SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Special functions

Function Manual

Continued on next page

Valid for

Control
SINUMERIK 840D sl / 840DE sl / 828D

CNC software   Version 4.8 SP3

08/2018
6FC5397-2BP40-6BA2

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: 6FC5397-2BP40-6BA2
Ⓟ 08/2018 Subject to change

SINUMERIK 840D sl / 828D Special functions

Function Manual

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Document order number: 6FC5397-2BP40-6BA2
℗ 08/2018 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.

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)

Note

Signal address

The description of functions include as <signal address> of an NC/PLC interface signal, only the address valid for SINUMERIK 840D sl. The signal address for SINUMERIK 828D should be taken from the data lists "Signals to/from ..." at the end of the particular description of functions.

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	Value range
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

1 Fundamental safety instructions

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 (parameter assignments) 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. Siemens’ products and solutions 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. firewalls and/or network segmentation) are in place.

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

Industrial security (http://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 are available and that 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 (http://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, malware 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.
- Protect the drive against unauthorized changes by activating the "know-how protection" drive function.

F2: Multi-axis transformations 2

2.1 Brief description

2.1.1 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)

2.1.2 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.

2.1.3 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.

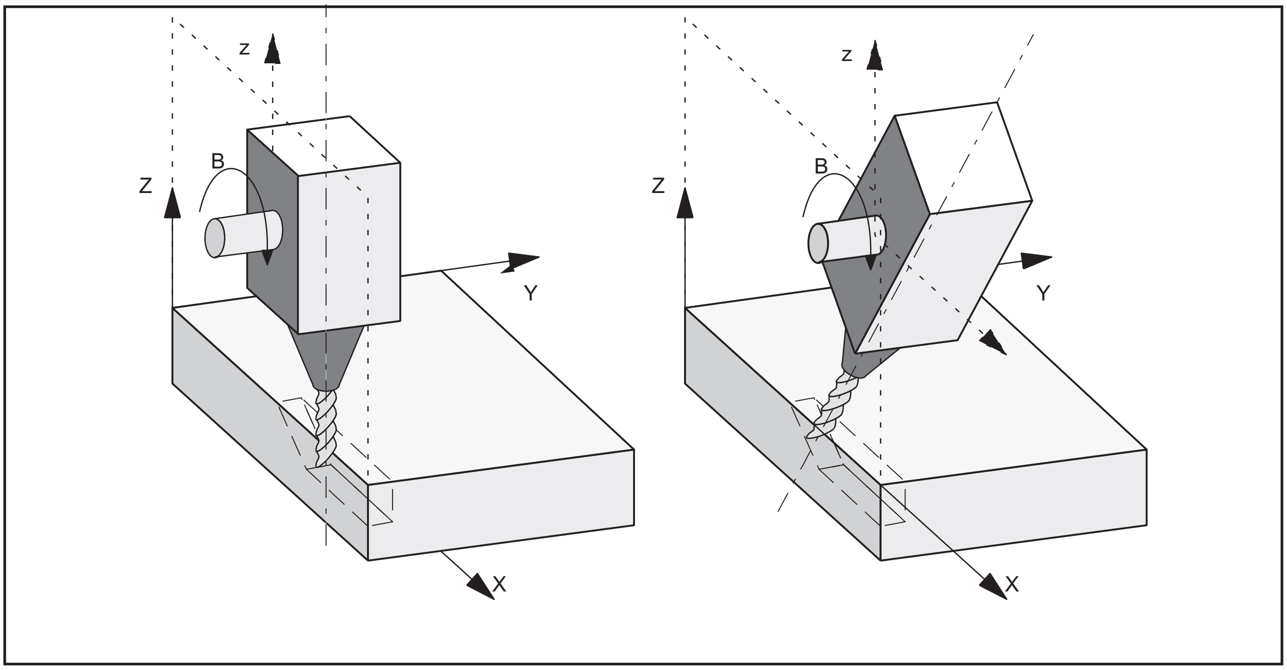


Figure 2-1 Schematic diagram of 3-axis transformation

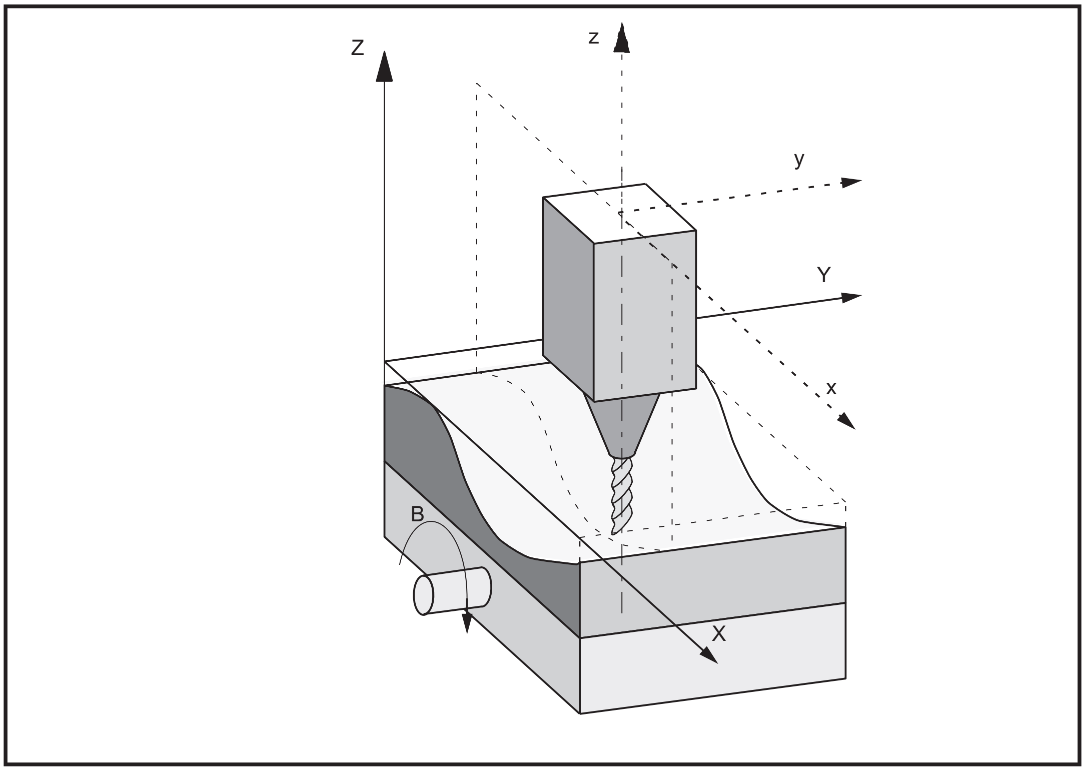


Figure 2-2 Schematic diagram of a 4-axis transformation with moveable workpiece

2.1.4 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.

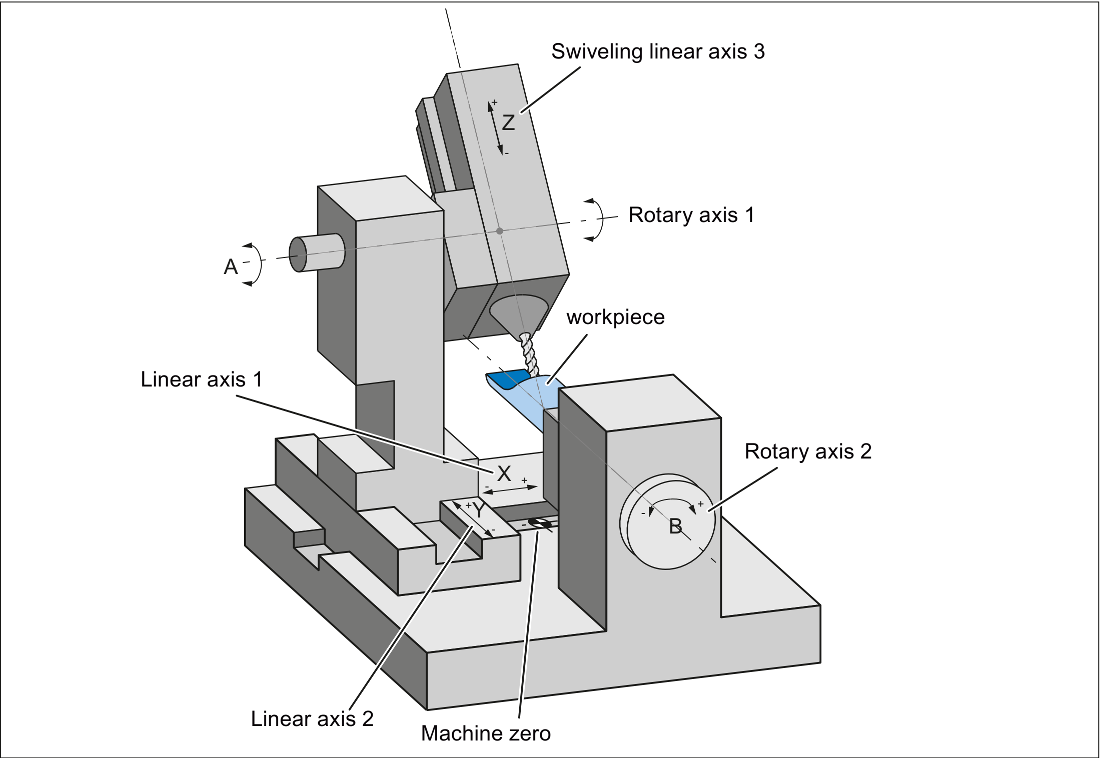


Figure 2-3 Schematic diagram of a machine with swiveling linear axis

2.1.5 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)

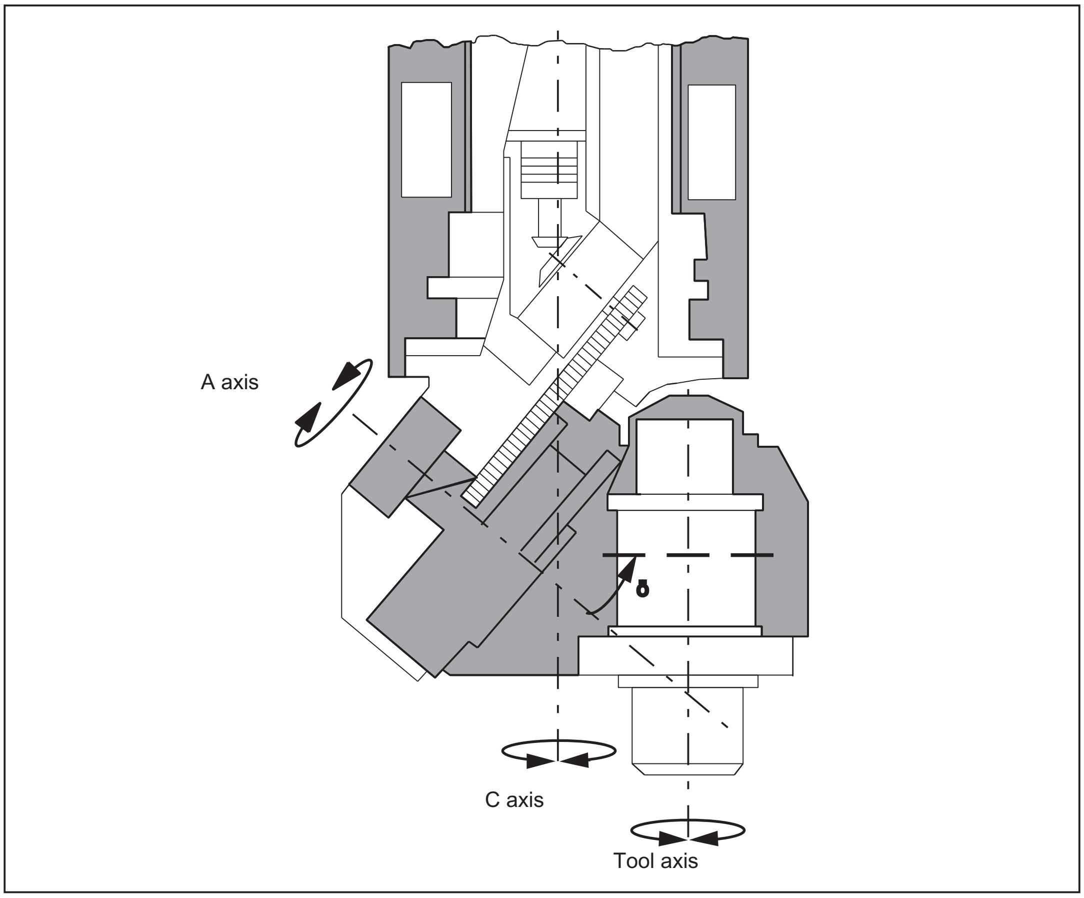


Figure 2-4 Schematic diagram of a machine tool with universal milling head

2.1.6 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 kinematics, 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`

2.1.7 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)

2.1.8 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.

2.1.9 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.

2.1.10 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.

2.1.11 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

2.1.12 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.

2.2 5-axis transformation

2.2.1 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.

2.2.2 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.

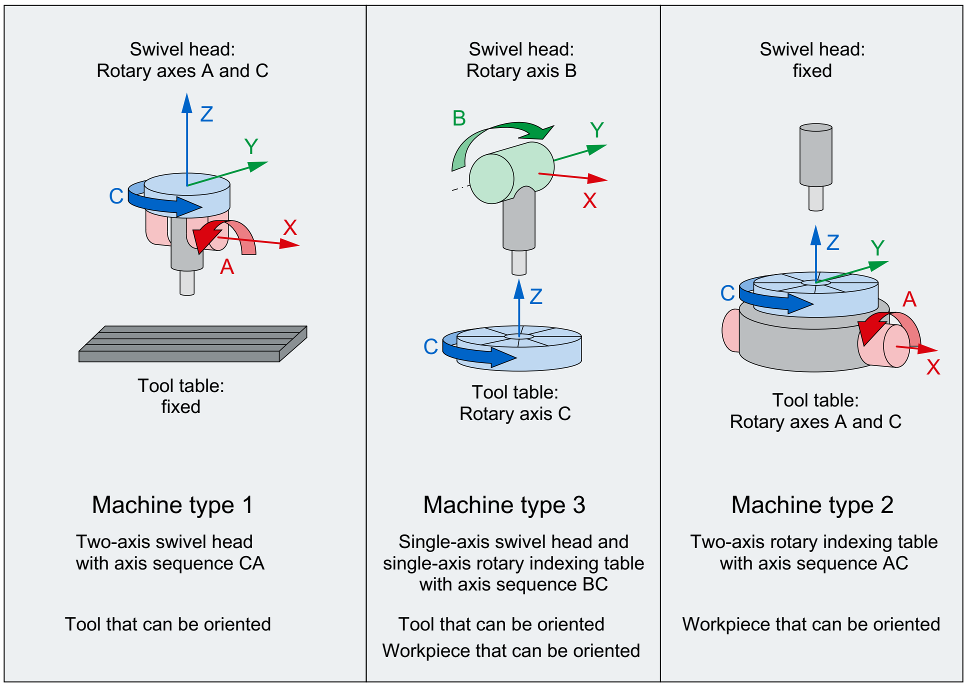


Figure 2-5 Machine types for 5-axis transformation

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

2.2.3 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 from last table swivel joint to zero point of table

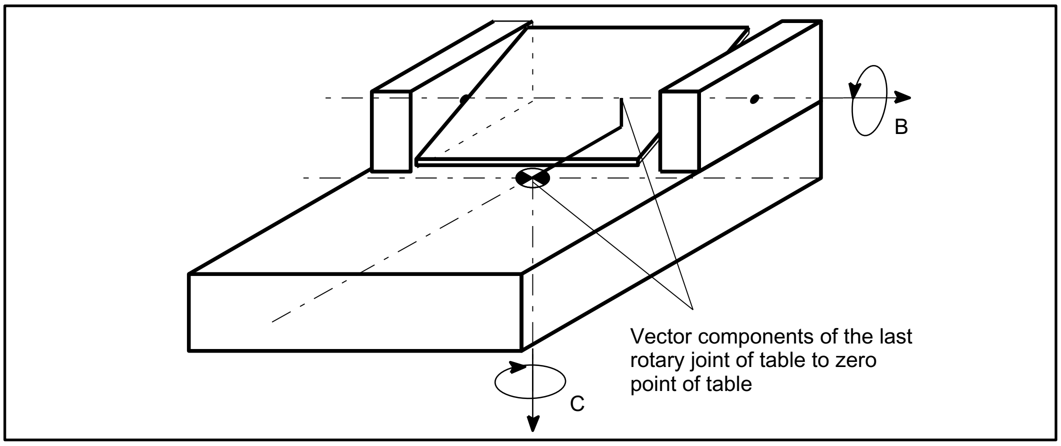
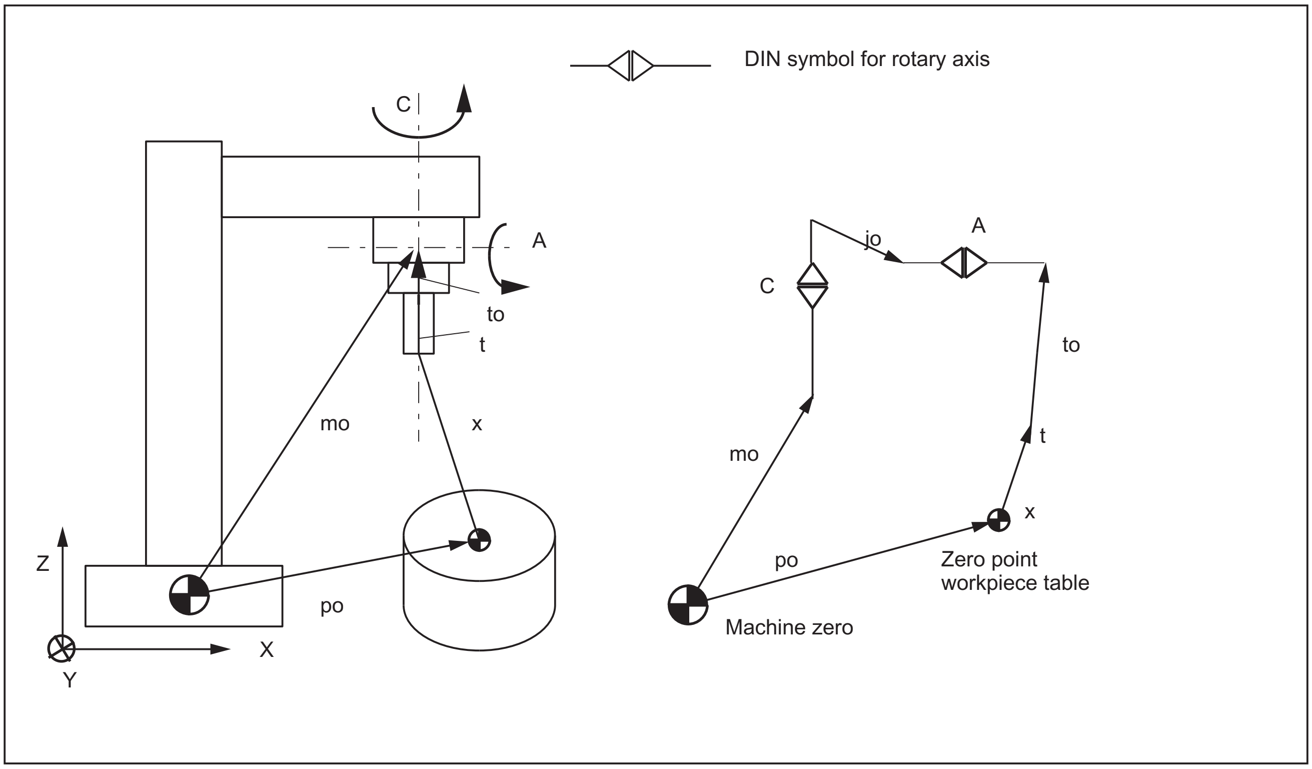

B
C
Vector components of the last rotary joint of table to zero point of table

Figure 2-6 Machine data MD24500 $MC_TRAFO5_PART_OFFSET_1 for machine type 2

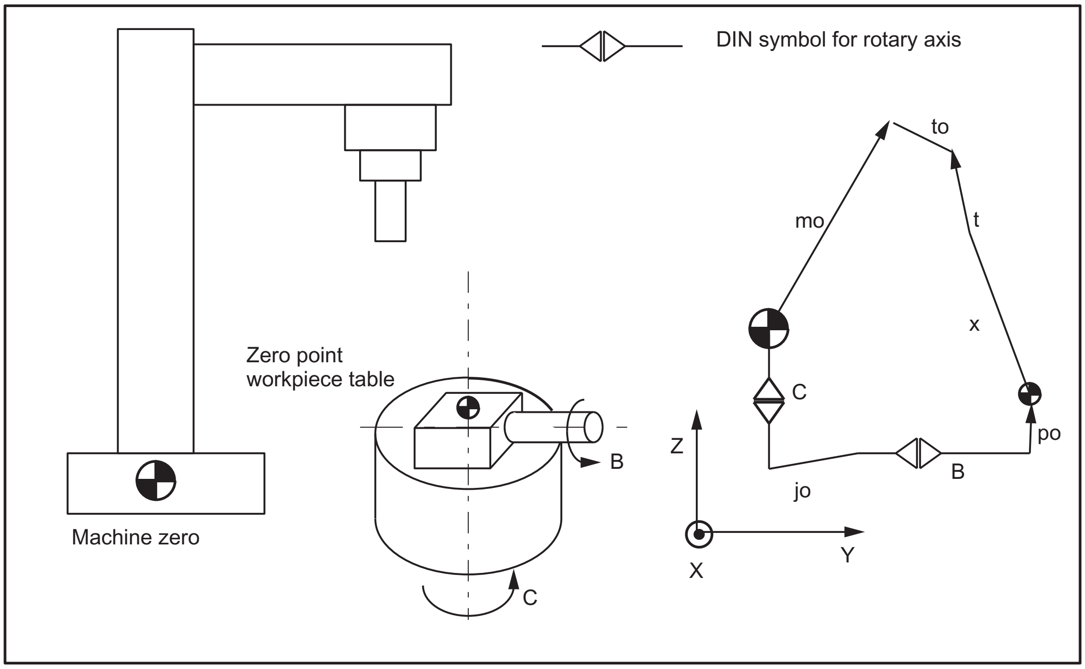
  - Machine type 3 (single-axis swivel head and single-axis rotary table)
    Vector from joint of rotary table to zero point of table
- $MC_TRAFO5_JOINT_OFFSET_<x> (vector of kinematic offset)
  where x = 1, 2, ... maximum number of transformations in the channel
  - Machine type 1 and 2:
    Vector from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



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 2-7 Schematic diagram of CA kinematics, moved tool



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 2-8 Schematic diagram of CB kinematics, moved workpiece

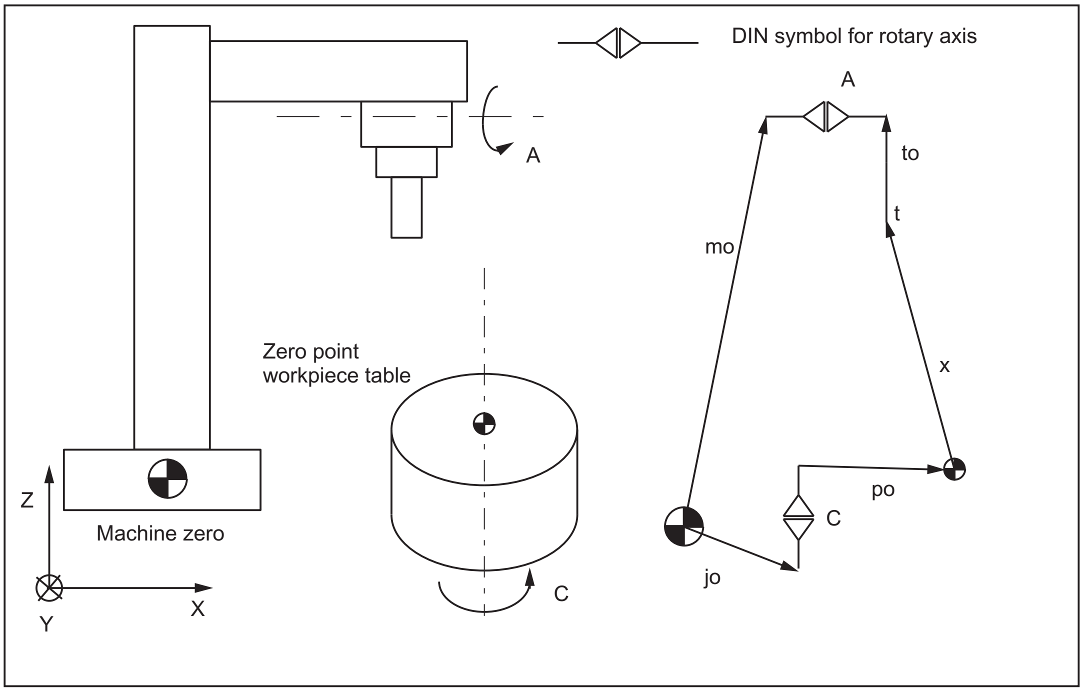


Figure 2-9 Schematic diagram of AC kinematics, moved tool, moved workpiece

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

<Val-ue>	Meaning
TRUE	The sign is not inverted. The traversing direction is as defined in MD32100 $MA_AX_MO-TION_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).

2.2.4 Tool orientation

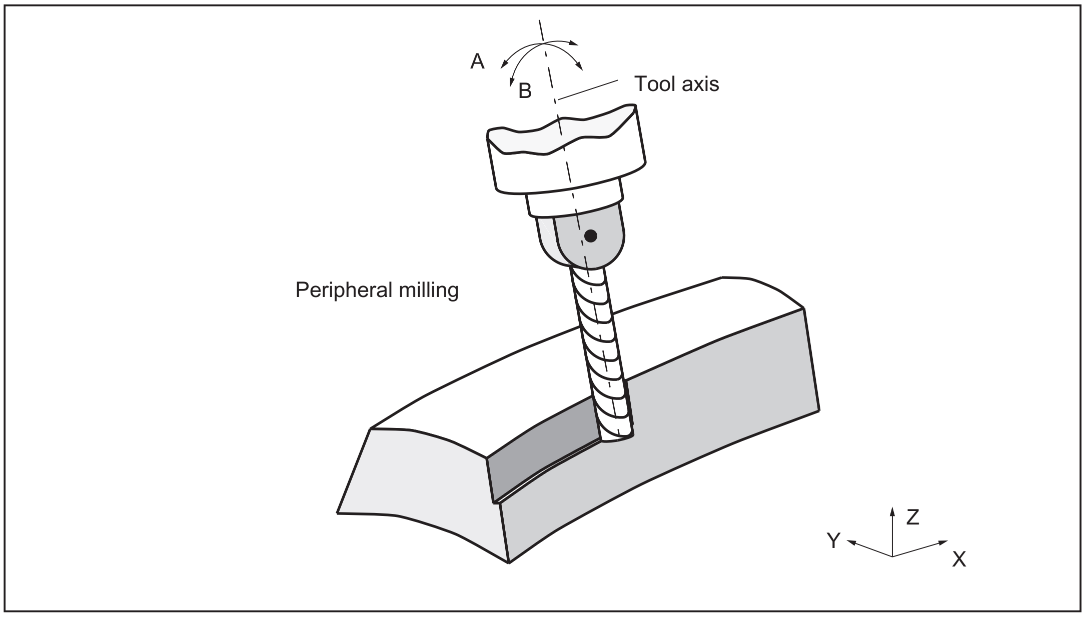


Figure 2-10 Machining of workpieces with 5-axis transformation

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.

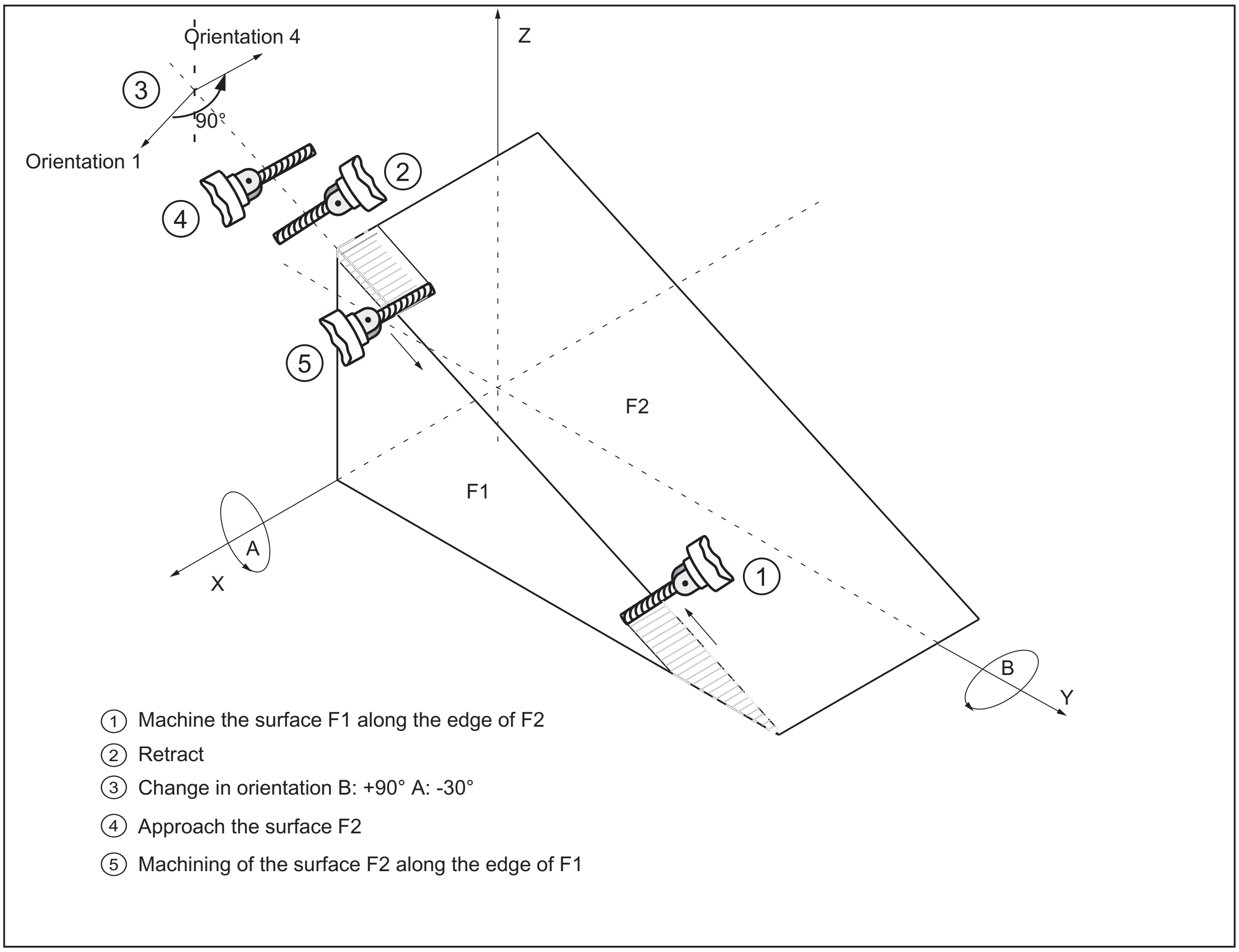

Orientation 4
Z
③
90°
Orientation 1
②
④
⑤
F2
F1
A
X
①
B
Y
① Machine the surface F1 along the edge of F2
② Retract
③ Change in orientation B: +90° A: -30°
④ Approach the surface F2
⑤ Machining of the surface F2 along the edge of F1

Figure 2-11 Change in cutter orientation while machining inclined edges

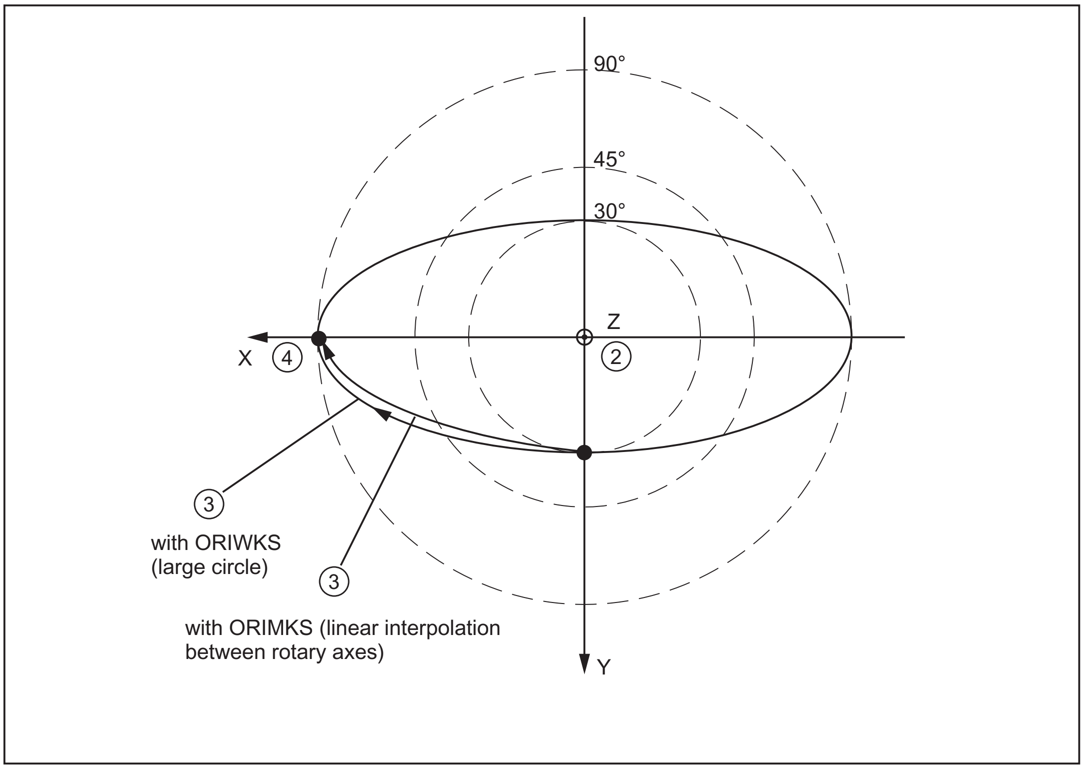


Figure 2-12 Change in orientation while machining inclined edges

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:

Reference:
Function Manual, Basic Machine; Tool Offset; Orientable Toolholders (W1)

2.2.5 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 88)" 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.

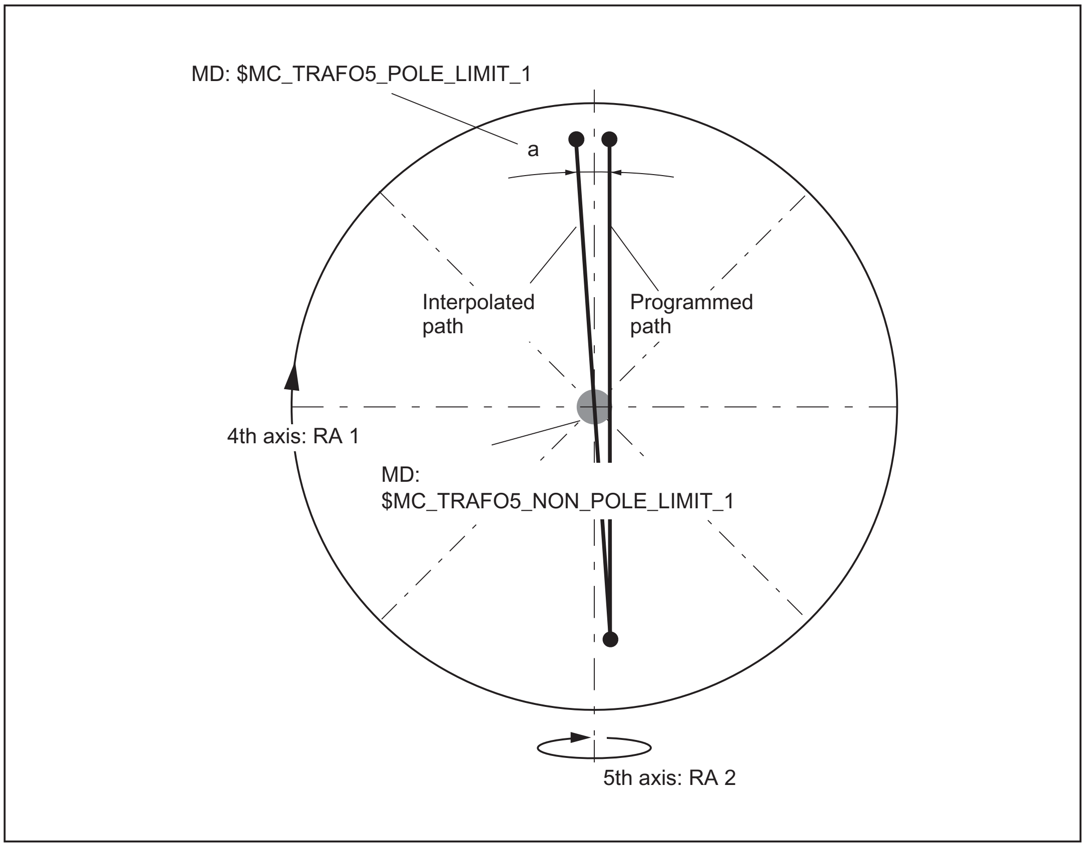


Figure 2-13 5-axis transformation; orientation path in pole vicinity. Example for machine type 1: 2-axis swivel head with rotary axis RA 1 (4th axis of transformation) and rotary axis RA 2 (5th axis of transformation)

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".

2.3 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.

2.4 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 41)" 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"

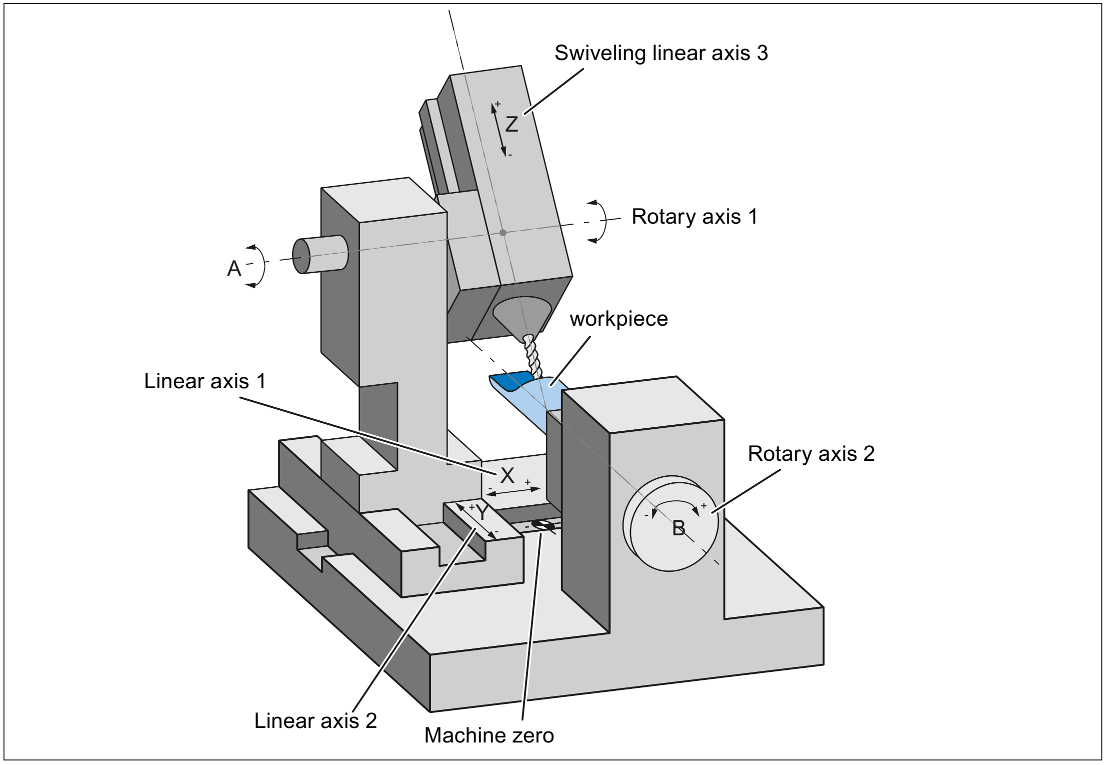


Figure 2-14 Example: Machine with swiveling linear axis Z

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>
1. Rotary axis	2. rotary axis	swiveled linear axis	Bits 6 - 0
A	B	Z	10,00 000
A	C	Y	10,00 001

Kinematics			<type>
1. Rotary axis	2. rotary axis	swiveled linear axis	Bits 6 - 0
B	A	Z	10,00 010
B	C	X	10,00 011
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 toolholding flange aligns with the table zero (*). Is this is technically not possible, vector **to** must be corrected by the deviations.

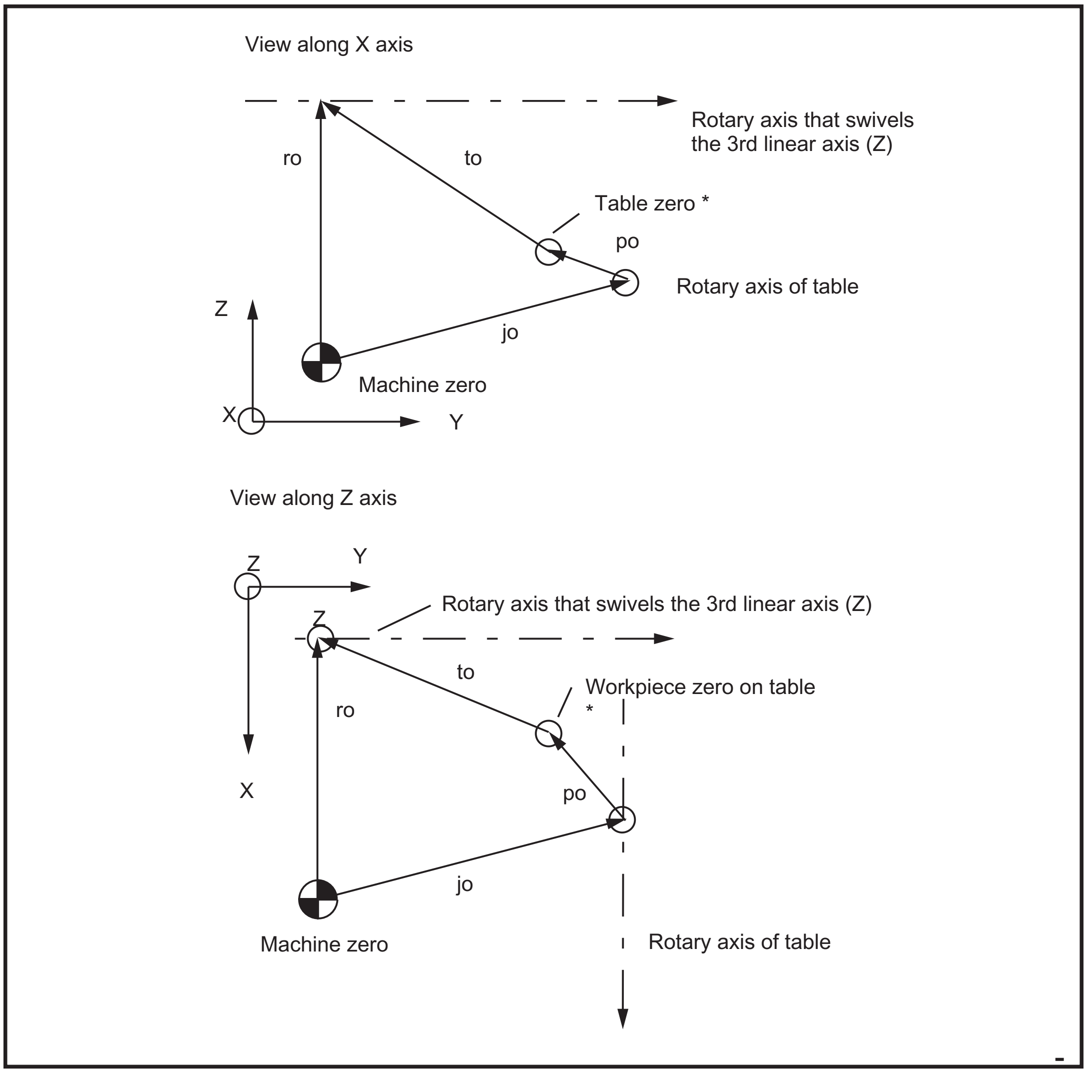


Figure 2-15 Projections of the vectors to be set in the machine data

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.

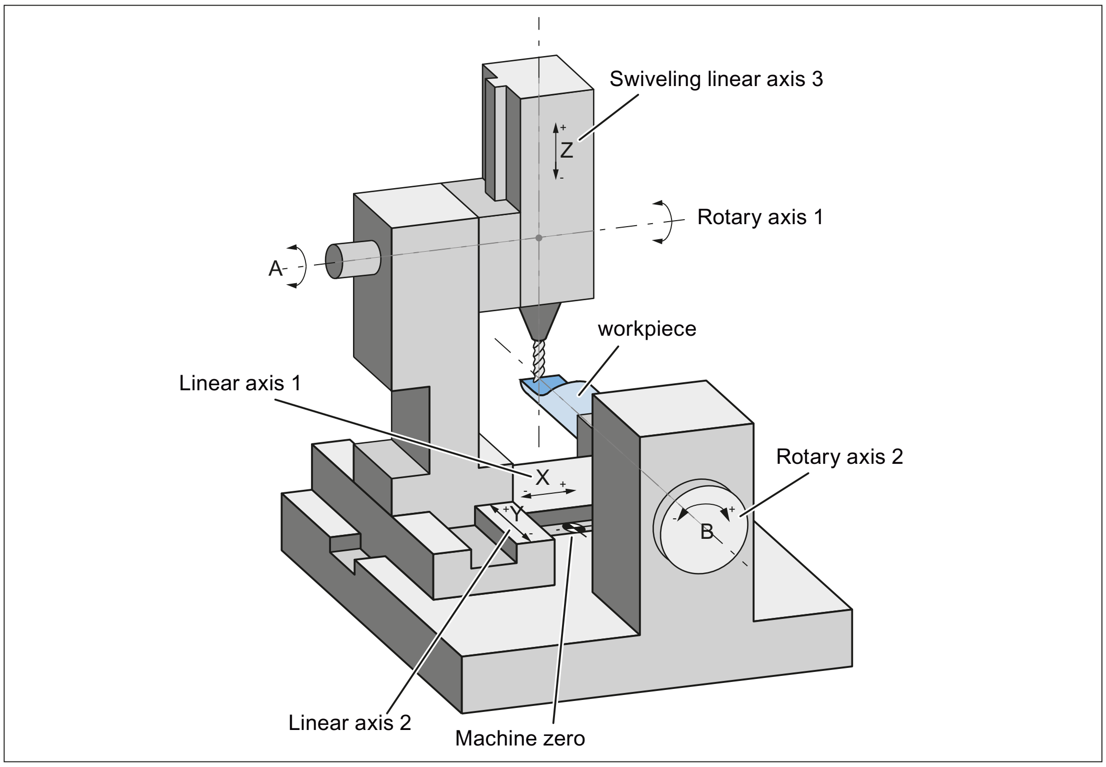


Figure 2-16 Machine in the zero position

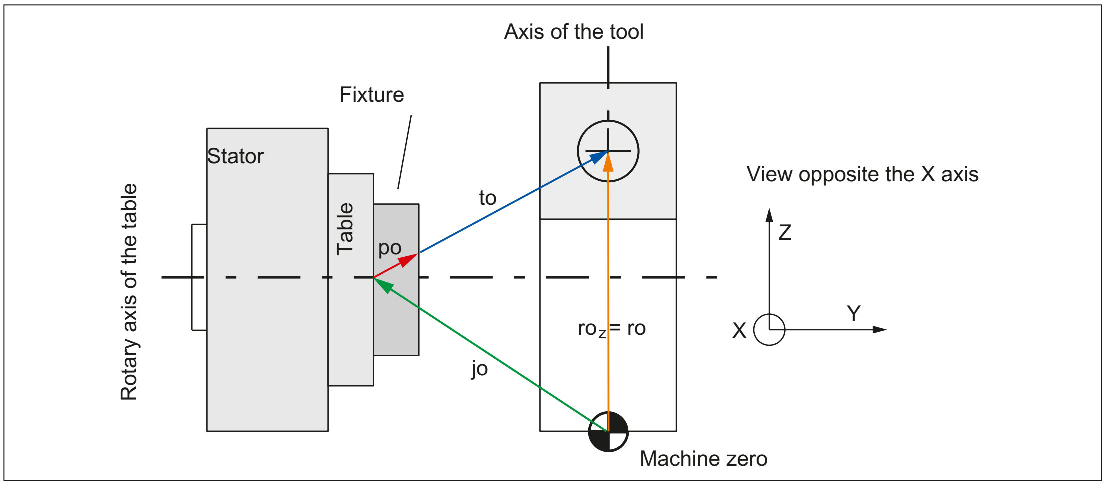


Figure 2-17 Front view: Vectors for machine in the zero position

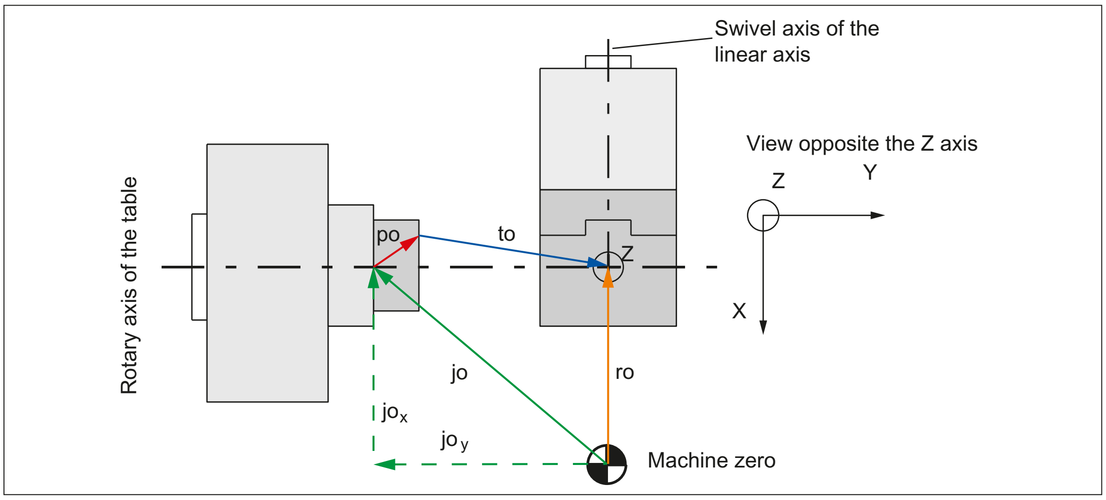


Figure 2-18 Top view: Vectors for machine in the zero position

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 70)".

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 52)").

2.5 Cardan milling head

2.5.1 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 45)". 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.

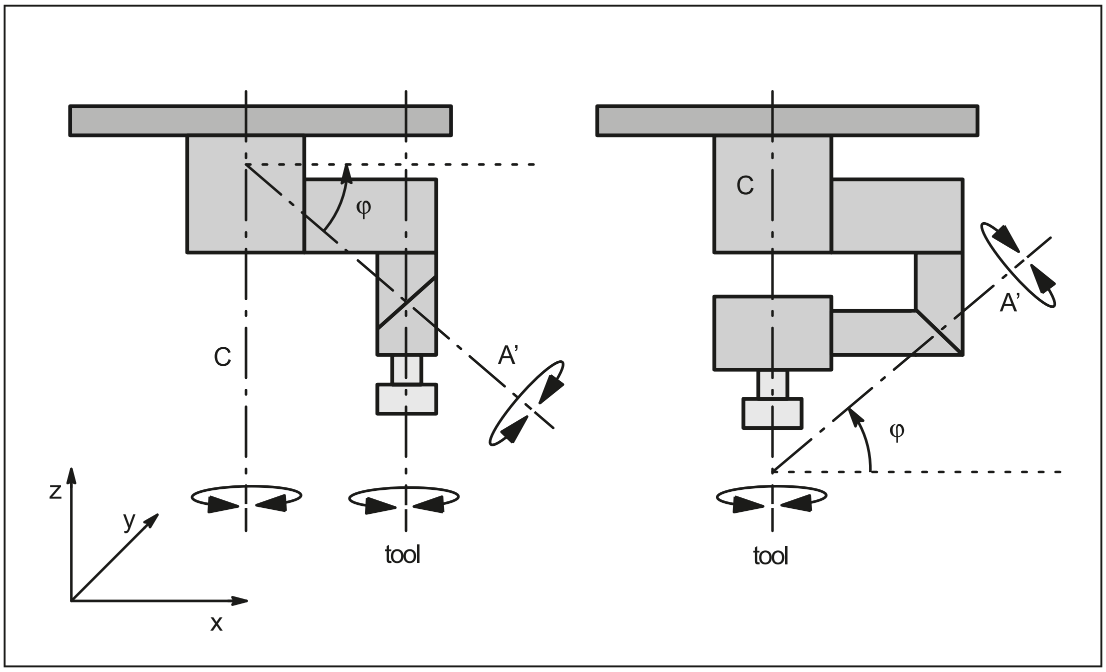


Figure 2-19 Schematic representation of the cardan milling head versions

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

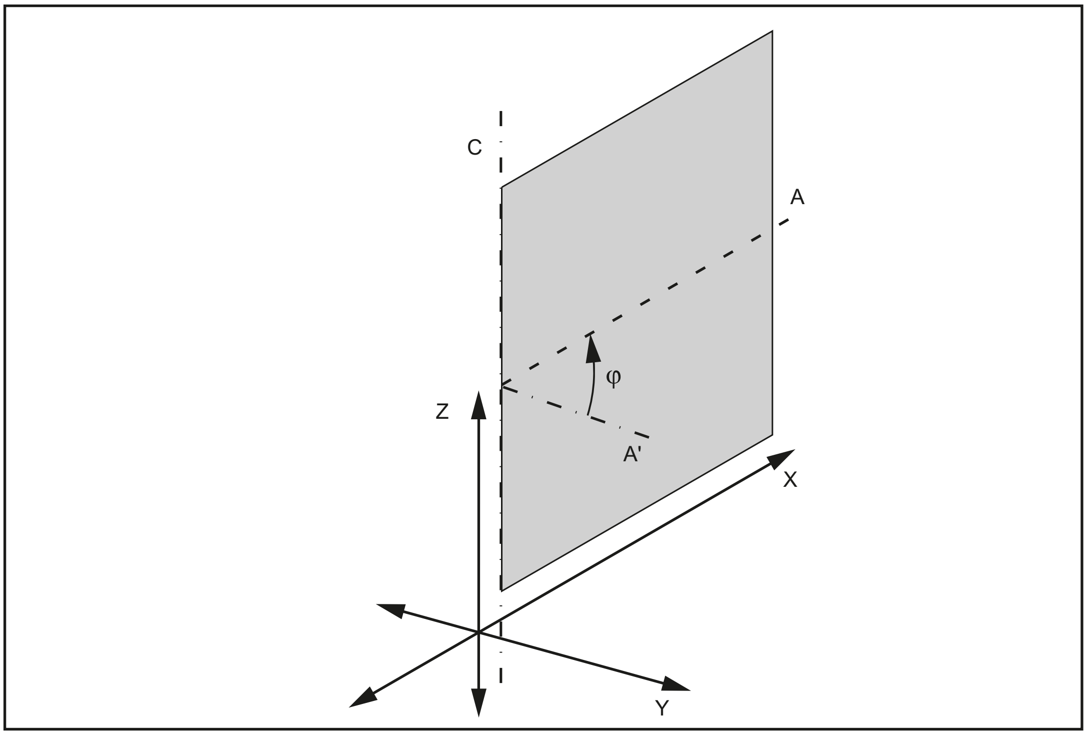


Figure 2-20 Position of axis A'

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.

2.5.2 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.

2.5.3 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.

2.6 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.

References

A detailed description of the machine data can be found in:

Parameter Manual, Detailed Machine Data Description

2.7 Generic 5-axis transformation and variants

2.7.1 Functionality

Scope of functions

The scope of functions of generic 5-axis transformation covers implemented 5-axis transformations (see Section "5-axis transformation (Page 45)") 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 66)").

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`.

2.7.2 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 47)"):

$MC_TRAFO_TYPE_1, ..., _8

Additional types have been introduced for generic 5-axis transformation:

Table 2-1 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

2.7.3 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:

Reference:
Function Manual, Basic Machine; Tool Offset; Orientable Toolholders (W1)

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 first rotary axis) or
    MD24670 $MC_TRAFO5_AXIS1_2[n] (direction first 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.

2.7.4 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 2-2 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

2.7.5 Extension of the generic transformation to six axes - 840D sl only

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 1)	57 2)
Machine type	1	2	3	4
Toolorientation	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, which in the kinematic chain, is located next to the **workpiece**. The **third** rotary axis of the transformation is the rotary axis, which in the kinematic chain, is located next to the **tool**. When the second rotary axis traverses, then the third rotary axis is also moved.				
2) The **first** rotary axis of the transformation is the rotary axis, which in the kinematic chain, is located next to the **workpiece**. The **third** rotary axis of the transformation is the rotary axis, which in the kinematic chain, is located next to the **tool**. 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 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 121)").

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.

References: Function Manual, Basic Machine; tool offsets (W1), Chapter 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 179)

2.7.6 Extension of the generic transformation to 7 axes - 840D sl only

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.

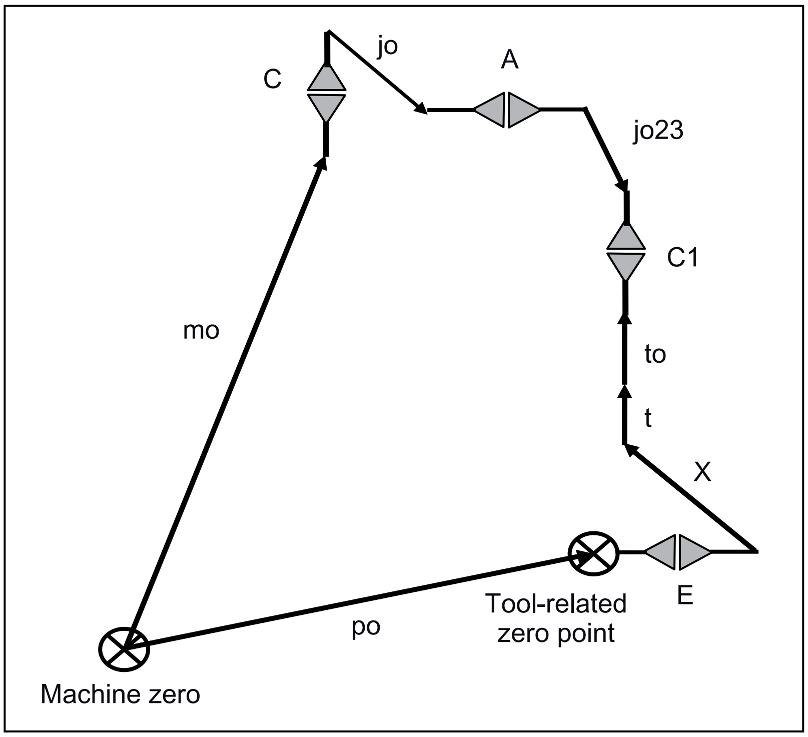
The 7-axis transformation types only cover those kinematics in which the three linear axes form a rectangular Cartesian coordinate system, i.e. no kinematics are covered in which at least one rotary axis lies between two linear axes in the kinematic chain.

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] $MC_TRAFO_AXES_IN_1[6..8]	This machine data is to be assigned with default value of 0. (default setting) 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 2-21 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.

2.7.7 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:
Reference:
Function Manual Extension Functions; Kinematic Transformation (M1)

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 90)" and "Orientation axes (Page 110)").

Note

For further information about the programming of rotations please refer to:
References:
Programming Manual, Job Planning; Chapter: "Transformations"

2.8 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 117)").

2.8.1 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:

References:

Programming Manual, Job Planning, Transformations; Section: 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)

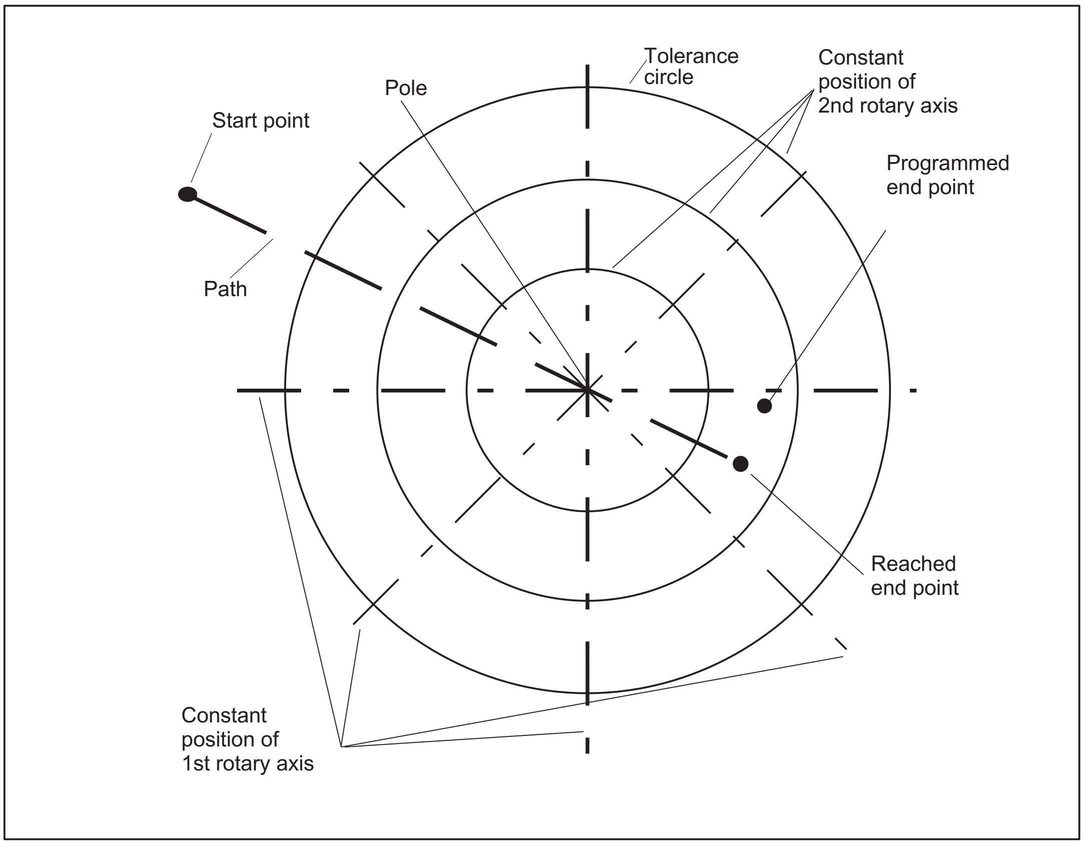

Tolerance circle
Constant position of 2nd rotary axis
Pole
Start point
Programmed end point
Path
Reached end point
Constant position of 1st rotary axis

Figure 2-22 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.

2.9 Orientation

2.9.1 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:

Reference:
Function Manual, Basic Functions; Tool Offset (W1), Chapter: "Orientable tool carrier"

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.

2.9.2 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 "Examples for generic axis transformations (Page 178)".

2.9.3 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 angle 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`	

References

Function Manual, Basic Functions; Chapter "B1: Continuous-path mode, Exact stop, Look Ahead" > "Compressor functions"

2.9.4 Smoothing of the orientation characteristic

2.9.4.1 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.

2.9.4.2 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

2.9.4.3 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.

2.9.5 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

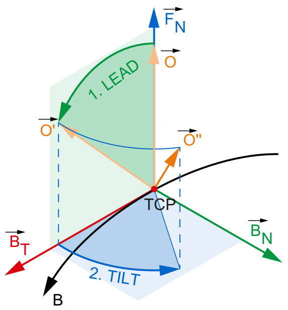
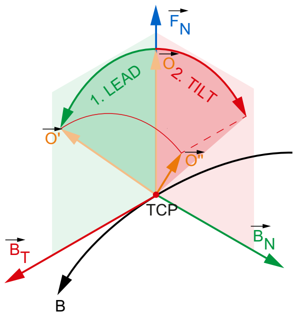
Setting for path relative orientation

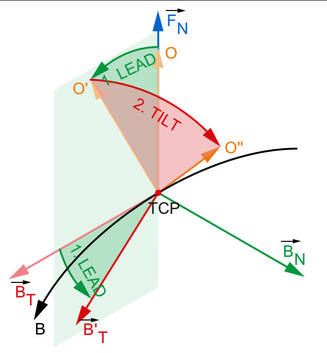
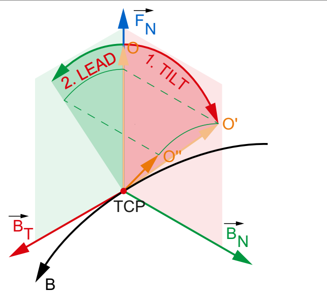
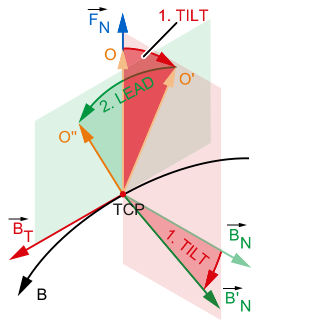
With the machine data, the following decimal-coded functions are parameterized:

- 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>

Value	Meaning	
Units digit (Orientation relative to the path)		
xxx0	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.	
xxx1	In the **complete block**, the tool orientation has the reference to the path tangent and surface normal vector programmed with `LEAD` and `TILT`.	
Tens digit (Interpretation of the **angles of rotation** `LEAD` **and** `TILT`) The subsequently described interpretation of LEAD and TILT angle is valid for the angle programmed for path-relative orientation interpolation `ORIPATH` and `ORIPATHS` with `LEAD=` and `TILT=` as well as also for the offsets of the LEAD and TILT angle, which also for non-path-relative orientation interpolation can be programmed using system variables $P_OFF_LEAD / $P_OFF_TILT or $AC_OFF_LEAD / $AC_OFF_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=…` **Note** 2: When applying a path-relative offset to a **programmed** orientation vector O, then the orientation vector assumes the role of the surface normal vector N.		
xx0x	 $\vec{F}_N$ $\vec{O}$ 1. LEAD $\vec{O'}$ $\vec{O''}$ TCP $\vec{B}_T$ $\vec{B}_N$ 2. TILT B	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''
xx1x	 $\vec{F}_N$ $\vec{O}$ 1. LEAD 2. TILT $\vec{O'}$ $\vec{O''}$ TCP $\vec{B}_T$ $\vec{B}_N$ B	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''

xx2x	 $\vec{F}_N$ O 1. LEAD O' 2. TILT O'' TCP $\vec{B}_N$ $\vec{B}_T$ 1. LEAD B $\vec{B'}_T$	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	 $\vec{F}_N$ 2. LEAD O 1. TILT O' O'' TCP $\vec{B}_T$ $\vec{B}_N$ B	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	 $\vec{F}_N$ 1. TILT O 2. LEAD O' O'' TCP $\vec{B}_T$ $\vec{B}_N$ 1. TILT B $\vec{B'}_N$	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 lift/retraction 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: $0 \leq R < 1$

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 121)").

2.9.6 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:
Reference:
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, a 5th degree polynomial angles.

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.
THETA:	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 six axes - 840D sl only (Page 78)").

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 eexample **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:
References:
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 anglesare 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 iflinear 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:	Oreintation 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.

2.9.7 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

System variable		Meaning	
$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>	
$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

System variable		
$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>].

2.10 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 174)".

2.10.1 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)

2.10.2 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

References:
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 174)".

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 (for description, see Section "Tool orientation (Page 52)")
- 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 swivelled in the plane spanned by the start and end vectors (large circle interpolation). With 6 transformation axes a rotation around the orientation vector is excuted 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).

2.10.3 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 thetool (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:

Reference:
Programming Manual, Production Planning; 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:

Reference:
Function Manual, Basic Machine; Tool Offset (W1)

2.10.4 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.

2.10.5 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

2.11 Orientation vectors

2.11.1 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:

Reference:
Programming Manual; Production Planning

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.

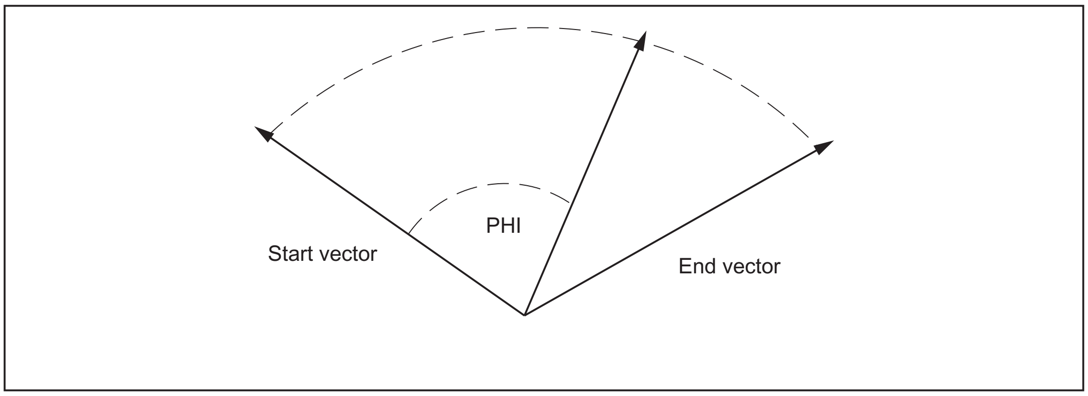


Figure 2-23 Rotation of the orientation vector in the plane between start and end vector

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.

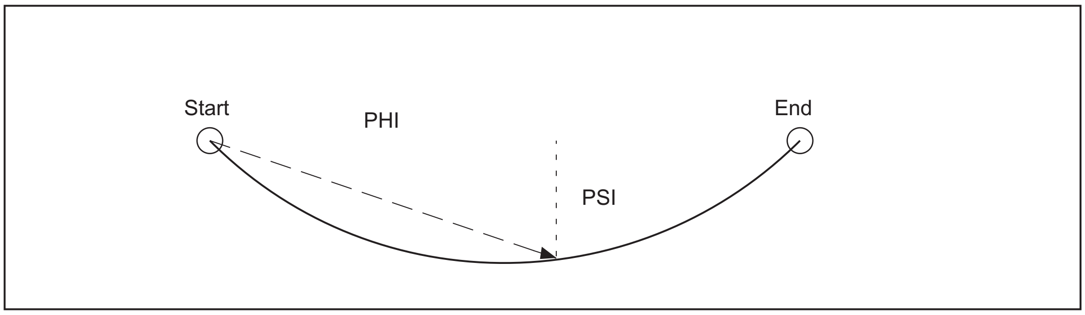


Figure 2-24 Movement of the orientation vector in plan view

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:

Reference:
Programming Manual; Production Planning

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

2.11.2 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:

$$\theta u=\theta_s+d_1u+d_2u^2+d_3u^3+d_4u^4+d_5u^5 \qquad (14)$$

For parameter interval 0 ... 1, this produces the following values for linear coefficients:

$$d_1=\theta_e - \theta_s - d_2 - d_3 - d_4 - d_5 \qquad (15)$$

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).

2.11.3 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.

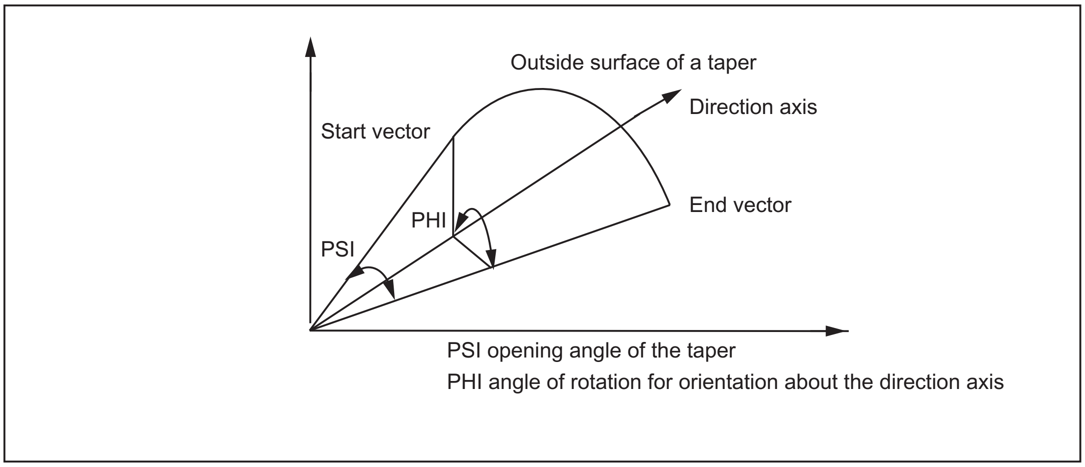

Outside surface of a taper
Direction axis
Start vector
End vector
PHI
PSI
PSI opening angle of the taper
PHI angle of rotation for orientation about the direction axis

Figure 2-25 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 counter-clockwise 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 99)").

Examples

Various changes in orientation are programmed in the following program example:

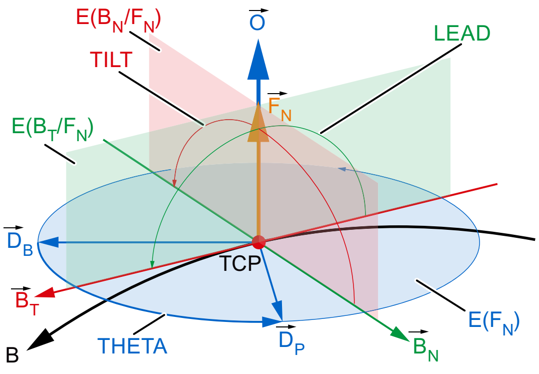
Program code	Comment
`...`	
`N10 G1 X0 Y0 F5000`	
`N20 TRAORI`	`; orientation transformation active.`
`N30 ORIVECT`	`; interpolate tool orientation as vector`
`N40 ORIPLANE`	`; select large circle interpolation.`
`N50 A3=0 B3=0 C3=1`	
`N60 A3=0 B3=1 C3=1`	`; orientation in the Y/Z plane by 45`
	`; degrees 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`
`...`	

2.12 Superimposition of the tool orientation, programmable

2.12.1 Function description

The orientation can be corrected dynamically in small sections that affect the tool orientation by real-time superimposition of the tool orientation, which can be used independently of the machine kinematics.

Fundamentals of the orientation



B	Programmed path
TCP	Tool center point
B_T	Path tangent vector
B_N	Path normal vector
O	Orientation vector
F_N	Surface normal vector $F_N = B_T \times B_N$
D_B	Basic rotation vector
D_P	Programmed rotation vector
$E(B_T/F_N)$	Plane spanned by vectors B_T and F_N
$E(B_N/B_N)$	Plane spanned by vectors B_N and B_N
$E(F_N)$	Plane of rotation vertical to F_N
LEAD	Angle of rotation about the path normal vector B_N
TILT	Angle of rotation about the path tangent vector B_T
THETA	Angle of rotation about the orientation vector O or surface normal vector F_N

Figure 2-26 Fundamentals of orientation (1)

Superimposition options

The following options for superimposing the tool orientation are available:

- Writing the orientation offsets via program variables ($P_...) in an NC program.
- Writing the orientation offsets via system variables ($AC_...) in a synchronized action.
- Activating the parameterizable function "Dynamic offset of the tool orientation" via machine data. Then no additional programming is required in an NC program or a synchronized action.

NC program

In an NC program, a superimposition can be programmed in addition to the current tool orientation. The superimposition then takes effect in every block from the time of its programming and can be changed from block to block as needed. The following types of superimposition are possible:

- Offset vector regarding the **orientation vector**
- Offset vector regarding the **rotation vector**
- Path-relative offset vector
- Rotation of the **orientation vector** about any vector

Synchronized action

In a synchronized action, the following kinds of dynamic superimposition are possible by specifying the corresponding system variables:

- Offset vector regarding the **orientation vector**
- Offset vector regarding the **rotation vector**
- Path-relative offset vector

Supplementary conditions

Superimposing the tool orientation is only possible if an orientation transformation is active.

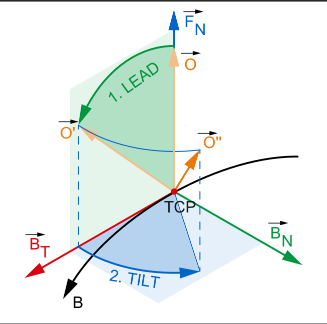
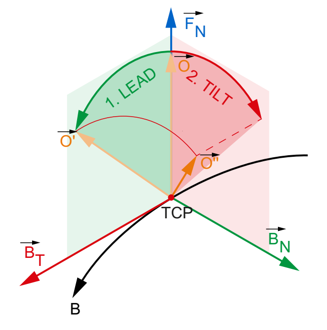
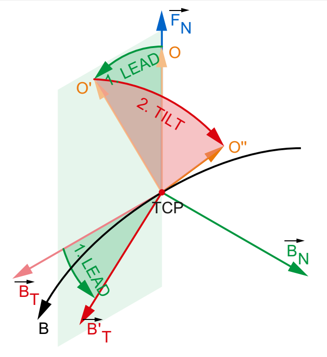
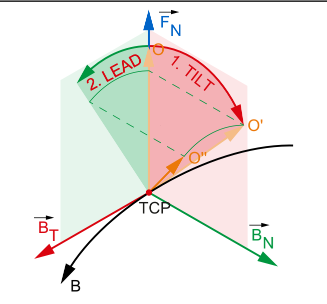
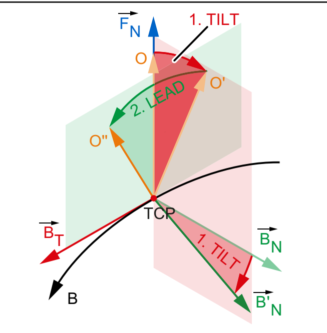
2.12.2 Commissioning: Machine data, channel-specific

2.12.2.1 Execution sequence of the angles of rotation LEAD and TILT

The execution sequence of the angles of rotation **LEAD** and `TILT` is decimally coded using the `decade` of the channel-specific machine data:

MD21094 $MC_ORIPATH_MODE = <Value>

Value	Meaning
The subsequently described interpretation of LEAD and TILT angle is valid for the angle programmed for path-relative orientation interpolation `ORIPATH` and `ORIPATHS` with `LEAD=` and `TILT=` as well as also for the offsets of the LEAD and TILT angle, which also for non-path-relative orientation interpolation can be programmed using system variables $P_OFF_LEAD / $P_OFF_TILT or $AC_OFF_LEAD / $AC_OFF_TILT. For path-relative orientation, the coordinate system is formed by the vectors path tangent T and programmed surface normal vector N. **Note** • Programming the surface normal vector N – at the block **start**: `A4=… B4=… C4=…` – at the block **end**: `A5=… B5=… C5=…` • When applying a path-relative offset to a **programmed** orientation vector O, then the orientation vector assumes the role of the surface normal vector N.	

xx0x		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''
xx1x		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''
xx2x		1st rotation `LEAD`: Rotation of the orientation vector O and the path tangent vector B_T around the path normal vector $B_N \Rightarrow$ O' and B'_T
		2nd rotation `TILT`: Rotation of the new orientation vector O' around the new path tangent vector $B'_T \Rightarrow$ O''
xx3x		1st rotation `TILT`: Rotation of the new orientation vector O around the path tangent vector $B_T \Rightarrow$ O'
		2nd rotation `LEAD`: Rotation of the new orientation vector O' around the path normal vector $B_N \Rightarrow$ O''
xx4x		1st rotation `TILT`: Rotation of the orientation vector O and the path normal vector B_N around the path tangent vector $B_T \Rightarrow$ O' and B'_N
		2nd rotation `LEAD`: Rotation of the new orientation vector O' around the new path normal vector $B'_N \Rightarrow$ O''

2.12.2.2 Absolute or incremental orientation superimposition

Various settings regarding the preprocessing ($S_...) and main run variables ($AC_...) of the orientation superimposition are made using the channel-specific machine data:

MD21096 $MC_OFF_ORI_MODE, <bit> = <value>

Bit	Value	Meaning
Settings regarding the main run variables (AC_...)		
0	Behavior during a channel reset	
	0	The offset is deselected during a RESET
	1	The offset is **not** deselected during a RESET
1	Behavior in JOG mode	
	0	No superimposition of the tool orientation due to the system variables for the offset of the tool orientation.
	1	An overlaid movement due to the system variables for offset of the tool orientation is interpolated.
2	0	$AC_OFF_O: Absolute
	1	$AC_OFF_O: Incremental (integrator)
3	0	$AC_OFF_R: Absolute
	1	$AC_OFF_R: Incremental (integrator)
4	0	$AC_OFF_LEAD: Absolute
	1	$AC_OFF_LEAD: Incremental (integrator)
5	0	$AC_OFF_TILT: Absolute
	1	$AC_OFF_TILT: Incremental (integrator)
6	0	$AC_OFF_THETA: Absolute
	1	$AC_OFF_THETA: Incremental (integrator)
7	0	$AC_OFF_O_ANGLE: Absolute
	1	$AC_OFF_O_ANGLE: Incremental (integrator)
8	0	$AC_OFF_R_ANGLE: Absolute
	1	$AC_OFF_R_ANGLE: Incremental (integrator)
9 1)	0	Alarm 20301 "Superimposed orientation not possible" is displayed
	1	Alarm 20301 "Superimposed orientation not possible" is **not** displayed.
Settings regarding the preprocessing variables ($S_...)		
16	0	$P_OFF_O: Absolute
	1	$P_OFF_O: Incremental (integrator)
17	0	$P_OFF_R: Absolute
	1	$P_OFF_R: Incremental (integrator)
18	0	$P_OFF_LEAD: Absolute
	1	$P_OFF_LEAD: Incremental (integrator)
19	0	$P_OFF_TILT: Absolute
	1	$P_OFF_TILT: Incremental (integrator)
20	0	$P_OFF_THETA: Absolute
	1	$P_OFF_THETA: Incremental (integrator)
21	0	$P_OFF_O_ANGLE: Absolute
	1	$P_OFF_O_ANGLE: Incremental (integrator)

Bit	Value	Meaning
22	0	$P_OFF_R_ANGLE: Absolute
	1	$P_OFF_R_ANGLE: Incremental (integrator)
1) If an orientation, which cannot be accepted by the current kinematics or the machine, would come into being as a result of the superimposition of the orientation, the alarm 20301 is triggered. The responses are: • The superimposition of the orientation is not carried out. • The display of alarm 20301 depends on the parameterization. • This does not interrupt processing.		

