. PL = 1 is active, if no PL value is programmed.

These G commands define the reference direction of the angle of rotation. The meaning of the programmed angle of rotation changes accordingly.

Supplementary conditions

The angle of rotation or rotation vector can only be programmed in all four modes if the interpolation type `ORIROTA` is active.

1. Rotary axis positions
2. Euler angle via `A2, B2, C2`
3. RPY angle via `A2, B2, C2`
4. Direction vector via `A3, B3, C3`

If `ORIROTR` or `ORIROTT` is active, the angle of rotation can only be programmed directly with `THETA`.

The other programming options must be excluded in this case, since the definition of an absolute direction of rotation conflicts with the interpretation of the angle of rotation in these cases. Possible programming combinations are monitored and an alarm is output if applicable.

A rotation can also be programmed in a separate block without an orientation change taking place. In this case, `ORIROTR` and `ORIROTT` are irrelevant. In this case, the angle of rotation is always interpreted with reference to the absolute direction (`ORIROTA`).

A programmable rotation of the orientation vector is only possible when an orientation transformation (`TRAORI`) is active.

A programmed orientation rotation is only interpolated if the machine kinematics allow rotation of the tool orientation (e.g. 6-axis machines).

3.11.3 Extended interpolation of orientation axes

Functionality

To execute a change in orientation along the peripheral surface of a cone located in space, it is necessary to perform an extended interpolation of the orientation vector. The vector around which the tool orientation is to be rotated must be known. The start and end orientation must also be specified. The start orientation is given by the previous block and the en orientation must either be programmed or defined by other conditions.

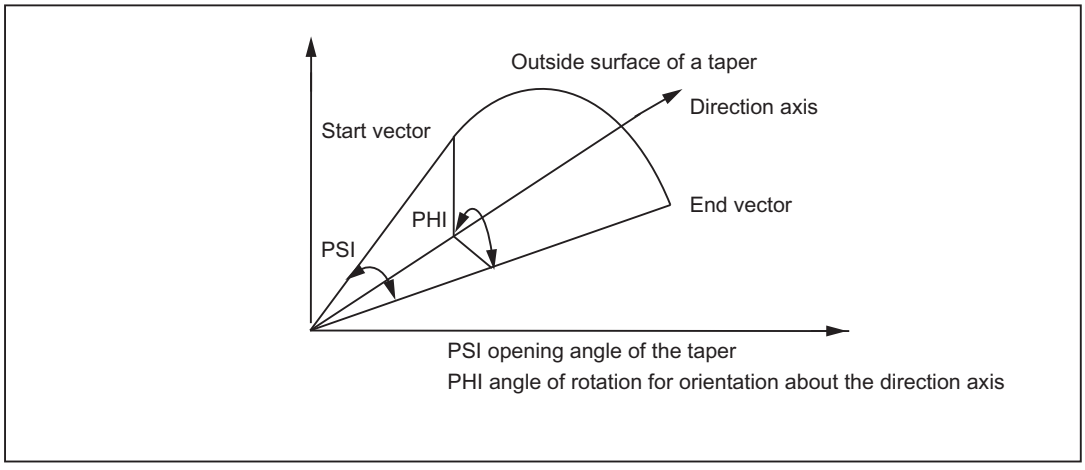


Figure 3-25 Change in orientation of the peripheral surface of a cone located in space

Required definitions

Generally, the following data is required:

- The **start orientation** is defined by the end orientation of the previous block.
- The **end orientation** is defined either by specifying the vector (with A3, B3, C3), the Euler angles or RPY angles (with A2, B2, C2) or by programming the positions of the rotary axis (with A, B, C).
- The **rotary axis of the taper** is programmed as a (normalized) vector with A6, B6, C6.
- The **opening angle of the cone** is programmed degrees with the identifier (**nut**ation angle). The **value range** of this angle is limited to the interval between 0 degrees and 180 degrees. The values 0 degrees and 180 degrees must not be programmed. If an angle is programmed outside the valid interval, an alarm is generated.
  In the special case where NUT = 90 degrees, the orientation vector in the plane is interpolated perpendicular to the direction vector (large circle interpolation).
  The sign of the programmed opening angle specifies whether the traversing angle is to be greater or less than 180 degrees.
  In order to define the cone, the **direction vector** or its **opening angle** must be programmed. Both may not be specified at the same time.
- A further option is to program an **intermediate orientation** that lies between the start and end orientation.

Programming

ORIPLANE	**ori**entation interpolation in a **plane**: Interpolation in a plane (large circle interpolation)
ORICONCW	**ori**entation interpolation on a **con**e **c**lock**w**ise: Interpolation on the peripheral surface of a cone in the clockwise direction
ORICONCCW	**ori**entation interpolation on a **con**e **c**ounter **c**lock**w**ise: Interpolation on the peripheral surface of a cone in the counter-clockwise direction.

Programming of the **direction vector** is carried out using the identifiers A6, B6, C6 and is specified as a (normalized) vector.

Note

Programming of an end orientation is **not absolutely** necessary. If no end orientation is specified, a full outside cone with 360 degrees is interpolated.

The **opening angle of the taper** is programmed with NUT= <angle>, where the angle is specified in degrees.

Note

An end orientation **must** be specified. A complete outside cone with 360 degrees cannot be interpolated in this way. The sign of the opening angle defines whether the traversing angle is to be greater or less than 180 degrees.

The identifiers have the following meanings:

NUT = +...	Traverse angle smaller than or equal to 180 degrees
NUT = -...	Traverse angle greater than or equal to 180 degrees

A positive sign can be omitted when programming.

Settings for intermediate orientation

ORICONIO	**ori**entation interpolation on a **con**e with **i**ntermediate **o**rientation: Interpolation on a conical peripheral surface with intermediate orientation setting

If this G command is active, it is necessary to specify an **intermediate orientation** with A7, B7, C7 which is specified as a (normalized) vector.

Note

Programming of the end orientation is **absolutely** necessary in this case.

The **change in orientation** and the **direction of rotation** is defined uniquely by the three vectors Start, End and Intermediate orientation.

All three vectors must be different from each other. If the programmed intermediate orientation is parallel to the start or end orientation, a linear large circle interpolation of the orientation is executed in the plane that is defined by the start and end vector.

Angle of rotation and opening angle

The following may be programmed additionally for the angle of the cone:

PHI	**angle of rotation** for orientation about the direction axis
PSI	**Opening angle** of the cone

Besides this polynomials of the 5th degree (max.) can be programmed as follows:

PO[PHI] = (a2, a3, a4, a5)
PO[PSI] = (b2, b3, b4, b5)

Constant and linear coefficients are defined by start and end orientation respectively.

Further interpolation options

It is possible to interpolate the oreintation on a cone which connects tangentially to the previous change of orientation. This orientation interpolation is achieved by programming the G command `ORICONTO`.

`ORICONTO`	**ori**entation interpolation on a **con**e with **t**angential **o**rientation: Interpolation on a peripheral surface of the cone with tangential transition

A further option for orientation interpolation is to describe the change in orientation through the path of a 2nd contact point on the tool.

`ORICURVE`	**ori**entation interpolation with a second **curve**: Interpolation of orientation with specification of motion of two contact points of the tool.

The coordinates for the movement of the 2nd contact point of the tool must be specified. This additional curve in space is programmed with XH, YH, ZH.

Besides the two end values, additional polynomials can be programmed in the following form:

PO[XH] = (xe, x2, x3, x4, x5): (xe, ye, ze) of the end point of the curve, and

PO[YH] = (ye, y2, y3, y4, y5): xi, yi, zi the coefficients of the polynomials

PO[ZH] = (ze, z2, z3, z4, z5): of the 5th degree maximum.

This type of interpolation can be used to program points (G1) or polynomials (POLY) for the two curves in space.

Note

Circles or involutes are specificaly not allowed. It is also possible to activate a spindle interpolation with BSPLINE. The programmed end points of both curves in space are then interpreted as nodes.

Other types of splines (ASPLINE and CSPLINE) and the activation of a compressor (COMPON, COMPCURV, COMPCAD) are not permitted here.

Supplementary conditions

The extended interpolation of orientations requires that all necessary orientation transformations be considered, since these belong to the functional scope.

Activation

A change in orientation on any peripheral surface of a cone in space is activated with the G command of group 51 through extended interpolation of the orientation vector, using the following commands:

`ORIPLANE`	Interpolation in a plane with specification of the end orientation (same as ORIVECT)
`ORICONCW`	Interpolation on a peripheral surface of a taper in clockwise directionwith specification of the end orientation and taper direction or opening angle of the cone.
`ORICONCCW`	Interpolation on the peripheral surface of a cone in the counterclockwise direction. Specification of the end orientation andt cone direction or opening angle of the taper.
`ORICONIO`	Interpolation on a peripheral surface of a cone with specification of end orientation and an intermediate orientation.
`ORICONTO`	Interpolation on a peripheral surface of a cone with tangential transition and specification of end orientation.
`ORICURVE`	Interpolation of orientation with specification of motion of two contact points of the tool.
`ORIPATH`	Tool orientation in relation to the path
`ORIPATHS`	Tool orientation in relation to the path, when, for example, a kink in the orientation path, e.g. at a corner in the contour, is to be smoothed (see Section "Path-relative orientation (ORIPATH, ORIPATHS, ORIROTC) (Page 200)").

Examples

Various changes in orientation are programmed in the following program example:

```
Program code                              Comment
...
N10 G1 X0 Y0 F5000
N20 TRAORI                                ; orientation transformation active.
N30 ORIVECT                               ; interpolate tool orientation as vector
N40 ORIPLANE                              ; select large circle interpolation.
N50 A3=0 B3=0 C3=1
N60 A3=0 B3=1 C3=1                        ; orientation in the Y/Z plane by 45
                                          ; degrees rotated, at the block end the
                                          ; orientation (0, 1 / (root of 2),
                                          ; 1 / (root of 2) is reached.
N70 ORICONCW                              ; the orientation vector is interpolated on a
                                          ; conical surface with direction
N80 A6=0 B6=0 C6=1 A3=1 B3=0 C3=1; (0, 0, 1) to the orientation
                                          ; (1 / (root of 2), 0, 1 / (root of 2)
                                          ; in the clockwise direction,
                                          ; the angle of rotation is 270 degrees.
                                          ; The tool orientation traverses a complete
N90 A6=0 B6=0 C6=1                        ; rotation on the same peripheral surface of the
                                          cone
...
```

3.12 Online tool length offset

Functionality

Effective tool length can be changed in real time so that the length changes are also considered for changes in orientation of the tool. The system variable $AA_TOFF[<geometry axis name>] includes tool length compensations in 3-D according to the three tool directions.

None of the tool parameters are changed. The actual compensation is performed internally by means of transformations using an orientable tool length compensation.

The number of active compensation directions must be the same as the number of active geometry axes. All offsets can be active at the same time.

Application

The online tool length compensation function can be used for:

- Orientation transformations (TRAORI)
- Orientable tool carriers (TCARR)

Note

The online tool length offset is an option. This function is only practical in conjunction with an active orientation transformation or an active orientable toolholder.

Further information
Function Manual Tools; Tool offset, orientable tool carriers

Block preparation

In the case of block preparation in run-in, the tool length offset currently active in the main run is considered. In order to utilize the maximum permissible axis velocities as far as possible, it is necessary to halt the block preparation with a stop preprocessing command (STOPRE) while a tool offset is being generated.

The tool offset is always known at the time of run-in when the tool length offsets are not changed after program start or if more blocks have been processed after changing the tool length offsets than the IPO buffer can accommodate between run-in and main run. This ensures that correct axis velocities are applied quickly.

The dimension for the difference between the currently active compensation in the interpolator and the compensation that was active at the time of block preparation can be polled in the system variable $AA_TOFF_PREP_DIFF[ ].

Note

Changing the effective tool length using online tool length offset produces changes in the compensatory movements of the axes involved in the transformation in the event of changes in orientation. The resulting velocities can be higher or lower depending on machine kinematics and the current axis position.

MD21190 $MC_TOFF_MODE (operation of tool offset)

The following machine data can be used to set whether the content of the synchronization variable $AA_TOFF[ ] is to be approached as an absolute value or whether an integrating behavior is to take place:

MD21190 $MC_TOFF_MODE

The integrating behavior of $AA_TOFF[ ] allows 3D remote control. The integrated value is available via the system variable $AA_TOFF_VAL[ ].

The following machine data and setting data are available for configuring online tool length compensation:

Machine data / setting data	Meaning for online tool length offset
MD21190 $MC_TOFF_MODE	The contents of $AA_TOFF[ ] are traversed as an absolute value or integrated
MD21194 $MC_TOFF_VELO (speed online tool offset)	Speed of online tool length offset
MD21194 $MC_TOFF_ACCEL (acceleration online tool offset)	Acceleration of online tool length offset
SD42970 $SC_TOFF_LIMIT (upper limit of offset value $AA_TOFF)	Upper limit of tool length offset value

With the acceleration margin, 20% is reserved for the overlaid movement of online tool length offset, which can be changed via the following machine data:

MD20610 $MC_ADD_MOVE_ACCEL_RESERVE(acceleration margin for overlaid movements)

Activation

The TOFFON instruction can be used to activate online tool length offset from the part program for at least one tool direction, if the option is available. During activation an offset value can be specified for the relevant direction of compensation and this is immediately traversed.

Example: TOFFON(Z, 25).

Repeated programming of the instruction TOFFON( ) with an offset causes the new offset to be applied. The offset value is added to variables $AA_TOFF[ ] as an absolute value.

Note

For further information about programming plus programming examples, please see:

Further information
Programming Manual Advanced; Transformations

As long as online tool length offset is active, the VDI signal at the NC → PLC interface in the following interface signal is set to 1:

DB21, ... DBX318.2 (TOFF active)

While a correction movement is active, the VDI → signal in the following interface signal is set to 1:

DB21, ... DBX318.3 (TOFF movement active)

Reset

Compensation values can be reset with the TOFFOF( ) command. This instruction triggers a preprocessing stop.

Accumulated tool length compensations are cleared and incorporated in the basic coordinate system. The run-in is synchronized with the current position in main run. Since no axes can be traversed here, the values of $AA_IM[ ] do not change. Only the values of the variables

$AA_IW[ ] and $AA_IB[ ] are changed. These variables now contain the deselected share of tool length compensation.

Once "Online tool length offset" has been deselected for a tool direction, the value of system variable $AA_TOFF[ ] or $AA_TOFF_VAL[ ] is zero for this tool direction. The following interface signal is set to 0:

DB21, ... DBX318.2 (TOFF active)

Alarm 21670

An existing tool length offset must be deleted via TOFFOF( ) so that alarm 21670 "Channel %1 block %2, illegal change of tool direction active due to $AA_TOFF active" is suppressed:

- When the transformation is deactivated with TRAFOOF
- On switch-over from CP to PTP travel.
- If a tool length offset exists in the direction of the geometry axis during geometry replacement.
- If a tool length offset is present during change of plane.
- When changing from axis-specific manual travel in the JOG mode to PTP as long as a tool length offset is active. There is no switchover to PTP.

Mode change

Tool length compensation remains active even if the mode is changed and can be executed in any mode.

If a tool length compensation is interpolated on account of $AA_TOFF[ ] during mode change, the mode change cannot take place until the interpolation of the tool length compensation has been completed. Alarm 16907 "Channel %1 action %2 ALNX possible only in stop state" is issued.

Behavior with REF and block search

Tool length offset is not considered during reference point approach REF in JOG mode.

The instructions TOFFON( ) and TOFFOF( ) are not collected and output in an action block during block search.

System variable

In the case of online tool length offset, the following system variables are available to the user:

System variable	Meaning for online tool length offset
$AA_TOFF[ ]	Position offset in the tool coordinate system
$AA_TOFF_VAL[ ]	Integrated position offset in the WCS
$AA_TOFF_LIMIT[ ]	Query whether the tool length offset value is close to the limit
$AA_TOFF_PREP_DIFF[ ]	Magnitude of the difference between the currently active value of $AA_TOFF[ ] and the value prepared as the current motion block.

Further information
List Manual System Variables

Boundary conditions

The online tool length offset function is an option and is available during "generic 5-axis transformation" by default and for "orientable toolholders".

If the tool is not perpendicular to the workpiece surface during processing or the contour contains curvatures whose radius is smaller than the compensation dimension, deviations compared to the actual offset surface are produced. It is not possible to produce exact offset surfaces with one tool length compensation alone.

3.13 Examples

3.13.1 Example of a 5-axis transformation

```
CHANDATA(1)

$MA_IS_ROT_AX[AX5] = TRUE
$MA_SPIND_ASSIGN_TO_MACHAX[AX5] = 0
$MA_ROT_IS_MODULO[AX5]=0


;------------------------------------------------------------------------------------------
; general 5-axis transformation
;
; kinematics:                    1. rotary axis is parallel to Z
;                                2. rotary axis is parallel to X
;                                Movable tool
;------------------------------------------------------------------------------------------


$MC_TRAFO_TYPE_1 = 20

$MC_ORIENTATION_IS_EULER = TRUE

$MC_TRAFO_AXES_IN_1[0] = 1
$MC_TRAFO_AXES_IN_1[1] = 2
$MC_TRAFO_AXES_IN_1[2] = 3
$MC_TRAFO_AXES_IN_1[3] = 4
$MC_TRAFO_AXES_IN_1[4] = 5
```

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3

$MC_TRAFO5_PART_OFFSET_1[0] = 0
$MC_TRAFO5_PART_OFFSET_1[1] = 0
$MC_TRAFO5_PART_OFFSET_1[2] = 0
$MC_TRAFO5_ROT_AX_OFFSET_1[0] = 0
$MC_TRAFO5_ROT_AX_OFFSET_1[1] = 0

$MC_TRAFO5_ROT_SIGN_IS_PLUS_1[0] = TRUE

$MC_TRAFO5_ROT_SIGN_IS_PLUS_1[1] = TRUE
$MC_TRAFO5_NON_POLE_LIMIT_1 = 2.0

$MC_TRAFO5_POLE_LIMIT_1 = 2.0
$MC_TRAFO5_BASE_TOOL_1[0] = 0.0
$MC_TRAFO5_BASE_TOOL_1[1] = 0.0
$MC_TRAFO5_BASE_TOOL_1[2] = 5,0

$MC_TRAFO5_JOINT_OFFSET_1[0] = 0.0
$MC_TRAFO5_JOINT_OFFSET_1[1] = 0.0
$MC_TRAFO5_JOINT_OFFSET_1[2] = 0.0

CHANDATA(1)
M17

Program example for general 5-axis transformation:

Program code	**Comment**
`; Definition of tool T1`	
`$TC_DP1[1,1] = 10`	`; Type`
`$TC_DP2[1,1] = 0`	
`$TC_DP3[1,1] = ;z  length compensation vector G17`	
`$TC_DP4[1,1] = 0.`	`; y`
`$TC_DP5[1,1] = 0.`	`; x`
`$TC_DP6[1,1] = 0.`	`; Radius`
`$TC_DP7[1,1] = 0`	
`$TC_DP8[1,1] = 0`	
`$TC_DP9[1,1] = 0`	
`$TC_DP10[1,1] = 0`	
`$TC_DP11[1,1] = 0`	
`$TC_DP12[1,1] = 0`	

Approach initial position:

```
N100 G1 x1 y0 z0 a0 b0 F20000 G90 G64 T1 D1 G17 ADIS=.5 ADISPOS=3
```

Orientation vector programming:

```
N110 TRAORI(1)
N120 ORIWKS
N130 G1 G90
N140 a3 = 0 b3 = 0 c3 = 1 x0
N150 a3 = 0 b3 =-1 c3 = 0
N160 a3 = 1 b3 = 0 c3 = 0
N170 a3 = 1 b3 = 0 c3 = 1
N180 a3 = 0 b3 = 1 c3 = 0
N190 a3 = 0 b3 = 0 c3 = 1
```

Euler angles program:

```
N200 ORIMKS
N210 G1 G90
N220 a2 = 0  b2 = 0  x0
N230 a2 = 0  b2 = 90
N240 a2 = 90 b2 = 90
N250 a2 = 90 b2 = 45
N260 a2 = 0  b2 =-90
N270 a2 = 0  b2 = 0
```

Axis programming:

```
N300 a0 b0 x0
N310 a45
N320 b30
```

TOFRAME:

```
N400 G0 a90 b90 x0 G90
N410 TOFRAME
N420 z5
N430 x3 y5
N440 G0 a0 b0 x1 y0 z0 G90

N500 TRAFOOF
m30
```

3.13.2 Example of a 3-axis and 4-axis transformation

3.13.2.1 Example of a 3-axis transformation

Example: For the schematically represented machine (see "Figure 3-1 Schematic diagram of 3-axis transformation (Page 139)"), the 3-axis transformation can be projected as follows:

```
Program code                                    Comment
$MC_TRAFO_TYPE_n = 18
$MC_TRAFO_GEOAX_ASSIGN_TAB_n[0] = 1             ; Assignment of channel axes to geometry
                                                  axes
$MC_TRAFO_GEOAX_ASSIGN_TAB_n[1] = 0
$MC_TRAFO_GEOAX_ASSIGN_TAB_n[2] = 3
$MC_TRAFO_AXES_IN_n[0] = 1                      ; X axis is channel axis 1
$MC_TRAFO_AXES_IN_n[1] = 0                      ; Y axis is not used
$MC_TRAFO_AXES_IN_n[2] = 3                      ; Z axis is channel axis 3
$MC_TRAFO_AXES_IN_n[4] = 0                      ; There is no second rotary axis
```

3.13.2.2 Example of a 4-axis transformation

Example: For the schematically represented machine (see "Figure 3-2 Schematic diagram of a 4-axis transformation with moveable workpiece (Page 140)"), however, with an additional axis (Y), the 4-axis transformation can be configured as follows:

```
Program code                                    Comment
$MC_TRAFO_TYPE_n = 18

$MC_TRAFO_GEOAX_ASSIGN_TAB_n[0] = 1
$MC_TRAFO_GEOAX_ASSIGN_TAB_n[1] = 2
$MC_TRAFO_GEOAX_ASSIGN_TAB_n[2] = 3

$MC_TRAFO_AXES_IN_n[0] = 1                    ; X axis is channel axis 1
$MC_TRAFO_AXES_IN_n[1] = 2                    ; Y axis is channel axis 2
$MC_TRAFO_AXES_IN_n[2] = 3                    ; Z axis is channel axis 3

$MC_TRAFO_AXES_IN_n[4] = 0                    ; There is no second rotary axis
```

3.13.3 Example of a universal milling head

General

The following two subsections show the main steps which need to be taken in order to activate a transformation for the universal milling head.

Machine data

; machine kinematics CA' with tool orientation in zero position in the z direction
$MC_TRAFO_TYPE_1 = 148

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3

; angle of second rotary axis
$MC_TRAFO5_NUTATOR_AX_ANGLE_1 = 45

Program

```
; Definition of tool T1
$TC_DP1[1,1] = 120                      ; Type
$TC_DP2[1,1] = 0                        ;
$TC_DP3[1,1] = 20                       ; Z length offset vector G17
$TC_DP4[1,1] = 8.                       ; Y
$TC_DP5[1,1] = 5.                       ; X

TRAORI(1)                               ; Activation of transformation
ORIMKS                                  ; Orientation reference to MCS
G0 X1 Y0 Z0 A0 B0 F20000 G90 G64 T1 D1 G17
; Programming of direction vector
G1 G90
a3 = 0 b3 = 1 c3 = 0
; Programming in Euler angles
G1 G90
a2 = 0 b2 = 0 X0

; Programming of rotary axis motion
G1 X10 Y5 Z20 A90 C90

m30
```

References:
Programming Manual, Fundamentals

3.13.4 Example for orientation axes

Example 1:

3 orientation axes for the 1st orientation transformation for kinematics with 6 transformed axes. Rotation must be done in the following sequence:

- firstly about the Z axis.
- then about the Y axis and
- finally about the Z axis again.

The tool vector must point in the X direction.

Program code	Comment
CHANDATA(1)	
$MC_TRAFO5_TOOL_VECTOR_1=0	;Tool vector in X direction
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[0]=4	;Channel index 1st orient. axis
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[1]=5	;Channel index 2nd orient. axis
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[2]=6	;Channel index 3rd orient. axis
$MC_ORIAX_TURN_TAB_1[0]=3	;Z direction
$MC_ORIAX_TURN_TAB_1[1]=2	;Y direction
$MC_ORIAX_TURN_TAB_1[2]=3	;Z direction
CHANDATA(1)	
M17	

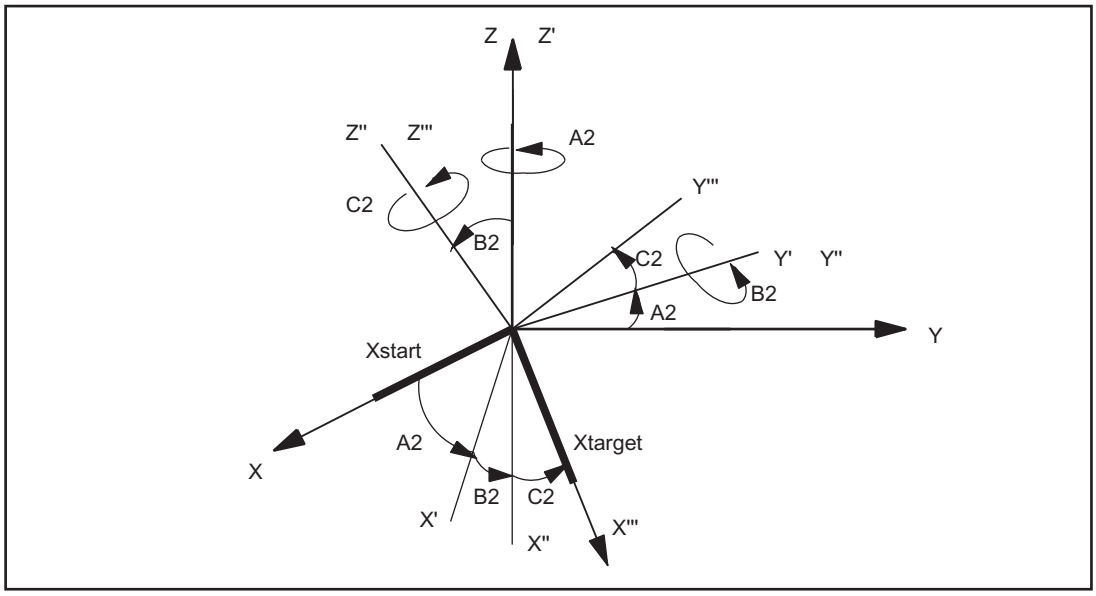


Figure 3-26 3 orientation axes for the 1st orientation transformation for kinematics with 6 transformed axes

Example 2:

3 orientation axes for the 2nd orientation transformation for kinematics with 5 transformed axes. Rotation must be done in the following sequence:

- firstly about the X axis.
- then about the Y axis and
- finally about the Z axis.