Incremental (integrator)

The system variable contains the sum of all written superimposition values.

Reading the system variables

A system variable provides different values depending on the parameterization:

- Absolute: The system variable provides the last written value
- Incremental: The system variable provides the sum of all previously written values

2.12.3 Commissioning: Setting data, channel-specific

2.12.3.1 Angle limitation of the orientation superimposition

With the channel-specific setting data, the maximum possible angle between the current orientation vector O or rotation vector D and the new orientation vector O' or rotation vector D' that results from the orientation superimposition is limited:

SD42664 $SC_OFF_ORI_LIMIT[<index>] = <value>

Index	Meaning
0	Maximum angle between the current orientation vector O and the new orientation vector O' resulting from the orientation superimposition
1	Maximum angle between the current rotation vector D and the new rotation vector D' resulting from the orientation superimposition

If the specified maximum value is exceeded due to an orientation superimposition, the change of the new orientation vector O' or rotation vector D' on the possible maximum value is limited.

The active limiting is displayed for the system variable $AC_OFF_ORI_LIMIT (Page 152):

2.12.4 Programming: Preprocessing variable ($P_OFF_...)

2.12.4.1 Overview

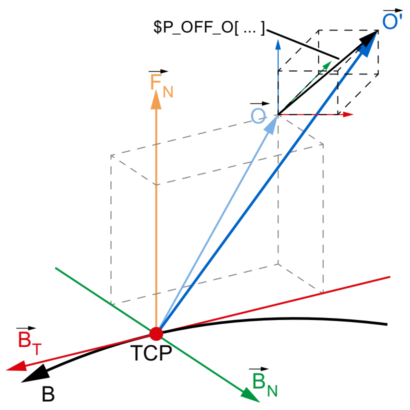
The following system variables of the orientation superimposition are preprocessing variables and can therefore only be read and written via NC programs:

Name	Meaning
$P_OFF_O (Page 136) 2)	Offset vector of the orientation
$P_OFF_R (Page 139) 1) 2)	Offset vector of the rotation vector of the orientation
$P_OFF_LEAD (Page 142) 3)	Offset value of the LEAD angle
$P_OFF_TILT (Page 142) 3)	Offset value of the TILT angle
$P_OFF_THETA (Page 142) 1) 3)	Offset value of the THETA angle (rotation angle about the orientation vector)
$P_OFF_O_DIR (Page 145) 2)	Direction vector about which the orientation vector is rotated with the rotation angle $P_OFF_O_ANGLE
$P_OFF_O_ANGLE (Page 145)	Rotation angle about which the orientation vector is rotated by the direction vector $P_OFF_O_DIR
$P_OFF_R_DIR (Page 148) 2)	Direction vector about which the rotation vector is rotated with the rotation angle $P_OFF_R_ANGLE 1)
$P_OFF_R_ANGLE (Page 148)	Rotation angle about which the rotation vector is rotated by the direction vector $P_OFF_R_DIR 1)
1) Only relevant for 6-axis kinematics	
2) The superimposition of the orientation with the angles LEAD, TILT and THETA is always relative to the current path tangent and is therefore **path-relative**.	
3) The superimposition of the orientation by means of a direction vector is relative to the workpiece coordinate system (WCS).	

The system variables are described in detail in the following sections.

2.12.4.2 Offset vector: $P_OFF_O

Function



Offset vector $P_OFF_O

With the system variable $P_OFF_O[...], an offset vector is specified in the workpiece coordinate system (WCS) in any spatial directions. The new orientation vector O' results from the addition of the offset vector to the current orientation vector O.

O	Orientation vector before vector addition
O'	Orientation vector after vector addition
F_N	Surface normal vector $F_N = B_T \times B_N$

TCP	Tool center point
B	Programmed path
B_T	Path tangent vector
B_N	Path normal vector

Syntax

```
$P_OFF_O[<i>] = <value>
```

Meaning

`$P_OFF_O`:	Offset vector	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	1: X coordinate (abscissa) 2: Y coordinate (ordinate) 3: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL

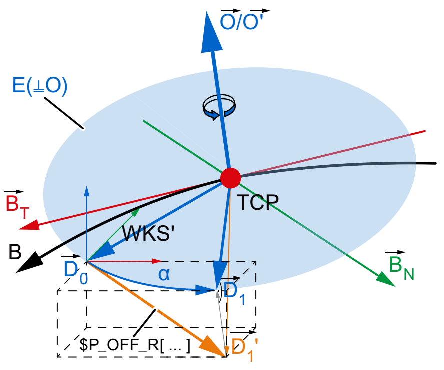
Example

The orientation vector is moved by the offset vector (1.0 ; 1.0; 1.0):

```
Program code                              Comment
                                          ; Offset vector:
N100 $P_OFF_O[1] = 1.0                    ; X coordinate = 1.0
N110 $P_OFF_O[2] = 1.0                    ; Y coordinate = 1.0
N120 $P_OFF_O[3] = 1.0                    ; Z coordinate = 1.0
```

2.12.4.3 Offset vector: $P_OFF_R

Function



Offset vector $P_OFF_R

For the rotation of the orientation vector, an offset vector is specified in the workpiece coordinate system (WCS) in any spatial directions via the system variable $P_OFF_R[...].

The resulting rotation vector D_1' results from the vector addition of the offset vector to the current rotation vector D_0. The new rotation vector D_1 results from the projecting of the resulting rotation vector D_1' into the plane of rotation.

The orientation vector O is rotated in direction D_1 about the angle between D_0 and D_1:⇒ O'

O	Orientation vector before vector addition
O'	Orientation vector after vector addition
F_N	Surface normal vector $F_N = B_T \times B_N$

TCP	Tool center point
B	Programmed path
B_T	Path tangent vector
B_N	Path normal vector
D_0	Current rotation vector
D_1'	Resulting rotation vector
D_1	The new rotation vector is the projection of the resulting rotation vector D_1' into the plane of rotation E(⊥O)

Syntax

```
$P_OFF_R[<i>] = <value>
```

Meaning

`$P_OFF_R`:	Offset vector	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	1: X coordinate (abscissa) 2: Y coordinate (ordinate) 3: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL

Example

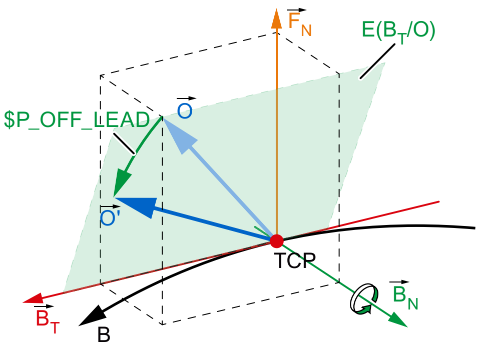
The offset vector (25.0 ; -10.0; -10.0) is added to the current rotation vector D_0:

Program code	Comment
	`; Offset vector:`
`N100 $P_OFF_R[1] =  25.0`	`; X coordinate = 25.0`
`N110 $P_OFF_R[2] = -10.0`	`; Y coordinate = -10.0`
`N120 $P_OFF_R[3] = -10.0`	`; Z coordinate = -10.0`

2.12.4.4 Offset angle: $P_OFF_LEAD, $P_OFF_TILT and $P_OFF_THETA

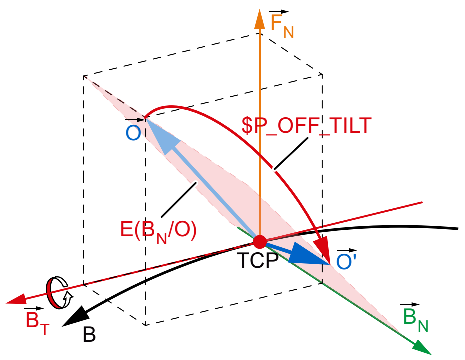
Function

With the system variables $P_OFF_LEAD, $P_OFF_TILT and $P_OFF_THETA, a **path-relative** superimposition of the orientation is specified in a coordinate system that is spanned by the **current orientation vector** and the **path tangent**:



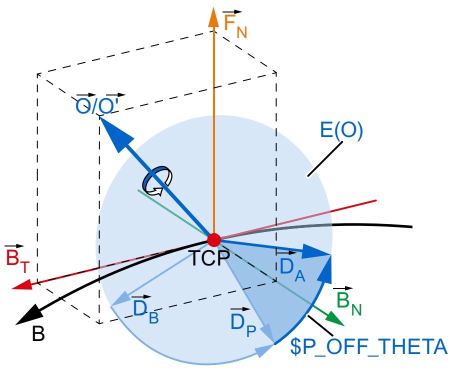
LEAD angle offset $P_OFF_LEAD

The LEAD angle offset causes a rotation of the orientation vector O about the path normal vector B_N in the plane $E(B_T/O)$ spanned by the current orientation vector and the path tangent vector $B_T \Rightarrow O'$



TILT angle offset $P_OFF_TILT

The TILT angle offset causes a rotation of the orientation vector O about the path tangent vector in the plane $E(B_N/O)$ spanned by the current orientation vector and the path normal vector $B_N \Rightarrow O'$



THETA angle offset $P_OFF_THETA

The THETA angle offset causes a rotation of the rotation vector in the plane E(O) vertical to the current orientation vector $O \Rightarrow O'$

F_N	Surface normal vector $F_N = B_T \times B_N$
TCP	Tool center point
B	Programmed path

B_T	Path tangent vector
B_N	Path normal vector
D_B	Basic rotation vector
D_P	Programmed rotation vector
D_A	Current rotation vector

Note

Undefined coordinate system

If no geometry axes are traversed, i.e. no tangent direction is defined, or the orientation vector is aligned parallel to the path tangent, the coordinate system to which the offset angles refer is undefined. The programmed offset angles are then no longer in effect.

Interpretation of the angle of rotation LEAD and TILT

With the machine data MD21094 $MC_ORIPATH_MODE (Page 99), the interpretation of the rotation angles LEAD and TILT are set.

Absolute or incremental type of action

The machine data MD21096 $MC_OFF_ORI_MODE (Page 133) is used to set whether the superimposition of the orientation is absolute or incremental.

Syntax

```
$P_OFF_LEAD  = <value>
$P_OFF_TILT = <value>
$P_OFF_THETA = <value>
```

Meaning

`$P_OFF_LEAD`, `$P_OFF_TILT`, `$P_OFF_THETA` :	Offset value of the LEAD, TILT and THETA angles	
	Data type:	REAL
	Value range:	± 90.0°
	Default value:	0.0
`<value>`:	Angle	
	Data type:	REAL

Example

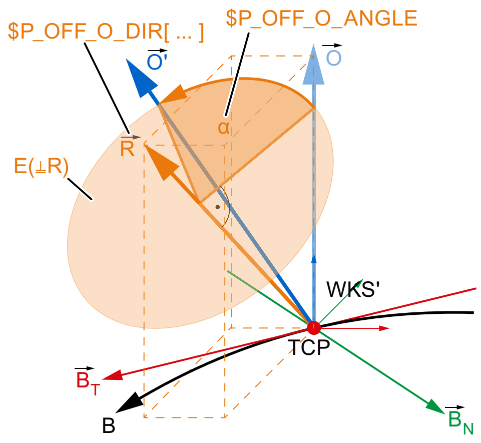
The following angle offsets are programmed:

- LEAD angle = 10.0°
- TILT angle = 5.0°
- THETA angle = 45.0°

Program code	Comment
`N100 $P_OFF_LEAD  = 10.0`	`; LEAD angle = 10.0°`
`N120 $P_OFF_TILT  =  5.0`	`; TILT angle =  5.0°`
`N130 $P_OFF_THETA = 45.0`	`; THETA angle = 45.0°`

2.12.4.5 Direction vector and angle: $P_OFF_O_DIR and $P_OFF_O_ANGLE

Function



With the system variables $P_OFF_O_DIR[...], a direction vector R is specified in the workpiece coordinate system (WCS) in any spatial directions.

The orientation vector O is rotated about this direction vector with the angle α specified with the system variable $P_OFF_O_ANGLE ⇒ O'

The angle α lies in a plane vertical to the direction vector R.

O	Orientation vector before the superimposition
O'	Orientation vector after the superimposition
E(⊥R)	Plane vertical to the direction vector R

TCP	Tool center point
B	Programmed path
B_T	Path tangent vector
B_N	Path normal vector
WCS	Workpiece coordinate system

Syntax

```
$P_OFF_O_DIR[<i>] = <value>
$P_OFF_O_ANGLE = <value>
```

Meaning

`$P_OFF_O_DIR`:	Direction vector for rotating the orientation	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	1: X coordinate (abscissa) 2: Y coordinate (ordinate) 3: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL

`$P_OFF_O_ANGLE`:	Rotation angle for rotating the orientation	
	Data type:	REAL
	Range of values:	± 90.0°
	Default value:	0.0
`<value>`:	Angle	
	Data type:	REAL

Example

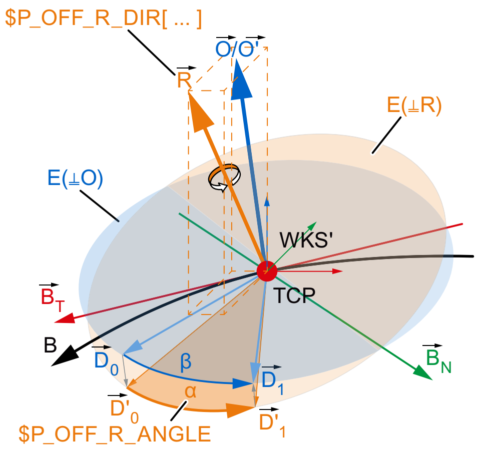
The orientation vector is rotated 45.0° about the direction vector (1.0 ; 1.0; 1.0):

```
Program code                        Comment
                                    ; rotation vector:
N100 $P_OFF_O_DIR[1] = 1.0          ; X coordinate = 1.0
N110 $P_OFF_O_DIR[2] = 1.0          ; Y coordinate = 1.0
N120 $P_OFF_O_DIR[3] = 1.0          ; Z coordinate = 1.0
N130 $P_OFF_O_ANGLE = 45.0          ; angle of rotation = 45.0°
```

2.12.4.6 Direction vector and angle: $P_OFF_R_DIR and $P_OFF_R_ANGLE

Function

With the system variable $P_OFF_R_DIR[...], a direction vector R is specified in the workpiece coordinate system (WCS) in any spatial directions. The rotation vector D_0 is rotated about this direction vector with the angle α specified with the system variable $P_OFF_R_ANGLE ⇒ D_1



For this purpose, the rotation vector D_0 is projected from the orientation plane E(⊥O) into the rotation plane E(⊥R): D_0 ⇒ D_0'

After this, the rotation vector D_0' is rotated about the angle α in the plane of rotation: D_0' ⇒ D_1'

Then the new rotation vector D_1' is projected back into the orientation plane: D_1' ⇒ D_1

The resulting angle of rotation β results from the angular difference of rotation vectors D_0 and D_1

The tool is then rotated about the orientation vector O with the rotation angle β.

O / O'	Orientation vector before and after the rotation
E(⊥O)	Plane vertical to the orientation vector O (orientation plane)
E(⊥R)	Plane vertical to the direction vector R (plane of rotation)

TCP	Tool center point
B	Programmed path
B_T	Path tangent vector
B_N	Path normal vector
D_0	Current rotation vector in the orientation plane E(⊥O)
D'_0	Current rotation vector in the rotation plane E(⊥R)
D'_1	End rotation vector in the rotation plane E(⊥R)
D_1	End rotation vector in the orientation plane E(⊥O)
α	Programmed rotation angle ($P_OFF_R_ANGLE)
β	Resulting angle of rotation
WCS	Tool coordinate system

Syntax

```
$P_OFF_R_DIR[<i>] = <value>
$P_OFF_R_ANGLE = <value>
```

Meaning

`$P_OFF_R_DIR`:	Direction vector for rotating the rotation vector	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	1: X coordinate (abscissa) 2: Y coordinate (ordinate) 3: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL

`$P_OFF_R_ANGLE`:	Rotation angle for rotating the rotation vector	
	Data type:	REAL
	Range of values:	± 90.0°
	Default value:	0.0
`<value>`:	Angle	
	Data type:	REAL

Example

The rotation vector is rotated 45.0° about the direction vector (1.0 ; 1.0; 1.0):

Program code	Comment
	; rotation vector:
`N100 $P_OFF_R_DIR[1] = 1.0`	; X coordinate = 1.0
`N110 $P_OFF_R_DIR[2] = 1.0`	; Y coordinate = 1.0
`N120 $P_OFF_R_DIR[3] = 1.0`	; Z coordinate = 1.0
`N130 $P_OFF_R_ANGLE = 45.0`	; angle of rotation = 45.0°

2.12.5 Programming: Main run variable ($AC_OFF_...)

2.12.5.1 Overview

As main run variables, the following system variables of the orientation superimposition can be read and written both via NC programs and synchronized actions. The functionality of the main run variables $AC_... listed here is identical to the respective preprocessing variable (Page 135) $P_... of the same name

Name	Meaning
$AC_OFF_O (Page 136) 2)	Offset vector of the orientation
$AC_OFF_R (Page 139) 1) 2)	Offset vector of the rotation vector of the orientation
$AC_OFF_LEAD (Page 142) 3)	Offset value of the LEAD angle
$AC_OFF_TILT (Page 142) 3)	Offset value of the TILT angle
$AC_OFF_THETA (Page 142) 1) 3)	Offset value of the THETA angle (rotation angle about the orientation vector)
$AC_OFF_O_DIR (Page 145) 2)	Direction vector about which the orientation vector is rotated with the rotation angle $AC_OFF_O_ANGLE
$AC_OFF_O_ANGLE (Page 145)	Rotation angle about which the orientation vector is rotated by the direction vector $AC_OFF_O_DIR
$AC_OFF_R_DIR (Page 148) 2)	Direction vector about which the rotation vector is rotated with the rotation angle $AC_OFF_R_ANGLE 1)
$AC_OFF_R_ANGLE (Page 148)	Rotation angle about which the rotation vector is rotated by the direction vector $AC_OFF_R_DIR 1)
1) Only relevant for 6-axis kinematics	
2) The superimposition of the orientation with the angles LEAD, TILT and THETA is always relative to the current path tangent and is therefore path-relative.	
3) The superimposition of the orientation by means of a direction vector is relative to the workpiece coordinate system (WCS).	

2.12.5.2 Status of the angle limitation

With the channel-specific setting data $SC_OFF_ORI_LIMIT (Page 134), the maximum possible angle between the current orientation vector O or rotation vector D and the new orientation vector O' or rotation vector D' that results from the orientation superimposition is specified. A limitation brought about as a result of this is displayed in the channel-specific system variable $AC_OFF_ORI_LIMIT.

Syntax

```
$AC_OFF_ORI_LIMIT[<index>] == <value>
```

Meaning

Index	Value	Meaning
0	TRUE	Angle limitation **active**: The angle between the current **orientation vector** O and the new orientation vector O' resulting from the orientation superimposition is **greater than** the value specified in $SC_OFF_ORI_LIMIT[**0**].
	FALSE	Angle limitation **not** active: The angle between the current **orientation vector** O and the new orientation vector O' resulting from the orientation superimposition is **less than/equal to** the value specified in $SC_OFF_ORI_LIMIT[**0**].
1	TRUE	Angle limitation **active**: The angle between the current **rotation vector** D and the new rotation vector D' resulting from the orientation superimposition is **greater than** the value specified in $SC_OFF_ORI_LIMIT[**1**].
	FALSE	Angle limitation **not** active: The angle between the current **rotation vector** D and the new rotation vector D' resulting from the orientation superimposition is **less than/equal to** the value specified in $SC_OFF_ORI_LIMIT[**1**].

2.12.6 Programming: NC program

2.12.6.1 Deletion of programmed main run superimposition values $AC_OFF_... (CORROF)

The following axis-specific overlays are deleted with the CORROF procedure:

- Additive work offsets (DRF offsets) set via handwheel traversal
- Position offsets programmed via the $AA_OFF system variable
- Overlays of the tool orientation programmed via the $AC_OFF_... system variable

A preprocessing stop is initiated through the deletion of an overlay value and the position component of the deselected overlaid movement is transferred to the position in the basic coordinate system. Whereby, no axis is traversed.

The position value that can be read via the $AA_IM system variable (current **MCS** setpoint of the axis) **does not change** in the **machine** coordinate system.

The position value that can be read via the $AA_IW system variable (current **WCS** setpoint of the axis) **changes** in the **workpiece**coordinate system because it now contains the deselected component of the overlaid movement.

Note

`CORROF` can be programmed in an **NC program**.

`CORROF` must **not** be programmed in a **synchronized action**.

Syntax

```
CORROF(<Axis>,"<String>"[,<Axis>,"<String>"])
```

Meaning

`CORROF`:	Procedure for the deselection of the following offsets and overlays of an axis: • DRF offset • Position offsets ($AA_OFF) • Overlay of the tool orientation ($AC_OFF_...)	
	Effective-ness:	Modal
`<Axis>`:	Axis identifier (channel, geometry or machine axis identifier)	
	Data type:	AXIS
`<String>`:	Character string for the definition of the overlay type	
	Data type:	BOOL
	Value	**Meaning**
	`DRF`	DRF offset
	`AA_OFF`	Position offset ($AA_OFF)
	`OFF_ORI`	Overlay of the tool orientation ($AC_OFF_...) **Note** The deselection of the overlay of the tool orientation is performed by deleting the axis-specific offsets of the orientation axes. An **arbitrary** channel axis can be specified as <Axis> parameter.

Examples

Example 1: Axis-specific deselection of a DRF offset (1)

A DRF offset is generated in the X axis by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

```
Program code               Comment
N10 CORROF(X,"DRF")        ; CORROF has the same effect as DRFOF here.
...
```

Example 2: Axis-specific deselection of a DRF offset (2)

A DRF offset is generated in the X and Y axes by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

Program code	Comment
`; Only the DRF offset of the X axis is deselected; the DRF offset of the Y axis is retained.`	
`; With DRFOF, both offsets would have been deselected.`	
`N10 CORROF(X,"DRF")`	
`...`	

Example 3: Axis-specific deselection of a $AA_OFF position offset

Program code	Comment
`; A position offset == 10 is interpolated for the X axis.`	
`N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5`	
`...`	
`; The position offset of the X axis is deselected: $AA_OFF[X]=0`	
`; The X axis is not traversed.`	
`; The position offset is added to the current position of the X axis.`	
`N80 CORROF(X,"AA_OFF")`	
`...`	

Example 4: Axis-specific deselection of a DRF offset and a $AA_OFF position offset (1)

A DRF offset is generated in the X axis by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

Program code	Comment
`; A position offset of 10 is interpolated for the X axis.`	
`N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5`	
`...`	
`; Only the DRF offset and the position offset of the X axis are deselected.`	
`; The DRF offset of the Y axis is retained.`	
`N70 CORROF(X,"DRF",X,"AA_OFF")`	
`...`	

Example 5: Axis-specific deselection of a DRF offset and a $AA_OFF position offset (2)

A DRF offset is generated in the X and Y axes by DRF handwheel traversal. No DRF offsets are operative for any other axes in the channel.

Program code	Comment
`; A position offset == 10 is interpolated for the X axis.`	
`N10 WHEN TRUE DO $AA_OFF[X]=10 G4 F5`	
`...`	
`; The DRF offset of the Y axis and the position offset of the X axis are deselected.`	
`; The DRF offset of the X axis is retained.`	

Program code	Comment
`N70 CORROF(Y,"DRF",X,"AA_OFF")`	
`...`	

Further information

$AA_OFF_VAL

Once the position offset has been deselected by means of $AA_OFF, system variable $AA_OFF_VAL (integrated distance of axis overlay) for the corresponding axis will equal zero.

$AA_OFF in JOG mode

Also in JOG mode, if $AA_OFF changes, the position offset will be interpolated as an overlaid movement if this function has been enabled via machine data MD 36750 $MA_AA_OFF_MODE.

$AA_OFF in synchronized action

If a synchronized action which immediately resets $AA_OFF (`DO $AA_OFF[<axis>]=<value>`) is active when the position offset is deselected using the `CORROF(<axis>,"AA_OFF")`, then $AA_OFF will be deselected and not reset, and alarm 21660 will be displayed. However, if the synchronized action becomes active later, e.g. in the block after `CORROF`, $AA_OFF will remain set and a position offset will be interpolated.

Automatic channel axis exchange

If an axis that is active in another channel has been programmed for a `CORROF`, it will be fetched into the channel with an axis exchange (requirement: MD30552 $MA_AUTO_GET_TYPE > 0) and the position offset and/or the DRF offset deselected.

2.12.6.2 Deletion of preprocessing superimposition values $P_OFF_...

If a currently effective preprocessing orientation superimposition value $P_OFF_... is to be deleted in a channel, you must proceed differently, depending on the setting in the machine data MD21096 $MC_OFF_ORI_MODE, bit n (absolute or incremental):

- Absolute
  The absolute mode is set in the machine data for the system variable with bit n = 0. The value 0.0 must be assigned to the system variable:
  `$P_OFF_... = 0.0`
- Incremental
  The incremental mode is set in the machine data for the system variable with bit n = 1. Its negative value must be assigned to the system variable:
  `$P_OFF_... = -$P_OFF_...`

2.12.7 Supplementary conditions

2.12.7.1 Reset behavior

Channel reset

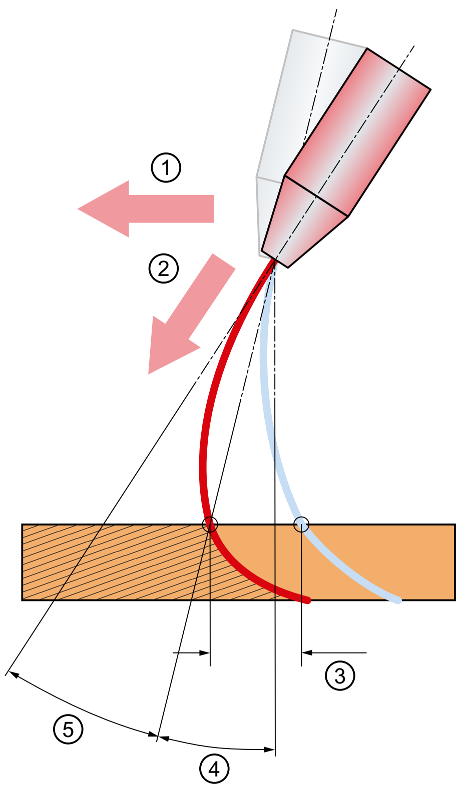
- The **preprocessing** superimposition values **$P**_OFF_... are **deleted** during a channel reset.
- Depending on the setting in the machine data MD21096 $MC_OFF_ORI_MODE, Bit 0, the **main run** superimposition values **$AC**_OFF_... are either deleted or continue to remain in effect upon channel reset.

2.13 Superimposition of the tool orientation, dynamic and speed-dependent

2.13.1 Function description

The path-relative superimposition of the tool orientation depends on the current path velocity. This property can be used particularly for water-jet machines to compensate the lag of the water jet. The lag occurs during processing due to the overlapping of the water jet speed and the current path velocity. In addition, the function can also be used to offset the conical expansion of the water jet from the outlet nozzle to the surface of the workpiece.

Lag correction



While it is being processed with a water jet, a dynamic lag error Δs occurs due to the overlapping of the traversing speed of the outlet nozzle and the output velocity of the water jet.

The actual impact point of the water jet on the surface of the workpiece deviates from the target impact point by this lag error Δs.

For a linear traversing movement, the "tail" that occurs due to the lag error has no effect on the shape of the cutting edge. For curved traversing movements, especially on contour corners, the lag error produces distorted workpiece edges.

The LEAD angle φ that is specified by the tool orientation is dynamically subjected to an offset angle Δφ by this function to compensate for the lag error.

①	Current path velocity v_B
②	Output velocity v_A
③	Lag error Δs

④ Current tool orientation, LEAD angle φ

⑤ Current correction angle Δφ

The correction angle Δφ is calculated as follows:

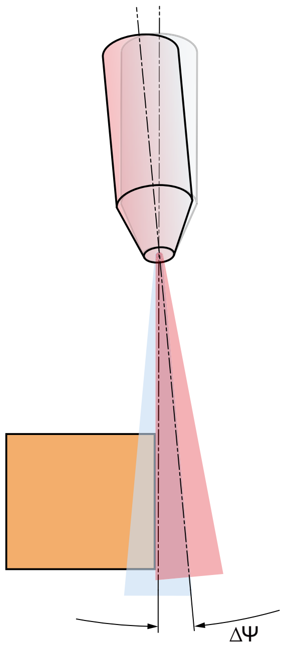
$$\Delta\varphi = \varphi_0 + \arctan\left(\frac{r * \cos\varphi}{1 + r * \cos\varphi}\right)$$

- r is the ratio of the path velocity to the outflow velocity: $r = v_B / v_A$
- φ is the LEAD angle specified by the current tool orientation
- φ_0 is an additional constant LEAD angle offset specified by the machine data MD21144 $MC_DYN_ORI_OFF_ANGLE[0]

For the tilt angles ψ (TILT) and the angle of rotation θ (THETA), only constant offsets which are independent of the path velocity can be specified:

- TILT angle offset: $\Delta\psi = \psi_0$ = (MD21144 $MC_DYN_ORI_OFF_ANGLE[**1**])
- THETA angle offset: $\Delta\theta = \theta_0$ = (MD21144 $MC_DYN_ORI_OFF_ANGLE[**2**])

Correct the jet expansion



During water jet processing, the jet expands after it exits the nozzle. The expansion can be statically corrected, i.e. independently of the path velocity, by an angle offset of the tilt angle ψ (TILT):

TILT angle offset: $\Delta\psi = \psi_0$ = (MD21144 $MC_DYN_ORI_OFF_ANGLE[**1**])

Δψ	Angle offset of the tilt angle ψ (TILT)

2.13.2 Commissioning: Machine data, channel-specific

Activation

The dynamic overlapping of the tool orientation is activated with the machine data:

MD21140 $MC_DYN_ORI_OFF_ON = TRUE

Water jet flow rate

The output velocity of the water jet is specified in the machine data:

MD21142 $MC_DYN_ORI_OFF_VEL = <velocity>

Additional offset angles

Additional constant offset angles (φ_0, ψ_0 and θ_0) can be specified in the machine data:

- MD21144 $MC_DYN_ORI_OFF_ANGLE[0] = <LEAD angle>
- MD21144 $MC_DYN_ORI_OFF_ANGLE[1] = <TILT angle>
- MD21144 $MC_DYN_ORI_OFF_ANGLE[2] = <THETA angle>

2.13.3 Programming: Main run variable ($AC_TOOL_...)

2.13.3.1 Overview

As main run variables, the following system variables of the dynamic, velocity-dependent orientation superimposition can be read both via NC programs and synchronized actions.

Name	Meaning
$AC_TOOL_O_CORR (Page 162)	Corrected orientation vector
$AC_TOOL_R_CORR (Page 163)	Corrected rotation vector
$AC_TOOL_O_CORRD (Page 164)	Superimposition vector of the orientation
$AC_TOOL_R_CORRD (Page 165)	Superimposition vector of the rotation

The system variables are described in detail in the following sections.

Further offset values

In addition to the above-mentioned system variables, the following system variables can also be used for the dynamic and speed-dependent superimposition of the tool orientation:

- $AC_OFF_O (Page 136)
- $AC_OFF_R (Page 139)
- $AC_OFF_LEAD (Page 142)

- $AC_OFF_TILT (Page 142)
- $AC_OFF_THETA (Page 142)

2.13.3.2 Corrected orientation vector

The corrected orientation vector of the dynamic, speed-dependent superimposition can be read in various coordinate systems via the system variable in an NC program or synchronous action.

Syntax

```
<Wert> = $AC_TOOL_O_CORR[<Index_1>, [<Index_2>]
```

Meaning

`$AC_TOOL_O_CORR`:	Corrected orientation vector	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`Index_1`:	Vector coordinate	
	Data type:	INT
	Values	**Meaning**
	1	X coordinate
	2	Y coordinate
	3	Z coordinate
`Index_2`:	Coordinate system	
	Data type:	INT
	Values	**Meaning**
	0	Basic coordinate system (BCS)
	1	Workpiece coordinate system (WCS)
	2	Settable zero system (SZS)
`<value>`:	Variable	
	Data type:	REAL

Example

Reading of the corrected orientation vector of the dynamic, speed-dependent superimposition in the workpiece coordinate system:

Program code

```
DEF REAL V_CorrOri[3] = REP(0.0)
DEF INT 3D_V_Start = 1
DEF INT BKS = 0, WKS = 1, ENS = 2

; read corrected rotation vector in the WCS:
; R parameter 100 = X coordinate
; R parameter 101 = Y coordinate
; R parameter 102 = Z coordinate
FOR n=0 TO 2
   V_CorrOri[n] = $AC_TOOL_O_CORR[3D_V_Start + n, WCS]
ENDFOR
```

2.13.3.3 Corrected rotation vector

The corrected rotation vector of the dynamic, speed-dependent superimposition can be read in various coordinate systems via the system variable in an NC program or synchronous action.

Syntax

```
<Value> = $AC_TOOL_R_CORR[<Index_1>, [<Index_2>]
```

Meaning

`$AC_TOOL_R_CORR`:	Corrected rotation vector	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`Index_1`:	Vector coordinate	
	Data type:	INT
	Values	**Meaning**
	1	X coordinate
	2	Y coordinate
	3	Z coordinate
`Index_2`:	Coordinate system	
	Data type:	INT
	Values	**Meaning**
	0	Basic coordinate system (BCS)
	1	Workpiece coordinate system (WCS)
	2	Settable zero system (SZS)
`<value>`:	Variable	
	Data type:	REAL

Example

Reading of the corrected rotation vector of the dynamic, speed-dependent superimposition in the workpiece coordinate system:

Program code

```
DEF REAL V_CorrRot[3] = REP(0.0)
DEF INT 3D_V_Start = 1
DEF INT BKS=0, WKS=1, ENS=2, RParVektor=100

; read corrected rotation vector in the WCS:
; R parameter 100 = X coordinate
; R parameter 101 = Y coordinate
; R parameter 102 = Z coordinate
FOR n=0 TO 2
   V_CorrRot[n] = $AC_TOOL_R_CORR[3D_V_Start+n, WCS]
ENDFOR
```

2.13.3.4 Superimposition vector of the orientation

The superimposition vector of the orientation can be read in various coordinate systems via the system variable in an NC program or synchronous action.

Syntax

```
<Value> = $AC_TOOL_O_CORRD[<Index_1>, [<Index_2>]
```

Meaning

`$AC_TOOL_O_CORRD`:	Superimposition vector of the orientation	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`Index_1`:	Vector coordinate	
	Data type:	INT
	Values	**Meaning**
	1	X coordinate
	2	Y coordinate
	3	Z coordinate
`Index_2`:	Coordinate system	
	Data type:	INT
	Values	**Meaning**
	0	Basic coordinate system (BCS)
	1	Workpiece coordinate system (WCS)
	2	Settable zero system (SZS)
`<value>`:	Variable	
	Data type:	REAL

Example

Reading of the superimposition vector of the orientation in the workpiece coordinate system:

Program code

```
DEF REAL V_Ori[3] = REP(0.0)
DEF INT 3D_V_Start = 1
DEF INT BKS = 0, WKS = 1, ENS = 2

; read superimposition vector of the orientation in the WCS:
; R parameter 100 = X coordinate
; R parameter 101 = Y coordinate
; R parameter 102 = Z coordinate
FOR n=0 TO 2
   V_Ori[n] = $AC_TOOL_O_CORRD[3D_V_Start + n, WCS]
ENDFOR
```

2.13.3.5 Superimposition vector of the rotation

The superimposition vector of the rotation can be read in various coordinate systems via the system variable in an NC program or synchronous action.

Syntax

```
<Value> = $AC_TOOL_R_CORRD[<Index_1>, [<Index_2>]
```

Meaning

`$AC_TOOL_R_CORRD`:	Superimposition vector of the rotation	
	Data type:	REAL
	Value range:	± max. REAL value
	Default value:	0.0, 0.0, 0.0
`Index_1`:	Vector coordinate	
	Data type:	INT
	Values	**Meaning**
	1	X coordinate
	2	Y coordinate
	3	Z coordinate
`Index_2`:	Coordinate system	
	Data type:	INT
	Values	**Meaning**
	0	Basic coordinate system (BCS)
	1	Workpiece coordinate system (WCS)
	2	Settable zero system (SZS)
`<value>`:	Variable	
	Data type:	REAL

Example

Reading of the superimposition vector of the rotation in the workpiece coordinate system:

Program code

```
DEF REAL V_Rot[3] = REP(0.0)
DEF INT 3D_V_Start = 1
DEF INT BKS = 0, WKS = 1, ENS = 2

; read superimposition vector of the orientation in the WCS:
; R parameter 100 = X coordinate
; R parameter 101 = Y coordinate
; R parameter 102 = Z coordinate
FOR n=0 TO 2
   V_Rot[n] = $AC_TOOL_R_CORRD[3D_V_Start + n, WCS]
ENDFOR
```

2.14 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.

Reference:
Function Manual, Basic Functions; Tool Offset, Section: Orientable toolholders (W1)

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 on-line tool offset)	Acceleration of online tool length offset
SD42970 $SC_TOFF_LIMIT (upper limit of off-set 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:

Reference:
Programming Manual, Job Planning; 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.

Reference:
Parameter 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 machining 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.

2.15 Examples

2.15.1 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
```

2.15.2 Example of a 3-axis and 4-axis transformation

2.15.2.1 Example of a 3-axis transformation

Example: For the schematically represented machine (see "Figure 2-1 Schematic diagram of 3-axis transformation (Page 40)"), 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

2.15.2.2 Example of a 4-axis transformation

Example: For the schematically represented machine (see "Figure 2-2 Schematic diagram of a 4-axis transformation with moveable workpiece (Page 40)"), 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`

2.15.3 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

Program code	Comment
`; Definition of tool T1`	

Program code	Comment
`$TC_DP1[1,1] = 120`	`; Type`
`$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

2.15.4 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 of first orientation axis`
`$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[1]=5`	`; Channel index of second orientation ax-is`
`$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[2]=6`	`; Channel index of third orientation axis`

Program code	Comment
`$MC_ORIAX_TURN_TAB_1[0]=3`	; Z direction
`$MC_ORIAX_TURN_TAB_1[1]=2`	; Y direction
`$MC_ORIAX_TURN_TAB_1[2]=3`	; Z direction
`CHANDATA(1)`	
`M17`	

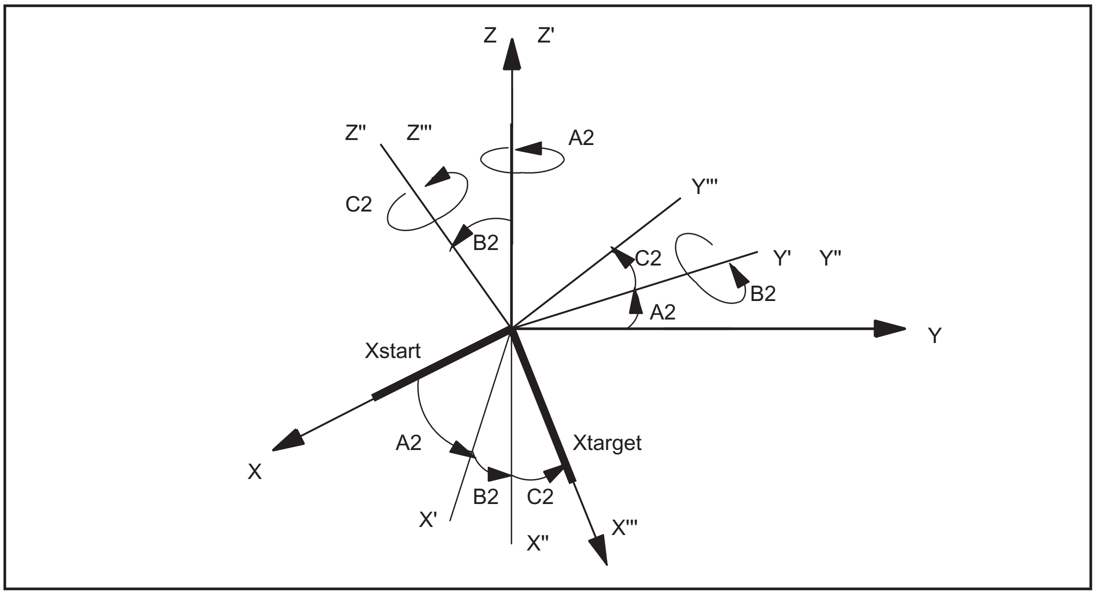


Figure 2-27 3 orientation axes for the 1st orientation transformation for kinematics with 6 transformed axes

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 of first orientation axis
`$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[1]=5`	; Channel index of second orientation axis
`$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[2]=0`	; Channel index of third orientation 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

Program code	Comment
CHANDATA(1)	
M17	

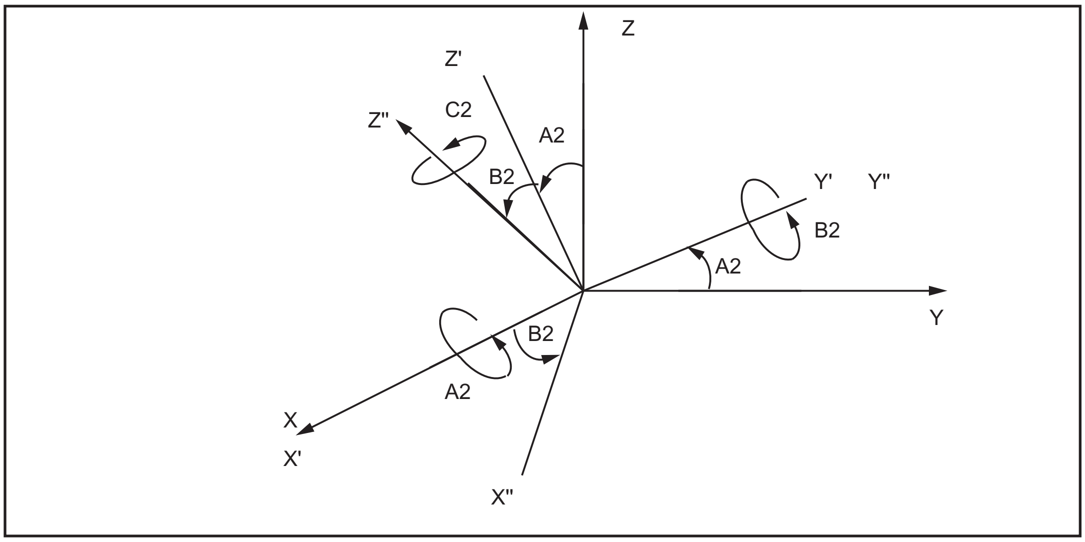


Figure 2-28 3 orientation axes for the 2nd orientation transformation for kinematics with 5 transformed axes

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.

References:
Programming Manual, Production Planning

2.15.5 Examples for orientation vectors

2.15.5.1 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) PO[PSI]=(10)`	`; in +X direction (end vector)`

2.15.5.2 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`

Program code	Comment
	; 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.

2.15.6 Examples for generic axis transformations

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

Program code		Comment
		; 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
		; 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	

2.15.6.1 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

Program code	Comment
`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`
`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`	

2.15.6.2 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.

2.15.6.3 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.

2.16 Data lists

2.16.1 Machine data

2.16.1.1 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
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

2.16.1.2 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

Number	Identifier: $MC_	Description
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
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]

Number	Identifier: $MC_	Description
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]
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]

Number	Identifier: $MC_	Description
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]
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

Number	Identifier: $MC_	Description
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

2.16.2 Setting data

2.16.2.1 General setting data

Number	Identifier: $SN_	Description
41110	JOG_SET_VELO	Geometry axes
41130	JOG_ROT_AX_SET_VELO	Orientation axes

2.16.2.2 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

2.16.3 Signals

2.16.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate PTP traversal	DB21, … .DBX29.4	-

2.16.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Transformation active	DB21, ...DBX33.6	DB330x.DBX1.6
Number of the active G commands of G group 25	DB21, ...DBB232	-
PTP traversal active	DB21, ...DBX317.6	-
Online tool length compensation (TOFF) active	DB21, ...DBX318.2	-
Online tool length compensation (TOFF): Compensation motion active	DB21, ...DBX318.3	-

G1: Gantry axes

3

3.1 Brief description

For gantry machines, each of various machine elements, such as the gantry and the transverse beams, are moved by several axes that operate in parallel. The axes that together move a machine part, are designated as gantry axes or gantry grouping. Because of the mechanical structure, the gantry axes are rigidly connected with each other and so must always be traversed synchronously by the control.

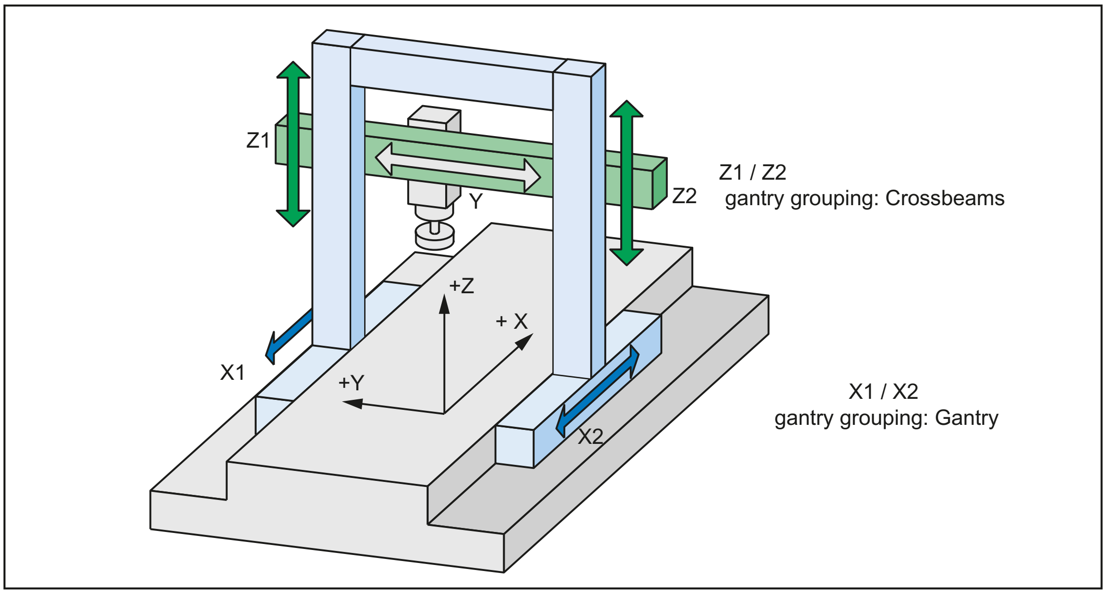


Figure 3-1 Example: Gantry-type milling machine with gantry and transverse beams

Guide axis

The guide axis of the gantry grouping is the axis that represents the gantry grouping. Only this axis is programmed to perform the traversing movements of the gantry grouping.

Synchronous axes

The synchronous axes of the gantry grouping are the axes that because of their coupling with the guide axis are also automatically traversed by the control. A guide axis can be assigned any number of synchronous axes.

Synchronization difference

The synchronous operation difference is the deviation of the axial actual value of a synchronous axis from its ideal position referred to the actual value of the guide axis. The control continually monitors the synchronous operation difference. A message is displayed if the alarm limit is exceeded. The complete gantry grouping is stopped when the alarm limit is exceeded. The limit values can be parameterized as machine data.

3.2 "Gantry axes" function

3.2.1 Definition of a gantry grouping

Definition

The axes of a gantry grouping are specified via the following axial machine data:

MD37100 $MA_GANTRY_AXIS_TYPE[AX1] = xy	
x	Tens decimal place: Type of gantry axis (guide or synchronous axis)
y	Ones decimal place: ID of the gantry grouping

A maximum of eight gantry groupings (gantry grouping ID: 1 - 8) can be defined. The gantry grouping ID must be unique in all channels or in all NCUs in accordance with the assigned axis.

In principle, a gantry grouping can be assigned any number of synchronous axes.

Example

Definition of a gantry grouping with ID=1, guide axis AX1 and synchronous axis AX2

- MD37100 $MA_GANTRY_AXIS_TYPE[AX1] = 01 (guide axis)
- MD37100 $MA_GANTRY_AXIS_TYPE[AX2] = 11 (synchronous axis)

Supplementary conditions

The following supplementary conditions apply to a gantry grouping:

- A gantry grouping must not contain a spindle.
- A synchronous axis must not be a concurrent POS axis.
- A synchronous axis must not belong to a transformation.
- A synchronous axis must not be a following axis of another axis coupling.
- A synchronous axis must not be a guide axis of another axis coupling.
- All axes of a gantry grouping must be of the same axis type, linear or rotary: MD30300 $MA_IS_ROT_AX (rotary axis/spindle)

Note

Drive optimization

At a SINAMICS S120 drive unit, a maximum of three drives can be optimized or measured at the same time (speed controller optimization / function generator). Therefore, for a coupling with more than three coupled drives at the same time, we recommend that these are distributed over several drive units.

3.2.2 Monitoring the synchronism difference

Limit values for monitoring

2 limit values can be specified for the synchronism difference.

Gantry warning limit

The gantry warning limit is set using the following machine data:

MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

The "Alarm limit exceeded" message is displayed if the synchronism difference exceeds the gantry warning limit. In addition, the NC/PLC-interface signal is set:

DB31, ... DBX101.3 = 1 (gantry warning limit exceeded)

After the alarm limit has been fallen below, the message and interface signal are automatically reset.

Note

Gantry warning limit

If the "Alarm limit exceeded" message is not to be displayed, then a value of 0 should be entered into MD37110.

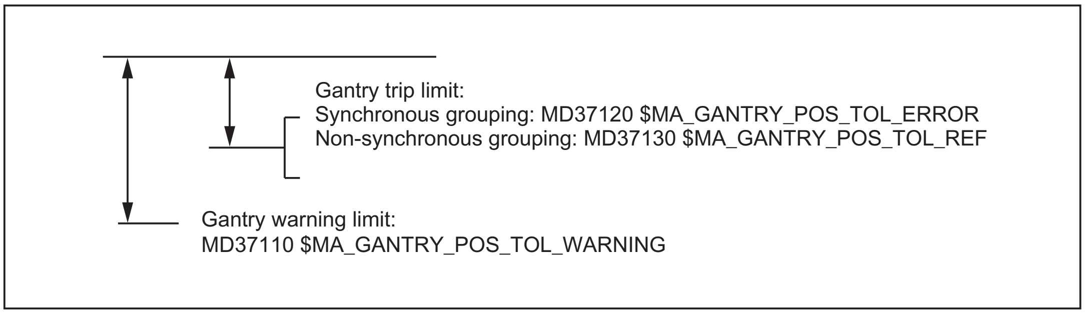
Gantry trip limit

The gantry trip limit is set using the following machine data:

- for the synchronized gantry grouping:
  MD37120 $MA_GANTRY_POS_TOL_ERROR
- for the non-synchronized gantry grouping:
  MD37130 $MA_GANTRY_POS_TOL_REF

Alarm 10653 "Error limit exceeded" is displayed if the synchronism difference exceeds the gantry trip limit. In addition, the NC/PLC-interface signal is set:

DB31, ... DBX101.2 = 1 (gantry trip limit exceeded)


Gantry trip limit:
Synchronous grouping: MD37120 $MA_GANTRY_POS_TOL_ERROR
Non-synchronous grouping: MD37130 $MA_GANTRY_POS_TOL_REF
Gantry warning limit:
MD37110 $MA_GANTRY_POS_TOL_WARNING

The alarm is also displayed if the gantry grouping is jammed (no controller enable, gantry grouping in the "Hold" state).

3.2.3 Extended monitoring of the synchronism difference

Activation of the extended monitoring

An extended monitoring of the synchronism difference can be activated using the following machine data:

MD37150 $MA_GANTRY_FUNCTION_MASK, Bit 0 = 1

For the extended monitoring, a synchronism difference between the leading and synchronous axis, obtained when tracking or when the gantry grouping is opened, is taken into account.

The extended monitoring becomes active after the NC boots after the first referencing (incremental encoder) or synchronization (absolute encoder).

Exceeding the gantry trip limit

If, when the extended monitoring is active, the trip limit of the synchronism difference is exceeded, Alarm 10653 "Error limit exceeded" is displayed.

In order to be able to reset the alarm, proceed as follows:

1. Deactivating extended monitoring:
   MD37150 $MA_GANTRY_FUNCTION_MASK, Bit 0 = 0
2. Deleting the synchronism difference displayed in the machine data:
   MD37135 $GANTRY_ACT_POS_TOL_ERROR = 0
3. Cancel alarm
4. Re-reference or re-synchronize the axes of the gantry grouping
5. Reactivating extended monitoring:
   MD37150 $MA_GANTRY_FUNCTION_MASK, Bit 0 = 1

3.2.4 Referencing and synchronization of gantry axes

Application

In cases where an incremental measuring system is being used for the guide axis or the synchronous axis, the measuring systems must be referenced during NC run-up, while maintaining the axis coupling.

After every axis in the gantry grouping has approached its reference point, any misalignment that may exist between the axes must be eliminated (gantry synchronization process). Once this has been done, the NC/PLC interface signal is set:

DB31, ... DBX101.5 = 1 (gantry grouping is synchronized)

For the sequence when referencing or synchronizing gantry axes, see Chapter "Referencing and synchronization of gantry axes (Page 194)".

3.2.5 Control dynamics

Use case

From the user perspective, a gantry grouping is exclusively traversed via the leading axis. The NC generates the setpoints of the synchronous axes directly from the setpoints of the leading axis in time synchronism and outputs these to them. To minimize the synchronous operation differences, the control system dynamics of all axes of a gantry grouping must be set identical (see Section "Start-up of gantry axes (Page 204)").

Note

Identical control dynamics must be set for all axes of a gantry grouping.

Disturbance characteristic

If faults occur, which cause an axis of the gantry to be stopped, then the complete gantry grouping is always stopped.

3.2.6 Opening the gantry grouping

Description

The axis coupling within a gantry grouping can be opened (dissolved) using the following machine data:

MD37140 $MA_GANTRY_BREAK_UP = 1 (invalidate gantry grouping)

⚠ **CAUTION**

No synchronous operation

If the gantry axes remain mechanically coupled, there is a risk of damage to the machine when the leading or synchronous axes are traversed in this operating state!

When the setting becomes active, the axes of the gantry grouping can be individually traversed in the JOG, AUTOMATIC and MDA modes.

The monitoring functions of the synchronism difference and/or the alarm and trip limits are not active.

The NC/PLC interface signal "Gantry grouping is synchronized" is reset:

DB31, ... DBX101.5 = 0

3.3 Referencing and synchronization of gantry axes

3.3.1 Introduction

Misalignment after starting

Immediately after the machine is switched on, the leading and synchronous axes may not be ideally positioned in relation to one another (e.g. misalignment of a gantry). Generally speaking, this misalignment is relatively small so that the gantry axes can still be referenced.

In special cases (e.g. gantry axes were stopped owing to a disturbance, power failure or EMERGENCY STOP), the dimensional offset must be checked for permissible tolerance values and a compensatory motion executed if necessary before the axes are traversed.

To execute this compensatory motion, the gantry grouping must be invalidated by means of the following machine data:

MD37140 $MA_GANTRY_BREAK_UP (invalidate gantry grouping)

Gantry synchronization process

All gantry axes must first be referenced and then synchronized after the control system is switched on. During gantry synchronization, all gantry axes approach **the reference position of the gantry grouping in the decoupled state**. The reference position of the gantry grouping for referencing the gantry axes corresponds to the **reference position of the leading axis:**

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distancecoded system)

Otherwise, the reference position is the **current actual position of the leading axis**.

These operations for referencing and synchronizing the gantry axes are executed automatically in accordance with a special flowchart.

Referencing process

The flowchart for referencing gantry axes using an incremental measuring system is as follows:

Section 1:

Referencing of the leading axis

The axis-specific referencing of the gantry axis will be started by the active machine function REF upon the leading axis' interface signal from the PLC user program:

DB31, ... DBX4.7/4.6 (traversing key plus/minus)

The leading axis approaches the reference point (operational sequence as for reference point approach).

References:
Function Manual Basic Functions; Reference Point Approach (R1)

The appropriate synchronous axes traverse in synchronism with the leading axis. Interface signal "Referenced/synchronized" of the leading axis is output to indicate that the reference point has been reached.

Section 2:

Referencing of the synchronous axes

As soon as the leading axis has approached its reference point, the synchronous axis is **automatically** referenced (as for reference point approach).

References:
Function Manual Basic Functions; Reference Point Approach (R1)

The dependency between the leading axis and synchronous axis is inverted in the control for this phase so that the leading axis now traverses in synchronism with the synchronous axis. IS "Referenced/synchronized" of the synchronous axis is output to indicate that the reference point has been reached. The gantry axis dependency then reverts to its previous status. If a further synchronous axis is defined in the grouping, then this is also referenced in the way described above.

Section 3:

Gantry synchronization process

Once all axes in the gantry grouping have been referenced, they must be synchronized with the defined reference position. The actual position of each gantry axis is first compared to the defined reference position of the leading axis.

The next step in the operating sequence depends on the difference calculated between the actual values of the leading and synchronous axes:

- difference is **smaller** than the gantry warning limit:
  MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)
  The gantry synchronization process is started **automatically**. The message "Synchronization in progress gantry grouping x" is output during this process.
  The message "Synchronization running gantry grouping x" can be suppressed with:
  MD37150 $MA_GANTRY_FUNCTION_MASK Bit 2 = 1
  All gantry axes traverse at a specific position value **in the decoupled state** at the velocity set in the machine data:
  MD34040 $MA_REFP_VELO_SEARCH_MARKER (creep velocity)
  The position value is defined by the leading axis:
  MD34100 $MA_REFP_SET_POS (reference point/destination point for distance-coordinated system)
  The absolute encoders and distanced-coded encoders of the leading axis will be set to the current actual position of the leading axis or to the reference point by the following machine data:
  MD34330 $MA_REFP_STOP_AT_ABS_MARKER (Distancecoded linear measuring system without destination point)
  For this operation, the axes traverse at the same velocity as set for reference point approach:
  MD34070 $MA_REFP_VELO_POS (reference point positioning velocity)
  As soon as all gantry axes have reached their target position (ideal position), IS "Gantry grouping is synchronized" is set to "1" followed by re-activation of the gantry axis coupling. The position actual value of all axes in the gantry grouping must now be identical. The gantry synchronization process is now complete.
- Difference is **higher** than the gantry warning limit for at least one synchronous axis:
  IS "Gantry synchronization read to start" is set to "1" and the message "Wait for synchronization start of gantry grouping x" is output. The gantry synchronization process is not started automatically in this case, but must be started explicitly by the operator or from the PLC user program. The process is initiated by IS "Start gantry synchronization" on the leading axis. The signal is set on the leading axis. The operational sequence is then the same as that described above.

The following flowchart illustrates the referencing and synchronization processes.

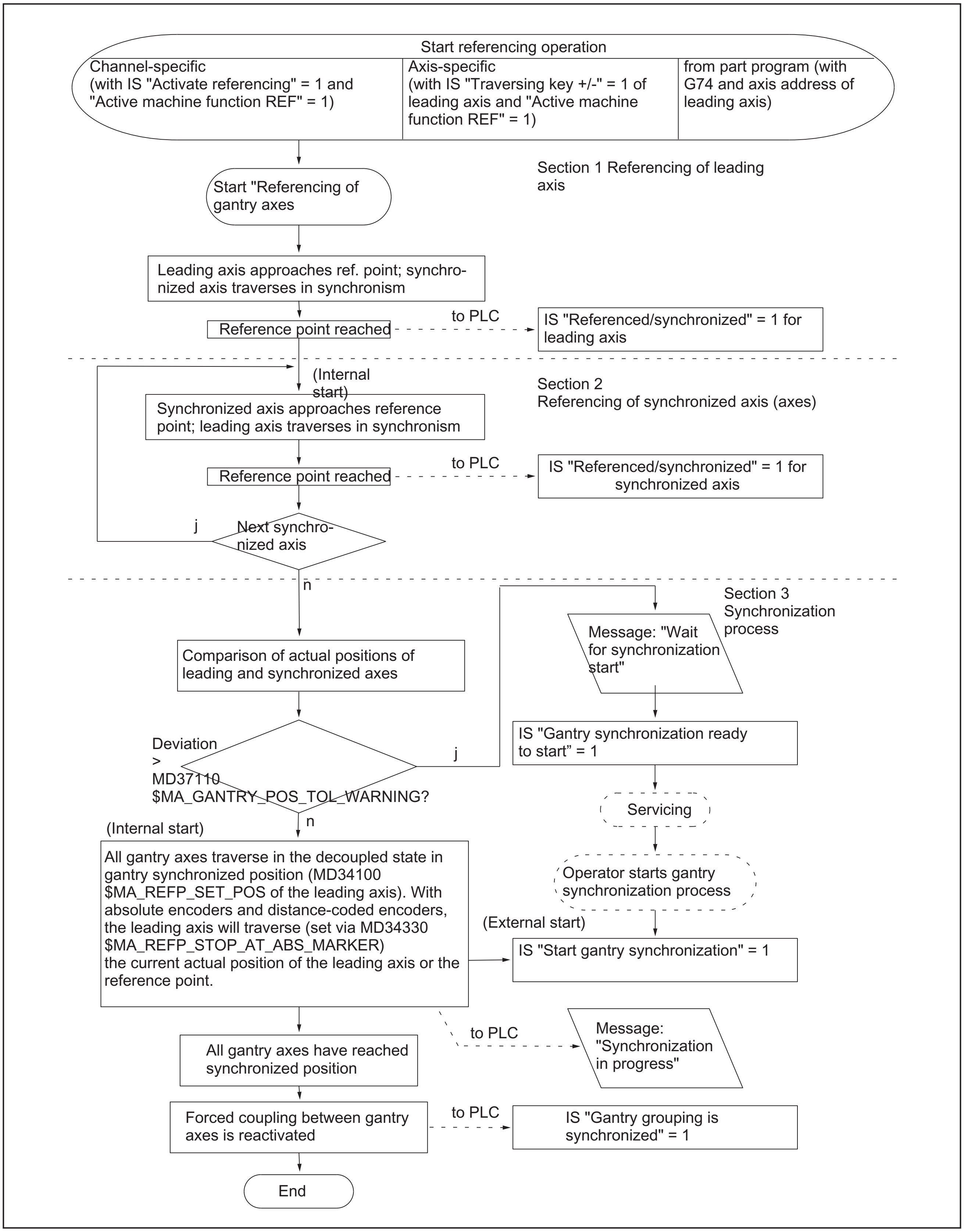

Start referencing operation
Channel-specific (with IS "Activate referencing" = 1 and "Active machine function REF" = 1)
Axis-specific (with IS "Traversing key +/-" = 1 of leading axis and "Active machine function REF" = 1)
from part program (with G74 and axis address of leading axis)
Start "Referencing of gantry axes
Section 1 Referencing of leading axis
Leading axis approaches ref. point; synchronized axis traverses in synchronism
Reference point reached
to PLC
IS "Referenced/synchronized" = 1 for leading axis
(Internal start)
Section 2 Referencing of synchronized axis (axes)
Synchronized axis approaches reference point; leading axis traverses in synchronism
Reference point reached
to PLC
IS "Referenced/synchronized" = 1 for synchronized axis
j
Next synchronized axis
n
Section 3 Synchronization process
Message: "Wait for synchronization start"
Comparison of actual positions of leading and synchronized axes
Deviation > MD37110 $MA_GANTRY_POS_TOL_WARNING?
j
IS "Gantry synchronization ready to start" = 1
Servicing
n
(Internal start)
All gantry axes traverse in the decoupled state in gantry synchronized position (MD34100 $MA_REFP_SET_POS of the leading axis). With absolute encoders and distance-coded encoders, the leading axis will traverse (set via MD34330 $MA_REFP_STOP_AT_ABS_MARKER) the current actual position of the leading axis or the reference point.
Operator starts gantry synchronization process
(External start)
IS "Start gantry synchronization" = 1
to PLC
Message: "Synchronization in progress"
All gantry axes have reached synchronized position
Forced coupling between gantry axes is reactivated
to PLC
IS "Gantry grouping is synchronized" = 1
End

Figure 3-2 Flowchart for referencing and synchronization of gantry axes

Synchronization process

A synchronization process is always required in the following cases:

- after the reference point approach of all axes included in a grouping,
- if the axes become de-synchronized (s below).

Operational sequence failure

If the referencing process described above is interrupted as a result of disturbances or a RESET, proceed as follows:

- **Abort within section 1 or 2:**
  Restart reference point with leading axis (see section 1)
- **Abort in section 3:**
  In cases where the gantry axes have not yet been referenced (IS "Referenced/ Synchronized" = 1), the gantry synchronization process can be started again with IS "Synchronize gantry grouping".

Restart gantry synchronization

Synchronization of the gantry axes can be started with IS "Start gantry synchronization" under the following conditions only:

- JOG/REF mode must be active. The following interface signal must be set:
  DB11, ... DBX5.2 = 1 (active machine function REF)
- DB31, ... DBX 101.5 = 0 (gantry grouping is synchronized)
- All grouping axes operate within the tolerance windows:
  DB31, ... DBX 101.4 = 1(gantry synchronization process ready to start)
- Axes are not referenced in the relevant NC channel
  DB21, ... DBX33.0 = 0 (referencing active)

If the gantry synchronization process is **not started from the referencing process** by means of IS "Start gantry synchronization process", then the reference position is not specified as target position for the synchronous axes:

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distancecoded system)

Instead, **the actual position of the leading axis** is specified as the target position and is approached in the uncoupled state.

Note

For the leading axis, automatic synchronization can be locked using the following NC/PLC interface signal:

DB31, ... DBX29.5 = 1 (no automatic synchronization process)

This always makes sense if no axis enabling signal has yet been issued for the axes. In this case, the synchronization process should also be started explicitly with the NC/PLC interface signal:

DB31, ... DBX29.4 = 1 (start gantry synchronization process)

Loss of synchronization

The gantry grouping becomes desynchronized as a result of:

- "Tracking" the gantry axes
- Loss of the reference position of a gantry axis, e.g. by "Parking" (no measuring system active)
- Re-referencing of gantry axis
- The gantry grouping is opened (dissolved) by:
  MD37140 $MA_GANTRY_BREAK_UP = 0 (invalidate gantry axis grouping)

The corresponding NC/PLC interface signal is reset:

- DB31, ... DBX60.4 or DBX60.5 == 0 (referenced/synchronized 1 or 2 respectively)
- DB31, ... DBX 101.5 == 0 (gantry grouping is synchronized)

If, in operation, gantry grouping synchronization is lost due to a fault, then synchronization can be restarted using the NC/PLC interface signal:

DB31, ... DBX29.4 == 1 (start gantry synchronization process)

Requirement is that the following applies to all axes of the gantry grouping:

DB31, ... DBX60.4 or DBX60.5 = 1 (referenced/synchronized 1 or 2 respectively)

In this case, the synchronizing axes traverse the current actual position of the leading axis in the decoupled state.

If, when the gantry grouping is traversing, the signal "Emergency Stop" (DB10, DBX56.2) is set and again reset, and the gantry axes have drifted apart less than the standstill tolerance of the synchronous axes, then these are automatically resynchronized. Automatic synchronization can be suppressed using the NC/PLC interface signal for the leading axis:

DB31, ... DBX29.5 = 1 (no automatic synchronization process)

Selecting the reference point

To ensure that the shortest possible paths are traversed when the gantry axes are referenced, the reference point values from leading and synchronous axes should be the same in the machine data:

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distancecoded system)

Allowance for deviations in distance between the zero mark and the reference point must be made for specific axes via the machine data:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)

MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset/absolute offset)

Referencing direction selection

The zero mark search direction of the synchronous axis can be defined via the machine data:

MD37150 $MA_GANTRY_FUNCTION_MASK, Bit 1

Bit	Value	Meaning
1	0	Zero mark search direction of the synchronous axis analog to machine data: MD34010 $MA_REFP_CAM_DIR_IS_MINUS
	1	Zero mark search direction of the synchronous axis the same as the leading axis

During referencing, the reference point value of the leading axis is specified as the target position for all axes in the grouping for the synchronization compensatory motion. This position is then approached without axis coupling. The absolute encoders and distanced-coded encoders of the leading axis will be set to the current actual position of the leading axis or to the reference point by the following machine data:

MD34330 $MA_REFP_STOP_AT_ABS_MARKER (Distancecoded linear measuring system without destination point)

If only one reference cam is used for the leading and synchronous axes, then this must be taken into account in the PLC user program.

3.3.2 Automatic synchronization

Automatic synchronization can take place:

- in the referencing mode (see Section "Introduction (Page 194)")
- in other modes, as described below:

If a gantry grouping is switched to follow-up mode, monitoring of the actual values between the leading and synchronized axes is disabled. The grouping is no longer synchronized as a result. Independent of axes positions, the following interface signal will be set to 0 (from leading axis)

DB31, ... DBX101.5 (gantry grouping is synchronous)

If the gantry grouping is switched from follow-up mode to position control mode, axis synchronism is automatically restored provided the actual-value monitor does not detect a difference between the positions of the leading and synchronized axes greater than the setting in the machine data:

MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

In this case, a new setpoint is specified for the synchronized axis (axes) without interpolation. The positional difference detected earlier is then corrected by the position controller. The correction causes only the synchronized axis (axes) to move.

The motional sequence of the synchronized axis (axes) is analogous to the situation in which the grouping switches from the "Hold" state to position control mode. In this case, the position specified by the position controller before the grouping is halted is set again on condition that the zero speed monitor has not activated alarm 25040 (with follow-up as alarm reaction) in the meantime.

The same tolerance window is used for this mode of automatic synchronization as for the zero speed monitoring function:

MD36030 $MA_STANDSTILL_POS_TOL (standstill tolerance)

Parameter rate dependence loads with machine data:

MD36012 $MA_STOP_LIMIT_FACTOR (exact stop coarse/fine and standstill factor)

Note

The following interface signal blocks automatic synchronization in all modes except the referencing mode:

DB31, ... DBX29.5 (no automatic synchronization)

Should the automatic synchronization be activated at this point, then the following interface signal must be reset:

DB31, ... DBX29.5 = 0 (no automatic synchronization)

Then switch one of the axes in the gantry grouping from follow-up mode to position-controlled mode. This is achieved with the interface signals:

DB31, ... DBX1.4 = 1 (follow-up mode)

DB31, ... DBX2.1 = 1 (servo enable)

3.3.3 Points to note

2nd position measuring system per gantry axis

Different types of position measuring systems can be mounted on the gantry axes of a grouping. Furthermore, each gantry axis is capable of processing two position measuring systems, it being possible to switch over from one system to the other at any time:

DB31, ... DBX1.5 (position measuring system 1)

DB31, ... DBX1.6 (position measuring system 2)

The maximum tolerance for position actual value switchover should be set to a lower value than the gantry warning limit:

MD36500 $MA_ENC_CHANGE_TOL (Max. tolerance for position actual value switchover)

The two position measuring systems must, however, have been referenced beforehand. The relevant measuring system must be selected before referencing is initiated. The operational sequence is then the same as that described above.

Channelspecific referencing

Gantry axes can also be referenced by channel with the following interface signal:

DB21, ... DBX1.0 (activate referencing)

The value of the leading axis' machine data is used for the axis sequence for channel-specific referencing:

MD34110 $MA_REFP_CYCLE_NR (Axis sequence for channel-specific referencing)

After the reference point of the leading axis has been reached, the synchronized axes are referenced first as described above.

Referencing from part program with G74

The referencing and synchronization process for gantry axes can also be initiated from the part program by means of command G74. In this case, only the axis name of the leading axis may be programmed. The operational sequence is analogous to that described for axis-specific referencing.

Position measuring system with distancecoded reference marks

In order that reference point approach operations do not have to travel through large distances, it is possible to use just one position measuring system - or a position measuring system as 2nd measuring system with distance-coded reference marks for gantry axes. In this way the measuring system is referenced after traversal of a short path (e.g. 20 mm). The procedure for referencing the gantry axes is the same as that described for the normal incremental measuring system.

References:

Function Manual Basic Functions; Reference Point Travel (R1)

Absolute encoder

During the course of the synchronization compensatory motion, all axes in the gantry axis grouping traverse to the reference point value of the leading axis defined in the machine data:

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distancecoded system)

The absolute encoders and distanced-coded encoders of the leading axis will be set to the current actual position of the leading axis or to the reference point by the following machine data:

MD34330 $MA_REFP_STOP_AT_ABS_MARKER (Distancecoded linear measuring system without destination point)

Activation of axis compensations

Compensation functions can be activated for both the leading axis and the synchronized axes. Compensation values are applied separately for each individual gantry axis. These values must therefore be defined and entered for the leading axis and the synchronized axes during start-up.

The compensations do not become operative internally in the control until the axis is referenced or the gantry grouping synchronized. The following applies:

Compensation type	Takes effect when	PLC interface signal
Backlash compensation	Axis is referenced	"Referenced/Synchronized"
LEC	Axis is referenced	"Referenced/synchronized"
Sag compensation	Gantry grouping is synchronized	"Gantry grouping is synchronized"
Temperature compensation	Gantry grouping is synchronized	"Gantry grouping is synchronized"

If a movement by the synchronized axis (axes) is caused by an active compensation, a travel command is displayed for the synchronized axis (axes) independently of the leading axis.

Monitoring functions effective

Analogous to normal NC axes, the following monitoring functions do not take effect for gantry axes until the reference point is reached (IS "Referenced/Synchronized"):

- Working area limits
- Software limit switch
- Protection zones

The axial machine data values are used as monitoring limit values for the synchronized axes as well.

Multi-channel block search

The cross-channel block search in Program Test mode (SERUPRO "**Se**arch **R**un by **Pro**gram test") can be used to simulate the traversal of gantry axis groupings.

Note

Further information regarding multi-channel block search SERUPRO can be found under:
References:
Function Manual, Basic Functions; Mode Group, Channel, Program Mode (K1),
Section: Program test

3.4 Start-up of gantry axes

General

Owing to the forced coupling which is normally present between guide and synchronous gantry axes, the gantry grouping must be started up as if it were an axis unit. For this reason, the axial machine data for the guide and synchronous axes must always be defined and entered jointly.

If the synchronous axis is overloaded due to a weaker dynamic response than the guide axis, alarm 10656 is displayed.

Special points to be noted with regard to starting up gantry axes are described below.

Traversing direction

As part of the start-up procedure, a check must be made to ensure that the direction of rotation of the motor corresponds to the desired traversing direction of the axis. Correct with the following axial machine data:

MD32100 $MA_AX_MOTION_DIR (traversing direction)

Entering gantry trip limits

For the monitoring of the actual position values of the synchronous axis in relation to the actual position of the guide axis, the limit values for termination, as well as for the guide and synchronous axes, should be entered corresponding to the specifications of the machine manufacturer:

- MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)
- MD37130 $MA_GANTRY_POS_TOL_REF (gantry shutdown limit when referencing)

Note

The control must then be switched off and then on again because the gantry axis definition and the trip limit values only take effect after power ON.

Response to setpoint changes and disturbances

Since digital drives respond well to disturbances and setpoint changes, there is no need for a compensatory control between the gantry axes. However, the gantry axes can only operate in exact synchronism if the parameters for the control circuits of the guide and synchronous axes are set to the **same dynamic response value**.

To ensure the best possible synchronism, the guide axis and synchronous axis must be capable of the **same dynamic response to setpoint changes**. The axial control loops (position, speed and current controllers) should each be set to the **optimum** value so that disturbances can be eliminated as quickly and efficiently as possible. The **dynamic response adaptation** function in the setpoint branch is provided to allow differing dynamic responses of axes to be matched without loss of control quality.

Axial optimization

The following control parameters must be set to the optimum axial value for both the guide axis and the synchronous axis:

- MD32200 $MA_POSCTRL_GAIN (servo gain factor)
- MD32620 $MA_FFW_MODE (feedforward control parameter)
- MD32610 $MA_VELO_FFW_WEIGHT (feedforward control factor for acceleration/speed)
- MD32650 $MA_AX_INERTIA (inertia for torque feedforward control)
- MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)
- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

References

Extended Functions Function Manual; Compensations (K3)

Same settings

The following control parameters must be set to the same value for the guide axis and synchronous axis:

- MD33000 $MA_FIPO_TYPE (fine interpolator type)
- MD32400 $MA_AX_JERK_ENABLE (axial jerk limitation)
- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)
- MD32420 $MA_JOG_AND_POS_JERK_ENABLE (basic position of axial jerk limitation)
- MD32430 $MA_JOG_AND_POS_MAX_JERK (axial jerk)

References

Basic Functions Function Manual; Velocities, Setpoint / Actual Value Systems, Closed-Loop Control (G2)

Dynamics matching

The guide axis and the coupled synchronous axis must be capable of the same dynamic response to setpoint changes. The same dynamic response means: The following errors are equal in magnitude when the axes are operating at the same speed.

The dynamic response adaptation function in the setpoint branch makes it possible to obtain an excellent match in the response to setpoint changes between axes which have different dynamic characteristics (control loops). The difference in equivalent time constants between the dynamically "weakest" axis and the other axis in each case must be specified as the dynamic response adaptation time constant.

Example

When the speed feedforward control is active, the dynamic response is primarily determined by the equivalent time constant of the "slowest" speed control loop.

Guide axis

MD32810 $MA_EQUIV_SPEEDCTRL_TIME [n] = 5 ms (equivalent time constant speed control loop for feedforward control)

Synchronous axis

MD32810 $MA_EQUIV_SPEEDCTRL_TIME [n] = 3 ms

- Time constant of dynamic response adaptation for synchronous axis:
  MD32910 $MA_DYN_MATCH_TIME [n] = 5 ms - 3 ms = 2 ms (time constant of dynamic response adaptation)

The dynamic response adaptation must be activated axially with the machine data:

MD32900 $MA_DYN_MATCH_ENABLE (dynamic response adaptation)

Check of dynamic response adaptation:

The following errors of the guide and synchronous axes must be equal in magnitude when the axes are operating at the same speed!

For the purpose of fine tuning, it may be necessary to adjust servo gain factors or feedforward control parameters slightly to achieve an optimum result.

Referencing gantry axes

The positions of the guide and synchronous axes reference points must first be set to almost identical values.

To ensure that the synchronization compensatory motion of the gantry axes is not started automatically, the gantry warning limit must be set to 0 at the first start-up before referencing:

MD37100 $MA_GANTRY_POS_TOL (gantry axis definition)

This will prevent a warning message being output during traversing motion.

In cases where an excessively high additional torque is acting on the drives due to misalignment between the guide and synchronous axes, the gantry grouping must be aligned before the axes are traversed. After this, the gantry axes must be referenced. For further details, see:

- Section "Referencing and synchronization of gantry axes (Page 194)"
- **References**: Basic Functions Function Manual; Reference Point Approach (R1)

After the guide and synchronous axes have been referenced, the difference between them must be determined by comparing the actual position value (HMI: operating area "Diagnostics" > "Service axes") and taken into account as the reference point offset:

- MD34080 $MA_REFP_MOVE_DIST (reference point distance)
- MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset / absolute offset)

The differences in distance between the zero mark and reference point must also be calculated for each gantry axis and adjusted. They are to be customized, via the following machine data, in such a way that the actual position values of the guide and synchronous axes are identical after execution of the compensatory motion:

- MD34080 $MA_REFP_MOVE_DIST (reference point distance)
- MD34090 $MA_REFP_MOVE_DIST_CORR (reference point offset / absolute offset)

Synchronizing gantry axes

The gantry synchronization is activated via the NC/PLC interface signal (see Section "Referencing and synchronization of gantry axes (Page 192)"):

DB31, ... DBX29.4 = 1 (start synchronization of gantry)

The completion of the synchronization is displayed via the NC/PLC interface signal:

DB31, ... DBX101.5 == 1 (gantry grouping is synchronous)

Once the axes have been synchronized, check that the dimensional offset between the guide and the synchronous axes is 0. If required, make corrections in the machine data mentioned above.

Input of gantry warning limit

Once the reference point values for the guide and synchronous axes have been optimized so that the gantry axes are perfectly aligned with one another after synchronization, the warning limit values for all axes must be entered in the following machine data:

MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

To do this, the value must be increased incrementally until it is just below the alarm (limit exceeded) response limit. It is particularly important to check the acceleration phases.

This limit value also determines the position deviation value at which gantry synchronization is automatically started in the control.

Calculating and activating compensations

In cases where the gantry axes require compensation (backlash, sag, temperature or leadscrew error), the compensation values for the guide axis **and** the synchronous axis must be calculated and entered in the appropriate parameters or tables.

References

Extended Functions Function Manual; Compensations (K3)

Function generator / measuring function

The activation of the function generator and measuring function for a synchronous axis is aborted with an error message. If an activation of the synchronous axis is absolutely necessary, e.g. to measure the machine, the guide and the synchronous axes must be temporarily interchanged.

Special cases

If **individual** axes have to be activated, the gantry groups must be temporarily canceled. As the second axis no longer travels in synchronism with the first axis, the activated axis must not be allowed to traverse beyond the positional tolerance.

If the gantry grouping is canceled, the following points must be noted:

- Always activate the traversing range limits and set them to the lowest possible values (position tolerance).
- Synchronize the gantry grouping first if possible and then execute a POWER-ON-RESET **without** referencing the axes again. This ensures that the traversing range limits always refer to the same position (i.e. that which was valid on power ON).
- Avoid using the step-change function. Position step changes are only permissible if they stay within the permitted tolerance.
- Always use an offset of 0 for the function generator and measuring function in contrast to the recommendations for normal axes.
- Set the amplitudes for function generator and measuring function to such low values that the activated axis traverses a shorter distance than the position tolerance allows. Always activate the traversing range limits as a check (see above).

References

Drive Functions Function Manual; Speed Control Loop (DD2)

Start-up support for gantry groupings

The start-up functions of the function generator and measuring are parameterized via the PI service. The traversing movement starts for all parameterized axes with NC start in JOG mode.

A window is displayed in the "Measuring function and function generator in gantry grouping" user interface. Two amplitude values, each with an offset and bandwidth, must be entered in this window. The first amplitude value applies to the measuring axis and the second to the other coupled axes.

3.5 Parameter assignment: Response to faults

Pulse suppression

The behavior of the gantry grouping with regard to faults that trigger pulse suppression can be set with the following axis-specific machine data:

MD30455 $MA_MISC_FUNCTION_MASK, bit 9 = <value>

<value>	Meaning
0	When a fault occurs that triggers the pulse suppression (e.g. measuring-circuit fault), the pulses in all other axes of the gantry grouping will also be suppressed. Result: Coast down of all axes of the gantry grouping.
1	When a fault occurs that triggers the pulse suppression (e.g. measuring-circuit fault), the pulses for only this axis will be suppressed. The other axes of the gantry grouping will be stopped. The pulses for these axes are also then suppressed. To ensure that the pulses these axes are not suppressed in advance, especially for axes to be held at the standstill position (vertical axes), the following drive parameters must be taken into account: • p1135[0...n] OFF3 ramp-down time • p1217 holding brake application time • p1227 zero speed detection monitoring time • p1228 pulse suppression, delay time For a detailed description of drive parameters, refer to: **References**: SINAMICS S120/S150 Parameter Manual

3.6 PLC interface signals for gantry axes

Special IS for gantry axes

The special NC/PLC interface signals of the coupled gantry axes are taken via the axial NC/PLC interface of the leading or synchronized axes. The table below shows all special gantry NC/PLC interface signals along with their codes and indicates whether the IS is evaluated on the leading axis or the synchronized axis.

NC/PLC interface signal	Direction of transfer	DB31, ... DBX...	Leading axis	Synchronous axis
Start gantry synchronization	PLC → NC	29.4	X	
No automatic synchronization	PLC → NC	29.5	X	
Gantry axis	NC → PLC	101.7	1	1
Gantry leading axis	NC → PLC	101.6	1	0
Gantry grouping is synchronized	NC → PLC	101.5	X	
Gantry synchronization ready to start	NC → PLC	101.4	X	
Gantry warning limit exceeded	NC → PLC	101.3		X
Gantry trip limit exceeded	NC → PLC	101.2		X

Effect of axial interface signals on gantry axes

a) Axial interface signals from the PLC to the axis (PLC → NC)

The axial interface signals from the PLC to the axis are always referred to all gantry axes in the grouping. In this case, all gantry axes (leading and synchronized axis) have equal priority.

For example, all axes in the gantry groupings will be simultaneously shut down when the following interface signal is set to "0" from the leading axis:

DB31, ... DBX2.1 (servo enable)

The following table shows the effect of individual interface signals (from PLC to axis) on gantry axes:

NC/PLC interface signal	DB31, ... DBX ...	Effect on	
		Leading axis	**Synchronous axis**
Axis/spindle disable	1.3	On all axes in gantry grouping	No effect
Position measuring system 1/2	1.5 and 1.6	Axial [1)]	Axial [1)]
Controller enable	2.1	On all axes in gantry grouping [2)]	
Delete distance to go (axial)	2.2	Axial	No effect
Clamping in progress	2.3	Axial	Axial
Reference point value 1-4	2.4 - 2.7	Axial	Axial
Feed stop	4.3	On all axes in gantry grouping	
Hardware limit switch plus/minus	12.0 and 12.1	Axial alarm: Brake request on all axes in gantry grouping	
2nd software limit switch plus/minus	12.2 and 12.3	Axial	Axial
Ramp-function generator fast stop (RFGFS)	20.1	On all axes in gantry grouping	
Select drive parameter set	21.0 - 21.2	Axial	Axial
Enable Pulses	21.7	Axial	Axial

[1)] DB31, ... DBX1.5 and 1.6 (position measuring system 1/2)

The switchover between position measuring systems 1 and 2 applies individually for each gantry axis. However, deactivation of both position measuring systems (known as the parking position) applies as a common signal for all gantry axes.

[2)] DB31, ... DBX2.1 (controller enable)

If the servo enable signal on one gantry axis is canceled, all axes in the gantry grouping are shut down simultaneously. The method by which shutdown is implemented (e.g. with fast stop) is identical for all gantry axes.

Either the "Follow-up" state (IS of one gantry axis = 1) or the "Stop" state (IS of all gantry axes = 0) is activated for all gantry axes, depending on interface signal:

DB31, ... DBX1.4 (follow-up mode)

b) Axial interface signals from the axis to the PLC (NC → PLC)

Each of the axial, axis-to-PLC interface signals for the synchronized axis and the leading axis is always set on an axis-specific basis and output to the PLC.

Example:

DB31, ... DBX60.4 resp. 60.5 (referenced/synchronized 1/2).

Exception:

When the leading axis is traversed, the interface signal will also be set for the synchronizing axis:

DB31, ... DBX64.6 and 64.7 (traverse command plus resp. minus)

3.7 Miscellaneous points regarding gantry axes

Manual traversing

It is not possible to traverse a synchronous axis directly by hand in JOG mode. Traverse commands entered via the traversing keys of the synchronous axis are ignored internally in the control. Rotation of the handwheel for the synchronous axis has no effect either.

Handwheel override

An overriding motion by means of the handwheel can only be applied to the guide axis in coupled axis mode. In this case, the synchronous axes traverse in synchronism with the guide axis.

DRF offset

A DRF offset can only be applied to the guide axis. In this case, the synchronous axes traverse in synchronism with the guide axis.

Programming in part program

Only the guide axis of a gantry axis grouping may be programmed in the part program. An alarm is generated while programming a synchronous axis, even when a gantry axis grouping is released (MD37140 $MA_GANTRY_BREAK_UP = 1).

PLC or command axes

Only the guide axis of the gantry grouping can be traversed by the PLC using FC 18 or as a command axis by means of synchronized actions.

References:

- Function Manual, Basic Functions, Basic PLC Program (P3)
- Function Manual, Synchronized Actions

PRESET

The PRESET function can only be applied to the guide axis. All axes in the gantry grouping are reevaluated internally in the control when PRESET is activated. The gantry axes then lose their reference and synchronization:

DB31, ... DBX101.5 (gantry grouping is synchronized) = 0

Channel assignment of the gantry axes

Please ensure that for a gantry grouping whose guide axis is known in several channels, its synchronous axes in these channels are also known. If this is not the case, Alarm 10651 is output with reason 60XX (XX is the objectionable gantry grouping).

Axis interchange

All axes in the gantry grouping are released automatically in response to a RELEASE command (guide axis).

An axis interchange of the guide axis of a closed gantry grouping is only possible, if all axes of the grouping are known in the channel in which they are to be transferred, otherwise alarm 10658 is signaled.

No automatic axis interchange and no automatic adjustment of the gantry axis conditions are undertaken while trying to reconnect a gantry grouping is released with MD37140 $MA_GANTRY_BREAK_UP = 1. The user is responsible for this. A check of the axis conditions is conducted after the break up and if necessary, a corresponding alarm 10658 is output.

Note

If a gantry grouping is to be closed again, the user must ensure that all axes of the grouping are in a channel with a corresponding axis condition.

Default for RESET

In an active gantry grouping, the following machine data parameterization is ignored for the synchronous axes:

MD30450 $MA_IS_CONCURRENT_POS_AX = 1 (Reset default: neutral axis/channel axis)

The state of the guide axis is assumed. The user is informed about the inappropriate configuration with display alarm 4300.

Display data

The position actual value display shows the actual values of both the guide axis and the synchronous axes. The same applies to the service display values in the "Diagnosis" operating area.

Software limit switch

The software limit switch monitor is processed for the guide axis only. If the guide axis crosses the limit switch, all axes in the gantry grouping are braked to a standstill.

Differences in comparison with the "Coupled motion" function

The main differences between the "gantry axes" and "coupled motion" functions are listed below:

- The axis coupling between the gantry axes must always be active. Separation of the axis coupling via part program is therefore not possible for gantry axes. In contrast, the coupled axis grouping can be separated by means of the part program and the axes then traversed individually.
- In the "Gantry Axes" function, the difference of the actual position values from the guide axis and synchronous axis is monitored continuously and the traversing motion is shut down if there are impermissible deviations. There is no monitoring for the "Coupled motion" function.
- Gantry axes must remain coupled even during referencing. For this reason, special procedures are applied for the reference point approach of gantry axes. In contrast, coupled-motion axes are referenced as individual axes.
- For the gantry axes to traverse without mechanical offset, the synchronous axes must be set like the guide axes from the control dynamics perspective. In contrast, the "coupled motion" function permits axes with different dynamic control response characteristics to be coupled.

References:
Function Manual, Special Functions, Coupled Motion (M3)

Block search with active coupling

Note

For an active coupling, it is recommended to only use block search type 5, "Block search via program test" (SERUPRO) for a block search.

3.8 Examples

3.8.1 Creating a gantry grouping

Introduction

Setting up gantry grouping, referencing its axes, aligning possible offsets - and finally, synchronizing the axes involved, are complicated procedures. The individual steps involved in the process are explained below by an example constellation.

Constellation

Machine axis 1 = gantry leading axis, incremental measuring system

Machine axis 3 = gantry synchronized axis, incremental measuring system

Machine data

The following machine data describes the original values at the beginning of the procedure. Individual settings must be corrected or added later according to the information below.

Gantry machine data

Axis 1

MD37100 $MA_GANTRY_AXIS_TYPE = 1 (gantry axis definition)

MD37110 $MA_GANTRY_POS_TOL_WARNING =0 (gantry warning limit)

MD37120 $MA_GANTRY_POS_TOL_ERROR = e.g. 1 (gantry trip limit)

MD37130 $MA_GANTRY_POS_TOL_REF = e.g. 100 mm (max. misalignment) (gantry shutdown limit when referencing)

MD37140 $MA_GANTRY_BREAK_UP = 0 (invalidate gantry axis grouping)

Axis 3

MD37100 $MA_GANTRY_AXIS_TYPE= 11

MD37110 $MA_GANTRY_POS_TOL_WARNING = 0

MD37120 $MA_GANTRY_POS_TOL_ERROR = e.g. 1 mm

MD37130 $MA_GANTRY_POS_TOL_REF = e.g. 100 mm (max. misalignment)

MD37140 $MA_GANTRY_BREAK_UP = 0

Reference point machine data (for first encoder each)

Axis 1

MD34000 $MA_REFP_CAM_IS_ACTIVE = TRUE

MD34010 $MA_REFP_CAM_DIR_IS_MINUS = e.g. FALSE

MD34020 $MA_REFP_VELO_SEARCH_CAM =

MD34030 $MA_REFP_MAX_CAM_DIST = corresponds to max. distance traversed

MD34040 $MA_REFP_VELO_SEARCH_MARKER =

MD34050 $MA_REFP_SEARCH_MARKER_REVERSE = e.g. FALSE

MD34060 $MA_REFP_MAX_MARKER_DIST = Difference betw. cam edge and 0 mark

MD34070 $MA_REFP_VELO_POS =

MD34080 $MA_REFP_MOVE_DIST = 0

MD34090 $MA_REFP_MOVE_DIST_CORR = 0

MD34092 $MA_REFP_CAM_SHIFT = 0

MD34100 $MA_REFP_SET_POS = 0

MD34200 $MA_ENC_REFP_MODE = 1

The reference point machine data (for the first encoder) of axis 3 must be specified analogously.

3.8.2 Setting the NC-PLC interface

Introduction

An automatic synchronization process when referencing axes must be disabled initially to prevent damaging/destroying grouping axes that are misaligned.

Disabling of automatic synchronization

The PLC user program sets the following for the axis data block of axis 1:

DB31, ... DBX29.4 = 0 (do not start gantry synchronization)

DB31, ... DBX29.5 = 1 (no automatic synchronization process)

The NC sets the following as a confirmation in the axis block of axis 1:

DB31, ... DBB101.4 = 0 (synchronization process not ready to start)

DB31, ... DBB101.6 = 1 (leading axis LA)

DB31, ... DBB101.7 = 1 (gantry axis)

The PLC user program sets for the axis data block of axis 3:

DB31, ... DBX29.4 = 0 (do not start gantry synchronization)

The NC sets the following as a confirmation in the axis block of axis 3:

DB31, ... DBB101.4 = 0 (synchronization process not ready to start)

DB31, ... DBB101.6 = 0 (synchronized axis GA)

DB31, ... DBB101.7 = 1 (gantry axis)

3.8.3 Commencing start-up

Referencing

The following steps must be taken:

- Select "REF" operating mode
- Start referencing for axis 1 (master axis)
- Wait until message "10654 Channel 1 Waiting for synchronization start" appears.

At this point in time, the NC has prepared axis 1 for synchronization and registers this at the interface signal:

DB31 DB31, ... DBB101.4 = 1 (synchronization process ready to start)

DB31, ... DBB101.6 = 1 (leading axis LA)

DB31, ... DBB101.7 = 1 (gantry axis)

In addition, the following steps must be taken:

- RESET
- Read off values in machine coordinate system:
  e. g.
  X = 0.941
  Y = 0.000
  XF = 0.000
- Enter the X value of master axis 1 with inverted sign in the machine data of slave axis 3:
  MD34090 $MA_REFP_MOVE_DIST_CORR = - 0.941 (reference point offset/absolute offset)

Note

This machine data is effective after power ON. To avoid having to perform a power ON now, the value can also be entered in the following machine data:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)

This machine data is then valid after a RESET.

- Start referencing again for axis 1 (master axis) with the modified machine data
- Wait until message "10654 Channel 1 Waiting for synchronization start" appears
- At this point in time, the NC has prepared axis 1 for synchronization and registers this at the interface signal:
  DB31 DB31, ... DBB101.4 = 1 (synchronization process ready to start)
  DB31, ... DBB101.6 = 1 (leading axis LA)
  DB31, ... DBB101.7 = 1 (gantry axis)
- Examine actual positions of machine. Case A or B might apply:

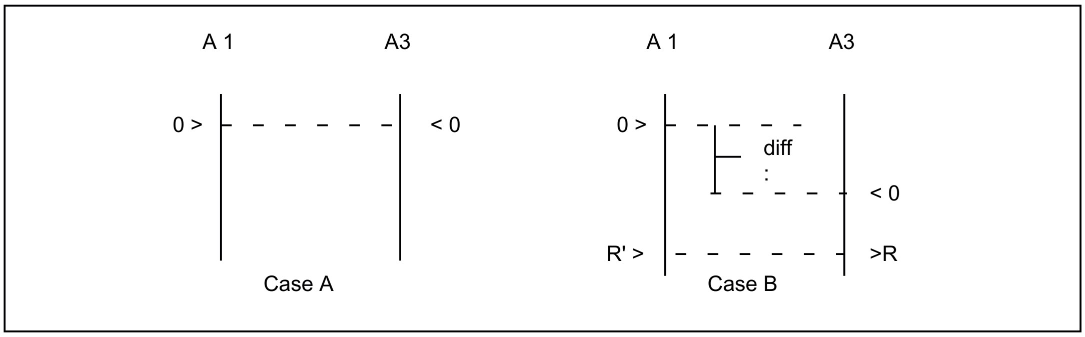


Figure 3-3 Possible results after referencing axis 1 (master axis)

If Case A applies, the synchronization process can be started immediately (see "start synchronization process" step). If Case B applies, the offset "diff" must be calculated and taken into account:

- Measuring of diff
- By using two appropriate, right-angled reference points R' and R" in the machine bed (right in picture), the difference in position in JOG can be traversed. The diff offset can then be read as the difference in the position display. The diff offset must be entered in the machine data of axis 3 (synchronized axis):
  MD34100 $MA_REFP_SET_POS
  Continue with 1st step (see above).
- Start gantry synchronization. PLC sets:
  DB31, ... DBX29.4 = 1 (start synchronization of gantry)

3.8.4 Setting warning and trip limits

If the gantry grouping is set and synchronized, the following machine data must then be set:

MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)

Proceed as follows

- Set the machine data for all axes with a large value to begin with:
  MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)
- Set a very small value in the machine data:
  MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)
  When you put a heavy, dynamic strain on the axes, the self-canceling alarm "10652 channel %1 axis %2 gantry warning limit exceeded" will be output repeatedly.
- Now increase the following machine data:
  MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)
  Repeat this until the alarm no longer appears. The interface indicates the status:
  DB31, ... DBX101.2 = 0 (gantry trip limit not exceeded)
  DB31, ... DBX101.3 = 0 (gantry warning limit not exceeded)
  DB31, ... DBX101.4 = 0 (gantry synchronization run not ready to start)
  DB31, ... DBX101.5 = 1 (gantry grouping is synchronized)
  DB31, ... DBX101.6 = 1 (gantry guide axis)
  DB31, ... DBX101.7 = 1 (gantry axis)

If the monitoring still only triggers very sporadically, it is possible to program a edge memory bit in the PLC user program.

- Enter the value calculated for the warning limit + a small safety provision in the following machine data:
  MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)

Error limit values

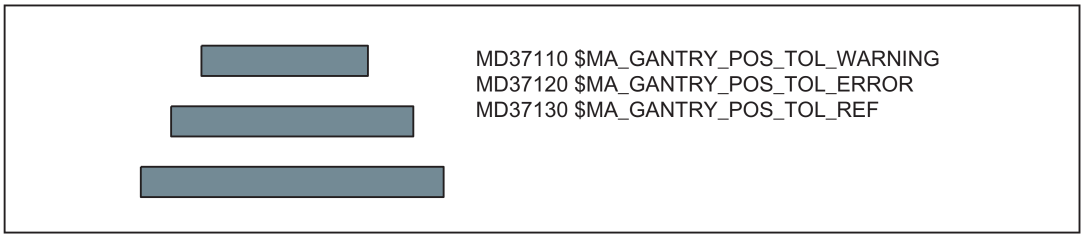
Values are entered in the following machine data:

MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)

MD37130 $MA_GANTRY_POS_TOL_REF (gantry shutdown limit when referencing)

These should have the following scales of magnitude at the end of the customizing process:


MD37110 $MA_GANTRY_POS_TOL_WARNING
MD37120 $MA_GANTRY_POS_TOL_ERROR
MD37130 $MA_GANTRY_POS_TOL_REF

Note

The same procedure must be followed when starting up a gantry grouping in which the coupled axes are driven by **linear motors** and associated measuring systems.

The error limits entered into machine data MD37110 and MD37120 are considered as additional tolerance values of the actual-value difference of the master and following axis if the interface bit "Gantry is synchronous" is not present (e.g. to be resynchronized after canceling alarms without gantry).

3.9 Data lists

3.9.1 Machine data

3.9.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
32200	POSCTRL_GAIN	K_V factor (servo gain)
32400	AX_JERK_ENABLE	Axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32420	JOG_AND_POS_JERK_ENABLE	Initial setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axial jerk
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforward control
32620	FFW_MODE	Feedforward control type
32650	AX_INERTIA	Moment of inertia for torque feedforward control

Number	Identifier: $MA_	Description
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant, current control loop for feed-forward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant, speed control loop for feed forward control
32900	DYN_MATCH_ENABLE	Dynamic adjustment
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
33000	FIPO_TYPE	Fine interpolator type
34040	REFP_VELO_SEARCH_MARKER	Shutdown velocity
34070	REFP_VELO_POS	Velocity when approaching the reference point
34080	REFP_MOVE_DIST	Reference point approach distance
34090	REFP_MOVE_DIST_CORR	Reference point offset
34100	REFP_SET_POS	Reference point value
34110	REFP_CYCLE_NR	Axis sequence for channel-specific referencing
34330	REFP_STOP_AT_ABS_MARKER	Distance-coded linear measuring system without destination point
36012	STOP_LIMIT_FACTOR	Factor, exact stop coarse/fine - and zero speed
36030	STANDSTILL_POS_TOL	Standstill tolerance
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
37100	GANTRY_AXIS_TYPE	Gantry axis definition
37110	GANTRY_POS_TOL_WARNING	Gantry warning limit
37120	GANTRY_POS_TOL_ERROR	Gantry shutdown limit
37130	GANTRY_POS_TOL_REF	Gantry shutdown limit when referencing
37140	GANTRY_BREAK_UP	Break up gantry axis grouping

3.9.2 Signals

3.9.2.1 Signals from mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Active machine function REF	DB11 DBX5.2	DB3100.DBX1.2

3.9.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referencing active	DB21, ... DBX33.0	DB330x.DBX1.0

3.9.2.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Start gantry synchronization	DB31, ... DBX29.4	DB380x.DBX5005.4
No automatic synchronization	DB31, ... DBX29.5	DB380x.DBX5005.5

3.9.2.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1, referenced/synchronized 2	DB31, ... DBX60.4/5	DB390x.DBX0.4/5
Gantry trip limit exceeded	DB31, ... DBX101.2	DB390x.DBX5005.2
Gantry warning limit exceeded	DB31, ... DBX101.3	DB390x.DBX5005.3
Gantry synchronization ready to start	DB31, ... DBX101.4	DB390x.DBX5005.4
Gantry grouping is synchronous	DB31, ... DBX101.5	DB390x.DBX5005.5
Gantry guide axis	DB31, ... DBX101.6	DB390x.DBX5005.6
Gantry axis	DB31, ... DBX101.7	DB390x.DBX5005.7

K6: Contour tunnel monitoring 4

4.1 Brief description

4.1.1 Contour tunnel monitoring - 840D sl only

Function

The absolute movement of the tool tip in space is monitored. The function operates channel specific.

Model

A round tunnel with a definable diameter is defined around the programmed path of a machining operation. Axis movements are stopped as an option if the path deviation of the tool tip is greater than the defined tunnel as the result of axis errors.

Response

In the event of a recognized deviation, the system reacts as quick as possible. However, at least one interpolator clock cycle will pass, before one of the following reactions will occur:

- An alarm is triggered when the tunnel is violated and the axes continue to traverse.
- Violation of the tunnel triggers an alarm and the axis movements are decelerated.

Braking methods

If the monitoring tunnel is violated, one of the following methods can be used to decelerate:

- Deceleration ramp
- Speed setpoint zero and follow-up mode

Use

The function can be used for 2D and 3D paths. For 2D the monitoring surface is defined by parallel lines to the programmed path. The monitoring area is defined by 2 or 3 geometry axis.

Monitoring of synchronized axes, positioning axes, etc. that are not geometry axes is performed directly on the machine axis plane with the "Contour monitoring".

Example

The following figure is a diagram of the monitoring area shown by way of a simple example.

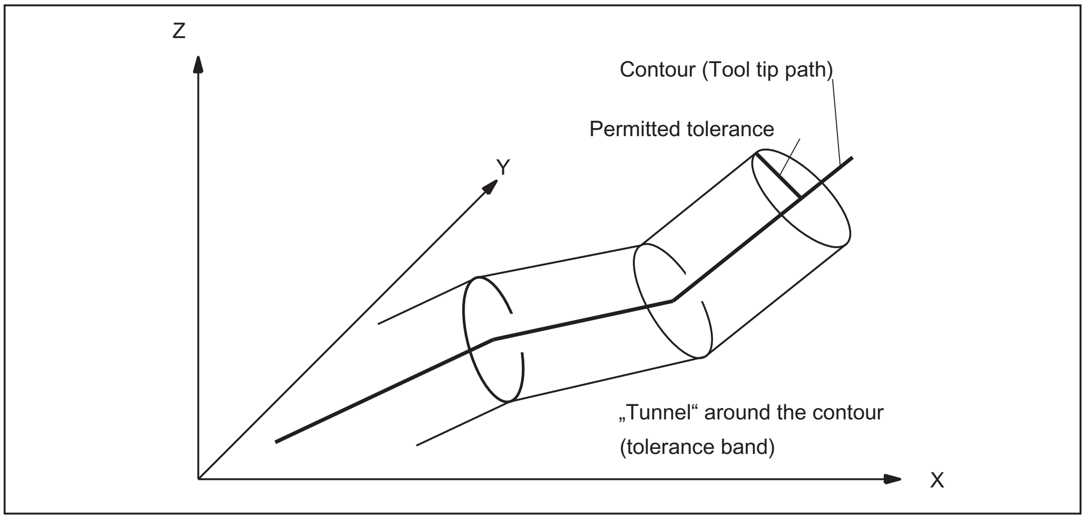


Figure 4-1 Position of the contour tunnel around the programmed path

As long as the calculated actual position of the tool tip remains inside the sketched tunnel, motion continues in the normal way. If the calculated actual position violates the tunnel, an alarm is triggered (in the default setting) and the axes are stopped by "Ramp Stop". This response to the violation of the tunnel can be disabled (alarm triggered but movement continued) or intensified (rapid stop) by means of a machine data setting.

Analysis

The calculated distance between the programmed path and the actual values can be routed to an analog output to analyze the progression of the contour errors during normal operation (quality control).

4.1.2 Programmable contour accuracy

Function

As an alternative to the function described in "Contour tunnel monitoring", i.e. monitoring of the machining accuracy and stopping machining if excessive deviations occur, another function is offered. With this function, the selected accuracy is always achieved with the path velocity being reduced if necessary. Detailed information about this function can be found under the subject "Programmable contour accuracy".

4.2 Contour tunnel monitoring - 840D sl only

Aim of the monitoring function

The aim of the monitoring function is to stop the movement of the axes if axis deviation causes the distance between the tool tip (actual value) and the programmed path (setpoint) to exceed a defined value (tunnel radius).

Tunnel size

The radius of the contour tunnel being monitored around the programmed path must be defined to implement the monitoring function:

MD21050 $MC_CONTOUR_TUNNEL_TOL (Response threshold for Contour tunnel monitoring)

If the machine data is set to 0.0, monitoring is not performed. The value of the machine data is transferred to the control for new configurations.

Parameterizable deceleration behavior

The deceleration behavior for the monitoring response can be set via the following machine data:

MD21060 $MC_CONTOUR_TUNNEL_REACTION (Reaction upon response of contour tunnel monitoring)

Value	Meaning
0	Display alarm and continue machining
1	Deceleration according to the deceleration ramps (default setting)
2	Rapid stop (speed setpoint = 0)

Encoder switchover

Switching between two encoder systems usually causes a sudden change in the actual position of the tool tip. This change resulting from encoder switchover must not be so large as to cause the tool tip to violate the monitoring tunnel. The radius defined in MD21050 must be higher than the allowed tolerance on actual value switchover:

MD36500 $MA_ENC_CHANGE_TOL (Max. tolerance for position actual value switchover)

Activation

The monitoring will only become active if the following conditions are met:

- MD21050 is higher than 0.0.
- At least two geometry axes have been defined.

Shutting down

Monitoring can be stopped by enabling the machine data setting:

MD21050 = 0.0.

Analysis output

The values of deviation of the actual value of the tool tip from the programmed path can – for analysis purposes – be output on a fast analog output (accuracy monitoring).

The assignment of an analog output to output the contour error is programmed in machine data:

MD21070 $MC_CONTOUR_ASSIGN_FASTOUT

Value	Meaning
0	No output (default setting)
1	Output to output 1
2	Output to output 2
…	…
8	Output to output 8

Scale

The tunnel radius set in MD21050 corresponds to a voltage of 10 V at the output.

4.3 Programmable contour accuracy

Function

The function "Programmable contour accuracy" limits the contour errors caused by control behavior and jerk filter to a specified value by reducing the path velocity on curved contours by the necessary amount. It allows the user to set a compromise between accuracy and productivity of a machining.

Note

The "LookAhead" function ensures that the velocity necessary for maintaining the required contour accuracy is not exceeded at any point along the path.

Configuration

The mode of operation and parameterization of the function is determined by the machine data:

MD20470 $MC_CPREC_WITH_FFW (programmable contour accuracy)

Value	Meaning
0	The "Programmable contour accuracy" function has no effect when feedforward control is also active.
1	The "Programmable contour accuracy" function also acts for feedforward control. With active feedforward control, the reduction of the path velocity is calculated on the basis of the effective K_V factor with feedforward control.
2	Like 1, the function, however, is parameterized with MD32415 $MA_EQUIV_CPREC_TIME (time constant for the programmable contour accuracy). The jerk filter is correctly taken into account. The SD42450 $SC_CONTPREC setting data determines the permissible contour errors (see "Parameterization").
3	Like for 2, but any contour accuracy programmed with CTOL has priority over SD42450 $SC_CONTPREC. The jerk filter is correctly taken into account. The programmed CTOL contour tolerance determines the permissible contour errors (see "Parameterization"). $SC_CONTPREC is only relevant if CTOL was not programmed.

For the MD20470 = 2 or 3 functional versions, the control assumes that there is a jerk filter time constant (MD32410 $MA_AX_JERK_TIME) for which the setting of the control section with feedforward control generates a negligible contour error. This value must be entered in the machine data MD32415 $MA_EQUIV_CPREC_TIME (see "Parameterization").

To calculate the contour error based on the set jerk filter type (MD32402 $MA_AX_JERK_MODE), the following value is used:

- For active feedforward control, the difference:
  MD32410 $MA_AX_JERK_TIME - MD32415 $MA_$MA_EQUIV_CPREC_TIME
- Without feedforward control, the complete value from MD32410 $MA_AX_JERK_TIME

This procedure allows the commissioning engineer to first change from an initially precise, but possibly excessively hard, setting by increasing the jerk filter time constants to a softer setting with a controlled loss of accuracy.

Supplementary conditions:

- The function does not use the "belt stop" jerk filter type (MD32412 $MA_AX_JERK_MODE = 3).
- The MD20470 = 2 or 3 functional versions are primarily intended for use with feedforward control. If one of the two functional versions is active for switched off feedforward control, a contour error that results from the K_v factor is added. This reduces the path velocity significantly faster.

Note

The MD20470 = 0 or 1 functional versions are no longer recommended. They only provide compatibility with older software versions.

Parameterization

Contour accuracy

The maximum contour error for the path of the geometry axes on curved contours is determined by:

- For MD20470 $MC_CPREC_WITH_FFW = 2 with the setting data:
  SD42450 $SC_CONTPREC (contour accuracy)
- For MD20470 $MC_CPREC_WITH_FFW = 3 with the contour tolerance programmed with CTOL.

The smaller the value and the lower the K_v factor of the geometry axes, the greater the path feedrate on curved contours is lowered.

Minimum path feedrate

The user can use the following setting data to specify a minimum path feedrate for the "Programmable contour accuracy" function:

SD42460 $SC_MINFEED (minimum path feed with CPRECON)

The feedrate will not limited below this value, unless a lower F value has been programmed or the dynamic limitations of the axes force a lower path velocity.

Time constant for the programmable contour accuracy

The equivalent time constant for the MD20470 = 2 or 3 functional versions (see "Configuration") is entered in the machine data:

MD32415 $MA_EQUIV_CPREC_TIME (time constant for the programmable contour accuracy)

MD32415 must contain that jerk filter time constant (MD32410 $MA_AX_JERK_TIME) for which the contour error for active feedforward control is negligibly small.

Programming

The "programmable contour accuracy" can be activated and deactivated in the part program with the CPRECON and CPRECOF modal G commands.

Example:

Program code	**Comment**
`N10 G0 X0 Y0`	
`N20 CPRECON`	`; Activate the "programmable contour accuracy".`
`N30 G1 G64 X100 F10000`	`; Machining with 10 m/min in the continuous-path mode.`
`N40 G3 Y20 J10`	`; Automatic feed limitation in circular block.`
`N50 G1 X0`	`; Feedrate again without limitation (10 m/min).`
`...`	
`N100 CPRECOF`	`; Deactivate the "programmable contour accuracy".`
`N110 G0 ...`	

The two CPRECON and CPRECOF modal G commands form G group 39 (programmable contour accuracy).

Behavior for part program start and after reset / part program end

For part program start and after reset / part program end, the configured control initial setting acts for the G group 39:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET / TP End)

MD20112 $MC_START_MODE_MASK (definition of the basic setting of the control after part program start)

Supplementary conditions

Positioning axes

The function considers only the geometry axes of the path. It does not have any effect of the velocities for the positioning axes.

References

Information about MD32402 $MA_AX_JERK_MODE (filter type for axial jerk limitation) and MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter), see:

Function Manual, Basic Functions; Acceleration (B2), Section: "Functions" > "Jerk filter (axis-specific)"

Information about CTOL, see:

Function Manual, Basic Functions; Continuous-Path Mode, Exact Stop, Look Ahead (B1), Section: "Contour/orientation tolerance"

4.4 Constraints

Availability of the "Contour tunnel monitoring" function

The function is an option ("Contour monitoring with tunnel function"), which must be assigned to the hardware via the license management.

Coupled motion

If coupled motion between two geometry axes is programmed with contour tunnel monitoring, this always results in activation of the contour tunnel monitoring. In this case, the contour tunnel monitoring must be switched off before programming the coupled motion:

MD21050 $MC_CONTOUR_TUNNEL_TOL = 0.0

4.5 Data lists

4.5.1 Machine data

4.5.1.1 Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Determination of basic control settings after Reset / TP End
20112	START_MODE_MASK	Definition of the initial control settings after part program start
20470	CPREC_WITH_FFW	Programmable contour accuracy
21050	CONTOUR_TUNNEL_TOL	Response threshold for contour tunnel monitoring
21060	CONTOUR_TUNNEL_REACTION	Reaction to response of contour tunnel monitoring
21070	CONTOUR_ASSIGN_FASTOUT	Assignment of an analog output for output of the contour error

4.5.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32402	AX_JERK_MODE	Filter type for axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32415	EQUIV_CPREC_TIME	Time constant for the programmable contour accuracy
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover

4.5.2 Setting data

4.5.2.1 Channelspecific setting data

Number	Identifier: $SC_	Description
42450	CONTPREC	Contour accuracy
42460	MINFEED	Minimum path feed for CPRECON

K7: Kinematic chain 5

5.1 Function description

5.1.1 Characteristics

This section describes how the kinematic structure of a machine can be mapped using a kinematic chain and parameterized in the control using system variables for NC functions such as "collision avoidance" or "kinematic transformation".

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.

For functions such as "collision avoidance" which also require a description of the machine geometry, see Chapter "K8: Geometric machine modeling (Page 263)".

Note

Graphical editor

As an alternative to writing the system variables in a part program, the machine can be modeled from the SINUMERIK Operate user interface:

Operating area: "Commissioning" > "NC" > "Machine model"

Changes to the machine model

The direct changes to the machine model at the system variables only become visible at the user interface after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

Changes to the machine model made at the user interface are immediately accepted in the system variables of the NC. The changes only become active after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

Kinematic structure

The kinematic structure of a machine comprises the following:

- Number and type of the machine axes: Linear or rotary axes
- Arrangement of the machine axes: Position and orientation
- Dependencies of the machine axes to one another: Which machine axis traverses with which other machine axis.

Kinematic chain

A kinematic structure of a machine is described using a kinematic chain with the following properties:

- A kinematic chain consists of an arbitrary number of interconnected elements.
- Parallel subchains can branch off from a kinematic chain.
- Only one active kinematic chain is available in the control.
- The active kinematic chain starts with the root element.
- Parameterized elements or subchains that are not connected to the root element or whose connection points toward the root element, are not part of the currently active kinematic chain.
- A kinematic chain is defined in the spatially fixed coordinates of the world coordinate system.

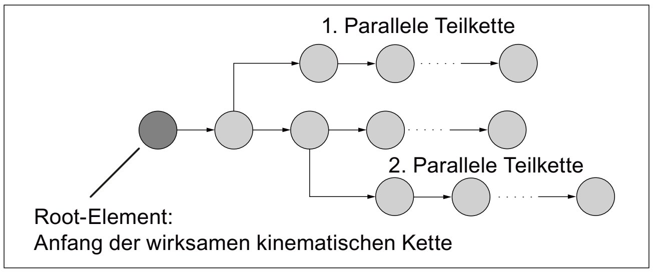


Figure 5-1 Example of a kinematic chain

Element

An element of a kinematic chain basically describes the transformation with which the local coordinate system of the previous element is mapped in the local coordinate system of the current element:

$K_{n-1} \Rightarrow T_n \Rightarrow K_n$

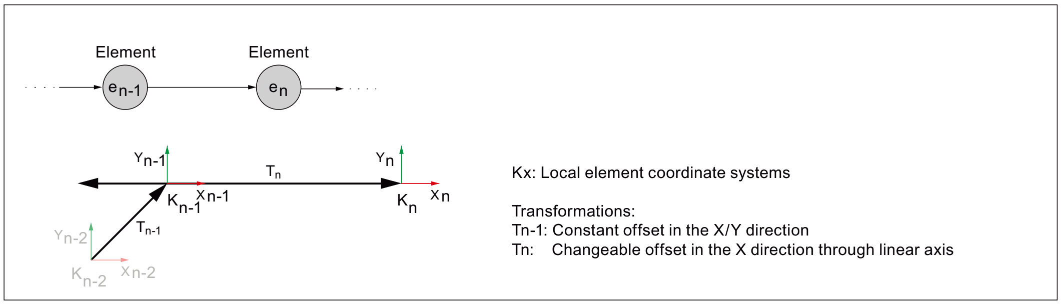

Element
Element
e_{n-1}
e_n
Y_{n-1}
T_n
Y_n
K_{n-1}
X_{n-1}
K_n
X_n
Y_{n-2}
T_{n-1}
K_{n-2}
X_{n-2}
Kx: Local element coordinate systems
Transformations:
Tn-1: Constant offset in the X/Y direction
Tn: Changeable offset in the X direction through linear axis

Figure 5-2 Local element coordinate systems

The following constant transformations are possible:

- Offset (type: OFFSET (Page 249))
- Rotation (type: ROT_CONST (Page 247))

The following changing transformations, based on the current position values of the machine axis assigned to the element (linear axis / rotary axis), are possible:

- Offset (type: AXIS_LIN (Page 240))
- Rotation (type: AXIS_ROT (Page 243))

A position or orientation change in an element, e.g. through a position change of the associated machine axis, affects all the **following** elements of the chain or parallel subchains.

The maximum number of possible elements can be parameterized using machine data (Page 235).

Parallel subchains

If a parallel subchain branches off from an element e_n, then for the kinematics, the branch is always **before** the element. For this reason, a change in element e_n, e.g. a position change of the associated machine axis, does **not** affect the subchain branch.

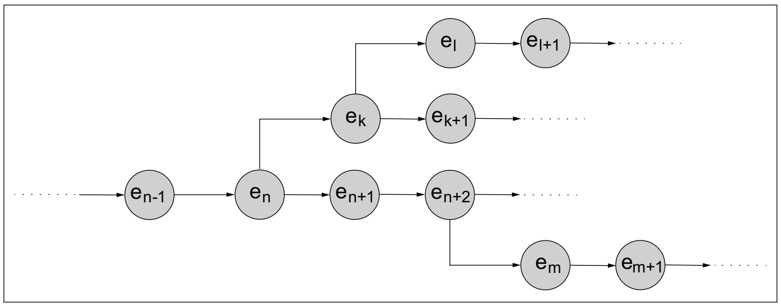


Figure 5-3 Parallel subchains branching off from elements e_n and e_{n+2}

Switch

A switch is a special element that is not kinematically active (type: SWITCH (Page 250)) in a kinematic chain, which can assume the CLOSED and OPEN states.

In the **OPEN** state, the connection from the previous to the following element is interrupted.

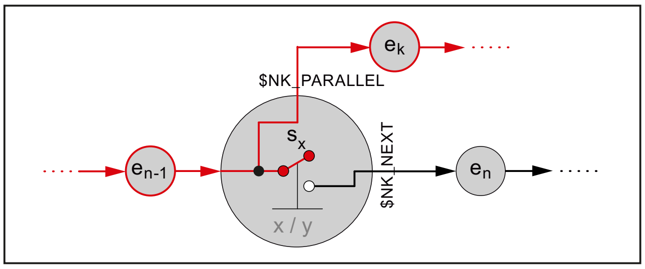


Figure 5-4 OPEN state

In the **CLOSED** state, the previous element is connected to the following element.

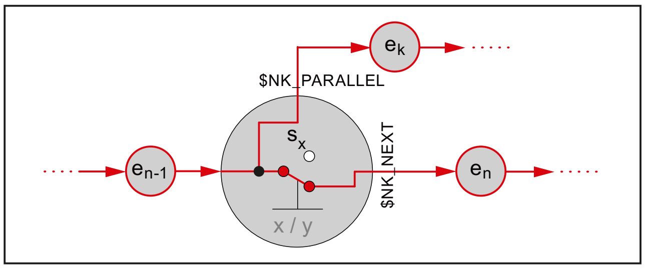


Figure 5-5 CLOSED state

The switch does not influence the connection to a parallel element.

The maximum number of possible switches can be parameterized usingmachine data (Page 235).

Note

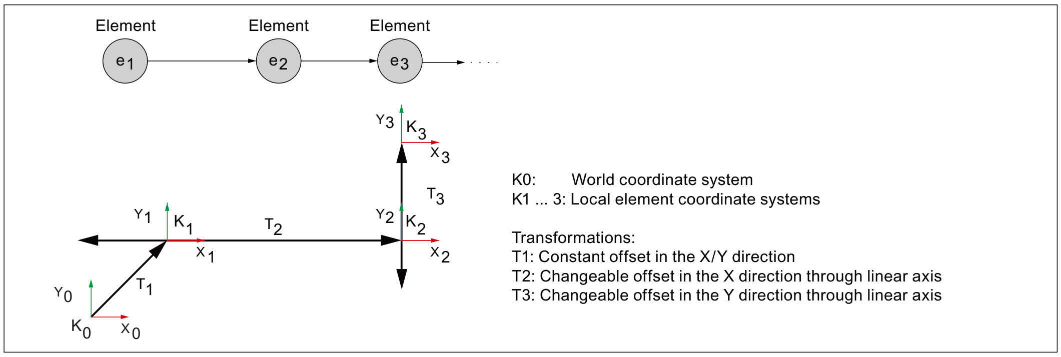
Local coordinate system

The local coordinate system of a switch is not rotated with respect to the world coordinate system.

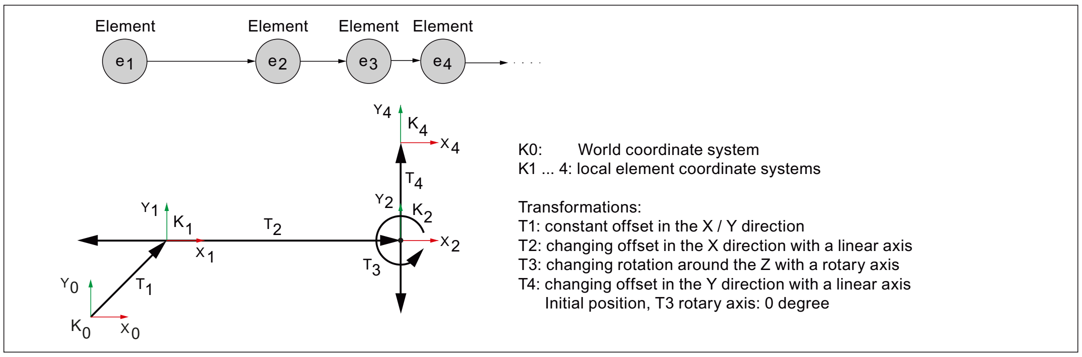
World coordinate system

In order to uniquely describe the kinematic stricture of a machine, all elements of the kinematic chain - as well as the orientation and offset vectors of the machine axes and the constant rotation/offsets - are referred to the world coordinate system.

Elements with **linear axes** do not change the orientation of the following elements. The orientation and offset vectors of the following elements can therefore refer to the world coordinate system without further supplementary conditions.


Element
e_1
Element
e_2
Element
e_3
Y_3
K_3
X_3
T_3
Y_2
K_2
X_2
Y_1
K_1
X_1
T_2
T_1
Y_0
K_0
X_0
K0: World coordinate system
K1 ... 3: Local element coordinate systems
Transformations:
T1: Constant offset in the X/Y direction
T2: Changeable offset in the X direction through linear axis
T3: Changeable offset in the Y direction through linear axis

Elements with **rotary axes** change the orientation of the following elements. This is the reason that the orientation and offset vectors of the following elements, for a defined initial position of the previous orientation changing elements, must be referred to the world coordinate system.


Element
e1
Element
e2
Element
e3
Element
e4
Y4
K4
X4
T4
Y2
K2
X2
T3
Y1
K1
X1
T2
Y0
T1
K0
X0
K0: World coordinate system
K1 ... 4: local element coordinate systems
Transformations:
T1: constant offset in the X / Y direction
T2: changing offset in the X direction with a linear axis
T3: changing rotation around the Z with a rotary axis
T4: changing offset in the Y direction with a linear axis
Initial position, T3 rotary axis: 0 degree

The origin and orientation of the world coordinate system can be freely selected in chain elements that are defined before the root element. For the active coordinate system after the root element, the following conditions must be met:

- Origin of the world coordinate system at the machine zero point
- Orientation of the world coordinate system so that the coordinate axes are arranged in the positive traversing direction of the machine linear main axes

Direction vectors

Within a kinematic chain, the direction vectors, which are used to specify the alignment of the machine axes, are always specified in absolute terms, i.e. referred to the world coordinate system.

5.2 Commissioning

5.2.1 General

5.2.1.1 Overview

The commissioning of the "Kinematic chain" function is performed using:

- Machine data
  - Specification of the quantity structure
  - Specification of the first element in the kinematic chain
- System variables
  - Specification of the kinematic properties of an element
  - Connection of the element to the kinematic chain

5.2.1.2 Structure of the system variables

The system variables are structured according to the following scheme:

- **$NK_**<Name>[<Index_1>]
- **$NK_**<Name>[<Index_1>, <Index_2>]

General

The system variables to describe the elements of kinematic chains have the following properties:

- The prefix for all system variables of the kinematic chain is **$NK_**, (N for NC, K for kinematic).
- 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 → 1. Element, index 1 → 2. Element, ... n → (n+1) element, with n = ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 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

5.2.2 Machine data

5.2.2.1 Maximum number of elements

The maximum number of elements for kinematic chains is set using machine data.

MD18880 $MN_MM_MAXNUM_KIN_CHAIN_ELEM = <number>

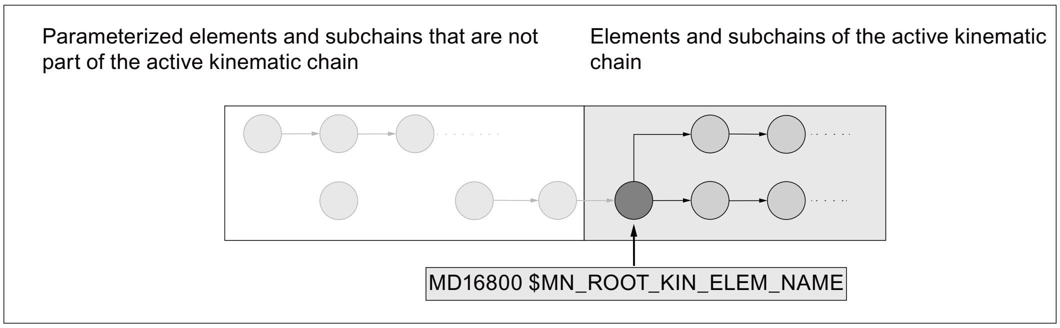
5.2.2.2 Root element

The following machine data is used to specify the root element, i.e. the first element of the currently active kinematic chain.

MD16800 $MN_ROOT_KIN_ELEM_NAME = "<element_name>"

All parameterized elements and subchains that are connected to the root element in such a way that the connection starts at the root element, are part of the currently active kinematic chain.

Parameterized elements or subchains that are not connected to the root element or whose connection points towards the root element, are not part of the currently active kinematic chain.


Parameterized elements and subchains that are not part of the active kinematic chain
Elements and subchains of the active kinematic chain
MD16800 $MN_ROOT_KIN_ELEM_NAME

5.2.2.3 Maximum number of switches

The maximum number of switches for kinematic chains is set with the machine data

MD18882 $MN_MM_MAX_NUM_KIN_SWITCHES = <Number>

5.2.3 System variables

5.2.3.1 Overview

System variables independent of any element

System variable	Meaning
$NK_SWITCH	Switch variable to open and close the switch

Element-specific system variables

Element-specific system variables are subdivided into variables that are independent of the type and dependent on the type:

- Variables independent of the type

System variable	Meaning
$NK_NAME	Name of the current element e_n
$NK_NEXT	Name of the next element e_{n+1}
$NK_PARALLEL	Name of a branching parallel element e_p **before** the current element e_n
$NK_TYPE	Type of the element

- Variables dependent on the type

System variable	Meaning
$NK_OFF_DIR	Offset or direction vector
$NK_AXIS	Machine axis or object name
$NK_A_OFF	Work offset for linear axis or rotary axis
$NK_SWITCH_INDEX	Switch index
$NK_SWITCH_POS	Switch position "CLOSED"

For the following types, variables dependent on the type are evaluated:

System variable	Type				
	OFFSET	AXIS_LIN	AXIS_ROT	ROT_CONST	SWITCH
$NK_OFF_DIR	x	x	x	x	-
$NK_AXIS	-	x	x	-	-
$NK_A_OFF	-	x	x	x	-
$NK_SWITCH_INDEX	-	-	-	-	x
$NK_SWITCH_POS	-	-	-	-	x

The system variables are described in detail in the following sections.

Note

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() (Page 253).

Change system variable values

If the value of one of the system variables listed above is changed, then the change becomes immediately visible on the user interface, e.g. SINUMERIK Operate. The machine model of the NC is only updated after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

5.2.3.2 $NK_NAME

Function

Enter the NC-wide unique element name in the system variable. The element is referenced via this name, e.g. within kinematic chains. The name is also displayed in the graphical editor of SINUMERIK Operate.

Syntax

```
$NK_NAME[<n>] = "<name>"
```

Meaning

`$NK_NAME`:	Name of the element	
	Data type:	STRING
	Default value:	"" (empty string)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<name>`:	Element name, max. string length: 31 characters	
	Data type:	STRING

Example

The 9th kinematic element is assigned the name "B axis":

Program code	Comment
`N100 $NK_NAME[8] = "B axis"`	`; 9th kinematic element,`
	`; name = "B axis"`

5.2.3.3 $NK_NEXT

Function

If the element is part of a kinematic chain, the name of the following element should be entered in the system variable.

Syntax

```
$NK_NEXT[<n>] = "<name>"
```

Meaning

`NK_NEXT`:	Name of the following element	
	Data type:	STRING
	Range of values:	All names contained in $NK_NAME (Page 237)
	Default value:	"" "" (empty string)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<name>`:	Element name, max. string length: 31 characters	
	Data type:	STRING

Example

The 9th kinematic element does not have a following element:

Program code	Comment

`N100 $NK_NEXT[8] = ""`	`; 9. kin. element,`
	`; following element = ""`

5.2.3.4 $NK_PARALLEL

Function

The name of the element that branches off **before** the current element should be entered in the system variable. The branching element is parallel to the current element. Changes to the current element, e.g. position changes of the assigned machine axis, have no effect on the parallel element.

Syntax

```
$NK_PARALLEL[<n>] = "<name>"
```

Meaning

`$NK_PARALLEL`:	Name of the parallel element	
	Data type:	STRING
	Range of values:	All names contained in $NK_NAME (Page 237)
	Default value:	"" "" (empty string)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<name>`:	Element name, max. string length: 31 characters	
	Data type:	STRING

Example

The "Offset to the rotary table" element is arranged parallel to the 9th kinematic element:

Program code	**Comment**
`N100 $NK_PARALLEL[8] = "Offset to the rotary table"`	`; Parallel to the 9th kin. element,` `; name = "Offset to the rotary table"`

5.2.3.5 $NK_TYPE

Function

The element type should be entered in the system variable:

Type	Description
AXIS_LIN (Page 240)	The element describes a linear machine axis (linear axis) with direction vector $NK_OFF_DIR and work offset $NK_A_OFF.
AXIS_ROT (Page 243)	The element describes a rotary machine axis (rotary axis) with direction vector $NK_OFF_DIR and work offset $NK_A_OFF or a spindle. **Note** The position of a spindle is considered differently depending on the function used by the kinematic chain: • Collision avoidance: Indeterminate position • Kinematic transformation: Depending on the setting in $NT_CNTRL, bit 1-3, the position corresponding to $NK_A_OFF or the speed setpoint is taken into consideration
ROT_CONST (Page 247)	The element describes a constant rotation around direction vector $NK_OFF_DIR and angle $NK_A_OFF.
OFFSET (Page 249)	The element describes a constant linear offset with the offset vector $NK_OFF_DIR.
SWITCH (Page 250)	The element defines a switch, which is switched using system variable $NK_SWITCH (Page 252).

Syntax

```
$NK_TYPE[<n>] = "<Typ>"
```

Meaning

`$NK_TYPE`:	Type of the element	
	Data type:	STRING
	Default value:	"" (empty string)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<Typ>`:	Element name, max. string length: 31 characters	
	Data type:	STRING
	Range of values:	"AXIS_LIN", "AXIS_ROT", "ROT_CONST", "OFFSET", "SWITCH"

Example

The 9th kinematic element is a rotary axis:

```
Program code                                Comment
N100 $NK_TYPE[8] = "AXIS_ROT"               ; 9th kin. element
                                            ; type = rotary axis
```

5.2.3.6 Type-dependent variables for $NK_TYPE = "AXIS_LIN"

$NK_OFF_DIR

Function

The direction vector along which the linear axis $NK_AXIS assigned to the element moves, should be entered in the system variable. The output coordinate system therefore results from the input coordinate system, offset by the current position value of the linear axis and the work offset specified in $NK_A_OFF.

Supplementary conditions:

- The direction vector must be specified as an absolute value, i.e. in relation to the **world coordinate system**.
- The direction vector must be specified so that it points in the positive traversing direction of the machine axis.
- The absolute value of the direction vector must be greater than $1*10^{-6}$.

Syntax

```
$NK_OFF_DIR[<n>,<k>] = <value>
```

Meaning

`$NK_OFF_DIR`:	Direction vector (X; Y; Z)	
	Data type:	REAL
	Range of values:	Direction vector: $1*10^{-6}$ < |Vector | ≤ max. REAL value
	Default value:	(0.0, 0.0, 0.0)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<k>`:	Coordinate index	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value

Example

; The linear axis of the 9th element moves along the direction vector. The direction vector is the unit vector (1; 0; 0), rotated around Z with γ=90° in the X/Y plane, and around X with α=10° in the Y/Z plane referred to the world coordinate system. The following values result from this for the individual components (x, y, z) of the direction vector:

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

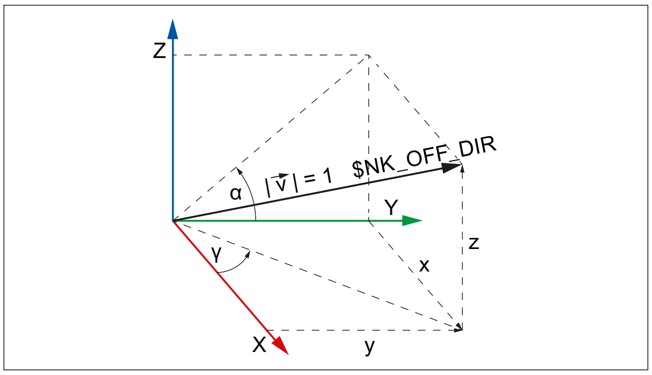


Figure 5-6 Direction vector, general

Program code	Comment
`; 9th kinematic element`	
`N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)`	`; 0 = X component`
`N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)`	`; 1 = Y component`
`N120 $NK_OFF_DIR[8,2] = SIN(10)`	`; 2 = Z component`

$NK_AXIS

Function

The name of the machine axis (MD10000 $MN_AXCONF_MACHAX_NAME_TAB) that is to be assigned to the element should be entered in the system variable.

The output coordinate system of the element results from the input coordinate system offset by the current position setpoint of the machine axis in the MCS and the offset specified in $NK_A_OFF. The position setpoint of the machine axis contains all the active work offsets and corrections.

In accordance with the AXIS_LIN type entered in $NK_TYPE, the machine axis must be a linear axis:

MD30300 $MA_IS_ROT_AX = 0

Syntax

```
$NK_AXIS[<n>] = <name>
```

Meaning

`$NK_AXIS`:	Machine axis name	
	Data type:	STRING
	Range of values:	Machine axis name
	Default value:	"" (empty string)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<value>`:	Machine axis name	
	Data type:	STRING
	Range of values:	Machine axis names

Example

The 9th kinematic element is assigned the machine axis with the name V1 as linear axis.

```
Program code                        Comment
N100 $NK_AXIS[8] = "V1"             ; 9th kin. element
                                    ; axis = machine axis V1
```

$NK_A_OFF

Function

An additional work offset can be entered in the system variable for the assigned machine axis ($NK_AXIS). This work offset is only effective within the kinematic chain.

Syntax

```
$NK_A_OFF[<n>] = <value>
```

Meaning

`$NK_A_OFF`:	Work offset	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value
	Default value:	0.0
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<value>`:	Offset value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value

Example

The linear axis zero point of the 9th kinematic element is moved by 30.0 mm compared to the modelled kinematics.

The unit in which the numerical value is interpreted depends on the current input system (inch/metric).

```
Program code                              Comment
N100 $NK_A_OFF[8] = 30.0                  ; 9th kin. element
                                          ; Work offset = 30.0 mm
```

5.2.3.7 Type-dependent variables for $NK_TYPE = "AXIS_ROT"

$NK_OFF_DIR

Function

The direction vector around which the rotary axis $NK_AXIS assigned to the element rotates, should be entered in the system variable. The output coordinate system is therefore calculated from the input coordinate system, rotated by the current position value of the rotary axis and the offset specified in $NK_A_OFF around the direction vector $NK_OFF_DIR.

Supplementary conditions:

- The direction vector must be specified as an absolute value, i.e. in relation to the **world coordinate system**.
- The direction vector must be specified so that, in accordance with the "right-hand rule", the thumb points in the direction of the vector when the rotary axis turns in the positive direction.
- The absolute value of the direction vector must be greater than $1*10^{-6}$.

Note

Spindle

If the assigned machine axis is a spindle, its position is taken into account in different ways specific to the function:

- Collision avoidance: Indeterminate position
- Kinematic transformation: depending on the setting in $NT_CNTRL, bit 1-3
  - Bit x == 0 → indeterminate position
  - Bit x == 1 → current position value + $NK_A_OFF,

Syntax

```
$NK_OFF_DIR[<n>,<k>] = <value>
```

Meaning

`$NK_OFF_DIR`:	Direction vector (X; Y; Z)	
	Data type:	REAL
	Range of values:	Direction vector: $1*10^{-6}$ < |Vector | ≤ max. REAL value
	Default value:	(0.0, 0.0, 0.0)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<k>`:	Coordinate index	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value

Example

The rotary axis of the 9th element rotates around the direction vector. The direction vector is the unit vector (1; 0; 0), rotated around Z with γ=90° in the X/Y plane, and around X with α=10° in the Y/Z plane referred to the world coordinate system. The following values result from this for the individual components (x, y, z) of the direction vector:

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

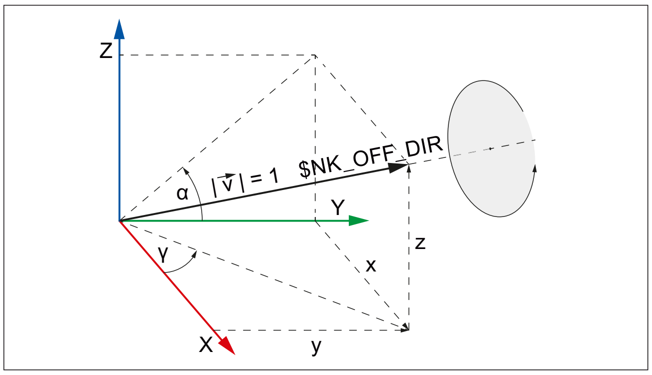


Figure 5-7 Direction vector, general

Program code	Comment
; 9th kinematic element	
N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)	; 0 = X component
N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)	; 1 = Y component
N120 $NK_OFF_DIR[8,2] = SIN(10)	; 2 = Z component

$NK_AXIS

Function

The name of the machine axis (MD10000 $MN_AXCONF_MACHAX_NAME_TAB) that is to be assigned to the element should be entered in the system variable.

The output coordinate system of the element results from the input coordinate system rotated through the current position setpoint of the machine axis in the MCS and the offset specified in $NK_A_OFF. The position setpoint of the machine axis contains all the active work offsets and corrections.

The machine axis type must correspond to the type entered in $NK_TYPE:

In accordance with the AXIS_ROT type entered in $NK_TYPE, the machine axis must be a rotary axis:

MD30300 $MA_IS_ROT_AX = 1

Syntax

```
$NK_AXIS[<n>] = <name>
```

Meaning

$NK_AXIS:	Machine axis name	
	Data type:	STRING
	Range of values:	Machine axis name
	Default value:	"" (empty string)
<n>:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)

`<value>`:	Machine axis name	
	Data type:	STRING
	Range of values:	Machine axis names

Example

The 9th kinematic element is assigned the machine axis with the name B1 as rotary axis.

Program code	Comment
`N100 $NK_AXIS[8] = "B1"`	`; 9th kin. element`
	`; axis = machine axis B1`

$NK_A_OFF

Function

An additional work offset can be entered in the system variable for the assigned machine axis ($NK_AXIS). This work offset is only effective within the kinematic chain.

Syntax

```
$NK_A_OFF[<n>] = <value>
```

Meaning

`$NK_A_OFF`:	Work offset	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value
	Default value:	0.0
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<value>`:	Offset value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value

Example

The rotary axis zero point of the 9th kinematic elements is moved through 30.0° compared to the modelled kinematics.

Program code	Comment
`N100 $NK_A_OFF[8] = 30.0`	`; 9th kin. element`
	`; Work offset = 30.0°`

5.2.3.8 Type-dependent variables for $NK_TYPE = "ROT_CONST"

$NK_OFF_DIR

Function

The direction vector around which the constant rotation is performed should be entered in the system variable. The output coordinate system is therefore calculated from the input coordinate system, rotated through the angle specified in $NK_A_OFF around the direction vector $NK_OFF_DIR.

Supplementary conditions:

- The direction vector must be specified as an absolute value, i.e. in relation to the **world coordinate system**.
- The direction vector must be specified so that, in accordance with the "right-hand rule", the thumb points in the direction of the vector when the rotary axis turns in the positive direction.
- The absolute value of the direction vector must be greater than 1*10^{-6}.

Syntax

```
$NK_OFF_DIR[<n>,<k>] = <value>
```

Meaning

`$NK_OFF_DIR`:	Direction vector (X; Y; Z)	
	Data type:	REAL
	Range of values:	Direction vector: 1*10^{-6} < |Vector | ≤ max. REAL value
	Default value:	(0.0, 0.0, 0.0)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<k>`:	Coordinate index	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value

Example

The output coordinate system of the 9th Elements therefore arises from the input coordinate system, rotated through the angle specified in $NK_A_OFF around the direction vector. The direction vector is the unit vector (1; 0; 0), rotated around Z with γ=90° in the X/Y plane, and around X with α=10° in the Y/Z plane referred to the world coordinate system. The following values result from this for the individual components (x, y, z) of the direction vector:

- x = cos(γ) * cos(α) = cos(90) * cos(10) = 0.0
- y = sin(γ) * cos(α) = sin(90) * cos(10) ≈ 0.985
- z = sin(α) = sin(10) ≈ 0.174

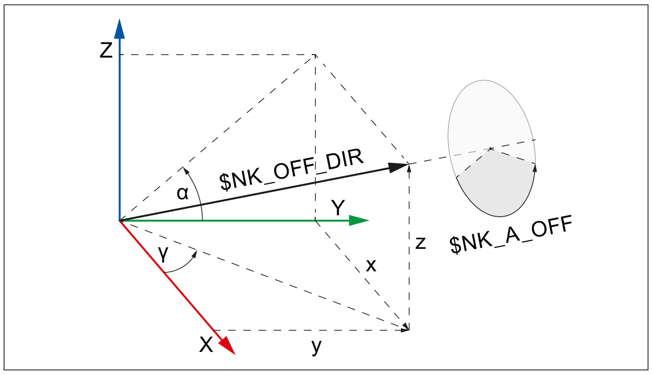


Figure 5-8 ROT_CONST

Program code	Comment
`; 9th kinematic element`	
`N100 $NK_OFF_DIR[8,0] = COS(90)*COS(10)`	`; 0 = X component`
`N110 $NK_OFF_DIR[8,1] = SIN(90)*COS(10)`	`; 1 = Y component`
`N120 $NK_OFF_DIR[8,2] = SIN(10)`	`; 2 = Z component`

$NK_A_OFF

Function

The angle through which the output coordinate system is rotated compared with the input coordinate system around the direction vector $NK_OFF_DIR should be entered in the system variable.

Syntax

```
$NK_A_OFF[<n>] = <value>
```

Meaning

`$NK_A_OFF`:	Angle of rotation	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value
	Default value:	0.0
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)

`<value>`:	Angle	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ + max. REAL value

Example

The angle of rotation of the 9th kinematic element is 30.0°.

Program code	Comment
`N100 $NK_A_OFF[8] = 30.0`	`; 9th kin. element`
	`; Angle of rotation = 30.0°`

5.2.3.9 Type-dependent variables for $NK_TYPE = "OFFSET"

$NK_OFF_DIR

Function

The offset vector by which the output coordinate system is moved compared to the input coordinate system should be entered in the system variable.

The offset vector must be specified as an absolute value, i.e. in relation to the **world coordinate system**.

Syntax

```
$NK_OFF_DIR[<n>,<k>] = <value>
```

Meaning

`$NK_OFF_DIR`:	Offset vector (X; Y; Z)	
	Data type:	REAL
	Range of values:	(- max. REAL value) ≤ x ≤ (+ max. REAL value)
	Default value:	(0.0, 0.0, 0.0)
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<k>`:	Coordinate index	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	(- max. REAL value) ≤ x ≤ (+ max. REAL value)

Example

The output coordinate system of the 9th element is obtained from the input coordinate system, offset through the offset vector (x, y, z) with the following coordinates referred to the world coordinate system:

- x = 10.0
- y = 20.0
- z = 30.0

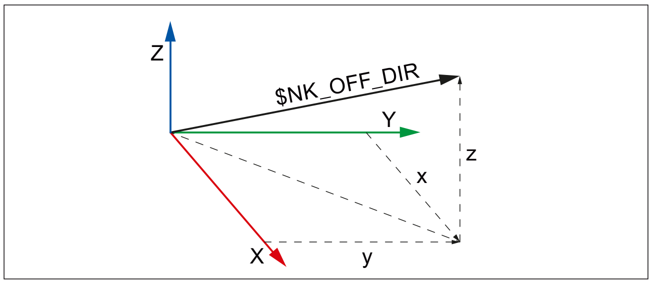


Figure 5-9 Offset vector

Program code	Comment
; 9th kinematic element	
N100 $NK_OFF_DIR[8,0] = 10.0	; 0 = X component
N110 $NK_OFF_DIR[8,1] = 20.0	; 1 = Y component
N120 $NK_OFF_DIR[8,2] = 30.0	; 2 = Z component

5.2.3.10 Type-dependent variables for $NK_TYPE = "SWITCH"

$NK_SWITCH_INDEX

Function

The switch is emulated using system variables $NK_SWITCH_INDEX and $NK_SWITCH_POS (see the following paragraph)

Index **i** must be entered in $NK_SWITCH_INDEX, with which the switch is closed and opened via system variable $NK_SWITCH[**<i>**].

Syntax

```
$NK_SWITCH_INDEX[<n>] = <i>
```

Meaning

$NK_SWITCH_ INDEX:	Index i, via which the switch is addressed using system variable $NK_SWITCH[**<i>**]	
	Data type:	INT
	Range of values:	-1, 0, 1, 2, ... ($MN_MAXNUM_KIN_SWITCHES - 1)
		-1: The state of the switch is always CLOSED
	Default value:	-1

`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<i>`:	Index value	
	Data type:	INT

Example

The 9th element of the kinematic chain is a switch, which is activated by system variable $NK_SWITCH[**2**].

Program code	Comment
`; 9th element`	
`N100 $NK_SWITCH_INDEX[8] = 2`	`; Index 2`

$NK_SWITCH_POS

Function

A switch is emulated by system variables $NK_SWITCH_INDEX (see the previous paragraph) and $NK_SWITCH_POS.

In $NK_SWITCH_POS, value p (close value) should be entered, where the switch is brought into the CLOSED position by system variable $NK_SWITCH[<**i**>].

Independent of $NK_SWITCH[<**i**>], the switch has the following states:

- **CLOSED**: The close value p of the switch, specified in $NK_SWITCH_POS is the same as the actual value of $NK_SWITCH[<i>].
  `$NK_SWITCH_POS[<n>] == $NK_SWITCH[<i>]`
  The previous element of the kinematic chain is connected with the output, i.e. the following element of the switch specified in $NK_NEXT.
- **OPEN**: The close value of the switch, specified in $NK_SWITCH_POS, is not the same as the actual value of $NK_SWITCH[<i>].
  `$NK_SWITCH_POS[<n>] ≠ $NK_SWITCH[<i>]`
  The previous element of the kinematic chain is **not**connected with the output, i.e. the following element of the switch specified in $NK_NEXT.

The close value can be freely selected.

Note

Parallel element $NK_PARALLEL

The connection to a parallel element specified in $NK_PARALLEL is not influenced by the switch. This means that the previous element is always connected to the element in the branch parallel with the switch.

Syntax

```
$NK_SWITCH_POS[<n>] = <p>
```

Meaning

`$NK_SWITCH_POS`:	Close value	
	Data type:	INT
	Range of values:	0 ≤ x ≤ positive max. INT value
	Default value:	0
`<n>`:	System variable or element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_KIN_CHAIN_ELEM - 1)
`<p>`:	Value for switch position CLOSED	
	Data type:	INT

Example

The 9th element of the kinetic chain is a switch, which is closed if $NK_SWITCH[3] == 1

```
Program code                                        Comment
; 9th element
N110 $NK_SWITCH_POS[8] = 1                          ; switch position = 1
```

See also

$NK_SWITCH (Page 252)

$MN_MAXNUM_KIN_SWITCHES (Page 235)

$MN_MM_MAXNUM_KIN_CHAIN_ELEM (Page 235)

5.2.3.11 $NK_SWITCH

Function

The switch variable comprises an array of switch positions i. The actual switch positions p should be entered in this array.

Function

To parameterize (Page 250) a switch$_i$ in a kinematic chain, switch$_i$with index i must be connected to the switch variable and its switch position p for the CLOSED state assigned to it.