The tool vector must point in the Z direction.

Program code	Comment
CHANDATA(1)	
$MC_TRAFO5_TOOL_VECTOR_2=2	;Tool vector in Z direction
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[0]=4	;Channel index 1st orient. axis
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[1]=5	;Channel index 2nd orient. axis
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[2]=0	;Channel index 3rd orient. axis
$MC_ORIAX_TURN_TAB_1[0]=1	;X direction
$MC_ORIAX_TURN_TAB_1[1]=2	;Y direction
$MC_ORIAX_TURN_TAB_1[2]=3	;Z direction
CHANDATA(1)	
M17	

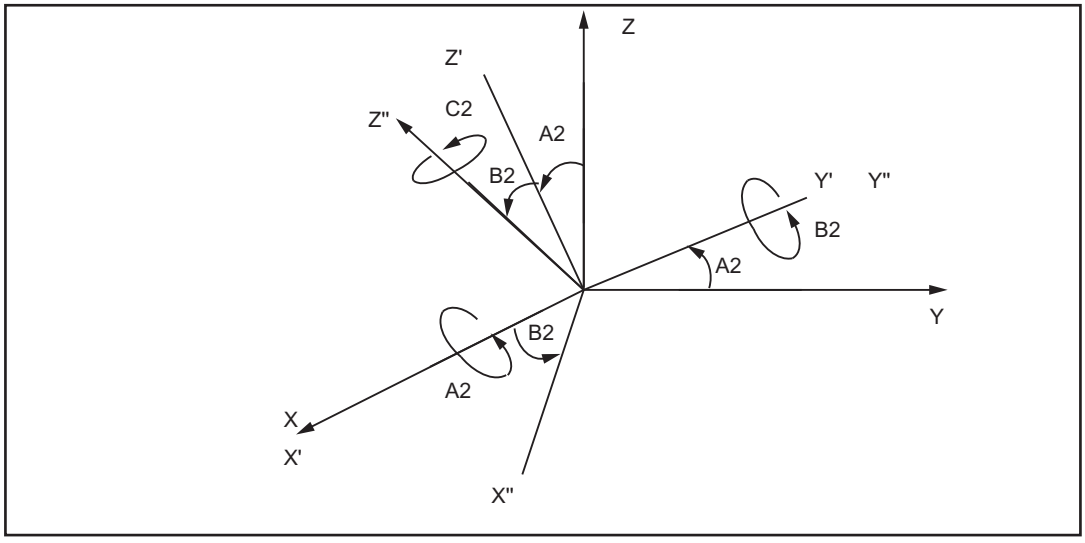


Figure 3-27 3 orientation axes for the 2nd orientation transformation for kinematics with 5 transformed axes

The rotation through angle C2 about the Z" axis is omitted in this case, because the tool vector orientation can be determined solely from angles A2 and B2 and no further degree of freedom is available on the machine.

Further information

Programming Manual Advanced

3.13.5 Examples for orientation vectors

3.13.5.1 Example for polynomial interpretation of orientation vectors

Orientation vector in Z-X plane

The orientation vector is programmed directly in the examples below. The resulting movements of the rotary axes depend on the particular kinematics of the machine.

```
Program code                    Comment
N10 TRAORI
N20 POLY                        ; Polynomial interpolation is possible.
N30 A3=0 B3=0 C3=1              ; Orientation in +Z direction (start vector)
N40 A3=1 B3=0 C3=0              ; Orientation in +X direction (end vector)
```

In N40, the orientation vector is rotated in the Z-X plane which is spanned by the start and end vector. Here, the PHI angle is interpolated in a line in this plane between the values 0 and 90 degrees (large circle interpolation).

The additional specification of the polynomials for the two angles PHI and PSI means that the interpolated orientation vector can lies anywhere between the start and end vector.

PHI angle using polynomial PHI

In contrast to the example above, the PHI angle is interpolated using the polynomial PHI(u) = (90-10)u + $10*u^2$ between the values 0 and 90 degrees.

The angle PSI is not equal to zero and is interpolated according to the following polynomial:

PSI(u) = $-10*u + 10*u^2$

The maximum "tilt" of the orientation vector from the plane between the start and end vector is obtained in the middle of the block (u = 1/2).

```
Program code                          Comment
N10 TRAORI
N20 POLY                              ; Polynomial interpolation is possible.
N30 A3=0 B3=0 C3=1                    ; Orientation in +Z direction (start vector)
N40 A3=1 B3=0 C3=0     PO[PHI]=(10)   ; in +X direction (end vector)
                       PO[PSI]=(10)
```

3.13.5.2 Example of rotations of orientation vector

Rotations with angle of rotation THETA

In the following example, the angle of rotation is interpolated in linear fashion from starting value 0 degrees to end value 90 degrees. The angle of rotation changes according to a parabola or a rotation can be executed without a change in orientation. tool orientation is rotated from the Y direction to the X direction.

```
Program code                              Comment
N10 TRAORI                                ; Activation of orientation transformation
N20 G1 X0 Y0 Z0 F5000                     ;
                                          ; Tool orientation
N30 A3=0 B3=0 C3=1 THETA=0                ; In Z direction with angle of rotation 0
N40 A3=1 B3=0 C3=0 THETA=90               ; In X direction and rotation
                                          ; By 90 degrees
N50 A3=0 B3=1 C3=0 PO[THT]=(180,90)       ; In Y direction and rotation
                                          ; To 180 degrees
N60 A3=0 B3=1 C3=0 THETA=IC(-90)          ; Remains constant and rotation
                                          ; To 90 degrees
N70 ORIROTT                               ; Angle of rotation relative to
                                          ; change in orientation.
N80 A3=1 B3=0 C3=0 THETA=30               ; Rotation vector in angle
                                          ; 30 degrees to X-Y plane.
```

N40 Linear interpolation of angle of rotation from starting value 0 degrees to end value 90 degrees.

N50 The angle of rotation changes from 90 degrees to 180 degrees in accordance with the parabola.

$\theta(u) = 90 + u^2$

N60 A rotation can also be programmed without a change in orientation taking place.

N80 Tool orientation is rotated from the Y direction to the X direction. The change in orientation takes place in the X-Y plane and the rotation vector describes an angle of 30 degrees to this plane.

3.13.6 Examples for generic axis transformations

3.13.6.1 Example of generic 5-axis transformation

The following example is based on a machine with rotatable tool on which the first rotary axis is a C axis and the second a B axis (CB kinematics). The basic orientation defined in the machine data is the bisecting line between the X and Z axes.

Relevant machine data is as follows:

```
CHANDATA(1)
$MC_TRAFO_TYPE_1 = 24                    ; General 5-axis transformation
                                         ; Rotatable tool
$MC_TRAFO5_AXIS1_1[0] = 0.0
$MC_TRAFO5_AXIS1_1[1] = 0.0
$MC_TRAFO5_AXIS1_1[2] = 1,0              ; 1. Rotary axis is parallel to Z.
$MC_TRAFO5_AXIS2_1[0] = 0.0
$MC_TRAFO5_AXIS2_1[1] = 1,0
$MC_TRAFO5_AXIS2_1[2] = 0.0              ; 2. Rotary axis is parallel to Y.
$MC_TRAFO5_BASE_ORIENT_1[0] = 1.0
$MC_TRAFO5_BASE_ORIENT_1[1] = 0,0
$MC_TRAFO5_BASE_ORIENT_1[2] = 1.0
M30
```

Example program:

```
Program code                          Comment
N10      $TC_DP1[1,1] = 120           ; End mill
N20      $TC_DP3[1,1]= 0              ; Length offset vector
N30
N40                                   ; Definition of tool carrier
N50      $TC_CARR7[1] = 1             ; Component of the 1st rotary axis
                                      ; In the X direction
N60      $TC_CARR11[1] = 1            ; Component of the 2nd rotary axis
                                      ; In the Y direction
N70      $TC_CARR13[1] = -45          ; Angle of rotation of 1st axis
N80      $TC_CARR14[1] = 0            ; Angle of rotation of 2nd axis
N90
N100     X0 Y0 Z0 B0 C0 F10000 ORIWKS G17
N110     TRAORI()                     ; Selection of basic transformation orientation
                                      ; from the machine data
N120     C3=1                         ; Orientation parallel to Z
                                      ; Set → B-45 C0
N130     T1 D1                        ; Basic orientation is now parallel to Z
N140     C3=1                         ; Orientation parallel to Z
                                      ; Set → B0 C0
N150     G19                          ; Basic orientation is now parallel to X
N160     C3=1                         ; Orientation parallel to Z
                                      ; Set → B-90 C0
N170     G17 TCARR=1 TCOABS           ; Basic orientation is now angle-
                                      ; halving Y-Z
N180     A3=1                         ; Orientation parallel to X
                                      ; Set → B-90 C-135
N190     B3=1 C3=1                    ; Orientation parallel to
```

Program code		Comment
		; basic orientation → B0 C0
N200	TRAORI(,2.0, 3.0, 6.0)	; Pass basic orientation in call
N210	A3=2 B3=3 C3=6	; Orientation parallel to
		; basic orientation → B0 C0
N220	TOFRAME	; Z axis points in the direction
		; of the orientation
N230	G91 Z7	; 7 mm in new Z direction
		; Traverse → X2 Y3 Z6
N240	C3=1	; Orientation parallel to
		; New Z axis → B0 C0
N250	M30	

3.13.6.2 Example of a generic 6-axis transformation

Call with parameterization using parameters

Activation of a 6-axis transformation with subsequent orientation changes and traversing:

Program code	Comment
N10 A0 B0 X0 Y0 Z0	
N20 TRAORI(1, ,,, 0,0,0, 0,1,0)	; Selection using parameters:
	; parameters 5, 6, 7: Orientation vector
	; Parameters 8, 9, 10: Orientation normal vector
N30 T1 D1 X10 Y20 Z30 A3=0.5	; Orientation change, rotation
C3=1 BN3=1 ORIPLANE ORIWKS	; and traversing motion
N40 B3=0.5 C3=1 AN3=-1	; Rotation, constant orientation
M30	

Call with parameterization using tool data

Defining a tool, where the orientation, with an orientation vector of (1.0 ; 0.0 ; 0.5) deviates from the standard. For G17, the orientation vector is in the X-Z plane, and starting from the X axis in the direction of the Z vector, inclined by 26.565 degrees:

$\tan^{-1}(\$TC\_DPV5[\ 2,\ 2\ ] / \$TC\_DPV3[\ 2,\ 2\ ]) = \tan^{-1}(0.5 / 1.01) = \tan^{-1}(0.5) = 26.565°$

The orientation normal vector is also specified. As only $TC_DPVN4[2,2] is not equal to zero, it points in the Y direction. Orientation vector and orientation normal vector are perpendicular to one another. Orthogonalization is not necessary. The programmed orientation normal vector is not modified.

Program code	Comment
N100 $TC_DP1[2,2]=120	; End mill
N110 $TC_DP3[2,2]= 20	; Length offset vector
N120 $TC_DPV[2,2]= 0	; Tool cutting edge orientation
	; Orientation vector tool cutting edge
N130 $TC_DPV3[2,2]= 1	; X component

Program code	Comment
N140 $TC_DPV4[2,2]= 0	; Y component
N150 $TC_DPV5[2,2]= 0.5	; Z component
	; Orientation normal vector
N160 $TC_DPVN3[2,2]= 0	; X component
N170 $TC_DPVN4[2,2]= 1	; Y component
N180 $TC_DPVN5[2,2]= 0	; Z component
N200 TRAORI( )	; Call with basic orientation
N210 A3=5 C3=10 BN3=1	; Bring rotary axes into the initial state
N220 C3=1	; Orientation in the Z direction ⇒
	; Tool rotated through 26.565 degrees
N230 THETA=IC(90)	; Orientation normal vector incremental
	; Rotated through 90 degrees. Vector points in negative X direction
N240 M30	

3.13.6.3 Example of a generic 7-axis transformation

Example of a generic 7-axis transformation

Activation of a 7-axis transformation with subsequent orientation changes and traversing:

Program	Comment
N10 TRAFOOF	
N20 a0 b0 c0 x0 y0 z0 e=0	
N30 $MC_TRAFO5_AXIS1_1[2] = 1	; 1. Rotary axis shows in Z direction
N40 $MC_TRAFO5_AXIS1_2[0] = 1	; 2. Rotary axis shows in X direction
N50 $MC_TRAFO5_AXIS1_3[2] = 1	; 3. Rotary axis shows in Z direction
N60 $MC_TRAFO7_EXT_AXIS1_1[0] = 1	; 7. Axis shows in X direction
N70 $MC_TRAFO_BASE_ORIENT_1[2] = 1	; Orientation vector
N80 $MC_TRAFO_BASE_ORIENT_NORMAL_1[1] = 1	; Orientation normal vector
N90 NEWCONF	
N100 traori()	
N110 G1 t1 d1 x10 y0 z50 c3=1 an3=1 bn3=1 orivect oriwks G19 F10000	
N120 G2 y50 z0 b3=1 e=DC(90) CR=50	; 1. Quadrant
N130 G2 y0 z-50 c3=-1 e=DC(180) CR=50	; 2. Quadrant
N140 G2 y-50 z0 b3=-1 e=DC(270) CR=50	; 3. Quadrant
N150 G2 y0 z50 c3=1 e=DC(0) CR=50	; 4. Quadrant
N200 M30	

Note

While traversing the quadrant in the example, only the 7th axis turns by 360 degrees. The machine remains in the fixed position.

3.13.6.4 Example for the modification of rotary axis motion

The machine is a 5-axis machine of machine type 1 (two-axis swivel head with CA kinematics) on which both rotary axes rotate the tool (transformation type 24). The first rotary axis is a modulo axis parallel to Z (C axis); the second rotary axis is parallel to Y (B axis) and has a traversing range from -5 degrees to +185 degrees.

To allow modification at any time, the following machine data has the value 2:

MD21180 $MC_ROT_AX_SWL_CHECK_MODE (check software limits for orientation axes)

```
N10 X0 Y0 Z0 B0 C0
N20 TRAORI( )                          ; basic orientation 5-axis transformation
N30 B-1 C10                            ; Rotary axis positions B-1 and C10
N40 A3=-1 C3=1 ORIWKS                  ; large circle interpolation in WCS
N50 M30
```

At the start of block N40 in the example program, the machine is positioned at rotary axis positions B-1 C10. The programmed end orientation can be achieved with either of the axis positions B-45 C0 (1st solution) or B45 C180 (2nd solution).

The first solution is selected initially, because it is nearest to the starting orientation and, unlike the second solution, can be achieved using large circle interpolation (ORIWKS). However, this position **cannot** be reached because of the axis limits of the B axis.

The second solution is therefore used instead, i.e. the end position is B45 C180. The end orientation is achieved by axis interpolation. The programmed orientation path cannot be followed.

3.14 Data lists

3.14.1 Machine data

3.14.1.1 General machine data

Number	Identifier: $MN_	Description
10620	EULER_ANGLE_NAME_TAB	Name of Euler angles or names of orientation axes
10630	NORMAL_VECTOR_NAME_TAB	Name of normal vectors
10640	DIR_VECTOR_NAME_TAB	Name of direction vectors

Number	Identifier: $MN_	Description
10642	ROT_VECTOR_NAME_TAB	Name of rotation vectors
10644	INTER_VECTOR_NAME_TAB	Name of intermediate vector components
10646	ORIENTATION_NAME_TAB	Identifier for programming a 2nd orientation path
10648	NUTATION_ANGLE_NAME	Name of orientation angle
10670	STAT_NAME	Name of position information
10672	TU_NAME	Name of position information of the axes
10674	PO_WITHOUT_POLY	Permits programming of PO[ ] without POLY having to be active

3.14.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups
20152	GCODE_RESET_MODE[n]	Setting after RESET/end of part program
20482	COMPRESS_MODE	Mode of the compressor
20621	HANDWH_ORIAX_MAX_INCR_SIZE	Limitation of handwheel increment
20623	HANDWH_ORIAX_MAX_INCR_VSIZE	Orientation velocity overlay
21094	ORIPATH_MODE	Setting for path relative orientation
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21102	ORI_DEF_WITH_G_CODE	Definition of orientation angles A2, B2, C2
21104	ORI_IPO_WITH_G_CODE	Definition of interpolation type for orientation
21106	CART_JOG_SYSTEM	Coordinate system for Cartesian JOG
21108	POLE_ORI_MODE	Behavior during large circle interpolation at pole position
21120	ORIAX_TURN_TAB_1[n]	Assignment of rotation of orientation axes about the reference axes, definition 1 [n = 0..2]
21130	ORIAX_TURN_TAB_2[n]	Assignment of rotation of orientation axes about the reference axes, definition 2 [n = 0..2]
21132	ORI_DISP_IS_MODULO[n]	Modulo display of the orientation axes positions [n = 0..2]
21134	ORI_DISP_MODULO_RANGE	Size of the module range for the display of the orientation axes
21136	ORI_DISP_MODULO_RANGE_START	Starting position of the module range for the display of the orientation axes
21150	JOG_VELO_RAPID_ORI[n]	Rapid traverse in jog mode for orientation axes in the channel [n = 0..2]
21155	JOG_VELO_ORI[n]	Orientation axis velocity in jog mode [n = 0..2]
21160	JOG_VELO_RAPID_GEO[n]	Rapid traverse in jog mode for geometry axes in the channel [n = 0..2]
21165	JOG_VELO_GEO[n]	Geometry axis velocity in jog mode [n = 0..2]
21170	ACCEL_ORI[n]	Acceleration for orientation axes [n = 0..2]
21180	ROT_AX_SWL_CHECK_MODE	Check software limits for orientation axes
21186	TOCARR_ROT_OFFSET_FROM_FR	Offset of TOCARR rotary axes
21190	TOFF_MODE	Operation of online offset in tool direction

Number	Identifier: $MC_	Description
21194	TOFF_VELO	Speed of online offset in tool direction
21196	TOFF_ACCEL	Acceleration of online offset in tool direction
24100	TRAFO_TYPE_1	Definition of transformation 1 in channel
24110	TRAFO_AXES_IN_1[n]	Axis assignment for transformation 1 [axis index]
24120	TRAFO_GEOAX_ASSIGN_TAB_1[n]	Assignment geometry axis to channel axis for transformation 1 [geometry no.]
24200	TRAFO_TYPE_2	Definition of transformation 2 in channel
24210	TRAFO_AXES_IN_2[n]	Axis assignment for transformation 2 [axis index]
24220	TRAFO_GEOAX_ASSIGN_TAB_2[n]	Assignment geometry axis to channel axis for transformation 2 [geometry no.]
24300	TRAFO_TYPE_3	Definition of transformation 3 in channel
24310	TRAFO_AXES_IN_3[n]	Axis assignment for transformation 3 [axis index]
24320	TRAFO_GEOAX_ASSIGN_TAB_3[n]	Assignment geometry axis to channel axis for transformation 3 [geometry no.]
24400	TRAFO_TYPE_4	Definition of transformation 4 in channel
24410	TRAFO_AXES_IN_4[n]	Axis assignment for transformation 4 [axis index]
24420	TRAFO_GEOAX_ASSIGN_TAB_4[n]	Assignment geometry axis to channel axis for transformation 4 [geometry no.]
24430	TRAFO_TYPE_5	Definition of transformation 5 in channel
24432	TRAFO_AXES_IN_5[n]	Axis assignment for transformation 5 [axis index]
24434	TRAFO_GEOAX_ASSIGN_TAB_5[n]	Assignment geometry axis to channel axis for transformation 5 [geometry no.]
24440	TRAFO_TYPE_6	Definition of transformation 6 in channel
24442	TRAFO_AXES_IN_6[n]	Axis assignment for transformation 6 [axis index]
24444	TRAFO_GEOAX_ASSIGN_TAB_6[n]	Assignment geometry axis to channel axis for transformation 6 [geometry no.]
24450	TRAFO_TYPE_7	Definition of transformation 7 in channel
24452	TRAFO_AXES_IN_7[n]	Axis assignment for transformation 7 [axis index]
24454	TRAFO_GEOAX_ASSIGN_TAB_7[n]	Assignment geometry axis to channel axis for transformation 7 [geometry no.]
24460	TRAFO_TYPE_8	Definition of transformation 8 in channel
24462	TRAFO_AXES_IN_8[n]	Axis assignment for transformation 8 [axis index]
24464	TRAFO_GEOAX_ASSIGN_TAB_8[n]	Assignment geometry axis to channel axis for transformation 8 [geometry no.]
24470	TRAFO_TYPE_9	Definition of transformation 9 in channel
24472	TRAFO_AXES_IN_9[n]	Axis assignment for transformation 9 [axis index]
24474	TRAFO_GEOAX_ASSIGN_TAB_9[n]	Assignment geometry axis to channel axis for transformation 9 [geometry no.]
24480	TRAFO_TYPE_10	Definition of transformation 10 in channel
24482	TRAFO_AXES_IN_10[n]	Axis assignment for transformation 10 [axis index]
24484	TRAFO_GEOAX_ASSIGN_TAB_10[n]	Assignment geometry axis to channel axis for transformation 10 [geometry no.]
24500	TRAFO5_PART_OFFSET_1[n]	Offset vector for 5-axis transformation 1 [n = 0.. 2]

Number	Identifier: $MC_	Description
24510	TRAFO5_ROT_AX_OFFSET_1[n]	Position offset of rotary axis 1/2 for 5-axis transformation 1 [axis no.]
24520	TRAFO5_ROT_SIGN_IS_PLUS_1[n]	Sign of rotary axis 1/2 for 5-axis transformation 1 [axis no.]
24530	TRAFO5_NON_POLE_LIMIT_1	Definition of pole range for 5-axis transformation 1
24540	TRAFO5_POLE_LIMIT_1	End angle tolerance with interpolation through pole for 5-axis transformation 1
24550	TRAFO5_BASE_TOOL_1[n]	Vector of base tool for activation of 5-axis transformation 1 [n = 0.. 2]
24558	TRAFO5_JOINT_OFFSET_PART_1[n]	Vector of kinematic offset in table for 5-axis transformation 1 [n = 0.. 2]
24560	TRAFO5_JOINT_OFFSET_1[n]	Vector of kinematic offset for 5-axis transformation 1 [n = 0.. 2]
24561	TRAFO6_JOINT_OFFSET_2_3_1[n]	Vector of kinematic offset for 6-axis transformation 2_3_1
24562	TRAFO5_TOOL_ROT_AX_OFFSET_1[n]	Offset of focus of 1st 5-axis transformation with swiveled linear axis.
24564	TRAFO5_NUTATOR_AX_ANGLE_1	Angle of 2nd rotary axis for the universal milling head
24570	TRAFO5_AXIS1_1[n]	Vector for the first rotary axis and the first orientation transformation. [n = 0.. 2]
24572	TRAFO5_AXIS2_1[n]	Vector for the second rotary axis and the first transformation [n = 0.. 2]
24673	TRAFO5_AXIS3_1[n]	Direction of third rotary axis for general 6-axis transformation (Transformer type 24, 40, 56, 57)
24574	TRAFO5_BASE_ORIENT_1[n]	Basic orientation for the first transformation [n = 0.. 2]
24576	TRAFO6_BASE_ORIENT_NORMAL_1[n]	Tool normal vector for the first transformation [n = 0.. 2]
24580	TRAFO5_TOOL_VECTOR_1	Tool vector direction for the first 5-axis transformation 1
24582	TRAFO5_TCARR_NO_1	TCARR number for the first 5-axis transformation 1
24585	TRAFO5_ORIAX_ASSIGN_TAB_1[n]	Assignment of orientation axes to channel axes for orientation transformation 1 [n = 0.. 2]
24590	TRAFO5_ROT_OFFSET_FROM_FR_2	Offset of transf. rotary axes from WO
24594	TRAFO7_EXT_ROT_AX_OFFSET_1	Angle offset of the 1st external rotary axis
24595	TRAFO7_EXT_AXIS1_1	Direction of the 1st external rotary axis
24600	TRAFO5_PART_OFFSET_2[n]	Offset vector for 5-axis transformation 2 [n = 0.. 2]
24610	TRAFO5_ROT_AX_OFFSET_2[n]	Position offset of rotary axis 1/2 for 5-axis transformation 2 [axis no.]
24620	TRAFO5_ROT_SIGN_IS_PLUS_2[n]	Sign of rotary axis 1/2 for 5-axis transformation 2 [axis no.]
24630	TRAFO5_NON_POLE_LIMIT_2	Definition of pole range for 5-axis transformation 2
24640	TRAFO5_POLE_LIMIT_2	End angle tolerance with interpolation through pole for 5-axis transformation 2
24650	TRAFO5_BASE_TOOL_2[n]	Vector of base tool with activation of 5-axis transformation 2 [n = 0.. 2]
24658	TRAFO5_JOINT_OFFSET_PART_2[n]	Vector of kinematic offset in table for 5-axis transformation 2 [n = 0.. 2]