```
$NK_SWITCH_INDEX[<n>] = <i>
$NK_SWITCH_POS[<n>]   = <p>
```

Switch$_i$ can then be closed and opened via index i of the switch variable:

Close: `$NK_SWITCH[<i>] = <p>`

Open: `$NK_SWITCH[<i>] ≠ <p>`

As many switches as required can be connected with an index of the switch variable.

Syntax

```
$NK_SWITCH[<i>] = <p>
```

Meaning

`$NK_SWITCH<i>`:	Switch variable with switch positions i	
	Data type:	INT
	Value range:	-1 ≤x ≤ positive max. INT value Note -1: Initial state OPEN
`<i>`:	Switch index	
	Data type:	INT
	Range of values:	0, 1, 2, ... (MD18882 $MN_MM_MAX_NUM_KIN_SWITCHES - 1)
`<p>`:	Switch setting	
	Data type:	INT
	Value range:	-1 ≤x ≤ positive max. INT value
		-1: Initial state OPEN

Example

The 9th kinematic element is assigned the name "B axis":

Program code	Comment
`N100 $NK_SWITCH[3] = 1`	`; actual switch position`
	`; for switch [3] = 1`

5.3 Programming

5.3.1 Deletion of components (DELOBJ)

The `DELOBJ()` function "deletes" components by resetting the assigned system variables to their default values:

- Elements from kinematic chains
- Protection areas, protection area elements and collision pairs
- Transformation data

Syntax

```
[<RetVal>=] DELOBJ(<CompType>[,,,<NoAlarm>)])
[<RetVal>=] DELOBJ(<CompType>,<Index1>[,,<NoAlarm>])
[<RetVal>=] DELOBJ(<CompType>[,<Index1>][,<Index2>][,<NoAlarm>])
```

Meaning

`DELOBJ`:	Deletion of elements from kinematic chains, protection areas, protection area elements, collision pairs and transformation data
`<CompType>`:	Component type to be deleted
	Data type: STRING
	Value: "KIN_CHAIN_ELEM" **Meaning**: System variables of all kinematic elements: $NK_...
	Value: "KIN_CHAIN_SWITCH" **Meaning**: System variable $NK_SWITCH[<i>]
	Value: "KIN_CHAIN_ALL" **Meaning**: All kinematic elements and switches. Is the same as the successive call of DELOBJ with "KIN_CHAIN_ELEM" and "KIN_CHAIN_SWITCH"
	Value: "PROT_AREA" **Meaning**: System variables of the protection areas: • $NP_PROT_NAME • $NP_CHAIN_NAME • $NP_CHAIN_ELEM • $NP_1ST_PROT
	Value: "PROT_AREA_ELEM" **Meaning**: System variables of the protection area elements of machine protection areas and/or automatic tool protection areas: • $NP_NAME • $NP_NEXT • $NP_NEXTP • $NP_COLOR • $NP_D_LEVEL • $NP_USAGE • $NP_TYPE • $NP_FILENAME • $NP_PARA • $NP_OFF • $NP_DIR • $NP_ANG
	Value: "PROT_AREA_COLL_PAIRS" **Meaning**: System variables of the collision pairs: • $NP_COLL_PAIR • $NP_SAFETY_DIST
	Value: "PROT_AREA_ALL" **Meaning**: All protection areas, protection area elements and collision pairs (system variable $NP_...) Is the same as the successive call of DELOBJ with "PROT_AREA," "PROT_AREA_ELEM," and "PROT_AREA_COLL_PAIRS"
	Value: "TRAFO_DATA" **Meaning**: System variables of all transformations $NT_...

`<Index1>`:	Index of the first component to be deleted (**optional**)	
	Data type:	INT
	Default value:	-1
	Range of values:	-1 ≤ x ≤ (maximum number of configured components -1)
	Value	**Meaning**
	0, 1, 2, ...	Index of the component to be deleted.
	-1	All components of the specified type are deleted. <Index2> is not evaluated.
`<Index2>`:	Index of the last components to be deleted (**optional**) If <Index2> is not programmed, only the system variables of the component referenced in <Index1> are deleted.	
	Data type:	INT
	Default value:	Only the system variables of the component referenced in <Index1> are deleted.
	Range of values:	<Index1> < x ≤ (max. number of configured components -1)
`<NoAlarm>`:	Alarm suppression (**optional**)	
	Data type:	BOOL
	Default value:	FALSE
	Value	**Meaning**
	FALSE	In the event of an error (<RetVal> < 0), program processing is stopped and an alarm displayed.
	TRUE	In the event of an error, the program processing is not stopped and no alarm displayed. Application: User-specific reaction corresponding to the return value
`<RetVal>`:	Function return value	
	Data type:	INT
	Range of values:	0, -1, -2, ... -7
	Value	**Meaning**
	0	No error occurred.
	-1	Call of the function without parameters. At least parameter <CompType> must be specified.
	-2	<CompType> identifies an unknown component
	-3	<Index1> is less than -1
	-4	<Index1> is greater than the configured number of components
	-5	<Index1> has a value not equal to -1 when deleting a component **group**
	-6	<Index2> is less than <Index1>
	-7	<Index2> is greater than the configured number of components

5.3.2 Index determination by means of names (NAMETOINT)

User-specific names are entered in the system variable arrays of type STRING. Based on the identifier of the system variables and the name, the `NAMETOINT()` function determines the index value belonging to the name under which it is stored in the system variable array.

Syntax

```
<RetVal> = NAMETOINT(<SysVar>,<Name>[,<NoAlarm>])
```

Meaning

`NAMETOINT`:	Determining the system variable index	
`<SysVar>`:	Name of the system variable array of typeSTRING	
	Data type:	STRING
	Range of values:	Name of all NC system variable arrays of type STRING
`<Name>`:	Character string or name for which the system variable index is to be determined.	
	Data type:	STRING
`<NoAlarm>`:	Alarm suppression (**optional**)	
	Data type:	BOOL
	Default value:	FALSE
	Value	**Meaning**
	TRUE	In the event of an error, the program processing is not stopped and no alarm displayed. Application: User-specific reaction corresponding to the return value
	FALSE	In the event of an error (<RetVal> < 0), program processing is stopped and an alarm displayed.
`<RetVal>`:	System variable index or error message	
	Data type:	INT
	Range of values:	-1 ≤ x ≤ (max. number of configured components -1)
	Value	**Meaning**
	≥ 0	The sought name has been found under the specified system variable index.
	-1	The sought name has not been found or an error has occurred.

Example

Program code	Comment
`DEF INT INDEX`	
`$NP_PROT_NAME[27]="Cover"`	
`...`	
`INDEX = NAMETOINT("$NP_PROT_NAME","Cover")`	`; INDEX == 27`

5.4 Example

5.4.1 Specifications

General Information

The principle approach when parameterizing a kinematic chain with three switches using a part program is shown using as example a 5-axis machine with three different tool heads, which are used alternately. All of the system variables, relevant for kinematic chain, are written to the part program:

- Kinematic chain $NK_...

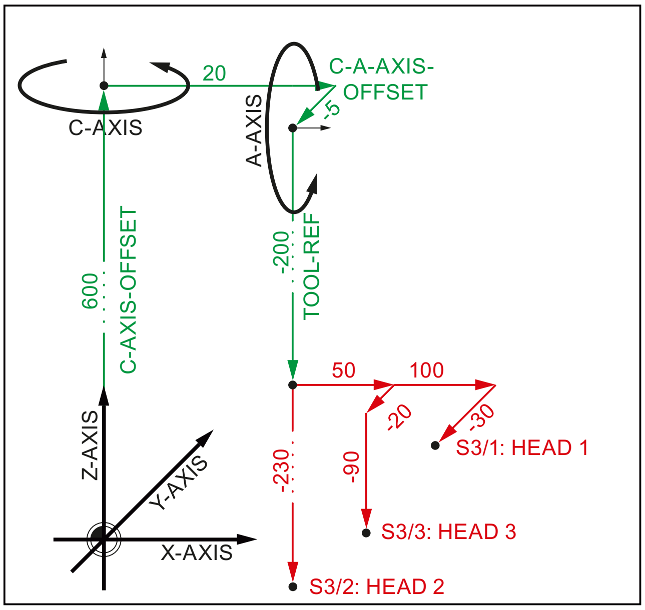
Option and machine data

The following option and machine data must be set for the example:

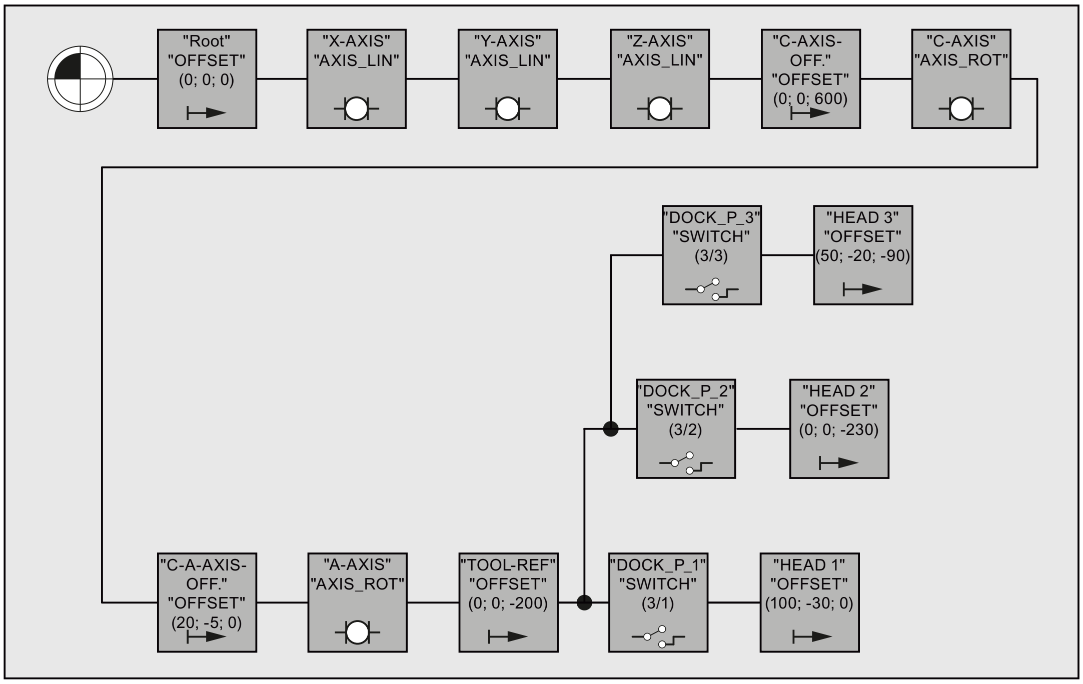
- MD19830 $ON_COLLISION_MASK.Bit 0 = 1
- MD18880 $MN_MM_MAXNUM_KIN_CHAIN_ELEM = 15

Kinematic chain

Schematic representation of the machine kinematics



Elements of the kinematic chain


"Root" "OFFSET" (0; 0; 0)
"X-AXIS" "AXIS_LIN"
"Y-AXIS" "AXIS_LIN"
"Z-AXIS" "AXIS_LIN"
"C-AXIS-OFF." "OFFSET" (0; 0; 600)
"C-AXIS" "AXIS_ROT"
"DOCK_P_3" "SWITCH" (3/3)
"HEAD 3" "OFFSET" (50; -20; -90)
"DOCK_P_2" "SWITCH" (3/2)
"HEAD 2" "OFFSET" (0; 0; -230)
"C-A-AXIS-OFF." "OFFSET" (20; -5; 0)
"A-AXIS" "AXIS_ROT"
"TOOL-REF" "OFFSET" (0; 0; -200)
"DOCK_P_1" "SWITCH" (3/1)
"HEAD 1" "OFFSET" (100; -30; 0)

The kinematic chain starts with an element of the type "Offset". When completely parameterizing the collision avoidance, these are assigned to all static protection areas of the machine

The offset element is followed by the kinematic elements of the linear machine axes X, Y and Z - as well as the offset and kinematic elements of rotary axes C and A.

The offset element for the reference point of the tool heads is followed by three switches to switch on and switch off the tool heads. As a result of $NK_INDEX = **3** all three switches refer to the same switch variable $NK_SWITCH[**3**]. As a result of the different switch positions (1, 2, 3), only the subchain of the active tool head is active.

5.4.2 Part program of the machine model

Program code

```
;============================================================
; Definitions
;============================================================
N10 DEF INT KIE_CNTR ; COUNTER FOR ELEMENTS OF THE KIN. CHAINS
N20 DEF INT RETVAL
;
;============================================================
; Initialization of collision data
;============================================================
; Reset all parameters to their initial values:
N30 RETVAL = DELOBJ("KIN_CHAIN_ELEM")
N40 KIE_CNTR = 0
;
;============================================================
; Kinematic chain
;============================================================
; KE1: OFFSET: Root
; ----------------------------------------------------------
N50 $NK_TYPE[KIE_CNTR] = "OFFSET"
N60 $NK_NAME[KIE_CNTR] = "ROOT"
N70 $NK_NEXT[KIE_CNTR] = "X-AXIS"
N80 KIE_CNTR = KIE_CNTR + 1
;
; ----------------------------------------------------------
; Kinematic element: LINEAR AXIS: X axis
; ----------------------------------------------------------
N90 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N100 $NK_NAME[KIE_CNTR] = "X-AXIS"
N110 $NK_NEXT[KIE_CNTR] = "Y-AXIS"
N120 $NK_AXIS[KIE_CNTR] = "X1"
;
N130 $NK_OFF_DIR[KIE_CNTR,0] = 1.0    ; X
N140 KIE_CNTR = KIE_CNTR + 1
;
; ----------------------------------------------------------
; Kinematic element: LINEAR AXIS: Y axis
; ----------------------------------------------------------
N150 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N160 $NK_NAME[KIE_CNTR] = "Y-AXIS"
N170 $NK_NEXT[KIE_CNTR] = "Z-AXIS"
N180 $NK_AXIS[KIE_CNTR] = "Y1"
;
N190 $NK_OFF_DIR[KIE_CNTR,1] = 1.0    ; Y
N200 KIE_CNTR = KIE_CNTR + 1
;
; ----------------------------------------------------------
; Kinematic element: LINEAR AXIS: Z axis
; ----------------------------------------------------------
N210 $NK_TYPE[KIE_CNTR] = "AXIS_LIN"
N220 $NK_NAME[KIE_CNTR] = "Z-AXIS"
N230 $NK_NEXT[KIE_CNTR] = "C-AXIS-OFFSET"
N240 $NK_AXIS[KIE_CNTR] = "Z1"
;
```

Program code

```
N250 $NK_OFF_DIR[KIE_CNTR,2] = 1.0     ; Z
N260 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: OFFSET: C axis
; -----------------------------------------------------------
N270 $NK_TYPE[KIE_CNTR] = "OFFSET"
N280 $NK_NAME[KIE_CNTR] = "C-AXIS-OFFSET"
N290 $NK_NEXT[KIE_CNTR] = "C-AXIS"
;
N300 $NK_OFF_DIR[KIE_CNTR,2] = 600.0     ; Z direction
N310 KIE_CNTR = KIE_CNTR + 1 ;
;
; -----------------------------------------------------------
; Kinematic element: ROTARY AXIS: C axis
; -----------------------------------------------------------
N320 $NK_TYPE[KIE_CNTR] = "AXIS_ROT"
N330 $NK_NAME[KIE_CNTR] = "C-AXIS"
N340 $NK_NEXT[KIE_CNTR] = "C-A-OFFSET"
N350 $NK_AXIS[KIE_CNTR] = "C1"
;
N360 $NK_OFF_DIR[KIE_CNTR,2] = 1.0     ; Z direction
N370 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: OFFSET: C-A axis
; -----------------------------------------------------------
N380 $NK_TYPE[KIE_CNTR] = "OFFSET"
N390 $NK_NAME[KIE_CNTR] = "C-A-OFFSET"
N400 $NK_NEXT[KIE_CNTR] = "A-AXIS"
;
N410 $NK_OFF_DIR[KIE_CNTR,0] = 20.0     ; X direction
N420 $NK_OFF_DIR[KIE_CNTR,1] = -5.0    ; Y direction
N430 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: ROTARY AXIS: A axis
; -----------------------------------------------------------
N440 $NK_TYPE[KIE_CNTR] = "AXIS_ROT"
N450 $NK_NAME[KIE_CNTR] = "A-AXIS"
N460 $NK_NEXT[KIE_CNTR] = "TOOL-REF"
N470 $NK_AXIS[KIE_CNTR] = "A1"
;
N480 $NK_OFF_DIR[KIE_CNTR,0] = 1.0     ; X direction
N490 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: OFFSET: Tool reference
; -----------------------------------------------------------
N500 $NK_TYPE[KIE_CNTR] = "OFFSET"
N510 $NK_NAME[KIE_CNTR] = "TOOL-REF"
N520 $NK_NEXT[KIE_CNTR] = "DOCKING_POINT 1"
;
N530 $NK_OFF_DIR[KIE_CNTR,2] = -200.0     ; Z direction
N540 KIE_CNTR = KIE_CNTR + 1
;
```

Program code

```
; -----------------------------------------------------------
; Kinematic element: Switch 3/1
; -----------------------------------------------------------
N550 $NK_TYPE[KIE_CNTR] = "SWITCH"
N560 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 1"
N570 $NK_NEXT[KIE_CNTR] = "HEAD 1"
N580 $NK_PARALLE[KIE_CNTR] = "DOCKING_POINT 2"
;
N590 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; Index 3
N600 $NK_SWITCH_POS[KIE_CNTR] = 1       ; switch position 1
N610 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: Switch 3/2
; -----------------------------------------------------------
N620 $NK_TYPE[KIE_CNTR] = "SWITCH"
N630 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 2"
N640 $NK_NEXT[KIE_CNTR] = "HEAD 2"
N650 $NK_PARALLE[KIE_CNTR] = "DOCKING_POINT 3"
;
N660 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; Index 3
N670 $NK_SWITCH_POS[KIE_CNTR] = 2       ; switch position 2
N680 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: Switch 3/3
; -----------------------------------------------------------
N690 $NK_TYPE[KIE_CNTR] = "SWITCH"
N700 $NK_NAME[KIE_CNTR] = "DOCKING_POINT 3"
N710 $NK_NEXT[KIE_CNTR] = "HEAD 3"
N720 $NK_PARALLE[KIE_CNTR] = ""
N730 $NK_SWITCH_INDEX[KIE_CNTR] = 3     ; Index 3
N740 $NK_SWITCH_POS[KIE_CNTR] = 3       ; switch position 3
N750 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: OFFSET: HEAD 1
; -----------------------------------------------------------
N760 $NK_TYPE[KIE_CNTR] = "OFFSET"
N770 $NK_NAME[KIE_CNTR] = "HEAD 1"
N780 $NK_NEXT[KIE_CNTR] = ""
;
N790 $NK_OFF_DIR[KIE_CNTR,0] = 100     ; X direction
N800 $NK_OFF_DIR[KIE_CNTR,1] = -30    ; Y direction
N810 KIE_CNTR = KIE_CNTR + 1
;
; -----------------------------------------------------------
; Kinematic element: OFFSET: HEAD 2
; -----------------------------------------------------------
N820 $NK_TYPE[KIE_CNTR] = "OFFSET"
N830 $NK_NAME[KIE_CNTR] = "HEAD 2"
N840 $NK_NEXT[KIE_CNTR] = ""
;
N850 $NK_OFF_DIR[KIE_CNTR,2] = -230     ; Z direction
N860 KIE_CNTR = KIE_CNTR + 1
;
```

Program code

```
; ----------------------------------------------------------
; Kinematic element: OFFSET: HEAD 3
; ----------------------------------------------------------
N870 $NK_TYPE[KIE_CNTR] = "OFFSET"
N880 $NK_NAME[KIE_CNTR] = "HEAD 3"
N890 $NK_NEXT[KIE_CNTR] = ""
;
N900 $NK_OFF_DIR[KIE_CNTR,0] = 50     ; X direction
N910 $NK_OFF_DIR[KIE_CNTR,1] = -20    ; Y direction
N920 $NK_OFF_DIR[KIE_CNTR,2] = -90     ; Z direction
N930 KIE_CNTR = KIE_CNTR + 1
End
```

5.5 Data lists

5.5.1 Machine data

5.5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
16800	ROOT_KIN_ELEM_NAME	Name of the first element in the kinematic chain
18880	MM_MAXNUM_KIN_CHAIN_ELEM	Maximum number of elements for kinematic chains
18882	MM_MAX_NUM_KIN_SWITCHES	Maximum number of switches for kinematic chains

5.5.2 System variables

Identifier	Description
$NK_NAME	Name of the kinematic element
$NK_NEXT	Name of the next element in the kinematic chain
$NK_PARALLEL	Name of the first element of a parallel branching kinematic chain before the element
$NK_TYPE	Type of the kinematic element
$NK_OFF_DIR	Depends on $NK_TYPE: Offset or direction vector
$NK_AXIS	Name of the assigned machine axis or object name
$NK_A_OFF	Work offset in the element for linear or rotary axes
$NK_SWITCH_INDEX	Index i, via which the switch is addressed using system variable $NK_SWITCH[<i>]
$NK_SWITCH_POS	Switch position CLOSED
$NK_SWITCH	Switch variable

K8: Geometric machine modeling 6

6.1 Function description

6.1.1 Features

This section describes how the geometry of machine parts can be mapped based on protection areas and parameterized in the control using system variables for NC functions such as "collision avoidance".

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.

By assigning a protective area to an element of the kinematic chain (Page 229) described in the previous section, the position and motion of the machine part can be clearly described within the machine space.

Note

Graphic editor

As an alternative to writing the system variables in a part program, the machine can be modeled from the SINUMERIK Operate user interface:

Operating area: "Commissioning" > "NC" > "Machine model"

Changes to the machine model

The direct changes to the machine model at the system variables only become visible at the user interface after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

Changes to the machine model made at the user interface are immediately accepted in the system variables of the NC. The changes only become active after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

Protection area

The protection areas are the central element when geometrically modeling a machine. The geometric dimensions of a machine part, its reference to the kinematic chain and additional general features are described by a protection area.

A protection area has the following parameters:

- Name of the protection area
- Name of the kinematic element to which the protection area is assigned
- Type of the protection area
- Name of the first protection area element

- Color and transparency of the protection area
- Detail level for the protection area
- Number of NC/PLC interface bits of the protection area
- Initialization status of the protection area
- Address of the geometric data of the machine element to the protected
  (only relevant for automatic protection areas)

Each parameter is mapped by a system variable. The individual parameters and/or system variables are described in detail in Chapter "System variables: Protection areas (Page 273)".

Kinematic chain

The corresponding protection area is assigned to an element of the kinematic chain (Page 229) in order to map the position and motion of a machine part. The geometric data of the protection area relates then to the local coordinate system of this kinematic element.

Type of the protection area

The following types of protection areas are available:

- Machine protection areas (type: "MACHINE")
  Machine protection areas are used for general machine modeling. They are used to map stationary and moving machine parts whose geometry is defined once upon commissioning and no longer changes when the machine is in operation.
- Automatic tool protection areas (type: "TOOL")
  Automatic tool protection areas are used to map tools. The geometry of the tool is not provided directly with this but is produced automatically by the control when the tool is activated.
  See section "Automatic tool protection areas (Page 266)".

Protection area element

Using a protection area element, the geometrical element used is described regarding its geometrical and general properties.

A protection area element has the following parameters:

- Name of the protection area element
- Name of the following protection area element
- Name of the following protection area element parallel to $NP_NEXT
- Color and transparency of the protection area element
- Detail level for the protection area element
- Use of the protection area element
- Type of the protection area element
- File name of the STL file that contains the geometric data of the protection area element
  (only relevant for type "FILE")
- Geometrical parameters of the protection area body
  (only relevant for types, "BOX", "SPHERE" or "CYLINDER")

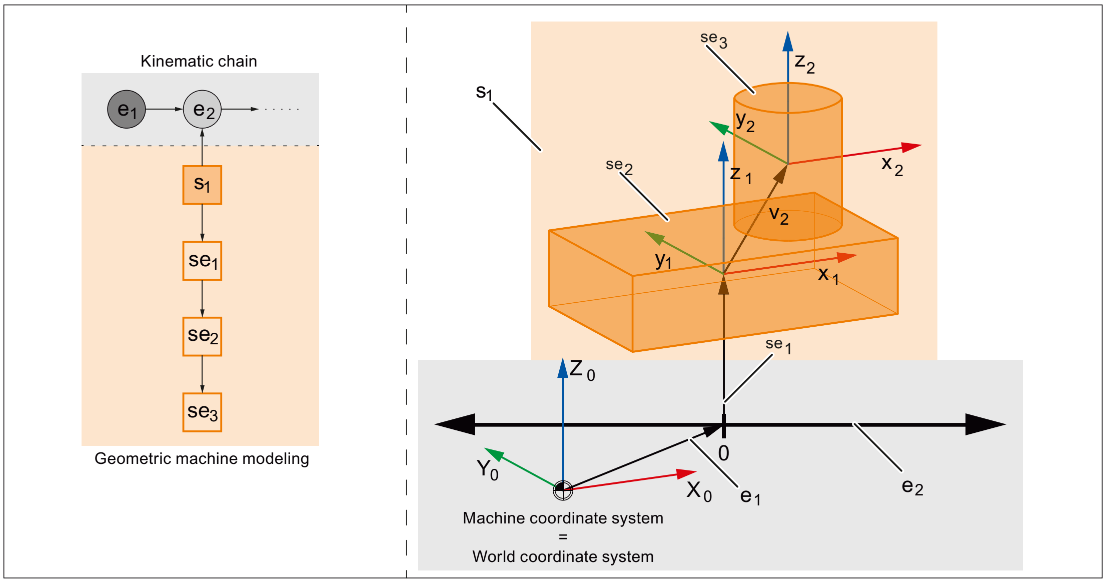
- Offset vector of the protection area element local coordinate system
- Direction vector for the rotation of the protection area element local coordinate system
- Angle for the rotation of the protection area element local coordinate system

Each parameter is mapped by a system variable. The individual parameters or system variables are described in detail in:

- Chapter "System variables: Protection area elements for machine protection areas (Page 285)"
- Chapter "System variables: Protection area elements for automatic tool protection areas (Page 306)"

Protection area, protection area elements and kinematic chain

Using an example of a protection area with two protection area elements, the following diagram shows the interrelationship between the protection area, its protection area elements and the assignment to an element of the kinematic chain.



e_1	Kinematic element 1, type "OFFSET", continuous offset
e_2	Kinematic element 2, type "AXIS_LIN", machine axis AX1
s_1	Protection area
se_1	Protection area element 1, type "FRAME", offset
se_2	Protection area element 2, type "BOX"
se_3	Protection area element 3, type "CYLINDER"

Figure 6-1 Protection area, protection area elements and kinematic chain

6.1.2 Automatic tool protection areas

Unlike machine protection areas whose geometry is defined once during the machine modeling and then no longer changes, the geometry of a tool protection area can change with every tool change. As a result the geometry of an automatic tool protection area is not described directly when the machine model is created, and instead the address (magazine number, magazine location, etc.) is provided under which the tool data is stored. The following actions are then carried out automatically by the control.

1. The tool modeling (see "Tool modeling" section below) produces an STL file.
2. A protection area element of the "FILE" type is produced and assigns the STL file.
3. The protection area element is assigned to the protection area (type "TOOL").

Tool definition independent of the tool mounting position

Normally the parameter "$NP_1ST_PROT (Page 277)" remains empty for the definition of a tool protection area. The name of the protection area element is only entered when the tool is activated by the control (see above).

In order for there to be a tool definition independently of the tool mounting position, a protection area element of the "FRAME" type can be assigned via the parameter "$NP_1ST_PROT" (transformation element). The transformations to align the tool can be carried out via this additional element. When a tool is activated the name of the internal protection area element is entered in the parameter "$NP_NEXT (Page 287)" of the transformation element by the control.

The following rules must be observed:

- The transformation element may only be of the "FRAME" type.
- Only one transformation element may be used for each tool protection area.
- The parameter "$NP_NEXTP" of the transformation element is not evaluated.

Tool reference point

The position of the tool reference point in the machine model is determined by the kinematic element to which the tool protection area is assigned. In addition the tool reference point can be offset within the tool protection area by an optional transformation element.

Kinematic transformations

The tool reference point may only be determined via the kinematic chain when defining a kinematic transformation. Offsets by the transformation element of the tool protection area are not taken into account.

NOTICE
Determination of the tool reference point for kinematic transformations
Offsets of the tool reference point by the transformation element of the tool protection area are not taken into account by kinematic transformations.

Tool modeling

The model of a tool is created by the control heuristically from the tool data. The tool data used for this (L1, L2, L3, R) is always the the overall dimensions resulting from the individual components, e.g. length plus wear, as is also entered in the program processing for tool offset.

Programmable tool offsets

Programmable tool offsets such as `OFFN` (offset to the programmed contour) are not taken into account as they may change in any block, even without a tool change.

Tool type-dependent model generation

The following tool types are distinguished for the model generation:

- Milling tool and all other tools that are neither turning tools or grinding tools
  - Modeling
    The tool is modeled by a cylinder with the height L1 and radius R. If the length L1 is negative the absolute value of L1 is used for the cylinder height. The sign L1 is taken into account when positioning the cylinder in the machine model. The cylinder axis is parallel to L1.
    Tool type 110 (ball end mill cutter) and 111 (face cutter) are modeled using half a sphere or sphere segments.
    If the radius is negative the absolute value of the radius is used. If the value for the radius is less than 1/3 mm, a radius of 1/3 mm is used.
  - Positioning
    The cylinder is positioned in the machine model using the tool length components L2 and L3.
    For milling tools (tool type 100 ... 199) and drilling tools (tool type 200 ... 299) on turning machines, the cylinder is positioned using the cutting edge position.
    Precondition: Cutting edge position == 5 ... 8
- Grinding tools
  - Modeling
    Grinding tools (grinding wheel, tool type 400 ... 499) are modeled by a cylinder with the tool length as the radius and double the tool radius as the height.
  - Positioning
    The cylinder is positioned in the machine model using the tool lengths L1, L2 and L3.
- Turning tools
  In the case of turning tools only the cutting tips are taken into account in the machine model but not their connection to the tool reference point.
  The following data is taken into account when modeling a cutting tip:
  - Tool type
  - Cutting edge position
  - Cutting edge radius
  - Clearance angle
  - Holder angle
  - Cutting tip length
  - Cutting tip width
  - Tip thickness (assumption: tip thickness = 10% tip length)

Tool model

A tool is modeled as standard with an accuracy of one third of the collision tolerance (Page 323). The geometric data of the modeled tool is stored in an internal file in STL format:

- Directory: _N_PROT3D_DIR/_N_TOOL_DIR
- Identifiers: Name of the associated protection area with prefix _N_ and ending with _STL

The coordinate system of the geometric data always has its origin at the point from which the tool length offsets point to the tip of the tool.

System variable

All parameters for an automatic tool protection area can be read via System variable (Page 285).

6.2 Commissioning

6.2.1 General

6.2.1.1 Overview

The commissioning of the "Collision avoidance" function is performed using:

- Machine data
  - Specifications for the quantity structure of protection areas, protection area elements, NC/PLC interface signals, triangles for the geometry modelling
  - Creation mode of the machine model
  - Creation mode for automatic tool protection areas
- System variables
  - Parameterization of the protection areas
  - Parameterization of the protection area elements of a protection area

6.2.1.2 Structure of the system variables

The system variables are structured according to the following scheme:

- **$NP_**<name>[<index_1>]
- **$NP_**<name>[<index_1>, <index_2>]

General

The system variables to describe the protection areas or protection area elements have the following properties:

- Prefix: **$NP_**, (N for NC, P for protection).
- They can be read and written via NC programs.
- They 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

System variables for protection areas

The individual protection areas are addressed via index_1. Index 0 → 1st protection area, index 1 → 2nd protection area, ... m → (m+1) protection area, where m = ($MN_MM_MAXNUM_3D_PROT_AREAS - 1)

All system variables of a protection area have the same index.

System variables for protection areas elements

The individual protection area elements are addressed via index_1. Index 0 → 1st protection area element, index 1 → 2nd protection area element, ... n → (n+1)th protection area element, where n = ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)

All system variables of a protection area element have the same index.

Index_2

Depending on the respective system variables, index_2 has different meanings.

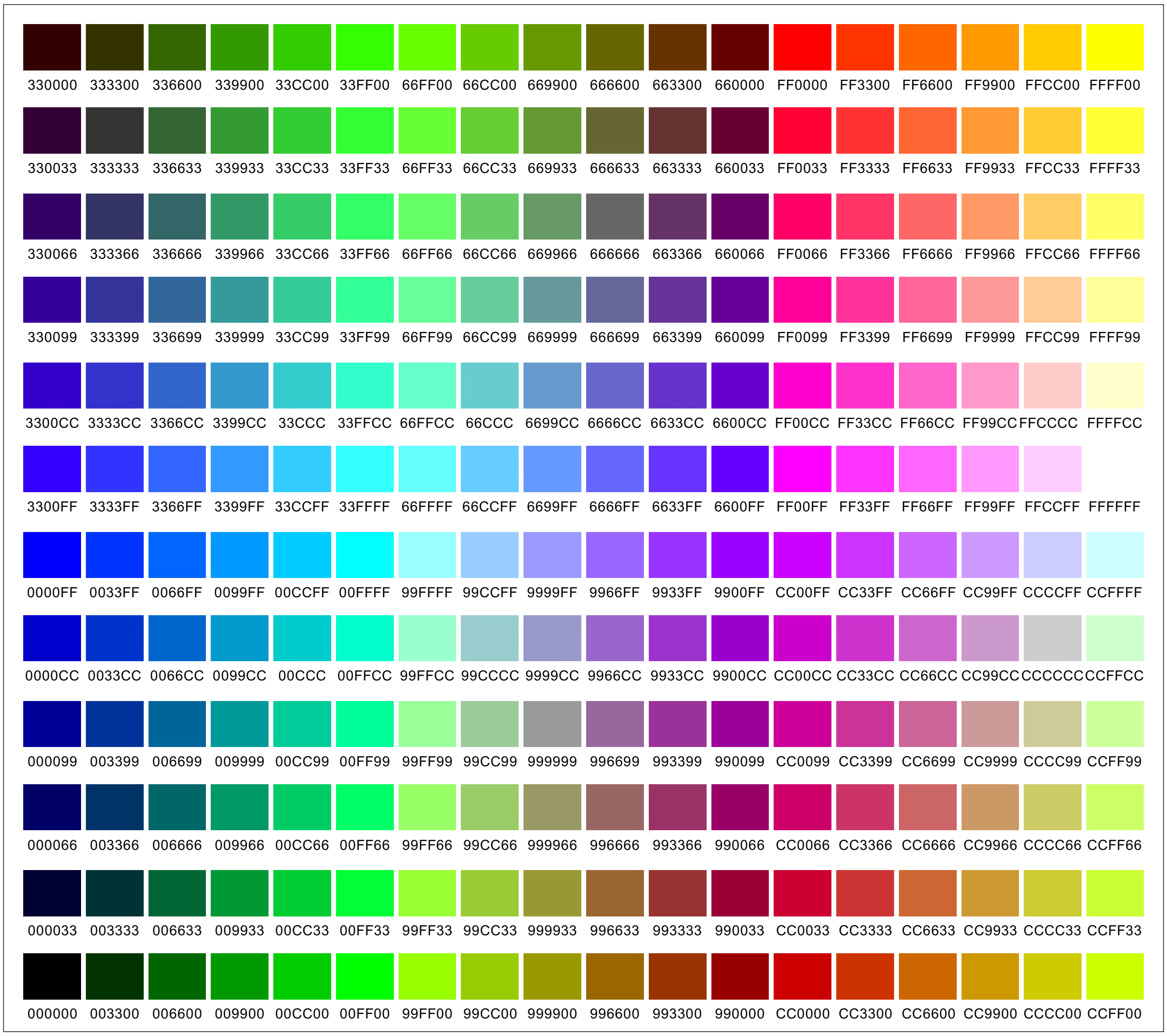
See also

Deletion of components (DELOBJ) (Page 253)

6.2.1.3 Color chart

The following color chart provides an overview of the RGB color values and the associated colors. A RGB color value comprises three bytes. One byte per color:

3rd byte	2nd byte	1st byte
Color value for red	Color value for green	Color value for blue
0 - 255_D or 0 - FF_H	0 - 255_D or 0 - FF_H	0 - 255_D or 0 - FF_H



330000	333300	336600	339900	33CC00	33FF00	66FF00	66CC00	669900	666600	663300	660000	FF0000	FF3300	FF6600	FF9900	FFCC00	FFFF00
330033	333333	336633	339933	33CC33	33FF33	66FF33	66CC33	669933	666633	663333	660033	FF0033	FF3333	FF6633	FF9933	FFCC33	FFFF33
330066	333366	336666	339966	33CC66	33FF66	66FF66	66CC66	669966	666666	663366	660066	FF0066	FF3366	FF6666	FF9966	FFCC66	FFFF66
330099	333399	336699	339999	33CC99	33FF99	66FF99	66CC99	669999	666699	663399	660099	FF0099	FF3399	FF6699	FF9999	FFCC99	FFFF99
3300CC	3333CC	3366CC	3399CC	33CCC	33FFCC	66FFCC	66CCC	6699CC	6666CC	6633CC	6600CC	FF00CC	FF33CC	FF66CC	FF99CC	FFCCCC	FFFFCC
3300FF	3333FF	3366FF	3399FF	33CCFF	33FFFF	66FFFF	66CCFF	6699FF	6666FF	6633FF	6600FF	FF00FF	FF33FF	FF66FF	FF99FF	FFCCFF	FFFFFF
0000FF	0033FF	0066FF	0099FF	00CCFF	00FFFF	99FFFF	99CCFF	9999FF	9966FF	9933FF	9900FF	CC00FF	CC33FF	CC66FF	CC99FF	CCCCFF	CCFFFF
0000CC	0033CC	0066CC	0099CC	00CCC	00FFCC	99FFCC	99CCCC	9999CC	9966CC	9933CC	9900CC	CC00CC	CC33CC	CC66CC	CC99CC	CCCCCC	CCFFCC
000099	003399	006699	009999	00CC99	00FF99	99FF99	99CC99	999999	996699	993399	990099	CC0099	CC3399	CC6699	CC9999	CCCC99	CCFF99
000066	003366	006666	009966	00CC66	00FF66	99FF66	99CC66	999966	996666	993366	990066	CC0066	CC3366	CC6666	CC9966	CCCC66	CCFF66
000033	003333	006633	009933	00CC33	00FF33	99FF33	99CC33	999933	996633	993333	990033	CC0033	CC3333	CC6633	CC9933	CCCC33	CCFF33
000000	003300	006600	009900	00CC00	00FF00	99FF00	99CC00	999900	996600	993300	990000	CC0000	CC3300	CC6600	CC9900	CCCC00	CCFF00

6.2.2 Machine data

6.2.2.1 Maximum number of protection areas

The maximum number of all types of parameterizable protection areas (Page 276) is specified with the machine data.

MD18890 $MN_MM_MAXNUM_3D_PROT_AREAS = <number>

6.2.2.2 Maximum number of protection area elements for machine protection areas

The maximum number of parameterizable protection area elements for machine protection areas is specified using machine data ($NP_PROT_TYPE == "MACHINE" (Page 276)).

MD18892 $MN_MM_MAXNUM_3D_PROT_AREA_ELEM = <number>

6.2.2.3 Maximum number of protection area elements for automatic tool protection areas

The maximum number of protection area elements for automatic tool protection areas is specified with the machine data. For each automatic tool protection area the control precisely creates one protection area element. As a consequence, using the value parameterized here, the maximum possible number of parameterizable automatic tool protection areas ($NP_PROT_TYPE == "TOOL" (Page 276)) is also simultaneously defined.

MD18893 $MN_MM_MAXNUM_3D_T_PROT_ELEM = <number>

6.2.2.4 Maximum number of NC/PLC interface signals for the preactivation of protection areas

The machine data communicates to the control the number of NC/PLC interface signals of interface DB10, DBX234.0 - DBX241.7 actually used (collision avoidance: activate protection area) with the machine data. The number of interface signals used increases the memory space required for each part program block. Counting the number of used NC/PLC interface signal starts at DB10, DBX234.0

MD18897 $MN_MM_MAXNUM_3D_INTERFACE_IN = <number>

References:
A detailed description of the interface signals is provided in the NC Variables and Interface Signals Parameter Manual.

6.2.2.5 Maximum number of triangles for machine protection areas

With the machine data, the maximum number of triangles for protection area bodies ($NP_TYPE == "FILE") of machine protection areas ($NP_PROT_TYPE == "MACHINE" (Page 276)) to be provided by the control is defined.

MD18895 $MN_MM_MAXNUM_3D_FACETS = <number>

6.2.2.6 Maximum number of triangles for automatic tool protection areas

With the machine data, the maximum number of triangles for protection area bodies of automatic tool protection areas to be provided by the control is defined.

MD18894 $MN_MM_MAXNUM_3D_FACETS_INTERN = <number>

The control system automatically models the protection area bodies based on the geometric data of the tool active at the time of creation. The number of triangles correspondingly increases:

- as the geometric complexity of the tool increases.
- as the parameterized collision tolerance decreases.

Note

Protection area bodies and collision tolerance

The protection area bodies of automatic tool protection areas are, as standard, created by the control with a precision of 1/3 of the collision tolerance (Page 323).

6.2.2.7 Creation mode for automatic tool protection areas

The machine data defines how the control creates protection area bodies of automatic tool protection areas.

MD18899 $MN_PROT_AREA_TOOL_MASK = <mode>

<Mode>		
Bit	**Val-ue**	**Meaning**
0	0	Do not apply heuristic model generation
	1	Heuristic model generation using tool data

6.2.3 System variables: Protection areas

6.2.3.1 Overview

A protection area is parameterized with the following system variables:

Name	Meaning
$NP_PROT_NAME	Name of the protection area
$NP_CHAIN_ELEM	Name of the kinematic element to which the protection area is as-signed
$NP_PROT_TYPE	Type of the protection area
$NP_1ST_PROT	Name of the first protection area element
$NP_PROT_COLOR	Color and transparency of the protection area.
$NP_PROT_D_LEVEL	Detail level for the protection area

Name	Meaning
$NP_BIT_NO	Number of NC/PLC interface bits of the protection area
$NP_INIT_STAT	Initialization status of the protection area
$NP_INDEX	Address of the geometric data of the machine element to be protected (only relevant for automatic protection areas)

The system variables are described in detail in the following sections.

Note

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the protection areas. To do this, set the system variables of the protection areas to their default values with the DELOBJ() (Page 253) function.

Change system variable values

If the value of one of the system variables listed above is changed, the change becomes immediately visible at the user interface, e.g. SINUMERIK Operate. The machine model of the NC is only updated after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

6.2.3.2 $NP_PROT_NAME

Function

Enter the NC-wide unique protection area name in the system variable. The protection area is referenced via this name, e.g. by a protection area element. The name is also displayed in the graphical editor of SINUMERIK Operate.

Syntax

```
$NP_PROT_NAME[<m>] = "<name>"
```

Meaning

`$NP_PROT_NAME`:	Name of the protection area	
	Data type:	STRING
	Default value:	"" (empty string)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<name>`:	Name of the protection area	
	Data type:	STRING

Example

The 6th protection area is assigned the name "Spindle":

Program code	Comment

`N100 $NP_PROT_NAME[5] = "Spindle"`	`; 6. Protection area,`
	`; name = "Spindle"`

6.2.3.3 $NP_CHAIN_ELEM

Function

Enter the name of the kinematic element (Page 237) to which the protection area will be connected in the system variable.

Note

Reference coordinate system

The geometric data of the protection area, starting from the first protection area element ($NP_1ST_PROT (Page 277)), refer to the local coordinate system of the kinematic element, with which the protection area is connected.

Syntax

```
$NP_CHAIN_ELEM[<m>] = "<name>"
```

Meaning

`$NP_CHAIN_ELEM`:	Name of the kinematic element to which the protection area will be connected	
	Data type:	STRING
	Default value:	"" (empty string)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<name>`:	Name of the kinematic element	
	Data type:	STRING
	Range of values:	Names parameterized in $NK_NAME (Page 237)

Example

The 6th protection area is connected to the kinematic element with the name "Z axis":

Program code	Comment

`N100 $NP_CHAIN_ELEM[5] = "Z ax-`
`is"`
	`; 6. Protection area,`
	`; name of the kin. element: "Z axis"`

6.2.3.4 $NP_PROT_TYPE

Function

The protection area type should be entered in the system variable:

- Machine protection area: "MACHINE"
  The protection area body is defined using one or several protection area elements.
  $NP_1ST_PROT (Page 277) refers to the first protection area element.
- Automatic tool protection area: "TOOL"
  The control calculates the dimensions of the protection area body from the tool data.
  $NP_INDEX (Page 283) refers to the tool.
- Workholder protection area: "FIXTURE"
- Workpiece protection area: "WORKPIECE"

Syntax

```
$NP_PROT_TYPE[<m>] = "<type>"
```

Meaning

`$NP_PROT_TYPE`:	Type of the protection area	
	Data type:	STRING
	Range of values:	"MACHINE", "TOOL", "FIXTURE", "WORKPIECE"
	Default value:	"" (empty string)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<Type>`:	Type	
	Data type:	STRING

Example

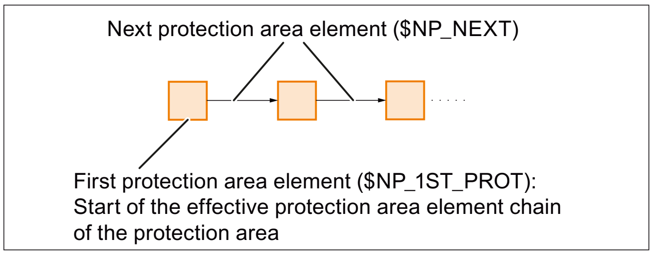
The 6th protection area is a machine protection area:

Program code	**Comment**
`N100 $NP_PROT_TYPE[5] = "MACHINE"`	`; 6. Protection area,`
	`; type = "MACHINE"`

6.2.3.5 $NP_1ST_PROT

Function

Enter the name of the first protection area element (Page 286) in the system variable.


Next protection area element ($NP_NEXT)
First protection area element ($NP_1ST_PROT):
Start of the effective protection area element chain
of the protection area

Syntax

```
$NP_1ST_PROT[<m>] = "<name>"
```

Meaning

`$NP_1ST_PROT`:	Name of the first protection area element of the protection area	
	Data type:	STRING
	Range of values:	Names parameterized in $NP_NAME (Page 286)
	Default value:	"" (empty string)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<name>`:	Protection area name	
	Data type:	STRING

Automatic tool protection areas, $NP_PROT_TYPE == "TOOL"

For automatic tool protection areas, only the following values are permissible for $NP_1ST_PROT:

- "" (empty string)
- Name of a protection area element, type "FRAME"

Behavior for value == "" (empty string)

When activating the associated tool, the control creates a protection area element for the tool with a unique, internal name and protection area body generated from the geometric data of the tool. The name is assigned system variable $NP_1ST_PROT.

- Protection area "TOOL" : $NP_1ST_PROT = "<internal name>"

The coordinates of the tool protection area refer to the local coordinate system of the kinematic element to which it is assigned.

Behavior for value == name of a protection area element, type "FRAME"

When activating the associated tool, the control creates a protection area element for the tool with a unique, internal name and protection area body generated from the geometric data of the tool. The name is assigned a system variable $NP_NEXT of the protection area element, type "FRAME", to which $NP_1ST_PROT refers.

- Protection area "TOOL" : $NP_1ST_PROT = "WKZ_Frame" →
  - Protection area element "WKZ_Frame" : $NP_NEXT = "<internal name>"

The coordinates of the tool protection area refer to the local coordinate system of the protection area element, type "FRAME".

Possible applications: Tool definition independent of the position where it is mounted on the machine.

Example

The 1st protection area element which makes up the 6th protection area has the name "Spindle head":

Program code	Comment
`N100 $NP_1ST_PROT[5] = "Spindle head"`	`; 6. Protection area,`
	`; 1. Protection area element = "spindle box"`

6.2.3.6 $NP_PROT_COLOR

Function

Enter the protection area element-specific value for alpha/transparency and color (ARGB) in the system variable. This value is used for the display of the protection area or protection area element on the user interface. If a separate value is entered for a protection area element in $NP_COLOR (Page 289), this is used for the display of the protection area element.

Structure

Alpha/transparency and color are specified as a double word in hexadecimal format: **AARRGGBB**$_H$

- 1. - 3. Byte: RGB color value. See Chapter "Color chart (Page 271)".
- 4. Byte: Alpha channel or transparency value

	Byte	Meaning	Range of values
OC	1	Blue	0 - 255$_D$ or 0 - FF$_H$
GG	2	Green	
RR	3	Red	
AA	4	Alpha channel or transparency 1)	
1) 0 = transparent or not visible, 255$_D$ = FF$_H$ = not transparent or filled			

Syntax

```
$NP_PROT_COLOR[<m>] = <value>
```

Meaning

`$NP_PROT_COLOR`:	Alpha/transparency and color value of the protection area	
	Data type:	DWORD
	Range of values:	00000000_H - $FFFFFFFF_H$
	Default value:	0000000_H (black, not visible)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<value>`:	Transparency and color value	
	Data type:	DWORD

Example

The 6th protection area is to be displayed half transparent and in a green-blue color on the user interface:

- AA = $7F_H$ = 127_D ≙ 50% transparency
- RR (red) = 00 ≙ no red component
- GG (green) = FF_H = 255_D ≙ 100% green color component
- BB (blue) = 33_H = 51_D ≙ 20% blue color component

Program code	Comment

`N100 $NP_PROT_COLOR[5] = 'H7F00FF33'`	`; 6. Protection area,`
	`; alpha/transparency and color value = H7F00FF33`

6.2.3.7 $NP_PROT_D_LEVEL

Function

The detail level as of which the protection area or the protection area elements are displayed on the user interface is specified via the system variable. If a separate value is entered for a protection area element in $NP_D_LEVEL (Page 290), this is used for the display of the protection area element.

Detail level

- Lowest detail level: 0
- Highest detail level: 3

If the detail level x is selected for the display on the HMI, all protection areas and protection area elements are displayed for which the following applies for the detail level D: D ≤ x

Syntax

```
$NP_PROT_D_LEVEL[<m>] = <value>
```

Meaning

`$NP_PROT_D_LEVEL`:	Detail level for the protection area	
	Data type:	INT
	Range of values:	0 ≤ D ≤ 3
	Default value:	0
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<value>`:	Detail level	
	Data type:	INT

Example

The 6th protection area is to be displayed as of detail level 3:

Program code	Comment
`N100 $NP_PROT_D_LEVEL[5] = 3`	`; 6. Protection area,`
	`; detail level = 3`

6.2.3.8 $NP_BIT_NO

Function

Enter the bit number (0, 1, 2, ... 63) of the NC/PLC interface signal with which the protection area is to be connected in system variable $NP_BIT_NO. If the protection area is not to be connected to an NC/PLC interface signal, enter the value -1.

NC/PLC interface

Activation / deactivation of the protection area can be requested via NC/PLC interface signals from the PLC user program or this can be used for feedback on the current status to the PLC user program:

- Requirement: DB10, DBX234.0 - DBX241.7
- Feedback: DB10, DBX226.0 - DBX233.7

Precondition

The status of the protection area must be "preactivated" or "PLC-controlled" in order that the assigned NC/PLC interface signal of the protection area is taken into account:

$NP_INIT_STAT (Page 282) == "P" (preactivated or PLC-controlled)

Syntax

```
$NP_BIT_NO[<m>] = <bit number>
```

Meaning

`$NP_BIT_NO`:	Bit number of the interface signal to activate/deactivate the protection area	
	Data type:	INT
	Range of values:	-1, 0, 1, 2, ... ($MN_MM_MAXNUM_3D_INTERFACE_IN - 1)
	Default value:	-1 (no interface signal selected)
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_3D_MAXNUM_PROT_AREAS - 1)
`<Bit number>`:	Bit number (0, 1, 2, ... 63) of the 64-bit interface	
	Data type:	INT

Example

The 6th protection area is assigned to the 18th bit of interface (DB10.DBX236.1):

Program code	Comment

`N100 $NP_BIT_NO[5] = 17`	`; 6. Protection area,`
	`; DB10.DBX236.1`

Assignment: Bit number to the interface signal

Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)
0	DBX234.0	DBX226.0	**8**	DBX235.0	DBX227.0	**16**	DBX236.0	DBX228.0	**24**	DBX237.0	DBX229.0
1	DBX234.1	DBX226.1	**9**	DBX235.1	DBX227.1	**17**	DBX236.1	DBX228.1	**25**	DBX237.1	DBX229.1
2	DBX234.2	DBX226.2	**10**	DBX235.2	DBX227.2	**18**	DBX236.2	DBX228.2	**26**	DBX237.2	DBX229.2
3	DBX234.3	DBX226.3	**11**	DBX235.3	DBX227.3	**19**	DBX236.3	DBX228.3	**27**	DBX237.3	DBX229.3
4	DBX234.4	DBX226.4	**12**	DBX235.4	DBX227.4	**20**	DBX236.4	DBX228.4	**28**	DBX237.4	DBX229.4
5	DBX234.5	DBX226.5	**13**	DBX235.5	DBX227.5	**21**	DBX236.5	DBX228.5	**29**	DBX237.5	DBX229.5
6	DBX234.6	DBX226.6	**14**	DBX235.6	DBX227.6	**22**	DBX236.6	DBX228.6	**30**	DBX237.6	DBX229.6
7	DBX234.7	DBX226.7	**15**	DBX235.7	DBX227.7	**23**	DBX236.7	DBX228.7	**31**	DBX237.7	DBX229.7

Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)
32	DBX238.0	DBX230.0	**40**	DBX239.0	DBX231.0	**48**	DBX240.0	DBX232.0	**56**	DBX241.0	DBX233.0
33	DBX238.1	DBX230.1	**41**	DBX239.1	DBX231.1	**49**	DBX240.1	DBX232.1	**57**	DBX241.1	DBX233.1
34	DBX238.2	DBX230.2	**42**	DBX239.2	DBX231.2	**50**	DBX240.2	DBX232.2	**58**	DBX241.2	DBX233.2

Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)	Bit →	DB10, (PLC → NC)	DB10, (NC → PLC)
35	DBX238.3	DBX230.3	43	DBX239.3	DBX231.3	51	DBX240.3	DBX232.3	59	DBX241.3	DBX233.3
36	DBX238.4	DBX230.4	44	DBX239.4	DBX231.4	52	DBX240.4	DBX232.4	60	DBX241.4	DBX233.4
37	DBX238.5	DBX230.5	45	DBX239.5	DBX231.5	53	DBX240.5	DBX232.5	61	DBX241.5	DBX233.5
38	DBX238.6	DBX230.6	46	DBX239.6	DBX231.6	54	DBX240.6	DBX232.6	62	DBX241.6	DBX233.6
39	DBX238.7	DBX230.7	47	DBX239.7	DBX231.7	55	DBX240.7	DBX232.7	63	DBX241.7	DBX233.7

References

A detailed description of the interface signals is provided in the NC Variables and Interface Signals Parameter Manual

6.2.3.9 $NP_INIT_STAT

Function

Enter the protection area initialization status in the system variable.

The status of a protection area is set to the parameterized initialization status in the following situations:

- When the control ramps up
- When calling the PROTA (Page 332) function after the protection area has been recreated during operation by writing the protection-area-specific system variables
- When calling the PROTA (Page 332) function with parameter "R"
- When calling the PROTS (Page 333) function with parameter "R"

Syntax

```
$NP_INIT_STAT[<m>] = "<status>"
```

Meaning

`$NP_INIT_STAT`:	Initialization status of the protection area	
	Data type:	STRING
	Range of values:	"A", "a", "I", "i", "P", "p"
	Value	**Protection area status**
	"A"or "a"	Activated
	"I"or "i"	Inactive
	"P"or "p"	Preactivated or PLC-controlled1)
	Default value:	"I" (inactive)

`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREAS - 1)
`<Status>`:	Initialization status	
	Data type:	STRING
1) The activation/deactivation is performed via: DB10.DBX234.0 - DBX241.7		

Example

The initialization status of the 6th protection area is set to "P" (preactivated or PLC-controlled):

Program code	Comment
`N100 $NP_INIT_STAT[5] = "P"`	`; 6. Protection area,`
	`; initialization status = "P"`

The current status depends on the state of the interface signal parameterized in $NP_BIT_NO (Page 280).

6.2.3.10 $NP_INDEX

Function

For automatic protection areas ($NP_PROT_TYPE (Page 276)), the address, under which the geometric data of the machine part, tool, etc. to be protected is saved, should be entered in the system variable. The control then automatically creates the geometrical dimensions of the protection area from the geometric data.

Example

For example, for an automatic tool protection area ($NP_PROT_TYPE == "TOOL") the geometrical dimensions of the protection area are created based on the tool data.

Syntax

```
$NP_INDEX[<m>,<i>] = <value>
```

Meaning

`$NP_INDEX`:	Address of the geometric data for the automatic protection areas	
	Data type:	INT[3]
`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREAS - 1)

`<i>`:	Index The meaning of the system variables `$NP_INDEX[<m>,<i>]`, with i = 0, 1, 2, ... is dependent on the type ($NP_PROT_TYPE) of the automatic protection area. See type-specific tables.	
	Data type:	INT
	Range of values:	0, 1, 2
`<value>`:	Address	
	Data type:	INT

Type: Automatic tool protection area ($NP_PROT_TYPE == "TOOL")

<i>	<value>	
	When tool management is active	**Without tool management**
0	Circular magazine: **Tool location number**	**Spindle number**
	No circular magazine: **Spindle number**	
1	**Magazine number**	---
2	**TOA area** 1)	
1) TOA area "1" can be addressed with 0 as well as also with 1.		

Example

The 6th protection area is an automatic tool protection area ($NP_PROT_TYPE == "TOOL"). The geometrical dimensions of the protection area should be generated from the geometric data of the tool at the following tool location:

- Tool location number: 1
- Magazine number: 9998 (spindle 1)
- TOA area: 1

Tool management is active.

```
Program code                                       Comment
; The dimensions of the 6th protection area are based on the tool data
; of the tool located at the following position:
N100 $NP_INDEX[5,0] = 1                            ; Tool location number = 1
N110 $NP_INDEX[5,1] = 9998                         ; magazine number = 9998 (spin-
                                                   dle 1)
N120 $NP_INDEX[5,2] = 1                            ; TOA area = 1
```

See also

$NP_PROT_TYPE (Page 276)

6.2.4 System variables: Protection area elements for machine protection areas

6.2.4.1 Overview

A protection area element of a machine protection area is parameterized with the following system variables:

Name	Meaning
$NP_NAME	Name of the protection area element
$NP_NEXT	Name of the following protection area element
$NP_NEXTP	Name of the following protection area element parallel to $NP_NEXT
$NP_COLOR	Color and transparency of the protection area element.
$NP_D_LEVEL	Detail level for the protection area element
$NP_USAGE	Use of the protection area element
$NP_TYPE	Type of the protection area element
$NP_FILENAME	File name of the STL file that contains the geometric data of the protection area element (only relevant for $NP_TYPE == "FILE")
$NP_PARA	Geometric parameters of the protection area body (only relevant for $NP_TYPE == "BOX" or "SPHERE" or "CYLINDER")
$NP_OFF	Offset vector of the protection area element local coordinate system
$NP_DIR	Direction vector for the rotation of the protection area element local coordinate system
$NP_ANG	Angle for the rotation of the protection area element local coordinate system

The system variables are described in detail in the following sections.

Note

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the protection area elements. To do this, set the system variables of the protection area elements to their default values with the DELOBJ() (Page 253) function.

Change system variable values

If the value of one of the system variables listed above is changed, the change becomes immediately visible at the user interface, e.g. SINUMERIK Operate. The machine model of the NC is only updated after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

6.2.4.2 $NP_NAME

Function

Enter the NC-wide unique protection area element name in the system variable. The protection area element is referenced via this name. The name is also displayed in the graphical editor of SINUMERIK Operate.

Syntax

```
$NK_NAME[<n>] = "<name>"
```

Meaning

`$NP_NAME`:	Name of the protection area element	
	Data type:	STRING
	Default value:	"" (empty string)
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<name>`:	Name of the protection area element	
	Data type:	STRING

Example

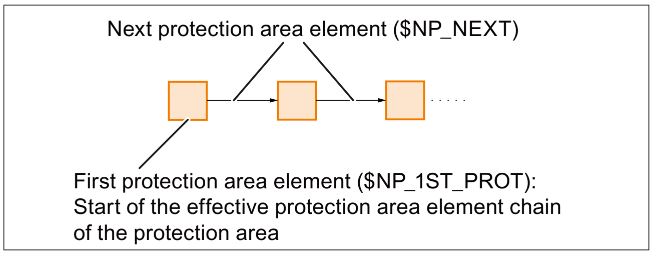
The 19th protection area element is assigned the name "Spindle head":

Program code	Comment
`N100 $NP_NAME[18] = "Spindle head"`	`; 19th protection area element,`
	`; name = "Spindle head"`

6.2.4.3 $NP_NEXT

Function

If a protection area is made up of several protection area elements, they must be linked. To do this, the name of the following protection area element must be entered in the system variable $NP_NEXT in every protection area element.


Next protection area element ($NP_NEXT)
First protection area element ($NP_1ST_PROT):
Start of the effective protection area element chain
of the protection area

If there is no following protection area element, the name must be entered as an empty string "".

Offset and rotation

An offset and/or rotation in the current protection area element ($NP_OFF (Page 302), $NP_DIR (Page 303) and $NP_ANG (Page 305)) affects the following protection area element specified in $NP_NEXT. This means that the specification of the spatial position and orientation of the following protection area element is relative to the current protection area element.

Syntax

```
$NK_NEXT[<n>] = "<name>"
```

Meaning

`$NP_NEXT`:	Name of the following protection area element	
	Data type:	STRING
	Range of values:	All names contained in $NP_NAME (Page 286)
	Default value:	"" (empty string)
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<name>`:	Protection area name	
	Data type:	STRING

Example

The following protection area element with the name "Coolant nozzle 1" is attached to the 19th protection area element:

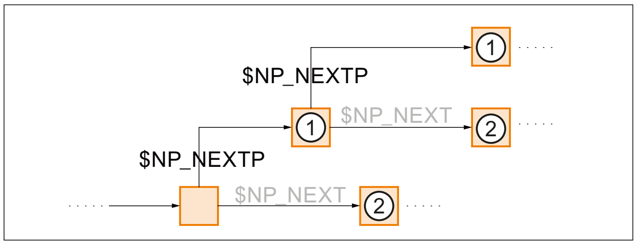
Program code	Comment

`N100 $NP_NAME[18] = "Coolant nozzle 1"`	`; 19th protection area element,`
	`; name of the following element: "Coolant nozzle 1"`

6.2.4.4 $NP_NEXTP

Function

The protection area element chain can be branched via the system variable $NP_NEXTP. To do this, the following protection area elements must be specified in the system variables $NP_NEXT and $NP_NEXTP in a protection area element. These protection area elements are then parallel to one another in two separate subchains.



① Following parallel protection area element

② Following protection area element of the same subchain

Figure 6-2 Protection area elements in parallel subchains

Application example

The separate subchains can be used, for example, to model different machine parts of a protection area for visualization or collision avoidance. Typically "C" (collision avoidance) is specified for the protection area element referred to by $NP_NEXT for use in $NP_USAGE (Page 291), and for the protection area element referred to in $NP_NEXTP, the value “V” (visualization).

Offset and rotation

An offset and/or rotation in the current protection area element ($NP_OFF (Page 302), $NP_DIR (Page 303) and $NP_ANG (Page 305)) affects the following protection area element specified in $NP_NEXTP. This means that the specification of the spatial position and orientation of the following protection area element is relative to the current protection area element.

Syntax

```
$NP_NEXTP[<n>] = "<name>"
```

Meaning

`$NP_NEXTP`:	Name of the branching protection area element	
	Data type:	STRING
	Range of values:	All names contained in $NP_NAME (Page 286)
	Default value:	"" (empty string)
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<name>`:	Protection area name	
	Data type:	STRING

Example

The following parallel protection area element with the name "Coolant nozzle 2" is attached to the 19th protection area element:

Program code	Comment

`N100 $NP_NEXTP[18] = "Coolant nozzle 2"`	`; 19. Protection area element,`
	`; name of the parallel following element: "Coolant nozzle 2"`

6.2.4.5 $NP_COLOR

Function

Enter the protection area element-specific value for alpha/transparency and color (ARGB) in the system variable. This value is used for the display of the protection area element on the user interface. If a specific value is not parameterized for a protection area element, the protection area-specific value from $NP_PROT_COLOR (Page 278) applies.

Structure

Alpha/transparency and color are specified as a double word in hexadecimal format: **AARRGGBB_H**

- 1. - 3. Byte: RGB color value. See Chapter "Color chart (Page 271)".
- 4. Byte: Alpha channel or transparency value

	Byte	Meaning	Range of values
OC	1	Blue	0 - 255_D or 0 - FF_H
GG	2	Green	
RR	3	Red	
AA	4	Alpha channel or transparency [1]	
1) 0 = transparent or not visible, 255_D = FF_H = not transparent or filled			

Syntax

```
$NP_COLOR[<n>] = <name>
```

Meaning

`$NP_COLOR`:	Alpha/transparency and color value of the protection area element	
	Data type:	DWORD
	Range of values:	00000000_H - $FFFFFFFF_H$
	Default value:	0000000_H (black, not visible)
`<m>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<value>`:	Transparency and color value	
	Data type:	DWORD

Example

The 19th protection area is to be displayed half transparent and in a green-blue color on the user interface:

- AA = $7F_H$ = 127_D ≙ 50% transparency
- RR (red) = 00 ≙ no red component
- GG (green) = FF_H = 255_D ≙ 100% green color component
- BB (blue) = 33_H = 51_D ≙ 20% blue color component

Program code	Comment
`N100 $NP_COLOR[18] = 'H7F00FF33'`	`; 19. Protection area,`
	`; alpha/transparency and color value = 'H7F00FF33'`

6.2.4.6 $NP_D_LEVEL

Function

The detail level as of which the protection area element is displayed on the user interface is specified via the system variable. If no value different from the default value is assigned for a protection area element the protection area-specific value takes effect $NP_PROT_D_LEVEL (Page 279).

Detail level

- Lowest detail level: 0
- Highest detail level: 3

If the detail level x is selected for the visualization of the machine model, all protection areas and protection area elements are displayed for which the following applies for the detail level D: D ≤ x

Syntax

```
$NP_D_LEVEL[<n>] = <value>
```

Meaning

`$NP_PROT_D_LEVEL`:	Detail level for the protection area element	
	Data type:	INT
	Range of values:	0 ≤ D ≤ 3
	Default value:	0
`<m>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<value>`:	Detail level	
	Data type:	INT

Example

The 19th protection area is always to be displayed ⇒ detail level 0:

Program code	Comment
`N100 $NP_PROT_D_LEVEL[18] = 0`	`; 19. Protection area,`
	`; detail level = 0`

6.2.4.7 $NP_USAGE

Function

Enter the protection area element usage in the system variable. The usage specifies how the protection area element is to be considered for the collision avoidance.

- Only visualization, no collision calculation
- Only collision calculation, no visualization
- Visualization and collision calculation

Usage	Meaning
Visualization	The protection area element is displayed in the machine model on the SINUMERIK Operate user interface
Collision calculation	The protection area element is also taken into account for the collision calculation

Syntax

```
$NP_USAGE[<n>] = "<value>"
```

Meaning

`$NP_USAGE`:	Use of the protection area element	
	Data type:	CHAR
	Range of values:	"V", "v", "C", "c", "A", "a"
	Value	**Meaning**
	"V" or "v"	Only visualization, no collision calculation
	"C" or "c"	Only collision calculation, no visualization
	"A" or "a"	Visualization and collision calculation
	Default value:	"A"
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<value>`:	Usage	
	Data type:	CHAR

Example

The 19th protection area element is to be displayed at the user interface and included in the collision calculation:

Program code	Comment
`N100 $NP_USAGE[18] = "A"`	`; 19. Protection area,`
	`; usage = "A"`

6.2.4.8 $NP_TYPE

Function

Enter the protection area element type in the system variable.

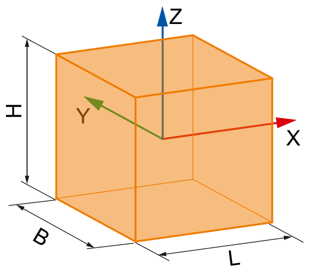
Type: "FRAME"

A protection area element of the "FRAME" type does not contain any bodies, but defines a coordinate transformation of the local coordinate system. The coordinate transformation affects all the following ($NP_NEXT (Page 287)) and/or parallel ($NP_NEXTP (Page 288)) protection area elements. The values of the coordinate transformation are set via:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

No parameters are specified in $NP_PARA (Page 301) for the "FRAME" type.

Type: "BOX"



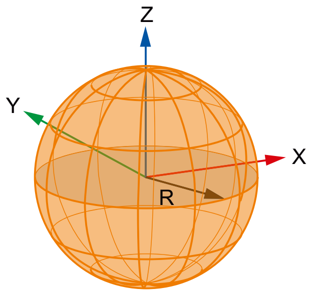
L	Length in the X direction
B	Width in the Y direction
H	Height in the Z direction

A protection area element of the "BOX" type defines a paraxial box in the local coordinate system of the protection area element. The mid-point of the box is at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

The parameters length, width and height are to be entered in $NP_PARA (Page 301)

Type: "SPHERE"



R	Radius of the sphere:

A protection area element of the "SPHERE" type defines a sphere in the local coordinate system of the protection area element. The mid-point of the sphere is at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

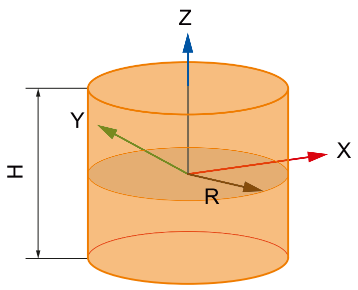
The parameter radius is to be entered in $NP_PARA (Page 301)

Note

Rotation

As the center point of the sphere and the starting point of the direction vector are in the coordinate origin of the local coordinate system of the protection area element, a rotation using the direction vector $NP_DIR (Page 303) and angle of rotation $NP_ANG (Page 305) has no effect on the position of the sphere.

Type: "CYLINDER"



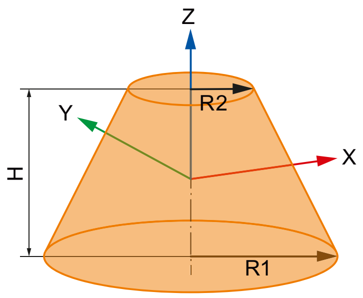
H	Height in the Z direction
R	Radius in the X/Y plane

A protection area element of the "CYLINDER" type defines a cylinder in the local coordinate system of the protection area element. The mid-point of the cylinder is at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

The parameters height and radius are to be entered in $NP_PARA (Page 301)

Type: "CONE"



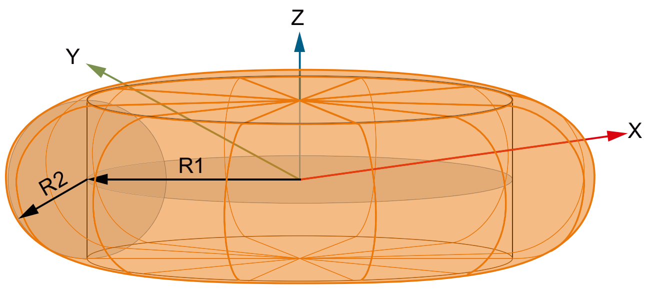
H	Height in the Z direction
R1	Radius 1 in the X/Y plane
R2	Radius 2 in the X/Y plane

A protection area element of the "CONE" type defines a cone in the local coordinate system of the protection area element. The mid-point of the cone (half cone height on the line of symmetry of the cone) is at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

The parameters height, radius 1, and radius 2 must be entered in $NP_PARA (Page 301)

Type: "TORUS"



R1	Major radius (= distance from the center of the circle to the center of the torus in the X/Y plane)
R2	Minor radius (= radius of the circle)

A protection area element of the "TORUS" type defines a filled torus in the local coordinate system of the protection area element. The shape of a torus can most easily be described by a circle revolved around an axis that lies in its plane. In contrast to a normal torus, however, the hole in the middle of a filled torus is filled in. The center of the filled torus lies at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

The parameters radius 1 and radius 2 must be entered in $NP_PARA (Page 301)

Type: "FILE"

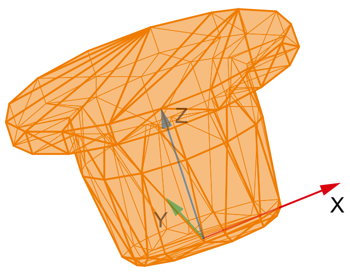


Figure 6-3 Example bodies in STL format

A protection area element of the "FILE" type defines a body whose geometry data is contained in the specified file in STL format (triangular areas), in the local coordinate system of the protection area element. The zero point of the body is at the origin of the local coordinate system. At the same time as the definition of the body, the local coordinate system can be transformed via the following system variables:

- Offset: $NP_OFF (Page 302)
- Rotation direction vector: $NP_DIR (Page 303)
- Angle of rotation: $NP_ANG (Page 305)

The parameter is to be entered in $NP_FILENAME (Page 296):

Syntax

```
$NP_TYPE[<n>] = "<type>"
```

Meaning

`$NP_TYPE`:	Type of the protection area element	
	Data type:	STRING
	Range of values:	"FRAME", "BOX", "SPHERE", "CYLINDER", "CONE", "TORUS", "FILE"
	Default value:	"" (empty string)
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<Type>`:	Type designation	
	Data type:	STRING

Example

The 19th protection area element is a cube:

Program code	Comment

`N100 $NP_TYPE[18] = "BOX"`	`; 19. Protection area element,`
	`; type = "Box"`

6.2.4.9 $NP_FILENAME

Function

For protection area elements of the "FILE" type ($NP_TYPE), enter the name of the file that describes the geometry data of the protection area element in the system variable.

The following file types are currently possible:

- STL files
- NPP files

STL files

An STL file (file extension .STL) must contain a description of the geometry data of a 3D body using triangles in the STL format (**S**tandard **T**essellation **L**anguage).

Search path

A search is made in the directories predefined on the CF card for the file entered in the system variables in the following sequence:

1. /oem/sinumerik/nck/prot_data/machine/3d_data/mm
2. /oem/sinumerik/nck/prot_data/machine/3d_data/inch

Interpretation of the length specifications

Depending on the storage directory, the length specifications contained in the STL file are interpreted in mm or inches:

- <Path>/**mm**: Interpretation of the length specifications in millimeters
- <Path>/**inch**: Interpretation of the length specifications in inches

---

Note

Maximum length of the file name

The maximum file name length, including point and the file extension is 49 characters. For more than 49 characters, an alarm is displayed when generating an archive.

NPP files

An NPP file (file extension .NPP) must contain a description of the geometry data of one or several protection area elements using NPP system variables (**N**C **P**rotection Area **P**rimitives). Using an NPP file, as protection area elements, only geometrically primitive cuboids, spheres and cylinders ($NP_TYPE: "BOX", "SPHERE" and "CYLINDER"), may be defined.

Search path

A search is made in the directories predefined on the CF card for the file entered in the system variables in the following sequence:

1. /oem/sinumerik/nck/prot_data/machine/3d_data/mm
2. /oem/sinumerik/nck/prot_data/machine/3d_data/inch

Interpretation of the length specifications

Depending on the storage directory, the length specifications contained in the NPP file are interpreted in mm or inches:

- <Path>/**mm**: Interpretation of the length specifications in millimeters
- <Path>/**inch**: Interpretation of the length specifications in inches

NPP file properties

- An NPP file must start with the following comments lines:
  `;COLLISION AVOIDANCE DATA`
  `;LOC_NP_ROOT_NAME="<Root_Name>"`
- As `<Root_Name>`, the name of the first protection area element contained in the NPP file, specified there under `$NP_NAME`, must be entered.
- Comment line are lines that start with the **;** character
- NPP files are allowed to contain empty lines

Properties of NPP system variables

NPP system variables contained in the NPP file have the following properties:

- Same name, significance and syntax as the corresponding system variables used in NC programs
- The system variables of the NC are **not** overwritten by the NPP system variables.
- **Within** an NPP file, the indices of NPP system variables must be **unique**.
- The indices of NPP system variables can be the **same** in **various** NPP files.
- The indices and the values assigned in the NPP system variables must be constants.

Supplementary conditions

- The values for $NP_COLOR (Page 289), $NP_D_LEVEL (Page 290), $NP_USAGE (Page 291) are, for the protection area elements defined in the NPP file, inherited from the protection area element in which they are integrated. As a consequence, all protection area elements of an NPP file have the same values for these properties.
- For positioning the protection area elements of an NPP file, the same conditions apply as for positioning the protection area elements with the system variables of the NC ($NP_TYPE: "BOX", "SPHERE" and "CYLINDER").
- No additional STL or NPP files are embedded in an NPP file.

Note

Maximum length of the file name

The maximum file name length, including point and the file extension is 49 characters. For more than 49 characters, an alarm is displayed when generating an archive.

Syntax

```
$NP_FILENAME[<n>] = "<name>"
```

Meaning

`$NP_FILENAME`:	Name of the STL or NPP file	
	Data type:	STRING
	Default value:	"" (empty string)

`<n>`:	System variable or protection area element index	
	Data type:	INT
	Value range:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<name>`:	Name of the STL or NPP file	
	Data type:	STRING

Examples

Using an STL file

The geometry data for the 19th protection area element are stored in file KUEHLDUESE_1.STL:

Program code	Comment
`N100 $NP_FILENAME[18] = "KUEHLDU-ESE_1.STL"`	`; 19th protection area element,` `; file name = "KUEHLDUESE_1.STL"`

Using an NPP file

The NPP file "Kopf_A.NPP" file is loaded into the system variables of the NC for the 19th protection area element. This contains the following three protection area elements `"Box-1"`, `"Sphere-1"` and `"Cylinder-1"`.

```
Program code                                         Comment
$NP_NAME[18]  = "Head"                               ; 19th protection area element
$NP_NEXT[18]  = ""
$NP_NEXTP[18] = ""
$NP_TYPE[18]  = "FILE"
$NP_OFF[18,0] =  80.0
$NP_OFF[18,1] = 100.0
$NP_OFF[18,2] = -50.0
$NP_DIR[18,0] =   0.0
$NP_DIR[18,1] =   0.0
$NP_DIR[18,2] =   0.0
$NP_ANG[18]   =   0.0
$NP_COLOR[18]   = 0                                  ; 1)
$NP_D_LEVEL[18] = 0                                  ; 1)
$NP_USAGE[18]   = "A"                                ; 1)
$NP_FILENAME[18] = "Kopf_A.npp"
; 1) Active for all protection area elements loaded from file Kopf_A.NPP
```

Content of file "Kopf_A.NPP"

```
Program code                                         Comment
;COLLISION AVOIDANCE DATA                            ; 1st header
;LOC_NP_ROOT_NAME = "Box-1"                          ; 2nd header
```

Program code	Comment
`$NP_NAME[0]   = "Box-1"`	; 1st protection area element
`$NP_NEXT[0]   = "Sphere-1"`	
`$NP_NEXTP[0]  = "Cylinder-1"`	
`$NP_TYPE[0]   = "BOX"`	
`$NP_PARA[0,0] =  340`	
`$NP_PARA[0,1] =  340`	
`$NP_PARA[0,2] =  340`	
`$NP_OFF[0,0]  =   42`	
`$NP_OFF[0,1]  =   73`	
`$NP_OFF[0,2]  = -100`	
`$NP_DIR[0,0]  =    0`	
`$NP_DIR[0,1]  =    0`	
`$NP_DIR[0,2]  =    1`	
`$NP_ANG[0]    =   30`	
`$NP_NAME[1]   = "Sphere-1"`	; 2nd protection area element
`$NP_NEXT[1]   = ""`	
`$NP_NEXTP[1]  = ""`	
`$NP_TYPE[1]   = "SPHERE"`	
`$NP_PARA[1,0] = 20`	
`$NP_PARA[1,1] = 0`	
`$NP_PARA[1,2] = 0`	
`$NP_OFF[1,0]  = 170`	
`$NP_OFF[1,1]  = 170`	
`$NP_OFF[1,2]  = 170`	
`$NP_DIR[1,0]  =   0`	
`$NP_DIR[1,1]  =   0`	
`$NP_DIR[1,2]  =   0`	
`$NP_ANG[1]    =   0`	
`$NP_NAME[2]   = "Cylinder-1"`	; 3rd protection area element
`$NP_NEXT[2]   = ""`	
`$NP_NEXTP[2]  = ""`	
`$NP_TYPE[2]   = "CYLINDER"`	
`$NP_PARA[2,0] =  20`	
`$NP_PARA[2,1] =  20`	
`$NP_PARA[2,2] =   0`	
`$NP_OFF[2,0]  = 170`	
`$NP_OFF[2,1]  = 170`	
`$NP_OFF[2,2]  = 170`	
`$NP_DIR[2,0]  =   1`	
`$NP_DIR[2,1]  =   2`	
`$NP_DIR[2,2]  =   3`	
`$NP_ANG[2]    =  73`	

6.2.4.10 $NP_PARA

Function

Enter the dimensions of the protection area body according to the type of the protection area element ($NP_TYPE (Page 292)) in the system variable.

Coordinate system

The local coordinate system in which the position of the protection area body is specified, is defined by the system variables $NP_OFF (Page 302), $NP_DIR (Page 303), $NP_ANG (Page 305).

Syntax

```
$NP_PARA[<n>,<i>] = <value>
```

Meaning

`$NP_PARA`:	Parameter values corresponding to the type of the protection area element					
	Data type:	REAL				
	Default value:	0.0				
`<n>`:	System variable or protection area element index					
	Data type:	INT				
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)				
`<i>`:	Parameter index					
	Data type:	INT				
	Range of values:	0, 1, 2				
	Parameter index	**Type of the protection area element**				
		BOX	**SPHERE**	**CYLINDER**	**CONE**	**TORUS**
	0	Length in X 1)	Radius 1)	Height in Z 1)	Height in Z 1)	Radius 1 1) 2)
	1	Width in Y 1)	---	Radius 1) 2)	Radius 1 1) 2)	Radius 2 1)
	2	Height in Z 1)	---	---	Radius 2 2)	---
`<value>`:	Parameter value					
	Data type:	REAL				
	Range of values:	0.0 < x ≤ max. REAL value				
1) The parameter value must not be 0.						
2) Radius in the X/Y plane						
---: Parameters that are not evaluated						

Example

The 19th protection area element is a cube with the dimensions:

- Length: 50.0 in the X axis
- Width: 100.0 in the Y axis
- Height: 75.5 in the Z axis

Program code	Comment
; 19th Protection area element,	
N100 $NP_TYPE[18] = "BOX"	; type = "BOX"
N120 $NP_PARA[18,0] = 50.0	; length in X = 50.0
N130 $NP_PARA[18,1] = 100.0	; width in Y = 100.0
N140 $NP_PARA[18,2] = 75.5	; height in Z = 75.5

6.2.4.11 $NP_OFF

Function

The offset vector by which the local coordinate system of the protection area element is moved compared to the coordinate system of the previous protection area element should be entered in the system variable.

Syntax

```
$NP_OFF[<n>,<i>] = <value>
```

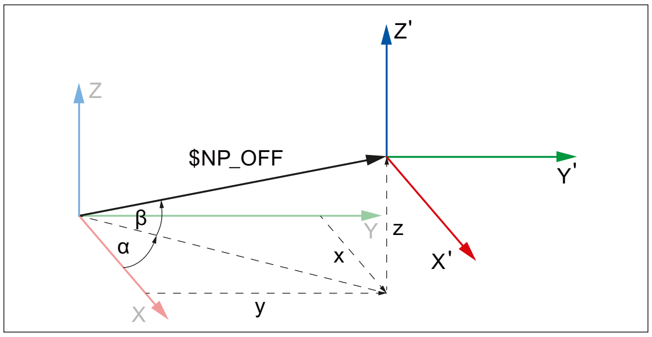
Meaning

`$NP_OFF`:	Offset vector	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value
	Default value:	(0.0, 0.0, 0.0)
`<m>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	0: X coordinate (abscissa) 1: Y coordinate (ordinate) 2: Z coordinate (applicate)
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value

Example

The local coordinate system of the 19th The local coordinate system of the protection area element is moved by the following vector compared to the coordinate system of the previous protection area element:

- X direction: 25.0
- Y direction: 50.0
- Z direction: 37.25



X, Y, Z	Coordinate system of the previous protection area element
X', Y', Z'	Coordinate system of the current protection area element

Program code	Comment
; 19. protection area element, offset vector	
N100 $NP_OFF[18,0] = 25.0	X = 25.0
N110 $NP_OFF[18,1] = 50.0	Y = 50.0
N120 $NP_OFF[18,2] = 37.25	Z = 37.25

6.2.4.12 $NP_DIR

Function

The direction vector through which the local coordinate system of the protection area element is rotated compared to the coordinate system of the previous protection area element should be entered in the system variable. The angle of rotation should be entered in $NP_ANG (Page 305).

Boundary conditions

- The absolute value of the direction vector must be greater than: $1*10^{-6}$
- A work offset parameterized in $NP_OFF (Page 302) is performed **before** the rotation.

Syntax

```
$NP_DIR[<n>,<i>] = <value>
```

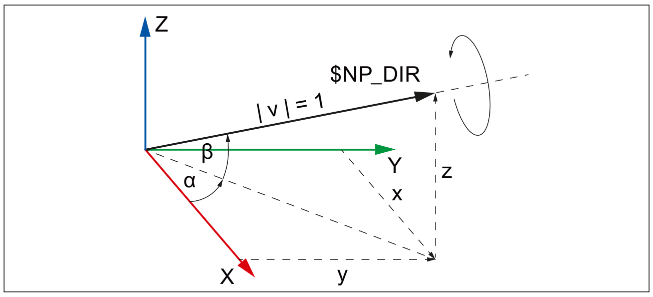
Meaning

`$NP_DIR`:	Direction vector	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value
	Default value:	(0.0, 0.0, 0.0)
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<i>`:	Coordinate index	
	Data type:	INT
	Range of values:	0 → X; 1 → Y: 2 → Z
`<value>`:	Coordinate value	
	Data type:	REAL
	Range of values:	- max. REAL value ≤ x ≤ ± max. REAL value

Example

The local coordinate system of the 19th protection area element is rotated around the direction vector compared to the coordinate system of the previous protection area element. The direction vector is the unit vector (1; 0; 0), rotated through α=90° in the X/Y plane and β=10° in the Y/Z plane, in relation to the world coordinate system. The following values result from this for the individual components of the direction vector:

- X component = cos(α) * cos(β) = cos(90) * cos(10) = 0.0
- Y component = sin(α) * cos(β) = sin(90) * cos(10) ≈ 0.985
- Z component = sin(β) = sin(10) ≈ 0.174



```
Program code                                              Comment
; 19. protection area element, direction vector
N100 $NP_DIR[18.0] = COS(90) * COS(10)                    ; 0 = X component
N110 $NP_DIR[18.1] = SIN(90) * COS(10)                    ; 1 = Y component
N120 $NP_DIR[18.2] = SIN(10)                              ; 2 = Z component
```

6.2.4.13 $NP_ANG

Function

The angle through which the local coordinate system of the protection area element is rotated around the direction vector ($NP_DIR (Page 303)) compared to the coordinate system of the previous protection area element should be entered in the system variable.

Syntax

```
$NP_ANG[<n>] = <value>
```

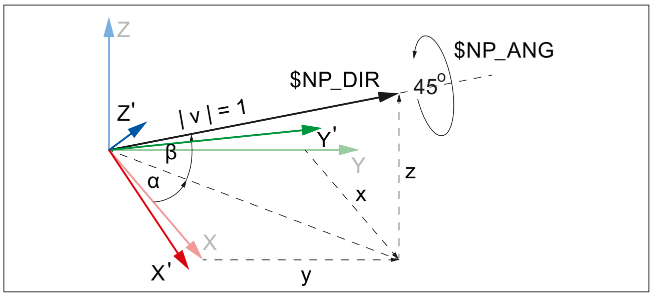
Meaning

`$NP_ANG`:	Angle of rotation	
	Data type:	REAL
	Range of values:	-360° < x ≤ 360°
	Default value:	0.0
`<n>`:	System variable or protection area element index	
	Data type:	INT
	Range of values:	0, 1, 2, ... ($MN_MM_MAXNUM_3D_PROT_AREA_ELEM - 1)
`<value>`:	Angle	
	Data type:	REAL

Example

The local coordinate system of the 19th protection area element is rotated by the angle by δ=45.0° around the direction vector compared to the coordinate system of the previous protection area element. The direction vector is the unit vector (1; 0; 0), rotated through α=90° in the X/Y plane and β=10° in the Y/Z plane, in relation to the world coordinate system. The following values result from this for the individual components of the direction vector:

- X component = cos(α) * cos(β) = cos(90) * cos(10) = 0.0
- Y component = sin(α) * cos(β) = sin(90) * cos(10) ≈ 0.985
- Z component = sin(β) = sin(10) ≈ 0.174
- Angle δ = 45.0°



Program code	Comment
`; 19. Protection area element, direction vector and angle of rotation`	
`N100 $NP_DIR[18.0] = COS(90)*COS(10)`	`; 0 = X component`
`N110 $NP_DIR[18.1] = SIN(90)*COS(10)`	`; 1 = Y component`
`N120 $NP_DIR[18.2] = SIN(10)`	`; 2 = Z component`
`N130 $NP_ANG[18] = 45.0`	`; Angle of rotation` δ = 45°

6.2.5 System variables: Protection area elements for automatic tool protection areas

The protection area element of an automatic tool protection area is defined using the subsequent system variables. The control automatically generates the values of the system variables from the geometric data of the associated tool, and these can only be read.

Name [1)]	Meaning	Analog to [2)]
$NP_T_NAME[<n>]	Name of the protection area element	$NP_NAME (Page 286)
$NP_T_TYPE[<n>]	Type of the protection area element	$NP_TYPE
$NP_T_FILENAME[<n>]	File name of the STL file that contains the geometric data of the protection area element (only relevant for $NP_T_TYPE == "FILE")	$NP_FILENAME (Page 296)
$NP_T_PARA[<n>,<i>]	Geometric parameters of the protection area body (only relevant for $NP_T_TYPE == "BOX" or "SPHERE" or "CYLINDER")	$NP_PARA
$NP_T_OFF[<n>,<i>]	Offset vector of the protection area element local coordinate system	$NP_OFF (Page 302)
$NP_T_DIR[<n>,<i>]	Direction vector for the rotation of the protection area element local coordinate system	$NP_DIR (Page 303)
$NP_T_ANG[<n>]	Angle for the rotation of the protection area element local coordinate system	$NP_ANG (Page 305)
1) n = 0, 1, ... ($MN_MM_MAXNUM_3D_T_PROT_ELEM – 1)		
2) The system variables of the automatic tool protection areas correspond to those of the machine protection areas.		

6.2.6 Boundary conditions

Protection area bodies for spindles

For spindles that are not in position-controlled operation, the associated protection area bodies are only modeled statically. As a result the following boundary conditions must be met when modeling protection area bodies connected with a spindle as a kinematic element:

- The protection area body **must be symmetrical around its axis of rotation**.
- The symmetrical axis of the protection area body must lie on the axis of rotation of the spindle (**collinear**)

This should be taken into account for all types of protection area bodies:

- The protection area body is a basic geometrical body (sphere, cylinder).
- The protection area body comprises several, geometrical basic bodies.
- The protection area body comprises triangles (STL file).
- The protection area body for an automatic tool protection area is created from the geometric tool data.

Note

Automatic tool protection areas

Only using tools that are symmetrical around the axis of rotation is recommended for automatic tool protection areas in connection with spindles.

Following protection areas

The rotational symmetry and collinearity of the protection area body in respect of the axis of rotation of the spindle must also be complied with for all protection areas that are connected with elements of the kinematic chain ($NP_NEXT, $NP_NEXTP) following on from the spindle.

Schematic example of such a kinematic chain: ... → (rotary axis/spindle) → (offset) → (linear axis) → (offset) → ...

Rotary axes

The boundary conditions stated above must also be observed in connection with rotary axes where these are also operated as spindles.

Tool reference point and kinematic transformation

The position of the associated tool reference point is determined in principle through the assignment of an automatic tool protection area to an element in the kinematic chain. However, the position of the tool reference point can be changed through offsets within the automatic tool protection area via the system variables $NP_T_OFF, $NP_T_DIR and $NP_T_ANG (Page 306). A change in position for the tool reference point of this type is not captured by the kinematic transformations. In the context of kinematic transformation it is therefore highly recommended that no offsets are used for the geometric machine modeling which move the tool reference point.

6.3 Data lists

6.3.1 Machine data

6.3.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
MD18890	$MN_MM_MAXNUM_3D_PROT_AREAS	Maximum number of protection areas
MD18892	$MN_MM_MAXNUM_3D_PROT_AREA_ELEM	Maximum number of protection area elements
MD18893	$MN_MM_MAXNUM_3D_T_PROT_ELEM	Maximum number of tool protection area elements
MD18897	$MN_MM_MAXNUM_3D_INTERFACE_IN	Maximum number of NC/PLC interface signals for the preactivation of protection areas
MD18895	$MN_MM_MAXNUM_3D_FACETS	Maximum number of triangles for protection areas
MD18894	$MN_MM_MAXNUM_3D_FACETS_INTERN	Maximum number of triangles for automatic tool protection areas
MD18899	$MN_PROT_AREA_TOOL_MASK	Creation mode for automatic tool protection areas

6.3.2 System variables

Identifier	Description
$NP_PROT_NAME	Name of the protection area
$NP_CHAIN_ELEM	Name of the kinematic element to which the protection area will be connected
$NP_PROT_TYPE	Type of the protection area
$NP_1ST_PROT	Name of the first protection area element of the protection area
$NP_PROT_COLOR	Transparency and color value of the protection area
$NP_PROT_D_LEVEL	Detail level for the protection area
$NP_BIT_NO	Bit number of the interface signal to activate/deactivate the protection area
$NP_INIT_STAT	Initialization status of the protection area
$NP_INDEX	Array for addressing the effective geometry data
$NP_NAME	Name of the protection area element
$NP_NEXT	Name of the following protection area element
$NP_NEXTP	Name of the branching protection area element
$NP_COLOR	Transparency and color value of the protection area element
$NP_D_LEVEL	Detail level for the protection area element
$NP_USAGE	Usage of the protection area element
$NP_TYPE	Type of the protection area element
$NP_FILENAME	Name of the STL file with the geometry data of the protection area element body
$NP_PARA	Parameter values corresponding to the type of the protection area element

Identifier	Description
$NP_OFF	Offset vector
$NP_DIR	Direction vector
$NP_ANG	Angle of rotation

K9: collision avoidance, internal 7

7.1 Function description

7.1.1 Options

The function "Collision avoidance" is an option that requires a license. The following versions are available:

- **Collision avoidance ECO** (machine): 6FC5800-0AS03-0YB0
  Properties:
  - Protection: Machine - Machine
  - HMI visualization
  - Only for single-channel control configurations.
  - Protection area elements: geometric primitives
- **Collision avoidance** (machine, working area): 6FC5800-0AS02-0YB0
  Properties:
  - as for Collision avoidance ECO
  - Protection area elements: Files in STL and NPP format

7.1.2 Characteristics

The "Collision avoidance" function is used to prevent collisions of machine parts and tool cutting edges while the machine axes are being traversed. To do this, the function cyclically calculates the clearance to the protection areas enveloping the bodies to be protected. If two protection areas approach one another to within a configurable safety clearance, an alarm is displayed and the NC program stopped before the relevant traversing block (AUTOMATIC, MDI mode) or the traversing motion is stopped (JOG mode).

Sequence

The collision avoidance is set up in the following sequence:

1. Release of the "Collision avoidance" function by setting the corresponding option (Page 311).
2. Setting the machine data for the basic parameterization of the functions:
   - Kinematic chain, section "Machine data (Page 235)"
   - Geometric machine modeling, section "Machine data (Page 272)"
   - Collision avoidance, section "Machine data (Page 323)"

3. Writing of the machine kinematic structure through kinematic elements.
   See Chapter "K7: Kinematic chain (Page 229)".
4. Writing of the protection areas and protection area elements as enveloping geometry of the machine parts, tools and workpieces to be protected. Assignment of the protection area to elements of the kinematic chain.
   See Chapter "K8: Geometric machine modeling (Page 263)".
5. Definition of collision pairs, i.e. of two protection areas that are to be monitored for collision.
   See Chapter "$NP_COLL_PAIR (Page 326)".
6. Triggering a recalculation of the kinematic and geometric model.
   See Chapter "Request recalculation of the machine model of the collision avoidance (PROTA) (Page 332)".
7. Activation of the protection areas to be monitored.
   See Chapter "Setting the protection zone status (PROTS) (Page 333)".
8. Optional: Use of the extended functions and system variables

Limits of the collision avoidance

The function cannot guarantee complete protection against a collision when traversing machine parts, tools of workpieces. On the one hand, the collision protection can only be as good as the parameterized kinematic and geometric model or the machine and the protection areas. On the other hand, bodies that have not been modeled cannot be monitored at all. For this reason, it remains the responsibility of the machine operator to ensure that a traversing motion is executed collision-free even when the collision protection is activated.

States of protection areas

Whether a protection area is taken into account for collisions, depends on the state of the protection area:

State	Meaning
Active	The protection area is taken into account for collisions.
Inactive	The protection area is **not** taken into account for collisions.
Preactivated	The protection area is taken into account for collisions. A collision alarm is only issued when it has also been activated by a protection area-specific NC/PLC interface signal.

State after control ramp-up

After the control has powered up, all protection areas are in the state corresponding to their respective setting in $NP_INIT_STAT. See Section "$NP_INIT_STAT (Page 282)".

State change

The state of a protection areas can be changed by:

- The PROTS() (Page 333) procedure
- Change of the initialization state to $NP_INIT_STAT followed by recalculation of the machine model through the PROTA() (Page 332)procedure.

Requirements

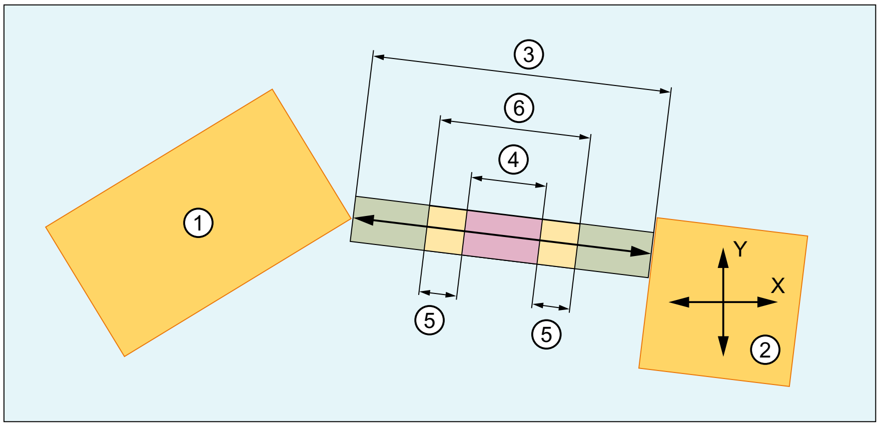
The following requirements must be satisfied so that the protection areas of a collision pair can be monitored:

- Axes or spindles: referenced/synchronized
  The position measurement system for the axes or spindles which move a protection area must be referenced or synchronized. If this has not been done the corresponding protection area is in an "inactive" state.
- External motion
  With traversing motions that are not performed by the NC, e.g. PLC axis or manually moved axis, the current axis position must be known in the NC.

7.1.3 Reaction of the control to a risk of collision

The collision avoidance takes the following parameterizable limit values into account for the collision detection:

- Collision tolerance
- Safety clearance



① Protection area 1 (static)
② Protection area 2 (can be moved in X and Y direction)
③ Current clearance
④ Safety clearance
⑤ Collision tolerance / 2
⑥ Collision clearance = safety clearance + collision tolerance

Figure 7-1 Current clearance, collision tolerance and safety clearance

Collision tolerance and safety clearance

Safety clearance

The safety clearance defines a clearance up to which two active protection areas, monitored for collision, can approach one another. The collision avoidance ensures that this clearance is not violated and that the collision is displayed.

The safety clearance can be set specifically for each collision pair via a system variable (Page 327).

For all collision pairs for which no specific safety clearance is set via a system variable, the general value that can be set via MD10622 $MN_COLLISION_SAFETY_DIST (Page 323) applies.

Collision tolerance

The collision tolerance defines an NC-wide additional clearance valid to the safety clearance. Two active protection areas monitored for collision may therefore approach each other up to the collision clearance (safety clearance + collision tolerance). Ideally, the collision avoidance stops the traversing motion of the protection areas exactly at the collision clearance and displays the collision. However, it is permissible that the collision tolerance is not complied with exactly or that a brief violation does not result in collision detection and the traversing motion is not stopped.

The collision tolerance is set the same for all collision pairs via MD10619 $MN_COLLISION_TOLERANCE (Page 323).

Note

Difference between collision tolerance and safety clearance

The collision tolerance can be violated and is permissible. The safety clearance is always maintained.

Responses in the operating mode: AUTOMATIC

Collision detection during preprocessing

In automatic mode, the traversing blocks of the active program are already checked during the preprocessing. If a collision is already detected there, this results in the following reactions:

- Stop of the traversing motions in the channel
- NC/PLC interface signal: DB21,... .DBX377.0 = 1 (collision avoidance: stop)
- Display of alarm 26260 with the block number of the relevant traversing block
- Cancelation of the program processing

Critical approach

Even in automatic mode, superimposed or asynchronous motions can occur that cannot be considered in advance. For this reason, the traversing velocity is reduced or the traversing motion is totally stopped at a critical approach of protection areas:

- Axes-specific NC/PLC interface signal when the traversing velocity is reduced:
  DB31, ...DBX77.0 == 1 (collision avoidance: velocity reduction)
- Channel-specific NC/PLC interface signal at a stop of the traversing motion:
  DB21, ...DBX377.0 == 1 (collision avoidance: stop)

Preactivated protection areas

If during the block processing in the **preprocessing**, it is determined in a traversing block that two protection areas, of which at least one is only **preactivated**, would collide if they were active, this does not yet produce the same reactions as described above in the "Collision detection during preprocessing". The reactions only occur when both protection areas are active.

If the block at the activation time is already being traversed in the main run, a collision is detected based on the collision calculation during preprocessing and the above reactions triggered. The collision is detected irrespective of whether the protection areas are actually colliding at the activation time.

Responses in the operating mode: JOG

If two protection areas approach one another when traversing in the JOG mode, the traversing velocity is continuously braked down to standstill when the collision clearance is reached. Alarm 26280 is output when the collision clearance is reached.

If the collision location is exited in the opposite direction, a continuously higher traversing velocity is possible depending on the clearance of the protection areas.

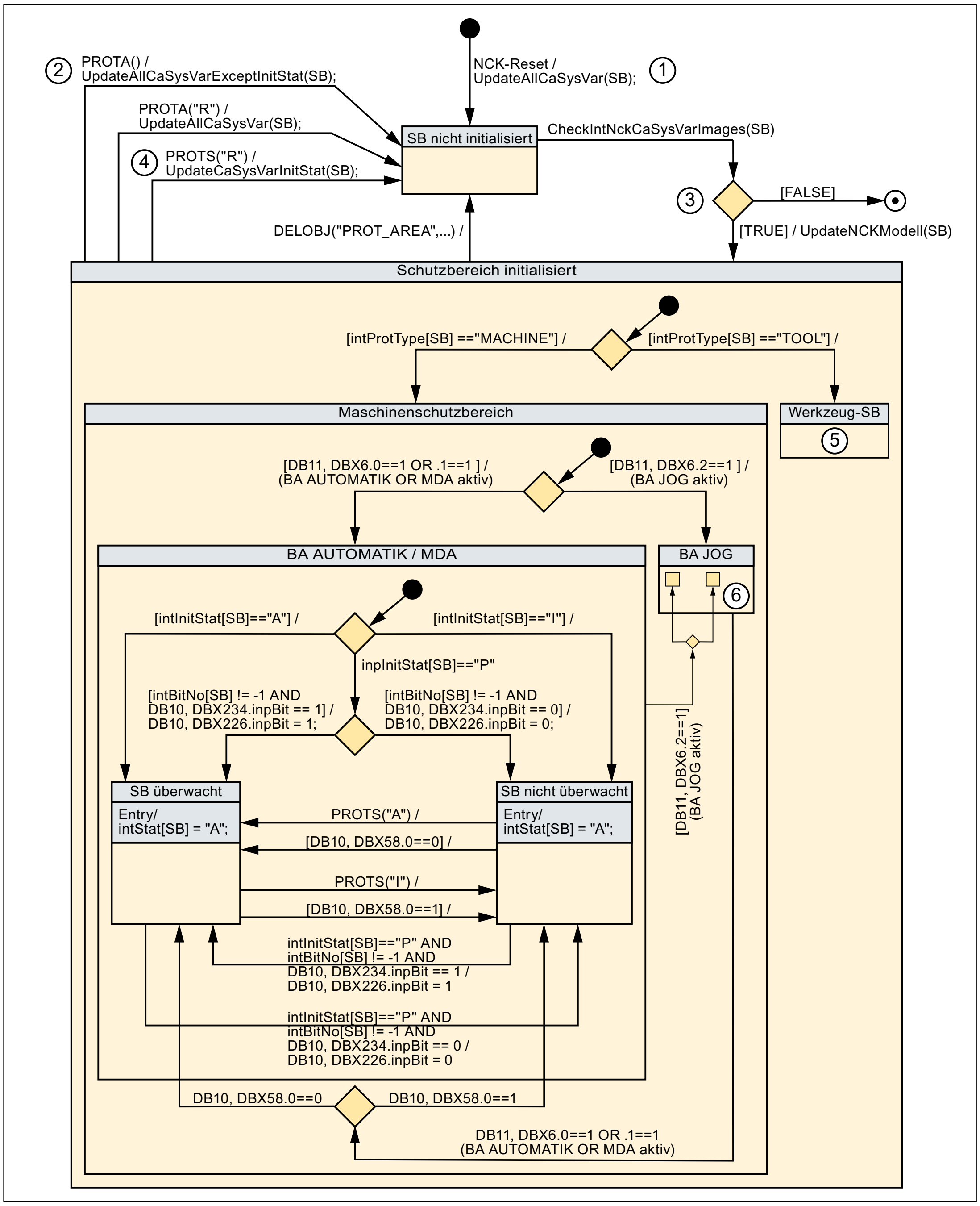
Traversing motion is always **canceled** when the collision clearance is reached. Continuation of the traversing motion always requires a new travel request (e.g by pressing a traversing key), irrespective of the traversing direction.

Responses in the operating mode: MDI

If two protection areas approach one another when traversing in the MDI mode, the traversing velocity is continuously braked down to standstill when the collision clearance is reached. Alarm 26280 is output when the collision clearance is reached.

Contrary to the AUTOMATIC, mode, for MDI, the traversing blocks are **not** checked in advance (run-in) for collision.

7.1.4 State diagram: Protection area


② PROTA() / UpdateAllCaSysVarExceptInitStat(SB);
NCK-Reset / UpdateAllCaSysVar(SB); ①
PROTA("R") / UpdateAllCaSysVar(SB);
④ PROTS("R") / UpdateCaSysVarInitStat(SB);
SB nicht initialisiert
CheckIntNckCaSysVarImages(SB)
③ [FALSE]
[TRUE] / UpdateNCKModell(SB)
DELOBJ("PROT_AREA",...) /
Schutzbereich initialisiert
[intProtType[SB] =="MACHINE"] /
[intProtType[SB] =="TOOL"] /
Maschinenschutzbereich
Werkzeug-SB ⑤
[DB11, DBX6.0==1 OR .1==1] / (BA AUTOMATIK OR MDA aktiv)
[DB11, DBX6.2==1] / (BA JOG aktiv)
BA AUTOMATIK / MDA
BA JOG ⑥
[intInitStat[SB]=="A"] /
[intInitStat[SB]=="I"] /
inpInitStat[SB]=="P"
[intBitNo[SB] != -1 AND DB10, DBX234.inpBit == 1] / DB10, DBX226.inpBit = 1;
[intBitNo[SB] != -1 AND DB10, DBX234.inpBit == 0] / DB10, DBX226.inpBit = 0;
[DB11, DBX6.2==1] (BA JOG aktiv)
SB überwacht
Entry/ intStat[SB] = "A";
SB nicht überwacht
Entry/ intStat[SB] = "A";
PROTS("A") /
[DB10, DBX58.0==0] /
PROTS("I") /
[DB10, DBX58.0==1] /
intInitStat[SB]=="P" AND intBitNo[SB] != -1 AND DB10, DBX234.inpBit == 1 / DB10, DBX226.inpBit = 1
intInitStat[SB]=="P" AND intBitNo[SB] != -1 AND DB10, DBX234.inpBit == 0 / DB10, DBX226.inpBit = 0
DB10, DBX58.0==0
DB10, DBX58.0==1
DB11, DBX6.0==1 OR .1==1 (BA AUTOMATIK OR MDA aktiv)

SB	Protection area
BA	Operating mode
①	UpdateAllCaSysVar(SB) function
	All system variables of the collision avoidance are read into NC-internal variables:
	int... = $N...

② UpdateAllCaSysVarExeptInitStat(SB) function
As with the UpdateAllCaSysVar(SB) function, but the $NP_INIT_STAT system variable is not read in. Internally in the NC, the last value of the intInitStat initialization status is therefore retained.

③ CheckIntNckCaSysVarImages(SB) function
The NC-internal variables read in from the system variables are checked for consistency.
Return value when an error is detected: FALSE; if error-free: TRUE.

④ UpdateCaSysVarInitStat(SB) function
Only the system variable $NP_INIT_STAT is read into the NC-internal variable intInitStat.

⑤ The internal structure of the "Tool protection area" state is the same as the "Machine protection area" state.

⑥ The internal structure of the "JOG mode" state is the same as the "AUTO/MDA mode" state.

7.1.5 Tools

Modeling

Protection areas for tools can be modeled automatically from the collision avoidance and be updated automatically following a tool change with some limitations. The following conditions must be fulfilled for this purpose:

- The protection area for the tool is modeled as an automatic tool protection area (type: TOOL). See system variable $NP_PROT_TYPE (Page 276)
- The tool is managed by the control tool management.
- The tool data stored in the tool management matches the actual geometric dimensions of the tool.
- The collision avoidance recognizes the completion of the tool change. Normally, by programming the corresponding tool offset number `Dx`, with x = 0, 1, 2, 3, ...
- The collision avoidance recognizes the automatic tool protection area for which the tool was changed.
- If the tool path refers to a normalized tool, the tool radius of the actual tool is specified as either positive or negative deviations referred to the normalized tool. In this case, collision avoidance calculates with the following values:
  - Positive value: Tool radius = specified value, however, as a minimum, the value of the parameterized collision tolerance
  - Negative value: Tool radius = value of the parameterized collision tolerance

Change to the machine model

If a tool which is located in a magazine or tool adapter modeled in the active machine model of the collision avoidance is changed in the machine then the machine model must be updated. This is the case where one of the following actions is carried out, for example:

- A tool is loaded / unloaded in a tool magazine modeled in the machine model.
  Example: Tool change in a revolver (circular) magazine.
  Update: The machine model update must be explicitly requested by the user using `PROTA` once the tool change has completed.
- The tool that is in a tool holder is changed.
  Example: Tool change in the tool holder of the main spindle.
  Update: The machine model is automatically updated following the tool change (standard: `M6`) with output of the programmed tool offset number `Dx` in the main run.

Tool changes

The collision avoidance only updates the active machine model after a tool change without an explicit request by `PROTA` if a tool offset is selected (output of the programmed tool offset number Dx at the NC/PLC interface).

However, at the control, tool changes can be made, which are not associated with a tool offset selection. However, for these changes the active machine model is not updated. These types of tool changes include, for example:

- Loading/unloading a tool at the user interface.
  SINUMERIK Operate: Operating area "Parameter" > "Tool list" > Vertical softkey: "Loading" or "Unloading"
- Carrying out a tool change via the PLC user program
- Directly writing to the tool buffer memory using the system variable `$TC_MPP6[9998, <location>]`
- `SETMS(<spindle number>)`: Changing the master spindle in the channel
- `TMMVTL`: PI service "Prepare magazine location for loading, unload tool"
  Reference Function Manual Basic Functions; Chapter: "P3: Basic PLC program for SINUMERIK 840D sl" > "Component descriptions" > "PI services" > "PI service: TMMVTL"
- `MVTOOL`: Command to move a tool
  Reference Function Manual Tool Management; Chapter "Programming > "NC language commands" > "MVTOOL - language command to move a tool"

If such a change is carried out, then the machine manufacturer must be requested to update the machine module via the PLC user program. Examples of options include:

- A new channel reset is requested if the channel is in the "reset" state. When the reset response is appropriately set (MD20110 $MC_RESET_MODE_MASK), then the actual tool offset number `Dx`is output again.
- Starting an ASUB or manufacturer's cycle, which includes the output of the tool offset number `Dx` and the request to update the machine model (`PROTA`).

Tool wear

Minimal tool changes do not have to be taken into account in the machine model, as generally they are far less than the collision clearance.

If a tool change occurs, which is relevant for collision avoidance, e.g. diameter changes for grinding tools, then these must be taken into account by explicitly requesting that the machine model is updated (`PROTA`).

No change to the machine model

The active machine model does **not** change if a fully modeled machine part with tools, e.g. a tool magazine, is moved in the machine.

Example: Revolver magazine on a lathe

The revolver magazine on a lathe is fully modeled in the machine model of the collision avoidance:

- The geometry of the magazine and the tools located inside it
- The motions of the magazine through the machine axes

A rotation of the revolver magazine then does not represent a change to the machine model:

- since no tools are changed inside the machine model the geometries of all the protection areas remain unchanged.
- As the motions of the protection areas by the machine axes are fully captured by the collision avoidance via the kinematic chain.

Supplementary conditions

Several spindles in the channel

For configurations with several spindles in the channel, the collision avoidance function assumes that a tool change takes place in the master spindle of the channel (S1). As a result only the automatic tool protection area of the master spindle is updated through the collision avoidance once the tool change has taken place.

Unsupported tool configurations

Tool configurations in accordance with ISO modes 4 and 5 (H numbers) along with "flat D numbers" are not supported by the collision avoidance.

7.1.6 Boundary conditions

Channel assignment

All of the machine components relevant for collision avoidance must be located in the **first channel** of the NC:

- All **axes** and **spindles** of the kinematic chain.
  See Chapter "K7: Kinematic chain (Page 229)"
- All **tools** of the automatic tool protection areas of the geometric machine modeling.
  See Chapter "K8: Geometric machine modeling (Page 263)"

Allowance for the following error

The collision avoidance uses the position setpoints of the relevant machine axes for the clearance calculation of the protection areas. However, the actual positions of the machine axes differ from the position setpoint by the following error. For this reason, there is also a difference between the position setpoint and the actual position for the protection areas. This difference must be taken into account by the user through configuration of a suitably large safety clearance or enlargement of the protection area.

Compensations

The various compensation functions of the NC – for instance, temperature, spindle and pitch error and sag compensation – ensure that positions programmed in the workpiece coordinate system are actually assumed in the machine coordinate system. The collision avoidance takes into account the position corrections made by the compensation functions.

⚠ WARNING

Risk of collision

If compensations are used for other purposes, e.g. in order to implement axis couplings in the machine coordinate system, the collision avoidance working with position setpoints can no longer be performed reliably. There is a risk of collision.

Actual value offset in the machine coordinate system with PRESETON

With active collision avoidance and use of an actual value offset in the machine coordinate system with PRESETON, it is the sole responsibility of the user to ensure that the geometric model of the collision avoidance remains consistent.

⚠ WARNING

Risk of collision

If an actual value offset is made in the machine coordinate system with PRESETON and the geometric model of the collision avoidance not adapted accordingly, the collision avoidance working with position setpoints can no longer be performed reliably. There is a risk of collision.

Block search

For the following types of block search there are **no** collision calculations carried out:

- Type 1: Block search without calculation
- Type 2: Block search with calculation at the contour
- Type 4: Block search with calculation at block end point

With the following type of block search collision calculations are carried out (in the preprocessing) for:

- Type 5: Block search with calculation in "Program test" (SERUPRO) mode

AUTOMATIC modes: Incomplete protection area data for collision

When a large number of protection areas have been configured, in exceptional cases, the following behavior can occur:

- Several protection areas have approached one another, down to the collision tolerance
- Only two protection areas are specified in Alarm 26260 "Collision **two** protection areas" that is displayed.
- The collision of the other protection areas is only displayed when manually traversing the axes after changing into the JOG mode.

7.2 Commissioning

7.2.1 General

7.2.1.1 Overview

The commissioning of the "Collision avoidance" function is performed using:

- Machine data
  - Specification of the quantity structure
  - Specification of general properties of the collision pairs
- System variables
  - Parameterization of the collision pairs and their properties

7.2.1.2 Structure of the system variables

The system variables are structured according to the following scheme:

$NP_<name>**[**<index_1>,<index_2>**]**

Note

Index_2 is not available for all system variables.

General

The system variables to describe the protection areas have the following properties:

- Prefix: **$NP_**, (N for NC, P for protection).
- They can be read and written via NC programs.
- They 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 protection areas are addressed via index_1. Index 0 → 1st protection area, index 1 → 2nd protection area, ... n → (n+1) protection area, where n = ($MN_MM_MAXNUM_3D_PROT_AREAS - 1)

All system variables of a protection area have the same index.

Index_2

For system variables that define a collision pair, the protection areas of the collision pair are addressed via index_2.

- 0 → 1. Protection area
- 1 → 2. Protection area

See also

Deletion of components (DELOBJ) (Page 253)

7.2.2 Machine data

7.2.2.1 Collision tolerance

The collision tolerance (accuracy of the collision check) for all protection areas of the NC monitored for collision is set with the machine data. If the clearance of two safety areas is less than the collision clearance, i.e. the sum of safety clearance (Page 323) and collision tolerance, then there is a collision.

MD10619 $MN_COLLISION_TOLERANCE = <collision tolerance>

Accuracy of automatic generated protection areas

The collision tolerance also determines the accuracy of the protection area bodies of protection areas produced automatically, e.g. automatic tool protection areas. The accuracy of the protection area bodies approached using triangle areas is 1/3 of the collision tolerance.

Effects

The smaller the collision tolerance is set, the greater the number of triangular areas required for the modelling of the automatically generated protection areas and the computation time required for the collision detection.

Recommended setting

Collision tolerance ≈ 1 mm

See also

Reaction of the control to a risk of collision (Page 313)

7.2.2.2 Safety clearance

The safety clearance for all protection areas of the NC monitored for collision is set with the machine data. The collision avoidance ensures that the safety clearance is not violated.

MD10622 $MN_COLLISION_SAFETY_DIST = <safety clearance>

Note

Collision pair-specific safety clearance

If a specific safety clearance has been set for a collision pair via the system variable $NP_SAFETY_DIST (Page 327), this has priority over the NC-specific safety clearance set in the machine data.

See also

Reaction of the control to a risk of collision (Page 313)

7.2.2.3 Maximum memory space

The maximum value of the memory space in KB that can be allocated to the collision avoidance is set with the machine data.

MD18896 $MN_MM_MAXNUM_3D_COLLISION = <value>

Value	Meaning
0	The maximum value of the memory space is determined automatically by the control using the following machine data: • MD18890 $MN_MM_MAXNUM_3D_PROT_AREAS • MD18892 $MN_MM_MAXNUM_3D_PROT_AREA_ELEM • MD18894 $MN_MM_MAXNUM_3D_FACETS_INTERN • MD18895 $MN_MM_MAXNUM_3D_FACETS
> 0	Maximum value = parameterized value [KB]

Note

Only one > 0 value must be entered in the machine data when one of the following alarms is displayed:

- Alarm 26262 "Insufficient memory space for the collision check of two protection areas"
- Alarm 26263 "Insufficient memory space for determining the clearance of two protection areas"

Used memory space

Various system variables are available to determine the memory space used by the collision avoidance. See section "Memory requirement (Page 329)".

7.2.2.4 Maximum number of collision pairs

The maximum number of possible collision pairs has an impact on:

- Length m of the system variable fields (e.g. $NP_COLL_PAIR[m, ...])
- The user memory required for collision avoidance
- The size of the commissioning archive

The maximum number of collision pairs can be restricted with machine data:

MD18898 $MN_MM_MAXNUM_3D_COLL_PAIRS = <value>

<Val-ue>	Meaning
0	The following applies to the maximum number of possible collision pairs MCP: MCP = maximum value of the machine data
x > 0	The following applies to the maximum number of possible collision pairs MCP: MCP = x, with 0 < x ≤ maximum value of the machine data
A value greater than the maximum permissible value of the machine data is limited internally to the maximum value. No feedback regarding this is sent to the user.	

7.2.2.5 Protection levels for collision avoidance On/Off

The protection level for switching the collision avoidance On/Off is set with the machine data via the user interface. The protection level can be specified according to operating mode and protection area type.

Machine data = <protection level>

Number	Identifier: $MN_	Meaning: Protection level for switching the collision avoidance On/Off
MD51160	ACCESS_WRITE_CA_MACH_JOG	Machine protection areas, JOG/MDI operating mode
MD51161	ACCESS_WRITE_CA_MACH_AUTO	Machine protection areas, AUTOMATIC mode,
MD51162	ACCESS_WRITE_CA_TOOL	Tool protection areas

References

A detailed description of protection levels can be found in:

Function Manual, Basic Functions, Section: "A2: Various NC/PLC interface signals and functions" > "Functions" > "Access protection via password and keyswitch"

7.2.3 System variables

7.2.3.1 Overview

A collision pair is parameterized with the following system variables:

Name	Meaning
$NP_COLL_PAIR	Name of a collision pair protection area
$NP_SAFETY_DIST	Safety clearance of the protection area pair

The system variables are described in detail in the following sections.

Note

Establish a defined initial state

It is recommended that a defined initial state be generated before parameterizing the collision avoidance. To do this, set the system variables of the collision avoidance to their default values with the DELOBJ() (Page 253) function.

Change system variable values

If the value of one of the system variables listed above is changed, the change becomes immediately visible at the user interface, e.g. SINUMERIK Operate. The machine model of the NC is only updated after explicitly requesting that the machine model is recalculated by calling the PROTA() (Page 332) or PROTS() (Page 333) function.

7.2.3.2 $NP_COLL_PAIR

Function

The names of the two protection areas, which form a collision pair, is entered in a system variable. The two protection areas can be in any sequence.

Collision pairs

As the collision check requires a lot of computation time, it does not make sense to monitor all parameterized protection areas for collision with one another through the collision avoidance. Examples in which a collision check does not make sense:

- Protection areas that cannot collide because of the construction
- Protection areas that have been defined without anchoring to the kinematic chain

The user must define which parameterized protection areas can actually collide on the machine and define them as collision pairs. Only these protection areas are monitored by the collision avoidance.

To define a collision pair, the names of the two protection areas must be entered in two system variables with the same collision pair index. One protection area under the protection area index 0, the other under the protection area index 1.

Part of a collision pair

Using the COLLPAIR() (Page 331) function, a check can be made as to whether two protection areas are parameterized as collision pair.

Syntax

```
$NP_COLL_PAIR[<m>,<i>] = "<name>"
```

Meaning

`$NP_COLL_PAIR`:	Name of the first or second protection area of a collision pair	
	Data type:	STRING
	Default value:	"" (empty string)
`<m>`:	System variable or collision pair index	
	Data type:	INT
	Range of values:	0, 1, 2, ... (M -1) 1)
`<i>`:	Protection area index	
	Data type:	INT
	Range of values:	0 (first protection area), 1 (second protection area)
`<name>`:	Protection area name	
	Data type:	STRING
1) M = n * (n - 1) / 2 where n = $MN_MM_MAXNUM_3D_PROT_AREAS		

Example

Two protection areas defined with the names "Rotary table" and "Tool in spindle" are to be checked for collision. The two protection areas are to be mutually monitored for collision. The definition for this is in the seventh collision pair:

Program code	Comment

`N100 $NP_COLL_PAIR[6,0] = "Rotary table"`	`; 7th collision pair,`
	`; 1st Protection area`

`N110 $NP_COLL_PAIR[6,1] = "Tool in spindle"`	`; 7th collision pair,`
	`; 2nd Protection area`

Supplementary conditions

- When forming a collision pair, it should be observed that every protection area of this pair has at least one protection area element that is marked with $NP_USAGE (Page 291) = "C" or "A". Otherwise, with the protection area, it is not permissible to perform a collision or clearance calculation (Page 334).
- The protection areas of a collision pair are only checked for collision when both protection areas are in the "Protection area monitored" state. See Chapter "State diagram: Protection area (Page 316)".

7.2.3.3 $NP_SAFETY_DIST

Function

The collision pair-specific safety clearance is entered in the system variable. The collision avoidance ensures that this safety clearance is not violated.

If a value not equal to 0.0 is entered in the system variable, the general safety clearance from MD10622 $MN_COLLISION_SAFETY_DIST (Page 323) is not taken into account for this collision pair.

If the value 0.0 is entered in the system variable, the safety clearance set in the machine data applies.

Syntax

```
$NP_SAFETY_DIST[<m>] = <value>
```

Meaning

`$NP_SAFETY_DIST`:	Safety clearance of the collision pair	
	Data type:	REAL
	Default value:	0.0

`<m>`:	System variable or protection area index	
	Data type:	INT
	Range of values:	0, 1, 2, ... (M -1) [1)]
`<value>`:	Safety clearance	
	Data type:	REAL
	Range of values:	0.0 ≤ x ≤ +max. REAL value
	Unit:	mm or inch depending on the current dimensions setting
1) M = n * (n - 1) / 2 where n = $MN_MM_MAXNUM_3D_PROT_AREAS		

Example

The safety clearance for the protection areas of the seventh collision pair should be 1.0 mm (input system: metric).

Program code	Comment
`N100 $NP_SAFETY_DIST[6] = 1.0`	`; 7. collision pair,`
	`; safety clearance = 1.0`

See also

Reaction of the control to a risk of collision (Page 313)

7.2.4 Extend system variables

7.2.4.1 Overview

Further information on the internal states and values of the collision avoidance can be read via the following system variables:

- State data (Page 329)
- Memory requirement (Page 329)
- Braking distance estimations (Page 330)

7.2.4.2 State data

The state data of the collision avoidance can be read via the following system variables (OPI variables)

System variable	OPI variable	Meaning
$AN_COLL_STATE[<m>]	anCollState[<m>]	Current state of a protection area (active/inactive) with regard to the collision avoidance
$AN_COLL_STATE_COND[<m>]	anCollStateCond[<m>]	Monitoring state (bit-coded) of a protection area
$AN_COLL_IPO_ACTIVE	anCollIpoActive	Activation state of the collision avoidance in the main run (active/inactive)
$AN_COLL_IPO_LIMIT	anCollIpoLimit	Velocity reduction through collision avoidance in the main run (active/inactive)
$AN_COLL_LOAD[<i>] 1)	anCollLoad[<i>] 1)	Computation time requirement for the collision avoid-ance function <i>
$AN_ACTIVATE_COLL_CHECK[<j>]	anActivateColl-Check[<j>]	Current state of the NC/PLC interface with index <j> for each 8 bytes: DB10, DBX234.0 - DBX.241.7 (activate protection areas)
$AN_COLL_CHECK_OFF	anCollCheckOff	Current state of the NC/PLC interface: DB10 DBX58.0 - 7 (disable protection area group)
$AA_COLLPOS[<a>]	aaCollPos	Position of axis <a> in the machine coordinate system (MCS) when the last collision alarm occurred
$AC_COLLPOS[<k>]	acCollPos	Vector <k> for the collision position in the world coor-dinate system when the last collision alarm occurred
a: Axis name i: 0 = function 1, 1 = function 2, 2 = function 3, ... j: Index 0, 1, 2, ... for one bit field each of 8 bytes. k: Coordinate index k = 1, 2, 3 for X, Y, Z coordinates m: System variable or protection area index 0, 1, 2, ... (MD18890 $MN_MM_MAXNUM_3D_PROT_AREAS - 1)		
1) The system variable can be reset by writing the value 0. Every other value is rejected with an error message.		

References

- For a detailed description of the system variables, refer to:
  List Manual, System Variables
- A detailed description of the interface signals can be found in:
  List Manual, Interface signals

7.2.4.3 Memory requirement

The data for the memory requirement of the collision avoidance can be read via the following system variables (OPI variables).

System variable	OPI variable	Meaning
$AN_COLL_MEM_AVAILABLE	anCollMemAvailable	Size of the memory space in KB reserved for the collision avoidance.
$AN_COLL_MEM_USE_MIN 1)	anCollMemUseMin 1)	Minimum value of the memory space used for the collision avoidance as a percentage of the reserved memory.

System variable	OPI variable	Meaning
$AN_COLL_MEM_USE_MAX [1)]	anCollMemUseMax [1)]	Maximum value of the memory space used for the collision avoidance as a percentage of the reserved memory.
$AN_COLL_MEM_USE_ACT [1)]	anCollMemUseAct [1)]	Actual value of the memory space used for the collision avoidance as a percentage of the reserved memory.
1) The system variable can be reset by writing the value 0. Every other value is rejected with an error message.		

References

For a detailed description of the system variables, refer to:

List Manual, System Variables

7.2.4.4 Braking distance estimations

The estimated total braking distance (linear approximation) and the proportional braking distances of superimposed motions for an axis can be read via the following system variables (OPI variables). The estimation only takes the current state of the axis into account. It returns, e.g. the braking distance 0.0 for an axis that is just at the reversal point as part of a circular path.

Note

The system variables are used for support in the development of user-specific functions as part of the collision avoidance, and similar functions.

Table 7-1 Basic coordinate system (BCS)

System variable	OPI variable	Meaning
Total braking distance		
$AA_DTBREB[<a>]	aaDtbreb	Estimated, linearly approximated total braking distance
Proportional braking distances for superimposed motions		
$AA_DTBREB_CMD[<a>]	aaDtbrebCmd	Command component
$AA_DTBREB_CORR[<a>]	aaDtbrebCorr	Correction component
$AA_DTBREB_DEP[<a>]	aaDtbrebDep	Coupling component
<a>: Axis name		

Table 7-2 Machine coordinate system (MCS)

System variable	OPI variable	Meaning
Total braking distance		
$AA_DTBREM[<a>]	aaDtbrem	Estimated, linearly approximated total braking distance
Proportional braking distances for superimposed motions		
$AA_DTBREM_CMD[<a>]	aaDtbremCmd	Command component
$AA_DTBREM_CORR[<a>]	aaDtbremCorr	Correction component

System variable	OPI variable	Meaning
$AA_DTBREM_DEP[<a>]	aaDtbremDep	Coupling component
<a>: Axis name		

References

For a detailed description of the system variables, refer to:

List Manual, System Variables

7.3 Programming

7.3.1 Check for collision pair (COLLPAIR)

The `COLLPAIR()` function determines whether two protection areas form a collision pair.

Syntax

```
[<RetVal> =] COLLPAIR(<Name_1>,<Name_2>[,<NoAlarm>)])
```

Meaning

`COLLPAIR`:	Check whether part of a collision pair		
`<RetVal>`:	Function return value		
	Data type:	INT	
	Value:	≥ 0	The two protection zones form a collision pair. Return value == collision pair index m (see $NP_COLL_PAIR (Page 326))
		-1	Either two strings have not been specified or at least one of the two is the zero string.
		-2	The protection zone specified in the first parameter has not been found.
		-3	The protection zone specified in the second parameter has not been found.
		-4	Neither of the two specified protection zones has been found.
		-5	Both specified protection zones have been found, but not together in a collision pair.
`<Name_1>`:	Name of the first protection zone		
	Data type:	STRING	
	Range of values:	Parameterized protection zone names	

`<Name_2>`:	Name of the second protection area		
	Data type:	STRING	
	Range of values:	Parameterized protection zone names	
`<NoAlarm>`:	Alarm suppression (**optional**)		
	Data type:	BOOL	
	Value:	FALSE (Default)	In the event of an error (<RetVal> < 0), the program processing is stopped and an alarm displayed.
		TRUE	In the event of an error, the program processing is not stopped and no alarm displayed. Application: User-specific reaction corresponding to the return value

See also

State diagram: Protection area (Page 316)

7.3.2 Request recalculation of the machine model of the collision avoidance (PROTA)

If system variables of the kinematic chain $NK_..., the geometric machine modeling or the collision avoidance $NP_... are written in the part program, the `PROTA` procedure must subsequently be called so that the change becomes effective in the NC-internal machine model of the collision avoidance.

Syntax

```
PROTA[(<Par>)]
```

Meaning

`PROTA`:	Request recalculation of the machine model of the collision avoidance		
	• Triggers a preprocessing stop. • Must be alone in the block.		
`<Par>`:	Parameter (**optional**)		
	Data type:	STRING	
	Value:	---	No parameters. The machine model is recalculated. The states of the protection areas are retained.
		"R"	The machine model is recalculated. The protection areas are set to their initialization status corresponding to $NP_INIT_STAT (Page 282).

Supplementary conditions

Simulation

The `PROTA` procedure must not be used in part programs in conjunction with the simulation (simNC).

Example: Avoiding the `PROTA` call while the simulation is active.

Program code	Comment
...	
IF $P_SIM == FALSE	; IF simulation not active
PROTA	THEN recalculate collision model
ENDIF	; ENDIF
...	

See also

Setting the protection zone status (PROTS) (Page 333)

7.3.3 Setting the protection zone status (PROTS)

The `PROTS()` procedure sets the state of protection areas to the specified value.

Syntax

```
PROTS(<State>[,<Name_1>,...,<Name_n>])
```

Meaning

`PROTS`:	Sets the state of protection areas		
	• Must be alone in the block.		
`<State>`:	Status to which the specified protection zones are to be set		
	Data type:	CHAR	
	Value:	"A"or "a"	Status: Active
		"I"or "i"	Status: Inactive
		"P"or "p"	Status: Preactivated or PLC-controlled 1)
		"R"or "r"	Status: NC-internal value of the initialization status 2)

`<Name_1> ... <Name_n>`:	Name of one or more protection areas that are to be set to the specified status (**optional**) If no name is specified, the specified status is set for all defined protection zones.	
	Data type:	STRING
	Range of values:	Parameterized protection zone names
	Note The maximum number of protection areas that can be specified as parameters depends only on the maximum possible number of characters per program line.	
$^{1)}$ The activation/deactivation is performed via: DB10.DBX234.0 - DBX241.7 $^{2)}$ The status is set to the NC-internal value of the initialization status, i.e. to the value that the system variable $NP_INIT_STAT (Page 282) had at the time of the last PROTA( ) (Page 332) call.		

7.3.4 Determining the clearance of two protection zones (PROTD)

The `PROTD()` function calculates the clearance of two protection areas.

Function properties:

- The clearance calculation is performed independent of the protection area status (activated, deactivated, preactivated).
- To calculate the clearance of two protection areas, only protection area elements are used, which are marked with $NP_USAGE (Page 291) = "C" or "A". Protection area elements of the protection area, which are marked with $NP_USAGE = "V", are not taken into consideration.
- Protection areas, where all protection area elements of the protection area are marked with $NP_USAGE = "V", cannot be used for the clearance calculation.
- The clearance calculation is performed with the positions valid at the end of the previous block.
- Overlays that are included in the main run calculation (e.g. DRF offset or external zero offset) are included in the clearance calculation with the values valid at the function **interpretation time**.

Note

Synchronization

When using the `PROTD()` function, it is the sole responsibility of the user to synchronize the main run and preprocessing, if required, with the `STOPRE` preprocessing stop.

Collision

If there is a collision between the specified protection areas, the function returns a clearance of 0.0. There is a collision if both the protection areas touch or intersect each other.

The safety clearance for the collision check (MD10622 $MN_COLLISION_SAFETY_DIST) is not taken into account in the clearance calculation.

Syntax

```
[<RetVal> =] PROTD([<Name_1>],[<Name_2>],VAR <Vector>[,<System>])
```

Meaning

`PROTD:`	Calculates the clearance of the two specified protection areas.		
	• Must be alone in the block.		
`<RetVal>:`	Function return value: Absolute clearance value of the two protection areas or 0.0 with collision (see above: Collision paragraph)		
	Data type:	REAL	
	Range of values:	0.0 ≤ x ≤ +max. REAL value	
`<Name_1>,` `<Name_2>:`	Names of the two protection areas whose clearance is to be calculated (**optional**)		
	Data type:	STRING	
	Range of values:	Parameterized protection area names	
	Default value:	"" (empty string) If no protection areas have been specified, the function calculates the current smallest clearance from all the activated and preactivated protection areas in the collision model.	
`<Vector>:`	Return value: 3-dimensional clearance vector from protection area <Name_2> to protection area <Name_1> with: • <Vector>[0]: X coordinate in the world coordinate system • <Vector>[1]: Y coordinate in the world coordinate system • <Vector>[2]: Z coordinate in the world coordinate system For collision: <Vector> == zero vector		
	Data type:	VAR REAL [3]	
	Range of values:	<Vector> [n]: 0.0 ≤ x ≤ ±max. REAL value	
`<System>:`	Measuring system (inch/metric) for clearance and clearance vector (**optional**)		
	Data type:	BOOL	
	Value:	FALSE (Default)	Measuring system corresponding to the currently active G command from G group 13 (G70, G71, G700, G710).
		TRUE	Measuring system corresponding to the set basic system: MD52806 $MN_ISO_SCALING_SYSTEM

7.4 Example

7.4.1 Specifications

General Information

The basic procedure when parameterizing collision avoidance using a part program is shown using a simplified 3-axis milling machine as example. All of the system variables relevant for collision avoidance are written to the part program:

- Kinematic chain $NK_...
- Geometric machine modeling $NP_...
- Collision avoidance $NP_...

The machine model is subsequently activated in the part program, so that after executing the part program, the collision avoidance has been completely parameterized in the 3-axis milling machine and is active.

Option and machine data

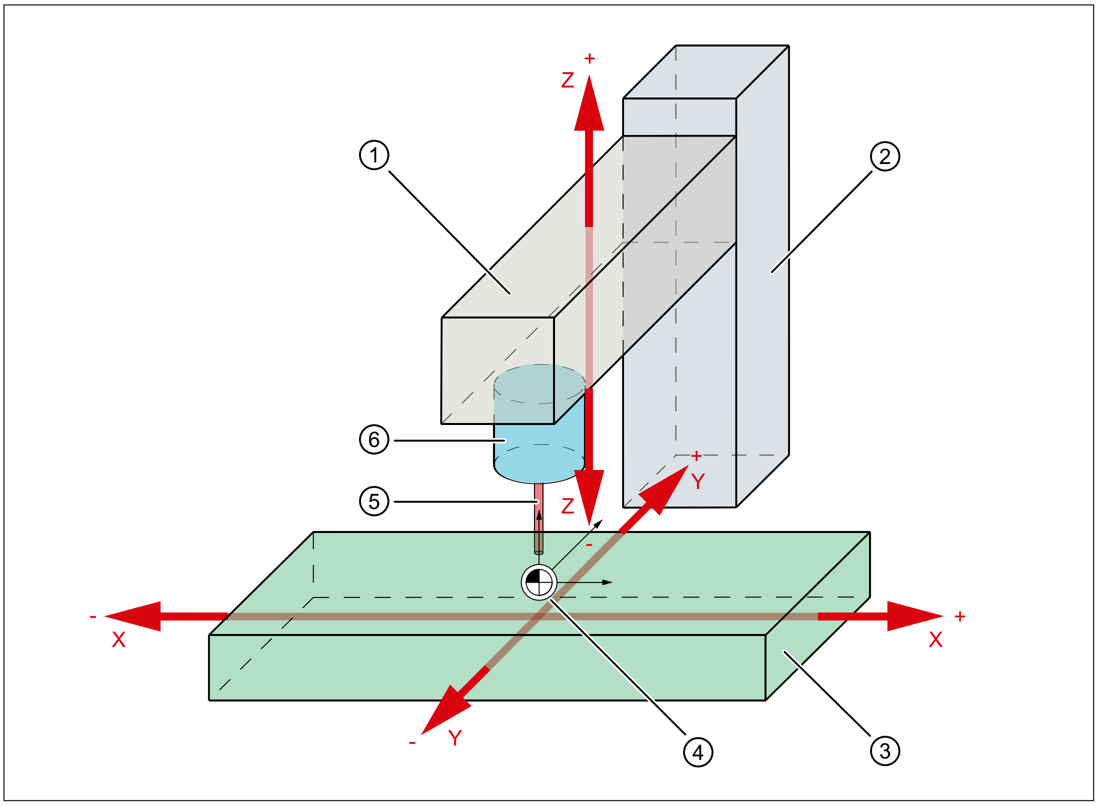
The following option and machine data must be set for the example:

No.	Option data: $ON_	Value
MD19830	COLLISION_MASK	1

No.	Machine data: $MN_	Value
MD10619	COLLISION_TOLERANCE	1
MD18880	MM_MAXNUM_KIN_CHAIN_ELEM	10
MD18890	MM_MAXNUM_3D_PROT_AREAS	10
MD18892	MM_MAXNUM_3D_PROT_AREA_ELEM	10
MD18893	MM_MAXNUM_3D_T_PROT_ELEM	1
MD18894	MM_MAXNUM_3D_FACETS_INTERN	1000
MD18895	MM_MAXNUM_3D_FACETS	3000
MD18896	MM_MAXNUM_3D_COLLISION	0
MD18897	MM_MAXNUM_3D_INTERFACE_IN	16
MD18899	PROT_AREA_TOOL_MASK	1

Principle design of the 3-axis milling machine

The principle machine design is shown in the following diagram.



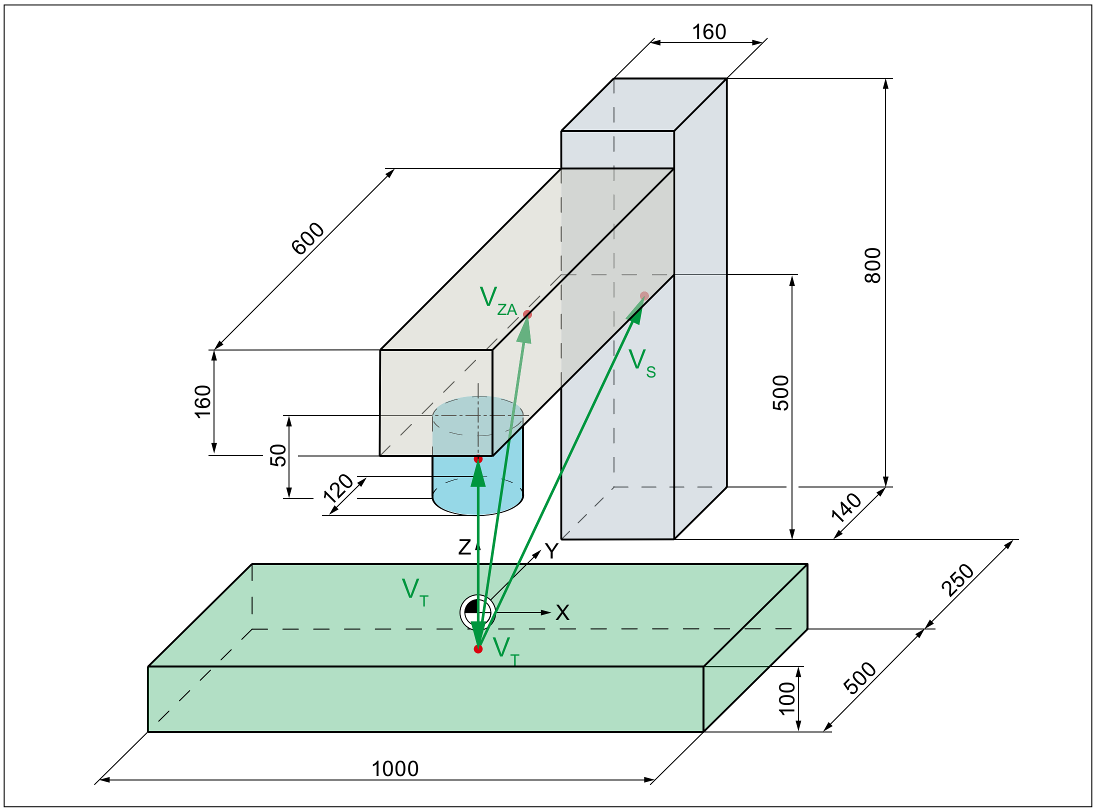
①	Z axis
②	Stand
③	Table
④	Machine zero = reference point
⑤	Tool
⑥	Tool holder

The machine parts and/or protection areas are assigned to the following machine axes.

Machine parts and/or protection areas	Machine axis
Table	X1, Y1
Z axis	Z1
Column	---
Tool holder	Z1
Tool	Z1

Dimension drawing

The dimensions of the protection area elements as well as their position (vectors to the center point of the protection area element) referred to the machine zero point are specified in the following dimension drawing.



Vectors to the center point of the protection area elements

v_{tool}	Tool adapter (0;0;25)
v_{ZA}	Z axis (0;200;130)
v_S	Column (0;570;350)
v_T	Table (0;0;-50)

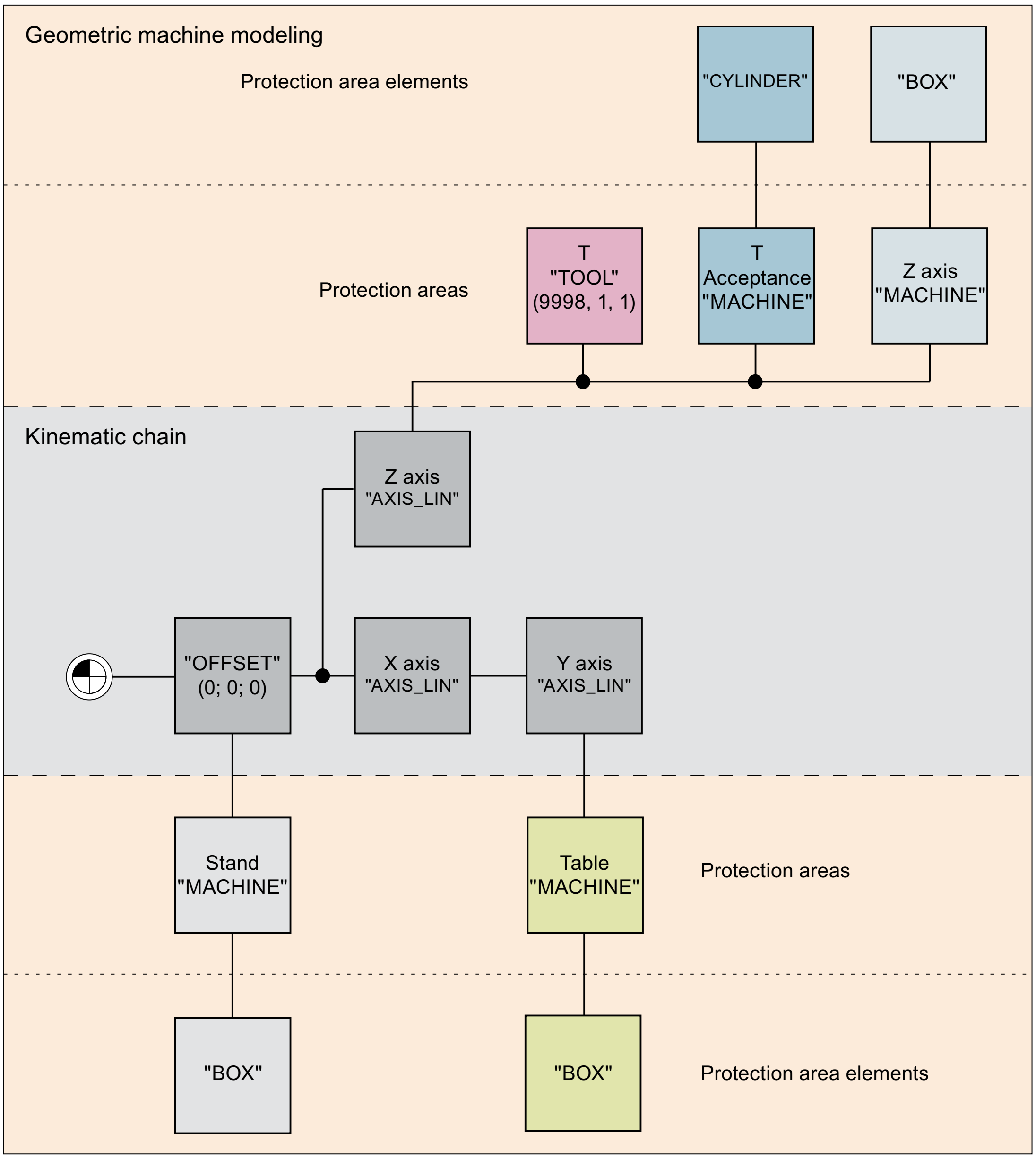
Kinematic chain

The kinematic chain (see next diagram) starts with an "Offset" type element. These are assigned to all static protection areas of the machine. In the example, this is only the "Column" protection area.

The kinematic elements of the machine axes follow the offset element:

- Z axis and X axis traverse independently of one another ⇒ $NK_PARALLEL
- The Y axis traverses dependent on the X axis ⇒ $NK_NEXT

The various protection areas of the geometric machine modeling are assigned the kinematic elements of the machine axes



Collision pairs

For the example, it is assumed that only the following collision pairs are to be taken into account:

- Tool adapter - table
- Tool - table

7.4.2 Part program of the machine model

Program code

```
;***********************************************************
;************************ Example **********************
; milling machine: 3 linear axes, 1 spindle
;   table   => X1, Y1
;   Z axis, tool adapter, tool => Z1
;***********************************************************
; status: 2/11/2013, 15:34
;
;===========================================================
; collision machine data used
;===========================================================
; MD10619 $MN_COLLISION_TOLERANCE = 1
;
; MD18880 $MN_MM_MAXNUM_KIN_CHAIN_ELEM    =  100
; MD18890 $MN_MM_MAXNUM_3D_PROT_AREAS     =   10
; MD18892 $MN_MM_MAXNUM_3D_PROT_AREA_ELEM =   10
; MD18893 $MN_MM_MAXNUM_3D_T_PROT_ELEM    =  100
; MD18894 $MN_MM_MAXNUM_3D_FACETS_INTERN  = 1000
; MD18895 $MN_MM_MAXNUM_3D_FACETS         = 3000
; MD18896 $MN_MM_MAXNUM_3D_COLLISION      =    0
; MD18897 $MN_MM_MAXNUM_3D_INTERFACE_IN   =   16
; MD18899 $MN_PROT_AREA_TOOL_MASK         =    1
;
; MD19830 $ON_COLLISION_MASK = 1   ; option
;
;
;===========================================================
; definitions
;===========================================================
DEF INT RETVAL = 0      ; return value of the delete function
;
DEF INT C_NKE = 0       ; index for kinematic elements
DEF INT C_NPC = 0       ; index for protection areas
DEF INT C_NPE = 0       ; index for protection area elements
DEF INT C_NPP = 0       ; index for collision pairs
;
;
```

Program code

```
;============================================================
; initialization of the collision data
;============================================================
MSG("protection areas")
G4 F3
; reset all parameters to their initial setting
;
RETVAL = DELOBJ("KIN_CHAIN_ELEM")
IF (RETVAL <> 0)
   MSG("error: DELOBJ KIN_CHAIN_ELEM")
   G4 F5
ENDIF
;
RETVAL = DELOBJ("PROT_AREA_ALL")
IF RETVAL <> 0
   MSG("error: DELOBJ PROT_AREA_ALL")
   G4 F5
ENDIF
;
RETVAL = DELOBJ("PROT_AREA_COLL_PAIRS")
IF RETVAL <> 0
   MSG("error: DELOBJ PROT_AREA_COLL_PAIRS")
   G4 F5
ENDIF
;
;
;============================================================
; kinematic chain
;============================================================
; KE1: ROOT
; ----------------------------------------------------------
$NK_NAME[C_NKE]       = "ROOT"
$NK_NEXT[C_NKE]       = "X axis"
$NK_PARALLEL[C_NKE]   = ""
$NK_TYPE[C_NKE]       = "OFFSET"
;
$NK_OFF_DIR[C_NKE, 0] = 0.0     ; X
$NK_OFF_DIR[C_NKE, 1] = 0.0     ; Y
$NK_OFF_DIR[C_NKE, 2] = 0.0     ; Z
;
$NK_AXIS[C_NKE]       = ""
$NK_A_OFF[C_NKE]      = 0.0
;
C_NKE = C_NKE + 1           ; next kinematic element
;
```

Program code

```
; ------------------------------------------------------------
; kinematic element: X axis
; ------------------------------------------------------------
$NK_NAME[C_NKE]       = "X axis"
$NK_NEXT[C_NKE]       = "Y axis"
$NK_PARALLEL[C_NKE]   = "Z axis"
$NK_TYPE[C_NKE]       = "AXIS_LIN"
;
$NK_OFF_DIR[C_NKE, 0] = 1.0     ; X
$NK_OFF_DIR[C_NKE, 1] = 0.0     ; Y
$NK_OFF_DIR[C_NKE, 2] = 0.0     ; Z
;
$NK_AXIS[C_NKE]       = "X1"
$NK_A_OFF[C_NKE]      = 0.0
;
C_NKE = C_NKE + 1           ; next kinematic element
;
; ------------------------------------------------------------
; kinematic element: Y axis
; ------------------------------------------------------------
$NK_NAME[C_NKE]       = "Y axis"
$NK_NEXT[C_NKE]       = ""
$NK_PARALLEL[C_NKE]   = ""
$NK_TYPE[C_NKE]       = "AXIS_LIN"
;
$NK_OFF_DIR[C_NKE, 0] = 0.0     ; X
$NK_OFF_DIR[C_NKE, 1] = 1.0     ; Y
$NK_OFF_DIR[C_NKE, 2] = 0.0     ; Z
;
$NK_AXIS[C_NKE]       = "Y1"
$NK_A_OFF[C_NKE]      = 0.0
;
C_NKE = C_NKE + 1           ; next kinematic element
;
; ------------------------------------------------------------
; kinematic element: Z axis
; ------------------------------------------------------------
$NK_NAME[C_NKE]       = "Z axis"
$NK_NEXT[C_NKE]       = ""
$NK_PARALLEL[C_NKE]   = ""
$NK_TYPE[C_NKE]       = "AXIS_LIN"
;
$NK_OFF_DIR[C_NKE, 0] = 0.0     ; X
$NK_OFF_DIR[C_NKE, 1] = 0.0     ; Y
$NK_OFF_DIR[C_NKE, 2] = 1.0     ; Z
;
$NK_AXIS[C_NKE]      = "Z1"
$NK_A_OFF[C_NKE]     = 0.0
;
C_NKE = C_NKE + 1           ; next kinematic element
;
;
```

Program code

```
;============================================================
; protection areas with protection area elements
;============================================================
; protection area 1: Column
; ----------------------------------------------------------
$NP_PROT_NAME[C_NPC]  = "column"
$NP_PROT_TYPE[C_NPC]  = "MACHINE"
$NP_CHAIN_ELEM[C_NPC] = "ROOT"
$NP_1ST_PROT[C_NPC]   = "SBE column"
$NP_PROT_COLOR[C_NPC] = 'HFFA0A0A4'   ; AARRGGBB
$NP_BIT_NO[C_NPC]     = -1
$NP_INIT_STAT[C_NPC]  = "A"
;
C_NPC = C_NPC + 1                  ; next protection area
;
; ----------------------------------------------------------
; protection area element 1.1: SBE column
; ----------------------------------------------------------
$NP_NAME[C_NPE]     = "SBE column"
$NP_NEXT[C_NPE]     = ""
$NP_NEXTP[C_NPE]    = ""
$NP_TYPE[C_NPE]     = "BOX"
;                             cube dimensions
$NP_PARA[C_NPE,0]   = 160.0    ; length
$NP_PARA[C_NPE,1]   = 140.0    ; width
$NP_PARA[C_NPE,2]   = 800.0    ; height
;                             center point
$NP_OFF[C_NPE,0]    =   0.0    ; X
$NP_OFF[C_NPE,1]    = 570.0    ; Y: xxx rear table edge
$NP_OFF[C_NPE,2]    = 350.0    ; Z: Lower edge equal to lower edge table
;
$NP_DIR[C_NPE,0]    =   0.0
$NP_DIR[C_NPE,1]    =   0.0
$NP_DIR[C_NPE,2]    =   0.0
;
$NP_ANG[C_NPE]      =   0.0
;
$NP_COLOR[C_NPE]    =   0
$NP_D_LEVEL[C_NPE]  =   0
$NP_USAGE[C_NPE]    = "V"      ; V = visualize only
$NP_FILENAME[C_NPE] = ""
;
C_NPE = C_NPE + 1           ; next protection area element
;
```

Program code

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; protection area 2: Tool adapter
; ----------------------------------------------------------
$NP_PROT_NAME[C_NPC]  = "tool adapter"
$NP_PROT_TYPE[C_NPC]  = "MACHINE"
$NP_CHAIN_ELEM[C_NPC] = "Z axis"
$NP_1ST_PROT[C_NPC]   = "SBE tool adapter"
$NP_PROT_COLOR[C_NPC] = 'HFF0000FF'   ; AARRGGBB
$NP_BIT_NO[C_NPC]     = -1
$NP_INIT_STAT[C_NPC]  = "A"
;
C_NPC = C_NPC + 1                  ; next protection area
;
; ----------------------------------------------------------
; protection area element 2.1: SBE tool adapter
; ----------------------------------------------------------
$NP_NAME[C_NPE]     = "SBE tool adapter"
$NP_NEXT[C_NPE]     = ""
$NP_NEXTP[C_NPE]    = ""
$NP_TYPE[C_NPE]     = "CYLINDER"
;                            cylinder dimensions
$NP_PARA[C_NPE,0]   = 50.0   ; height
$NP_PARA[C_NPE,1]   = 60.0   ; radius
$NP_PARA[C_NPE,2]   =  0.0
;                            center point
$NP_OFF[C_NPE,0]    =  0.0   ; X
$NP_OFF[C_NPE,1]    =  0.0   ; Y
$NP_OFF[C_NPE,2]    = 25.0   ; Z: Half height
;
$NP_DIR[C_NPE,0]    =  0.0
$NP_DIR[C_NPE,1]    =  0.0
$NP_DIR[C_NPE,2]    =  0.0
;
$NP_ANG[C_NPE]      = 0.0
;
$NP_COLOR[C_NPE]    = 0
$NP_D_LEVEL[C_NPE]  = 0
$NP_USAGE[C_NPE]    = "A"
$NP_FILENAME[C_NPE] = ""
;
C_NPE = C_NPE + 1           ; next protection area element
;
```

Program code

```
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; protection area 3: Tool
; ----------------------------------------------------------
$NP_PROT_NAME[C_NPC]  = "WKZ"
$NP_PROT_TYPE[C_NPC]  = "TOOL"
$NP_CHAIN_ELEM[C_NPC] = "Z axis"
$NP_1ST_PROT[C_NPC]   = ""
$NP_PROT_COLOR[C_NPC] = 'HFFFF0000'   ; AARRGGBB
$NP_BIT_NO[C_NPC]     = -1
$NP_INIT_STAT[C_NPC]  = "A"
;                              only relevant for type "TOOL"
$NP_INDEX[C_NPC,0]    = 1      ; tool location No. / spindle No.
$NP_INDEX[C_NPC,1]    = 9998   ; magazine No. / -
$NP_INDEX[C_NPC,2]    = 1      ; TOA area
;
C_NPC = C_NPC + 1                  ; next protection area
;
```

Program code

```
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; protection area 4: Z axis
; ----------------------------------------------------------
$NP_PROT_NAME[C_NPC]  = "Z axis"
$NP_PROT_TYPE[C_NPC]  = "MACHINE"
$NP_CHAIN_ELEM[C_NPC] = "Z axis"
$NP_1ST_PROT[C_NPC]   = "SBE Z axis"
$NP_PROT_COLOR[C_NPC] = 'HFFA0A0A4'   ; AARRGGBB
$NP_BIT_NO[C_NPC]     = -1
$NP_INIT_STAT[C_NPC]  = "A"
;
C_NPC = C_NPC + 1                  ; next protection area
;
; ----------------------------------------------------------
; protection area element 4.1: SBE horizontal column
; ----------------------------------------------------------
$NP_NAME[C_NPE]     = "SBE Z axis"
$NP_NEXT[C_NPE]     = ""
$NP_NEXTP[C_NPE]    = ""

$NP_TYPE[C_NPE]     = "BOX"
;
$NP_PARA[C_NPE,0]   = 160.0
$NP_PARA[C_NPE,1]   = 600.0
$NP_PARA[C_NPE,2]   = 160.0
;
$NP_OFF[C_NPE,0]    =   0.0
$NP_OFF[C_NPE,1]    = 200.0
$NP_OFF[C_NPE,2]    = 130.0
;
$NP_DIR[C_NPE,0]    = 0.0
$NP_DIR[C_NPE,1]    = 0.0
$NP_DIR[C_NPE,2]    = 0.0
;
$NP_ANG[C_NPE]      = 0.0
;
$NP_COLOR[C_NPE]    = 0
$NP_D_LEVEL[C_NPE]  = 0
$NP_USAGE[C_NPE]    = "A"
$NP_FILENAME[C_NPE] = ""
;
C_NPE = C_NPE + 1           ; next protection area element
;
```

Program code

```
; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; protection area 5: Table
; -------------------------------------------------------
$NP_PROT_NAME[C_NPC]  = "table"
$NP_PROT_TYPE[C_NPC]  = "MACHINE"
$NP_CHAIN_ELEM[C_NPC] = "Y axis"
$NP_1ST_PROT[C_NPC]   = "SBE table"
$NP_PROT_COLOR[C_NPC] = 'HFF00FF00'   ; AARRGGBB
$NP_BIT_NO[C_NPC]     = -1
$NP_INIT_STAT[C_NPC]  = "A"
;
C_NPC = C_NPC + 1                  ; next protection area
;
; ---------------------------------------------------------
; protection area element 5.1: SBE table
; ---------------------------------------------------------
$NP_NAME[C_NPE]      = "SBE table"
$NP_NEXT[C_NPE]      = ""
$NP_NEXTP[C_NPE]     = ""
$NP_TYPE[C_NPE]      = "BOX"
;
$NP_PARA[C_NPE,0]    = 1000.0
$NP_PARA[C_NPE,1]    = 500.0
$NP_PARA[C_NPE,2]    = 100.0
;
$NP_OFF[C_NPE,0]     =   0.0
$NP_OFF[C_NPE,1]     =   0.0
$NP_OFF[C_NPE,2]     = -50.0
;
$NP_DIR[C_NPE,0]     = 0.0
$NP_DIR[C_NPE,1]     = 0.0
$NP_DIR[C_NPE,2]     = 0.0
;
$NP_ANG[C_NPE]       = 0.0
;
$NP_COLOR[C_NPE]     = 0
$NP_D_LEVEL[C_NPE]   = 0
$NP_USAGE[C_NPE]     = "A"
$NP_FILENAME[C_NPE] = ""
;
C_NPE = C_NPE + 1           ; next protection area element
;
;
```

Program code

```
;============================================================
; collision pairs
;============================================================
$NP_COLL_PAIR[C_NPP, 0] = "Tool adapter"
$NP_COLL_PAIR[C_NPP, 1] = "Table"
;
C_NPP = C_NPP + 1                  ; next collision pair
;
$NP_COLL_PAIR[C_NPP, 0] = "T"
$NP_COLL_PAIR[C_NPP, 1] = "Table"
;
C_NPP = C_NPP + 1                  ; next collision pair
;
;
;============================================================
; activating the machine model
;============================================================
PROTA
PROTS("A")
;
M2
;======================== END ===========================
```

7.5 Data lists

7.5.1 Machine data

7.5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
MD10619	COLLISION_TOLERANCE	Collision tolerance
MD10622	COLLISION_SAFETY_DIST	Safety clearance
MD18896	MM_MAXNUM_3D_COLLISION	Memory location for the collision avoidance

7.5.2 System variables

Identifier	Description
$NP_COLL_PAIR	Name of the first or second protection area of a collision pair
$NP_SAFETY_DIST	Safety clearance of the collision pair
$AN_COLL_STATE	Current state of a protection area with regard to the collision avoidance
$AN_COLL_STATE_COND	Monitoring state (bit-coded) of a protection area

Identifier	Description
$AN_COLL_IPO_ACTIVE	Activation state of the collision avoidance in the main run
$AN_COLL_IPO_LIMIT	Velocity reduction through collision avoidance in the main run
$AN_COLL_LOAD	Computation time requirement for the collision avoidance function
$AN_ACTIVATE_COLL_CHECK	Current state of the NC/PLC interface DB10, DBX234.0 - DBX.241.7 (activate protection areas)
$AN_COLL_CHECK_OFF	Current state of the NC/PLC interface DB10, DBB58 (switch off protection area groups depending on the operating mode)
$AA_COLLPOS	Position of an axis in the MCS when the last collision alarm occurred
$AC_COLLPOS	Vector for the collision position in the world coordinate system when the last collision alarm occurred
$AN_COLL_MEM_AVAILABLE	Size of the memory space in KB reserved for the collision avoidance.
$AN_COLL_MEM_USE_MIN	Minimum value of the memory space used for the collision avoidance as a percentage of the reserved memory.
$AN_COLL_MEM_USE_MAX	Maximum value of the memory space used for the collision avoidance as a percentage of the reserved memory.
$AN_COLL_MEM_USE_ACT	Actual value of the memory space used for the collision avoidance as a percentage of the reserved memory.
$AA_DTBREB	Estimated, linearly approximated total braking distance (BCS)
$AA_DTBREB_CMD	Command component (BCS)
$AA_DTBREB_CORR	Correction component (BCS)
$AA_DTBREB_DEP	Coupling component (BCS)
$AA_DTBREM	Estimated, linearly approximated total braking distance (MCS)
$AA_DTBREM_CMD	Command component (MCS)
$AA_DTBREM_CORR	Correction component (MCS)
$AA_DTBREM_DEP	Coupling component (MCS)

7.5.3 Signals

7.5.3.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Collision avoidance: Deactivate protection area **group** (PLC → NC)	DB10.DBX58.0 - 7	DB2600.DBX2.0 - 7
Collision avoidance: Activate protection area (PLC → NC)	DB10.DBX234.0 - DBX241.7	DB2600.DBX4.0 - DBX11.7

7.5.3.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Collision avoidance: Protection area active	DB10.DBX226.0 - DBX233.7	DB2700.DBX20.0 - DBX27.7

7.5.3.3 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Collision avoidance: Stop	DB21, ...DBX377.0	DB33xx.DBX4005.0

7.5.3.4 Signals from axis

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Collision avoidance: Velocity reduction	DB31, ...DBX77.0	DB39xx.DBX1003.0

K11 Collision avoidance, external 8

8.1 Description of the function

8.1.1 Options

The function "Collision avoidance, external" is an option that requires a license:

- **Collision avoidance ADVANCED** (machine, workpiece): 6FC5800-0AS04-0YB0

8.1.2 Characteristics

The function "Collision avoidance, external" provides a proprietary data interface at the Industrial Ethernet interface X120 of the NCU module. The following data are transferred in real time via this interface:

- Setpoint and actual values of the machine axes for monitoring and visualizing machine movements
- Status data and input data for influencing traversing movements
- Signals for monitoring the external application

Based on these data, an independent application for machine visualization and collision avoidance can be implemented by a SINUMERIK product partner on an external computer.