Number	Identifier: $MC_	Description
24660	TRAFO5_JOINT_OFFSET_2[n]	Vector of kinematic offset for 5-axis transformation 2 [n = 0.. 2]
24661	TRAFO6_JOINT_OFFSET_2_3_2[n]	Vector of kinematic offset for 6-axis transformation 2_3_2
24662	TRAFO5_TOOL_ROT_AX_OFFSET_2[n]	Offset of focus of 2nd 5-axis transformation with swiveled linear axis.
24664	TRAFO5_NUTATOR_AX_ANGLE_2	Angle of the 2nd rotating axis for the universal milling head
24670	TRAFO5_AXIS1_2[n]	Vector for the first rotary axis and the second orientation transformation [n = 0.. 2]
24672	TRAFO5_AXIS2_2[n]	Vector for the second rotary axis and the first transformation [n = 0.. 2]
24673	TRAFO5_AXIS3_2[n]	Direction of third rotary axis for generic 6-axis transformation (type 24, 40, 56, 57)
24674	TRAFO5_BASE_ORIENT_2[n]	Basic orientation for the second transformation [n = 0.. 2]
24676	TRAFO6_BASE_ORIENT_NORMAL_2[n]	Tool normal vector for the second transformation [n = 0.. 2]
24680	TRAFO5_TOOL_VECTOR_2	Tool vector direction for the second 5-axis transformation 2
24682	TRAFO5_TCARR_NO_2	TCARR number for the second 5-axis transformation 2
24685	TRAFO5_ORIAX_ASSIGN_TAB_2[n]	Assignment of orientation axes to channel axes for orientation transformation 2 [n = 0.. 2]
24694	TRAFO7_EXT_ROT_AX_OFFSET_2	Angle offset of the 2nd external rotary axis
24695	TRAFO7_EXT_AXIS1_2	Direction of the 2nd external rotary axis
25294	TRAFO7_EXT_ROT_AX_OFFSET_3	Angle offset of the 3rd external rotary axis
25295	TRAFO7_EXT_AXIS1_3	Direction of the 3rd external rotary axis
25394	TRAFO7_EXT_ROT_AX_OFFSET_4	Angle offset of the 4th external rotary axis
25395	TRAFO7_EXT_AXIS1_4	Direction of the 4th external rotary axis
28580	MM_ORIPATH_CONFIG	Configuration for path relative orientation ORIPATH

3.14.1.3 Channelspecific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Basic setting of the G groups
20152	GCODE_RESET_MODE[n]	Setting after RESET/end of part program
20482	COMPRESS_MODE	Mode of the compressor
20621	HANDWH_ORIAX_MAX_INCR_SIZE	Limitation of handwheel increment
20623	HANDWH_ORIAX_MAX_INCR_VSIZE	Orientation velocity overlay
21094	ORIPATH_MODE	Setting for path relative orientation
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21102	ORI_DEF_WITH_G_CODE	Definition of orientation angles A2, B2, C2
21104	ORI_IPO_WITH_G_CODE	Definition of the interpolation type for orientation
21106	CART_JOG_SYSTEM	Coordinate system for Cartesian JOG

Number	Identifier: $MC_	Description
21108	POLE_ORI_MODE	Behavior during large circle interpolation at pole position
21120	ORIAX_TURN_TAB_1[n]	Assignment of rotation of orientation axes about the reference axes, definition 1 [n = 0..2]
21130	ORIAX_TURN_TAB_2[n]	Assignment of rotation of orientation axes about the reference axes, definition 2 [n = 0..2]
21132	ORI_DISP_IS_MODULO[n]	Modulo display of the orientation axes positions [n = 0..2]
21134	ORI_DISP_MODULO_RANGE	Size of the module range for the display of the orientation axes
21136	ORI_DISP_MODULO_RANGE_START	Starting position of the module range for the display of the orientation axes
21150	JOG_VELO_RAPID_ORI[n]	Rapid traverse in jog mode for orientation axes in the channel [n = 0..2]
21155	JOG_VELO_ORI[n]	Orientation axis velocity in jog mode [n = 0..2]
21160	JOG_VELO_RAPID_GEO[n]	Rapid traverse in jog mode for geometry axes in the channel [n = 0..2]
21165	JOG_VELO_GEO[n]	Geometry axis velocity in jog mode [n = 0..2]
21170	ACCEL_ORI[n]	Acceleration for orientation axes [n = 0..2]
21180	ROT_AX_SWL_CHECK_MODE	Check software limits for orientation axes
21186	TOCARR_ROT_OFFSET_FROM_FR	Offset of TOCARR rotary axes
21190	TOFF_MODE	Operation of online offset in tool direction
21194	TOFF_VELO	Speed of online offset in tool direction
21196	TOFF_ACCEL	Acceleration of online offset in tool direction
24100	TRAFO_TYPE_1	Definition of transformation 1 in channel
24110	TRAFO_AXES_IN_1[n]	Axis assignment for transformation 1 [axis index]
24120	TRAFO_GEOAX_ASSIGN_TAB_1[n]	Assignment geometry axis to channel axis for transformation 1 [geometry no.]
24200	TRAFO_TYPE_2	Definition of transformation 2 in channel
24210	TRAFO_AXES_IN_2[n]	Axis assignment for transformation 2 [axis index]
24220	TRAFO_GEOAX_ASSIGN_TAB_2[n]	Assignment geometry axis to channel axis for transformation 2 [geometry no.]
24300	TRAFO_TYPE_3	Definition of transformation 3 in channel
24310	TRAFO_AXES_IN_3[n]	Axis assignment for transformation 3 [axis index]
24320	TRAFO_GEOAX_ASSIGN_TAB_3[n]	Assignment geometry axis to channel axis for transformation 3 [geometry no.]
24400	TRAFO_TYPE_4	Definition of transformation 4 in channel
24410	TRAFO_AXES_IN_4[n]	Axis assignment for transformation 4 [axis index]
24420	TRAFO_GEOAX_ASSIGN_TAB_4[n]	Assignment geometry axis to channel axis for transformation 4 [geometry no.]
24430	TRAFO_TYPE_5	Definition of transformation 5 in channel
24432	TRAFO_AXES_IN_5[n]	Axis assignment for transformation 5 [axis index]
24434	TRAFO_GEOAX_ASSIGN_TAB_5[n]	Assignment geometry axis to channel axis for transformation 5 [geometry no.]
24440	TRAFO_TYPE_6	Definition of transformation 6 in channel
24442	TRAFO_AXES_IN_6[n]	Axis assignment for transformation 6 [axis index]

Number	Identifier: $MC_	Description
24444	TRAFO_GEOAX_ASSIGN_TAB_6[n]	Assignment geometry axis to channel axis for transformation 6 [geometry no.]
24450	TRAFO_TYPE_7	Definition of transformation 7 in channel
24452	TRAFO_AXES_IN_7[n]	Axis assignment for transformation 7 [axis index]
24454	TRAFO_GEOAX_ASSIGN_TAB_7[n]	Assignment geometry axis to channel axis for transformation 7 [geometry no.]
24460	TRAFO_TYPE_8	Definition of transformation 8 in channel
24462	TRAFO_AXES_IN_8[n]	Axis assignment for transformation 8 [axis index]
24464	TRAFO_GEOAX_ASSIGN_TAB_8[n]	Assignment geometry axis to channel axis for transformation 8 [geometry no.]
24470	TRAFO_TYPE_9	Definition of transformation 9 in channel
24472	TRAFO_AXES_IN_9[n]	Axis assignment for transformation 9 [axis index]
24474	TRAFO_GEOAX_ASSIGN_TAB_9[n]	Assignment geometry axis to channel axis for transformation 9 [geometry no.]
24480	TRAFO_TYPE_10	Definition of transformation 10 in channel
24482	TRAFO_AXES_IN_10[n]	Axis assignment for transformation 10 [axis index]
24484	TRAFO_GEOAX_ASSIGN_TAB_10[n]	Assignment geometry axis to channel axis for transformation 10 [geometry no.]
24570	TRAFO5_AXIS1_1[n]	Vector for the first rotary axis and the first orientation transform. [n = 0.. 2]
24572	TRAFO5_AXIS2_1[n]	Vector for the second rotary axis and the first transformation [n = 0.. 2]
24574	TRAFO5_BASE_ORIENT_1[n]	Basic orientation for the first transformation [n = 0.. 2]
24576	TRAFO6_BASE_ORIENT_NORMAL_1[n]	Tool normal vector for the first transformation [n = 0.. 2]
24585	TRAFO5_ORIAX_ASSIGN_TAB_1[n]	Assignment of orientation axes to channel axes for orientation transformation 1 [n = 0.. 2]
24590	TRAFO5_ROT_OFFSET_FROM_FR_2	Offset of transf. rotary axes from WO
24594	TRAFO7_EXT_ROT_AX_OFFSET_1	Angle offset of the 1st external rotary axis
24595	TRAFO7_EXT_AXIS1_1	Direction of the 1st external rotary axis
24664	TRAFO5_NUTATOR_AX_ANGLE_2	Angle of the 2nd rotating axis for the universal milling head
24670	TRAFO5_AXIS1_2[n]	Vector for the first rotary axis and the second orientation transformation [n = 0.. 2]
24672	TRAFO5_AXIS2_2[n]	Vector for the second rotary axis and the first transformation [n = 0.. 2]
24674	TRAFO5_BASE_ORIENT_2[n]	Basic orientation for the second transformation [n = 0.. 2]
24676	TRAFO6_BASE_ORIENT_NORMAL_2[n]	Tool normal vector for the second transformation [n = 0.. 2]
24685	TRAFO5_ORIAX_ASSIGN_TAB_2[n]	Assignment of orientation axes to channel axes for orientation transformation 2 [n = 0.. 2]
24694	TRAFO7_EXT_ROT_AX_OFFSET_2	Angle offset of the 2nd external rotary axis
24695	TRAFO7_EXT_AXIS1_2	Direction of the 2nd external rotary axis
25294	TRAFO7_EXT_ROT_AX_OFFSET_3	Angle offset of the 3rd external rotary axis
25295	TRAFO7_EXT_AXIS1_3	Direction of the 3rd external rotary axis

Number	Identifier: $MC_	Description
25394	TRAFO7_EXT_ROT_AX_OFFSET_4	Angle offset of the 4th external rotary axis
25395	TRAFO7_EXT_AXIS1_4	Direction of the 4th external rotary axis
28580	MM_ORIPATH_CONFIG	Configuration for path relative orientation ORIPATH

3.14.2 Setting data

3.14.2.1 General setting data

Number	Identifier: $SN_	Description
41110	JOG_SET_VELO	Geometry axes
41130	JOG_ROT_AX_SET_VELO	Orientation axes

3.14.2.2 Channelspecific setting data

Number	Identifier $SC_	Description
42475	COMPRESS_CONTOUR_TOL	Maximum contour deviation for the compressor
42476	COMPRESS_ORI_TOL	Maximum angular deviation of the tool orientation for the compressor
42477	COMPRESS_ORI_ROT_TOL	Maximum angular deviation when rotating the tool for the compressor
42650	CART_JOG_MODE	Coordinate system for Cartesian manual travel
42660	ORI_JOG_MODE	Definition of virtual kinematics for JOG
42670	ORIPATH_SMOOTH_DIST	Smoothing path of the orientation
42672	ORIPATH_SMOOTH_TOL	Tolerance when smoothing the orientation
42970	AA_OFF_LIMIT	Upper limit for offset value $AA_TOFF

K12 transformation definitions with kinematic chains 4

4.1 Function description

4.1.1 Characteristics

In this chapter, a description is provided as to how transformations are mapped using a kinematic chain and parameterized in the control system using system variables. The system variables are retentatively saved in the NC and can be archived and/or read as "NC data" via SINUMERIK Operate using the commissioning archive.

Transformations via kinematic chains

The machine kinematics relevant to kinematic transformations is described by two kinematic chains:

- One chain points from the machine zero point (zero point of the world coordinate system) to the workpiece reference point (part segment).
- The other chain points from the machine zero point to the tool reference point (tool segment).

One of the two chains can also be empty (for pure table or tool kinematics).

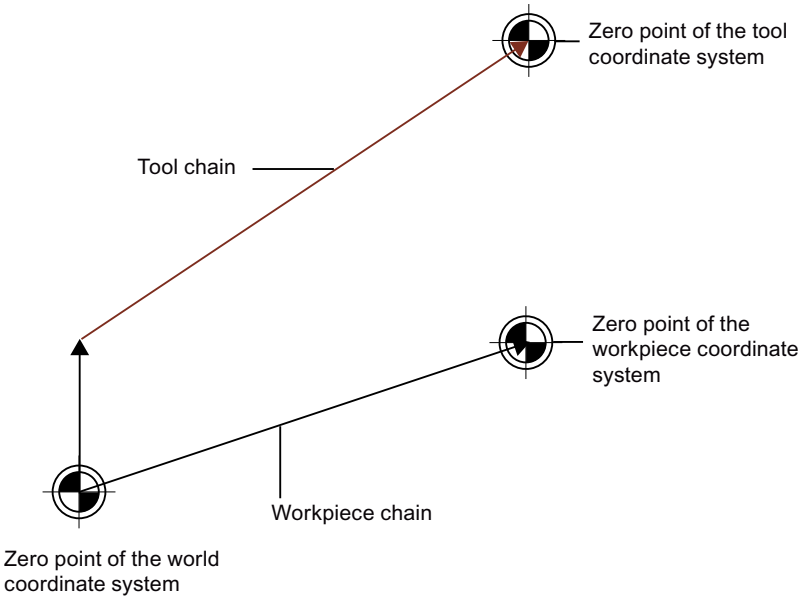


Figure 4-1 Definition of a transformation via kinematic chains

Both kinematic chains are merged internally into a single kinematic chain, which, by definition, begins at the workpiece reference point and ends at the tool reference point.

Definition of the kinematic transformation

The definition of kinematic transformations via kinematic chains is used to standardize the definition of the previous transformation types.

The following transformations are taken into consideration:

- TRAORI
  - TRAORI_DYN; dynamic orientation transformation
  - TRAORI_STAT; static orientation transformation
- TRANSMIT_K; face side transformation
- TRACYL_K; peripheral surface transformation
- TRAANG_K; oblique angle transformation

4.1.2 Definition of kinematic transformations

Transformations with kinematic chains are defined in two steps:

- Definition of the machine kinematics
- Definition of the transformation

Defining machine kinematics

The machine kinematics (geometry of the machine) is described via kinematic chains. For detailed information, see Function Manual "Basic Functions", Section "Kinematic chains".

The relevant system variables are described in the Function Manual "Basic Functions", Section "Kinematic chains". These system variables are structured according to the scheme $NK_... Important system variables include:

System variable	
$NK_NAME	Name of the current element
$NK_NEXT	Name of the next element
$NK_PARALLEL	Defines a parallel kinematic chain. As a result, the second chain can be defined parallel to the first chain.
$NK_TYPE	Defines the type of the element, e.g. AXIS_LIN, AXIS_ROT, OFFSET, ROT_CONST, SWITCH
$NK_OFF_DIR	Defines the rotation vector relative to the machine axis
$NK_AXIS	Name of the machine axis
$NK_A_OFF	Defines the work offset of the assigned machine axis.

Defining transformation

The description of the machine kinematics with kinematic chains is not sufficient to completely specify a kinematic transformation. The transformations can be completely defined via system variables with the $NT_... prefix. The system variables available for the transformation are divided into the following parts:

- System variables that are relevant for all transformations.
- System variables that are only relevant for specific transformations.

You can find a complete list under System variables for general transformation types (Page 278). Among other things, the following general system variables are available:

System variable	Description
$NT_NAME	Name of the transformation dataset.
$NT_TRAFO_INDEX	Identifier for alternative transformation activation; a transformation defined with kinematic chains can also be activated using the conventional language commands (e.g. TRAORI) instead of the TRAFOON method.
$NT_TRAFO_TYPE	Type of transformation (TRAORI_DYN, TRAORI_STAT, TRAANG_K, TRANSMIT_K, TRACYL_K)
$NT_T_CHAIN_LAST_ELEM	Name of the last element of the kinematic chain to the **tool**.
$NT_P_CHAIN_LAST_ELEM	Name of the last element of the kinematic chain to the **workpiece**.
$NT_T_CHAIN_FIRST_ELEM	Name of the first element in the tool chain
$NT_P_CHAIN_FIRST_ELEM	Name of the first element of the workpiece chain
$NT_GEO_AX_NAME	Names of the elements of the geometry axes. With $NT_GEO_AX_NAME, a maximum of 3 linear axes are referenced. The linear axes carry out the compensating movements which result from the kinematic transformation.
$NT_ROT_AX_NAME	Names of the elements of the orientation axes. With $NT_ROT_AX_NAME, a maximum of 3 linear axes are referenced.
$NT_CLOSE_CHAIN_T	Reference element for closing the kinematic chain to the workpiece.
$NT_ROT_AX_OFFSET	Position offset of a/the rotary axis for the active transformation.
$NT_CNTRL	Control word

An orientation transformation is defined, for example, via the elements displayed in the graphic.

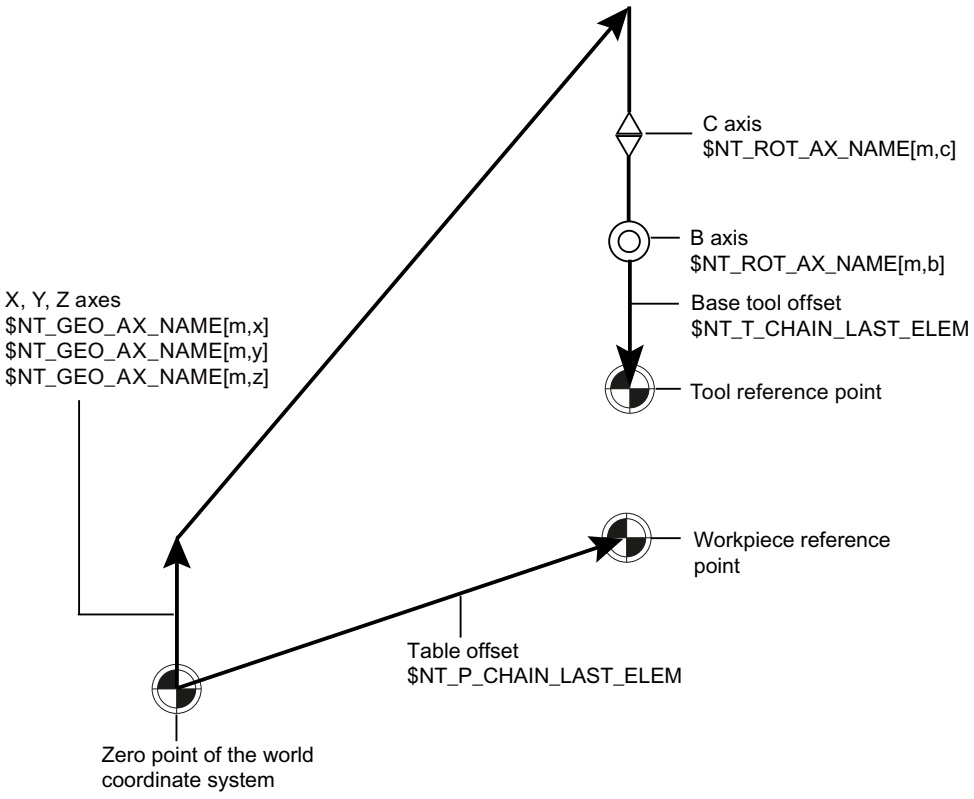


Figure 4-2 Transformation elements in the kinematic chain

See also

Examples (Page 325)

4.1.3 Dynamic orientation transformation TRAORI_DYN

A dynamic orientation transformation is understood to mean a kinematic transformation in which the movements of any rotary axes are compensated by compensation movements of a maximum of three linear axes in such a way that the coordinates of the tool tip in the workpiece coordinate system remain unchanged. The same applies for movements of additional (redundant) linear axes.

The system variables that are especially used for orientation transformations are described under Additive system variable for orientation transformation (Page 303).

The different orientation transformations are mapped via the dynamic orientation transformation:

5-axis transformation

Different versions are possible for the 5-axis transformation, see 5-axis transformation (Page 145). Generally this transformation is comprised of up to 3 linear axes and up to 2 rotary axes.

3 and 4-axis transformation

3 and 4-axis transformations are special cases of the 5-axis transformation, see 3-axis and 4-axis transformations (Page 160). Possible versions are listed below:

Transformation	Characteristics
3-axis transformation	2 translation axes (linear axes) 1 rotary axis
4-axis transformation	3 translation axes (linear axes) 1 rotary axis

Defining machine kinematics

The corresponding machine kinematics must be defined via a kinematic chain.

Linear axes (geometry axes)

The movements of the rotary axes are compensated via compensating movements of up to three linear axes. The linear axes are defined via the kinematic chain as an element of the type "AXIS_LIN". For the transformation, they are referenced with $NT_GEO_AX_NAME. The geometry axes do not have to be vertical on top of one another and their position relative to one another can be changed by means of rotary axes. However, the limitation that they cannot be linearly dependent on one another applies.

The position of linear axes that are not defined as geometry axes is not changed by the transformation.

Rotary and orientation axes

The rotary axes are defined via the kinematic chain as an element of the type "AXIS_ROT". For the transformation, they are referenced with $NT_ROT_AX_NAME and are thus designated as orientation axis. The position of these axes is assumed to be changeable to achieve a specific orientation.

An offset from zero can be defined using system variable `$NT_ROT_AX_OFFSET` for the rotary axes (1, 2, or 3) of the orientation transformation.

All of the other rotary axes are redundant rotary axes and are included in the calculation with their current position values (constant rotations). Redundant machine axes (real rotary axes) must be located in the kinematic chain in front of the first orientation axis. Additional redundant rotary axes (manual rotary axes or rotating machine axes) can be inserted at any point.

The number of rotary axes is restricted to 6.

Tool orientation

The tool orientation can be specified as vector. The system variables $NT_BASE_ORIENT or $NT_BASE_ORIENT_NORMAL are available for this. The orientations come into effect if no tool is selected. If the vectors are not explicitly defined, the default setting is (0, 0, 1).

- $NT_BASE_ORIENT[n, 0 ...2]; defines the basic tool orientation via a vector, as shown in the following example:
  - $NT_BASE_ORIENT[1, 0] = 1.0; component in X-direction
  - $NT_BASE_ORIENT[1, 1] = 1.5; component in Y-direction
  - $NT_BASE_ORIENT[1, 2] = 100; component in Z-direction
- $NT_BASE_ORIENT_NORMAL[n, 0 ...2] defines a vector which stands vertically on the basic tool orientation for orientation transformations with three degrees of freedom. If the vectors are not explicitly defined, the default setting is (0, 1, 0).
- $NT_IGNORE_TOOL_ORIENT[n]; if the system variable is set, the values from the system variables $NT_BASE_ORIENT and $NT_BASE_ORIENT_NORMAL are also used when a tool is active.

Treatment of singular points (poles)

If poles (Page 157) are encountered over the course of processing, the behavior at the pole can be set via two system variables:

- $NT_POLE_LIMIT; defines the end angle tolerance, i.e. the angle by which the polar axis can deviate from the programmed value for the 5-axis transformation.
- $NT_POLE_TOL; defines the end angle tolerance for interpolation through the pole.

Real-time velocity monitoring

The machine data $MC_TRAFO_ORI_ONLINE_CHECK_LIM and $MC_TRAFO_ORI_ONLINE_CHECK_LIMR define the linear or rotary position difference between the pre-run and main run as of which the real-time velocity monitoring and limiting is activated. The position differences are formed from the positions of the linear and rotary axes.

Use of CYCLE832 (High Speed Settings)

When Cycle832 with transformations is used, the rotary axes are determined in the following way:

- The rotary axes that are contained in the first transformation with a kinematic chain of the type TRAOR_DYN are entered.
- If there is no transformation with a kinematic chain, the rotary axes that are contained in the last transformation defined via machine data are entered.

The reference radius for both rotary axes is then set via FGREF.

Machine measurement for orientation transformation

With the CORRTRAFO (Page 318) function, measured lever arm lengths and axis directions can be written in correction elements:

- $NT_CORR_ELEM_T[n, 0 ...3]; names of the correction elements (type "OFFSET") in the kinematic chain (**tool chain**) which can be written to by CORRTRAFO.
- $NT_CORR_ELEM_P[n, 0 ...3]; names of the correction elements (type "OFFSET") in the kinematic chain (**workpiece chain**) which can be written to by CORRTRAFO.

Example of kinematics

The example schematically shows two-axis swivel head kinematics with the axis sequence CA, as a spatial image and as a kinematic chain.

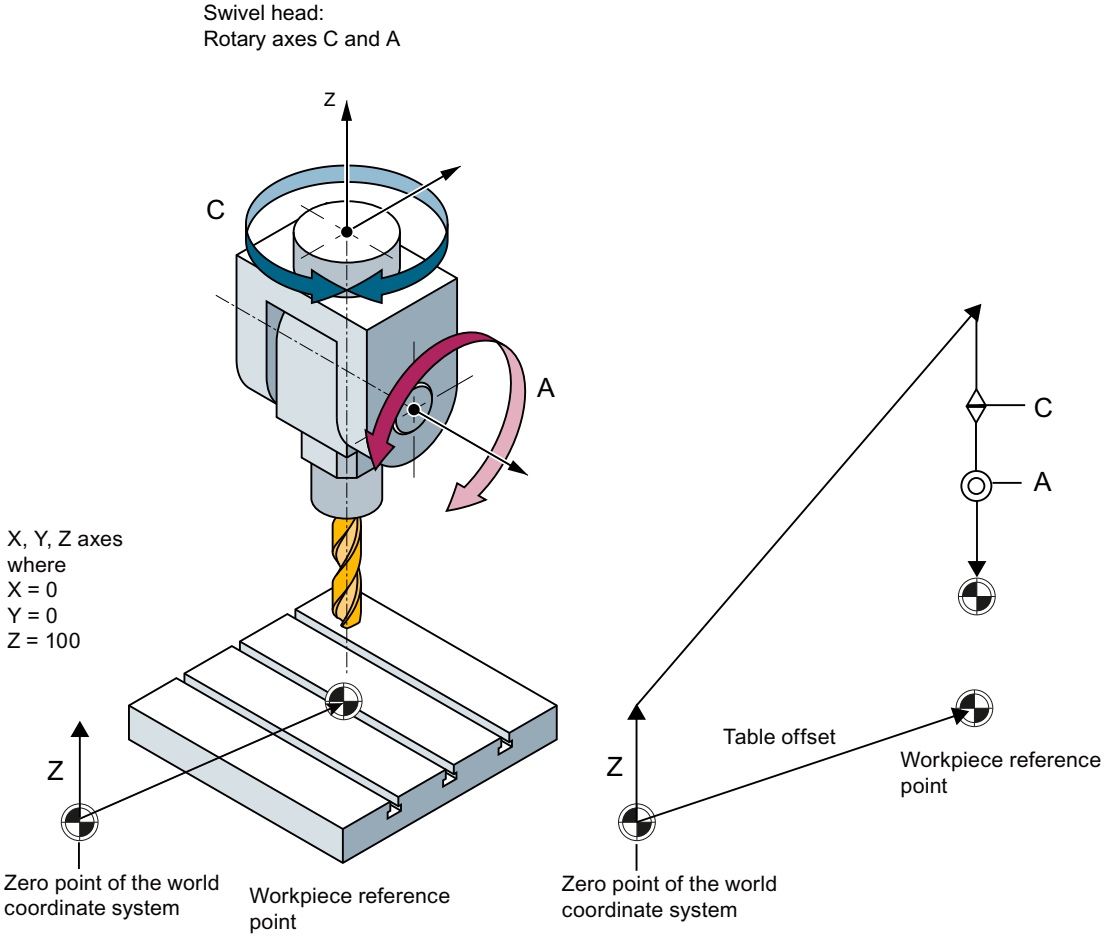


Figure 4-3 TRAORI example

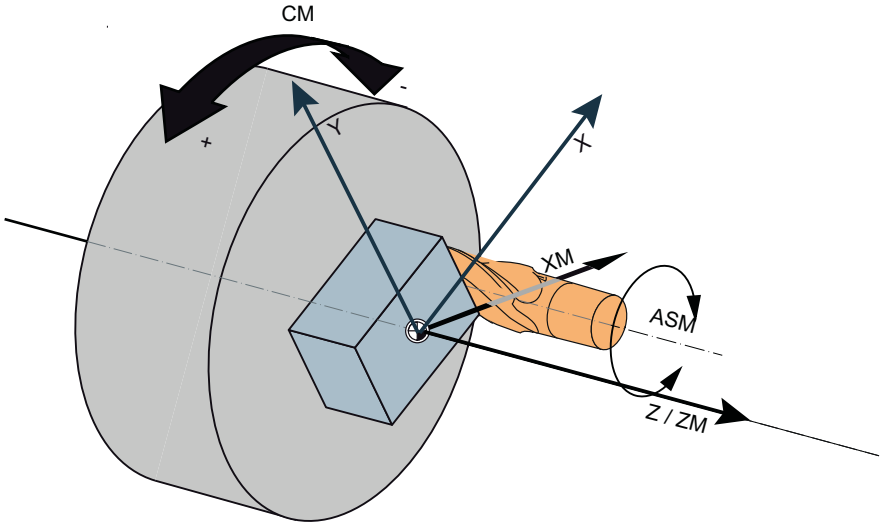
4.1.4 Face end transformation TRANSMIT

In this chapter, a description is provided as to how a end face transformation is mapped using a kinematic chain and parameterized in the control system using system variables. The TRANSMIT transformation permits end face machining (drill holes, contours) on turning machines.

The system variables are retentively saved in the NC and can be archived and/or read as "NC data" via SINUMERIK Operate using the commissioning archive.

The machine kinematics are defined using two kinematic chains, see Definition of kinematic transformations (Page 256).

The following graphic shows an example of a end face transformation:



X, Y, Z	Geometry axes
CM	Machine axis; rotary axis
XM	Machine axis; linear axis, perpendicular to rotary axis
ZN	Machine axis; linear axis, parallel to rotary axis
ASM	Machine axis; work spindle

Figure 4-4 TRANSMIT

Supplementary conditions

- There must be exactly one rotary axis (the polar axis), with its name in $NT_ROT_AX_NAME[n,1].
- One to three entries for linear axes are possible in the system parameters $NT_GEO_AX_NAME[n,i].

Coordinate systems for end face transformations

The workpiece coordinate system and the basic coordinate system are independent of the world coordinate system in which the kinematic chains are defined.

- The Z axis is parallel to the polar axis (axis CM - see above) and parallel to the longitudinal axis, if it is defined.
- The geometry axes X, Y and Z are vertical to one another and form a perpendicular coordinate system in the order 1st, 2nd and 3rd geometry axis.
- If the longitudinal axis ($NT_GEO_AX_NAME[n, 2]) exists, it must always be parallel to the rotary axis ($NT_ROT_AX_NAME[n, 1]).
- If there are two linear axes, they do not have to be orthogonal to one another.
- If there are three linear axes, two axes must be orthogonal to one another. A third axis can be at an oblique angle to one of the other two axes. There can only be one axis pair in which the angle of the two axes is not equal to 90°. An oblique axis cannot be positioned perpendicularly on another axis.

Kinematic chains for end face transformations

The transmit transformation is activated via the system variable $NT_TRAFO_TYPE = "TRANSMIT_K".

Definition of the linear axes

- The X-axis is defined via the system variable $NT_GEO_AX_NAME[n,0]. This is the linear axis of the actual TRANSMIT transformation. In the standard scenario, this axis is vertical on the polar axis (see above).
- An entry in $NT_GEO_AX_NAME[n, 2] is optional. This axis (typically the Z-axis) is the longitudinal axis. It must be defined parallel to the rotary axis ($NT_ROT_AX_NAME[n, 1]).
- An entry in $NT_GEO_AX_NAME[n, 1] is optional. This axis (typically the Y-axis) stands vertically on the radial axis ($NT_GEO_AX_NAME[n, 0]) and the polar axis ($NT_ROT_AX_NAME[n, 1]) in the standard scenario.
  This axis (the center offset axis) is generally used to compensate a tool offset in the Y direction so that machining is possible up to the turning center.

Definition of a rotary axis

- The rotary axis is defined via the system variable $NT_ROT_AX_NAME[n,1]. This axis must always be available. The direction of rotation of the rotary axis is defined via $NT_CNTRL Bit 11.
- An offset can be defined via system variable $NT_ROT_AX_OFFSET[n, 0] for the active TRANSMIT transformation.

Defining the order of the axes

Various settings for the transformation are defined using system variable $NT_CNTRL[n] (Page 298). With the bits 16-18, it is binary coded how the channel axes that are entered in $NT_ROT_AX_NAME[n, 1], in $NT_GEO_AX_NAME[n. 0] and in $NT_GEO_AX_NAME[n. 2]

are assigned to the geometry axes. The three binary numbers must correspond to the following decimal numbers:

Decimal	Order of the geometry axes		
0	X	Y	Z
1	Z	X	Y
2	Y	Z	X
3	Z	Y	X
4	X	Z	Y
5	Y	X	Z

Behavior when passing through the pole

The behavior when passing through the pole is defined via the system variable $NT_POLE_SIDE_FIX

VALUE	Meaning
0	Pole crossing The tool center point path (linear axis) must cross the pole on a continuous path.
1	Rotation around the pole. The tool center point path must be restricted to a positive traversing range of the linear axis (in front of turning center).
2	Rotation around the pole. The tool center point path must be restricted to a negative traversing range of the linear axis. (behind turning center).

Offset for the base tool

The system variable $NT_BASE_TOOL_COMP (Page 302) can be used to set whether an offset is entered for the base tool in the transformation frame ($P_TRAFRAME) upon activation of the transformation for each separate geometry axis. With the offset, the components of the base tool are compensated in such as way that no change occurs in the component of the WCS. For this function, $P_TRAFRAME must be configured via the machine data $MC_MM_SYSTEM_FRAME_MASK bit 6.

The compensation can, for example, be used for a tool carrier, where the dimensions of the head part of the tool carrier are not part of the tool length.

The separating point between the base tool and the tool can be defined via the system variable $NT_T_REF_ELEM. If the system variable is blank, then the separating point is defined at the end point of the tool chain.

Work offset of the rotary axis

In system variable $NT_ROT_OFFSET_FROM_FRAME (Page 294), you set whether the offset for orientation axes is to be automatically applied. If a value is defined in the system variable, then the programmed offset is automatically imported upon switch-on from the work offset that is active for orientation axes:

Importing the rotary axis offset from the work offset upon selection of the transformation:

- 0 = Axial offset of the rotary axis is not taken into account.
- 1 = axial offset of the rotary axis is taken into account.
- 2 = Axial offset of the rotary axis is taken into account up to the SZS. SZS is the settable zero system. The SZS frames contain transformed rotations around the rotary axis.

Definition of the last element in the kinematic chain

$NT_CNTRL Bit 19 is used to define that the last element of the kinematic chain is a rotary axis or a constant rotation. The direction vector of the rotary axis defines the Z direction of the tool coordinate system. If the system variable $NK_A_OFF of this chain element contains a value not equal to zero, the tool coordinate system is also rotated about the coordinate axis with this angle.

If $NT_CNTRL Bit 20 is also defined, the sign of the Z direction of the axis for the determination of the WCS is inverted.

Example program for TRANSMIT

You will find a simple part program using TRANSMIT under Part program for TRANSMIT (Page 332).

4.1.5 Cylinder surface transformation TRACYL

In this chapter, a description is provided as to how cylinder surface transformations (TRACYL) are mapped using a kinematic chain and parameterized in the control system using system variables.

The system variables are retentively saved in the NC and can be archived and/or read as "NC data" via SINUMERIK Operate using the commissioning archive.

The machine kinematics are defined using two kinematic chains, see Definition of kinematic transformations (Page 256).

The following graphic shows an example of a cylinder surface transformation:

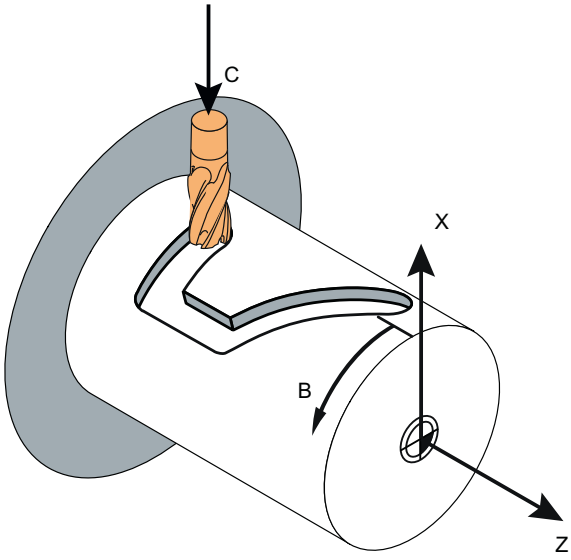


Figure 4-5 Cylinder surface transformation (TRACYL)

Activating a transformation with TRAFOON

In addition to the name of the transformation, the reference diameter can also be specified for TRACYL and a setting as to whether the "Slot side offset" function is active is also possible:

```
TRAFOON (<transformation name>, <diameter>, <k>)
```

With k = 1, the slot side offset is carried out, see Activating a transformation (TRAFOON) (Page 317).

Supplementary conditions

- There must be exactly one rotary axis (the polar axis), with its name in $NT_ROT_AX_NAME[n,1].
- There can be 1 to 3 entries for linear axes in the system parameters $NT_GEO_AX_NAME[n,i].

Coordinate systems for cylinder surface transformations

The workpiece coordinate system and the basic coordinate system are independent of the world coordinate system in which the kinematic chains are defined.

- The Z axis is parallel to the polar axis (axis B - see above) and parallel to the longitudinal axis, if it is defined.
- The geometry axes X, Y and Z are vertical to one another and form a perpendicular coordinate system in the order 1st, 2nd and 3rd geometry axis.
- If the longitudinal axis ($NT_GEO_AX_NAME[n, 2]) exists, it must always be parallel to the rotary axis ($NT_ROT_AX_NAME[n, 1]).

- If there are two linear axes, they do not have to be orthogonal to one another.
- If there are three linear axes, two axes must be orthogonal to one another. A third axis can be at an oblique angle to one of the other two axes. There can only be one axis pair in which the angle of the two axes is not equal to 90°. An oblique axis cannot be positioned perpendicularly on another axis.

Kinematic chains for cylinder surface transformations

The cylinder surface transformation is activated via the system variable $NT_TRAFO_TYPE = "TRACYL_K".

Definition of the linear axes

- The X-axis is defined via the system variable $NT_GEO_AX_NAME[n,0]. This is the linear axis of the actual TRACYL transformation. In the standard scenario, this axis is vertical on the rotary axis (see above). The slot depth is set at this axis.
- An entry in $NT_GEO_AX_NAME[n, 1] is optional. This axis (typically the Y-axis) stands vertically on the radial axis ($NT_GEO_AX_NAME[n, 0]) and the polar axis ($NT_ROT_AX_NAME[n, 1]) in the standard scenario.
  This axis is the slot side offset axis. With this axis, the offset at the extent of the machining cylinder is compensated when the tool radius compensation is active, so that the slot sides remain perpendicular to the slot base.
- An entry in $NT_GEO_AX_NAME[n, 2] is optional. This axis (typically the Z-axis) is the longitudinal axis. It must be defined parallel to the rotary axis ($NT_ROT_AX_NAME[n, 1]).

Definition of a rotary axis

- The rotary axis is defined via the system variable $NT_ROT_AX_NAME[n,1]. This axis must always be available. The direction of rotation of the rotary axis is defined via $NT_CNTRL Bit 11.
- An offset can be defined via system variable $NT_ROT_AX_OFFSET[n, 0] for the active TRACYL transformation.

Defining the order of the axes

Various settings for the transformation are defined using system variable $NT_CNTRL[n] (Page 297). With the bits 16-18 it is binary coded how the channel axes that are entered in $NT_ROT_AX_NAME[n, 1], $NT_GEO_AX_NAME[n. 0] and $NT_GEO_AX_NAME[n. 2] are assigned to the geometry axes. The three binary numbers must correspond to the following decimal numbers:

Decimal	Order of the geometry axes		
0	X	Y	Z
1	Z	X	Y
2	Y	Z	X
3	Z	Y	X
4	X	Z	Y
5	Y	X	Z

Slot side offset

In connection with bit 9 = 1, you can set whether TRACYL is operated with or without slot side offset upon activation of the transformation via TRAFOON. "Slot side active" corresponds to transformation type 514.

If you are working with slot side offset, $NT_GEO_AX_NAME[n,1] must contain a reference to a linear axis (slot side offset axis).

The following settings via $NT_CNTRL are possible with bit 9 and bit 10:

Bit 10	Bit 9	Description
0	0	TRACYL without slot side offset
0	1	TRACYL with slot side offset
1	0	TRACYL with the capability of switching between the functions with and without slot side offset via a transformation parameter upon activation of the transformation (TRAFOON). The slot side offset is inactive by default.
1	1	TRACYL with the capability of switching between the functions with and without slot side offset via a transformation parameter upon activation of the transformation (TRAFOON). The slot side offset is active by default.

Work offset of the rotary axis

Setting is done via the system variable $NT_ROT_OFFSET_FROM_FRAME (Page 294). If a value is defined in the system variable, then the programmed offset is automatically imported upon switch-on from the work offset that is active for orientation axes:

Importing the rotary axis offset from the work offset upon selection of the transformation:

- 0 = Axial offset of the rotary axis is not taken into account.
- 1 = axial offset of the rotary axis is taken into account.
- 2 = Axial offset of the rotary axis is taken into account up to the SZS. SZS is the settable zero system. The SZS frames contain transformed rotations around the rotary axis.

Definition of the last element in the kinematic chain

$NT_CNTRL Bit 19 is used to define that the last element of the kinematic chain is a rotary axis or a constant rotation. The direction vector of the rotary axis defines the Z direction of the tool coordinate system. If the system variable $NK_A_OFF of this chain element contains a value not equal to zero, the tool coordinate system is also rotated about the coordinate axis with this angle.

If $NT_CNTRL Bit 20 is also defined, the sign of the Z direction of the axis for the determination of the WCS is inverted.

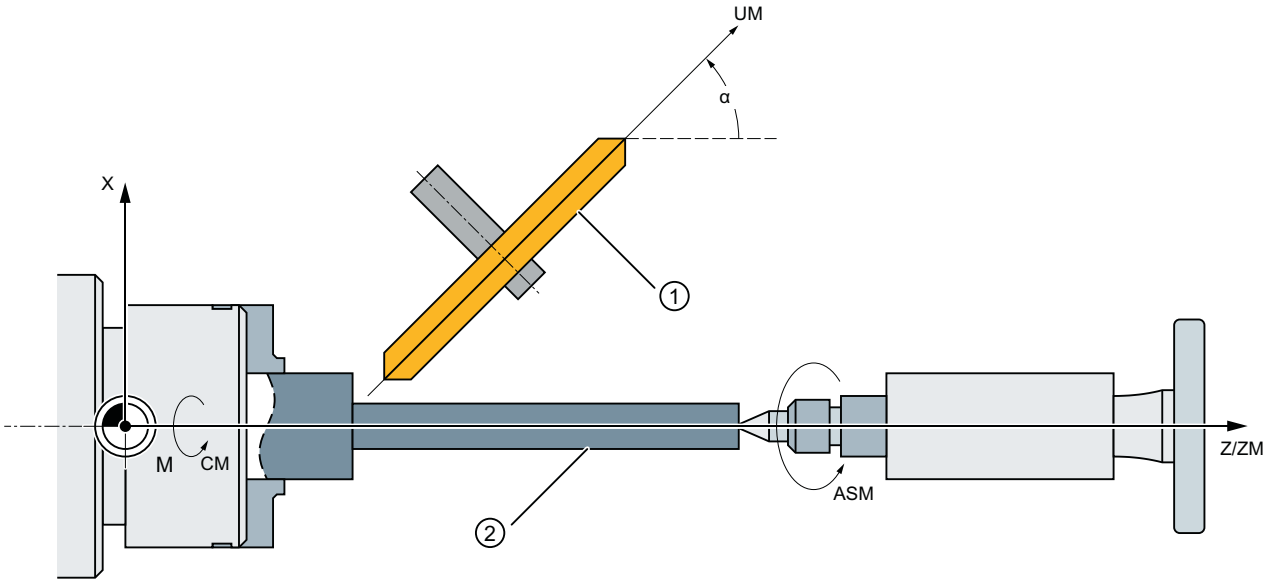
Example program for TRACYL

You will find a simple part program using TRACYL under Part program for TRACYL (Page 335).

4.1.6 Oblique angle transformation (inclined axis) TRAANG_K

In this chapter, a description is provided as to how an oblique angle transformation (TRAANG_K) is mapped using a kinematic chain and parameterized in the control system using system variables.

The system variables are retentively saved in the NC and can be archived and/or read as "NC data" via SINUMERIK Operate using the commissioning archive.



①	Grinding disk
②	Workpiece
X	Geometry axis
Z	Geometry axis
ZM	Machine axis
UM	Machine axis
α	Angle of inclined axis
CM	Machine axis (spindle)
ASM	Machine axis (tailstock)

Figure 4-6 Inclined axis

The oblique angle transformation is a special case of orientation transformation.

The oblique angle transformation is activated using system variable $NT_TRAFO_TYPE = "TRAANG_K".

Program example for TRAANG

You will find a simple part program using TRAANG under Part program for TRAANG (Page 338).

4.1.7 Static orientation transformation TRAORI_STAT

Static orientation transformations differ from dynamic orientation transmissions in that it is not possible to carry out interpolator compensating movements in such a way that the tool tip adheres to the path programmed in the NC program in the workpiece coordinate system. However, all of the functions are available for approaching the correct end points, taking the required compensation movements into consideration.

A static orientation formation makes sense for machines in which the orientation is fixed once it is set and cannot be changed during the process, for example when rotating on a milling machine.

The static orientation transformation is defined using the same system variables as a dynamic orientation transformation, see Dynamic orientation transformation TRAORI_DYN and Definition of kinematic transformations (Page 256).

Axes for a static orientation transformation

The orientation axes of static orientation transformations can also be spindles or Hirth-coupled axes. Spindles behave like constant kinematic rotations, i.e. if a position is to be assigned to them in the kinematic transformation, this must be entered in the associated system parameters $NK_A_OFF[n].

The type "Static orientation transformation" is activated via the system variable §NT_TRAFO_TYPE = TRAORI_STAT.

System variables for the definition of the static orientation transformation

The machine kinematics are defined via the kinematic chain, see Definition of kinematic transformations (Page 256).

The static orientation transformation is defined via the following system variables:

- System variables for general transformation types (Page 278)
- Additive system variable for orientation transformation (Page 303)

Reference

Information on orientable tool carriers via kinematic chains can be found in the "Basic Functions" Function Manual in section "Tool offset" under "Orientable tool carriers".

See also

Dynamic orientation transformation TRAORI_DYN (Page 258)

4.1.8 Concatenating transformations (TRACON_K)

In transformations with kinematic chains, non-orthogonal axes can be defined via the kinematic chain. This means that transformations no longer have to be concatenated with TRACON to map non-orthogonal axes.

Transformation type "TRACON_K" (Page 281) is used in combination with kinematic chains in order to chain an "Open Architecture (OA)" transformation with one of the standard transformations. More detailed information is provided in the "Open Architecture (OA)" documentation.

Further information

An exhaustive description of the chaining of transformations without kinematic chains can be found in Chained transformations.

See also

$NT_TRACON_CHAIN (Page 316)

Chained transformations (Page 81)

4.1.9 Effective transformations upon reset

Active transformation upon reset

The kinematic transformation that is in effect during a reset is defined via the machine data $MC_TRAFO_RESET_NAME.

If no name has been defined for $MC_TRAFO_RESET_NAME, $MC_TRAFO_RESET_VALUE is active or no transformation is active upon reset. The contents of this machine data specifies the number of the conventional transformation to be activated. Conventional transformations are not defined via kinematic chains, but via machine data.

Furthermore, the machine data $MC_RESET_MODE_MASK Bit 7 acts as it does for conventional transformations. If bit 7 is set, the active transformation is retained after the reset or the end of the part program.

4.1.10 Persistent transformations

Persistent transformation

A persistent transformation is a transformation which is retained even when another (different) transformation is switched off with TRAFOOF. This transformation is identical to the transformation that is in effect during the reset (Page 271).

If a different transformation is selected, the persistent transformation is replaced by the newly selected transformation. The newly selected transformation is NOT concatenated with the persistent transformation.

4.1.11 Simulation/simultaneous recording with kinematic chains

Machine data $MNS_FUNCTION_MASK_SIM

To correctly carry out the simulation and simultaneous recording, the machine data MD 51226 $MNS_FUNCTION_MASK_SIM Bit 22 must be set.

The transformations TRAORI_DYN, TRANSMIT_K, TRACYL_K and TRAANG_K are affected by this.

4.1.12 Migration of kinematic transformations

Conventional transformations

Conventional transformations are understood to mean transformations that are defined via machine data. These transformation definitions remain valid and can be used without restrictions.

The variant available as of V4.8. SP2 also allows kinematic transformations to be defined via kinematic chains and system variables.

Managing data only once

For the parallel use of conventional transformations and transformations via kinematic chains, the problem occurs in which identical machine kinematics may be parameterized with different NC data, namely, once with machine data and once with data from the system variables of the kinematic chains and the transformations. As a result, inconsistencies in the data management can occur.

Activating a transformation

The following functions are available for the activation of kinematic transformations:

- Syntax for transformations via kinematic chains: TRAFOON (Page 317) (<name>)
- Syntax for conventional transformations: TRAORI(...), TRANSMIT(...), TRACYL(...), TRAANG(...).

Transformations which have been defined via kinematic chains can be called via the syntax for conventional transformations.

To this end, the transformation that is to be called must be defined via the system variable $NT_TRAFO_INDEX (Page 280).

The following also applies:

- If a transformation has been defined conventionally and with a kinematic chain, the transformation is called via a kinematic chain.
- No check is made to determine whether the called transformation is of a type that is compatible with the transformation type of the original call.
- Call parameters of a conventional transformation call are forwarded to the transformation defined with kinematic chains.

Restrictions for TRAANG_K

For a conventional TRAANG transformation (transformation of "oblique axis"), the angle of the oblique axis is defined opposing the corresponding coordinate axis in a perpendicular coordinate system by means of machine data (24700 $MC_TRAANG_ANGLE_x).

However, this angle can be changed during a transformation call, e.g. TRAANG(<α>), in a parameter.

Therefore, specifying the angle parameter is only permitted when this angle is identical to the angle that results from the definition of the kinematic chain.

Orientation transformations

For orientation transformations, which are parameterized with kinematic chains, the content of the system variable $P_TRAFO is defined for compatibility reasons.

With an active orientation transformation, $P_TRAFO provides the values that would be contained in the machine data $MC_TRAFO_TYPE_x for the conventional parameterization of a corresponding transformation, i.e. the value 24 for "tool" kinematics, the value 40 for "workpiece" kinematics, and either the value 56 or 57 for mixed kinematics.

The value 57 can only occur with 6-axis transformations if the tool chain contains one orientation axis and the workpiece chain contains two orientation axes.

4.1.13 Frames for kinematic transformations

Frames for the description of kinematic chains

With the system variables $P_TRAFRAME_T or $P_TRAFRAME_P, frames can be read out which describe the offset and the rotation of a coordinate system secured at the tool reference point or at the workpiece reference point, compared to the zero point of the world coordinate system.

The contents of these frame variables can also be read via the BTSS interface.

The zero point of the world coordinate system is the point from which the kinematic chains (Page 255) are defined for describing the machine kinematics.

4.1.14 Tool lengths

Tool lengths

If the portion of the kinematic chain that leads to the tool is part of the tool, then the tool reference point and the tool reference position that was set via $NT_T_REF_ELEM are no longer identical.

The following applies to the NC functions, GETTCORR, SETTCORR and LENTOAX, for which the tool reference point is important:

The portion of the kinematic chain that lies between the tool reference position and the tool reference point is handled like an orientable tool carrier, which has been defined via machine data.

4.2 Commissioning

4.2.1 General

4.2.1.1 Overview

The startup of the function "Transformations via kinematic chains" is done by means of:

- Machine data
  - Specification of the quantity structure
  - Specification of the first element in the kinematic chain
- System variables
  - Definition of the transformation type
  - Specification of the kinematic properties of an element
  - Connection of the element to the kinematic chain

4.2.1.2 Structure of the system variables

The system variables are structured according to the following scheme:

- **$NT_**<Name>[<Index_1>]
- **$NT_**<Name>[<Index_1>, <Index_2>]

General

The system variables to describe the elements of kinematic chains have the following properties:

- The prefix for all of the system variables of the kinematic chain is **$NT_**, (N for NC, K for transformation).
- The system variables can be read and written via NC programs.
- The system variables can be stored in archives and loaded to the NC again.

Data type

STRING

All system variables of the STRING data type have the following properties:

- Maximum string length: 31 characters
- No distinction is made between upper and lower case
  Example: "Axis1" is identical to "AXIS1"
- Spaces and special characters are permitted
  Example: "Axis1" is not identical to "Axis 1"
- Names that **start** with **two** underscores "__" are reserved for system purposes and must **not** be used for user-defined names.

Note

Leading space

Since spaces are valid and distinct characters, names that **start** with a **space** followed by **two** underscores "__" can, in principle, be used for user-defined names. However, because they can be easily mistaken for system names, this procedure is **not** recommended.