8.2 Commissioning

8.2.1 Machine data

8.2.1.1 Functions of external collision avoidance

The functions of external collision avoidance are enabled with machine data:

MD16900 $MN_COLLISION_EXT_FUNKTION_MASK.Bit x = 1

Bit	Meaning
0	Activate UDP interface
1 - 31	reserved

8.2.1.2 Preview time

The preview time is set with machine data:

MD16901 $MN_COLLISION_EXT_PREVIEW_TIME = <preview time>

8.2.1.3 Preview time step

The size of a time step for calculating the movement preview is set in machine data:

MD16902 $MN_COLLISION_EXT_PREVIEW_STEP = <preview time>

8.2.1.4 Timeout time

The time within which the external application must have reported back to the control for sign-of-life monitoring is set in machine data:

MD16903 $MN_COLLISION_EXT_TIMEOUT = <timeout time>

If the external application does not report back within the set time, alarm 26300 "Sign of life failure of the external collision avoidance" is displayed.

K12 transformation definitions with kinematic chains 9

9.1 Function description

9.1.1 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).

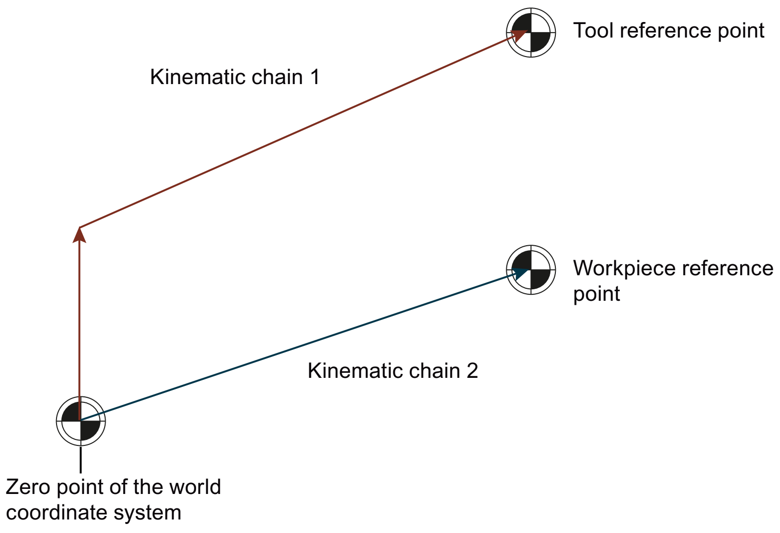

Tool reference point
Kinematic chain 1
Workpiece reference point
Kinematic chain 2
Zero point of the world coordinate system

Figure 9-1 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

See also

System variables (Page 236)

K7: Kinematic chain (Page 229)

9.1.2 Definition of kinematic transformations

Transformations with kinematic chains are defined via 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 K7: Kinematic chain (Page 229).

The system variables that are relevant to this are described under System variables (Page 236). 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 375). 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_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_CNTRL`	Control word

An orientation transformation is defined, for example, via the elements displayed in the graphic.

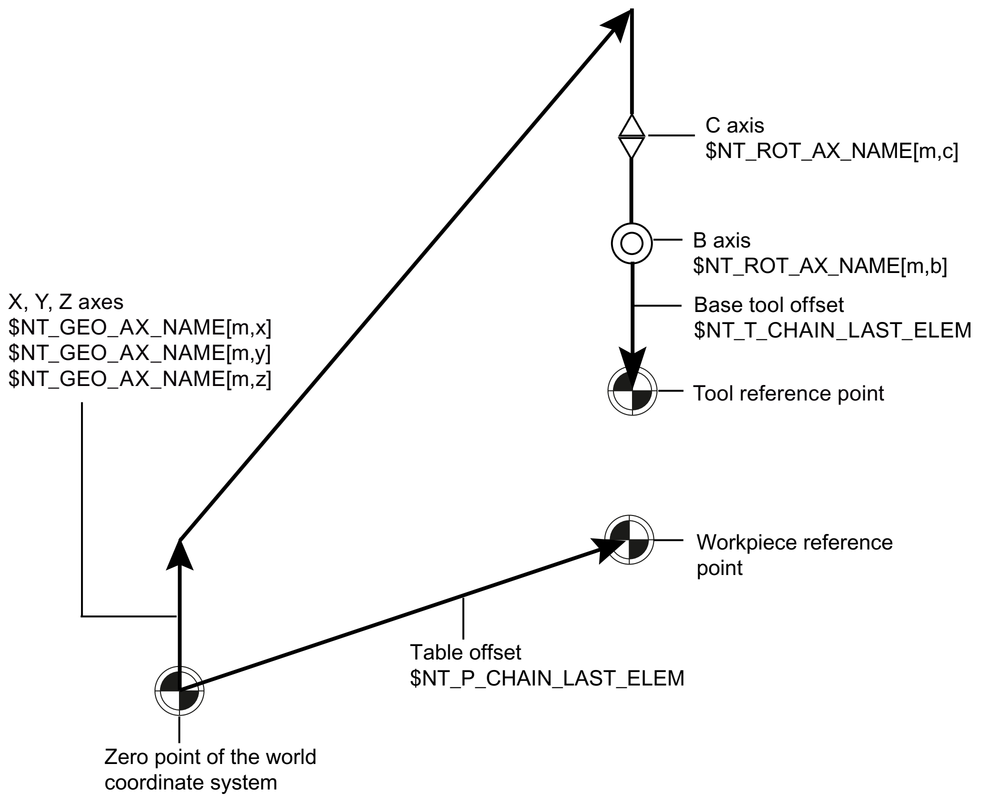


Figure 9-2 Transformation elements in the kinematic chain

See also

Examples (Page 421)

9.1.3 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 up to three linear axes in such a way that the coordinates of the tool center point 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 399).

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 45). 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 59). 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.

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 56) 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.

Machine measurement for orientation transformation

With the CORRTRAFO (Page 413) 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.

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.

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.

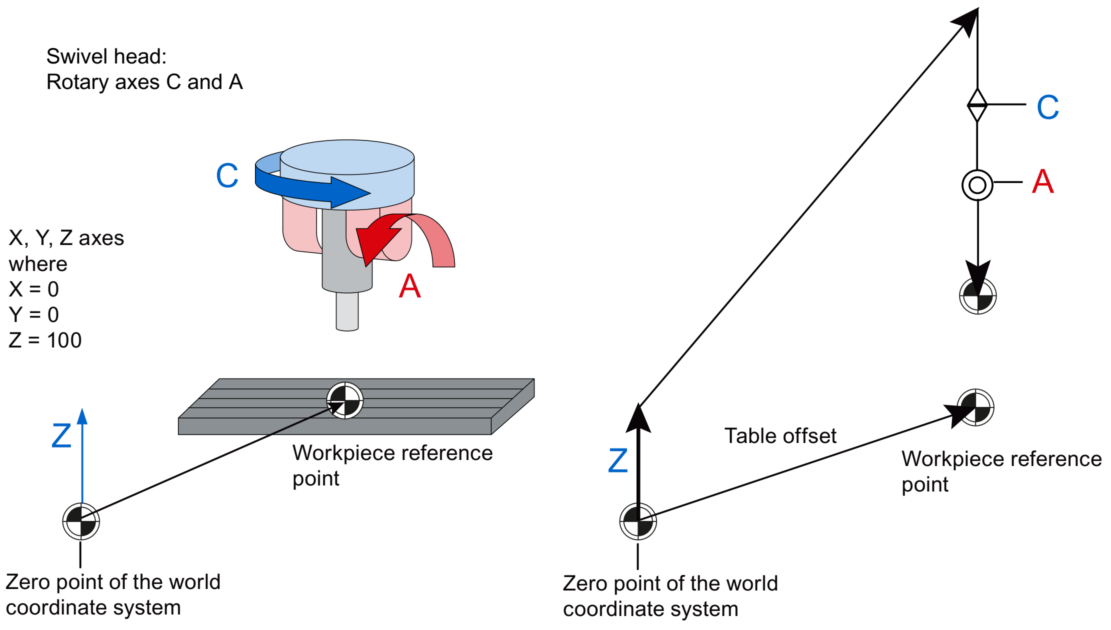

Swivel head:
Rotary axes C and A
C
A
X, Y, Z axes
where
X = 0
Y = 0
Z = 100
Z
Workpiece reference point
Zero point of the world coordinate system
C
A
Z
Table offset
Workpiece reference point
Zero point of the world coordinate system

Figure 9-3 TRAORI example

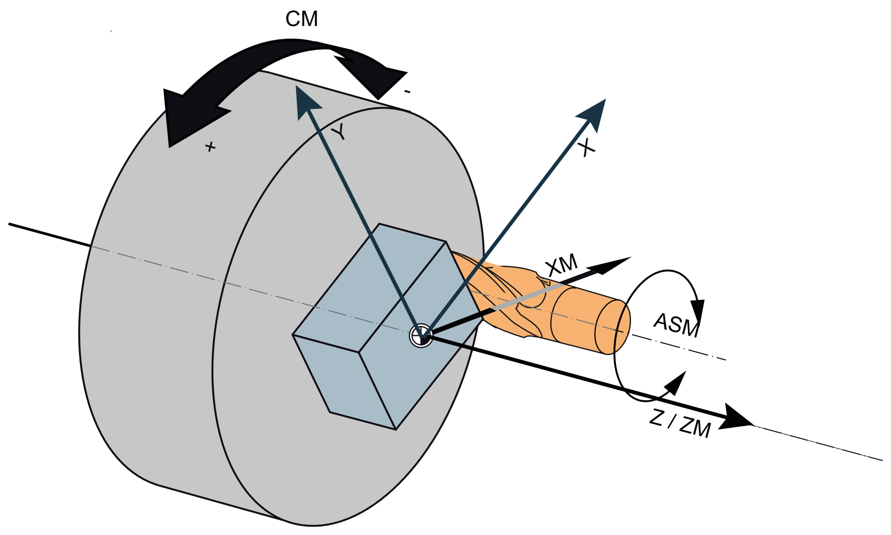
9.1.4 Face end transformation TRANSMIT

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 via two kinematic chains, see Definition of kinematic transformations (Page 354).

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 9-4 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.

Defining the order of the axes

Various settings for the transformation are defined via the system variable $NT_CNTRL (Page 395)[n]. 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 $NT_BASE_TOOL_COMP (Page 399) can be used to set whether an offset is entered for the base tool in the transformation frame ($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 390), 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 about 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 428).

9.1.5 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 via two kinematic chains, see Definition of kinematic transformations (Page 354).

The following graphic shows an example of a cylinder surface transformation:

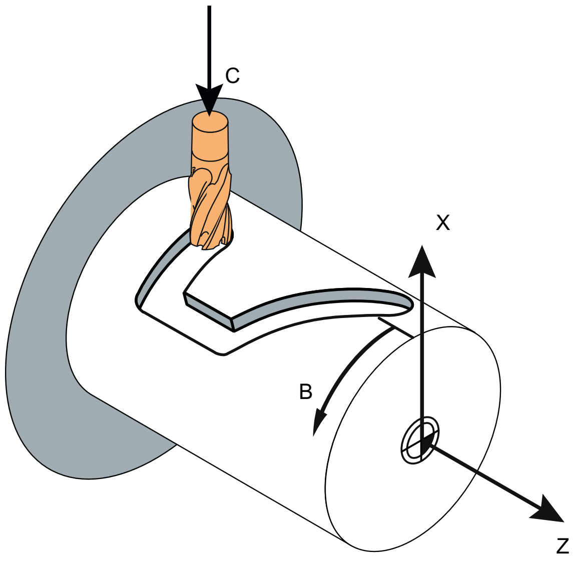


Figure 9-5 Cylinder surface transformation (TRACYL)

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 412).

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.

Defining the order of the axes

Various settings for the transformation are defined via the system variable $NT_CNTRL (Page 394)[n]. 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 9	Bit 10	Description
0	0	TRACYL without slot side offset
0	1	TRACYL with slot side offset

Bit 9	Bit 10	Description
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 390). 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 about 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.

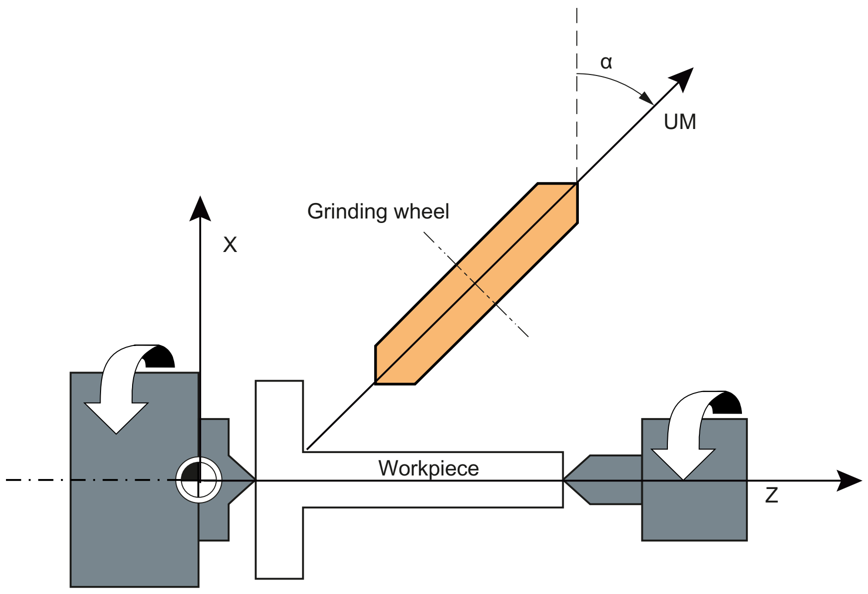
Example program for TRACYL

You will find a simple part program using TRACYL under Part program for TRACYL (Page 431).

9.1.6 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.



X	Geometry axis
Z	Geometry axis
ZM	Machine axis
UM	Machine axis
α	Angle of inclined axis

Figure 9-6 TRAANG_K example

The oblique angle transformation is a special case of orientation transformation.

The transmit transformation is activated via the system variable $NT_TRAFO_TYPE = "TRAANG_K".

Example program for TRAANG

You will find a simple part program using TRAANG under Part program for TRAANG (Page 434).

9.1.7 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 with the same system variables as a dynamic orientation transformation, see Dynamic orientation transformation TRAORI_DYN (Page 356) and Definition of kinematic transformations (Page 354).

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 354).

The static orientation transformation is defined via the following system variables:

- System variables for general transformation types (Page 375)
- Additive system variable for orientation transformation (Page 399)

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".

9.1.8 Concatenating transformations (TRACON_K)

Concatenating transformations

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.

The concatenation can be defined within the transformation definition via a transformation of the type "TRACON_K".

References:

A detailed description of concatenation of transformations without kinematic chains is given in Functional Manual "Extended Functions", Section "Kinematic Transformation, under "Chained transformations".

Structure of a transformation concatenation

Transformation chains are concatenated via the following syntax:

- Definition of the partial transformations 1 - x
- Definition of a transformation of the type "TRACON_K" with $NT_TRAFO_TYPE
- Definition of the elements of the concatenation with $NT_TRACON_CHAIN (Page 411). Via the system variables, the partial transformations are listed in the order in which they are to be carried out.

9.1.9 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.

9.1.10 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 369).

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.

9.1.11 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.

Maintaining 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 kinematic chains. 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 412)(<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 377).

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.

9.1.12 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 353) are defined for describing the machine kinematics.

9.1.13 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.

9.2 Commissioning

9.2.1 General

9.2.1.1 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

9.2.1.2 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 → 1. Element, index 1 → 2. Element, ... n → (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

9.2.2 Machine data

9.2.2.1 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>

9.2.2.2 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>

9.2.2.3 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>

9.2.2.4 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>

9.2.2.5 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>

9.2.2.6 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>

9.2.3 Use of system variables for kinematic transformations

Usage table

The following table shows which $NT data is relevant for the individual types of transformations.

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_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	-

Variable	BTSS (column ind.)	TRAORI_STAT	TRAORI_DYN	TRANS-MIT_K	TRAC-YL_K	TRAANG_K
$NT_CLOSE_CHAIN_T[n]	1226	x	x	x	x	x
$NT_BASE_ORIENT[n,0..2]	1280-1285	x	x	-	-	-
$NT_BASE_ORIENT_NORMAL[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
$NT_BASE_TOOL_COMP	1396	-	-	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).

9.2.4 System variables for general transformation types

9.2.4.1 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 kinematic chain to the **tool**.
$NT_P_CHAIN_LAST_ELEM	Name of the last element of the kinematic chain to the **workpiece**
$NT_T_REF_ELEM	Name of the reference element for tool length

System variable	Meaning
$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 **work-piece**.
$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**
$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() (Page 253).

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.

9.2.4.2 $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"`

9.2.4.3 $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 ten-thousands 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	
	Range of values:	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`

9.2.4.4 $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"`

9.2.4.5 $NT_T_CHAIN_FIRST_ELEM

Function

The kinematics of a transformation is described by up to two kinematic subchains, which begin in the respective root element (Page 229), i.e. the first element of the active kinematic chain. One subchain leads from the root element to the **workpiece** reference point (see Definition of kinematic transformations (Page 354)). 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 ($NK_NAME (Page 237)) 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:	""
	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)

`<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] = "Base-offset_2"`	`; 1st transformation,` `; first element for the tool reference point`

9.2.4.6 $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 (Page 229), 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 354)). 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 ($NK_NAME (Page 237)) 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:	""
	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 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`

9.2.4.7 $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 (Page 229), 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 354)). The other subchain, which is described here, leads to the **tool** reference point.

The name ($NT_NAME (Page 377)) 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:	""
	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 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`

9.2.4.8 $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 (Page 229), 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 354)). The other subchain, which is described here, leads to the **workpiece** reference point.

The name ($NT_NAME (Page 377)) 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:	""
	Value range:	Element names of the currently active kinematic chain
`<n>`:	System variable or transformation index	
	Data type:	INT
	Range of val-ues:	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] = "Ta-bleOffset"`	`; 2nd transformation,`
	`; element for the workpiece reference point`

9.2.4.9 $NT_T_REF_ELEM

Function

The name ($NT_NAME (Page 377)) 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`

9.2.4.10 $NT_GEO_AX_NAME

Function

A maximum of three names ($NT_NAME (Page 377)) of the elements of the kinematic chain that is currently in effect, which define the geometry axes (linear axes) of the transformation type (Page 354) 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)
```

9.2.4.11 $NT_ROT_AX_NAME

Function

A maximum of three names ($NT_NAME (Page 377)) of the elements of the kinematic chain that is currently in effect, which define the rotary axes of the transformation type (Page 354) 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
```

9.2.4.12 $NT_CLOSE_CHAIN_P

Function

The name ($NT_NAME (Page 377)) 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_Off-set_3"`	`; 1st transformation,`
	`; element for closing the subchain`
	`; to the **workpiece** reference point`

See also

$NT_CNTRL (Page 395)

9.2.4.13 $NT_CLOSE_CHAIN_T

Function

The name ($NT_NAME (Page 377)) 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 395)), 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`

9.2.4.14 $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.

- If `$NT_ROT_OFFSET_FROM_FRAME=2` , the following applies for TRACYL or TRANSMIT: Axial offset of the rotary axis is taken into account up to the SZS. SZS is the settable zero system. The SZS frames contain transformed rotations about the rotary axis.
- If `$NT_ROT_OFFSET_FROM_FRAME=2` , the following applies for TRAORI or TRAORY_DYN:
  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. The offset is only applied to the offset of the orientation axes if this $NT_ROT_OFFSET_FROM_FRAME has the value TRUE and the work offsets remain active when the transformation is switched on, i.e. MD10602 $MN_FRAME_GEOAX_CHANGE_MODE > 0.

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:	Char
	Default value:	"0"

`<n>`:	System variable or transformation index	
	Data type:	DINT
	Range of values:	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.`

9.2.4.15 $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.`

9.2.4.16 $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)
```

9.2.4.17 $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`	; -

9.2.4.18 $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 388) 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 389) 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.		
16 ... 18		Binary coding via bits 12, 13 and 14 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.		

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
	Range of values:	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,`
	`; control signals: 'Bdcbbbaaa0'`

9.2.4.19 $NT_ROT_AX_CNT

Function

The system variable `$NT_ROT_AX_CNT` provides the number of the relevant rotary axes in the subchain 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 subchain or tool chain	
	Data type:	INT
	Default value:	-1; content is not evaluated
`<n>`:	System variable or transformation index	
	Data type:	INT
	Range of values:	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 subchain • 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] = "NumberRotAx-es"`	`; provides the number of rotary axes for the 1st kinematic chain that is defined as a subchain`

9.2.4.20 $NT_BASE_TOOL_COMP

Function

The correction of the tool offset must be entered in the system variable `$NT_BASE_TOOL_COMP` with the aid of a frame.

Syntax

```
$NT_BASE_TOOL_COMP[<n>] = "<>"
```

Meaning

`$NT_BASE_TOOL_COMP`:		
	Data type:	
	Default value:	
`<n>`:	System variable or transformation index	
	Data type:	INT
	Range of values:	1, 2, ... (MD18866 $MN_MM_NUM_KIN_TRAFOS - 1)
`<>`:		
	Data type:	

Example

The 2nd transformation is assigned the name "B axis":

```
Program code                         Comment
N100 $NT_NAME[1] = "Processing"      ; 2nd transformation,
                                     ; name = "processing"
```

9.2.5 Additive system variable for orientation transformation

9.2.5.1 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

System variable	Meaning
$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.
$NT_CNTRL	Bit-coded 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() (Page 253).

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.

9.2.5.2 $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)

9.2.5.3 $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)
```

9.2.5.4 $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 (Page 247), 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)
	Value range:	- max. REAL value ≤ x ≤ + 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 orientation axes	
	Data type:	INT
	Range of values:	0: 1st Orientation axes 1: 2nd Orientation axes 2: 3rd Orientation axes
`<OriAxPos>`:	Position value	
	Data type:	REAL

Example

The positions of the orientation axes, which result from the parameterized constant rotation, are: (0.0°, 0.0°, 45.0°)

```
Program code                            Comment
                                        ; 1st transformation,
                                        ; positions of the orientation axes:
N100 $NT_ROT_AX_POS[1,0] =  0.0         ; 1st orientation axis
N110 $NT_ROT_AX_POS[1,1] =  0.0         ; 2nd orientation axis
N120 $NT_ROT_AX_POS[1,2] = 45.0         ; 3rd orientation axis
```

9.2.5.5 $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.

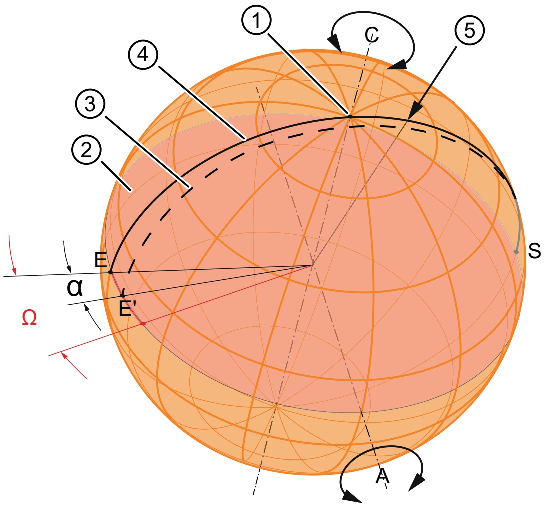
End angle

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`

9.2.5.6 $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>] = "<ElementName>"
```

Meaning

`$NT_POLE_TOL`:	End angle tolerance for pole interpolation	
	Data type:	REAL
	Default value:	1.0
	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)
`<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`

9.2.5.7 $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 400) and $NT_BASE_ORIENT_NORMAL (Page 401) 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

9.2.5.8 $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 413). 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
	Range of values:	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
`Corr_1 = $NT_CORR_ELEM_T(1,2)`	Refers to a correction element in the tool chain of transformation "1" and position "2".

9.2.5.9 $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 413). 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
	Range of values:	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
`Corr_1 = $NT_CORR_ELEM_P(1,2)`	Refers to a correction element in the workpiece chain of transformation "1" and position "2".

9.2.6 Additive system variable for face transformation (TRANSMIT)

9.2.6.1 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() (Page 253).

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.

9.2.6.2 $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

9.2.7 Additive system variables for transformation chains (TRACON_K)

9.2.7.1 $NT_TRACON_CHAIN

Function

The name of a partial transformation of a transformation chain must be entered in the system variable.

Syntax

```
$NT_TRACON_CHAIN[<n>,<m>] = <Namce>
```

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)
`<m>`:	Maximum number of transformations of a transformation chain	
	Data type:	INT
	Range of values:	0 ... MD_MAX_CONCATENATED_ TRAFOS
`<name>`:	Name of the partial transformation	
	Data type:	STRING

Example

Names of partial transformations 1 and 2:

Program code	Comment
`N2502 $NT_TRACON_CHAIN[NT_CNTR, 0] = "5-axis transformation C-A"`	`Partial transformation 1`
`N2503 $NT_TRACON_CHAIN[NT_CNTR, 1] = "inclined axis"`	`Partial transformation 2`

9.3 Programming

9.3.1 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 377)	
	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.`

9.3.2 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.

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 (unit 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 (unit 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 (unit 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 unit 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 (unit to tens position):		
	Unit 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 (unit to thousands position):		
	Unit 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 374) or • SD41611 $SN_CORR_TRAFO_DIR_MAX (Page 374)	
		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 374)

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 measures lever arm lengths and axis directions on machines with orientation transformation and writes these values into 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.

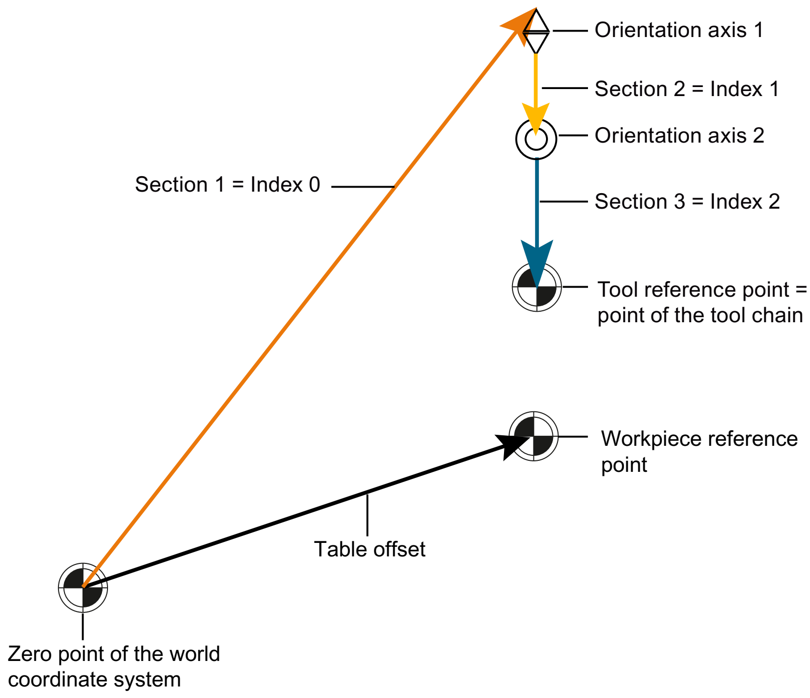

Orientation axis 1
Section 2 = Index 1
Orientation axis 2
Section 1 = Index 0
Section 3 = Index 2
Tool reference point = end point of the tool chain
Workpiece reference point
Table offset
Zero point of the world coordinate system

Figure 9-7 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.

9.4 Examples

9.4.1 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".

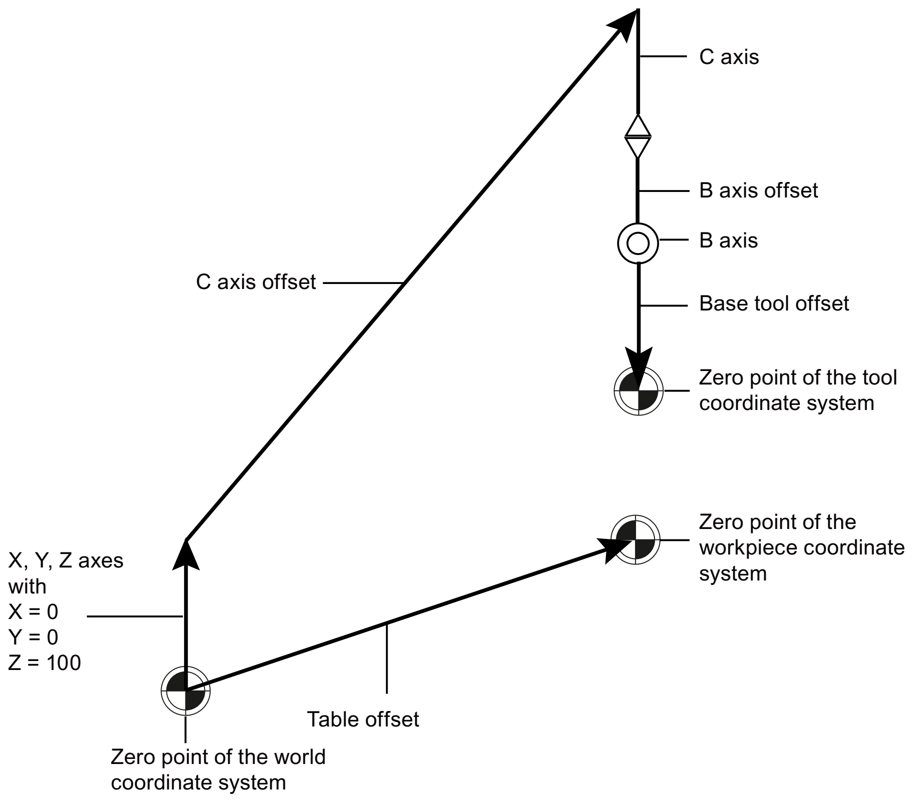

C axis
B axis offset
B axis
Base tool offset
Zero point of the tool coordinate system
C axis offset
Zero point of the workpiece coordinate system
X, Y, Z axes with X = 0 Y = 0 Z = 100
Table offset
Zero point of the world coordinate system

Figure 9-8 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

9.4.2 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
```

9.4.3 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
```

9.4.4 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
```

9.4.5 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 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,1]       = "Z-Axis"
N2300   $NT_GEO_AX_NAME[_TRA_CNTR,2]       = "Y-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
```

M3: Coupled axes

10

10.1 Coupled motion

10.1.1 Brief description

10.1.1.1 Function

The "coupled motion" function enables the definition of simple axis links between a master axis and a slave axis, taking into consideration a coupling factor.

Coupled motion has the following features:

- Any axis of the NC can be defined as a master axis.
- Any axis of the NC can be defined as a coupled axis with a specific coupling factor.
- The master axis and coupled motion axis or axes together form a coupled axis grouping.
- Any number of coupled motion axes can be assigned to a master axis.
- A total of 2 leading axes may be assigned to each coupled motion axis.
- A coupled motion axis can be the master axis of a further coupled axis grouping.
- Traversing movements of the master axis are executed in synchronism on all slave axes based on the coupling factor.
- Coupled motion axes can be moved independently of the master axis while the coupling is active (overlaid movements).
- The master and coupled motion axes of a coupled axis grouping are defined, and the coupling switch on/switch off, by programming instructions in the part program or by synchronized action.
- Coupled motion is also possible in the following manual modes: JOG, JOG REF, JOG INC, etc.

10.1.1.2 Preconditions

"Coupled motion" function

The "Coupled motion" function is a fixed component of the NC software.

Generic coupling

The coupled motion functionality is also available in the generic coupling.

However, for **basic operation** of generic coupling, the following restrictions apply:

- The maximum number of coupled motion groupings is limited to 4.
- Only 1 leading axes may be assigned to each coupled motion axis.
- Cascading is not possible.

Note

These restrictions do not apply when NC software is supplied with the relevant options of generic coupling (refer to " Preconditions (Page 510) " in the "Brief description" of generic coupling).

10.1.2 General functionality

The "Coupled motion" function allows the definition of simple axis couplings. Coupling is performed from one leading axis to one or more following axes, the so-called coupled motion axes. A separate coupling factor can be specified for each coupled motion axis.

Coupled axis grouping

The leading axis and all the coupled motion axes assigned to it together form a coupled axis grouping. If the leading axis is traversed, all coupled motion axes traverse in accordance with their coupling factors.

A coupled axis grouping can consist of any combination of linear and rotary axes.

Leading axes

Any axis of the NC, including simulated axes, can be used as leading axis.

Coupled axes

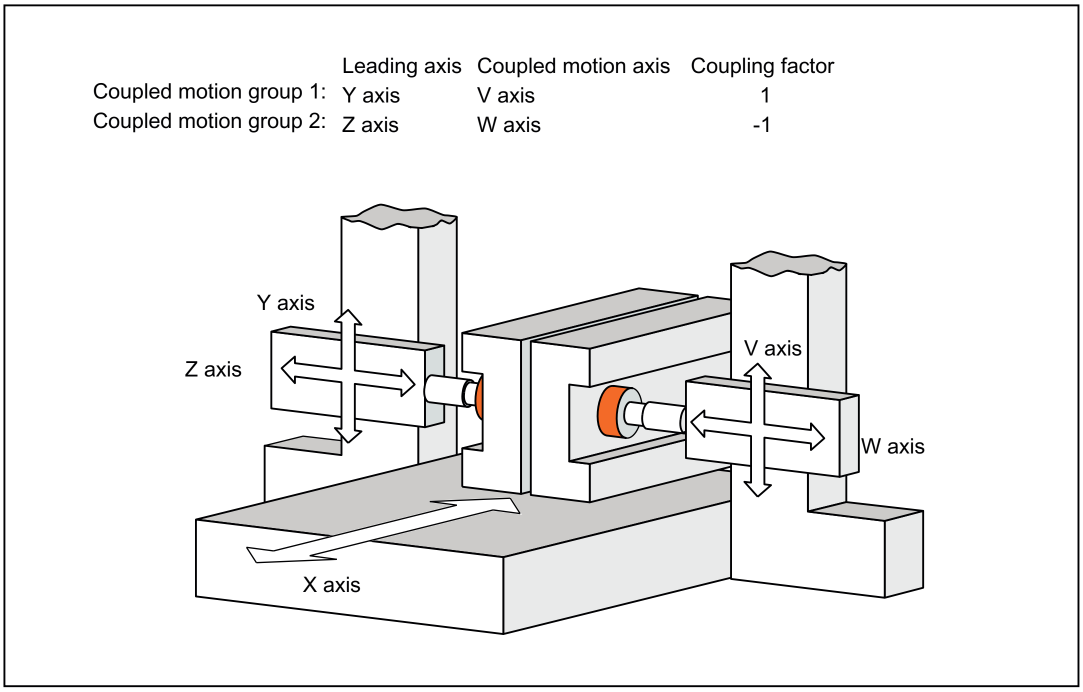
Any axis of the NC can be used as coupled motion axis.

Coupling factor

The ratio in which the coupled motion axis moves in relation to the leading axis is specified via the coupling factor.

Coupling factor K = motion of the coupled motion axis / motion of the leading axis

Negative coupling factors (motion of the coupled motion axis in the opposite direction) are also permitted.



	Leading axis	Coupled motion axis	Coupling factor
Coupled motion group 1:	Y axis	V axis	1
Coupled motion group 2:	Z axis	W axis	-1

Figure 10-1 Application example: Two-sided machining

Multiple couplings

Up to 2 leading axes can be assigned to one coupled motion axis. The traversing movement of the coupled motion axis then results from the sum of the traversing movements of the leading axes.

Dependent coupled motion axis

A coupled motion axis is a "dependent coupled motion axis" when it traverses as a result of a leading axis movement.

Independent coupled motion axis

A coupled motion axis is an "independent coupled motion axis" when it traverses as a result of a direct traverse instruction. The traversing movement resulting from the coupled motion axis is then the sum total of the traversing movements as a "dependent" and an "independent" coupled motion axis.

Coupled motion axis as leading axis

A coupled motion axis can at the same time be the leading axis of a further coupled axis grouping.

Coordinate system

Coupled axis motion is always executed in the base coordinate system (BCS).

Activating/deactivating

Coupled motion can be activated/deactivated via the part programs and synchronous actions. In this context please ensure activating/deactivating is undertaken with the same programming:

- Activate: Part program → Deactivate: Part program
- Activate: Synchronous action → Deactivate: Synchronized action

Synchronization on-the-fly

If switch on is performed while the leading axis is in motion, the coupled motion axis is first accelerated to the velocity corresponding to the coupling. The position of the leading axis at the time the velocities of the leading and coupled motion axes are synchronized then serves as the start position for further coupled motion.

Operating modes

Coupled motion is effective in the AUTOMATIC, MDA and JOG modes.

Reference point approach

The following applies for referencing of axes of a coupled axis grouping:

- Leading axes
  When referencing the leading axis of a coupled axis grouping, the coupling to all coupled motion axes is retained. The coupled motion axes move in synchronism with the leading axis, as a function of their coupling factors.
- Coupled motion axis: JOG/REF mode
  When referencing a coupled motion axis of a coupled axis grouping, the coupling to the leading axis is cancelled. If the coupling is canceled, the following alarm is displayed:
  Alarm 16772 "Channel *Channel No.* block *Block No.* Axis *Axis No.* is following axis, coupling is opened."
  The coupling is not activated again until JOG/REF mode is cancelled.
  The display of this alarm can be suppressed using the following machine data:
  MD11410 $MN_SUPPRESS_ALARM_MASK, Bit 29 = 1 (mask supporting special alarm generation)

  CAUTION

  No coupling

  When the coupled motion axis is referenced, the coupling to the leading axis is cancelled. If referencing is executed immediately with the leading axis, i.e. without changing JOG/REF mode, the coupled motion axis does not traverse with the leading axis.

- Coupled motion axis: Part program command `G74`
  It is not possible to reference a coupled motion axis of a coupled axis grouping using the `G74` programming instruction.

Distance-to-go: Coupled motion axis

The distance-to-go of a coupled motion axis refers to the total residual distance to be traversed from dependent and independent traversing.

Delete distance-to-go: Coupled motion axis

Delete distance-to-go for a coupled motion axis only results in aborting of the independent traversing movement of the leading axis.

Response to NC Start

The behavior of the coupled-axis groupings during NC Start depends on the setting in the machine data:

MD20112 $MC_START_MODE_MASK (definition of initial control settings for NC-START)

Bit	Value	Meaning
8	0	Coupled-axis groupings are maintained in NC Start.
	1	Coupled-axis groupings are phased out in NC Start.

Response to RESET/part program end

The behavior of the coupled-axis groupings during RESET/ part program end depends on the setting in the machine data:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET / TP end)

Bit	Value	Meaning
8	0	Coupled-axis groupings are invalidated on RESET / part program end.
	1	Coupled-axis groupings remain active even beyond RESET/part program end.

Note

If with NC RESET or end of part program in a channel, the leading axis is not stopped as well (cross-channel coupling, command axis, PLC axis, etc.), the requested RESET cannot be completed.

Because of traversing of the leading axis, the coupled-motion axis is still active for the channel in which the RESET is requested. With suitable actions (NC RESET in the channel of the leading axis, stopping of the command or PLC axis), the leading axis must also be stopped in parallel to the coupled-motion axis.

10.1.3 Programming

10.1.3.1 Definition and switch on of a coupled axis grouping (TRAILON)

Definition and switch on of a coupled axis grouping take place simultaneously with the `TRAILON` part program command.

Programming

Syntax:		`TRAILON(<coupled motion axis>, <leading axis>, [<coupling factor>])`
Effectiveness:		modal
Parameters:		
	Coupled motion axis:	Type: AXIS Range of values:All defined axis and spindle names in the channel
	Leading axis:	Type: AXIS Range of values:All defined axis and spindle names in the channel
	Coupling factor:	The ratio of the traversing movement of the coupled motion axis to the leading axis is specified via the optional coupling factor: Coupling factor = Path of the coupled-motion axis/path of the leading axis A negative coupling factor results in motion in opposite directions for the leading and coupled motion axis. Type: REAL Range of values:± (2,2 * 10-308 … 1,8 * 10+308) Default value: +1.0

Example:

Program code	Comment
`TRAILON(V,Y,2)`	`; Definition and switch on of the coupling of the coupled-motion axis V with leading axis Y. The coupling factor is 2.`

10.1.3.2 Switch off (TRAILOF)

Switch off of the coupling of a coupled-motion axis with a leading axis takes place through the `TRAILOF` part program command.

Programming

Syntax:	`TRAILON`(<coupled motion axis>, <leading axis>) `TRAILOF`(<coupled-motion axis>)
Effectiveness:	modal

Parameters:

Coupled motion axis:	Type: AXIS	
	Range of values:	All defined axis and spindle names in the channel
Leading axis:	Type: AXIS	
	Range of values:	All defined axis and spindle names in the channel

Example:

Program code	Comment

`TRAILOF(V,Y)`	`; Deactivation of the coupling of the coupled-motion`
	 `; axis`
	 `V to leading axis Y.`

10.1.4 Effectiveness of PLC interface signals

Independent coupled motion axis

All the associated channel and axis specific interface signals of the coupled-motion axis are effective for the independent motion of a coupled-motion axis, e.g.:

- DB21, ... DBX0.3 (Activate DRF)
- DB31, ... DBX0.0 - 0.7 (feedrate override)
- DB31, ... DBX1.3 (axis blocking)
- DB31, ... DBX2.1 (control system enable)
- DB31, ... DBX4.0 - 4.2 (activate handwheel)
- DB31, ... DBX4.3 (feed stop)
- …

This allows the speed to be changed for the independent motion of a coupled motion axis using a feedrate override or a DRF offset to be defined using the handwheel in AUTOMATIC and MDA modes.

Dependent coupled motion axis

With respect to the motion of a coupled motion axis, which is dependent on the leading axis, only the coupled-motion axis interface signals that effect termination of the motion (e.g. axis-specific feed stop, axis inhibit, control system enable, etc.) are effective.

Leading axis

When a coupled axis grouping is active, the interface signals (IS) of the leading axis are applied to the appropriate coupled motion axis via the axis coupling, i.e.

- A position offset or feed control action of the leading axis is applied via the coupling factor to effect an appropriate position offset or feed control action in the coupled motion axis.
- Shutdown of the leading axis as the result of an interface signal (e.g. axis-specific feed stop, axis inhibit, servo enable, etc.) causes the corresponding coupled motion axis to shut down.

Position measuring system 1/2 (DB31, ... DBX1.5/1.6))

Switch-over of the position measuring system for the leading and coupled motion axes is not inhibited for an active coupled axis grouping. The coupling is not canceled.

Recommendation: Switch-over when the coupling is deactivated.

Tracking (DB31, ... DBX1.4)

Activation of tracking for an axis is done via the PLC program by setting the following NC/PLC interface signals:

DB31, ... DBX2.1 = 0 (control system enable)

DB31, ... DBX1.4 == 1 (tracking mode)

When activating tracking mode for a coupled axis grouping, the specified NC/PLC interface signals must be set simultaneously for all axes (master and slave axes) of the coupled axis group.

If tracking mode is activated for the master axis only, a permanent offset results within the coupling.

Whether and which axis is a leading or a following axis can be seen from the following NC/PLC interface signals and system variables:

DB31, ... DBX99.0 (leading axis/spindel active)

DB31, ... DBX99.1 (following axis/spindel active)

$AA_COUP_ACT[axis name] (see: Status of coupling)

10.1.5 Status of coupling

The coupling status of an axis can be determined using the following system variables:

$AA_COUP_ACT [axis name]

Value	Meaning
0	No coupling active
1, 2, 3	Tangential tracking
4	Synchronous spindle coupling
8	Coupled motion active

Value	Meaning
16	Master value coupling
32	Following axis of electronic gearbox

Note

Only one coupling mode may be active at any given time.

10.1.6 Dynamics limit

The dynamics limit is dependent on the type of activation of the coupled axis grouping:

- Activation in part program
  If activation is performed in the part program and all leading axes are active as program axes in the activating channel, the dynamic response of all coupled-motion axes is taken into account during traversing of the leading axes to avoid overloading the coupled-motion axes.
  If activation in part program takes place with leading axes that are not active as program axes in the activating channel ($AA_TYP ≠ 1), then the dynamics of the coupled motion axes is not considered while traversing the leading axes. This can result in an overload for coupled motion axes with a dynamic response which is less than that required for the coupling.
- Activation in synchronized action
  If activation is performed in a synchronous action, dynamics of the coupled motion axes is not taken into account during traversing of the leading axis. This can result in an overload for coupled motion axes with a dynamic response which is less than that required for the coupling.

⚠ CAUTION

Axis overload

If a coupled motion grouping

- in synchronized actions
- is activated in the part program with leading axes, that are not program axes in the channel of the coupled motion axes,

it is the special responsibility of the user/machine manufacturer to provide suitable measures to ensure that an overload of the coupled motion axes does not occur through traversing of the leading axis.

10.1.7 Supplementary conditions

Control system dynamics

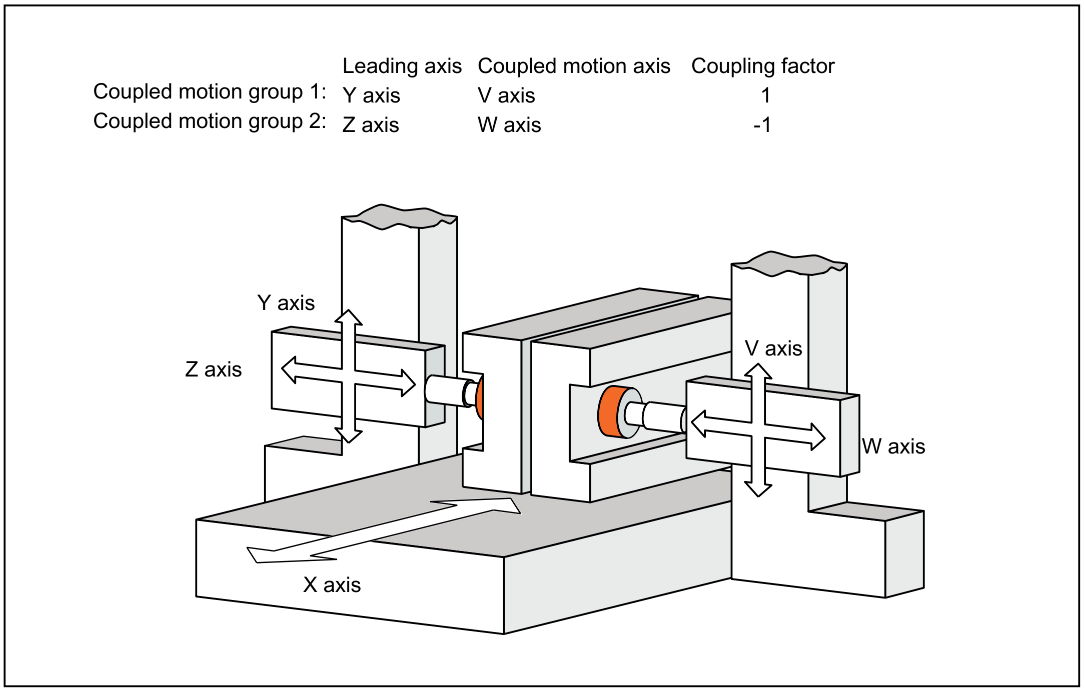
It is recommended to align the position control parameters of the leading axis and the coupled motion axis within a coupled axis group.

Note

Alignment of the position control parameters of the leading axis and the coupled motion axis can be performed via a parameter set changeover.

10.1.8 Examples

Application Example: Two-sided machining



Example 1

Example of an NC part program for the axis constellation shown in Fig.:

Program code	Comment
`TRAILON(V,Y,1)`	`; Activation of 1st coupled axis group`
`TRAILON(W,Z,-1)`	`; Activation of 2nd coupled axis group`

Program code	Comment
G0 Z10	; Infeed Z and W axes in opposite axial directions
G0 Y20	; Infeed of Y and V axes in same axis direction
G1 Y22 V25	; Superimpose dependent and independent movement of coupled motion axis "V"
TRAILOF(V,Y)	; Deactivate 1st coupled axis group
TRAILOF(W,Z)	; Deactivate 2nd coupled axis group

Example 2

The dependent and independent movement components of a coupled motion axis are added together for the coupled motion. The dependent component can be regarded as a co-ordinate offset with reference to the coupled motion axis.

Program code	Comment
N01 G90 G0 X100 U100	;
N02 TRAILON(U,X,1)	; Activation of coupled axis group
N03 G1 F2000 X200	; Dependent movement of U, Upos=200, UTrail=100
N04 U201	; Independent movement, Upos=U201+UTrail=301
N05 X250	; Dependent movement of U, UTrail=UTrail(100)+50=150, Upos=351
N06 G91 U100	; Independent movement of Upos(351)+U100=451
N07 G90 X0	; Dependent movement of U, Upos=Upos(451)-UTrail(250)=201
N10 TRAILOF(U,X)	

10.2 Curve tables - 840D sl only

10.2.1 Product brief

10.2.1.1 Function

The "curve tables" function can be used to define the complex sequence of motions of an axis in a curve table.

Any axis can be defined as a leading axis and a following axis can be traversed by taking a curve table into account.

The command variable in these motion sequences is an abstract master value, which is generated by the control or derived from an external variable (e.g. simulated position of an axis).

Creating curve tables is performed via part program sequences.

The curve tables in the static NC memory remain valid after the part program has been been closed or after POWER DOWN.

Curve tables can be saved in dynamic NC memory for faster access. Please note that tables need to be reloaded after run-up.

Axis groupings with curve tables must be reactivated independently of the storage location of the curve table after POWER ON.

Linear curve table segments are stored in separate areas to save memory space.

10.2.1.2 Preconditions

Memory configuration

Static NC memory

Memory space for curve tables in static NC memory is defined by machine data:

MD18400 $MN_MM_NUM_CURVE_TABS (number of curve tables)

MD18402 $MN_MM_NUM_CURVE_SEGMENTS (number of curve segments)

MD18403 $MN_MM_NUM_CURVE_SEG_LIN (number of linear curve segments)

MD18404 $MN_MM_NUM_CURVE_POLYNOMS (number of curve table polynomials)

Dynamic NC memory

Memory space for curve tables in dynamic NC memory is defined by machine data:

MD18406 $MN_MM_NUM_CURVE_TABS_DRAM (number of curve tables)

MD18408 $MN_MM_NUM_CURVE_SEGMENTS_DRAM (number of curve segments)

MD18409 $MN_MM_NUM_CURVE_SEG_LIN_DRAM (number of linear curve segments)

MD18410 $MN_MM_NUM_CURVE_POLYNOMS_DRAM (number of curve table polynomials)

10.2.2 General functionality

Curve table

A functional relation between a command variable "master value" and an abstract following value is described in the curve table.

A following variable can be assigned uniquely to each master value within a defined master value range.

Curve segment

The functional relation can be subdivided into separate sections of the master value axes, called curve segments.

Within a curve segment, the relation between the master value and following value is generally described by a polynomial up to the third order. Polynomials up to the 5th degree are also permissible.

Reference:
Programming Manual, Production Planning

Curve segments are used if:

- Polynomes or circles are programmed
- Spline is active
- Compressor is active
- Polynomials or circles are generated internally (chamfer/rounding, approximate positioning with `G643`, WRK etc.)

Tool radius compensation

Curve tables are available in which it is possible to specify the tool radius compensation in the table definition even if polynomial blocks or blocks with no motion for an axis, or jumps for the following axis, occur in the curve table (`G41/G42/G40` in the table definition).

The equidistant curve (tool center point path of tool radius compensation) of a curve consisting of polynomials can no longer be displayed exactly using polynomials. The associated curve tables must be approximated stepwise, using polynomials in this case. This means that the number of segments in the curve table no longer matches the number of programmed segments. The number of segments required for the curve table is defined by the bend of the curve. The larger the curvature for the programmed curve, the more segments are required for the curve table.

On account of tool radius compensation for curve tables, more memory may be required. Selection option of the memory type should not produce shortage of static NC memory.

Selection of memory type

While defining a curve table, it can be defined whether the curve table is created in the static or dynamic NC memory.

Note

Table definitions in the static NC memory are available even after control system run-up. Curve tables of the dynamic NC memory must be redefined after every control system run-up.

10.2.3 Memory organization

Memory configuration

The storage place available for the curve table in the static and dynamic NC memory is defined during memory configuration (see Section "Memory configuration (Page 451)").

Memory optimization

In a curve table with linear segments, the linear segments can be stored efficiently in the memory only if the two following machine data items are > 0:

MD18403 $MC_MM_NUM_CURVE_SEG_LIN (number of linear curve segments in the static NC memory)

MD18409 $MC_MM_NUM_CURVE_SEG_LIN_DRAM (number of linear curve segments in the dynamic NC memory)

If no memory areas are created with this machine data, then the linear segments are stored as polynomial segments.

Alarm in case of insufficient memory

If memory has been configured for tables with linear and polynomial segments via machine data and memory for linear segments runs out when generating a linear table, the the memory for polynomial segments is used for the linear segments (if available). In this case, memory is "wasted", as a polynomial segment requires an unnecessary amount of memory to store a linear segment. This circumstance is conveyed through an alarm, which also discloses the number of unnecessarily used polynomial segments. The alarm only displays a**warning** and does not result in the interruption of the program or the generation of the curve table.

If a curve table consists of linear segments and polynomials of a high degree, a memory area for linear segments and a memory area for polynomial segments is required for the storage of the curve table. An alarm is output if insufficient memory is available in the relevant areas. The alarm parameters can be used to detect the resources that are insufficient.

Insufficient memory

If a curve table cannot be created, because sufficient memory is not available, then the newly created table is deleted immediately after the alarm.

If insufficient is available, then one or more table(s) that is/are no longer required can be deleted with `CTABDEL` or, alternatively, memory can be reconfigured via machine data.

Temporary curve table

When a curve table is created, a temporary curve table is set up first in memory, which is then extended block by block. Finally (`CTABEND`), the table is checked for consistency. The temporary table is converted to a table that can be used in a coupling only if it is found to be consistent.

Same table number

A new curve table may have the same number as an existing table. The new curve table then overwrites the existing table with the same number. This is done only if the new curve table does not contain any errors. If an error is detected in the new table, the old table is not overwritten.

If the user wishes to have the option of overwriting an existing curve table without deleting it first, then he will need to dimension the table memory so that there is always enough extra memory to accommodate the table to be overwritten.

Overwriting curve tables

Curve tables that are not active in a master value coupling and are locked with `CTABLOCK`() may be overwritten.

Deleting curve tables

Curve tables that are not active in a master value coupling and are locked with `CTABLOCK`() may be overwritten.

10.2.4 Commissioning

10.2.4.1 Memory configuration

A defined storage space is available for the curve tables in the static and dynamic NC memory, which is defined through the following machine data:

Static NC memory	
MD18400 $MN_MM_NUM_CURVE_TABS	Defines the **number of curve tables** that can be stored in the static NC memory.
MD18402 $MN_MM_NUM_CURVE_SEGMENTS	Defines the **number of curve table segments** that can be stored in the static NC memory.
MD18403 $MN_MM_NUM_CURVE_SEG_LIN	Defines the **maximum number of linear segments** in the static NC memory.
MD18404 $MN_MM_NUM_CURVE_POLYNOMS	Defines the **number of curve table polynomes** that can be stored in the static NC memory.

Dynamic NC memory	
MD18406 $MN_MM_NUM_CURVE_TABS_DRAM	Defines the **number of curve tables** that can be stored in the dynamic NC memory.
MD18408 $MN_MM_NUM_CURVE_SEGNENTS_DRAM	Defines the **number of curve table segments** that can be stored in the dynamic NC memory.
MD18409 $MN_MM_NUM_CURVE_SEG_LIN_DRAM	Defines the **maximum number of linear segments** in the dynamic NC memory.
MD18410 $MN_MM_NUM_CURVE_POLYNOMS_DRAM	Defines the **number of curve table polynomes** that can be stored in the dynamic NC memory.

Note

A curve table with linear segments can be stored efficiently in the memory only if:

MD18403 > 0 or MD18409 > 0

If no memory areas are created with this machine data, then the linear segments are stored as polynomial segments.

10.2.4.2 Tool radius compensation

MD20900

Tool radius compensation can produce segments for which the following axis or leading axis have no movement. A missing movement of the following axis does not normally represent any problem. As against this, a missing movement of the leading axis requests a specification as to how such discontinuities are to be handled, i.e., whether or not a curve table should be generated in these cases. This specification is done in the machine data settings:

MD20900 $MC_CTAB_ENABLE_NO_LEADMOTION (curve tables with discontinuity of the following axis)

Value	Description
0	No curve tables that contain a discontinuity in the following axis are produced. Alarm 10949 is output and program processing is aborted.
1	Curve tables with a discontinuity in the following axis can be generated. If a segment contains a discontinuity in the following axis, Alarm 10955 is output but program processing is continued.
2	Curve tables with a discontinuity in the following axis can be created without an alarm being output.

Note

In the case of a curve table that contains segments without leading axis movement (this means that the following axis jumps at this point), the following axis can only make a jump within its dynamic limits (max. velocity and max. acceleration). This means that there is always a deviation from the programmed curve.

10.2.4.3 Specification of memory type

MD20905

If there is no memory specification while defining or deleting a curve table, the memory type can be determined through the following machine data:

MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE (default memory type for curve tables)

Value	Description
0	Curve tabels are normally created in the static NC memory.
1	Curve tabels are normally created in the dynamic NC memory.

10.2.5 Programming

Definition

The following modal language commands work with curve tables: (The parameters are explained at the end of the list of functions.)

- Beginning of definition of a curve table:
  `CTABDEF`(following axis, leading axis, n, applim, memType)
- End of definition of a curve table:
  `CTABEND`()
- Deleting curve table(s):
  `CTABDEL`(n)
  ; curve table n
  `CTABDEL`(n, m)
  ; [n < m], more than one in the range of numbers
  ; it is deleted in static "SRAM" and in dynamic "DRAM" of NC memory.
- `CTABDEL`(n, m, memType)
  ; Delete with memory specification:
  Those curve tables with the numbers in the range, which are in the specified memory type, will be deleted. All other curve tables are retained.
  Delete all tables in a particular memory type:
  `CTABDEL`(, , "DRAM")
  `CTABDEL`(, , "SRAM")
  `CTABDEL`()
  ; all in DRAM or
  ; all in SRAM:
  ; all, irrespective of memory type
- Read the following value for a master value
  `CTAB`(master value, n, degrees, [following axis, lead axis])
- Read the master value for a following value
  `CTABINV`(following value, approx. master value, n, degrees, [following axis, leading axis])

Access to curve table segments

- Read start value (following axis value) of a table segment
  `CTABSSV`(leading value, n, degrees, [following axis, leading axis])
- Read end value (following axis value) of a table segment
  `CTABSEV`(master value, n, degrees, [following axis, master axis])

Note

If curve table functions such as `CTAB`(), `CTABINV`(), `CTABSSV`() etc., in **synchronous actions** are used, only **main traverse variable**, e.g. $AC_PARAM[...] or $R[...] is permissible for the return value and the argument "degrees" of the function.

Example:

ID=1 WHEN TRUE DO $R1 = `CTABSSV`(10, 1, $R2)

or

ID=1 WHEN TRUE DO $AC_PARAM[1] = `CTABSSV`(10, 1, $AC_PARAM[2])

Enable/cancel blocking

The following functions can be used to enable or cancel deletion and overwrite blocks for parts. programs.

- **Enable** deletion and overwrite **block**.
  General form: `CTABLOCK`(n, m, memType)
- **Cancel** deletion and overwrite **block**.
  `CTABUNLOCK` releases the tables locked with `CTABLOCK`. Tables involved in an active coupling remain locked, i.e. they cannot be deleted. However, the `CTABLOCK` command is cancelled, i.e. the table can be deleted as soon as the coupling is deactivated. It is not necessary to call `CTABUNLOCK` again.
  General form: `CTABUNLOCK`(n, m, memType)

Applications of the forms:

Curve table with number n

`CTABLOCK`(n)

Curve tables in the number range n to m.

`CTABLOCK`(n, m)

All curve tables, irrespective of memory type

`CTABLOCK`()

All curve tables in the specified memory type

`CTABLOCK`(, , memType)

Curve table with number n

`CTABUNLOCK`(n)

Curve tables in the number range n to m.

`CTABUNLOCK`(n, m)

All curve tables, irrespective of memory type

`CTABUNLOCK()`

All curve tables in the specified memory type

`CTABUNLOCK(, , memType)`

Other commands for calculating and differentiating between curve tables for applications for diagnosing and optimizing the use of resources:

- Number of **defined** tables **irrespective** of memory type
  `CTABNO()`
- Number of **defined** tables in SRAM or DRAM of NC memory
  `CTABNOMEM(memType)`
- Number of **possible** curve tables in memory memType.
  `CTABFNO(memType)`
- Table number of **nth** curve table.
  General form: `CTABID(n, memType)`
  Generates the table number of the nth curve table with memory type memType. `CTABID(1, memType)` is used to read out the highest curve number (105) of the memory type specified.
  `CTABID(n)`
  Generates the table number of the nth curve table in the memory specified using the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE (default memory type for curve tables)
  `CTABID(p)`
  Generates the ID (table number) of the curve table entered in the memory as the pth curve table.

  Note

  If for example, the sequence is changed between consecutive calls of `CTABID()`, then `CTABID(n, ...)` can be used to provide a **different** curve table than the one provided before the change.

- Indicates **the block state** of curve table number n.
  `CTABISLOCK(n)`
- **Checks** curve table number n.
  `CTABEXISTS(n)`
- Returns the **memory** in which curve table number n is stored.
  `CTABMEMTYP(n)`
- Returns the **table periodicity**.
  `CTABPERIOD(n)`
- Number of **curve segments** already used in memory memType.
  `CTABSEG(memType, segType)`
- Number of curve segments used in **curve table** number n
  `CTABSEGID(n, segType)`
- Number of **still** possible **curve segments** in memory memType.
  `CTABFSEG(memType, segType)`

- **Maximum** number of possible **curve segments** in memory memType.
  `CTABMSEG`(memType, segType)
- Number of **polynomials already used** in memory memType.
  `CTABPOL`(memType)
- Number of **curve polynomials** used by curve table number n.
  `CTABPOLID`(n)
- Number of **still** possible **polynomials** in memory memType.
  `CTABFPOL`(n)
- **Maximum** number of possible **polynomials** in memory memType.
  `CTABMPOL`(n)

Boundary values of curve tables

Behavior of the leading axis/following axes on the edges of the curve table:

- The value at the **beginning** of the curve table is read by a following axis.
  `CTABTSV`(n, degrees, F axis), following value at the beginning of the curve table
- The value at the **end** of the curve table is read by a following axis.
  `CTABTEV`(n, degrees, FAxis), following value at the end of the curve table
- The value at the **beginning** of the curve table is read by the leading axis.
  `CTABTSP`(n, degrees, FAxis), master value at the beginning of the curve table
- The value at the **end** of the curve table is read by the leading axis.
  `CTABTEP`(n, degrees, FAxis), master value at the end of the curve table
- Determine the value range of the following value.
  `CTABTMIN`(n, FAxis), minimum following value of curve table
  `CTABTMAX`(n, FAxis), maximum following value of the curve table

Parameter

- Following axis:
  Name of axis via which the following axis is programmed in the definition.
- Leading axis:
  Name of axis via which the leading axis is programmed.
- n, m
  Numbers for curve tables.
  Curve table numbers can be freely assigned. They are used exclusively to uniquely identify a curve table.
  In order to delete a curve table area using the command `CTABDEL`(n, m) must be greater than n.
- p
  Entry location (in memType memory area)

- applim:
  Behavior at the curve table edges.
  - 0 non-periodic (table is processed only once, even for rotary axes).
  - 1 periodic, modulo (the modulo value corresponds to the LA table values).
  - 2 periodic, modulo (LA and FA are periodic).
- Master value
  Position value for which a following value is to be determined.
- Slave value
  Position value for which a master value is to be calculated.
- aproxmastervalue
  Position value that can be used to determine a unique master value in the case of an ambiguous reversing function of the curve table.
- degrees
  Parameter in which the pitch of the table function is returned.
- memType
  Optional parameter for specifying memory type to be used n curve tables.
  Possible values:
  "SRAM" curve table is created in static NC memory.
  "DSRAM" curve table is created in dynamic NC memory.
  If an invalid type is entered, the value -2 is returned.
  If the parameter is omitted, then the memory type set via the following machine data takes effect:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE (default memory type for curve tables)
- segType
  Optional parameter for entry of segment type
  Possible values:
  segType "L" linear segments
  segType "P" Polynomial segments

Reference:
Programming Manual, Production Planning; Axis Couplings, Section: Curve tables (CTAB)

Restrictions

The following restrictions apply when programming:

- It is not permissible that the NC block generates a preprocessing stop.
- No discontinuities may occur in leading axis motion.
- Any block that contains a traverse instruction for the following axis must also include a traverse for the leading axis.
- The direction of motion of the leading axis must not reverse at any point in the rule of motion, i.e. the position of the leading axis must always be unique within the sequence of motions. The programmed contour may not move perpendicular to the leading axis.

- Axis names from gantry axis groups cannot be used to define a table (only leading axis are possible).
- Depending on the following machine data, jumps in the following axis may be tolerated if a movement is missing in the leading axis.
  MD20900 $MC_CTAB_ENABLE_NO_LEADMOTION (curve tables with discontinuity of the following axis)
  The other restrictions listed above still apply.

Axis assignment

Does not take effect until coupling is activated with curve table.

Note

The dynamic limit values of the motion commands for a curve table are not checked until activation or interpolation.

Starting value

The first motion command in the definition of a curve table defines the starting value for the leading and following value.

All instructions that cause a preprocessing stop must be hidden.

Example 1

Without tool radius compensation, without memory type

Program code	Comment
`N100 CTABDEF(AX2, AX1, 3,0)`	`; start of the definition for the non-`
	`; periodic curve table no. 3`
`N110 AX1=0 AX2=0`	`; 1st Motion instruction specifies the start value`
	`; leading value: 0, Following value: 0`
`N110 AX1=20 AX2=0`	`; 1st Curve segment: Master value: 0...20,`
	`; following value: Starting value …0`
`N120 AX1=100 AX2=6`	`; 2nd Curve segment: Master value: 20...100,`
	`; following value: 0...6`
`N130 AX1=150 AX2=6`	`; 3rd Curve segment: Master value: 100...150,`
	`; following value 6`
`N130 AX1=180 AX2=0`	`; 4th Curve segment: Master value: 150...180,`
	`; following value: 6...0`
`N200 CTABEND`	`; end of the definition, the curve table`
	`; is generated in its internal representation.`
	`; the advance is reorganized to the status`
	`; at the start of N100`

Example 2

Example of a curve table with active tool radius compensation:

Prior to definition of a curve table with `CTABDEF()`, tool radius compensation must not be active; otherwise alarm 10942 is generated. This means that tool radius compensation **must** be activated within the definition of the curve table. Similarly, it must be deactivated again before the end of the curve table definition, using `CTABEND`.

Program code	Comment
`N10 CTABDEF(Y, X, 1, 0)`	`; start of the definition for the non-`
	`; periodic curve table no. 1`
`N20 X0 Y0`	
`N30 G41 X10 Y0`	`; tool radius compensation on`
`N40 X20 Y20`	
`N50 X40 Y0`	
`N60 X60 Y20`	
`N70 X80 Y0`	
`N80 G40 X90 Y0`	`; tool radius compensation off`
`N90 CTABEND`	

Tool radius compensation is activated in block N30; this causes the approach movement for radius compensation to be made in this block. Similarly, the approach movement for deactivation of the radius compensation is made in block N80.

Note

The value pairs between `CTABDEF` and `CTABEND`must be specified for precisely the axis names that have been programmed in `CTABDEF` as the leading axis name and following axis name. In the case of programming errors, alarms or incorrect contours may be generated.

10.2.6 Access to table positions and table segments

Reading table positions

With the program commands `CTAB` and `CTABINV` the following value for a master value (`CTAB`) can be read from the part program and from synchronous actions, or alternatively the master value can be read off for a following value. The pitch value can be used to calculate the speed of the following axis or leading axis at any position in the table.

Reading segment positions

Segment positions of a curve table for the value for the following axis can be read using the `CTABSSV` and `CTABSEV` calls.

The language commands `CTABSSV` and `CTABSEV` generally provide the start and end values of the internal segments of the curve tables for the following axis. These values only agree with the programmed values of the curve tables if the programmed segments can be converted

1:1 to the internal segments of the curve table. This is always the case if only G1 blocks or axis polynomials are used to define the curve tables and no other functions are active.

Programmed sections may under certain circumstances **not** be transformed unchanged into internal curve segments if:

1. Circles or involutes are programmed
2. Chamfer or rounding is active (CHF, RND)
3. Smoothing with G643 is active
4. Compressor is active (COMPON, COMPCURV, COMPCAD)
5. Tool radius compensation is active for polynomial interpolation.

In these cases, the language commands CTABSSV and CTABSEV may not be used to query the start and end points of the programmed segments.

CTABINV

When using the inversion function for the curve tables CTABINV, it must be noted that the following value mapped to the leading value may not be unique.

Within a curve table, the following value can assume the same value for any number of master value positions. In order to resolve this ambiguity, the program command CTABINV requires a further parameter, in addition to the following value, which it uses to select the 'correct' master value. CTABINV always returns the master value that is closest to this auxiliary parameter. This auxiliary value can, for example, be the master value from the previous interpolator clock cycle.

Note

Although the auxiliary parameter permits calculation of a unique result for the reversal function of the curve table, it should be noted that numerical inaccuracies may give rise to contours, which can cause the reversal function to produce results that deviate from those that would be obtained in a calculation where the accuracy is unrestricted.

Optional parameters

The functions CTAB, CTABINV, CTABSSV and CTABSEV have optional parameters for the leading and following axes. If one of these parameters is programmed, the master value and following value are modified using the scaling factors of the relevant axes.

This is particularly important if axes have been configured with different length units (inch/metric). If no optional parameters are programmed, the master value and following value are treated as path positions in the conversion from external to internal representation. This means that the values are multiplied according to the configured resolution (decimal places) and the remaining decimal places are truncated.

Identifying the segment associated with master value X

Example of reading the segment starting and end values for determining the curve segment associated with master value X = 30 using `CTABSSV` and `CTABSEV`:

Program code	Comment
`N10 DEF REAL STARTPOS`	`; Beginning of the definition for the start position of the curve table`
`N20 DEF REAL ENDPOS`	
`N30 DEF REAL GRADIENT`	
`N100 CTABDEF(Y,X,1,0)`	`; Start of table definition`
`N110 X0 Y0`	`; Start position, 1st table segment`
`N120 X20 Y10`	`; End position, 1st table segment =`
	`; Start position 2nd table segment`
`N130 X40 Y40`	
`N140 X60 Y10`	`; 3. Curve segment`
`N150 X80 Y0`	`; 4. Curve segment`
`N160 CTABEND`	`; End of table definition`
`N200 STARTPOS=CTABSSV(30.0,1,GRADIENT)`	`; Start position Y in segment 2 = 10.`
`N210 ENDPOS=`	`; End position Y in segment 2 = 40`
`CTABSEV(30.0,1,GRADIENT)`	`; Segment 2 belongs to LW X = 30.0`

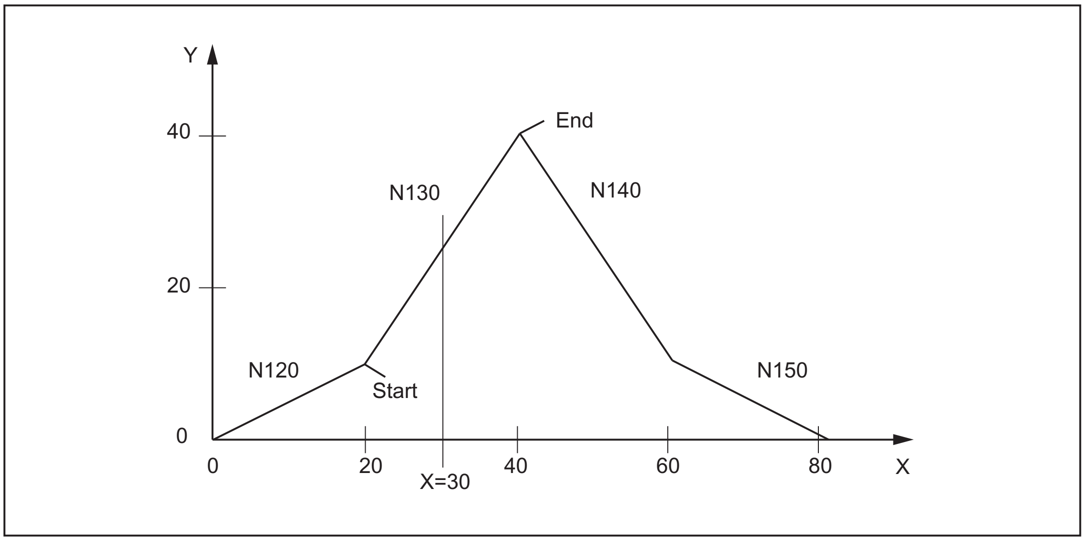


Figure 10-2 Determining the curve segment associated with master value X = 30

Reading values at start and end

The values of the following axes and of the master axis at the start and end of a curve table can be read with the following calls:

R10 =`CTABTSV`(n, degrees, F axis), following value at the beginning of the curve table

R10 =`CTABTEV`(n, degrees, F axis), following value at the beginning of the curve table

R10 =`CTABTSP`(n, degrees, F axis), following value at the beginning of the curve table

R10 =`CTABTEP`(n, degrees, F axis), following value at the beginning of the curve table

Value range of the following value

The following example illustrates how the minimum and maximum values of the table are determined using `CTABTMIN` and `CTABTMAX`:

Program code	**Comment**
N10 DEF REAL STARTVAL	; Beginning of the definition for the start and initial values of the curve table
N20 DEF REAL ENDVAL	
N30 DEF REAL STARTPARA	
N40 DEF REAL ENDPARA	
N50 DEF REAL U_MINVAL	
N60 DEF REAL U_MAXVAL	
N70 DEF REAL GRADIENT	
…	
N100 CTABDEF(Y,X,1,0)	; Start of table definition
N110 X0 Y10	; Start value of the 1st table segment
N120 X30 Y40	; End position 1st table segment = ; start position 2nd table segment
N130 X60 Y5	; End position of the 2nd table segment ...
N140 X70 Y30	; End position of the 3rd curve segment ...
N150 X80 Y20	; End position of the 4th curve segment ...
N160 CTABEND	; End of table definition
...	
N200 STARTPOS=CTABTSV(1,GRADIENT)	; STARTPOS = 10
	; Start position of table as well as
N210 ENDPOS=CTABTEV(1,GRADIENT)	; ENDPOS = 20
	; End position of table
N220 STARTPARA=CTABTSP(1,GRADIENT)	; STARTPARA = 0
	; Master value at beginning of curve table
N230 ENDPARA=CTABTEP(1,GRADIENT)	; ENDPARA = 80
	; Read master value at the end of the curve table from the value range of the following axis
N240 U_MINVAL=CTABTMIN(1)	; Minimum value when Y = 5 and
N250 U_MAXVAL=CTABTMAX(1)	; Maximum value when Y = 40

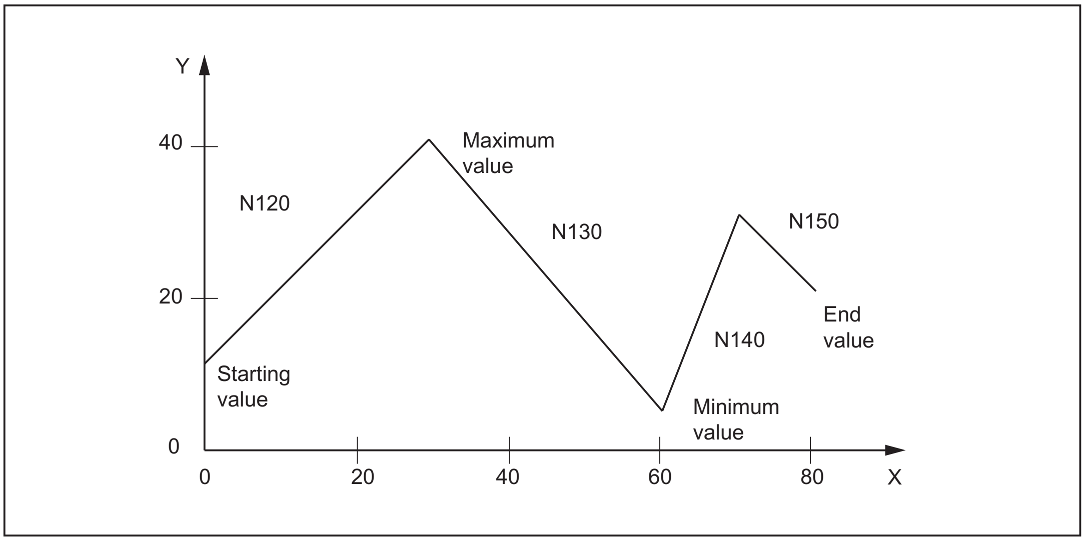


Figure 10-3 Determining the minimum and maximum values of the table

10.2.7 Activation/deactivation

Activation

The coupling of real axes to a curve table is activated through this command:

`LEADON` (<Following axis>, <Leading axis>, <n>)

with <n> =Number of the curve table

Activation is possible:

- In the part program
- in the definition of a synchronous action

Example:

```
...
N1000 LEADON(A,X,3)        ; Axis A follows the master value X according to the rule
                             of motions defined in Curve Table No. 3
...
```

Deactivation

The switch off of the coupling to a curve table takes place through the following command:

`LEADON` (<Following axis>, <Leading axis>)

Deactivation is possible:

- In the part program
- in synchronized actions

Note

While programming `LEADOF`, the abbreviated form is also possible without specification of the leading axis.

Example:

```
...
N1010 LEADOF(A,X)          ; The coupling of Axis A with its leading axis is canceled
...
```

Multiple use

A curve table can be used several times in a single part program to couple different channel axes.

10.2.8 Modulo-leading axis special case

Position is absolute

When an axial master value coupling is active, the position of the following axis via a curve table is unique, i.e. an absolute assignment to the master axis exists.

This means that, when a modulo rotary axis is used as the master axis, the position of the master axis is absolute. In other words, the position of the modulo rotary axis entered in the curve table is absolute, and not modulo-reduced.

Example

Let the position of a modulo rotary axis with `LEADON` be 210°. The position 210° degrees is used as the starting value in the curve table. After one rotation of the modulo axis, the axis position is again displayed as 210°. The absolute position 570° is however taken as the input value in the curve table:

210° + 1 round (360°) = 570°

10.2.9 Behavior in AUTOMATIC, MDA and JOG modes

Activation

An activated curve table is functional in the AUTOMATIC, MDA and JOG modes.

Basic setting after run-up

No curve tables are active after run-up.

10.2.10 Effectiveness of PLC interface signals

Dependent following axis

With respect to the motion of a following axis that is dependent on the leading axis, only the following axis interface signals that effect termination of the motion (e.g. axis-specific feed stop, axis inhibit, servo enable, etc.) are effective.

Leading axis

In an activated axis group, the interface signals move the leading axis through the axis coupling to the associated following axis, i.e.:

- feed control of the leading axis causes a corresponding feed control of the following axis.
- A shutdown of the leading axis through interface signals (e.g., axis-specific feed stop, axis inhibit, servo enable etc.) causes the corresponding following axis to shut down.

The effect of the axis inhibit of the leading axis on the following axis can be prevented through the following MD setting:

MD37160 $MA_LEAD_FUNCTION_MASK, Bit 1 = 1

Position measuring system 1/2 (DB31, ... DBX1.5/1.6))

Switch-over of the position measuring system for the leading and following axes is not inhibited for an active coupled axis group. The coupling is not canceled.

Recommendation: Switch the measuring system over when the coupling is deactivated.

10.2.11 Diagnosing and optimizing utilization of resources

The following functions allow **parts programs** to get information on the current utilization of curve tables, table segments and polynomials.

One result of the diagnostic functions is that resources still available can be used **dynamically** with the functions, without necessarily having to increase memory usage. The description of the parameters in Chapter "Programming Curve Tables" also applies to the following functions.

a) Curve tables

- Determine total number of defined tables.
  The definition applies to all memory types (see also `CTABNOMEM`)
  `CTABNO()`
- Number of defined tables in SRAM or DRAM of NC memory.
  `CTABNOMEM (memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE (default memory type for curve tables)
  Result:
  >= 0: Number of defined curve tables
  -2: Invalid memory type
- Determine number of curve tables still possible in memory.
  `CTABFNO(memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD220905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Number of possible tables
  -2: Invalid memory type
- Determine the table number of the pth table in the memory type specified optionally
  `CTABID(p, memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  Table number or
  Alarm for invalid p or memType

When using the `CTABID(p, memType)` function, no assumptions should be made regarding the sequence of the curve tables in the memory. The `CTABID(p, ...)` function supplies the ID (table number) of the curve table entered in memory as the pth curve table.

If the sequence of curve tables in memory changes between consecutive calls of `CTABID()`CTABID(), e.g. due to the deletion of curve tables with `CTABDEL()`, the `CTABID(p, ...)` function can supply a different curve table with the same number.

To prevent this from happening, the curve tables concerned can be locked, using the `CTABLOCK(...)` language command. In this case, it should be noted that the curve tables concerned are then unlocked with `CTABUNLOCK()`.

- Determine **block condition**
  Table n
  `CTABISLOCK(n)`
  Result:
  > 0: Table is blocked
  Reason for block:
  1: by `CTABLOCK`()
  2: by an active coupling
  3: by `CTABLOCK()`and by an active coupling
  = 0: Table is not blocked
  - 1: Table does not exist
- Check whether the curve table **exists**
  `CTABEXISTS(n)`
  Result:
  1: Table exists
  0: Table does not exist
- Determine **memory type** of a curve table
  `CTABMEMTYP(n)`
  Result:
  0: Table in static SRAM NC memory
  1: Table in dynamic "DRAM" NC memory
  -1: Table does not exist
- by an active coupling **periodic**
  `CTABPERIOD(n)`
  Result:
  0: Table is not periodic
  1: Table is periodic in the leading axis
  2: Table is periodic in the leading and following axes
  -1: Table does not exist

b) Curve table segments

- Determine number of **used** curve segments of the type memType in the memory range.
- `CTABSEG(memType, segType)`
- If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Number of curve segments
  -2: Invalid memory type
  If segType is not specified, the sum is produced via linear and polynomisl segments in the memory type.
  -2: segType not equal "L" or "P"

- Determine number of used curve segments of the type memType in the memory range
  `CTABSEGID(n, segType)`
  Result:
  >= 0: Number of curve segments
  -1: Curve table with number n does not exist
  -2: segType not equal "L" or "P"
- Determine number of **free** curve segments of the type memType in the memory range
  `CTABFSEG(memType, segType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Number of free curve segments
  -2: Invalid memory type, segType not equal "L" or "P"
- Determine **maximum** number of possible curve segments of the type segType in the memory
  `CTABMSEG(memType, segType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Maximum number of possible curve segments
  -2: Invalid memory type, segType not equal "L" or "P"

c) Polynomials

- Determine the number of **used** polynomials of the memory type
  `CTABPOL(memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Number of polynomials already used in the memory type
  -2: Invalid memory type
- Determine the number of curve polynomials used by a curve table
  `CTABPOLID(n)`
  Result:
  >=0: Number of used curve polynomials
  -1: Curve table with number n does not exist
- Determine the number of **free** polynomials of the memory type
  `CTABFPOL(memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Number of free curve polynomials
  -2: Invalid memory type
- Determine the **maximum** number of polynomials of the memory type
  `CTABMPOL(memType)`
  If memType is not specified, the memory type specified in the following machine data:
  MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE
  Result:
  >= 0: Maximum number of possible curve polynomials
  -2: Invalid memory type

10.2.12 Supplementary conditions

Transformations

Transformations are not permissible in curve tables. `TRAANG` is an exception.

TRAANG

If `TRAANG` is programmed, the rule of motion programmed in the basic co-ordinate system is transformed to the associated machine co-ordinate system. In this way it is possible to program a curve table as Cartesian co-ordinates for a machine with inclined linear axes.

The condition that stipulates that "the direction of motion of the leading axis must not reverse at any point of the rule of motion" must then be met in the machine co-ordinate system. Please note that this condition in the basic co-ordinate system does not have the same meaning as in the machine co-ordinate system, since the contour tangents are changed by the transformation.

10.2.13 Examples

Definition of a curve table with linear sets

```
%_N_TAB_1_NOTPERI_MPF
;$PATH=/_N_WKS_DIR/_N_KURVENTABELLEN_WPD
; Def.TAB1 0-100mm Kue1/1 notperio.
N10 CTABDEF(YGEO,XGEO,1,0)                    ; FA=Y LA=X Curve No..=1 Not
                                                period.
N1000 XGEO=0 YGEO=0                           ; Start values
N1010 XGEO=100 YGEO=100
CTABEND
M30
```

Definition of a curve table with polynomial sets

```
%_N_TAB_1_NOTPERI_MPF
;$PATH=/_N_WKS_DIR/_N_KURVENTABELLEN_WPD
; Def.TAB1 0-100mm Kue1/1 notperio.
N10 CTABDEF(Y,X,1,0)                                    ; FA=Y LA=X Curve
                                                          No..=1 Not period.
N16 G1 X0.000 Y0.000
N17 POLY PO[X]=(31.734,0.352,-0.412) PO[Y]=(3.200,2.383,0.401)
N18 PO[X]=(49.711,-0.297,0.169) PO[Y]=(7.457,1.202,-0.643)
N19 PO[X]=(105.941,1.961,-0.938) PO[Y]=(11.708,-6.820,-1.718)
N20 PO[X]=(132.644,-0.196,-0.053) PO[Y]=(6.815,-2.743,0.724)
```

```
%_N_TAB_1_NOTPERI_MPF
N21 PO[X]=(147.754,-0.116,0.103) PO[Y]=(3.359,-0.188,0.277)
N22 PO[X]=(174.441,0.578,-0.206) PO[Y]=(0.123,1.925,0.188)
N23 PO[X]=(185.598,-0.007,0.005) PO[Y]=(-0.123,0.430,-0.287)
N24 PO[X]=(212.285,0.040,-0.206) PO[Y]=(-3.362,-2.491,0.190)
N25 PO[X]=(227.395,-0.193,0.103) PO[Y]=(-6.818,-0.641,0.276)
N26 PO[X]=(254.098,0.355,-0.053) PO[Y]=(-11.710,0.573,0.723)
N26 PO[X]=(254.098,0.355,-0.053) PO[Y]=(-11.710,0.573,0.723)
N27 PO[X]=(310.324,0.852,-0.937) PO[Y]=(-7.454,11.975,-1.720)
N28 PO[X]=(328.299,-0.209,0.169) PO[Y]=(-3.197,0.726,-0.643)
N29 PO[X]=(360.031,0.885,-0.413) PO[Y]=(0.000,-3.588,0.403)
CTABEND
N30 M30
```

Definition of a periodic curve table

Table No: 2

Master value range: 0 - 360

The following axis traverses from N70 to N90, a movement from 0 to 45 and back to 0.