Index_1

The individual elements are addressed via index_1. Index 0 is reversed, index 1 → 1. Element, index 2 → 2. Element, ... (n+1) → (n+1) element, with n = ($MN_MM_NUM_KIN_TRAFOS - 1)

All system variables of an element have the same index.

Index_2

For system variables that contain a vector, the coordinates of the vector are addressed via index_2.

- 0 → X axis
- 1 → Y axis
- 2 → Z axis

4.2.2 Machine data

4.2.2.1 Maximum number of transformations with kinematic chains

The maximum number of transformations that can be defined with kinematic chains is set using the machine data

MD18866 $MN_MM_NUM_KIN_TRAFOS = <number>

4.2.2.2 Name of the reset transformation

The name of a transformation is specified with the machine data that is selected during run-up/ reset or at the end of the part program.

MD20142 $MC_TRAFO_RESET_NAME = <name>

4.2.2.3 Activation limit of the real-time dynamic monitoring (linear axes)

The activation limit of the real-time dynamic monitoring for linear axes is specified with the machine data. Real-time dynamic limiting is activated when the deviation at a linear axis participating in the transformation or the effective tool length for an orientation transformation deviates by more than the defined limit.

MD21198 $MC_ORI_TRAFO_ONLINE_- CHECK_LIM = <value>

4.2.2.4 Activation limit of real-time dynamic monitoring (rotary axes)

The activation limit of real-time dynamic monitoring for rotary axes is specified with the machine data. Real-time dynamic limiting is activated when the deviation at a linear axis participating in the transformation or the effective tool length for an orientation transformation deviates by more than the defined limit.

MD21198 $MC_ORI_TRAFO_ONLINE_- CHECK_LIMR = <value>

4.2.2.5 Correction value for offset vectors for CORRTRAFO

The maximum permitted correction value for offset vectors for CORRTRAFO is parameterized with the setting data.

SD41610 $SN_CORR_TRAFO_LIN_MAX = <value>

4.2.2.6 Angular deviation for rotation vectors for CORRTRAFO

The maximum permitted angular deviations for rotation vectors for CORRTRAFO is parameterized with the setting data.

SD41611 $SN_CORR_TRAFO_DIR_MAX = <name>

4.2.3 Use of system variables for kinematic transformations

Usage table

The following table shows which $NT data is relevant for the individual types of transformations.

Note

Changed data

Changed data (e.g. $NT_xxx or $NK_yyy) enter into effect in the NC program after `NEWCONF` or after a `RESET` (program end).

Variable	BTSS (column ind.)	TRAORI_STAT	TRAORI_DYN	TRANS-MIT_K	TRAC-YL_K	TRAANG_K
$NT_NAME[n]	1200	x	x	x	x	x
$NT_TRAFO_INDEX[n]	1295	x	x	x	x	x
$NT_TRAFO_TYPE[n]	1201-1203	x	x	x	x	x
$NT_T_CHAIN_LAST_ELEM[n]	1204	x	x	x	x	x
$NT_P_CHAIN_LAST_ELEM[n]	1207	x	x	x	x	x
$NT_T_CHAIN_FIRST_ELEM[n]	1203	x	x	x	x	x
$NT_P_CHAIN_FIRST_ELEM[n]	1206	x	x	x	x	x
$NT_T_REF_ELEM[n]	1208	x	x	x	x	x
$NT_GEO_AX_NAME[n,0..2]	1210-1222	0,1,2	0,1,2	0,1,2	0,1,2	0,1,2
$NT_ROT_AX_NAME[n,0..2]	1220-1222	0,1,2	0,1,2	1	1	-
$NT_CLOSE_CHAIN_T[n]	1226	x	x	x	x	x
$NT_BASE_ORIENT[n,0..2]	1280-1285	x	x	-	-	-
$NT_BASE_ORIENT_NOR-MAL[n,0..2]	1283-1285	x	x	-	-	-
$NT_ROT_OFF-SET_FROM_FRAME[n]	1288	x	x	x	x	-
$NT_POLE_LIMIT[n]	1286	-	x	-	-	-
$NT_POLE_TOL[n]	1287	x	x	-	-	-
$NT_IGNORE_TOOL_ORIENT[n]	1289	x	x	-	-	-
$NT_ROT_AX_POS[n]	1230-1232	x	x	-	-	-
$NT_CORR_ELEM_T[n, 0..3]	1235-1238	x	x	-	-	-
$NT_CORR_ELEM_P[n, 0..3]	1320-1323	x	x	-	-	-
$NT_TRAFO_IN-CLUDES_TOOL[n]	1290	-	-	-	-	x
$NT_POLE_SIDE_FIX[n]	1291	-	-	x	-	-
$NT_CNTRL[n]	1294	x	x	x	x	x
$NT_AUX_POS[n,0..2]	1300-1302	~	~	~	~	~
$NT_IDENT[n, 0..2]	1310-1312	~	~	~	~	~
$NT_ROT_AX_CNT[n, 0..1]	1316-1317	x	x	x	x	x

Variable	BTSS (column ind.)	TRAORI_STAT	TRAORI_DYN	TRANS-MIT_K	TRAC-YL_K	TRAANG_K
$NT_BASE_TOOL_COMP	1396	-	-	x	x	-
$NT_ROT_AX_OFFSET[n, 0....2]	1324-1326	x	x	x	x	-

"x" affects the transformation.

"-" has no effect on the transformation

"~" has no effect on the transformation in the NCK, but can be properly used for purposes outside of NC.

"Numbers" permitted axis indices (0 to 2)

"Column index"; column index of the variables in the operator panel interface in block N_PA (0x35).

4.2.4 System variables for general transformation types

4.2.4.1 Overview

System variable	Meaning
$NT_NAME	Name of the transformation dataset
$NT_TRAFO_INDEX	Identifier for alternative transformation activation
$NT_TRAFO_TYPE	Transformation type
$NT_T_CHAIN_LAST_ELEM	Name of the last element of the tool chain
$NT_P_CHAIN_LAST_ELEM	Name of the last element of the workpiece chain
$NT_T_CHAIN_FIRST_ELEM	Name of the first element in the tool chain
$NT_P_CHAIN_FIRST_ELEM	Name of the first element of the workpiece chain
$NT_T_REF_ELEM	Name of the reference element for tool length
$NT_GEO_AX_NAME	Names of the elements of the geometry axes
$NT_ROT_AX_NAME	Names of the elements of the orientation axes
$NT_CLOSE_CHAIN_P	Reference element for closing the kinematic chain to the **tool**
$NT_CLOSE_CHAIN_T	Reference element for closing the kinematic chain to the **workpiece**.
$NT_ROT_OFF-SET_FROM_FRAME	Importing the rotary axis offset from the work offset upon selection of the transformation:
$NT_TRAFO_INCLUDES_TOOL	Defines whether the tool length is handled in the transformation or - if the transformation permits it - outside of the transformation.
$NT_AUX_POS	Auxiliary position for cycles, e.g. retraction position **Note** Has no meaning for the transformation
$NT_IDENT	Identification number for user-specific program management **Note** Has no meaning for the transformation
$NT_CORR_ELEM_T	Names of a maximum of four correction elements in the kinematic chain for the **tool**

System variable	Meaning
$NT_CORR_ELEM_P	Names of a maximum of four correction elements in the kinematic chain for the **workpiece**
$NT_CNTRL	Control word

The system variables are described in detail in the following sections.

Note

Element names

The system does not monitor as to whether the element names of the kinematic chain, which are referenced to parameterize transformation, were allocated a multiple number of times. If names such as these are available a multiple number of times, then the system data of the transformation always refer to the corresponding element with the lowest index.

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the kinematic chain. To do this, the system variables of the kinematic chain should be set to their default value using function DELOBJ().

Change system variable values

A change to system variable values of the transformation data $NT_... only become effective when the NEWCONF machine data becomes effective. This can be done using the softkey on the user interface in the operating area "Commissioning" > "Machine data " > "Set MD active", in a program using the `NEWCONF` command or using a program end reset, channel reset or a warm or cold restart.

4.2.4.2 $NT_NAME

Function

The NC-wide unique name of the transformation dataset or transformation must be entered in the system variable. The transformation is referenced via this name, e.g. upon activation with TRAFOON. If the zero string ("") is entered as the name, the transformation is considered to be undefined.

Syntax

```
$NT_NAME[<n>] = "<Name>"
```

Meaning

`$NT_NAME`:	Name of the transformation	
	Data type:	STRING
	Default value:	"" (empty string)

<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<name>:	Transformation name, max. string length: 31 characters	
	Data type:	STRING

Example

The name "5-axis transformation C-B" is assigned to the first transformation with kinematic chains:

Program code	Comment
N100 $NT_NAME[1] = "5-axis transformation C-B"	; 1st transformation, ; name = "5-axis transformation C-B"

4.2.4.3 $NT_TRAFO_INDEX

Function

To ensure that a transformation can also be called via the corresponding conventional command, e.g. the orientation transformation via TRAORI, an NC-wide, unique identifier must be entered in the system variable. The identifier identifies a transformation with kinematic chains by means of the transformation type, transformation number and channel number to which a transformation that is called conventionally with this identifier is diverted (TRAORI(<identifier>, ...)). The identifier is decimally coded.

Syntax

```
$NT_TRAFO_INDEX[<n>] = <Id>
```

Meaning

$NT_TRAFO_INDEX:	Identifier, decimally coded (CCBBA)		
	Data type:	INT	
	Default value:	0	
	Decimal place	**Meaning**	
	Ones digit (xxxxA)	Transformation type The type of the transformation that is parameterized in this dataset:	
		Value	**Meaning**
		1	TRAORI
		2	TRANSMIT
		3	TRACYL
		4	TRAANG
		5	TRACON
		Value range:	1, 2, 3, 4, 5

$NT_TRAFO_INDEX: (continued)	Tens and hundreds digits (xxBBx)	Transformation number The number <n>, which references the nth channel-specific machine dataset of this type (MD24100ff or MD25100ff) for a conventional call of a transformation type, e.g. orientation transformation TRAORI(<n>). The number **can** be specified for a conventional call of the first transformation of a type (e.g. TRAORI, TRAORI(), TRAORI(0), TRAORI(1)). All of the calls reference the 1st dataset of the transformation type in the channel. The number **must** be specified after the call of the second transformation of a type (e.g. TRAORI(2)).	
		Value range:	1, 2, 3, ... max. dataset number of the transformation type
$NT_TRAFO_INDEX: (continued)	Thousands and tenthousands digit (CCxxx)	Channel number The channel number defines which channel the current dataset is valid for. If the conventional transformation call of an orientation transformation TRAORI takes place, for example, in the second channel, a transformation with kinematic chains must be parameterized with a channel number equal to 2.	
		Value range:	0, 1, 2, 3, ... max. number of channels, 99 • 0 and 1: valid for channel 1 • 2, 3, ...: valid for channel 2, 3, ... • 99: valid for all channels
<n>:	System variable or transformation index		
	Data type:	INT	
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)	
<Id>:	Identifier		
	Data type:	INT	

Example

The call of the first orientation transformation TRAORI(2) in the third channel should be diverted to the second transformation with kinematic chains.

Program code	Comment
N100 $NT_TRAFO_INDEX[1] = 03021	; 1st transformation with kin. chains, ; 1st orientation transformation for channel 3

4.2.4.4 $NT_TRAFO_TYPE

Function

The designation of the transformation type must be entered in the system variable.

Syntax

```
$NT_TRAFO_TYPE[<n>] = "<TrafoType>"
```

Meaning

$NT_TRAFO_TYPE:	Transformation type		
	Data type:	STRING	
	Default value:	""	
	Value range:	**Value**	**Meaning**
		"TRAORI_DYN"	Dynamic orientation transformation with orientation axes
		"TRAORI_STAT"	Static orientation transformation without orientation axes
		"TRAANG"	Inclined angle transformation without orientation axes
		"TRAANG_K"	Oblique angle transformation (The extension "_K" distinguishes it from a conventionally defined TRAANG transformation)
		"TRANSMIT_K"	Face end transformation TRANSMIT
		"TRACON_K"	Concatenating transformations
<n>:	System variable or transformation index		
	Data type:	INT	
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)	
<TrafoType>:	Transformation type		
	Data type:	STRING	

Example

The first transformation with kinematic chains describes a dynamic orientation transformation with orientation axes:

Program code	**Comment**
N100 $NT_TRAFO_TYPE[1] = "TRAORI_DYN"	; 1st transformation, ; transformation type = "TRAORI_DYN"

4.2.4.5 $NT_T_CHAIN_FIRST_ELEM

Function

The kinematics of a transformation is described by a maximum of two kinematic subchains, which begin in the respective root element, i.e. the first element of the kinematic chain that is in effect. One subchain leads from the root element to the **workpiece** reference point (see Definition of kinematic transformations (Page 256)). The other subchain leads to the **tool** reference point. If the subchain does not start, as is usual, in the root element, the first element of the active kinematic subchain can be defined for the tool chain and for the workpiece chain using the system variables

The name ($NT_NAME (Page 279)) of the element of the kinematic subchain that is currently active, which defines the starting point of the subchain for the tool reference point, must be entered in the system variable $NT_T_CHAIN_FIRST_ELEM.

Note

Pure tool kinematics

The first element only has to be defined in $NT_P_CHAIN_FIRST_ELEM in special circumstances; otherwise the system variable can remain empty. In this case, the ROOT element is used as the first element.

Syntax

```
$NT_T_CHAIN_FIRST_ELEM[<n>] = "<ElementName>"
```

Meaning

$NT_T_CHAIN_FIRST_ELEM:	Name of the element that defines the starting point of the currently active kinematic chain for the tool reference point.	
	Data type:	STRING
	Default value:	""
	Range of values:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<ElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic subchain that defines the starting point of the subchain for the workpiece reference point is "Baseoffset_2":

Program code	Comment
N100 $NT_T_CHAIN_FIRST_ELEM[1] = "Baseoffset_2"	; 1st transformation, ; first element for the tool reference point

4.2.4.6 $NT_P_CHAIN_FIRST_ELEM

Function

The kinematics of a transformation is described by a maximum of two kinematic subchains, which begin in the respective root element, i.e. the first element of the kinematic chain that is in effect. One subchain leads from the root element to the **tool** reference point (see Definition of kinematic transformations (Page 256)). The other subchain, which is described here, leads to the **workpiece** reference point. If the subchain does not start, as is usual, in the root element, the first element of the active kinematic subchain can be defined for the tool and workpiece chain using the system variables.

The name ($NT_NAME (Page 279)) of the element of the kinematic chain that is currently in effect, which defines the starting point of the subchain for the workpiece reference point, must be entered in the system variable $NT_P_CHAIN_FIRST_ELEM.

Note

Pure workpiece kinematics

The first element only has to be defined in $NT_T_CHAIN_FIRST_ELEM in special circumstances; otherwise the system variable can remain empty. In this case, the ROOT element is used as the first element.

Syntax

```
$NT_P_CHAIN_FIRST_ELEM[<n>] = "<ElementName>"
```

Meaning

`$NT_P_CHAIN_FIRST_ELEM`:	Name of the first element of the kinematic subchain that is currently active defined at the endpoint of the tool reference point.	
	Data type:	STRING
	Default value:	""
	Range of values:	Element names of the currently active kinematic chain
`<n>`:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
`<ElementName>`:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain that defines the end point of the subchain for the workpiece reference point is "TableOffset":

Program code	Comment
N100 $NT_P_CHAIN_FIRST_ELEM[1] = "Base-offset"	; 2nd transformation, ; element for the workpiece reference point

4.2.4.7 $NT_T_CHAIN_LAST_ELEM

Function

The kinematics of a transformation is described by a maximum of two kinematic subchains, which begin in the respective root element, i.e. the first element of the kinematic chain that is in effect. One subchain leads from the root element to the **workpiece** reference point (see Definition of kinematic transformations (Page 256)). The other subchain, which is described here, leads to the **tool** reference point.

The name ($NT_NAME (Page 279)) of the element of the kinematic chain that is currently in effect, which defines the end point of the subchains for the tool reference point, must be entered in the system variable $NT_T_CHAIN_LAST_ELEM.

Note

Pure tool kinematics

If, for pure tool kinematics, the kinematics of the transformation is completely described by the subchain for the tool reference point, the parameterization of the system variable $NT_T_CHAIN_LAST_ELEM is sufficient. The system variable for describing the subchain for the workpiece reference point ($NT_P_CHAIN_LAST_ELEM) can remain blank.

Syntax

```
$NT_T_CHAIN_LAST_ELEM[<n>] = "<ElementName>"
```

Meaning

$NT_T_CHAIN_LAST_ELEM:	Name of the element of the kinematic chain currently in effect, which defines the end point for the tool reference point, starting from the root element.	
	Data type:	STRING
	Default value:	""
	Range of values:	Element names of the currently active kinematic chain

<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<ElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain that defines the end point of the subchain for the tool reference point is "Basetool":

Program code	Comment
N100 $NT_T_CHAIN_LAST_ELEM[1] = "base tool"	; 1st transformation, ; element for the tool reference point

4.2.4.8 $NT_P_CHAIN_LAST_ELEM

Function

The kinematics of a transformation is described by a maximum of two kinematic subchains, which begin in the respective root element, i.e. the first element of the kinematic chain that is in effect. One subchain leads from the root element to the **tool** reference point (see Definition of kinematic transformations (Page 256)). The other subchain, which is described here, leads to the **workpiece** reference point.

The name ($NT_NAME (Page 279)) of the element of the kinematic chain that is currently in effect, which defines the end point of the subchains for the workpiece reference point, must be entered in the system variable $NT_P_CHAIN_LAST_ELEM.

Note

Pure workpiece kinematics

If, for pure workpiece kinematics, the kinematics of the transformation is completely described by the subchain for the workpiece reference point, the parameterization of the system variable $NT_P_CHAIN_LAST_ELEM is sufficient. The system variable for describing the subchain for the tool reference point ($NT_T_CHAIN_LAST_ELEM) can remain blank.

Syntax

```
$NT_P_CHAIN_LAST_ELEM[<n>] = "<ElementName>"
```

Meaning

$NT_P_CHAIN_LAST_ELEM:	Name of the element of the kinematic chain currently in effect, which defines the end point for the workpiece reference point, starting from the root element.	
	Data type:	STRING
	Default value:	""
	Range of values:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<ElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain that defines the end point of the subchain for the workpiece reference point is "TableOffset":

Program code	Comment
N100 $NT_P_CHAIN_LAST_ELEM[1] = "TableOffset"	; 2nd transformation, ; element for the workpiece reference point

4.2.4.9 $NT_T_REF_ELEM

Function

The name ($NT_NAME (Page 279)) of the element of the kinematic chain that is currently in effect, which defines the tool reference position, i.e. the reference point for the tool length calculation, must be entered in the system variable. The tool reference position is located at the **starting point** of the kinematic element.

If no name of an element is entered in the system variable, the tool reference position is identical to the tool reference point.

Syntax

```
$NT_T_REF_ELEM[<n>] = "<RefElementName>"
```

Meaning

$NT_T_REF_ELEM:	Name of the element of the kinematic chain that is currently in effect, which defines the tool reference position.	
	Data type:	STRING
	Default value:	"" (tool reference position = tool reference point)
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<RefElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain that is currently in effect, which defines the tool reference position, is "ToolRefPoint":

Program code	Comment
N100 $NT_T_REF_ELEM[1] = "ToolRef-Point"	; 1st transformation, ; element for the tool reference position

4.2.4.10 $NT_GEO_AX_NAME

Function

A maximum of three names ($NT_NAME (Page 279)) of the elements of the kinematic chain that is currently in effect, which define the geometry axes (linear axes) of the transformation type (Page 256) to be parameterized, must be entered in the system variable. The geometry axes carry out the linear compensating movements, which result from the kinematic transformation.

Syntax

```
$NT_GEO_AX_NAME[<n>,<k>] = "<GeoAxElementName>"
```

Meaning

$NT_GEO_AX_NAME:	Names of the elements of the kinematic chain currently in effect, which define the geometry axes	
	Data type:	STRING
	Default value:	("", "", "")
	Value range:	Element names of the currently active kinematic chain

<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index for the elements that define the geometry axes	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
<GeoAxElementName>:	Name of an element of the kinematic chain that is currently in effect, which defines a geometry axis.	
	Data type:	STRING

Example

For the first transformation, the names of the elements of the kinematic chain currently in effect, which define the geometry axes, are "X_AXIS", Y_AXIS" and "Z_AXIS", must be entered.

```
Program code                                    Comment
                                                ; 1st transformation,
                                                ; elements of the geometry axes:
N100 $NT_GEO_AX_NAME[1,0] = "X_AXIS"            ; X coordinate (abscissa)
N110 $NT_GEO_AX_NAME[1,1] = "Y_AXIS"            ; Y coordinate (ordinate)
N120 $NT_GEO_AX_NAME[1,2] = "Z_AXIS"            ; Z coordinate (applicate)
```

4.2.4.11 $NT_ROT_AX_NAME

Function

A maximum of three names ($NT_NAME (Page 279)) of the elements of the kinematic chain that is currently in effect, which define the rotary axes of the transformation type (Page 256) to be parameterized, must be entered in the system variable.

- Orientation transformation: The orientation axes A, B, C
- Face end (TRANSMIT) and cylinder surface transformation (TRACYL): The rotary axis, which rotates the workpiece and, together with a linear axis, defines the polar coordinate system.

Order of definitions

The maximum three rotary axes, beginning at Index 0, must be entered in the system variables in the order in which they are defined in the kinematic chain that is currently in effect.

The kinematic chain must be run through as follows:

1. Starting point: **Workpiece** reference point
2. Elements of the chain
3. End point: **Tool** reference point

The elements that define the rotary axes must be seamlessly entered in the system variable with their names, starting at Index 0.

Syntax

```
$NT_ROT_AX_NAME[<n>,<k>] = "<RotAxElementName>"
```

Meaning

$NT_ROT_AX_NAME:	Names of the elements of the kinematic chain currently in effect, which define the rotary axes	
	Data type:	STRING
	Default value:	("", "", "")
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index for the elements that define the rotary axes	
	Data type:	INT
	Range of values:	0: 1st rotary axis in the definition sequence 1: 2nd rotary axis in the definition sequence 2: 3rd rotary axis in the definition sequence
<RotAxElementName>:	Name of an element of the kinematic chain that is currently in effect, which defines a rotary axis.	
	Data type:	STRING

Example

For the first transformation, the names of the elements of the kinematic chain currently in effect, which define the rotary axes, are "ROT_AXIS_1", ROT_AXIS_2" and "ROT_AXIS_3" must be entered.

```
Program code                                    Comment
                                                ; 1st transformation,
                                                ; elements of the rotary axes:
N100 $NT_ROT_AX_NAME[1,0] = "ROT_AXIS_1"        ; 1st rotary axis
N110 $NT_ROT_AX_NAME[1,1] = "ROT_AXIS_2"        ; 2nd rotary axis
N120 $NT_ROT_AX_NAME[1,2] = "ROT_AXIS_3"        ; 3rd rotary axis
```

4.2.4.12 $NT_ROT_AX_OFFSET

Function

System variable `$NT_ROT_AX_OFFSET` allows you to enter an angle offset for the rotary axes of the active transformation.

- For TRAORI_K, these are the rotary axes 1, 2, and 3.
- For TRANSMIT_K and TRACYL_K it is rotary axis 1.

Supplementary conditions

- System variable `$NT_ROT_AX_OFFSET` for Transmit or Tracyl transformations without a kinematic chain corresponds to machine data $MC_TRANSMIT_ROT_AX_OFFSET_n or $MC_TRACYL_ROT_AX_OFFSET_n.
- System variable `$NT_ROT_AX_OFFSET` for orientation transformations without a kinematic chain corresponds to machine data $MC_TRAFO5_ROT_AX_OFFSET_n.

Syntax

```
$NT_ROT_AX_OFFSET[<n,m>] = <offset_angle>
```

Meaning

`$NT_ROT_AX_OFFSET`	Angle offset for rotary axes	
	Data type:	REAL
	Default value:	-1; content is not evaluated
`<n>`:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
`<m>`:	Index of the variable $NT_ROT_AX_CNT; specifies for which rotary axis the offset is set.	
	Data type:	INT
	Range of values	• 0 ... 2; for orientation transformation • 1; for TRANSMIT_K and TRACYL_K

Example

Provides the number of rotary axes for the subchain:

Program code	**Comment**
`N100 $NT_ROT_AX_CNT[1,0] = 180`	`; assigns offset "180 °" to rotary axis "0" of transformation "1".`

4.2.4.13 $NT_CLOSE_CHAIN_P

Function

The name ($NT_NAME (Page 279)) of the element of the currently active kinematic chain, at the end of which the subchain for the **workpiece** reference point is closed if $NT_CNTRL, Bit 7 == 1, can be entered in the system variable.

If no name is entered in the system variable, the subchain is closed independently of $NT_CNTRL, Bit 7, at the end of the last element (**workpiece** reference point).

Syntax

```
$NT_CLOSE_CHAIN_P[<n>] = "<ElementName>"
```

Meaning

$NT_CLOSE_CHAIN_P:	Name of the element of the kinematic chain currently in effect, at the end of which the subchain for the **workpiece** reference point is closed	
	Data type:	STRING
	Default value:	""
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<ElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain, at the end of which the part chain for the **workpiece** reference point is closed, is: "P_Offset_3"

Program code	Comment
N100 $NT_CLOSE_CHAIN_P[1] = "P_Offset_3"	; 1st transformation, ; element for closing the subchain ; to the **workpiece** reference point

See also

$NT_CNTRL (Page 298)

4.2.4.14 $NT_CLOSE_CHAIN_T

Function

The name ($NT_NAME (Page 279)) of the element of the currently active kinematic chain, at the end of which the subchain for the **tool** reference point is closed if $NT_CNTRL, Bit 8 == 1 ($NT_CNTRL (Page 298)), can be entered in the system variable.

If no name is entered in the system variable, the subchain is closed independently of $NT_CNTRL, Bit 8, at the end of the last element (**tool** reference point).

Syntax

```
$NT_CLOSE_CHAIN_T[<n>] = "<ElementName>"
```

Meaning

$NT_CLOSE_CHAIN_T:	Name of the element of the kinematic chain currently in effect, at the end of which the subchain for the **tool** reference point is closed	
	Data type:	STRING
	Default value:	""
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<ElementName>:	Name of an element of the currently active kinematic chain	
	Data type:	STRING

Example

For the first transformation, the name of the element of the kinematic chain, at the end of which the part chain for the **tool** reference point is closed, is: "T_Offset_3"

Program code	**Comment**
N100 $NT_CLOSE_CHAIN_T[1] = "T_Offset_3"	; 1st transformation, ; element for closing the subchain ; to the **tool** reference point

4.2.4.15 $NT_ROT_OFFSET_FROM_FRAME

Function

In the system variable $NT_ROT_OFFSET_FROM_FRAME it must be entered whether the programmable offset for orientation axes will be automatically imported from the zero point offset for the orientation axes, which becomes active upon switch-on of an orientation transformation.

- TRAORI or TRAORY_DYN:
  - If `$NT_ROT_OFFSET_FROM_FRAME=1` the following applies:
    The offsets for the orientation axes of the 5/6 axis transformation of the rotary axes, for which the tool orientation is in the basic setting, are automatically defined. Thus, the offset from an active work offset of the rotary axes, which is active as soon as the transformation is switched on, is applied.
  - If `$NT_ROT_OFFSET_FROM_FRAME=0`, then the following applies:
    The offset of the orientation axes is only applied if the work offsets when switching on the transformation are kept, i.e. MD10602 $MN_FRAME_GEOAX_CHANGE_MODE is > 0.
- TRACYL and/or TRANSMIT
  - If `$NT_ROT_OFFSET_FROM_FRAME=2`, the following applies:
    Axial offset of the rotary axis is taken into account up to the SZS. SZS is the settable zero system. The SZS frames contain transformed rotations around the rotary axis.

Note

Automatic application of work offsets

Automatic application of the active work offsets to the offset is mainly only practical for the polar axis in table kinematics. i.e. for example, in AC kinematics for the C axis. If the offset is changed for another orientation axis, in particular, the non-polar axis, this changes the kinematic behavior of the transformation. The can result in the compensatory movement of the linear axes no longer being correct.

Syntax

```
$NT_ROT_OFFSET_FROM_FRAME[<n>] = "<ROTOffset>"
```

Meaning

`$NT_ROT_OFFSET_FROM_FRAME`:	Importing the rotary axis offset from the work offset upon selection of the transformation: • 0 = Axial offset of the rotary axis is not taken into account. • 1 = axial offset of the rotary axis is taken into account. • 2 = Axial offset of the rotary axis is taken into account up to the SZS or is automatically applied (see above).	
	Data type:	INT
	Default value:	0

<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	0, 1, 2
ROTOffset	Rotary axis offset	

Example

The 2nd transformation is assigned the name "B axis":

Program code	Comment
N100 $NT_ROT_OFFSET_FROM_FRAME[1] = 1	; axial offset of the rotary axis is taken into account.

4.2.4.16 $NT_TRAFO_INCLUDES_TOOL

Function

Information about whether the tool length is to be handled within the transformation or outside of the transformation must be entered in the system variable.

Syntax

```
$NT_TRAFO_INCLUDES_TOOL[<n>] = "<>"
```

Meaning

$NT_TRAFO_INCLUDES_TOOL:	Defines whether the tool length is handled within or outside of the transformation.	
	Data type:	BOOL
		• 0: Tool length is handled outside of the transformation. • 1: Tool length is handled within the transformation.
	Default value:	0
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)

Example

The 2nd transformation is assigned the name "B axis":

Program code	Comment
N100 $NT_TRAFO_INCLUDE_TOOL[1] = "1"	; 1st transformation, ; tool length is processed within the transformation.

4.2.4.17 $NT_AUX_POS

Function

A position vector, e.g. for use in user-specific cycles, can be entered in the system variable. The system variable is not evaluated in the NC. The values are, however, automatically recalculated for an inch/metric switchover.

Syntax

```
$NT_AUX_POS[<n>,<k>] = <Value>
```

Meaning

$NT_AUX_POS:	User-specific position vector	
	Data type:	REAL
	Default value:	(0.0, 0.0, 0.0)
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index of the vector coordinates	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
<Value>:	Coordinate value	
	Data type:	REAL

Example

A position vector (1.0, 1.0, 1.0) is entered for the first transformation:

```
Program code                                Comment
                                            ; 1st transformation,
                                            ; position vector:
N100 $NT_AUX_POS[1,0] = 1.0                 ; X coordinate (abscissa)
N110 $NT_AUX_POS[1,1] = 1.0                 ; Y coordinate (ordinate)
N120 $NT_AUX_POS[1,2] = 1.0                 ; Z coordinate (applicate)
```

4.2.4.18 $NT_IDENT

Function

Up to three user-specific numeric values, analogous to the system variable $TC_CARR37, can be entered in the system variable as an identifier for the tool carrier or the current transformation for program management. The system variable is not evaluated in the NC.

Syntax

```
$NT_IDENT[<n>,<k>] = <Value>
```

Meaning

$NT_IDENT:	User-specific administration data	
	Data type:	INT
	Default value:	(0, 0, 0)
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index of the administration data	
	Data type:	INT
	Range of values:	0, 1, 2
<Value>:	User-specific value	
	Data type:	INT

Example

The following administration data is entered for the first transformation: (1000, 100, 0)

Program code	Comment
	; 1st transformation,
	; position vector:
N100 $NT_AUX_POS[1,0] = 1000	; identifier: Toolholder
N110 $NT_AUX_POS[1,1] = 100	; identifier: Transformation
N120 $NT_AUX_POS[1,2] = 0	; -

4.2.4.19 $NT_CNTRL

Function

With this system variable, the behavior of the transformation in specific situations can be influenced using bit code.

Bit	Value	Meaning
0	---	Not assigned
1 - 3	Orientation axes as **speed-controlled spindles** Bit 1: 1st orientation axis Bit 2: 2nd orientation axis Bit 3: 3rd orientation axis	
	0	The orientation axis is operated as a speed-controlled axis.
	1	The orientation axis is operated as a speed-controlled spindle. **Note** The following scenarios are currently supported: • Only the 1st orientation axis (Bit 1 = 1) is operated as a speed-controlled spindle • Only the 3rd orientation axis (Bit 3 = 1) is operated as a speed-controlled spindle Some examples are the applications "Turning on milling machines" or "5-axis milling", in which the third orientation axis is not operated with position control.

Bit	Value	Meaning
4 - 6	Orientation axes with **Hirth joint** Bit 4: 1st orientation axis Bit 5: 2nd orientation axis Bit 6: 3rd orientation axis	
	0	The orientation axis is not Hirth-coupled.
	1	The orientation axis is Hirth-coupled. **Note** • Parameters of the Hirth joint The following machine data is evaluated for the parameters of the Hirth joint: – MD30501 $MA_INDEX_AX_NUMERATOR – MD30502 $MA_INDEX_AX_DENOMINATOR – MD30503 $MA_INDEX_AX_OFFSET MD30505 $MA_HIRTH_IS_ACTIVE is not evaluated. I.e. the axis does not have to be parameterized as a genuine Hirth axis. • Modulo rotary axes For modulo rotary axes (MD30310 $MA_ROT_IS_MODULO==TRUE), MD30330 $MA_MODULO_RANGE is evaluated instead of MD30501 $MA_INDEX_AX_NUMERATOR. The distances of the permitted axis positions are then calculated for: Distance = $MA_MODULO_RANGE / $MA_INDEX_AX_DENOMINATOR MD30503 $MA_INDEX_AX_OFFSET is also evaluated for modulo rotary axes
7	An additional constant element of the type "OFFSET" is automatically NC-internally inserted into the subchain for the **workpiece** reference point in such a way that the chain is closed.	
	0	The element connects the **workpiece** reference point to the machine zero point.
	1	The element connects the end point of the element that is referenced by $NT_CLOSE_CHAIN_P (Page 292) to the machine zero point.
8	An additional constant element of the type "OFFSET" is automatically NC-internally inserted into the subchain for the **tool** reference point in such a way that the chain is closed.	
	0	The element connects the **tool** reference point to the machine zero point.
	1	The element connects the end point of the element that is referenced by $NT_CLOSE_CHAIN_T (Page 293) to the machine zero point.
9	The scope of functions of TRANSMIT or TRACYL transformations is specified more precisely. If a center offset axis has been specified via $NT_ROT_AX_NAME, the behavior can be set.	
	0	The center offset axis can be interpolated in any way and has no influence on the transformation.
	1	The center offset axis compensates the tool offset in the Y direction. Processing up to the center of rotation is possible.
10	0	Parameterizes TRACYL without slot side offset.
	1	Parameterizes TRACYL with slot side offset. In connection with bit 9 = 1, you can set whether TRACYL is operated with or without slot side offset upon activation of the transformation via TRAFOON. The default setting is slot side offset active If you are working with slot side offset, $NT_GEO_AX_NAME must contain a reference to a linear axis (slot side offset axis).

Bit	Value	Meaning		
11	0	The direction of rotation of the pole axis remains unchanged.		
	1	The direction of rotation of the pole axis is inverted.		
12 ... 15		Reserved for OEM transformations.		
16 ... 18		Binary coding using bits 16, 17 and 18 (bits H10000 - H40000) defines how the channel axes, which are entered in $NT_GEO_AX_NAME [n,1], $NT_GEO_AX_NAME [n,2] and $NT_GEO_AX_NAME [n,3], are assigned to the geometry axes. The geometry axis identifiers are assigned in the order (X, Y, Z).		
	Decimal	Order of the geometry axes		
	0	X	Y	Z
	1	Z	X	Y
	2	Y	Z	X
	3	Z	Y	X
	4	X	Z	Y
	5	Y	X	Z
19	0			
	1	The last kinematic chain element, which defines the tool reference point, must be a rotary axis. The rotation vector of the rotary axis then defines the Z direction of the tool coordinate system. The rotation about the tool Z axis that is defined in this way results from the corresponding definition for the local coordinate system of an axis in kinematic chains. If the system variable $NK_A_OFF of this chain element contains a value not equal to zero, the tool coordinate system is also rotated about the coordinate axis with this angle.		
20	0	The sign of the axis for determining the tool coordinate system in the Z direction is retained.		
	1	The sign of the axis for determining the tool coordinate system in the Z direction is inverted if also bit 19 is set.		

Syntax

```
$NT_CNTRL[<n>] = "<Value>"
```

Meaning

$NT_CNTRL:	Bit-coded control variable of the transformation ('Bdcbbbaaa0')	
	Data type:	INT
	Default value:	0
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<Value>:	Bit-coded control value	
	Data type:	INT

Example

The third orientation axis of the first transformation is Hirth-coupled.

```
Program code                                    Comment
N100 $NT_CNTRL[1] = 'B001000000'                ; 1st transformation,
```

4.2.4.20 $NT_ROT_AX_CNT

Function

The system variable $NT_ROT_AX_CNT provides the number of the relevant rotary axes in the workpiece or in the tool chain. The relevant rotary axes in this sense are the rotary axes that are defined in the system variable $NT_ROT_AX_Name.

Syntax

```
$NT_ROT_AX_CNT[<n,m>] = <NumberRotAxes>
```

Meaning

$NT_ROT_AX_CNT:	Number of relevant rotary axes of a transformation in the workpiece chain or tool chain	
	Data type:	INT
	Default value:	-1; content is not evaluated
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<m>:	Index of the variable $NT_ROT_AX_CNT; specifies which kinematic chain will be selected.	
	Data type:	INT
	Value range	• 0; selects the workpiece chain • 1; selects the tool chain

Example

Provides the number of rotary axes for the subchain:

```
Program code                                    Comment
N100 $NT_ROT_AX_CNT[1,0] = 2                    ; Transformation 1 has two rotary axes.
```

4.2.4.21 $NT_BASE_TOOL_COMP

Function

Using bit-coded system variable `$NT_BASE_TOOL_COMP`, for each of the geometry axes you can separately set whether for the base tool an offset is entered in transformation frame $P_TRAFRAME. This means that when the transformation is selected, no change is made in the WCS component.

This function is only available if system frame $P_TRAFRAME was configured using MD28082 $MC_MM_SYSTEM_FRAME_MASK bit6.

The system variable is only relevant for TRANSMIT and TRACYL.

Syntax

```
$NT_BASE_TOOL_COMP[<n>, <m>]
```

Meaning

`$NT_BASE_TOOL_COMP`:	Bit-coded system variable	
	Data type:	INT
	Default value:	
`<n>`:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
`<m>`:	Bits for coding the offset: • Bit 0 acts in the direction of the X axis (geometry axis 1) • Bit 1 acts in the direction of the Y axis (geometry axis 2) • Bit 2 acts in the direction of the Z axis (geometry axis 3)	
	Data type:	INT

Example

The 2nd transformation is assigned the name "B axis":

Program code	**Comment**
`N100 $NT_BASE_TOOL_COMP[n,0];`	`; Transformation "n" is compensated for in the X-axis direction.`

4.2.5 Additive system variable for orientation transformation

4.2.5.1 Overview

System variable	Meaning
$NT_BASE_ORIENT	Basic tool orientation
$NT_BASE_ORIENT_NORMAL	Normal vector of the orientation
$NT_ROT_AX_POS	Axis positions of the orientation axes, which are parameterized as constant rotations
$NT_POLE_LIMIT	End angle tolerance with interpolation through pole
$NT_POLE_TOL	End angle tolerance for pole interpolation
$NT_IGNORE_TOOL_ORIENT	Always take tool orientation from $NT_BASE_ORIENT / $NT_BASE_ORIENT_NORMAL.
$NT_CORR_ELEM_T	Names of the correction elements in the tool chain that can be written to by the CORRTRAFO(...) function.
$NT_CORR_ELEM_P	Names of the correction elements in the subchain that can be written to by the CORRTRAFO(...) function.

The system variables are described in detail in the following sections.

Note

Element names

The system does not monitor as to whether the element names of the kinematic chain, which are referenced to parameterize transformation, were allocated a multiple number of times. If names such as these are available a multiple number of times, then the system data of the transformation always refer to the corresponding element with the lowest index.

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the kinematic chain. To do this, the system variables of the kinematic chain should be set to their default value using function DELOBJ().

Change system variable values

Changes to system variable values of the transformation data $NT_... only become effective when the NEWCONF machine data becomes effective. This can be done using the softkey on the user interface in the operating area "Commissioning" > "Machine data " > "Set MD active", in a program using the `NEWCONF` command or using a program end reset, channel reset or a warm or cold restart.

4.2.5.2 $NT_BASE_ORIENT

Function

The rotation vector of the basic tool orientation must be entered in the system variable. The basic tool orientation is in effect if no tool is selected.

Syntax

```
$NT_BASE_ORIENT[<n>,<k>] = <VectorComp>
```

Meaning

$NT_BASE_ORIENT:	Rotation vector of the basic tool orientation (X; Y; Z)	
	Data type:	REAL
	Default value:	(0.0, 0.0, 1.0)
	Value range:	Direction vector: $1*10^{-6}$ < |Vector | ≤ max. REAL value
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index of the coordinates	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
<VectorComp>:	Coordinate value of the vector component	
	Data type:	STRING

Example

The rotation vector of the basic tool orientation of the first transformation has the value (1.0, 2.0, 3.0):

```
Program code                                Comment
                                            ; 1st transformation,
                                            ; vector components:
N100 $NT_BASE_ORIENT[1,0] = 1.0             ; X coordinate (abscissa)
N110 $NT_BASE_ORIENT[1,1] = 2.0             ; Y coordinate (ordinate)
N120 $NT_BASE_ORIENT[1,2] = 3.0             ; Z coordinate (applicate)
```

4.2.5.3 $NT_BASE_ORIENT_NORMAL

Function

The normal vector of the basic tool orientation must be entered in the system variable. The system variable or the normal vector of the basic tool orientation is only relevant for transformations with three degrees of orientation freedom if no tool is selected.

Syntax

```
$NT_BASE_ORIENT_NORMAL[<n>,<k>] = <VectorComp>
```

$NT_BASE_ORIENT_NORMAL:	Normal vector of the basic tool orientation (X; Y; Z)	
	Data type:	REAL
	Default value:	(0.0, 1.0, 0.0)
	Value range:	Normal vector: $1*10^{-6}$ < |Vector | ≤ max. REAL value
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index of the coordinates	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
<VectorComp>:	Coordinate value of the vector component	
	Data type:	STRING

Example

The normal vector of the basic tool orientation of the first transformation has the value (2.0, -1.0, 3.0):

```
Program code                                    Comment
                                                ; 1st transformation,
                                                ; vector components:
N100 $NT_BASE_ORIENT_NORMAL[1,0] =  2.0         ; X coordinate (abscissa)
N110 $NT_BASE_ORIENT_NORMAL[1,1] = -1.0         ; Y coordinate (ordinate)
N120 $NT_BASE_ORIENT_NORMAL[1,2] =  3.0         ; Z coordinate (applicate)
```

4.2.5.4 $NT_ROT_AX_POS

Function

The positions of the orientation axes, which result from the constant rotations which are parameterized in elements of the kinematic chain of the type ROT_CONST, must be entered in the system variable. The positions of the orientation axes that are effective due to this element can thus be described without having to know the kinematic structure of the transformation or the indices of the associated kinematic elements.

Syntax

```
$NT_ROT_AX_POS[<n>,<k>] = <OriAxPos>
```

Meaning

$NT_ROT_AX_POS:	Positions of the orientation axes due to the constant rotation	
	Data type:	REAL
	Default value:	(0.0, 0.0, 0.0)
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Index of the orientation axes	
	Data type:	INT
	Range of values:	0: 1st orientation axis 1: 2nd orientation axis 2: 3rd orientation axis
<OriAxPos>:	Position value	
	Data type:	REAL

Example

The positions of the orientation axes, which result from the parameterized constant rotation, are: (0, 0, 45.0°)

Program code	**Comment**
	; 1st transformation,
	; positions of the orientation axes:
N100 $NT_ROT_AX_POS[1,0] = 0	; 1st orientation axis
N110 $NT_ROT_AX_POS[1,1] = 0	; 2nd orientation axis
N120 $NT_ROT_AX_POS[1,2] = 45.0	; 3rd orientation axis

4.2.5.5 $NT_POLE_LIMIT

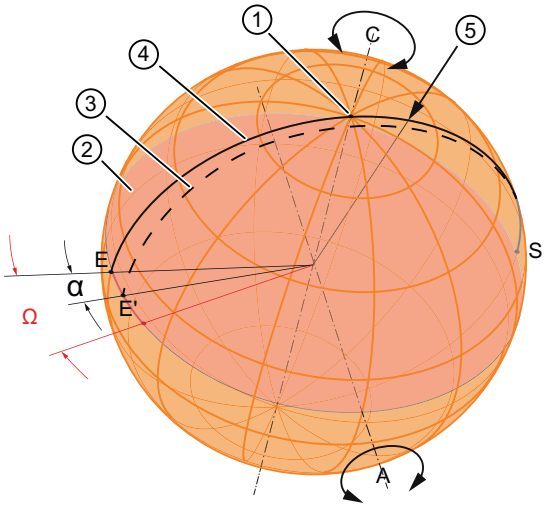
Function

The maximum permitted angle by which the C axis can deviate from its programmed position during a large circle interpolation at the end of a block is entered in the system variable.

End angle

If a traversing movement is programmed that is not to pass exactly through the pole, but within the area near to the pole defined by MD 24530 ($MC_TRAFO5_NON_POLE_LIMIT_n), the movement will deviate from the specified path because the interpolation runs exactly through the pole point. As a result, the position at the end point of the fourth axis (pole axis) deviates from the programmed value.
A pole angle, which defines a tolerance circle about the pole, can be parameterized in the

system variables. If a programmed tool path passes by the pole within the tolerance circle, a switchover is made from orientation interpolation to linear/rotary axis interpolation.



①	Pole
②	Equatorial plane
③	Programmed path
④	Path traveled through the pole
⑤	End angle
A	Rotary axis about the X coordinate axis of the WCS
C	Rotary axis about the Z coordinate axis of the WCS
S	Starting point of the interpolation / traversing block
E	Approached end point
E'	Programmed end point
α	End angle
Ω	Maximum permitted end angle ($NT_POLE_LIMIT)

Syntax

```
$NT_POLE_LIMIT[<n>] = <PoleTolAngle>
```

Meaning

$NT_POLE_LIMIT:	Pole angle	
	Unit:	Degrees
	Data type:	REAL
	Default value:	2.0
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value

<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<PoleTolAngle>:	Tolerance value	
	Data type:	REAL

Example

A pole angle of 2.54° is set for the first transformation:

Program code	Comment
N100 $NT_POLE_LIMIT[1] = 2.54	; 1st transformation, ; pole angle

4.2.5.6 $NT_POLE_LIMIT

Function

The maximum permitted angle by which the C axis can deviate from its programmed position during a large circle interpolation at the end of a block is entered in the system variable.

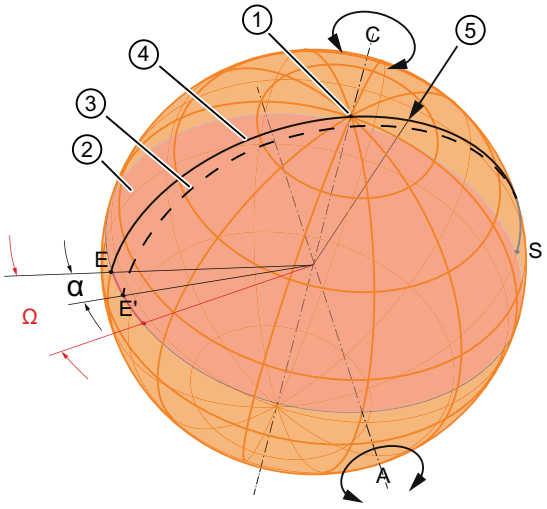
End angle

In the 5-axis transformation, the two orientation axes of the tool span a spherical coordinate system, consisting of meridians and parallels.

If a traversing movement is programmed that is not to pass exactly through the pole, but within the area near to the pole defined by MD 24530 ($MC_TRAFO5_NON_POLE_LIMIT_n), the movement will deviate from the specified path because the interpolation runs exactly through the pole point. As a result, the position at the end point of the fourth axis (pole axis) deviates from the programmed value.
The system variable specifies the angle by which the pole axis for the 5-axis transformation can deviate from the programmed value if there is a switchover from the programmed interpolation to the interpolation through the pole point.
If a large deviation results, an error message is displayed (Alarm 14112) and the interpolation is not carried out.

For a conventional 5 or 6-axis interpolation, a pole position is identified by the fact that the tool orientation does not change during a rotation of a rotary axis. If a programmed path then runs in the vicinity of the pole, extremely high traversing speeds can result for individual rotary axes to maintain the orientation. If the programmed path runs exactly through the pole, this problem may not arise under certain circumstances.

A pole angle, which defines a tolerance circle about the pole, can be parameterized in the system variables. If a programmed tool path passes by the pole within the tolerance circle, a switchover is made from orientation interpolation to linear/rotary axis interpolation.



①	Pole
②	Equatorial plane
③	Programmed path
④	Path traveled through the pole
⑤	End angle
A	Rotary axis about the X coordinate axis of the WCS
C	Rotary axis about the Z coordinate axis of the WCS
S	Starting point of the interpolation / traversing block
E	Approached end point
E'	Programmed end point
α	End angle
Ω	Maximum permitted end angle ($NT_POLE_LIMIT)

Syntax

```
$NT_POLE_LIMIT[<n>] = <PoleTolAngle>
```

Meaning

$NT_POLE_LIMIT:	Pole angle	
	Unit:	Degrees
	Data type:	REAL
	Default value:	2.0
	Value range:	- max. REAL value ≤ x ≤ + max. REAL value

<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<PoleTolAngle>:	Tolerance value	
	Data type:	REAL

Example

A pole angle of 2.54° is set for the first transformation:

Program code	Comment
N100 $NT_POLE_LIMIT[1] = 2.54	; 1st transformation, ; pole angle

4.2.5.7 $NT_POLE_TOL

Function

The end angle tolerance for interpolation through the pole for the first 5/6-axis transformation must be entered in the system variable.

The system variable is only evaluated by the generic 5/6-axis transformation.

If the programmed end orientation lies within the polar cone and within the tolerance cone specified by this MD, the pole axis does not move and retains its start positions. In contrast, the other rotary axis accepts the programmed angle.

As a result, there is a deviation of the end orientation from the programmed orientation.

Another meaning of this system variable is the handling of the programmed end orientation for non-perpendicular kinematics. As a rule, not all of the tool orientations can be set for these machine kinematics. If an orientation that lies outside of the settable range is programmed, alarm 14112 "Programmed orientation path not possible" is displayed.

Syntax

```
$NT_POLE_TOL[<n>] = <PoleTolLim>
```

Meaning

$NT_POLE_TOL:	End angle tolerance for pole interpolation	
	Data type:	REAL
	Default value:	0.0
	Value range:	0.0 ≤ x ≤ 45.0
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)

<PoleTolLim>:	Angle	
	Data type:	REAL

Example

An end angle tolerance of 2.54° is defined for the first transformation:

Program code	Comment
N100 $NT_POLE_TOL[1] = 2.54	; 1st transformation, ; end angle tolerance

4.2.5.8 $NT_IGNORE_TOOL_ORIENT

Function

Each tool that is known in the controller has a defined orientation. This tool orientation is normally used as the basis for the calculations of the movements of the orientation axes. The system variable can be used to define a situation in which, even with a tool active, it is not the tool orientation, but the orientation parameterized in the system variables $NT_BASE_ORIENT (Page 303) and $NT_BASE_ORIENT_NORMAL (Page 304) that is used for the calculations of the movements of the orientation axes.

Syntax

```
$NT_IGNORE_TOOL_ORIENT[<n>] = <Value>
```

Meaning

$NT_IGNORE_TOOL_ORIENT:	Switchover to the orientation parameterized in the system variables $NT_BASE_ORIENT and $NT_BASE_ORIENT_NORMAL for calculating the movements of the orientation axes	
	Data type:	BOOL
	Default value:	FALSE
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<Value>:	Value	
	Data type:	BOOL

Example

For the first transformation, the orientation parameterized in the system variables $NT_BASE_ORIENT and $NT_BASE_ORIENT_NORMAL for calculating the movements of the orientation axes is activated:

```
Program code                                    Comment
N100 $NT_IGNORE_TOOL_ORIENT[1] = TRUE           ; 1st transformation,
                                                ; orientation as per system variables
```

4.2.5.9 $NT_CORR_ELEM_T

Function

This system data is used to reference a maximum of 4 constant chain elements ($NK_NAME) in the tool chain, which are provided for accepting offset values (linear offsets), as they are determined in measurement cycles, for example. The offset values are calculated by the function CORRTRAFO (Page 318). This is only important for orientation transformations.