```
N10 DEF REAL DEPPOS
N20 DEF REAL GRADIENT
N30 CTABDEF(Y,X,2,1)
N40 G1 X=0 Y=0
N50 POLY
N60 PO[X]=(45.0)
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)
N80 PO[X]=(270.0)
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)
N100 PO[X]=(360.0)
N110 CTABEND
N130 G1 F1000 X0                        ; Testing the curve by coupling Y to X
N140 LEADON(Y,X,2)
N150 X360
N160 X0
N170 LEADOF(Y,X)
N180 DEPPOS = CTAB(75.0,2,GRADIENT)     ; Reading the table position at master
                                          value 75.0 from the curve table with
                                          Table No. 2
N190 G0 X75 Y=DEPPOS                    ; Positions of leading and following
                                          axis
N200 LEADON(Y,X,2)                      ; No synchronization of the following
                                          axis is required after the coupling
                                          is activated
N210 G1 X110 F1000
N220 LEADOF(Y,X)
```

```
N10 DEF REAL DEPPOS
N230 M30
```

10.3 Master value coupling - 840D sl only

10.3.1 Product brief

10.3.1.1 Function

The "axial master value coupling" function can be used to process short programs cyclically with close coupling of the axes to one another and a master value that is either generated internally or input from an external source.

The master value can, for example, be derived from a conveyor belt or a vertical shaft.

The axial master value coupling can be switched on and off in the NC part program or via synchronized action.

10.3.1.2 Preconditions

The option "Master Value Coupling and Curve Tables Interpolation" or the relevant option of generic coupling (refer to " Preconditions (Page 510) " in the "Brief description" for generic coupling) is a prerequisite for utilization of the function.

10.3.2 General functionality

Curve table

With the axial master value coupling, a leading and a following axis are moved in synchronism. It is possible to assign the position of the following axis via a curve table or the resulting polynomial uniquely to a position of the leading axis - simulated if necessary.

Master value object

The master value object is the input variable for the curve table.

The following can be defined as the position of the master value object:

- The axis actual position (actual value measured by transmitter)
  or
- The setpoint position (calculated from interpolator) (default)

If the leading axis is interpolated by the same NCU, the setpoint coupling delivers a better follow-up response than is possible for actual value coupling (in the same IPO cycle).

Virtual leading axis / simulated master value

If the leading axis is not interpolated by the same NCU, the interpolator that is implemented in the NCU for this particular leading axis can be used for master value simulation. The following machine data settings must be defined for this:

MD30132 $MA_IS_VIRTUAL_AX[n] = 1 (axis is virtual axis)

MD30130 $MA_CTRLOUT_TYPE[n] = 0 (simulation as output type of setpoint)

Properties of master value simulation:

- Separation of IPO and servo.
- Actual values of the axis are recorded.
- Setpoint values are produced by IPO but not passed on to the servo motor.
- When switching over to master value coupling, the simulation can be programmed with the last actual value read, whereas the path of the actual value is generally outside the control of the NCU.
- If, for purposes of master value simulation, the master value object is switched from actual value coupling to setpoint value coupling and a traversing command is issued for the leading axis in the same interpolator clock cycle, the interpolator for the axis is initialized by the NC so that the master value produces a constant path in the first derivation.

Note

Virtual axes assigned to a real drive must remain unblocked.

Offset and scaling

The setpoint value for the following axis can be offset and scaled. The following setting data is used for this:

SD43102 $SA_LEAD_OFFSET_IN_POS (offset of master value with coupling to this axis)

SD43104 $SA_LEAD_SCALE_IN_POS (scaling of master value with coupling to this axis)

SD43106 $SA_LEAD_OFFSET_OUT_POS (offset of function value of the curve table)

SD43108 $SA_LEAD_SCALE_OUT_POS (scaling of function value of the curve table)

If (x) is a periodic curve table and this is interpreted as oscillation, the offset and scaling can also be interpreted as follows:

SD43102 $SA_LEAD_OFFSET_IN_POS[Y] offsets the phase of the oscillation.

SD43106 $SA_LEAD_SCALE_OUT_POS[Y] affects the amplitude.

SD43108 $SA_LEAD_OFFSET_OUT_POS[Y] offsets the center of the oscillation

If the coupling is activated and synchronous, the new set position is approached as soon as values are written to this setting data.

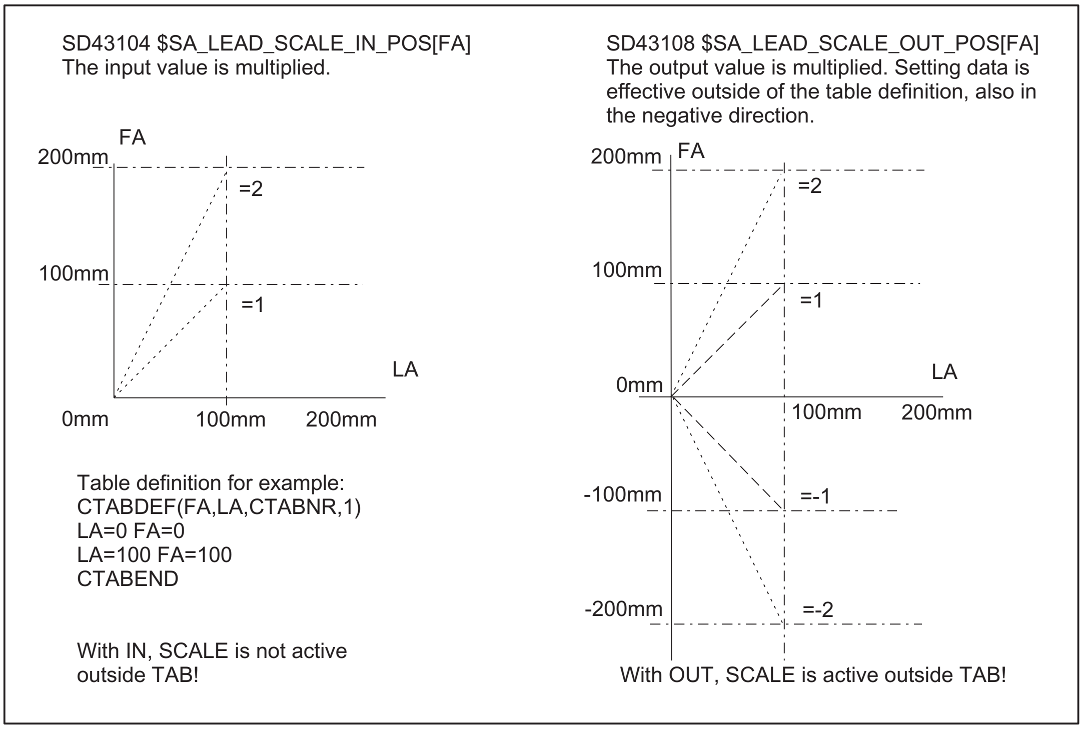

SD43104 $SA_LEAD_SCALE_IN_POS[FA]
The input value is multiplied.
FA
200mm
100mm
=2
=1
LA
0mm
100mm
200mm
Table definition for example:
CTABDEF(FA,LA,CTABNR,1)
LA=0 FA=0
LA=100 FA=100
CTABEND
With IN, SCALE is not active outside TAB!
SD43108 $SA_LEAD_SCALE_OUT_POS[FA]
The output value is multiplied. Setting data is effective outside of the table definition, also in the negative direction.
200mm
FA
=2
100mm
=1
0mm
LA
100mm
200mm
-100mm
=-1
-200mm
=-2
With OUT, SCALE is active outside TAB!

Figure 10-4 Master value coupling offset and scaling (multiplied)

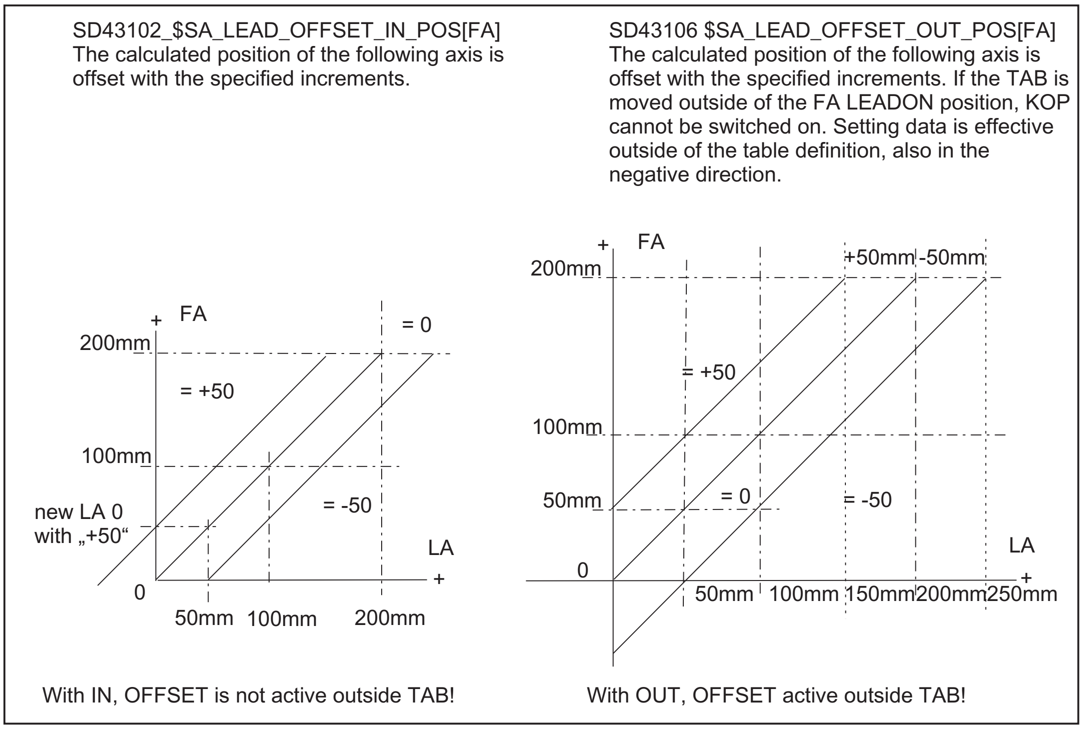

SD43102_$SA_LEAD_OFFSET_IN_POS[FA]
The calculated position of the following axis is offset with the specified increments.
+ FA
200mm
= 0
= +50
100mm
= -50
new LA 0 with „+50“
LA
+
0
50mm
100mm
200mm
With IN, OFFSET is not active outside TAB!
SD43106 $SA_LEAD_OFFSET_OUT_POS[FA]
The calculated position of the following axis is offset with the specified increments. If the TAB is moved outside of the FA LEADON position, KOP cannot be switched on. Setting data is effective outside of the table definition, also in the negative direction.
+ FA
200mm
+50mm
-50mm
= +50
100mm
50mm
= 0
= -50
LA
0
+
50mm
100mm
150mm
200mm
250mm
With OUT, OFFSET active outside TAB!

Figure 10-5 Master value coupling offset and scaling (with increment offset)

Reaction to Stop

All leading value coupled following axes react to channel stop and MODE GROUP stop.

Master value coupled following axes react to a stop due to end of program (`M30`, `M02`) if they have not been activated by static synchronized actions (`IDS=...`). The following machine data is to be observed in this connection:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET / TP End)

MD20112 $MC_START_MODE_MASK (definition of initial control settings for NC-START)

Leading axis and following axis must always interpolate in the same channel. A following axis located in a different channel cannot be coupled (axis replacement).

START and mode change enable a following axis in the master value coupling that has been stopped.

RESET also enables a stopped following axis in master value coupling. If enabling by RESET is not desired, or if it is dangerous (e.g. because the following axis is coupled to an external master value not controlled by the NC), then MD20110 must be programmed such that master value couplings are switched off with RESET (=2001H, i.e. bit 13 set to 1).

Axial functions

Actual value coupling causes a position offset between the leading and following axis. The cause of this is the IPO cycle-based dead time in the position controller, which lies between the actual value of the leading axis and the following axis.

Normally, the position offset and the following error is compensated by a linear extrapolation of the master value to the extent of this dead time, i.e. dead time compensation is active in master value coupling. The following MD setting is defined to deactivate the dead time compensation:

MD37160 $MA_LEAD_FUNCTION_MASK Bit 0 = 0

Interface to axis exchange

A master value coupled following axis receives its setpoint values from curve tables. **Overlaid programming of this axis is not possible in the part program.** Therefore, the master value-coupled following axis is removed from the channel in the same way as for axis exchange. This is carried out automatically when the coupling is activated in the part program.

If the coupling is to be activated with synchronized action, then it must be prepared for it in advance with `RELEASE`.

After a master value coupling has been deactivated, the former following axis can be programmed again in the part program.

Spindles in master value coupling

A spindle can only be used as the master value coupled following axis if it has been switched to axis mode beforehand. The machine data parameter block of the axis drive then applies.

Example: Activation from synchronized action

Program code	Comment
`SPOS=0`	
`B=IC(0)`	`; switch spindle to axis operation.`
`RELEASE(Y)`	`; Enable for synchronized action.`
`ID=1 WHEN ($AA_IM[X]<-50) DO LEADON(B,X,2)`	`; Y is coupled to X via curve table No. 2.`

10.3.3 Programming

Definition and activation

An axis master value coupling is defined and activated simultaneously with the modal effective language command `LEADON`.

Syntax:
`LEADON(<FA>,<LA>,<CTABn>)`

Meaning:

`<FA>`	Following axis as geometry-, channel- or machine axis name (X, Y, Z,...)
`<LA>`	Leading axis as geometry-, channel- or machine axis name (X, Y, Z,...) Software axis is also possible: MD30130 $MA_CTRLOUT_TYPE=0 (setpoint output type)
`<CTABn>`	Number of curve table Range of values: 1 to 999

Example:

Program code	Comment
`LEADON(Y,X,1)`	`; Definition and activation of a master value coupling between the leading axis X and the following axis Y. Curve Table No. 1 should be used to calculate the following value.`

Boundary conditions:

- No reference point is required to activate the coupling.
- A defined following axis cannot be traversed in the JOG mode (not even if the "Synchronized run fine" or. "synchronized run coarse" interface signal is not there).
- An activated coupling must first be deactivated with `LEADOF` before it can be activated again with `LEADON`. The settings in the following machine data are to be considered in this connection:
  MD20112 MC_START_MODE_MASK (definition of initial control system settings with NC-START)
  MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET/TP-End)

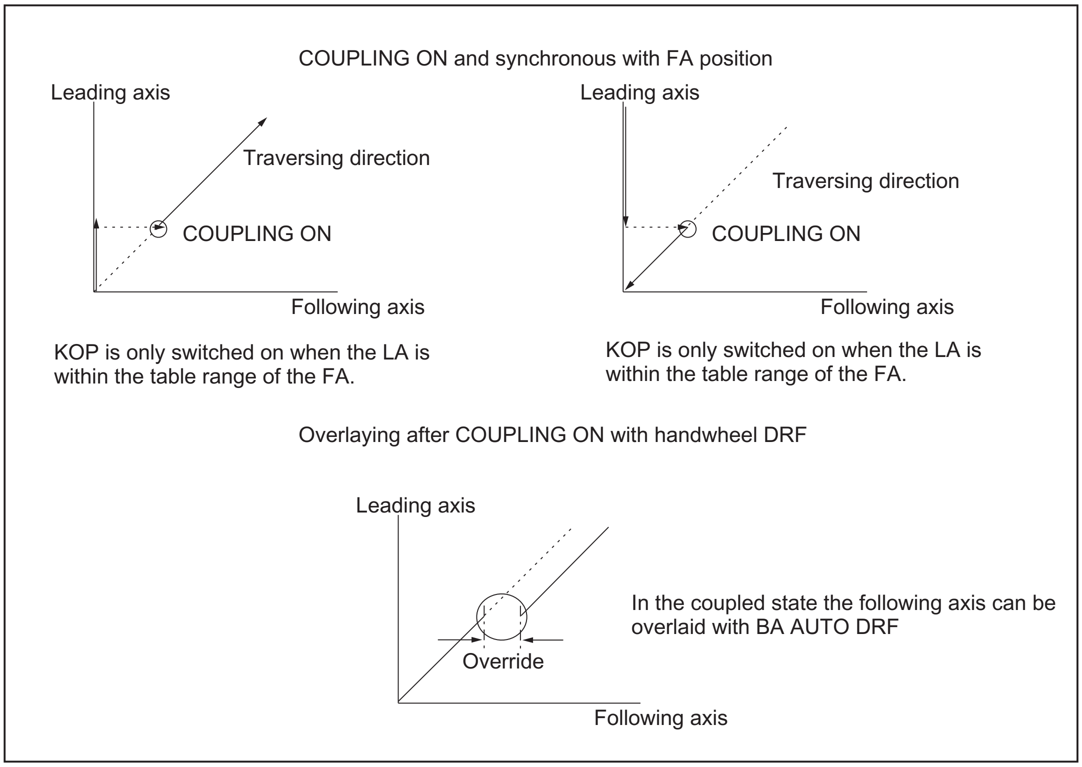

COUPLING ON and synchronous with FA position
Leading axis
Traversing direction
COUPLING ON
Following axis
KOP is only switched on when the LA is within the table range of the FA.
Leading axis
Traversing direction
COUPLING ON
Following axis
KOP is only switched on when the LA is within the table range of the FA.
Overlaying after COUPLING ON with handwheel DRF
Leading axis
In the coupled state the following axis can be overlaid with BA AUTO DRF
Override
Following axis

Figure 10-6 Activating master value coupling

Switch off

An axis master value coupling is deactivated with the model language command `LEADOF`.

When the axis master value coupling is deactivated, the following axis becomes the command axis and a stop command is generated implicitly for the following axis. The stop command can be overwritten by another command with a synchronous action.

Syntax:
```
LEADON(<FA>,<LA>)
```

Meaning:

`<FA>`	Following axis as geometry-, channel- or machine axis name (X, Y, Z,...)
`<LA>`	Leading axis as geometry-, channel- or machine axis name (X, Y, Z,...) Software axis is also possible: MD30130 $MA_CTRLOUT_TYPE=0 (setpoint output type)

Example:

Program code	Comment
`LEADOF(Y,X,1)`	`; Switching off of master value coupling between the leading axis X and the following axis Y.`

Note

Activating / deactivating the axis master value coupling with `LEADON` / `LEADOF` is permissible both in the part program and in synchronous actions.

References:
Function Manual, Synchronized Actions

Coupling type

The coupling type is defined by the following axis-specific setting data:

SD43100 $SA_LEAD_TYPE[<LA>] (type of master value)

<LA>: Leading axis as geometry axis name, channel axis name or machine axis name (X, Y, Z,...)

Value	Meaning
0	Actual value coupling (this type of coupling must be used for external leading axes)
1	Setpoint coupling (default setting)
2	Simulated master value (note virtual axis, not evaluated for FA)

Switch-over between actual and setpoint value coupling is possible at any time (preferably in the idle phase).

System variables of the master value

The following master value system variables can only be read from part program and from synchronous actions:

System variable	Meaning
$AA_LEAD_V[ax]	Velocity of the leading axis
$AA_LEAD_P[ax]	Position of the leading axis
$AA_LEAD_P_TURN	Master value position Portion that is deducted during modulo reaction. The actual (not modulo-reduced) position of the leading axis is: $AA_LEAD_P_TURN + $AA_LEAD_P

The speeds and positions of simulated master values (when $SA_LEAD_TYPE[ax]=2) can be written in **and** read from the part program and synchronous actions.

System variable	Meaning
$AA_LEAD_SV[ax]	Simulated master value velocity per IPO cycle
$AA_LEAD_SP[ax]	Simulated position in MCS

System variables of the following axis

For the following axis, the following system variables can be read in the part program and synchronized action:

System variable	Meaning		
$AA_SYNC[ax]	Condition of the coupling between following and leading axes		
	Value	**Meaning**	
	0	not synchronized	
	1	"Coarse synchronous operation" 1)	
	2	"Fine synchronous operation" 2)	
	3	"Coarse synchronous operation" 1) AND "Fine synchronous operation" 2)	
		Corresponds to:	
		1)	• MD37200 $MA_COUPLE_POS_TOL_COARSE • DB31, ... DBX98.1 (coarse synchronous operation)
		2)	• MD37210 $MA_COUPLE_POS_TOL_FINE • DB31, ... DBX98.0 (fine synchronous operation)
$AA_IN_SYNC[ax]	Condition of the synchronization between following and leading axes		
	Value	**Meaning**	
	0	Synchronization not started or ended.	
	1	Synchronization is running, i. e., the following axis is synchronized.	
	$AA_IN_SYNC[ax] corresponds to the NC/PLC interface signal: DB31, ... DBX99.4 (synchronization running)		

Note

If the following axis is not enabled for travel, it is stopped and is no longer synchronous.

10.3.4 Behavior in AUTOMATIC, MDA and JOG modes

Efficiency

A master value coupling is active depending on the settings in the part program and in the following machine data:

MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET / TP End)

MD20112 $MC_START_MODE_MASK (definition of initial control system settings with NC-START)

Manual mode

Once a master axis coupling has been activated, traversal of the master axis (e.g. with rapid traverse or incremental dimension INC1 ... INC10000) results in a movement of the slave axis, allowing for the curve table definition.

Referencing

A master value coupled-following axis should be referenced prior to activation of the coupling. A following axis cannot be referenced when the coupling is activated.

Deletion of distance-to-go

When deletion of distance-to-go is performed for a leading axis, all axes in the associated, activated master value coupling are shut down.

Basic setting after POWER ON

No master value couplings are active after POWER ON. (Options with ASUB).

Behavior after NC start/RESET

The following behavior results, depending on the settings of the machine data:

MD20110 $MC_RESET_MODE_MASK (bit 13) (definition of initial control settings after RESET / TP End)

MD20112 $MC_START_MODE_MASK (bit 13) (definition of initial control system settings with NC-START)

- MD20110 $MC_RESET_MODE_MASK=2001H
  &&
  MD20112 $MC_START_MODE_MASK=0H
  → Master value coupling remains valid after `RESET` and `START`
- MD20110 $MC_RESET_MODE_MASK=2001H
  &&
  MD20112 $MC_START_MODE_MASK=2000H
  → Master value coupling remains valid after `RESET` and is canceled with `START`. However, master value coupling activated via IDS=... remains valid.
- MD20110 $MC_RESET_MODE_MASK=1H
  → Master value coupling is canceled with `RESET` irrespective of machine data:
  MD20112 $MC_START_MODE_MASK
  Master value coupling activated via IDS=... can only be deactivated via an operator front panel reset and remains valid after program end/reset (M30, M02).
- MD20110 $MC_RESET_MODE_MASK=0H
  → Master value coupling remains valid after `RESET` and is canceled with `START`, irrespective of machine data:
  MD20112 $MC_START_MODE_MASK
  However, master value coupling activated via IDS=... remains valid.

Reference:
Function Manual, Basic Functions; Coordinate Systems, Axis Types,
Axis Configurations, ... (K2)

Activating, deactivating

Master value couplings activated via a static synchronous action (IDS=...) are:

- not deactivated during program start, regardless of the value of machine data:
  MD20110 $MC_RESET_MODE_MASK (definition of initial control settings after RESET / TP End)
  and
  MD20112 $MC_START_MODE_MASK (definition of initial control system settings with NC-START)
- not deactivated during program end reset (M30, M02), regardless of the value of machine data:
  MD20110 $MC_RESET_MODE_MASK

10.3.5 Effectiveness of PLC interface signals

Leading axis

When a coupled axis group is active, the interface signals (IS) of the leading axis are applied to the appropriate following axis via axis coupling. i.e.:

- a feed control action of the leading axis is applied via the master value coupling to effect an appropriate feed control action in the following axis.
- shutdown of the leading axis as the result of an IS (e.g. axis-specific feed stop, axis inhibit, servo enable, etc.) causes the corresponding coupled motion axis to shut down.

Position measuring system 1/2 (DB31, ... DBX1.5/1.6))

Switch-over of the position measuring system for the leading and following axes is not inhibited for an active coupled axis group. The coupling is not canceled.

Recommendation: Switch the measuring system over when the coupling is deactivated.

10.3.6 Special characteristics of the axis master value coupling function

Control dynamics

Depending on the application in question, it may be advisable to match the position controller parameter settings (e.g. servo gain factor ($K_V$factor)) of the leading axis and following axis in an axis grouping. It may be necessary to activate other parameter sets for the following axis. The dynamics of the following axis should be the same or better than those of the leading axis.

Status of coupling

See Chapter "Status of coupling (Page 444)"

Actual value display

The display of the actual value is updated for all axes of in a master value coupled axis grouping (only real axes) coupled via a master value.

Interpolation

When the movement defined in the curve table is interpolated, axis position and axis speed are calculated for a master value and its speed.

Logging

The curve tables generated by the definition of motion sequences are stored in the battery-backed memory.

The curve tables are not lost when the control system is switched off.

These functions have no effect on cyclic machines because they are performed without operator actions. Nor does it make sense to perform automatic (re-)positioning via the NC with external master values.

10.3.7 Supplementary conditions

External master value axes

When using the `REPOS` or `REPOSA` parts program instructions in conjunction with external master value axes, it should be ensured that these are released by the channel or switched to a "neutral state" using the `RELEASE` instruction.

When attempting to reposition without release of the axis, the message "Wait: Feed stop active" is displayed and the processing of the part program is not continued.

10.4 Electronic gear (EG)

10.4.1 Product brief

10.4.1.1 Function

General

The "electronic gear" function makes it possible to control the movement of a following axis, depending on up to five master axes. The relationship between each leading axis and the following axis is defined by the coupling factor. Following axis motion components derived from the individual leading axis motion components have an additive effect.

The coupling can be based on:

- Actual value of the leading axis
- Setpoint of the leading axis

The following functions of a gear grouping can be programmed using part program instructions:

- Defining
- Switch on
- Switch off
- Delete

Curve tables

Non-linear relationships between lead and following axes can also be implemented using curve tables.

Cascading

Electronic gearboxes can be cascaded, i.e. the following axis of an electronic gearbox can be the leading axis for a subsequent electronic gearbox.

Synchronous position

An additional function for synchronizing the following axis allows a synchronous position to be selected:

- Approach next division (tooth gap) time-optimized
- Approach next division (tooth gap) path-optimized
- Approach in positive direction of axis rotation, absolute
- Approach in negative direction of axis rotation, absolute
- Traverse time-optimized with respect to programmed synchronized position
- Traverse path-optimized with respect to programmed synchronized position

Application Examples:

- Machine tools for gear cutting
- Gear trains for production machines

10.4.1.2 Preconditions

The "Electronic Gearbox" option or the relevant option of generic coupling (refer to " Preconditions (Page 510) " in the "Brief description" of Generic Coupling) is required for the usage of function.

10.4.2 Electronic gear (EG)

Function

With the aid of the "Electronic gearbox" the movement of a **following axis FA** can be interpolated dependent of up to five **leading axes LA**. The relationship between each leading axis and the following axis is defined by a coupling factor. The following axis motion components derived in this manner from the individual leading axis motion components have an additive effect.

FA_{set} = SynPosFA + (LA_{1}-$SynPosLA_{1}$)*CF_{1} +... +(LA_{5}-$SynPosLA_{5}$)*CF_{5}

with:

SynPosFA, $SynPosLA_{i}$: from call EGONSYN (see below)

$FA_{setpoint}$: Partial setpoint of the following axis.

LA_i: Setpoint or actual value of the ith leading axis (depending on the type of coupling - see below)

KF_i: Coupling factor of the ith leading axis (see below)

All paths are referred to the basic co-ordinate system **BCS**.

When an EG axis group is activated, it is possible to synchronize the leading axes and following axis in relation to a defined starting position.

From the part program a gearbox group can be:

- defined,
- activated,
- deactivated,
- deleted.

Extensions

The influence of each of the 5 leading axes can be specified using a **curve table** as an alternative to a transmission ratio (KF=numerator/denominator).

It is thus possible for each curve (except for the special case of a straight line) for the leading axis to influence the following axis in a **non-linear** manner. The function can only be used with EGONSYN.

The function EG can be activated with curve tables with EGON.

The function EGONSYNE is available for approaching the synchronized position of the following axis with a specified approach mode.

For special applications, it may be advisable configure the position controller as a **PI controller**.

⚠ CAUTION
Application faults
Knowledge of the control technology and measurements with servo trace are an absolute prerequisite for using this function.

References:

- CNC Commissioning Manual: NC, PLC, Drive
- Function Manual, Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)

Coupling type

The following axis motion can be derived from either of the following:

- Setpoints of leading axes
- Actual values of leading axes

The reference is set in the definition call for the EG axis group:

EGDEF

(see Chapter "Definition of an EG axis group (Page 491)")

Coupling factor

The coupling factor must be programmed for each leading axis in the group. It is defined by numerator/denominator.

Coupling factor values numerator and denominator are entered per leading axis with the following activation calls:

EGON

EGONSYN

EGONSYNE

(see Chapter "Activating an EG axis group (Page 492)")

Number of EG axis groups

Several EG axis groups can be defined at the same time. The maximum possible number of EG axis groupings is set in the following machine data:

MD11660 $MN_NUM_EG

The maximum permissible number of EG axis groups is 31.

Note

The option must be enabled.

EG cascading

The following axis of an EG can be the leading axis of another EG. For a sample configuration file, see Chapter "Examples".

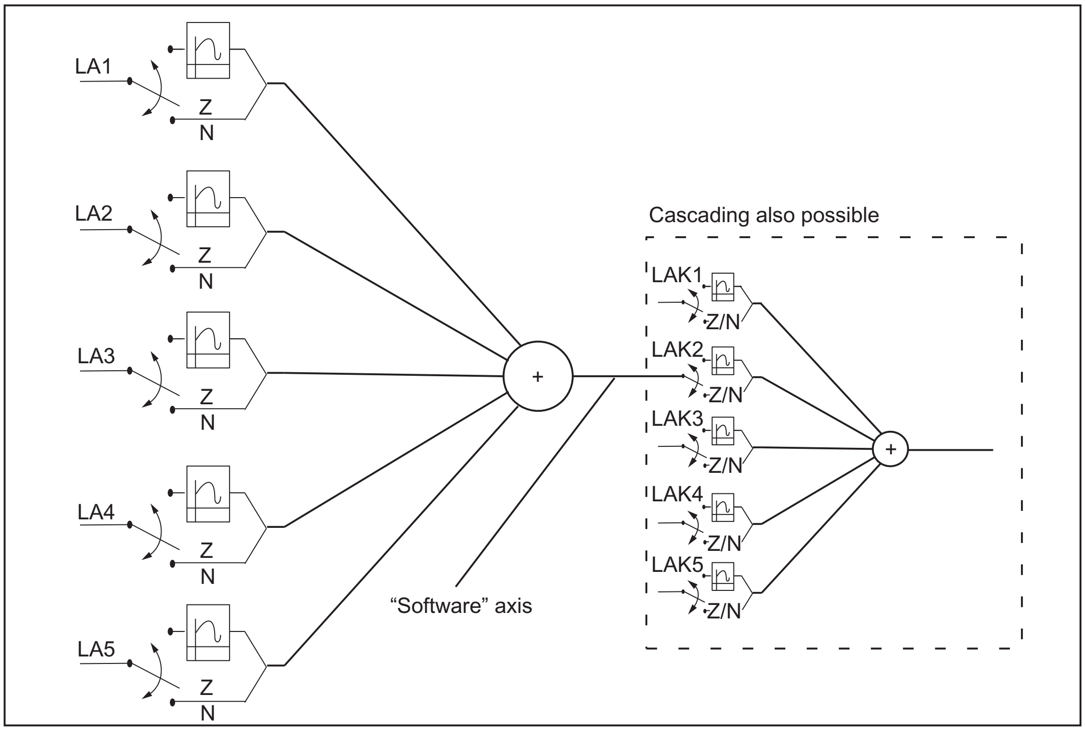


Figure 10-7 Block diagram of an electronic gearbox

Synchronous positions

To start up the EG axis group, an approach to defined positions for the following axis can first be requested.

Synchronous positions are specified with:

EGONSYN (see below for details)

EGONSYNE (extended EGONSYN call).

Synchronization

If a gear is started with EGON(), EGONSYN() or EGONSYNE() see below, the actual position of the following axis is only identical to the setpoint position defined by the rule of motion of the gear specified by the positions of the leading axes at this time if the part program developer makes sure that it is. The control then uses the motion of the following axis to ensure that the setpoint and actual positions of the following axes correspond as quickly as possible if the leading axes are moved further. This procedure is called synchronization. After synchronization of the following axis, the term **synchronous** gearing is used.

Activation response

An electronic gearbox can be activated in two different ways:

1. On the basis of the axis positions that have been reached up to now in the course of processing the command to activate the EG axis group is issued without specifying the synchronizing positions for each individual axis.
   EGON (see Chapter "Activating an EG axis group (Page 492)")
2. The command to activate the EG axis group specifies the synchronized positions for each axis. From the point in time when these positions are reached, the EG should be synchronized.
   EGONSYN (see Chapter "Activating an EG axis group (Page 492)")
3. The command to activate the EG axis group specifies the synchronized positions and approach mode for each axis. From the point in time when these positions are reached, the EG should be synchronized.
   EGONSYNE (see Chapter "Activating an EG axis group (Page 492)")

Synchronization with EGON

With EGON(), no specifications are made for the positions at which the following axis is to be synchronized. The control system activates the EG and issues the signal "Synchronized position reached".

Synchronization for EGONSYN

1. With EGONSYN(), the positions of the leading axes and the synchronization position for the following axis are specified by the command.

- The control then traverses the following axis with just the right acceleration and velocity to the specified synchronization position so that the following axis is in position with the leading axes at its synchronization position.
- If the following axis is **stationary: I**f the "Feed stop/spindle stop" DB 31, ... DBX 4.3 is set for the following axis, the following axis is **not** set in motion by EGON or EGONSYN. A traverse command is issued and block changing is blocked until the axis-specific feed is enabled. EGOSYN is topped by RESET transformed into EGON. The programmed synchronized positions are deleted.
- If the following axis is **not stationary**: The NST "Feed Stop/Spindle Stop" DB31, ... DBX4.3 has no direct influence on the electronic gearbox. As before, it does have an indirect effect on the leading axes, if these are located in the same channel.
- For **channel specific** feed enabling and for override nothing is implemented. Override still has no direct influence on the electronic gearbox. The **axis-specific** feed enable is set, depending on the current override setting.

Synchronization for EGONSYNE

With EGONSYN(), the positions of the leading axes and the synchronization position for the following axis are specified by the command.

The control system moves the following axis to the synchronized position according to the program approach mode.

Synchronization abort with EGONSYN and EGONSYNE

1. The EGONSYN/EGONSYNE command is aborted under the following conditions and changed to an EGON command:

- RESET
- Axis switches to tracking

The defined synchronization positions are ignored. Synchronous traverse monitoring still takes synchronized positions into account.

Aborting position synchronization generates alarm 16774.

The alarm may be suppressed with the following machine data:

MD11410 $MN_SUPPRESS_ALARM_MASK Bit31 = 1

Synchronous monitoring

The synchronism of the gearbox is monitored in each interpolator clock cycle on the basis of the actual values of the following and leading axes. For this purpose, the actual values of the axes are computed according to the rule of motion of the coupling. The **synchronism difference** is the difference between the actual value of the following axis and the value calculated from the leading axis actual values according to the rule of motion. The synchronism difference can be queried from the part program (see below).

Changes in the synchronism difference

The mass inertia of the axis systems during acceleration can cause dynamic fluctuations in the synchronism difference. The synchronism difference is checked continuously and the tolerance values in the machine data used to produce interface signals.

The synchronism difference is compared with the following machine data:

MD37200 $MA_COUPLE_POS_TOL_COARSE

MD37210 $MA_COUPLE_POS_TOL_FINE

Depending on the result of this comparison, the following signals are set:

NST "Synchronous travel fine" DB31, ... DBX98.0

NST "synchronous traverse coarse" DB31, ... DBX98.1

Difference > .. TOL_COARSE

As long as the synchronism difference is greater than the following machine data, the gearbox is not synchronized and neither IS "Coarse synchronism" DB 31, ... DBX 98.1 nor IS "Fine synchronism" DB 31, ... DBX 98.0 is active:

MD37200 $MN_COUPLE_POS_TOL_COARSE

Instead, the following interface signal is displayed:

NST "synchronization running" DB31, ... DBX99.4

Difference < .. TOL_COARSE

As long as the synchronism difference is smaller than the following machine data, IS "Coarse synchronism" DB 31, ... DBX 98.1 is at the interface and IS "Fine synchronism" DB31, ... DBX99.4 is deleted:

MD37200 $MN_COUPLE_POS_TOL_COARSE

Difference > .. TOL_FINE

If the synchronism difference is smaller than the following machine data, then NST "synchronous traverse fine" DB31, ... DBX98.0 is at the interface:

MD37210 $MA COUPLE_POS_TOL_FINE

Synchronism difference for EG cascades

Synchronism difference for EG cascades is the deviation of the actual position of the following axis from setpoint position that results fro the rule of motion for the real axes involved.

Example:

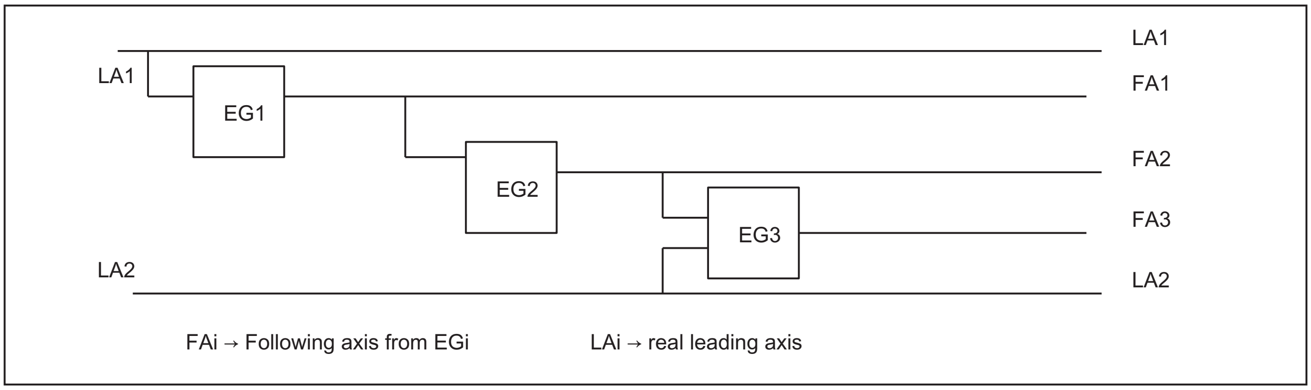


Figure 10-8 Three-level EG cascade

According to the definition given, the synchronism difference of following axis FA3 in the example below is determined by the value of following axis $FA3_{Act}$ and the value of leading axis $FA2_{Act}$ and $LA2_{Act}$, but not by $LA1_{Act}$ and $FA1_{Act}$.

If FA2 is not a real axis, the actual value $FA2_{Act}$ is not available. In this case, the **setpoint** of the axis derived solely from the leading axis value $FA1_{Act}$ must be used instead of the actual value of the setpoint of the axis.

Other signals

If an EGON(), EGONSYN() or EGONSYNE() block is encountered in the main run, the signal "Coupling active" is set for the following axis. If the following axis is only overlaid, the signals "Coupling active" and "Axis override" are set. If EGON(), EGONSYN() or EGONSYNE() is active and the following axis is also overlaid, the signals "Coupling active" and "Axis override" are also set.

IS "following spindle active" DB31, ... DBX 99.1: Coupling active,

IS "overlaid movement" DB31, ... DBX98.4: axis is overlaid, IS "Enable following axis override" DB31, ... DBX26.4

In the case of the commands EGON() and EGONSYNE(), the "Enable following axis override" signal must be present for the gear to synchronize to the specified synchronization position for the following axis. If it is not present, alarm 16771 "Override movement not enabled" is issued. If the signal is present, the following axis travels to the synchronized position with the calculated acceleration and at the velocity set for the approach mode.

Further monitoring signals

Machine data MD37550 $MA_EG_VEL_WARNING allows a percentage of the speeds and accelerations to be specified in the following machine data MD32000 $MA_MAX_AX_VELO and MD32300 $MA_MAX_AX_ACCEL, with reference to the following axis, which results in the generation of the following interface signals:

IS "Speed warning threshold" DB31, ... DBX98.5

IS "Acceleration warning threshold" DB31, ... DBX98.6

The monitoring signals can be used as trigger criteria for emergency retraction (see Chapter "Trigger sources (Page 610)").

Machine data MD37560 $MA_EG_ACC_TOL allows a percentage with reference to machine data MD32300 $MA_MAX_AX_ACCEL of the following axis to be defined, and the IS signal "axis accelerates" DB31, ... DBX99.3 to be generated.

Request synchronism difference

1. The result of the synchronism difference calculation can be read as an amount in the part program with system variable $VA_EG_SYNCDIFF. The relevant value with sign is available in the system variables $VA_EG_SYNCDIFF_S. The following meanings apply:

- Negative value (in positive traverse direction for lead and following axis): The following axis lags behind its calculated setpoint position.
- Positive value (in positive traverse direction for lead and following axis): The following axis leads before its calculated setpoint position (overswing).

The amount of the synchronism difference with sign corresponds to the system variables without sign from $VA_EG_SYNCDIFF.

$VA_EG_SYNCDIFF[ax] = ABS($VA_EG_SYNCDIFF_S[ax])

Block change mode

1. When an EG axis group is activated, it is possible to specify the conditions under which a part program block change is to be executed:
2. The specification is made with a string parameter with the following meaning:
3. "NOC": Immediate block change
4. "FINE": Block change if "Fine synchronism" is present
5. "COARSE": Block change if "Coarse synchronism" is present
6. "IPOSTOP": Block change if "Setpoint synchronism" is present

---

Note

When programmed in activation calls EGON, EGONSYN, EGONSYNE, each of the above strings can be abbreviated to the first two characters.

If no block change has been defined for the EG axis group and none is currently specified, "FINE" applies.

10.4.3 Definition of an EG axis group

Note

The following definition commands and switch instructions of the electronic gearbox must all be contained in **only one block** of a parts program.

All commands of the electronic gearbox result in a **preprocessing stop**, except for the activation commands:

- EGON
- EGONSYN
- EGONSYNE

Definition and activation

The definition described below and activation are separate processes. An activation is not possible unless it has been defined previously.

Definition of an EG axis group

An EG axis group is defined through the input of the following axis and at least one, but not more than five, leading axis, each with the relevant coupling type:

EGDEF(following axis, leading axis1, coupling type1, leading axis2, coupling type 2,...)

The coupling type does not need to be the same for all leading axes and must be programmed separately for each individual leading axis.

Coupling type:

Evaluate actual value of leading axis: 0

Evaluate setpoint of leading axis: 1

The coupling factors are preset to zero when the EG axis group is defined. As such, the group has no effect on the following axis until it is activated. (See EGON, EGONSYN, EGONSYNE).

Preconditions for defining an EG axis group:

No existing axis coupling may already be defined for the following axis. (If necessary, an existing axis must be deleted with EGDEL.)

EGDEF triggers preprocessing stop with alarm.

For an example of how to use the EG gearbox for gear hobbing, please see Chapter "Examples", "Electronic Gearbox for Gear Hobbing".

EGDEF

The gearbox definition with EGDEF should also be used unaltered when one or more leading axes affect the following axis via a curve table.

The variant extended with the addition of non-linear coupling via curve tables is illustrated in an extended example in Chapter "Extended Example with non-linear Components".

10.4.4 Activating an EG axis group

Without synchronization

The EG axis group is activated **without synchronization selection** with:

EGON(FA, block change mode, LA1, Z1, N1, LA2 , Z2, N2,..LA5, Z5, N5.)

The coupling is activated immediately.

With:

FA: Following axis

Depending on block change mode, the next block will be activated:

"NOC": Block change takes place immediately

"FINE": Block change is performed in "Fine synchronism"

"COARSE": Block change is performed in "Coarse synchronism"

"IPOSTOP": Block change is performed for setpoint-based synchronism

LA_i: Axis name of the leading axis i

Z_i: Counter for coupling factor of leading axis i

N_i: Denominator for coupling factor of leading axis i

Only the leading axes previously specified with the EGDEF command may be programmed in the activation line. At least one leading axis must be programmed.

The positions of the leading axes and following axis at the instant the grouping is switched on are stored as "Synchronized positions". The "Synchronized positions" can be read with the system variable $AA_EG_SYN.

With synchronization

The EG axis group is activated **with synchronization selective**:

1. EGONSYN

EGONSYN(FA, block change mode, SynPosFA, LA_i, $SynPosLA_i$, Z_LA_i, N_LA_i)

With:

FA: Following axis

Block change mode:

"NOC": Block change takes place immediately

"FINE": Block change is performed in "Fine synchronism"

"COARSE": Block change is performed in "Coarse synchronism"

"IPOSTOP": Block change is performed for setpoint-based synchronism

SynPosFA: Synchronized position of the following axis

LA_i: Axis name of the leading axis i

SynPosLAi: Synchronized position of leading axis i

Zi: Counter for coupling factor of leading axis i

Ni: Denominator for coupling factor of leading axis i

Note

The parameters indexed with i must be programmed for at least one leading axis, but for no more than five.

Only leading axes previously specified with the EGDEF command may be programmed in the activation line.

Through the programmed "Synchronized positions" for the following axis (SynPosFA) and for the leading axes (SynPosLA), positions are defined for which the axis grouping is interpreted as *synchronous*. If the electronic gear is not in the synchronized state when the grouping is switched on, the **following axis** traverses to its defined synchronized position.

The position specification of the synchronized positions is specified in the configured basic system independently of the programmable dimensions (`G70`/`G71`).

If the axis grouping includes modulo axes, their position values are reduced in the modulo, This way the fastest possible synchronized position is approached reliably, e. g., the next tooth from the tooth spacing (360 degree* Zn/Ni) and the associated synchronized positions. (So-called *relative synchronization*.)

thereby ensuring that they approach the fastest possible synchronized position. (So-called *relative synchronization*, e.g. the next tooth gap after "centering".)

If the following interface signal has not been set for the following axis, the axis will not travel to the synchronization position.

DB31, ... DBX26.4 (Enable following axis override)

Instead the program is stopped at the EGONSYN block and the self-clearing alarm 16771 is issued until the above mentioned signal is set.

2. EGONSYNE

EGONSYNE(FA, block change mode, SynPosFA, Approach mode, LA_i,$SynPosLA_i$, Z_LA_i, N_LA_i)

with:

"FA": Following axis

Block change mode:

"NOC": Block change takes place immediately

"FINE": Block change is performed in "Fine synchronism"

"COARSE": Block change is performed in "Coarse synchronism"

"IPOSTOP"**:** Block change is performed for setpoint-based synchronism

SynPosFA**:** Synchronized position of the following axis

Approach mode:

"NTGT": NextToothGapTime-optimized, the next tooth gap is approached time-optimized (preset is used if no setting is applied).

"NTGP": The next tooth gap is approached time-optimized.

"ACN": AbsoluteCoordinateNegative, Absolute measurement specification, rotary axis traverses in negative rotation direction

"ACP": AbsoluteCoordinatePositive, Absolute measurement specification, rotary axis traverses in positive rotation direction

"DCT": DirectCoordinateTime-optimized, Absolute measurement specification, rotary axis traverses time-optimized to programmed synchronized position

"DCP": DirectCoordinatePath-optimized, Absolute measurement specification, rotary axis traverses path-optimized to programmed synchronized position

LA_i: Axis name of the leading axis i

SynPosLAi: Synchronized position of leading axis i

Zi: Counter for coupling factor of leading axis i

Ni: Denominator for coupling factor of leading axis i

Note

The parameters indexed with i must be programmed for at least one leading axis, but for no more than five.

The function is active only for modulo following axes that are coupled to modulo leading axes.

Tooth gap

The tooth gap is defined as 360 degrees * Zi / Ni

Example:

EGONSYNE(A, "FINE", FASysPos, "Traversing mode", B, LASynPos, 2, 10)

Tooth gap: 360*2/10 = 72 (degrees)

Approach response with FA at standstill

In this case, the time-optimized and path-optimized traversing modes are identical.

The table below shows the target positions and traversed paths with direction marker (in brackets) for the particular approach modes:

Programmed synchronized position FaSysPos	Position of the following axis before EGONSYNE	Traversing mode NTGT/ NTGP	Traversing mode DCT/DCP	Traversing mode ACP	Traversing mode ACN
110	150	182 (+32)	110 (-40)	110 (+320)	110 (-40)
110	350	326 (-24)	110 (+120)	110 (+120)	110 (-240)
130	0	346 (-14)	130 (+130)	130 (+130)	130 (-230)
130	30	58 (+28)	130 (+100)	130 (+100)	130 (-260)
130	190	202 (+12)	130 (-60)	130 (+300)	130 (-60)
190	0	334 (-26)	190 (-170)	190 (+190)	190 (-170)
230	0	14 (+14)	230 (-130)	230 (+230)	230 (-130)

Approach response for moving FA

The following axis moves at almost maximum velocity in the positive direction when the coupling is activated by EGONSYNE. The programmed synchronized position of the following axis is 110, the current position 150. This produces the two alternative synchronized positions 110 and 182 (see table above).

In the case of traversing mode NTGP (path-optimized), synchronized position 182 is selected independent of the current velocity. This has the shortest distance from the current position of the following axis. Traversing mode NTGT (time-optimized) considers the current speed of the following axis and produces a deceleration on account of the limit for the maximum axis speed to reach synchronism in the shortest possible time (see Figure).

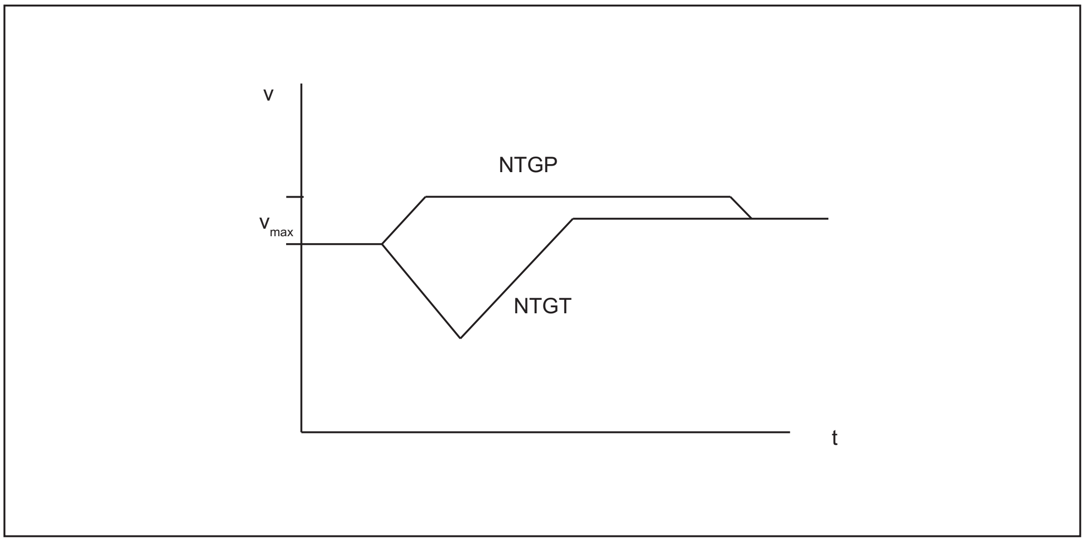


Figure 10-9 Reaching the next tooth gap, FA path-optimized (top) vs. time-optimized (bottom)

Sample notations

EGONSYNE(A, "FINE", 110, "NTGT", B, 0, 2, 10)

couple A to B, synchronized position A = 110, B = 0, coupling factor 2/10, approach mode = NTGT

EGONSYNE(A, "FINE", 110, "DCT", B, 0, 2, 10)

couple A to B, synchronized position A = 110, B = 0, coupling factor 2/10, approach mode = DCT

EGONSYNE(A, "FINE", 110, "NTGT", B, 0, 2, 10, Y, 15, 1, 3)

couple A to B, synchronized position A = 110, B = 0, Y = 15,

Coupling factor to B = 2/10, Coupling factor to Y = 1/3,

approach mode = NTGT

With synchronization

The syntax specified above applies with the following different meanings:

If a **curve table** is used for a leading axis, then the denominator of the linear coupling of the coupling factor (N_i) must be set to 0 (denominator 0 would be impermissible for linear couplings).

For control, denominator zero is the indicator that the counter of the coupling factor (Z_i) is to be interpreted as the number of the curve table to be used. The curve table with the specified number must already be defined at power on (in accordance with Chapter "Curve Tables").

The leading axis specified (LA_i) corresponds to the one specified for coupling via coupling factor (linear coupling).

10.4.5 Deactivating an EG axis group

Variant 1

There are different ways to deactivate an active EG axis grouping.

EGOFS(following axis)

The electronic gear is deactivated. The following axis is braked to a standstill. This call triggers a preprocessing stop.

Variant 2

The following parameterization of the command makes it possible to **selectively** control the influence of individual leading axes on the motion of the following axis.

EGOFS(following axis, leading axis 1, ... leading axis 5)

Note

At least one leading axis must be specified.

The influence of the specified leading axes on the slave is selectively inhibited. This call triggers a preprocessing stop.

If the call still includes active leading axes, then the slave continues to operate under their influence. If the influence of all leading axes is excluded by this method, then the following axis is braked to a standstill.

If the command EGONSYN is deactivated selectively, no axis movement is performed.

Variant 3

EGOFC(following spindle)

The electronic gear is deactivated. The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. This call triggers a preprocessing stop.

Note

Call for following **spindles** available. For EGOFC a spindle name must be programmed.

10.4.6 Deleting an EG axis group

An EG axis grouping must be switched off, as described in Chapter "Switching off a EG Axis Group", before its definition can be deleted.

EGDEL(following axis)

The defined coupling of the axis grouping is deleted. Additional axis groups can be defined by means of EGDEF until the maximum number of simultaneously activated axis groups is reached.

This call triggers a preprocessing stop.

10.4.7 Interaction between rotation feedrate (G95) and electronic gearbox

The FPR( ) part program command can be used to specify the following axis of an electronic gear as the axis, which determines the rotational feedrate. The following behavior is applicable in this case:

- The feedrate is determined by the setpoint velocity of the following axis of the electronic gear.
- The setpoint velocity is calculated from the speeds of the leading spindles and modulo axes (which are not path axes) and from their associated coupling factors.
- Velocity components from other leading axes and overlaid motions of the following axis are not taken into account.

References: /V1/, Feeds

10.4.8 Response to POWER ON, RESET, operating mode change, block search

Function	Behavior regarding electronic gearbox	
	Coupling state	**Configuration**
Mode change	Is retained	Is retained
End of part program	Is retained	Is retained
Reset	Is retained	Is retained
Power On 1)	Is not retained	Is not retained
1) **No** coupling is active after Power On.		

Block search

A search while a coupling is active (EG) is possible under the following supplementary conditions:

- Simulation is exclusively with setpoint coupling.
- No cross-channel leading axes may be disabled.
- Axis movements for which all real positions are known to the NC.

If it is technically not sensible or not possible to permit the target block for a block search with calculation or SERUPRO within a part program section with active coupling, this section can be locked for "continue machining at the contour".

Programming

- IPTRLOCK()
- IPTRUNLOCK()

References

(K1) Mode group, channel, program operation, reset response , Section "Block search type 5 SERUPRO" > "Lock program section for continue machining at the contour"

10.4.9 System variables for electronic gearbox

Application

The following system variables can be used in the part program to scan the current states of an EG axis group and to initiate appropriate reactions if necessary:

Table 10-1 System variables, R means: Read access possible

Name	Type	Access		Preprocessing stop		Meaning, value	Cond. Index
		part pro-gram	Sync act.	part pro-gram	Sync act.		
$AA_EG_ TYPE[a,b]	INT	R		R		Type of coupling: 0: actual value 1: Setpoint value coupling	Axis name a: Following axis b: Leading axis
$AA_EG_ NUMERA[a,b]	REAL	R		R		Numerator of coupl. factor KF KF = numerator/denomina-tor preset: 0 Number of curve table when $AA_EG_DE-NOM[a,b] is 0.	Axis name a: Following axis b: Leading axis
$AA_EG_ DENOM[a,b]	REAL	R		R		Denominator of coupl. fact. KF KF = numerator/denomina-tor preset: 1 Denominator must be posi-tive. Denominator is 0, if instead of the numerator $AA_EG_NUMERA[a,b] the number of a curve table is specified.	Axis name a: Following axis b: Leading axis
$AA_EG_ SYN[a,b]	REAL	R		R		Synchronized position for specified leading axis Default: 0	Axis name a: Following axis b: Leading axis
$AA_EG_ SYNFA[a]	REAL	R		R		Synchronized position for specified following axis Default: 0	Axis name a: Following axis

Name	Type	Access		Preprocessing stop		Meaning, value	Cond. Index
		part pro-gram	Sync act.	part pro-gram	Sync act.		
$P_EG_BC[a]	STRING	R		R		Block change criterion for EG activation calls: EGON, EGONSYN: "NOC": immediately "FINE": Synchronoous tra-verse fine "COARSE": Synchronous traverse coarse "IPOSTOP": Setpoint-based synchronism	Axis name a: Following axis
$AA_EG_ NUM_LA[a]	INT	R		R		Number of leading axes de-fined with EGDEF. 0 if no axis has been defined as a following axis with EGDEF.	Axis name a: Following axis
$AA_EG_ AX[n,a]	AXIS	R		R		Axis name of leading axis whose index n has been specified.	Axis name n: Index of leading axis in the EG coupling 0 ... 4 a: Following axis
$AA_EG_ ACTIVE[a,b]	BOOL	R		R		Determine power-on state of a leading axis: 0: Switched off 1: activated	Axis name a: Following axis b: Leading axis
$VA_EG_ SYNCDIFF[a]	REAL	R	R	R		Actual value of synchron-ism difference. A compari-son of machine data MD37200 $MA_COU-PLE_POS_TOL_COARSE and MD37210 $MA_COU-PLE_POS_TOL_FINE pro-duces interface signals.	Axis name a: Following axis

10.4.10 Examples

10.4.10.1 Example using linear couplings

Use of axes

The following diagram shows the configuration of a typical gear hobbing machine. The machine comprises five numerically closed loop controlled axes and an open loop controlled main spindle. These are:

- The rotary motion of the workpiece table (C) and hobbing cutter (B).
- The axial axis (Z) for producing the feed motion over the entire workpiece width.
- The tangential axis (Y) for moving the hobbing cutter along its axis.

- The radial axis (X) for infeeding the cutter to depth of tooth.
- The cutter swivel axis (A) for setting the hobbing cutter in relation to the workpiece as a function of cutter lead angle and angle of inclination of tooth.

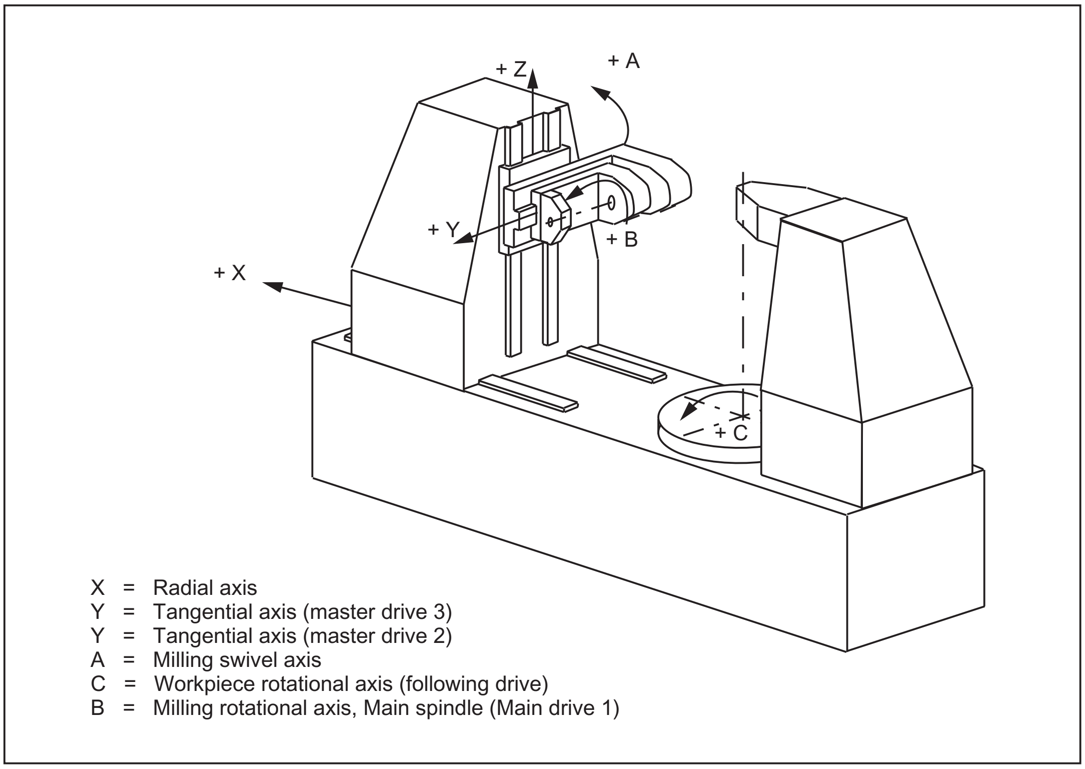
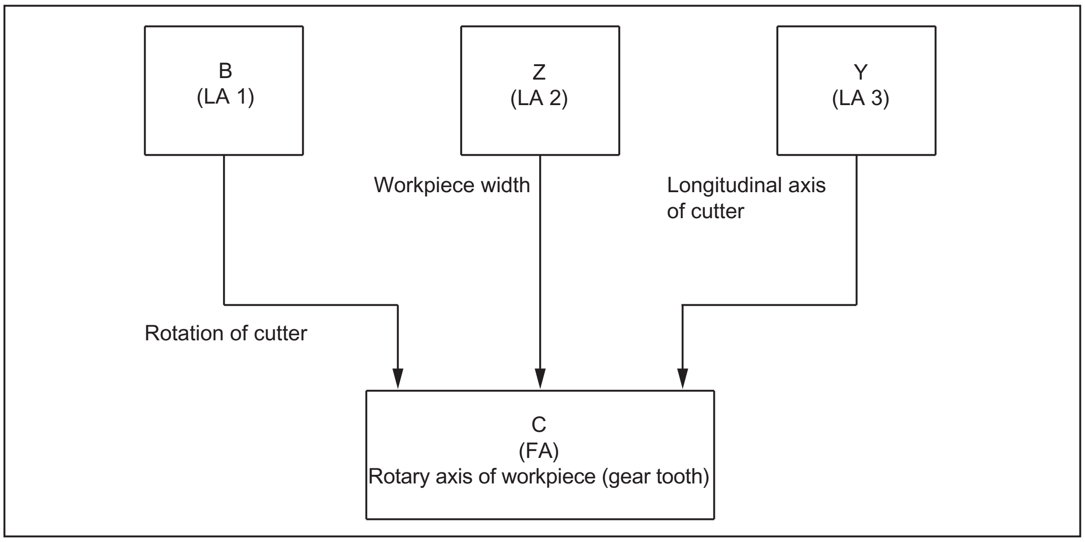

+ Z
+ A
+ Y
+ B
+ X
+ C
X = Radial axis
Y = Tangential axis (master drive 3)
Y = Tangential axis (master drive 2)
A = Milling swivel axis
C = Workpiece rotational axis (following drive)
B = Milling rotational axis, Main spindle (Main drive 1)

Figure 10-10 Definition of axes on a gear hobbing machine (example)

The functional interrelationships on the gear hobbing machine are as follows:



In this case, the workpiece table axis (C) is the following axis which is influenced by three master drives.

The setpoint of the following axis is calculated cyclically with the following logic equation:

$n_c = n_b * (z_0 / z_2) + v_z * (u_{dz} / z_2) + v_y * (u_{dy} / z_2)$

with:	n_c	= Rotational speed of workpiece axis (C)
	n_b	= Rotational speed of milling spindle (B)
	z_0	= Number of gears of the hobbing machine
	z_2	= Number of teeth of the workpiece
	v_z	= Feed velocity of axial axis (Z)
	v_y	= Feed velocity of tangential axis (Y)
	u_{dz}	= Axial differential constant
	u_{dy}	= Tangential differential constant

Quantities which influence the setpoint of workpiece axis C

The first addend of the above equation determines the speed ratio between workpiece table and cutter, and thus the number of teeth of the workpiece.

The second addend effects the necessary additional rotation of the C axis as a function of the axial feed motion of the cutter to produce the tooth inclination on helical teeth.

The third component also makes allowance for additional rotation of the C axis to compensate for the tangential movement of the cutter in relation to the workpiece, thus ensuring that the tool is equally stressed over its entire length.

Workpiece/tool parameter

The values z_0, z_2, u_{dz}, and u_{dy} are workpiece- or tool-dependent and are specified by the NC operator or in the part program.

Differential constants

Differential constants u_{dz} and u_{dy} make allowance for the angle of workpiece teeth and for cutter geometry. These differential constants can be determined in user-specific cycles.

$u_{dz} = (\sin \beta° / (m_n * \pi)) * 360$ [degrees/mm]

$u_{dy} = (\cos \gamma° / (m_n * \pi□)) * 360$ [degrees/mm]

with:	m_n	= Normal module (in mm)
	β°	= Incline angle of gear wheel
	γ°	= Pitch angle of hobbing machine

Extract from part program:

Program code	Comment
`EGDEF(C,B,1,Z,1,Y,1)`	; Definition of EG axis grouping with setpoint coupling (1) from B, Z, Y to C (following axis).
`EGON(C,"FINE",B,z0,z2,Z,udz,z2,Y,udy,z2)`	; Activate coupling.
…	

10.4.10.2 Extended example with non-linear components

Introduction

The following example extends the example (see "Figure 10-10 Definition of axes on a gear hobbing machine (example) (Page 501)") with the following:

- Machine error compensations which are not linearly dependent on the Z axis, and
- a Z-axis dependent component with tooth geometry.
  This can be used to create a slightly convex tooth surface, so that the centre of the tooth is stressed more than the edges during operation.

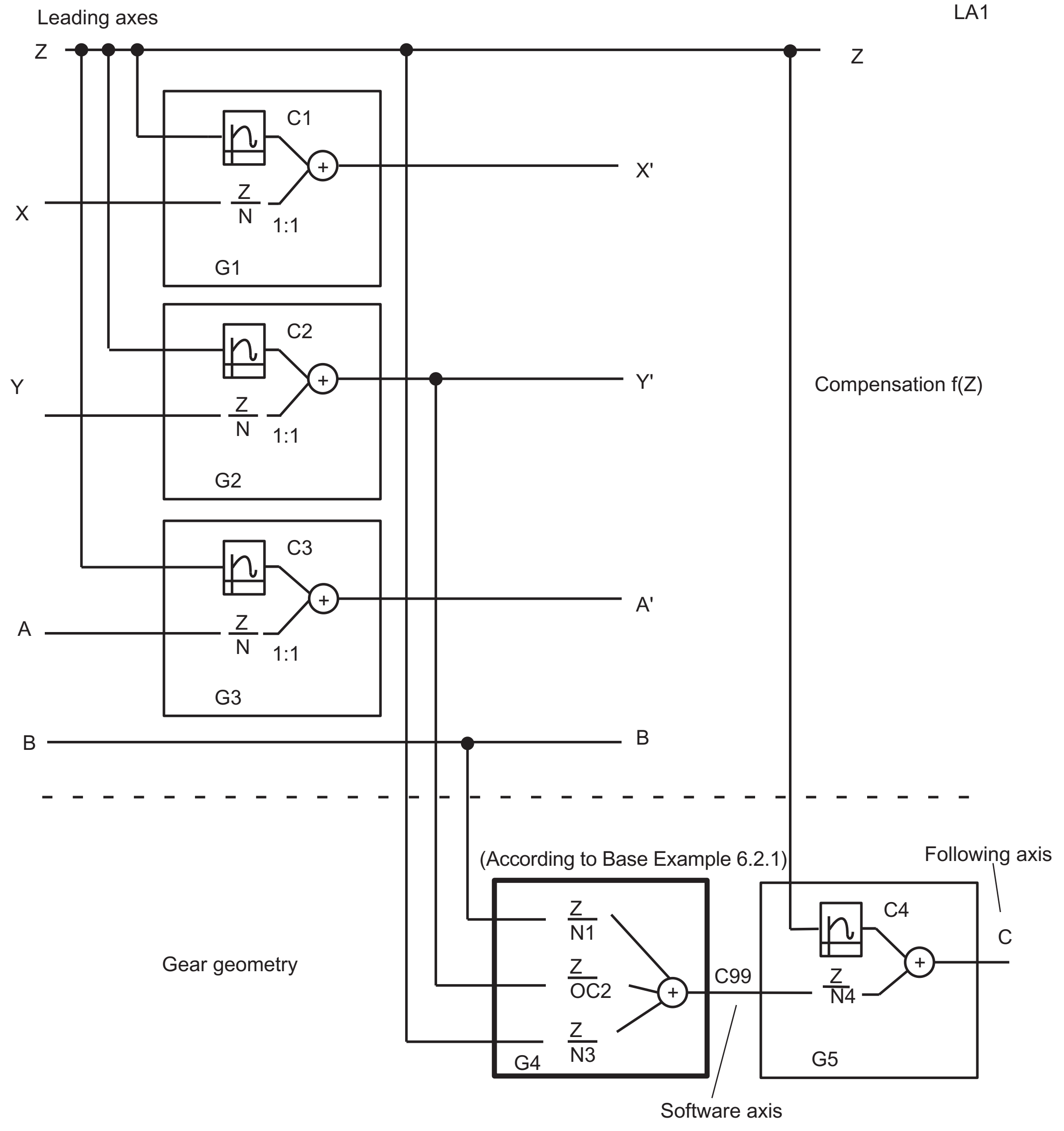


Figure 10-11 Extended example with non-linear machine fault compensation and non-linear components on the tooth geometry

The following section of a part program is intended to illustrate the general concept; supplementary curve tables and gear wheel/machine parameters are still to be added. Components to be added are marked with <...> . Stated parameters may also have to be modified, e.g. coupling factors.

Program code			**Comment**
N100		CTABDEF(X, Z, 1, 0)	; declaration and specification of non-periodic curve table C1
N110	< ... >		; Preset of curve table: Curve points or polynomial blocks
N190		CTABEND	
N200		CTABDEF(Y, Z, 2, 0)	; declaration and specification of non-periodic curve table C2
N210	< ... >		; Preset of curve table: Curve points or polynomial blocks
N290		CTABEND	
N300		CTABDEF(A, Z, 3, 0)	; declaration and specification of non-periodic curve table C3
N310	< ... >		; Preset of curve table: Curve points or polynomial blocks
N390		CTABEND	
N400		CTABDEF(C, Z, 4, 0)	; declaration and specification of non-periodic curve table C4
N410	< ... >		; Preset of curve table: Curve points or polynomial blocks
N490		CTABEND	
N500		EGDEF(X, Z, 1)	; Declaration of path via C1, setpoint coupling
N510		G1 F1000 X10	; declaration of command component of X
N520		EGONSYN(X, "NOC", <SynPosX>, Z, <SynPosX_Z>, **1**, 0)	; Path switch-on via C**1**
N600		EGDEF(Y, Z, 1)	; Declaration of path via C2, setpoint coupling
N610		G1 F1000 Y10	; declaration of command component of Y
N620		EGONSYN(Y, "COARSE", <SynPosY>, Z, <SynPosY_Z>, **2**, 0)	; Path switch-on via C**2**
N700		EGDEF(A, Z, 1)	; Declaration of path via C3, setpoint coupling
N710		G1 F1000 A10	; declaration of command component of A
N720		EGONSYN(A, "FINE", <SynPosA>, Z, <SynPosA_Z>, **3**, 0)	; Path switch-on via C**3**
; 1. Gear stage, C99 is the software axis between the two electronic gears			
N800		EGDEF(C99, Y, 1, Z, 1, B, 1)	
N810		EGONSYN(C99, "NOC", <SynPosC99>, B, <SynPosC99_B>, 18, 2, &	; Switch-on of leading axis B

Program code		Comment
	Y, <SynPosC99_Y>, R1 * π, 1, &	; Switch-on of leading axis Y
	Z, <SynPosC99_Z>, 10, 1)	; Switch-on of leading axis Z
		; "&" character means: command continued in next line, no LF nor comment permissible in program
; 2. Gear stage		
N900	EGDEF(C, C99, 1, Z, 1)	; declaration of following C99 of step 1 as leading axis of step 2,
		; Setpoint value coupling
N910		; Declaration of path via C4, setpoint cou-pling
N920	EGONSYN(C, "NOC", <SynPosC>, C99, <SynPosC_C99>, 1, 1, &	; Switch-on of software axis C99
	Z, <SynPosC_Z>, **4**, 0)	; and of leading axis Z via C**4**
N999	M30	

Machine data

Only one section is specified, which extends beyond the necessary geometry/channel configuration and machine axis parameters.

$MN_NUM_EG = 5	; Maximum number of gearboxes
$MN_MM_NUM_CURVE_TABS = 5	; Maximum number of curve tables
$MN_MM_NUM_CURVE_SEGMENTS = 50	; Maximum number of curve segments
$MN_MM_NUM_CURVE_POLYNOMS = 100	; Maximum number of curve polynomials

Setting data

If the scaling described in Section "Electronic gear (EG) (Page 482) is used, the function value from the following machine data changes according to the displacement:

MD43108 $SD_LEAD_SCALE_OUT_POS[**4**] = 1.2 ; scaling for table C**4**

System variables

In accordance with the above definitions, the following values are entered in the associated system variables by the control.

Reference:
List Manual, System Variables

The system variables listed below are only used **for explanatory purposes**!