There must always be an orientation axis in the kinematic chain between two of these elements. This means that all 4 chain elements can only be occupied for 6-axis transformations in which all 3 orientation axes are defined in the tool chain; whereas, for 5-axis transformations, for example, this system data can only contain a maximum of three entries.

The entire kinematic chain, from the machine zero point (reference point of the kinematic chain) to the tool holder, is divided into a maximum of 4 sections by the orientation axes. There can be a maximum of one correction element in each of these sections. The correction element with the index n must be located in the nth section (example: $NT_CORR_ELEM_T[k, 1] must refer to a chain element between the first and second orientation axis of the tool chain).

Syntax

```
$NT_CORR_ELEM_T[<n>, <k>] = "<CORR_TElementName>"
```

Meaning

$NT_CORR_ELEM_T:	Names of the elements of the kinematic chain currently in effect.	
	Data type:	STRING
	Default value:	""
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<k>:	Position in the kinematic tool chain	
	Data type:	INT
	Value range:	0...3

<CORR_TElementName>:	Name of an element of the kinematic chain that is currently in effect, which accepts an offset value (linear offset).	
	Data type:	STRING

Example

Defines an correction element in the tool chain that is used to include offset values.

Program code	Comment
$NT_CORR_ELEM_T(1,2) = Corr_1;	Refers to a correction element in the tool chain of transformation "1" and position "2".

4.2.5.10 $NT_CORR_ELEM_P

Function

This system data is used to reference a maximum of 4 constant chain elements ($NK_NAME) in the part chain, which are provided for accepting offset values (linear offsets), as they are determined in measurement cycles, for example. The offset values are calculated by the function CORRTRAFO (Page 318). This is only important for orientation transformations.

There must always be an orientation axis in the kinematic chain between two of these elements. This means that all 4 chain elements can only be occupied for 6-axis transformations in which all 3 orientation axes are defined in the tool chain; whereas, for 5-axis transformations, for example, this system data can only contain a maximum of three entries.

The entire kinematic chain, from the machine zero point (reference point of the kinematic chain) to the workpiece center point (part), is divided into a maximum of 4 sections by the orientation axes. There can be a maximum of one correction element in each of these sections. The correction element with the index n must be located in the nth section (example: $NT_CORR_ELEM_T[k, 1] must refer to a chain element between the first and second orientation axis of the part chain).

Syntax

```
$NT_CORR_ELEM_P[<n>,<m>,<k>] = "<CORR_PElementName>"
```

Meaning

$NT_CORR_ELEM_P:	Names of the elements of the kinematic chain currently in effect.	
	Data type:	STRING
	Default value:	""
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Value range:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)

<k>:	Position in the kinematic tool chain	
	Data type:	INT
	Value range:	0...3
<CORR_PElementName>:	Name of an element of the kinematic chain that is currently in effect, which accepts an offset value (linear offset).	
	Data type:	STRING

Example

Defines an correction element in the workpiece chain that is used to accept offset values.

Program code	Comment
`$NT_CORR_ELEM_P(1,2) = Corr_1`	`Refers to a correction element in the workpiece chain of transformation "1" and position "2".`

4.2.6 Additive system variable for face transformation (TRANSMIT)

4.2.6.1 Overview

System variable	Meaning
$NT_POLE_SIDE_FIX	Limitation of the working area before and after the pole

The system variables are described in detail in the following sections.

Note

Element names

The system does not monitor as to whether the element names of the kinematic chain, which are referenced to parameterize transformation, were allocated a multiple number of times. If names such as these are available a multiple number of times, then the system data of the transformation always refer to the corresponding element with the lowest index.

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the kinematic chain. To do this, the system variables of the kinematic chain should be set to their default value using function DELOBJ().

Change system variable values

Changes to system variable values of the transformation data $NT_... only become effective when the NEWCONF machine data becomes effective. This can be done using the softkey on the user interface in the operating area "Commissioning" > "Machine data " > "Set MD active", in a program using the `NEWCONF` command or using a program end reset, channel reset or a warm or cold restart.

4.2.6.2 $NT_POLE_SIDE_FIX

Function

Using the system variable, the limitation of the work area must be set before and after the pole.

Syntax

```
$NT_POLE_SIDE_FIX[<n>] = <POLESIDEFIX>
```

Meaning

$NT_POLE_SIDE_FIX:	Names of the elements of the kinematic chain currently in effect, which define the rotary axes	
	Data type:	INT
	Default value:	""
	Value range:	Element names of the currently active kinematic chain
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<PoleSIDEFIX>:	Limiting the work area: • 0 = no limitation; traversing through the pole is allowed. • 1 = work area of the linear axis for positions ≥ 0 (if tool length offset is parallel to linear axis = 0). • 2 = work area of the linear axis for positions ≤ 0 (if tool length offset is parallel to linear axis = 0).	
	Data type:	STRING

Example

For the first transformation, the names of the elements of the kinematic chain currently in effect, which define the rotary axes, are "ROT_AXIS_1", ROT_AXIS_2" and "ROT_AXIS_3".

```
Program code                                    Comment
                                                ; 1st transformation,
                                                ; elements of the rotary axes:
N100 $NT_ROT_AX_NAME[1,0] = "ROT_AXIS_1"        ; 1st rotary axis
N110 $NT_ROT_AX_NAME[1,1] = "ROT_AXIS_2"        ; 2nd rotary axis
N120 $NT_ROT_AX_NAME[1,2] = "ROT_AXIS_3"        ; 3rd rotary axis
```

4.2.7 Additive system variables for transformation chains (TRACON_K)

4.2.7.1 $NT_TRACON_CHAIN

Function

For chained transformations ($NT_TRAFO_TYPE[...] = "TRACON_K") with this system variable, the names of the part transformations are specified in the order in which the transformation of the BKS to the MKS is to be carried out.

Syntax

```
$NT_TRACON_CHAIN[<n>,<m>] = "<Name>"
```

Meaning

$NT_TRACON_CHAIN:	System variable for specifying the partial transformations for chained transformations	
<n>:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
<m>:	Position of the partial transformation in the transformation chain	
	Data type:	INT
	Range of values:	0, 1
<name>:	Name of the partial transformation	
	Data type:	STRING

Example

```
Program code                                                              Comment
...
N2502 $NT_TRACON_CHAIN[2,0] = "Open Architecture Transformation"          ; Partial transformation 1
N2503 $NT_TRACON_CHAIN[2,1] = "Inclined axis"                             ; Partial transformation 2
...
```

4.3 Programming

4.3.1 Activating a transformation (TRAFOON)

A transformation defined with kinematic chains is activated with the predefined TRAFOON procedure. The call must be alone in a block.

Note

Alternatively, a transformation defined with kinematic chains can also be activated via conventional NC commands, such as TRAORI or TRANSMIT. For this purpose, an appropriate value, not equal to zero, must be entered in the $NT_TRAFO_INDEX system variable.

For further information on $NT_TRAFO_INDEX see "System Variables List Manual".

Syntax

```
TRAFOON(<Trafoname>,<Diameter>,<k>)
```

Meaning

TRAFOON:	Procedure for activating a transformation defined with kinematic chains		
<Trafoname>:	Name of the transformation data set		
	Data type:	STRING	
	Range of values:	All names of transformation data sets defined via $NK_NAME (Page 279)	
	Note: The name of the transformation data set must be unique. It must only occur once in $NT_NAME.		
<Diameter>:	Reference or working diameter (TRACYL only)		
	Data type:	REAL	
	The value must be > 1.		
<k>:	Defines the use of the groove side offset (TRACYL only).		
	Data type:	BOOL	
	Value:	FALSE	Without groove side offset
		TRUE	With groove side offset
	Corresponds to the TRACYL transformation type 514 (groove side offset can be programmed). If <k> is not specified, the parameterized setting of bit 10 in $NT_CNTRL[<n>] applies.		

Example

Program code	Comment
TRAFOON["Trans_1"]	Activates the transformation with the name Trans_1.

4.3.2 Modifying the orientation transformation after the machine measurement (CORRTRAFO)

For machines with orientation transformations that were defined by means of kinematic chains, the user can use the predefined CORRTRAFO function in order to modify the offset vectors or the direction vectors of the orientation axes in the kinematic model of the machine after a machine measurement.

Note

The correction values written with the CORRTRAFO function are not immediately effective in the transformation. The correction values do not become effective until after a transformation deselection, NEWCONF and transformation selection.

Syntax

```
<Corr_Status> = CORRTRAFO(<Corr_Vect>, <Corr_Index>, <Corr_Mode>,
[ <No_Alarm>])
```

Meaning

CORRTRAFO:	Function call

<Corr_Status>:	Function return value	
	Data type:	INT
	Values:	

0	The function was executed without an error.
1	No transformation is active.
2	The currently active transformation is not an orientation transformation.
3	The active orientation transformation was not defined with kinematic chains.
10	The <Corr_Index> call parameter is negative.
11	The <Corr_Mode> call parameter is negative.
12	Invalid reference to a section of a subchain (units position of <Corr_Index>). The value must not be greater than the number of orientation axes in the subchain.
13	Invalid reference to the orientation axis of a subchain (units position of <Corr_Index>). The value must be less than the number of orientation axes in the subchain.
14	Invalid reference to a subchain (tens position of <Corr_Index>). Only the values 0 and 1 are permissible (reference to part or tool chain). This error number occurs if the subchain to which <Corr_Index> refers does not exist.
15	There is no correction element in the section referred to with the <Corr_Index> parameter ($NT_CORR_ELEM_P or $NT_CORR_ELEM_T).
20	Invalid correction mode (units position of <Corr_Mode>). Only the values 0 and 1 are permissible.
21	Invalid correction mode (tens and/or hundreds position of <Corr_Mode>). Only the units position can be not equal to zero when writing an axis direction.
30	The hundreds position of <Corr_Mode> is invalid. Only the values 0 and 1 are permissible.
31	The thousands position of <Corr_Mode> is invalid. Only the values 0 and 1 are permissible.
40	The direction vector that is to be taken as axis direction is the zero vector. This error can only happen if the thousands position of <Corr_Mode> is equal to 0. If the thousands position of this parameter is equal to 1 (monitoring of the maximum correction deactivated), the zero vector can also be written.
41	For the correction of an offset vector, the difference to the current value in at least one coordinate is greater than the maximum value specified by the setting data SD41610 $SN_CORR_TRAFO_LIN_MAX. The <Corr_Vect> parameter will be overwritten by an error vector. This also applies when the processing is aborted with alarm (see <No_Alarm> parameter). In the components whose correction value has exceeded the permissible limit, the error vector has the difference, with the correct sign, between the determined correction value and the limit. The content of the components that have not exceeded their limit is zero.

		42	For the correction of a direction vector, the angular displacement compared to the current direction is greater than the maximum value specified by the setting data SD41611 $SN_CORR_TRAFO_DIR_MAX.
		43	The attempt to write a system variable was rejected because of missing write rights.
<Corr_Vect>:	Correction vector The content of the correction vector is defined by the following parameters **<**Corr_Index**>** and <Corr_Mode>. If <Corr_Status> = 41, the content of the vector is overwritten (see above).		
	Data type:	REAL	
<Corr_Index**>**:	Section whose correction element is to be modified / index of the orientation axis whose direction vector is to be modified		
	Data type:	INT	
	The <Corr_Index> parameter is decimal coded (units to tens position):		
	Units position:	Contains the index of the section or the orientation axis in the subchain.	
	Tens position:	Refers to the subchain.	
		0x	Workpiece chain
		1x	Tool chain

<Corr_Mode>:	Correction mode		
	Data type:	INT	
	The <Corr_Index> parameter is decimal coded (units to thousands position):		
	Units position:	Specifies which element is to be corrected.	
		xxx**0**	Correction of a linear offset vector
		xxx**1**	Correction of the direction vector of an orientation axis
	Tens position:	Specifies how the correction element to which the content of <Corr_Index> refers, is to be modified.	
		xx**0**x	The correction vector is written immediately to the correction element. This variant can be used to immediately write the correction element without the index <n> of the relevant system data ($NK_OFF_DIR[<n>, ...]) having to be known.
		xx**1**x	As 0, but with the difference that the transferred correction value is interpreted in world coordinates. A difference between variants 0 and 1 can always occur when the kinematic chain in the initial state (positions of all orientation axes equal to 0) contains other rotations.
		xx**2**x	As 1, but with the difference that the correction value refers to the entire section, i.e. a value is entered in the correction element so that the entire section reaches the length defined by the correction value.
		Note: The values 1 and 2 are not permissible when writing the direction vector of an orientation axis.	
	Hundreds position:	Specifies how the content of the <Corr_Vect> parameter is to be interpreted.	
		x**0**xx	The transferred correction vector <Corr_Vect> contains the entire new length of the correction element or the section to which the <Corr_Index> in conjunction with the tens position of <Corr_Mode> refers (absolute correction).
		x**1**xx	The transferred correction vector <Corr_Vect> only contains the difference compared to the current length of the correction element or the section to which the <Corr_Index> in conjunction with the tens position of <Corr_Mode> refers (incremental correction).
		Note: For the correction of the direction vector of an orientation axis, the content of the hundreds position must be 0.	
	Thousands position:	Specifies whether the correction is to be limited by the following maximum value: • SD41610 $SN_CORR_TRAFO_LIN_MAX (Page 276) or • SD41611 $SN_CORR_TRAFO_DIR_MAX (Page 276)	
		0xxx	Monitoring of the maximum correction is active.
		1xxx	Monitoring of the maximum correction is not active.

<No_Alarm>:	Behavior in the event of an error (return value > 0) (**optional**)		
	Data type:	BOOL	
	Value:	FALSE (default)	In the event of an error, the program processing is stopped and alarm 14103 is displayed.
		TRUE	In the event of an error, the program processing is not stopped and no alarm is displayed. Application: User-specific reaction corresponding to the return value

Note

In the event of an error when the function is called, either an alarm is output or an error number returned (see <No_Alarm> parameter), so that the user can respond in a suitable way to the error state. The cause of the error is described in more detail through an alarm parameter. An error number returned instead of an alarm is identical to the alarm parameter.

See also

Angular deviation for rotation vectors for CORRTRAFO (Page 276)

Further information on CORRTRAFO

The kinematic structure of a machine with orientation transformation is described by one or two kinematic chains (subchains), starting from the zero point of the world coordinate system. One of the two chains, the **tool chain**, ends at the reference point of the tool, the other chain, the **workpiece chain** ends in the zero point of the basic coordinate system.

The CORRTRAFO function writes lever arm lengths and axis directions on machines with an orientation transformation in special correction elements. A kinematic chain is described, for example, with elements of the type OFFSET, which are defined via $ NK_TYPE.

CORRTRAFO works with sections

The two subchains can each be divided into a maximum of four sections:

- Section 1 begins at the starting point of the chain and ends at the first orientation axis.
- Section 2 is the section between orientation axis 1 and orientation axis 2.
- Section 3 is the section between orientation axis 2 and orientation axis 3.
- Section 4 is the section between orientation axis 3 and the end of the tool or workpiece chain.

Each section may contain constant chain elements of the type OFFSET or ROT_CONST.

The following figure shows an orientation transformation with 2 orientation axes.

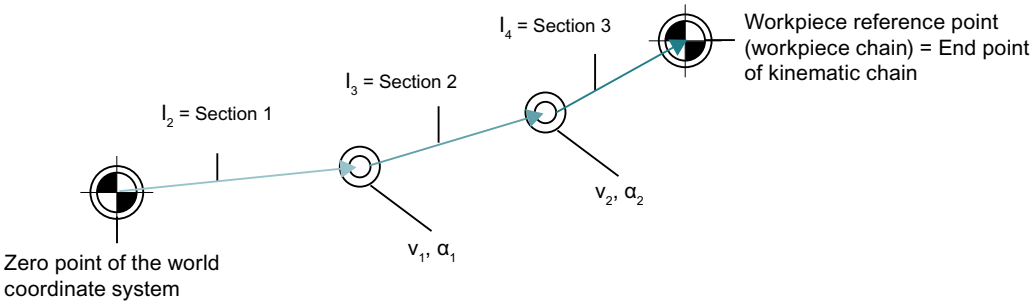


Figure 4-7 CORRTRAFO example

The sections are clearly defined: If you run through the kinematic subchain from the starting point to the end point, the first section has the index 0, the next the index 1, and so on. The index of the last section is then always equal to the number of orientation axes.

Correction elements

A reference can be made to a constant kinematic chain element (chain element of the type $NK_TYPE[<n>] = "OFFSET") in each section with the $NT_CORR_ELEM_T[<n>, 0 ... 3] or $NT_CORR_ELEM_P[<n>, 0 ... 3] system variables. The correction values that were determined during the machine measurement are written to these elements with the CORRTRAFO function.

Example with transformation index = 1:

- $NT_CORR_ELEM_T[1,0] = "C_AXIS_OFFSET"; Offset of the C axis (orientation axis 1) in section 1 is defined as correction element.
- $NT_CORR_ELEM_T[1,1] = "B_AXIS_OFFSET"; Offset of the B axis (orientation axis 2) in section 2 is defined as correction element.
- $NT_CORR_ELEM_T[1,2] = "BASE_TOOL_OFFSET"; Offset of the B axis from the tool reference point in section 3 is defined as correction element.

The sequence of the references in $NT_CORR_ELEM_T/P[<n>, 0 ... 3] must correspond to the sections described above, i.e. only one chain element can be in $NT_CORR_ELEM_T/P [<n>, 0] which is before the first orientation axis, etc.

The CORRTRAFO function writes the values determined by measuring the machine into the correction elements defined in this way. The modification of the correction values is defined in CORRTRAFO via the <Corr_Mode> parameter.

Closing a chain

If bit 7 or bit 8 are set in the $NT_CNTRL[<n>] system variable, additional constant chain elements that establish a connection from the end point of the chain to the machine zero point are automatically inserted internally at the end of the workpiece chain (bit 7) or before the starting point of the tool chain (bit 8) ("close chain").

These automatically inserted elements cannot be written externally, only read (see the $AC_TRAFO_CORR_ELEM_P/T system variables).

Point to close the tool chain

If the $NT_CLOSE_CHAIN_T system variable is not empty, the tool chain is not closed at the end point of the chain, but rather at the end point of the designated chain element. Other chain elements that are behind this point result in a corresponding work offset when the transformation is activated.

Index of an orientation axis

In addition to the constants offsets between the orientation axes, the direction vectors of the orientation axes can also be written with the CORRTRAFO function. The index of an orientation axis is the index that results when the kinematic subchain is run through from the origin to the end, where the count starts at zero. The index of an orientation axis is therefore always the same as the index of the preceding section.

The index of an orientation axis can also be determined with the $AC_TRAFO_ORIAX_LOC system variable.

Maximum permissible change of a chain element

The maximum permissible change of a chain element can be limited by the two setting data SD41610 $SN_CORR_TRAFO_LIN_MAX for offset vectors and SD41611 $SN_CORR_TRAFO_DIR_MAX for direction vectors of the orientation axes. SD41610 $SN_CORR_TRAFO_LIN_MAX specifies the maximum amount by which each individual vector component can be changed with regard to its reference value. SD41611 $SN_CORR_TRAFO_DIR_MAX specifies the maximum angle by which the direction of the axis vector can be changed with regard to its reference value. The reference value is always the corresponding value that is active in the transformation that is active when CORRTRAFO is called. This means that the changed content of the kinematic data may have no effect on the method of operation of the CORRTRAFO function after the activation of the transformation.

4.4 Examples

4.4.1 Settings for TRAORI_DYN

General Information

An example of the basic procedure for parameterizing a transformation with a kinematic chain via a part program is shown on the basis of a 5-axis machine. All of the system variables, relevant for kinematic chain, are written to the part program:

- Kinematic chains with $NK_...
- Transformation with kinematic chain $NT_...

Option and machine data

The following option and machine data must be set for the example:

- MD18866 $MN_MM_NUM_KIN_TRAFOS = 2

Transformation with a kinematic chain

Two kinematic chains are defined for the description:

- One kinematic chain points to the workpiece reference point (workpiece coordinate system).
- The second kinematic chain points to the tool reference point.
- The transformation type is "TRAORI_DYN".

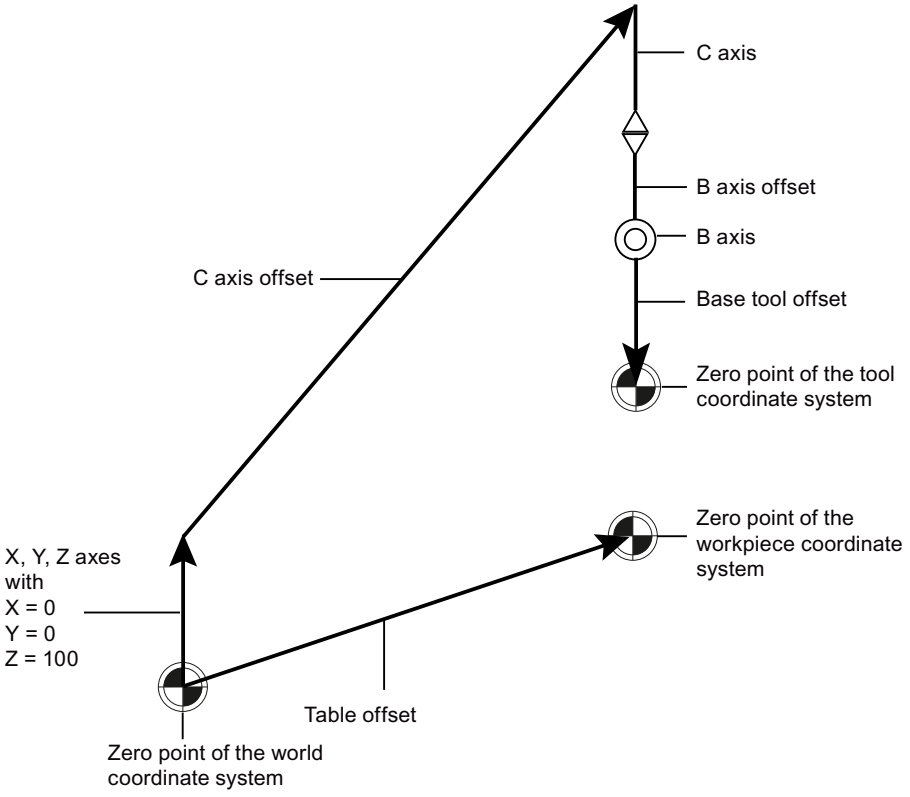


Figure 4-8 Example kinematics for dynamic orientation transformation

The kinematics is created in several steps:

- The two root elements of the kinematic chain are defined:
  - `$NK_NEXT\[KIE_CNTR] = "X axis"`; chain points to the tool reference point.
  - `$NK_PARALLEL\[KIE_CNTR] = "Table offset"`; chain points to the workpiece reference point.
- The three mutually perpendicular linear axes X, Y and Z are defined starting from the zero point of the world coordinate system.
  - In the rotation vectors `$NK_OFF_DIR`, which define the axis directions, only the individual component that is not equal to zero is described. All of the linear axes have their origin in the same point, because no constant elements are defined between the individual axes and no zero point offsets have been specified for the axes either.

- The C axis offset is a constant element, which defines the distance between the linear axes X, Y, Z and the first rotary axis (C axis). The C axis is a rotary axis which points in the Z direction.
- The B axis offset is a constant element, to which the B axis connects.
- The conclusion is formed by the constant offset "`Basetool`".
- The kinematic chain to the workpiece reference point is formed by the constant element "`Table offset`" of the type Offset.
- The following orientation axes are defined:
  - C axis
  - B axis
- The following linear compensating axes are defined:
  - X axis
  - Y axis
  - Z axis

4.4.2 Part program for TRAORI_DYN

Example program "5-axis transformation C-B"

Program code

```
;============================================================
; Definitions
;============================================================
N10 DEF INT KIE_CNTR = 0  ; counter for elements of the kin. chains
;============================================================
; deletion of all transformation data sets and kinematic chain elements
;============================================================
N20 IF (DELOBJ("TRAFO_DATA") < 0)
N30     SETAL(61000)
N40 ENDIF

N50 IF (DELOBJ("KIN_CHAIN_ELEM") < 0)
N60     SETAL(61001)
N70 ENDIF
;============================================================
; definition of the root element and of the kinematic chain to the tool
reference point
;============================================================
N80 $NK_NAME[KIE_CNTR]        = "ROOT"
N90 $NK_TYPE[KIE_CNTR]        = "OFFSET"
N100 $NK_NEXT[KIE_CNTR]       = "X axis"
N110 $NK_PARALLEL[KIE_CNTR]   = "Table offset"
N120 KIE_CNTR = KIE_CNTR + 1
;============================================================
; Definition of the linear axis in X direction
;============================================================
N130 $NK_NAME[KIE_CNTR]       = "X axis"
N140 $NK_TYPE[KIE_CNTR]       = "AXIS_LIN"
N150 $NK_NEXT[KIE_CNTR]       = "Y axis"
N160 $NK_AXIS[KIE_CNTR]       = "X1"
N170 $NK_OFF_DIR[KIE_CNTR,0]  = 1.0
N180 KIE_CNTR = KIE_CNTR + 1
;============================================================
; definition of the linear axis in Y direction
;============================================================
N190 $NK_NAME[KIE_CNTR]       = "Y axis"
N200 $NK_TYPE[KIE_CNTR]       = "AXIS_LIN"
N210 $NK_NEXT[KIE_CNTR]       = "Z axis"
N220 $NK_AXIS[KIE_CNTR]       = "Y1"
N230 $NK_OFF_DIR[KIE_CNTR,1]  = 1.0
N240 KIE_CNTR = KIE_CNTR + 1
;============================================================
; Definition of the linear axis in Z direction
;============================================================
N250 $NK_NAME[KIE_CNTR]       = "Z axis"
N260 $NK_TYPE[KIE_CNTR]       = "AXIS_LIN"
N270 $NK_NEXT[KIE_CNTR]       = "C axis offset"
N280 $NK_AXIS[KIE_CNTR]       = "Z1"
N290 $NK_OFF_DIR[KIE_CNTR,2]  = 1.0
N300 KIE_CNTR = KIE_CNTR + 1
;============================================================
; definition of the distance between the linear axes and the first rotary
axis
;============================================================
N310 $NK_NAME[KIE_CNTR]       = "C axis offset"
```

Program code

```
N320 $NK_TYPE[KIE_CNTR]       = "OFFSET"
N330 $NK_NEXT[KIE_CNTR]       = "C axis"
N340 $NK_OFF_DIR[KIE_CNTR,0]  = 200.0

N350 $NK_OFF_DIR[KIE_CNTR,2]  = 300.0
N360 KIE_CNTR = KIE_CNTR + 1
;============================================================
; definition of the C axis in the Z direction - refers to axis C1
;============================================================
N370 $NK_NAME[KIE_CNTR]       = "C axis"
N380 $NK_TYPE[KIE_CNTR]       = "AXIS_ROT"
N390 $NK_NEXT[KIE_CNTR]       = "B axis offset"
N400 $NK_AXIS[KIE_CNTR]       = "C1"
N410 $NK_OFF_DIR[KIE_CNTR,2]  = 1.0
N420 KIE_CNTR = KIE_CNTR + 1
;============================================================
; definition of the B axis with an offset between the B and C axes
;============================================================
N430 $NK_NAME[KIE_CNTR]       = "B axis offset"
N440 $NK_TYPE[KIE_CNTR]       = "OFFSET"
N450 $NK_NEXT[KIE_CNTR]       = "B axis"
N460 $NK_OFF_DIR[KIE_CNTR,2]  = -150.0
N470 KIE_CNTR = KIE_CNTR + 1

N480 $NK_NAME[KIE_CNTR]       = "B axis"
N490 $NK_TYPE[KIE_CNTR]       = "AXIS_ROT"
N500 $NK_NEXT[KIE_CNTR]       = "Base tool"
N510 $NK_AXIS[KIE_CNTR]       = "B1"
N520 $NK_OFF_DIR[KIE_CNTR,0]  = 1.0
N530 KIE_CNTR = KIE_CNTR + 1
;============================================================
; conclusion of the kinematic chain with an offset
;============================================================
N540 $NK_NAME[KIE_CNTR]       = "Base tool"
N550 $NK_TYPE[KIE_CNTR]       = "OFFSET"
N560 $NK_NEXT[KIE_CNTR]       = ""
N570 $NK_OFF_DIR[KIE_CNTR,2]  = -50.0
;============================================================
; definition of the kinematic chain to the table reference point
;============================================================
N580 KIE_CNTR = KIE_CNTR + 1
N590 $NK_NAME[KIE_CNTR]       = "Table offset"
N600 $NK_TYPE[KIE_CNTR]       = "OFFSET"
N610 $NK_OFF_DIR[KIE_CNTR,0] = 200.0
N620 $NK_OFF_DIR[KIE_CNTR,2]  = 100.0
N630 KIE_CNTR = KIE_CNTR + 1
;============================================================
; Definition of the transformation (TRAORI)
;============================================================
N640 $NT_NAME[1]              = "5-axis transformation C-B"
N650 $NT_T_CHAIN_LAST_ELEM[1] = "base tool"
N660 $NT_P_CHAIN_LAST_ELEM[1] = "Table offset"
N670 $NT_TRAFO_TYPE[1]        = "TRAORI_DYN"
;============================================================
; Definition of the orientation and geometry axes
;============================================================
```

Program code

```
N680 $NT_ROT_AX_NAME[1,0]     = "C axis"
N690 $NT_ROT_AX_NAME[1,1]     = "B axis"
N700 $NT_GEO_AX_NAME[1,0]     = "X axis"
N710 $NT_GEO_AX_NAME[1,1]     = "Y axis"
N720 $NT_GEO_AX_NAME[1,2]     = "Z axis"
End
```

4.4.3 Part program for TRANSMIT

Example program "5-axis transformation C-B"

Program code

```
;============================================================

; Simple example of TRANSMIT with kinematic chain:
;*****************************************************
N10     DEF INT _KIE_CNTR
N20     DEF INT _TRA_CNTR
N30     R2 = DELOBJ("TRAFO_DATA")
N40     R2 = DELOBJ("KIN_CHAIN_ELEM")
N50     _KIE_CNTR  = 0
N60     _TRA_CNTR  = 1
; Definition of the kinematic chain
;*****************************************************
N70     $NK_NAME[_KIE_CNTR]       = "ROOT"
N80     $NK_TYPE[_KIE_CNTR]       = "OFFSET"
N90     $NK_NEXT[_KIE_CNTR]       = "X-Axis"
N100    $NK_PARALLEL[_KIE_CNTR]   = "C-Axis"
N110    _KIE_CNTR = _KIE_CNTR + 1
N120    $NK_NAME[_KIE_CNTR]       = "X-Axis"
N130    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N140    $NK_NEXT[_KIE_CNTR]       = "Z-Axis"
N150    $NK_AXIS[_KIE_CNTR]       = "X1"
N160    $NK_OFF_DIR[_KIE_CNTR,0]  = 1.0
N170    _KIE_CNTR = _KIE_CNTR + 1
N180    $NK_NAME[_KIE_CNTR]       = "Z-Axis"
N190    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N200    $NK_NEXT[_KIE_CNTR]       = "Y-Axis"
N210    $NK_AXIS[_KIE_CNTR]       = "Z1"
N220    $NK_OFF_DIR[_KIE_CNTR,2]  = 1.0
N230    _KIE_CNTR = _KIE_CNTR + 1
N240    $NK_NAME[_KIE_CNTR]       = "Y-Axis"
N250    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N260    $NK_NEXT[_KIE_CNTR]       = ""
N270    $NK_AXIS[_KIE_CNTR]       = "Y1"
N280    $NK_OFF_DIR[_KIE_CNTR,1]  = 1.0
N290    _KIE_CNTR = _KIE_CNTR + 1
N300    $NK_NAME[_KIE_CNTR]       = "C-Axis"
N310    $NK_TYPE[_KIE_CNTR]       = "AXIS_ROT"
N320    $NK_NEXT[_KIE_CNTR]       = ""
N330    $NK_AXIS[_KIE_CNTR]       = "C1"
N340    $NK_OFF_DIR[_KIE_CNTR,2]  = -1.0
N350    _KIE_CNTR = _KIE_CNTR + 1

; Definition of the kinematic transformation:
;*****************************************************
; 1. TRANSMIT 256
;*****************************************************
N360    $NT_NAME[_TRA_CNTR]                  = "Trafo Transmit_1"
N370    $NT_TRAFO_TYPE[_TRA_CNTR]            = "TRANSMIT_K"
N380    $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]     = "C-Axis"
N390    $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]     = "Z-Axis"
N400    $NT_GEO_AX_NAME[_TRA_CNTR,0]         = "X-Axis"
N410    $NT_GEO_AX_NAME[_TRA_CNTR,1]         = ""
N420    $NT_GEO_AX_NAME[_TRA_CNTR,2]         = "Z-Axis"
N430    $NT_ROT_AX_NAME[_TRA_CNTR,0]         = ""
N440    $NT_ROT_AX_NAME[_TRA_CNTR,1]         = "C-Axis"
```

Program code

```
N450    $NT_ROT_AX_NAME[_TRA_CNTR,2]         = ""
N460    $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N470    $NT_CNTRL[_TRA_CNTR]                 = 'H0'
N480    _TRA_CNTR = _TRA_CNTR + 1
; 2nd TRANSMIT 257
;*****************************************************
N490    $NT_NAME[_TRA_CNTR]                  = "Trafo Transmit_2"
N500    $NT_TRAFO_TYPE[_TRA_CNTR]            = "TRANSMIT_K"
N510    $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]     = "C-Axis"
N520    $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]     = "Y-Axis"
N530    $NT_GEO_AX_NAME[_TRA_CNTR,0]         = "X-Axis"
N540    $NT_GEO_AX_NAME[_TRA_CNTR,1]         = "Y-Axis"
N550    $NT_GEO_AX_NAME[_TRA_CNTR,2]         = "Z-Axis"
N560    $NT_ROT_AX_NAME[_TRA_CNTR,0]         = ""
N570    $NT_ROT_AX_NAME[_TRA_CNTR,1]         = "C-Axis"
N580    $NT_ROT_AX_NAME[_TRA_CNTR,2]         = ""
N590    $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N600    $NT_CNTRL[_TRA_CNTR]                 = 'H200'    ; TRANSMIT 257
N610    _TRA_CNTR = _TRA_CNTR + 1

;*****************************************************
; Activate kinematic chain and transformation:
;*****************************************************
N620    NEWCONF
N630    STOPRE
;*****************************************************************************
************************

End
```

4.4.4 Part program for TRACYL

Example program "TRACYL"

Program code

```
;============================================================

; Simple example of TRACYL with kinematic chain:
;*****************************************************
N640    $NK_NAME[_KIE_CNTR]       = "ROOT"
N650    $NK_TYPE[_KIE_CNTR]       = "OFFSET"
N660    $NK_NEXT[_KIE_CNTR]       = "Z-Axis"
N670    $NK_PARALLEL[_KIE_CNTR]   = "C-Axis"
N680    _KIE_CNTR = _KIE_CNTR + 1
N690    $NK_NAME[_KIE_CNTR]       = "Z-Axis"
N700    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N710    $NK_NEXT[_KIE_CNTR]       = "X-Axis"
N720    $NK_AXIS[_KIE_CNTR]       = "Z1"
N730    $NK_OFF_DIR[_KIE_CNTR,2]  = 1.0
N740    _KIE_CNTR = _KIE_CNTR + 1
N750    $NK_NAME[_KIE_CNTR]       = "X-Axis"
N760    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N770    $NK_NEXT[_KIE_CNTR]       = "Y-Axis"
N780    $NK_AXIS[_KIE_CNTR]       = "X1"
N790    $NK_OFF_DIR[_KIE_CNTR,0]  = 1.0
N800    _KIE_CNTR = _KIE_CNTR + 1
N810    $NK_NAME[_KIE_CNTR]       = "Y-Axis"
N820    $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N830    $NK_NEXT[_KIE_CNTR]       = ""
N840    $NK_AXIS[_KIE_CNTR]       = "Y1"
N850    $NK_OFF_DIR[_KIE_CNTR,1]  = 1.0
N860    _KIE_CNTR = _KIE_CNTR + 1
N870    $NK_NAME[_KIE_CNTR]       = "C-Axis"
N880    $NK_TYPE[_KIE_CNTR]       = "AXIS_ROT"
N890    $NK_NEXT[_KIE_CNTR]       = ""
N900    $NK_AXIS[_KIE_CNTR]       = "C1"
N910    $NK_OFF_DIR[_KIE_CNTR,2]  = -1.0

; Definition of the kinematic transformation:
;*****************************************************
; 1. TRACYL 512
;*****************************************************
N920    $NT_NAME[_TRA_CNTR]                   = "Trafo Tracyl 512"
N930    $NT_TRAFO_TYPE[_TRA_CNTR]             = "TRACYL_K"
N940    $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]      = "X-Axis"
N950    $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]      = "C-Axis"
N960    $NT_GEO_AX_NAME[_TRA_CNTR,0]          = "X-Axis"
N970    $NT_GEO_AX_NAME[_TRA_CNTR,1]          = ""
N980    $NT_GEO_AX_NAME[_TRA_CNTR,2]          = "Z-Axis"
N990    $NT_ROT_AX_NAME[_TRA_CNTR,0]          = ""
N1000   $NT_ROT_AX_NAME[_TRA_CNTR,1]          = "C-Axis"
N1010   $NT_ROT_AX_NAME[_TRA_CNTR,2]          = ""
N1020   $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N1030   $NT_CNTRL[_TRA_CNTR]                  = 'H0'
N1040   _TRA_CNTR = _TRA_CNTR + 1
; 2nd TRACYL 513
;*****************************************************
N1050   $NT_NAME[_TRA_CNTR]                   = "Trafo Tracyl 513"
N1060   $NT_TRAFO_TYPE[_TRA_CNTR]             = "TRACYL_K"
N1070   $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]      = "Y-Axis"
```

Program code

```
N1080   $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]     = "C-Axis"
N1090   $NT_GEO_AX_NAME[_TRA_CNTR,0]         = "X-Axis"
N1100   $NT_GEO_AX_NAME[_TRA_CNTR,1]         = "Y-Axis"
N1110   $NT_GEO_AX_NAME[_TRA_CNTR,2]         = "Z-Axis"
N1120   $NT_ROT_AX_NAME[_TRA_CNTR,0]         = ""
N1130   $NT_ROT_AX_NAME[_TRA_CNTR,1]         = "C-Axis"
N1140   $NT_ROT_AX_NAME[_TRA_CNTR,2]         = ""
N1150   $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N1160   $NT_CNTRL[_TRA_CNTR]                 = 'H200'    ; TRACYL 513
N1170   _TRA_CNTR = _TRA_CNTR + 1
; 3rd TRACYL 514
;*****************************************************
N1180   $NT_NAME[_TRA_CNTR]                  = "Trafo Tracyl 514"
N1190   $NT_TRAFO_TYPE[_TRA_CNTR]            = "TRACYL_K"
N1200   $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]     = "Y-Axis"
N1210   $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]     = "C-Axis"
N1220   $NT_GEO_AX_NAME[_TRA_CNTR,0]         = "X-Axis"
N1230   $NT_GEO_AX_NAME[_TRA_CNTR,1]         = "Y-Axis"
N1240   $NT_GEO_AX_NAME[_TRA_CNTR,2]         = "Z-Axis"
N1250   $NT_ROT_AX_NAME[_TRA_CNTR,0]         = ""
N1260   $NT_ROT_AX_NAME[_TRA_CNTR,1]         = "C-Axis"
N1270   $NT_ROT_AX_NAME[_TRA_CNTR,2]         = ""
N1280   $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N1290   $NT_CNTRL[_TRA_CNTR]                 = 'H400'    ; TRACYL 514
N1300   _TRA_CNTR = _TRA_CNTR + 1

;*****************************************************
;Activate kinematic chain and transformation:
;*****************************************************
N1310   NEWCONF
N1320   STOPRE
;*****************************************************************************
************************
End
```

4.4.5 Part program for TRAANG

Example program "TRAANG"

Program code

```
;============================================================

; Simple example of TRAANG with kinematic chain
;*****************************************************
N2000   $NK_NAME[_KIE_CNTR]       = "ROOT"
N2010   $NK_TYPE[_KIE_CNTR]       = "OFFSET"
N2020   $NK_NEXT[_KIE_CNTR]       = "X-Axis"
N2030   $NK_PARALLEL[_KIE_CNTR]   = ""
N2040   _KIE_CNTR = _KIE_CNTR + 1
N2050   $NK_NAME[_KIE_CNTR]       = "X-Axis"
N2060   $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N2070   $NK_NEXT[_KIE_CNTR]       = "Z-Axis"
N2080   $NK_AXIS[_KIE_CNTR]       = "X1"
N2090   $NK_OFF_DIR[_KIE_CNTR,0]  = COS(20)
N2100   $NK_OFF_DIR[_KIE_CNTR,2]  = SIN(20)
N2110   _KIE_CNTR = _KIE_CNTR + 1
N2120   $NK_NAME[_KIE_CNTR]       = "Z-Axis"
N2130   $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N2140   $NK_NEXT[_KIE_CNTR]       = "Y-Axis"
N2150   $NK_AXIS[_KIE_CNTR]       = "Z1"
N2160   $NK_OFF_DIR[_KIE_CNTR,2]  = 1.0
N2170   _KIE_CNTR = _KIE_CNTR + 1
N2180   $NK_NAME[_KIE_CNTR]       = "Y-Axis"
N2190   $NK_TYPE[_KIE_CNTR]       = "AXIS_LIN"
N2200   $NK_NEXT[_KIE_CNTR]       = ""
N2210   $NK_AXIS[_KIE_CNTR]       = "Y1"
N2220   $NK_OFF_DIR[_KIE_CNTR,1]  = 1.0
N2230   _KIE_CNTR = _KIE_CNTR + 1

; definition of the kinematic transformation:
;*****************************************************
; TRRANG
;*****************************************************
N2240   $NT_NAME[_TRA_CNTR]                  = "Trafo Traang"
N2250   $NT_TRAFO_TYPE[_TRA_CNTR]            = "TRAANG_K"
N2260   $NT_T_CHAIN_LAST_ELEM[_TRA_CNTR]     = "Y-Axis"
N2270   $NT_P_CHAIN_LAST_ELEM[_TRA_CNTR]     = ""
N2280   $NT_GEO_AX_NAME[_TRA_CNTR,0]         = "X-Axis"
N2290   $NT_GEO_AX_NAME[_TRA_CNTR,2]         = "Y-Axis
N2300   $NT_GEO_AX_NAME[_TRA_CNTR,1]         = "Z-Axis"
N2310   $NT_ROT_OFFSET_FROM_FRAME[_TRA_CNTR] = 1
N2320   _TRA_CNTR = _TRA_CNTR + 1

;*****************************************************
; Activate kinematic chain and transformation:
;*****************************************************
N2330   NEWCONF
N2340   STOPRE
;*************************************************************************
************************
```

Program code

```
End
```

Appendix

A

A.1 List of abbreviations

A	
O	Output
ADI4	(Analog drive interface for 4 axes)
AC	Adaptive Control
ALM	Active Line Module
ARM	Rotating induction motor
AS	Automation system
ASCII	American Standard Code for Information Interchange: American coding standard for the exchange of information
ASIC	Application-Specific Integrated Circuit: User switching circuit
ASUB	Asynchronous subprogram
AUXFU	Auxiliary function: Auxiliary function
STL	Statement List
UP	User Program

B	
OP	Operating Mode
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BERO	Contact-less proximity switch
BI	Binector Input
BICO	Binector Connector
BIN	BINary files: Binary files
BIOS	Basic Input Output System
BCS	Basic Coordinate System
BO	Binector Output
OPI	Operator Panel Interface

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle: Compile cycles
CEC	Cross Error Compensation
CI	Connector Input
CF Card	Compact Flash Card

C	
CNC	Computerized Numerical Control: Computer-Supported Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication
CPA	Compiler Projecting Data: Configuring data of the compiler
CRT	Cathode Ray Tube: picture tube
CSB	Central Service Board: PLC module
CU	Control Unit
CP	Communication Processor
CPU	Central Processing Unit: Central processing unit
CR	Carriage Return
CTS	Clear To Send: Ready to send signal for serial data interfaces
CUTCOM	Cutter radius Compensation: Tool radius compensation

D	
DAC	Digital-to-Analog Converter
DB	Data Block (PLC)
DBB	Data Block Byte (PLC)
DBD	Data Block Double word (PLC)
DBW	Data Block Word (PLC)
DBX	Data block bit (PLC)
DDE	Dynamic Data Exchange
DDS	Drive Data Set: Drive data set
DIN	Deutsche Industrie Norm
DIO	Data Input/Output: Data transfer display
DIR	Directory: Directory
DLL	Dynamic Link Library
DO	Drive Object
DPM	Dual Port Memory
DPR	Dual Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function: Differential revolver function (handwheel)
DRIVE-CLiQ	Drive Component Link with IQ
DRY	Dry Run: Dry run feedrate
DSB	Decoding Single Block: Decoding single block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DW	Data Word
DWORD	Double Word (currently 32 bits)

E	
I	Input
EES	Execution from External Storage
I/O	Input/Output
ENC	Encoder: Actual value encoder
EFP	Compact I/O module (PLC I/O module)
ESD	Electrostatic Sensitive Devices
EMC	ElectroMagnetic Compatibility
EN	European standard
ENC	Encoder: Actual value encoder
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory: Erasable, electrically programmable read-only memory
ePS Network Services	Services for Internet-based remote machine maintenance
EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System
ESR	Extended Stop and Retract
ETC	ETC key ">"; softkey bar extension in the same menu

F	
FB	Function Block (PLC)
FC	Function Call: Function Block (PLC)
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: Memory that works without address specification and whose data is read in the same order in which they was stored
FIPO	Fine interpolator
FPU	Floating Point Unit: Floating Point Unit
CRC	Cutter Radius Compensation
FST	Feed Stop: Feedrate stop
FBD	Function Block Diagram (PLC programming method)
FW	Firmware

G	
GC	Global Control (PROFIBUS: Broadcast telegram)
GDIR	Global part program memory
GEO	Geometry, e.g. geometry axis
GIA	Gear Interpolation dAta: Gear interpolation data
GND	Signal Ground
GP	Basic program (PLC)
GS	Gear Stage
GSD	Device master file for describing a PROFIBUS slave

G	
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data: Global user data

H	
HEX	Abbreviation for hexadecimal number
AuxF	Auxiliary function
HLA	Hydraulic linear drive
HMI	Human Machine Interface: SINUMERIK user interface
MSD	Main Spindle Drive
HW	Hardware

I	
IBN	Commissioning
ICA	Interpolatory compensation
IM	Interface Module: Interconnection module
IMR	Interface Module Receive: Interface module for receiving data
IMS	Interface Module Send: Interface module for sending data
INC	Increment: Increment
INI	Initializing Data: Initializing data
IPO	Interpolator
ISA	Industry Standard Architecture
ISO	International Standardization Organization

J	
JOG	Jogging: Setup mode

K	
K_V	Gain factor of control loop
K_p	Proportional gain
$K_Ü$	Transformation ratio
LAD	Ladder Diagram (PLC programming method)

L	
LAI	Logic Machine Axis Image: Logical machine axes image
LAN	Local Area Network
LCD	Liquid Crystal Display: Liquid crystal display
LED	Light Emitting Diode: Light-emitting diode
LF	Line Feed

L	
PMS	Position Measuring System
LR	Position controller
LSB	Least Significant Bit: Least significant bit
LUD	Local User Data: User data (local)

M	
MAC	Media Access Control
MAIN	Main program: Main program (OB1, PLC)
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine Control Panel: Machine control panel
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MDS	Motor Data Set: Motor data set
MSGW	Message Word
MCS	Machine Coordinate System
MM	Motor Module
MPF	Main Program File: Main program (NC)
MCP	Machine control panel

N	
NC	Numerical Control: Numerical control with block preparation, traversing range, etc.
NCU	Numerical Control Unit: NC hardware unit
NRK	Name for the operating system of the NC
IS	Interface Signal
NURBS	Non-Uniform Rational B-Spline
WO	Work Offset
NX	Numerical Extension: Axis expansion board

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OP	Operator Panel: Operating equipment
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	Options: Options
OLP	Optical Link Plug: Fiber optic bus connector
OSI	Open Systems Interconnection: Standard for computer communications

P	
PIQ	Process Image Output
PII	Process Image Input
PC	Personal Computer
PCIN	Name of the SW for data exchange with the control
PCMCIA	Personal Computer Memory Card International Association: Plug-in memory card standardization
PCU	PC Unit: PC box (computer unit)
PG	Programming device
PKE	Parameter identification: Part of a PIV
PIV	Parameter identification: Value (parameterizing part of a PPO)
PLC	Programmable Logic Control: Adaptation control
PN	PROFINET
PNO	PROFIBUS user organization
PO	POWER ON
POU	Program Organization Unit
POS	Position/positioning
POSMO A	Positioning Motor Actuator: Positioning motor
POSMO CA	Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory; AC infeed
POSMO CD	Positioning Motor Compact DC: Like CA but with DC infeed
POSMO SI	Positioning Motor Servo Integrated: Positioning motor, DC infeed
PPO	Parameter Process data Object: Cyclic data telegram for PROFIBUS DP transmission and "Variable speed drives" profile
PPU	Panel Processing Unit (central hardware for a panel-based CNC, e.g SINUMERIK 828D)
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program Test
PSW	Program control word
PTP	Point-To-Point: Point-To-Point
PUD	Program global User Data: Program-global user variables
PZD	Process data: Process data part of a PPO

Q	
QEC	Quadrant Error Compensation

R	
RAM	Random Access Memory: Read/write memory
REF	REFerence point approach function
REPOS	REPOSition function
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid Override: Input correction

R	
RP	R Parameter, arithmetic parameter, predefined user variable
RPA	R Parameter Active: Memory area in the NC for R parameter numbers
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion
RTS	Request To Send: Control signal of serial data interfaces
RTCP	Real Time Control Protocol

S	
SA	Synchronized Action
SBC	Safe Brake Control: Safe Brake Control
SBL	Single Block: Single block
SBR	Subroutine: Subprogram (PLC)
SD	Setting Data
SDB	System Data Block
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test: Block search, program test
SFB	System Function Block
SFC	System Function Call
SGE	Safety-related input
SGA	Safety-related output
SH	Safe standstill
SIM	Single in Line Module
SK	Softkey
SKP	Skip: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper Motor
SMC	Sensor Module Cabinet Mounted
SME	Sensor Module Externally Mounted
SMI	Sensor Module Integrated
SPF	Sub Routine File: Subprogram (NC)
PLC	Programmable Logic Controller
SRAM	Static RAM (non-volatile)
TNRC	Tool Nose Radius Compensation
SRM	Synchronous Rotary Motor
LEC	Leadscrew Error Compensation
SSI	Serial Synchronous Interface: Synchronous serial interface
SSL	Block search
STW	Control word
GWPS	Grinding Wheel Peripheral Speed
SW	Software
SYF	System Files: System files
SYNACT	SYNchronized ACTion: Synchronized Action

T	
TB	Terminal Board (SINAMICS)
TCP	Tool Center Point: Tool tip
TCP/IP	Transport Control Protocol / Internet Protocol
TCU	Thin Client Unit
TEA	Testing Data Active: Identifier for machine data
TIA	Totally Integrated Automation
TM	Terminal Module (SINAMICS)
TO	Tool Offset: Tool offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a lathe
TTL	Transistor-Transistor Logic (interface type)
TZ	Technology cycle

U	
UFR	User Frame: Work offset
SR	Subprogram
USB	Universal Serial Bus
UPS	Uninterruptible Power Supply

V	
VDI	Internal communication interface between NC and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]
VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VI	Voltage Input
VO	Voltage Output
FDD	Feed Drive

W	
SAR	Smooth Approach and Retraction
WCS	Workpiece Coordinate System
T	Tool
TLC	Tool Length Compensation
WOP	Workshop-Oriented Programming
WPD	Workpiece Directory: Workpiece directory
TRC	Tool Radius Compensation
T	Tool
TO	Tool Offset
TM	Tool Management
TC	Tool change

X	
XML	Extensible Markup Language

Z	
WOA	Work Offset Active: Identifier for work offsets
ZSW	Status word (of drive)

Index

7

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

T

U

V

W