```
; ************** Gear X (G1)
$AA_EG_TYPE[X, Z] = 1                  ; Setpoint value coupling
$AA_EG_NUMERA[X, Z] = 1                ; curve table No. = 1
$AA_EG_DENOM[X, Z] = 0                 ; nominator = 0 → curve table applies
$P_EG_BC[X] = "NOC"                    ; Block change criterion
$AA_EG_NUM_LA[X] = 1                   ; Number of leading axes
$AA_EG_AX[0, X] = Z                    ; name of leading axis
$AA_EG_SYN[X,Z] = <SynPosX_Z>          ; Synchronized position of leading axis Z
$AA_EG_SYNFA[X] = <SynPosX>            ; Synchronized position of the following axis
; ************** Gear Y (G2)
$AA_EG_TYPE[Y, Z] = 1                  ; Setpoint value coupling
$AA_EG_NUMERA[Y, Z] = 2                ; curve table No. = 2
$AA_EG_DENOM[Y, Z] = 0                 ; nominator = 0 → curve table applies
$P_EG_BC[Y10] = "COARSE"               ; Block change criterion
$AA_EG_NUM_LA[Y] = 1                   ; Number of leading axes
$AA_EG_AX[0, Y] = Z                    ; name of leading axis
$AA_EG_SYN[Y, Z] = <SynPosY_Z>         ; Synchronized position of leading axis Z
$AA_EG_SYNFA[Y] = <SynPosY>            ; Synchronized position of the following axis
; ************** Gear A (G3)
$AA_EG_TYPE[A, Z] = 1                  ; Setpoint value coupling
$AA_EG_NUMERA[A, Z] = 3                ; curve table No. = 3
$AA_EG_DENOM[A, Z] = 0                 ; nominator = 0 → curve table applies
$P_EG_BC[A10] = "FINE"                 ; Block change criterion
$AA_EG_NUM_LA[A] = 1                   ; Number of leading axes
$AA_EG_AX[0, A] = Z                    ; name of leading axis
$AA_EG_SYN[A, Z] = <SynPosA_Z>         ; Synchronized position of leading axis Z
$AA_EG_SYNFA[A] = <SynPosA>            ; Synchronized position of the following axis
; ************** Gear C99 (G4)
$AA_EG_TYPE[C99, Y] = 1                ; Setpoint value coupling
$AA_EG_NUMERA[C99, Y] = 18             ; numerator for coupling factory
$AA_EG_DENOM[C99, Y] = 2               ; denominator for coupling factory
$AA_EG_TYPE[C99, Z] = 1                ; Setpoint value coupling
$AA_EG_NUMERA[C99, Z] = R1 * π         ; numerator for coupling factorz
$AA_EG_DENOM[C99, Z] = 1               ; denominator for coupling factorz
$AA_EG_TYPE[C99, B] = 1                ; Setpoint value coupling
$AA_EG_NUMERA[C99, B] = 10             ; numerator for coupling factorb
$AA_EG_DENOM[C99, B] = 1               ; denominator for coupling factorb
$P_EG_BC[C99] = "NOC"                  ; Block change criterion
$AA_EG_NUM_LA[C99] = 3                 ; Number of leading axes
$AA_EG_AX[0, C99] = Y                  ; name of leading axis Y
$AA_EG_AX[1, C99] = Z                  ; name of leading axis Z
```

```
$AA_EG_AX[2, C99] = B                    ; name of leading axis B
$AA_EG_SYN[C99, Y] = <SynPosC99_Y>       ; Synchronized position of leading axis Y
$AA_EG_SYN[C99, Z] = <SynPosC99_Z>       ; Synchronized position of leading axis Z
$AA_EG_SYN[C99, B] = <SynPosC99_B>       ; Synchronized position of leading axis B
$AA_EG_SYNFA[C99] = <SynPosC99>          ; Synchronized position of the following axis
; *************** Gear C (G5)
$AA_EG_TYPE[C, Z] = 1                    ; Setpoint value coupling
$AA_EG_NUMERA[C, Z] = 4                  ; curve table No. = 4
$AA_EG_DENOM[C, Z] = 0                   ; nominator = 0 → curve table applies
$AA_EG_TYPE[C, C99] = 1                  ; Setpoint value coupling
$AA_EG_NUMERA[C, C99] = 1                ; numerator for coupling factorC99
$AA_EG_DENOM[C, C99] = 1                 ; denominator for coupling factorC99
$P_EG_BC[C] = "NOC"                      ; Block change criterion
$AA_EG_NUM_LA[C] = 2                     ; Number of leading axes
$AA_EG_AX[0, C] = Z                      ; name of leading axis Z
$AA_EG_AX[1, C] = C99                    ; name of leading axis C99
$AA_EG_SYN[C, Z] = <SynPosC_Z>           ; Synchronized position of leading axis Z
$AA_EG_SYN[C, C99] = <SynPosC_C99>       ; Synchronized position of leading axis C99
$AA_EG_SYNFA[C] = <SynPosC>              ; Synchronized position of leading axis C
```

Machine data

Extract from MD:

```
; ************** Channel 1
CHANDATA(1)
; ************** Axis 1, "X"
$MC_AXCONF_GEOAX_NAME_TAB[0] = "X"
$MC_AXCONF_CHANAX_NAME_TAB[0] = "X"
$MC_AXCONF_MACHAX_USED[0]=1
$MN_AXCONF_MACHAX_NAME_TAB[0] = "X1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX1] = 0
$MA_IS_ROT_AX[AX1] = FALSE
; *************** Axis 2, "Y"
$MC_AXCONF_GEOAX_NAME_TAB[1]="Y"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "Y"
$MC_AXCONF_MACHAX_USED[1] = 2
$MN_AXCONF_MACHAX_NAME_TAB[1] = "Y1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX2] = 0
$MA_IS_ROT_AX[AX2] = FALSE
```

```
; *************** Axis 3, "Z"
$MC_AXCONF_GEOAX_NAME_TAB[2] = "Z"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "Z"
$MC_AXCONF_MACHAX_USED[2]=3
$MN_AXCONF_MACHAX_NAME_TAB[2] = "Z1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX3] = 0
$MA_IS_ROT_AX[AX3] = FALSE
; *************** Axis 4, "A"
$MC_AXCONF_CHANAX_NAME_TAB[3] = "A"
$MC_AXCONF_MACHAX_USED[3]=4
$MN_AXCONF_MACHAX_NAME_TAB[3] = "A1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX4]=0
$MA_IS_ROT_AX[AX4] = TRUE
$MA_ROT_IS_MODULO[AX4] = TRUE
; *************** Axis 5, "B"
$MC_AXCONF_CHANAX_NAME_TAB[4] = "B"
$MC_AXCONF_MACHAX_USED[4]=5
$MC_SPIND_DEF_MASTER_SPIND = 1
$MN_AXCONF_MACHAX_NAME_TAB[4] = "B1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX5] = 1
$MA_IS_ROT_AX[AX5] = TRUE
$MA_ROT_IS_MODULO[AX5] = TRUE
; ************** Axis 6, "C"
$MC_AXCONF_CHANAX_NAME_TAB[5] = "C"
$MC_AXCONF_MACHAX_USED[5]=6
$MN_AXCONF_MACHAX_NAME_TAB[5] = "C1"
$MA_SPIND_ASSIGN_TO_MACHAX[AX6] = 0
$MA_IS_ROT_AX[AX6] = TRUE
$MA_ROT_IS_MODULO[AX6] = TRUE
; ************** Axis 10, "C99"
$MC_AXCONF_CHANAX_NAME_TAB[9] = "C99"
$MC_AXCONF_MACHAX_USED[9]=10
$MA_SPIND_ASSIGN_TO_MACHAX[AX10] = 0
$MA_IS_ROT_AX[AX10] = TRUE
$MA_ROT_IS_MODULO[AX10] = TRUE
```

10.5 Generic coupling

10.5.1 Brief description

10.5.1.1 Function

Function

"Generic Coupling" is a general coupling function, combining all coupling characteristics of existing coupling types (coupled motion, master value coupling, electronic gearbox and synchronous spindle).

The function allows flexible programming:

- Users can select the coupling properties required for their applications (building block principle).
- Each coupling property can be programmed individually.
- The coupling properties of a defined coupling (e.g. coupling factor) can be changed.
- Later use of additional coupling properties is possible.
- The coordinate reference system of the following axis (Base co-ordinate system or Machine co-ordinate system) is programmable.
- Certain coupling properties can also be programmed with synchronous actions.
  References:
  Function Manual, Synchronized Actions

Adaptive cycles

Previous coupling calls for coupled motion (TRAIL*), Master value coupling (LEAD*), Electronic Gearbox (EG*) and Synchronous spindle (COUP*) are still supported via adaptive cycles (see Section "Adaptive cycles (Page 560)").

10.5.1.2 Requirements

CP version

The generic coupling is available in a basic version and four optional versions:

- CP-STATIC
- CP-BASIC
- CP-COMFORT
- CP-EXPERT

This structure is based on the following considerations:

- Functional scope and required application knowledge increase from the basic version to the optional CP_EXPERT version.
- The number of required couplings (following axes, following spindles) and their properties are decisive in the selection of versions.
  - Example of simultaneous operation:
    If sequential operation of 1 x synchronous spindle pair for part transfer from the main to the supplementary spindle and the 1 x multi-edge turning is required, then the CP-BASIC option is suitable and sufficient. However, if it cannot be excluded that both operations can overlap (multi-edge turning running when part transfer is started), the CP-COMFORT option would be required.
  - Example of property:
    If a coupled axis grouping with a leading axis is required, the base version is sufficient. For coupled motion groups with two leading axes, one of the optional versions is required.
- Individual versions are independent of each other. They can be combined and can be activated simultaneously.

Table 10-2 Quantity structure, which is dependent on the execution of coupling modules that can be simultaneously activated

Type	CP versions allow one or more different CPSETTYPE coupling objects simultaneously:	Base version	CP-STATIC	CP-BASIC	CP-COMFORT	CP-EXPERT
A	Coupled motion	4	-	4	4	8
B	Synchronous spindle with |1|:1 coupling	-	1	-	-	-
C	o./u. Synchronous spindles, multi-edge turning o./u. Master value coupling / curve tables interpolation o./u. MCS coupling	-	-	1	4	8
D	o./u. Electronic gearbox "plain" o./u. Free generic coupling "basic"	-	-	-	1	8
E	o./u. Electronic gear o./u. Free generic coupling	-	-	-	-	5

o./u. stands for over/under

Table 10-3 Scaling of availability of coupling properties

Type A	Type B	Type C	Type D	Type E	
20	1	13	9	5	Maximum number of `CPSETTYPE`-related functionalities (per type)
TRAIL - coupled motion					
20	-	13	9	5	Maximum number of coupled motion groups with the following properties: → refer to `CPSETTYPE="TRAIL"`1)
1		2	2	2	Maximum number of master values
-		-	+	+	Coupling between a spindle and an axis

Type A	Type B	Type C	Type D	Type E	
+		+	+	+	Coupling between a rotary axis and a linear axis
-		+	+	+	From part program and synchronous actions
+		+	+	+	Superimposition / speed difference permitted
-		-	-	+	Cascading permitted
BCS		BCS / MCS	BCS / MCS	BCS / MCS	Coordinate reference (default): `CPFRS="BCS"`)
Synchronous spindle with |1|:1 coupling					
-	1	-	-	-	Maximum number of synchronous spindles / multi-edge turning with the following properties: → refer to `CPSETTYPE="COUP"`1)
	1				Maximum number of master values
	-				From part program and synchronous actions
	-				Superimposition / speed difference permitted
	-				Cascading permitted
	MCS				Coordinate reference fix (`CPFRS="MCS"`)
COUP - synchronous spindle/multi-edge turning					
-	-	13	9	5	Maximum number of synchronous spindles / multi-edge turning with the following properties: → refer to `CPSETTYPE="COUP"`1)
		1	1	1	Maximum number of master values
		-	-	-	From part program and synchronous actions
		+	+	+	Superimposition / speed difference permitted
		-	-	-	Cascading permitted
		MCS	MCS	MCS	Coordinate reference fix (`CPFRS="MCS"`)
LEAD - master value coupling / curve tables interpolation					
-	-	13	9	5	Maximum number of master value couplings / curve tables interpolation with the following properties: → refer to `CPSETTYPE="LEAD"`1)
		1	1	1	Maximum number of master values
		+	+	+	From part program and synchronous actions
		+	+	+	Superimposition / speed difference permitted
		-	-	+	Cascading permitted
		BCS / MCS	BCS / MCS	BCS / MCS	Coordinate reference (default): `CPFRS="BCS"`)
EG - electronic gearbox					
-	-	-	9	5	Maximum number of electronic gearboxes with the following properties: → refer to `CPSETTYPE="EG"`1)
			3	5	Maximum number of master values
			-	-	From part program and synchronous actions
			+	+	Superimposition / speed difference permitted
			-	+	Cascading permitted
			BCS / MCS	BCS / MCS	Coordinate reference (default): `CPFRS="BCS"`)

Type A	Type B	Type C	Type D	Type E	
			-	+	Non-linear coupling law (`CPLCTID`) permitted
CP - free generic coupling					
-	-	-	9	5	Maximum number of free generic couplings with the following properties: Default (corresponds to `CPSETTYPE="CP"`[1)])
			3	5	Maximum number of master values
			+	+	From part program and synchronous actions
			+	+	Superimposition / speed difference permitted
			-	+	Cascading permitted
			BCS / MCS	BCS / MCS	Coordinate reference (default): `CPFRS="BCS"`)
			-	+	Non-linear coupling law (`CPLCTID`) permitted
[1)] Refer to " Coupling types (CPSETTYPE) (Page 561) ".					

Note

Existing coupling options (Master value coupling, Electronic gearbox and Synchronous spindle) are not taken into consideration by generic coupling. Simultaneous operation of existing coupling options and generic coupling is only possible if the couplings refer to different axes/spindles.

Memory configuration

Memory space reserved in dynamic NC memory for generic coupling is defined in machine data:

MD18450 $MN_MM_NUM_CP_MODULES (maximum allowed number of CP coupling modules)

MD18452 $MN_MM_NUM_CP_MODUL_LEAD (maximum allowed number of CP master values)

Note

Recommendation: Expected maximum values, which can be expected simultaneously for this machine in its maximum configuration, should already be set during commissioning.

Hardware requirements

Systems with more than 6 axes are required when using the "CP-EXPERT" option.

10.5.2 Fundamentals

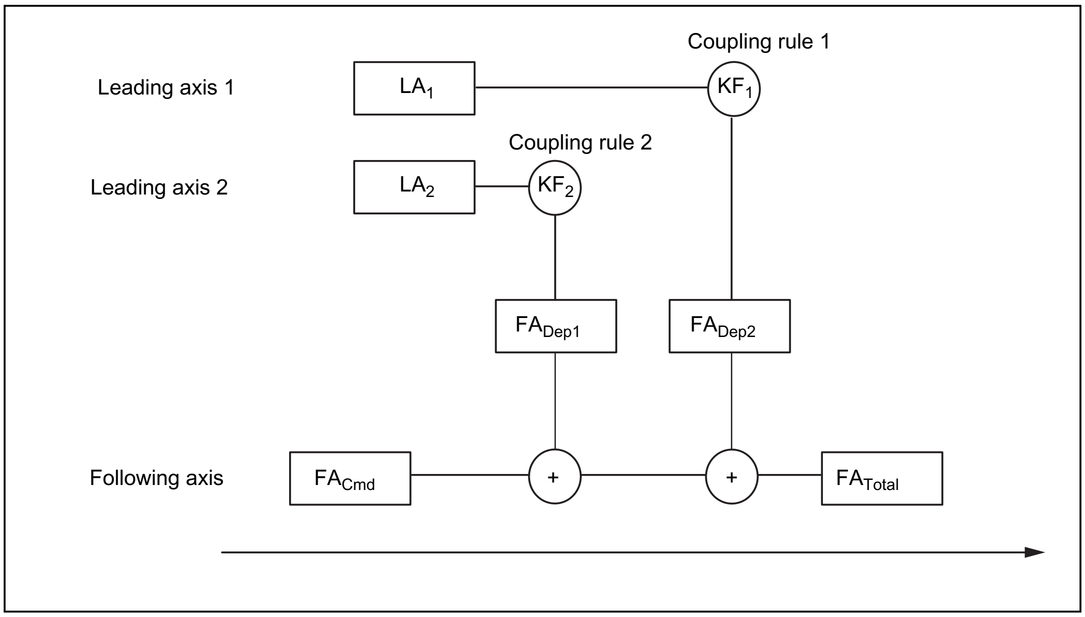
10.5.2.1 Coupling module

With the help of a coupling module, the motion of one axis (→ following axis) can be interpolated depending on other axes (→ leading axes).

Coupling rule

The relationships between leading axis/values and a following axis are defined by a coupling rule (coupling factor or curve table). The individual motion components from the individual leading axes/values have an additive effect.

The relationship is demonstrated by the following example (following axis with two leading axes):



FA_{Total}	Total setpoint value of the following axis
FA_{Cmd}	Setpoint value set in part program = independent motion component of the following axis
FA_{DEP1}	Dependent motion component of leading axis 1
FA_{DEP2}	Dependent motion component of leading axis 2
LA_1	Setpoint or actual value of the 1st leading axis
LA_2	Setpoint or actual value of the 2nd leading axis
$SynPosLA_1$	Synchronized position of the 1st leading axis
$SynPosLA_2$	Synchronized position of the 2nd leading axis
KF_1	Coupling factor of the 1st leading axis
KF_2	Coupling factor of the 2nd leading axis

The following axis position results from the overlay (summation) of the dependent motion components (FA_{DEP1} and FA_{DEP2}), which result from the individual coupling relationships to the leading axes, and of the independent motion component (FA_{Cmd}) of the following axis:

$FA_{Total} = FA_{Cmd} + FA_{DEP1} + FA_{DEP2}$

The motion components of the following axis are calculated as follows:

Dependent motion component of leading axis 1:	$FA_{DEP1} = (LA_1 - SynPosLA_1) * KF_1$
Dependent motion component of leading axis 2:	$FA_{DEP2} = (LA_2 - SynPosLA_2) * KF_2$
Independent motion component of the following axis:	FA_{Cmd}

Following axis overlay

The overlay of dependent and independent motion components of the following axis is called following axis overlay.

The independent motion component of the following axis can be programmed with the full range of available motion commands.

10.5.2.2 Keywords and coupling characteristics

Keywords

Programming is done via language commands, e.g. coupled motion with `TRAILON(X,Y,2)`. Keywords replace the language commands in generic coupling.

This has the following advantages:

- Coupling characteristics can be programmed individually (see following example).
- Programming of multiple couplings can be done in one block (since keywords do not require their own block).
  Advantage: Reduction of work off time

Example:

The properties set with the existing coupling call `TRAILON(X,Y,2)`(following axis, leading axis and coupling factor) are defined in the generic coupling with the following keywords:

```
CPON=(X1) CPLA[X1]=(X2) CPLNUM[X1,X2]=2
```

`CPON=(X1)`	Switch on coupling to following axis X1.
`CPLA[X1]=(X2)`	Define axis X2 as leading axis.
`CPLNUM[X1,X2]=2`	Set numerator of the coupling factor to 2.

Notation

In order to be uniquely assigned, keywords are furnished with the prefix "CP", for **C**ou**p**ling). Depending on meaning and application position, a third letter is used:

Keyword prefix	Meaning	Example
CP*	Describes the characteristics of the entire coupling.	CPON[1)]
CP**F***	Describes the characteristics of the following axis (**F**ollowing axis).	CPFPOS[1)]
CP**L***	Describes a property referring to the leading axis (**L**eading axis) or the coupling rule.	CPLON[1)], CPLNUM[1)]
CP**M***	Describes a property of the entire coupling for special states.	CPMRESET[1)]
[1)] For keyword meaning, please refer to the following table "Overview of all keywords and coupling characteristics".		

Overview of all keywords and coupling characteristics

The following table gives an overview of all keywords of the generic coupling and the programmable coupling characteristics:

Keyword	Coupling characteristics / meaning	Default setting (CPSET-TYPE="CP")
`CPDEF`	Creating a coupling module	
`CPDEL`	Deletion of a coupling module	
`CPLDEF`	Definition of a leading axis and creation of a coupling module	
`CPLDEL`	Deleting a leading axis of a coupling module	
`CPON`	Switching on a coupling module	
`CPOF`	Switching off a coupling module	
`CPLON`	Switching on a leading axis of a coupling module	
`CPLOF`	Switching off a leading axis of a coupling module	
`CPLNUM`	Numerator of the coupling factor	1.0
`CPLDEN`	Denominator of the coupling factor	1.0
`CPLCTID`	Number of curve table	Not set
`CPLSETVAL`	Coupling reference	CMDPOS
`CPFRS`	Co-ordinate reference system	BCS
`CPBC`	Block change criterion	NOC
`CPFPOS + CPON`	Synchronized position of the following axis when switching on	Not set
`CPLPOS + CPON`	Synchronized position of the leading axis when switching on	Not set
`CPFMSON`	Synchronization mode	CFAST
`CPFMON`	Behavior of the following axis at switching on	STOP

Keyword	Coupling characteristics / meaning	Default setting (CPSET-TYPE="CP")
`CPFMOF`	Behavior of the following axis at complete switch-off	STOP
`CPFPOS + CPOF`	Switch-off position of the following axis when switching off	Not set
`CPMRESET`	Coupling response to RESET	NONE
`CPMSTART`	Coupling behavior at part program start	NONE
`CPMPRT`	Coupling response when the part program starts under block search run via program test	NONE
`CPLINTR`	Offset value of the input value of a leading axis	0.0
`CPLINSC`	Scaling factor of the input value of a leading axis	1.0
`CPLOUTTR`	Offset value for the output value of a coupling	0.0
`CPLOUTSC`	Scaling factor for the output value of a coupling	1.0
`CPSYNCOP`	Threshold value of position synchronism "Coarse"	MD37200
`CPSYNFIP`	Threshold value of position synchronism "Fine"	MD37210
`CPSYNCOP2`	Second threshold value for the "Coarse" position synchronism	MD37202
`CPSYNFIP2`	Second threshold value for the "Fine" position synchronism	MD37212
`CPSYNCOV`	Threshold value of velocity synchronism "Coarse"	MD37220
`CPSYNFIV`	Threshold value of velocity synchronism "Fine"	MD37230
`CPMBRAKE`	Response of the following axis to certain stop signals and stop commands	1
`CPMVDI`	Response of the following axis to certain NC/PLC interface signals	0 (for bits 0 to 3, 5) 1 (for bits 4, 6)
`CPMALARM`	Suppression of special coupling-related alarm outputs	MD11410 MD11415
`CPSETTYPE`	Coupling type	CP

Note

Coupling characteristics, which are not explicitly programmed (in part program of synchronous actions), become effective with their default settings (see right hand column of the table).

Depending on the settings of the keyword `CPSETTYPE` instead of the default settings (`CPSETTYPE="CP"`) preset coupling characteristics can become effective (see Section "Coupling types (Page 561)").

10.5.2.3 System variables

The current state of a coupling characteristic set with a keyword, can be read and written to with the relevant system variable.

Note

When writing in the part program, PREPROCESSING STOP is generated.

Notation

The names of system variables are normally derived from the relevant keywords and a corresponding prefix.

The first letter of the prefix defines the access location when reading:

System variable prefix	Access location during read	Features
$**P**A_CP	Reading of channel referenced axis specific coupling characteristics in block preparation (**P**reparation)	Use in synchronous actions is not possible. Does not generate an implicit pre-processing stop.
$**A**A_CP	Reading of the current state of the coupling module (across channels).	Use in the part program and in syn-chronous actions is possible. Generates an implicit preprocess-ing stop when used in the part pro-gram.

Note

The preprocessing value of a $PA_CP.. CP system variable only differs from the values of the corresponding $AA_CP.. CP system variable during active part program processing.

At the end of the program or with abort, there is an appropriate synchronization of the preprocessing with the main run states.

System variable list

A list of all system variables which can be used in a generic coupling is contained in the data lists (see Section "System variables (Page 593)").

For a detailed description of system variables, refer to:
Reference:
Parameter Manual, System Variables

10.5.3 Creating/deleting coupling modules

10.5.3.1 Creating a coupling module (CPDEF)

An axial coupling module is created through the definition of the following axis.

Programming

Syntax:	`CPDEF=` (<following axis/spindle>)
Designation:	**C**ou**p**ling **Def**inition
Functionality:	Definition of a coupling module The coupling is not activated.
Following axis/ spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPDEF=(X2)`	`; A coupling module is created with axis X2 as following axis.`

Boundary conditions

- The maximum number of coupling modules is limited (see Section "Requirements (Page 510)").
- The application of `CPDEF` to an already created coupling module is possible and will not result in an alarm being generated.

10.5.3.2 Delete coupling module (CPDEL)

A coupling module created with `CPDEF` can be deleted with `CPDEL`.

Programming

Syntax:	`CPDEL=` (<following axis/spindle>)
Designation:	**C**ou**p**ling **Del**ete
Functionality:	Deletion of a coupling module. All leading axis modules are deleted with the coupling module and reserved memory is released.
Following axis/ spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPDEL=(X2)`	`; Deletion of the coupling module with following axis X2.`

Boundary conditions

- The switch command `CPDEL` results in a preprocessing stop with active coupling. Exception: `CPSETTYPE="COUP"` does not result in a preprocessing stop.
- Applying `CPDEL` to a coupling module active in the block preparation results in implicit deactivation of this coupling.
- Applying `CPDEL` to an undefined coupling module does not result in any action.

10.5.3.3 Defining leading axes (CPLDEF or CPDEF+CPLA)

The leading axes/spindles defined for a coupling can be programmed/created with the keyword `CPLDEF` or with the keyword `CPLA` in conjunction with `CPDEF`.

Programming with CPLDEF

Syntax:	`CPLDEF[FAx]=` (<leading axis/spindle>)
Designation:	**C**ou**p**ling **L**ead **A**xis **Def**inition
Functionality:	Definition of leading axis/spindle for following axis/spindle FAx. A leading axis/spindle module is created in the coupling module. If the coupling module of the following axis/spindle has not yet been created, the coupling module will be created implicitly.
Leading axis/ spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPLDEF[X2]=(X1)`	`; Definition of leading axis X1 for following axis X2.`

Programming with CPLA and CPDEF

Syntax:	`CPLA[FAx]=` (<leading axis/spindle>)
Designation:	**C**ou**p**ling **L**ead **A**xis

Functionality:	Definition of leading axis/spindle for following axis/spindle FAx.	
Leading axis/spindle:	Type:	AXIS
	Range of values:	All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPDEF=(X2) CPLA[X2]=(X1)`	`; Definition of leading axis X1 for following axis X2.`

Boundary conditions

- `CPLDEF` is only allowed in blocks without `CPDEF`/`CPON`/`CPOF`/`CPDEL`.
  (This limitation applies to the case where the keywords refer to the same coupling module.)
- The maximum number of leading axis modules per coupling module is limited (see Section "Requirements (Page 510)").
- Definition of leading axes on an already defined or active coupling module is possible. Any newly defined leading axes and their properties (e.g. coupling factor) are not active immediately. A corresponding switch-on command like (`CPON` or `CPLON`) is required.

10.5.3.4 Delete leading axes (CPLDEL or CPDEL+CPLA)

Defined leading axes can be deleted with `CPLDEL` or with `CPLA` in conjunction with `CPDEL`, i.e. removed from the coupling module.

Programming with CPLDEL

Syntax:	`CPLDEL[FAx]=` (<leading axis/spindle>)	
Designation:	**C**ou**p**ling **L**ead **A**xis **Del**ete	
Functionality:	Deletion of leading axis/spindle of following axis/spindle FAx. The leading axis/spindle module will be deleted and the corresponding memory will be released. If the coupling module does not have a leading axis/spindle any more, the coupling module will be deleted and the memory will be released.	
Leading axis/ spindle:	Type:	AXIS

Range of values:	All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPLDEL[X2]=(X1)`	`; Deletion of leading axis X1 of the coupling to following axis X2.`

Programming with CPLA and CPDEL

Syntax:	`CPLA[FAx]=` (<leading axis/spindle>)
Designation:	**C**ou**p**ling **L**ead **A**xis
Functionality:	Deleting a leading axis/spindle: The leading axis/spindle module will be deleted and the corresponding memory will be released. If the coupling module does not have a leading axis/spindle any more, the coupling module will be deleted and the memory will be released.
Leading axis/spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPDEL=(X2) CPLA[X2]=(X1)`	`; Deletion of leading axis X1 of the coupling to following axis X2.`

Boundary conditions

- `CPLDEF` is only allowed in blocks without `CPDEF`/`CPON`/`CPOF`/`CPDEL`.
  (This limitation applies to the case where the keywords refer to the same coupling module.)
- If an active leading axis is deleted, the coupling to this leading axis is implicitly deactivated.
- Deletion of the last leading axis results in the entire coupling module to be deleted.

10.5.4 Switching coupling on/off

10.5.4.1 Switching on a coupling module (CPON)

A defined coupling module is switched on with the switch command CPON.

Coupling characteristics like coupling reference can be programmed together with the switch on command (see Section "Programming coupling characteristics (Page 526)").

Without programming, a coupled motion group or a synchronous spindle pair becomes effective based on a setpoint coupling (default setting for CPLSETVAL) with the coupling rule 1:1 (default setting for CPLNUM/CPLDEN).

Programming

Syntax:	CPON= (<following axis/spindle>)
Designation:	**C**ou**p**ling **On**
Functionality:	Activate the coupling of the following axis to all defined leading axes.
Following axis/ spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
CPON=(X2)	; Activation of coupling of the following axis X2.

Boundary conditions

Application of CPON to an already active coupling results in a resynchronization. If applicable, changed coupling properties become effective as a result. Any lost synchronization (for example, following axis was in tracking mode) is restored.

10.5.4.2 Switch off coupling module (CPOF)

An activated coupling can be deactivated with the CPOF switching command. The deactivation, i.e. the switching off of the coupling to the leading axis, is performed in accordance with the set switch-off properties (see CPFMOF)

Programming

Syntax:	`CPOF=` (<Following axis / spindle>)
Designation:	**C**ou**p**ling **Of**f
Functionality:	Deactivate the coupling of the following axis to all defined leading axes.
Following axis/ spindle:	Type: AXIS
	Range of values: All defined axis and spindle names in the channel

Example:

Programming	Comment
CPOF=(X2)	; Deactivation of coupling of following axis X2.

Boundary conditions

- The switch command `CPOF` results in a preprocessing stop with active coupling.
  Exception: `CPSETTYPE="COUP"` does not result in a preprocessing stop
- A `CPOF` switching command on an already deactivated or deleted coupling module has no effect and is not executed.
- `CPOF` can be programmed in synchronous actions.

10.5.4.3 Switching on leading axes of a coupling module (CPLON)

`CPLON` activates the coupling of a leading axis to a following axis. If several leading axes are defined for a coupling module, they can be activated and deactivated separately with `CPLON`.

Programming

Syntax:	`CPLON[FAx]=` (<leading axis/spindle>)
Designation:	**C**ou**p**ling **L**ead Axis **On**
Functionality:	Activates the coupling of a leading axis/spindle to following axis/spindle FAx.
Leading axis/ spindle:	Type: AXIS

Range of values:	All defined axis and spindle names in the channel

Example:

Programming	Comment
`CPLON[X2]=(X1)`	`; The coupling of leading axis X1 to following axis X2 is activated.`

Boundary conditions

`CPON` can be programmed in synchronous actions.

10.5.4.4 Switching off leading axes of a coupling module (CPLOF)

`CPLOF` deactivates the coupling of a leading axis to a following axis. If several leading axes are defined for a coupling module, they can be deactivated separately with `CPLOF`.

Programming

Syntax:	`CPLOF[FAx]=` (<leading axis/spindle>)	
Identifiers:	**C**ou**p**ling **L**ead Axis **Of**f	
Functionality:	Deactivates the coupling of a leading axis/spindle to the following axis/spindle FAx.	
Leading axis/ spindle:	Type:	AXIS
	Range of values:	Axes of the channel

Example:

Programming	Comment
`CPLOF[X2]=(X1)`	`; The coupling of leading axis X1 to following axis X2 is deactivated.`

Constraints

`CPLOF` can be programmed in synchronous actions.

10.5.4.5 Implicit creation and deletion of coupling modules

Switch-on commands may also be used to create coupling modules (without prior definition with CPDEF).

Example

Programming	Comment
CPON=(X2) CPLA[X2]=(X1)	; Creates a coupling module for following axis X2 with leading axis X1 and activates the coupling module.
or	
CPLON[X2]=(X1)	
...	
CPOF=(X2)	; After its deactivation, the implicitly created coupling module is deleted.

Constraints

- Implicitly created coupling modules (via switch-on commands) are deleted once they are completely deactivated (CPOF).
  Advantage: Deleting them with CPDEL/CPLDEL is not necessary.
  Disadvantage (possibly): All coupling properties which were set with CPOF are lost.
- Implicitly created coupling modules can be transformed into explicit coupling modules with the following instruction CPDEF/CPLDEF. In this case CPOF does not delete the coupling module and the data is retained.

10.5.5 Programming coupling characteristics

10.5.5.1 Coupling rule (CPLNUM, CPLDEN, CPLCTID)

The functional relationship between the leading value and the following value is specified by a coupling rule for each leading axis. This functional relationship can be defined linear via a coupling factor or non-linear via a curve table. The following axis components calculated in this way from the individual leading values have an additive effect.

Programming: Coupling factor

When programming a coupling factor, a previously activated non-linear coupling relationship (curve table) is deactivated.

The coupling factor is programmed with numerator and denominator.

In the default state, i.e. without explicit programming after creation of a new coupling module, the numerator and denominator are each preset with 1.

If only the numerator is programmed, this is applied as the factor as the denominator is 1.

More exact linear relationships can be defined by programming numerators and denominators.

Numerator of the coupling factor

Syntax: `CPLNUM[FAx,LAx]=` <value>

Designation: **C**ou**pl**ing **L**ead **Num**erator

Functionality: Defines the numerator of the coupling factor for the coupling rule of the following axis/spindle FAx to the leading axis/spindle LAx.

Value: Type: REAL

Range of values: -2^{31} to +2^{31}

Default value: +1.0

Example:

Programming	Comment
`CPLNUM[X2,X1]=1.3`	`; The numerator of the coupling factor of the coupling of the following axis X2 to the leading axis X1 must be 1.3.`

Denominator of the coupling factor

Syntax: `CPLDEN[FAx,LAx]=` <value>

Designation: **C**ou**pl**ing **L**ead **Den**ominator

Functionality: Defines the denominator of the coupling factor for the coupling rule of the following axis/spindle FAx to the leading axis/spindle LAx.

Value: Type: REAL

Range of values: -2^{31} to +2^{31}

Default value: +1.0

Example:

Programming	Comment
`CPLDEN[X2,X1]=2`	`; The denominator of the coupling factor of the coupling of the following axis X2 to the leading axis X1 must be 2.`

Programming: Curve table

When programming a table number, a previously activated non-linear coupling relationship (coupling factor) is deactivated.

The leading axis specific coupling component for the leading value of the leading axis is calculated using the specified curve table.

Syntax:	`CPLCTID[FAx,LAx]=` <value>
Designation:	**C**ou**p**ling **L**ead **C**urve **T**able **Id**
Functionality:	Specifies the number of the curve table to be used to calculate how the leading axis/spindle must act on the following axis/spindle.
Value:	Type: INT
	Range of values: -2^{15} to $+2^{15}$

Example:

Programming	Comment
`CPLCTID[X2,X1]=5`	`; The leading axis specific coupling component of the coupling of the following axis X2 to the leading axis X1 is calculated with curve table No. 5.`

Supplementary conditions

- A coupling factor of zero (`CPLNUM=0`) is a permissible value. In this case, the leading axis/spindle does not provide a path component for the following axis/spindle, however, it remains a part of the coupling. Contrary to the switched-off state, the leading axis/spindle still has an influence on the following axis/spindle. This affects, for example, reactions to errors, limit switches and NC/PLC interface signals.
- `CPLDEN=0` is not a valid value and is rejected with alarm.
- `CPLNUM`, `CPLDEN` and `CPLCTID` can be programmed in synchronous actions.
- Availability of non-linear coupling relationships (`CPLCTID`) depends on selected options (see Section "Requirements (Page 510)").

10.5.5.2 Coupling relationship (CPLSETVAL)

The following value can be derived from either of the following:

- position setpoint of the leading axis
- speed setpoint of the leading axis
- position actual value of the leading axis

The following couplings can be programmed accordingly:

- Setpoint value coupling
- Speed coupling
- Actual value coupling

Programming

Syntax:	`CPLSETVAL[FAx,LAx]=` <value>
Identifiers:	**C**oup**l**ing **L**ead **Set Val**ue
Functionality:	Defines tapping of the leading axis/spindle LAx and the reaction point on the following axis/spindle FAx.
Coupling reference:	Type: STRING

Range of values:

"CMDPOS"	**Com**man**d**ed **Pos**ition	Setpoint value coupling
"CMDVEL"	**Com**man**d**ed **Vel**ocity	Speed coupling
"ACTPOS"	**Act**ual Value	Actual value coupling

Default value: "CMDPOS"

Example:

Programming	Comment
`CPLSETVAL[X2,X1]="CMDPOS"`	`; The coupling of following axis X2 to leading axis X1 is deducted from the setpoint.`

Constraints

For a coupling module speed coupling cannot be activated simultaneously with setpoint or actual value coupling of another leading axis.

10.5.5.3 Co-ordinate reference (CPFRS):

The co-ordinate reference of the following axis/spindle specifies in which co-ordinate reference system the coupling component resulting from the coupling is applied. in the base co-ordinate system or in the machine co-ordinate system.

It is further specified which co-ordinate reference the leading values of the leading axis spindle must have. When a transformation is active and when the machine co-ordinate system is specified as co-ordinate reference (`CPFRS="MCS"`), the initial transformation values are taken as leading values.

Programming

Syntax:	`CPFRS[FAx]=` (<co-ordinate reference>)
Identifiers:	**C**ou**p**ling **F**ollowing **R**elation **S**ystem
Functionality:	Defines the co-ordinate reference system for the coupling module of the following axis/spindle FAx.
Co-ordinate reference:	Type: STRING

Range of values:

”BCS”	**B**asis **C**o-ordinate **S**ystem	Basic Coordinate System
”MCS”	**M**achine **C**oordinate **S**ystem	Machine coordinate system

Default value: ”BCS”

Example:

Programming	Comment
`CPFRS[X2]="BCS"`	`; The base co-ordinate system is the co-ordinate reference for the coupling module with following axis X2.`

Constraints

- Co-ordinate reference has to be specified when creating a coupling module, else the default value is used. It is not possible to effect subsequent changes.
- Simultaneous active transformation and coupling via RESET is not supported.
  Solution: Switching off the coupling with `CPMRESET="OF"` with RESET switching it on again with `CPMSTART="ON"` in the part program.
- Simultaneous operation of the previous function "Axial Coupling in the Machine Co-ordinate System (MCS Coupling)" and the generic coupling is not supported.
- `CPFRS` is not available in the main run.

10.5.5.4 Block change behavior (CPBC)

The block change criterion can be used to specify under which conditions the block change with activated coupling is to be permitted in the processing of the part program. The status of the coupling influences the block change behavior. If the specified condition is not fulfilled, the block change is disabled. The block change criterion is only evaluated with an active coupling.

The block change criterion can be defined with the keyword `CPBC` or with the programming command `WAITC`. The instruction programmed last is valid.

Programming with PCBC

Syntax:	`CPBC[FAx]=` "<block change criterion>"
Identifiers:	**C**ou**p**ling **B**lock **C**hange Criterium
Functionality:	Defines block change criterion with active coupling.
Block change criterion:	Type: STRING

Range of values:

"NOC"	Block change is performed irrespective of the coupling status.
"IPOSTOP"	Block change is performed with setpoint synchronism.
"COARSE"	Block change is performed with actual value synchronism “coarse”.
"FINE"	Block change is performed with actual value synchronism “fine”.
Default value:	"NOC"

Example:

Programming	Comment
`CPBC[X2]="IPOSTOP"`	`; Block change during processing of the part program is done with setpoint synchronism (with active coupling to following axis X2).`

Programming with WATC

Syntax:	`WAITC(FAx1,BC)`
Identifiers:	**Wait** for **C**oupling Condition
Functionality:	Defines block change criterion with active coupling.
Parameter:	Fax: Designates the following axis and therefore the coupling module. BC: Defines the desired block change criterion.
FAx:	Type: STRING
	Range of values: Axes of the channel

BC:	Type: STRING	
	Range of values:	
	"NOC"	Block change is performed irrespective of the coupling status.
	"IPOSTOP"	Block change is performed with setpoint synchronism.
	"COARSE"	Block change is performed with actual value synchronism “coarse”.
	"FINE"	Block change is performed with actual value synchronism “fine”.
	Default value:	"NOC"

Example:

Programming	Comment
`WAITC(X2,"IPOSTOP")`	`; Block change during processing of the part program is done with setpoint synchronism (with active coupling to following axis X2).`

Constraints

`WAITC` can only occur singularly in a block, contrary to the keyword `CPBC`.

10.5.5.5 Synchronized position of the following axis when switching on (CPFPOS+CPON)

When switching on the coupling (`CPON`) approach of the following axis can be programmed for a specified synchronized position.

The synchronized position takes immediate effect at switch on. The total position, resulting from the synchronized position and the coupling rule, is approached according to the specified synchronization mode (`CPFMSON`), taking into account the dynamic response limits.

Programming

Syntax:	`CPON=FAx CPFPOS[FAx]=` <value>	
Designation:	**C**ou**p**ling **F**ollowing **Pos**ition	
Functionality:	Defines the synchronized position of the following axis when switching on. `AC`, `IC` and `GP` are possible in position specification.	
Value:	Type: REAL	
	Range of values:	All positions within the traverse range boundaries

Example:

Programming	Comment
CPON=X2 CPFPOS[X2]=100	; Activation of coupling to following axis X2. 100 is taken as synchronized position of following axis X2.

Supplementary conditions

- CPFPOS is only effective as synchronized position with the switch-on command CPON/ CPLON.
  The switch-off command CPOF evaluates CPFPOS as switch off position (see Section "Following Axis Position on Switch off (Page 538)").
- CPFPOS without switch-on command results in an alarm.
- If the synchronized position of the following axis is not set during switch on, then the current position of the following axis takes effect as synchronized position.
  The program instruction IC can be used to move the current position.
- The position specification is specified in the configured basic system independently of the programmable dimensions(G70/G71).

Part program section (Example)

Programming	Comment
CPON=(X2) CPFPOS[X2]=100	; Activation of coupling to following axis X2. 100 is taken as synchronized position of the following axis.
...	
G00 X2=123	; Following axis X2 is traversed to position 123.
CPON=(X2)	; The current position (=123) is taken as synchronized position of the following axis. The previously active synchronized position 100 becomes effective.

10.5.5.6 Synchronized position of the leading axis when switching on (CPLPOS)

The current leading axis position, taken as leading value, can be offset. The synchronized position of the leading axis therefore defines the zero point of the input variable.

Programming

Syntax:	CPLPOS[FAx,LAx]= <value>
Identifiers:	**C**oupling **L**ead **Pos**ition

Functionality:	Defines the synchronized position of the following axis at switch on. Only `AC` is possible in the position specification.
Value:	Type: REAL
	Range of values: All position within the traverse range boundaries

Example:

Programming	Comment
`CPLPOS[X2,X1]=200`	`; 200 is taken as synchronized position of the leading ax-is X1 of the coupling to following axis X2.`

Constraints

- `CPFPOS` can only set with the switch-on command `CPON` / `CPLON`. `CPFPOS` without switch-on command results in an alarm.
- If the synchronized position of the leading axis is not set with the switch on command (`CPON`), then the current position of the leading axis takes effect as synchronized position and therefore as zero point of the input variable.
- The position specification is specified in the configured basic system independently of the programmable dimensions(`G70` / `G71`).

Part program section (Example)

Programming	Comment
`CPON=(X2) CPFPOS[X2]=100 CPLPOS[X2,X1]=200`	`; Activation of coupling to following axis X2. 100 is taken as synchronized position of fol-lowing axis and 200 for leading axis X1.`
`...`	
`N20 X1=280 F1000`	`; Leading axis X1 is traversed to position 280.`
`CPON=(X2)`	`; The current position X1=280 is taken as synchronized position of the leading axis. The previously active synchronized position of the leading axis (200) becomes ineffective.`

10.5.5.7 Synchronization mode (CPFMSON)

Synchronization mode determines synchronization behavior during switch-on of the coupling.

Programming

Syntax: `CPFMSON[FAx]=` "<synchronization mode>"

Identifiers:	**C**ou**p**ling **F**ollowing **M**ode **S**trategy **O**n
Functionality:	Determines the synchronization mode during coupling.
Synchronization mode:	Type: STRING

Range of values:

"CFAST"	**C**losed Coupling **F**ast	The coupling is closed time-optimized.
"CCOARSE"	**C**losed **I**f Gab **C**oarse	The coupling is only closed when the following axis position, required according to the coupling rule, is in the range of the current following axis position.
"NTGT"	**N**ext **T**ooth **G**ap **T**ime Optimized	The next tooth gap is approached time-optimized.
"NTGP"	**N**ext **T**ooth **G**ap **P**ath Optimized	The next tooth gap is approached path-optimized.
"NRGT"	**N**ext **R**atio **G**ap **T**ime Optimized	The next segment is approached in a time-optimized manner, in accordance with the ratio of the number of gears to the number of teeth.
"NRGP"	**N**ext **R**atio **G**ap **P**ath Optimized	The next segment is approached in a path-optimized manner, in accordance with the ratio of the number of gears to the number of teeth.
"ACN"	**A**bsolute **C**o-ordinate **N**egative	For rotary axes only! The rotary axis traverses towards the synchronized position in the **negative** axis direction. Synchronization is effected immediately.
"ACP"	**A**bsolute **C**oordinate **P**ositive	For rotary axes only! The rotary axis traverses to the synchronized position in the **positive** axis direction. Synchronization is effected immediately.
"DCT"	**D**irect **C**o-ordinate **T**ime Optimized	For rotary axes only! The rotary axis traverses to the programmed synchronized position in **time-optimized** fashion. Synchronization is effected immediately.

"DCP"	**D**irect **C**o-ordinate **P**ath Optimized	For rotary axes only! The rotary axis traverses to the programmed synchronized position in **path-optimized** fashion. Synchronization is effected immediately.

Default value: "CFAST"

Example:

Programming	**Comment**
`CPFMSON[X2]="CFAST"`	`; CFAST is taken as synchronization mode of the coupling to following axis X2.`

10.5.5.8 Behavior of the following axis at switch-on (CPFMON)

The behavior of the following axis/spindle during switch-on of the coupling can be programmed with the keyword `CPFMON`.

Programming

Syntax:	`CPFMON[FAx]=` "<block change criterion>"
Identifiers:	**C**ou**p**ling **F**ollowing **M**ode **On**
Functionality:	Defines the behavior of the following axis/spindle during switch-on of the coupling.
Poweron response:	Type: STRING

Range of values:

"STOP"	**Stop**	For spindles only! An active motion of the following spindle is stopped before switch-on.
"CONT"	**Cont**inue	For spindles and main traverse axes only! The current motion of the following axis/spindle is taken over into the coupling as start motion.

"ADD"	**Add**itional	For spindles only! The motion components of the coupling operate in addition to the currently overlaid motion, i.e. the current motion of the following axis/spindle is retained as overlaid motion.

Default value: "STOP"

Example:

Programming	Comment
`CPFMON[X2]="CONT"`	`; The current motion of following axis X2 is taken over as start motion.`

10.5.5.9 Behavior of the following axis at switch-off (CPFMOF)

The behavior of the following axis/spindle during complete switch-off of an active coupling can be programmed with the keyword `CPFMOF`.

Programming

Syntax:	`CPFMOF[FAx]=` "<switch-off behavior>"
Identifiers:	**C**ou**p**ling **F**ollowing **M**ode **Of**f
Functionality:	Defines the behavior of the following axis/spindle during complete switch-off of the coupling.
Switch-off response:	Type: STRING

Range of values:

"STOP"	**Stop**	Stop of a following axis/spindle. An active overlaid motion is also braked to standstill. Then the coupling is opened, (deactivated).
"CONT"	**Cont**inue	For spindles and main traverse axes only! The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation.

Default value: "STOP"

Example:

Programming	Comment
`CPFMOF[S2]="CONT"`	`; The following spindle S2 continues to traverse at the speed that was applied at the instant of deactivation.`

10.5.5.10 Position of the following axis when switching off (CPFPOS+CPOF)

When switching off a coupling (`CPOF`) traversing to a certain position can be requested for the following axis.

Programming

Syntax:	`CPOF=(FAx) CPFPOS[FAx]=` <value>
Functionality:	Defines the switch-off position of the following axis FAx.
Value:	Type: REAL
	Range of values: All positions within the traverse range boundaries

Example:

Programming	Comment
`CPOF=(X2) CPFPOS[X2]=100`	`; Deactivation of coupling to following axis X2. 100 is approached as switch-off position of the fol-lowing axis.`

Supplementary conditions

- `CPFPOS` is only effective as switch-off position with the switch-off command `CPOF`. The switch command `CPON` evaluates `CPFPOS` as switch-on position (see Section "Synchronized Position of the Following Axis on Switch-on (Page 532)").
- The setting of a switch-off position is only permitted with the switch-off mode: `CPFMOF=STOP`
- Switch-off position is approached with maximum dynamics.
- Block change behavior depends on parameterization of the keyword `CPBC`.

10.5.5.11 Condition at RESET (CPMRESET)

With RESET, the coupling can be activated, deactivated or the current status can be retained. The behavior can be set separately for each coupling module.

Programming

Syntax:	`CPMRESET[FAx]=` "<Reset behavior>"
Identifiers:	**C**ou**p**ling **M**ode **RESET**
Functionality:	Defines the behavior of a coupling at RESET.
Reset response:	Type: STRING

Range of values:

"NONE"	The current state of the coupling is retained.
"ON"	When the appropriate coupling module is created, the coupling is switched on. All defined leading axis relationships are activated. This is also performed when all or parts of these leading axis relationships are active, i.e. resynchronization is performed even with a completely activated coupling.
"OF"	An active overlaid motion is also braked to standstill. The coupling is then deactivated. When the relevant coupling module was created without an explicit definition (`CPDEF`), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
"OFC"	Possible only in spindles! The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. The coupling is switched off. When the relevant coupling module was created without an explicit definition (`CPDEF`), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
"DEL"	An active overlaid motion is also braked to standstill. The coupling is then deactivated and then deleted.
"DELC"	Possible only in spindles! The following spindle continues to traverse at the speed/velocity that applied at the instant of deactivation. The coupling is deactivated and then deleted.

Default value: "NONE"

Example:

Programming	**Comment**
`CPMRESET[X2]="DEL"`	`; On RESET the coupling to following axis X2 is deactivated and then deleted.`

Constraints

- The coupling characteristics set with `CPMRESET` is retained until the coupling module is deleted with (`CPDEL`).
- For the coupling type (`CPSETTYPE="TRAIL"`, `"LEAD"`, `"EG"` or `"COUP"`) the response is defined by the following machine data during RESET:
  MD20110 $MC_RESET_MODE_MASK (definition of initial control system settings after RESET/TP-End)
  → See Section "Defaults" in " Coupling Types (CPSETTYPE) (Page 561) ".

10.5.5.12 Condition at parts program start (CPMSTART)

At part program start the coupling can be activated, deactivated or the current status can be retained. The behavior can be set separately for each coupling module.

Programming

Syntax:	`CPMSTART[FAx]=` <value>
Identifiers:	**C**ou**p**ling **M**ode **Start**
Functionality:	Defines the behavior of a coupling at part program start.
Value:	Type: STRING

Range of values:

"NONE"	The current state of the coupling is retained.
"ON"	When the appropriate coupling module is created, the coupling is switched on. All defined leading axis relationships are activated. This is also performed when all or parts of these leading axis relationships are active, i.e. resynchronization is performed even with a completely activated coupling.
"OF"	The coupling is switched off. When the relevant coupling module was created without an explicit definition (`CPDEF`), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
"DEL"	The coupling is deactivated and then deleted.

Default value: "NONE"

Example:

Programming	Comment
`CPMSTART[X2]="ON"`	`; At part program start, coupling to following axis X2 is switched on.`

Constraints

- The coupling characteristics set with `CPMSTART` are retained until the coupling module is deleted with (`CPDEL`).
- For the set coupling type (`CPSETTYPE="TRAIL"`, `"LEAD"`, `"EG"` or `"COUP"`), the response is defined by the following machine data during part program start:
  MD20112 $MC_START_MODE_MASK (Definition of the control default settings in case of NC START)
  → See Section "Defaults" in " Coupling Types (CPSETTYPE) (Page 561) ".

10.5.5.13 Status during part program start in search run via program test (CPMPRT)

When starting the part program under block search run via program test (SERUPRO), the coupling can be activated, deactivated or the current status can be retained. The behavior can be set separately for each coupling module.

Programming

Syntax:	`CPMPRT[FAx]=` <value>
Designation:	**C**ou**p**ling **M**ode **Pr**ogram **T**est
Functionality:	Defines the behavior of a coupling when the part program starts under block search run via program test.
Value:	Type: STRING

Value range:

"NONE"	The current state of the coupling is retained.
"ON"	When the appropriate coupling module is created, the coupling is switched on. All defined leading axis relationships are activated. This is also performed when all or parts of these leading axis relationships are active, i.e. resynchronization is performed even with a completely activated coupling.
"OFF"	The coupling is switched off. When the relevant coupling module was created without an explicit definition (`CPDEF`), the coupling module is deleted. Otherwise it is retained, i.e. it can still be used.
"DEL"	The coupling is deactivated and then deleted.

Default value: "NONE"

Example:

Programming	Comment
`CPMPRT[X2]="ON"`	`; When the part program starts under block search run via program test, the coupling to following axis X2 is switched on.`

Supplementary conditions

- The coupling characteristics set with `CPMPRT` is retained until the coupling module is deleted with (`CPDEL`).
- If `CPMPRT="NONE"` is set, then the response when the part program starts under block search run via program test (SERUPRO) is defined by `CPMSTART`.
- For the set coupling type (`CPSETTYPE="TRAIL"`, `"LEAD"`, `"EG"` or `"COUP"`), the response is defined by the following machine data when the part program starts during block search run via program test:
  MD22620 $MN_START_MODE_MASK_PRT(definition of initial control system settings with special start)
  MD22621 $MC_ENABLE_START_MODE_MASK_PRT (activation of MD22620)
  MD20112 $MC_START_MODE_MASK (definition of initial control settings for NC-START)
  → See Section "Defaults" in " Coupling types (CPSETTYPE) (Page 561) ".

10.5.5.14 Offset / scaling (CPLINTR, CPLINSC, CPLOUTTR, CPLOUTSC)

An existing coupling relationship between a following axis and a leading axis can be scaled and offset.

The effect of these functions on the total setpoint value of the following axes can be viewed from the following formula:

$$FA_{Total} = FA_{cmd} + \{ transOut_1 + scaleOut_1 * [(LA_1 - SynPosLA_1) * scaleIn_1 + transIn_1] * KF_1 \} + \{ transOut_2 + scaleOut_2 * [(LA2 - SynPosLA_2) * scaleIn_2 + transIn_2] * KF_2 \}$$

FA_{Total}	Total setpoint value of the following axis
FA_{Cmd}	Setpoint set in the part program
$LA_{1 / 2}$	Setpoint or actual value of the 1st or 2nd leading axis/value
$SynPosLA_{1 / 2}$	Synchronized position of the 1st or 2nd leading axis/value
$scaleIn_{1 / 2}$	Scaling factor of the 1st or 2nd lead value
$transIn_{1 / 2}$	Offset of the 1st or 2nd lead value
$KF_{1 / 2}$	Coupling factor of the 1st or 2nd leading axis/value
$scaleOut_{1 / 2}$	Scaling factor of the 1st or 2nd output value
$transOut_{1 / 2}$	Offset of the 1st or 2nd output value

Note

The scaling and offset values can be defined for each leading axis.

Programming

Offset of the input value

Syntax:	`CPLINTR[FAx,LAx]=` <value>
Designation:	**C**oup**l**ing **L**ead **In Tr**anslation Displacement
Functionality:	Defines the offset value for the input value of the LAx leading axis.
Value:	Type: REAL
	Default value: 0

Example:

Programming	Comment
`CPLINTR[X2,X1]=-50`	`; The input value of the leading axis X1 is moved in the negative direction by the value 50.`

Scaling the input value

Syntax:	`CPLINSC[FAx,LAx]=` <value>
Designation:	**C**oup**l**ing **L**ead **In Sc**ale Factor
Functionality:	Defines the scaling factor for the input value of the LAx leading axis.
Value:	Type: REAL
	Default value: 1

Example:

Programming	Comment
`CPLINSC[X2,X1]=0.5`	`; The input value of the leading axis X1 is multiplied with the factor 0.5.`

Offset of the output value

Syntax:	`CPLOUTTR[FAx,LAx]=` <value>
Designation:	**C**oup**l**ing **L**ead **Out Tr**anslation Displacement
Functionality:	Defines the offset value for the output value of coupling the following axis FAx to leading axis LAx.
Value:	Type: REAL

Default value: 0

Example:

Programming	**Comment**
`CPLOUTTR[X2,X1]=100`	`; The output value of the coupling of the following axis X2 with leading axis X1 is displaced by the value 100 in the positive direction.`

Scaling of the output value

Syntax:	`CPLOUTSC[FAx,LAx]=` <value>
Designation:	**C**ou**p**ling **L**ead **Out** **Sc**ale Factor
Functionality:	Defines the scaling factor for the output value of coupling the following axis FAx with leading axis LAx.
Value:	Type: REAL
	Default value: 1

Example:

Programming	**Comment**
`CPLOUTSC[X2,X1]=3`	`The output value of the coupling of the following axis X2 with leading axis X1 is multiplied with factor 3.`

Note

The following setting data used in the existing coupling type "Master value coupling" is considered in generic coupling independently of the set coupling type (`CPSETTYPE`):

SD43102 $SA_LEAD_OFFSET_IN_POS[FAx] (offset of master value)

SD43104 $SA_LEAD_SCALE_IN_POS[FAx] (scaling of master value)

SD43106 $SA_LEAD_OFFSET_OUT_POS[FAx] (offset of function value of the curve table)

SD43108 $SA_LEAD_SCALE_OUT_POS[FAx] (scaling of function value of the curve table)

These setting data have the following effect:

- on all leading axes that are coupled with the following axis via a curve table. This must be taken into account for couplings with more than one leading axis!
- in addition to the CP key words `CPLINTR`, `CPLINSC`, `CPLOUTTR` and `CPLOUTSC`.

10.5.5.15 Synchronism monitoring stage 1 (CPSYNCOP, CPSYNFIP, CPSYNCOV, CPSYNFIV)

Synchronism monitoring stage 1

In each interpolator clock cycle, the synchronous operation of the coupling group is monitored – both on the setpoint and actual value sides. The synchronous operation monitoring responds as soon as the **synchronous operation difference** (the difference between the setpoint or actual value of the following axis and the value calculated from the setpoints or actual values of the leading axes according to the coupling rule) reaches one of the following programmed threshold values:

- At setpoint / actual value coupling (see "Coupling relationship (CPLSETVAL) (Page 528)"):
  - "Coarse" position synchronous operation threshold value
  - "Fine" position synchronous operation threshold value
- For velocity coupling (see "Coupling relationship (CPLSETVAL) (Page 528)"):
  - Threshold value of "Coarse" speed synchronous operation
  - Threshold value of "Fine" speed synchronous operation

The actual synchronous operation difference can be read out with the following CP system variables:

System variable	Meaning
$AA_SYNCDIFF [FAx]	Synchronous operation difference of the setpoint
$VA_SYNCDIFF [FAx]	Synchronous operation difference of the actual value

Note

Synchronous operation differences are signed and enable the advance or delay of the following axis to be determined.

Status of the coupling during synchronous operation

State	Description
Not synchronized	Provided the synchronous operation difference is greater than the threshold value for position "coarse" synchronous operation or "coarse" speed synchronous operation, the coupled group is designated as non-synchronous.
"Coarse" synchronous operation reached	The synchronous operation difference has reached the threshold value for "coarse" position synchronous operation or "coarse" speed synchronous operation. In case the actual value-based synchronous operation difference, the following NC/PLC-interface signal is set: DB31, ... DBX98.1 (coarse synchronous operation)
"Fine" synchronous operation reached	The synchronous operation difference has reached the threshold value for "fine" position synchronous operation or "fine" speed synchronous operation. In case the actual value-based synchronous operation difference, the following NC/PLC-interface signal is set: DB31, ... DBX98.0 (fine synchronous operation)

The status of the coupling for the synchronous operation can be read with the following system variables:

System variable	Meaning	
$AA_SYNC [FAx]	State of the coupling	
	Value	**State**
	0	Not synchronized
	1	"Coarse" synchronous operation reached
	2	"Fine" synchronous operation reached

Signal reaction

Synchronous operation signals themselves do not stop the involved axes, but can release them via synchronized action or NC/PLC interface signals (see Section "R3: Extended stop and retract (Page 599)").

Configuration

The threshold values for the first stage of the synchronous operation monitoring will be adjusted:

- For setpoint / actual value coupling in the machine data:
  - MD37200 $MA_COUPLE_POS_TOL_COARSE (threshold value for "coarse synchronism")
  - MD37210 $MA_COUPLE_POS_TOL_FINE (threshold value for "fine synchronism")
- For speed coupling in the machine data:
  - MD37220 $MA_COUPLE_VELO_TOL_COARSE ("coarse" speed tolerance)
  - MD37230 $MA_COUPLE_VELO_TOL_FINE ("fine" speed tolerance)

Programming

CP keywords can also be used to program the threshold values for the first stage of the synchronous operation monitoring:

"Coarse" position synchronous operation threshold value

Syntax:	`CPSYNCOP[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Co**arse **P**osition
Functionality:	Defines the threshold value for the "Coarse' position synchronous operation.
Value:	Type: REAL
	The default value corresponds to the setting in the machine data: MD37200 $MA_COUPLE_POS_TOL_COARSE [FAx]

"Fine" position synchronous operation threshold value

Syntax:	`CPSYNFIP[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Fi**ne **P**osition
Functionality:	Defines the threshold value for the "Fine" position synchronous operation.
Value:	Type: REAL
	The default value corresponds to the setting in the machine data: MD37210 $MA_COUPLE_POS_TOL_FINE [FAx]

Threshold value of "Coarse" speed synchronous operation

Syntax:	`CPSYNCOV[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Co**arse **V**elocity
Functionality:	Defines the threshold value for the "Coarse" speed synchronous operation.
Value:	Type: REAL The default value corresponds to the setting in the machine data: MD37220 $MA_COUPLE_VELO_TOL_COARSE [FAx]

Threshold value of "Fine" speed synchronous operation

Syntax:	`CPSYNFIV[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Fi**ne **V**elocity
Functionality:	Defines the threshold value for the "Fine" velocity synchronous operation.
Value:	Type: REAL The default value corresponds to the setting in the machine data: MD37230 $MA_COUPLE_VELO_TOL_FINE [FAx]

Example

Program code	**Comment**
`CPDEF=(S2) CPLA[S2]=(S1)`	`; Definition of a spindle coupling: Leading spindle S1 to following spindle S2`
`CPON=(S2) CPSYNCOP[S2]=0.5 CPSYNFIP[S2]=0.25`	`; Activation of the coupling with following spindle S2. The threshold values for the position synchronous operation are set to 0.5 ("coarse") and 0.25 ("fine").`
`...`	

Supplementary conditions

- When considering the synchronous operation difference, an active coupling cascade is not taken into account. This means: if in the considered coupling module, the leading axis is a following axis in another coupling module, the current actual or setpoint position is still used as input variable for the calculation of the synchronous operation difference. This synchronous operation difference therefore does not show the total synchronous operation error of the cascade.
- If the leading axis is not a real axis, but a simulated axis, then the actual value is not available for the synchronous operation monitoring of the actual value. In this case, the modeled actual values (according to the machine data setting) are used.

10.5.5.16 Synchronous operation monitoring stage 2 (CPSYNCOP2, CPSYNFIP2)

Synchronism monitoring stage 2

For active CP position coupling (setpoint or actual value coupling, see "Coupling relationship (CPLSETVAL) (Page 528)"), after reaching the "COARSE"/"FINE" block change criterion (see "Block change behavior (CPBC) (Page 530)"), the second stage of the synchronous operation monitoring can be used to monitor that one of the threshold values of the first stage is maintained on the actual value side, independent of the synchronism tolerance.

The following two threshold values must configured or programmed for the second stage of synchronous operation monitoring:

- "Coarse" 2 position synchronous operation threshold value
- "Fine" 2 position synchronous operation threshold value

Technical background

As part of the "synchronous operation reached" and "exit synchronous window" monitoring functions, for a correct assessment of any problems that arise during the multi-edge turning (surface, waste, etc.) and sometimes also for oscillating synchronous spindle it is often very useful to define a tolerance range independent of the coarse/fine synchronous operation tolerances of the first stage of the synchronous operation monitoring (similar to exact stop and standstill monitoring for axes) in order to obtain a configurable error message or warning.

Configuration

The threshold values for the second stage of the synchronous operation monitoring are set in the machine data:

MD37202 $MA_COUPLE_POS_TOL_COARSE_2 (second threshold value for "coarse synchronous operation")

MD37212 $MA_COUPLE_POS_TOL_FINE_2 (second threshold value for "fine synchronous operation")

Note

If the appropriate threshold value = 0, the associated monitoring is inactive. This is also the default value so that the compatibility with older software versions is retained.

Programming

CP keywords can also be used to program the threshold values for the second stage of the synchronous operation monitoring:

"Coarse" 2 position synchronous operation threshold value

Syntax:	`CPSYNCOP2[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Co**arse **P**osition **2**
Functionality:	Defines the second threshold value for the "Coarse' position synchronous operation.
Value:	Type: REAL The default value corresponds to the setting in the machine data: MD37202 $MA_COUPLE_POS_TOL_COARSE_2 [FAx]

"Fine" 2 position synchronous operation threshold value

Syntax:	`CPSYNFIP2[FAx]=` <value>
Designation:	**C**ou**p**ling **Syn**chronous Difference **Fi**ne **P**osition **2**
Functionality:	Defines the second threshold value for the "Fine" position synchronous operation.
Value:	Type: REAL The default value corresponds to the setting in the machine data: MD37212 $MA_COUPLE_POS_TOL_FINE_2 [FAx]

The programming applies similarly to the general CP behavior in the part program only with the next switching command; for synchronized actions, immediately.

Sequence

Starting

The second stage of the synchronous operation monitoring function starts with active coupling as soon as the following conditions are fulfilled:

- The setpoint synchronous operation is reached:
  DB31, ... DBX99.4 (synchronization running) = 0
- The actual value related "coarse"/"fine" synchronous operation of the tolerance of the first stage of the synchronous operation monitoring (see "Synchronism monitoring stage 1 (CPSYNCOP, CPSYNFIP, CPSYNCOV, CPSYNFIV) (Page 545)") is reached:
  DB31, ... DBX98.1 (coarse synchronous operation) = 1 / DB31, ... DBX98.0 (fine synchronous operation) = 1

Monitor

As long as the synchronism difference of the actual the threshold values for "fine" 2 position synchronous operation and "coarse" 2 position synchronous operation are not exceeded, the following NC/PLC interface signals are set:

DB31, ... DBX103.4 (synchronous operation 2 fine)

DB31, ... DBX103.5 (synchronous operation 2 coarse)

If, due to temporary overload in machining process (e.g. infeed feedrate for multi-edge machining too high), the following axis/spindle can no longer follow the specifications of the leading axis(n)/spindle(s) and the deviation is greater than the set tolerance, the violation of the "fine"/"coarse" tolerance displays a display alarm that can be deleted:

Alarm 22026 "Channel %1 Block %2 Following axis/spindle %3 Synchronism (2): Coarse tolerance exceeded"

Alarm 22025 "Channel %1 Block %2 Following axis/spindle %3 Synchronism (2): Fine tolerance exceeded"

This does not interrupt the machining.

Note

Both alarms can also occur simultaneously.

Exit

The second stage of the synchronous operation monitoring ends in the following cases:

- Deactivation of the coupling (for the power-down command: CPOF, CPDEL, CPLOF, CPLDEL)
- New synchronization request by:
  - CPON / CPLON / CPDO / CPLOF
  - DB31, ... DBX31.4 (synchronize following spindle) = 1
  - Internal synchronization requests

  If, in these cases, the start conditions are satisfied again, the monitoring will be restarted.

- For coupled block changes (CPLNUM, CPLDEN, CPLCTID) in synchronized actions
- Resetting the setpoint synchronous operation because of missing enable signals for the following spindle (emergency stop, alarm responses)

The DB31, ... DBX103.4/5 signals are reset when the monitoring is ended.

Boundary conditions

Exclusion conditions

No monitoring is performed in the following cases:

- MD37202 or MD37212 = 0.
- Speed coupling is active (`CPLSETVAL="CMDVEL"`).
- DB31, ... DBX31.5 (inhibit synchronization of the FS) = 1
- Rapid stop of the following axis/spindle or one of the active leading axes/spindles.
- SERUPRO or block search is active.
- For channel-related run-in when the following axis/spindle or an active leading axis/spindle does not really move.
- DB31, ... DBX63.3 (axis/spindle disable active) = 1 for the following axis/spindle or for one of the active leading axes/spindles.

"Track the synchronous operation deviation" is active

Monitoring is suspended while the "track the synchronous operation deviation" function is active (DB31, ... DBX31.6 = 1; see "Tracking the deviation from synchronism (Page 576)").

Synchronous operation monitoring level 2 for the traditional coupling types

For the traditional coupled motion, master value coupling, electronic gearbox and synchronous spindle coupling types, the "stage 2 synchronous operation monitoring" is available only with the CP adaptive cycles (see "Adaptive cycles (Page 560) ").

Example

Program code	**Comment**
`G0 Z-300 X50`	
`CPDEF=(S2) CPLA=(S1) CPLNUM=2 CPLDEN=1 CPBC="FINE"`	`; Gear ratio in accordance with tool.`
`M3 S2000`	
`CPON=(S2) CPLA=(S1) CPSYNCOP2[X]=1.6 CPSYNFIP2[X]=0.8`	`; Block change is performed on reaching "fine synchronous operation".`
	`; Synchronous operation monitoring stage 2 is also activa-ted`
	`; (tolerances: coarse 1.6, fine 0.8).`
`X20`	
`G1 Z-200`	`; Machining block, material removal.` `; Synchronous operation determines the quality.`
`CPOF=(S2)`	`; Deselect coupling, deactivate the monitoring function.`

10.5.5.17 Reaction to stop signals and commands (CPMBRAKE)

The response of the following axis to certain stop signals and commands can be defined with the CP keyword `CPMBRAKE`.

Programming

Syntax:	`CPMBRAKE[FAx]=` <value>
Designation:	**C**ou**p**ling **M**ode **Brake**

Functionality: `CPMBRAKE` is a bit-coded CP keyword that defines the braking behavior of the following axis FAx for the following events:

Bit	Event
0	• NST DB31, ... DBX4.3 (feed stop / spindle stop) present or • CP SW limit stop is set (see "CP SW limit monitoring (Page 570)") **Note:** Bit 0 is significant only for a "Freely programmable" coupling type (`CPSETTYPE=CP`). **Value** / **Meaning** 0: The brake is not transferred to the leading axes. 1: The brake is transferred to the leading axes depending on the context.
1 - 31	Reserved

The following rules apply for the programming:

- `CPMBRAKE` must be programmed in one block with `CPDEF` or `CPON` (⇒ can only be programmed for an inactive coupling).
- While defining a coupling, the following values are captured without the explicit programming of `CPMBRAKE`:
  Bit 0 = 1 for pre-processing couplings with `CPSETTYPE=CP` (otherwise bit 0 = 0)

Examples

Example 1:

Programming	Comment
`CPDEF=(AX5) CPLA[AX5]=(AX4) CPMBRAKE[AX5]=0 ;`	Defining a coupling (leading axis Ax4 with following axis Ax5). NST "feed stop / spindle stop" or "CP SW limit stop" should not brake the coupling group.
`...`	

Example 2:

Programming	Comment
`CPDEF=(S2) CPLA[S2]=(S1) ;`	Definition of a spindle coupling: Leading spindle S1 with following spindle S2
`CPON=(S2) CPMBRAKE[S2]=1 ;`	Activation of the coupling with following spindle S2. NST "feed stop / spindle stop" or "CP SW limit stop" should brake the coupling group.
`...`	

10.5.5.18 Response to certain NC/PLC interface signals (CPMVDI)

The CP keyword `CPMVDI` can be used to define the coupling module's response to certain NC/PLC interface signals.

Programming

Syntax:	`CPMVDI[FAx]=` <value>
Designation:	**C**ou**p**ling **M**ode **VDI** Signal
Functionality:	`CPMVDI` is a bit-coded CP keyword, which defines the response of the coupling module of following axis FAx to certain NC/PLC interface signals. The bit combination operators `B_OR`, `B_AND`, `B_NOT`, and `B_XOR` can be used to set individual bits.

Bit	Meaning
0	Reserved.
1	Reserved.
2	Reserved.
3	The effect of NC/PLC interface signal DB31, ... DBX1.3 (axis/spindle disable) on the following axis/spindle can be set via bit 3: Bit 3 = 0 — DB31, ... DBX1.3 has **no** effect on the following axis/spindle. The state of the following axis/spindle with reference to the axis/spindle disable is derived solely from the state of the leading axes/spindles. Bit 3 = 1 — DB31, ... DBX1.3 has an effect on the following axis/spindle. The state of the leading axes/spindles with reference to the axis/spindle disable is not imposed on the following axis/spindle. **Note:** If bit 3 = 1, the axis/spindle disable state of the following axis/spindle only has an effect if the program test or SERUPRO states are not active (see also bit 5).
4	Bit 4 is used to define the enable for the dependent motion components when the NC/PLC interface signal DB31, … DBX1.3 (axis/spindle disable) has an effect on the following axis/spindle: Bit 4 = 0 — Dependent motion components of the leading axes/spindles become effective irrespective of the state of the axis/spindle disable of the relevant leading axis/spindle. Bit 4 = 1 — Dependent motion components of the leading axes/spindles only become effective if the state of the axis/spindle disable of the leading axis/spindle matches that of the axis/spindle disable of the following axis/spindle. Otherwise, the components are suppressed. **Note:** Bit 4 is only of significance if bit 3 is set, i.e. if the NC/PLC interface signal DB31, … DBX1.3 (axis/spindle disable) has an effect on the following axis/spindle (see bits 3 and 5).
5	The effect of NC/PLC interface signal DB21, ... DBX25.7 (program test selected) or DB21, ... DBX1.7 (activate program test) on the following axis/spindle can be set via bit 5: Bit 5 = 0 — DB21, ... DBX25.7 or DB21, ... DBX1.7 has **no** effect on the following axis/spindle. The state of the following axis/spindle with reference to the axis/spindle disable is derived solely from the state of the leading axes/spindles. Bit 5 = 1 — If the "program test" state is active for an axis of the coupling module, then for the following axis/spindle, its own state is active regarding the axis/spindle disable. The state of the leading axes/spindles with reference to the axis/spindle disable is not imposed on the following axis/spindle.

Note:
When bit 5 is set, the program test state still has an effect on the following axis/spindle, even if the leading axes/spindles have a different state.

6 Bit 6 is used to define the enable for the dependent motion components when the NC/PLC interface signal DB21, … DBX25.7 (program test selected) or DB21, ... DBX1.7 (activate program test) has an effect on the following axis/spindle:

Bit 6 = 0	Dependent motion components of the leading axes/spindles become effective irrespective of the state of the axis/spindle disable of the relevant leading axis/spindle.
Bit 6 = 1	Dependent motion components of the leading axes/spindles only become effective if the state of the axis/spindle disable of the leading axis/spindle matches that of the axis/spindle disable of the following axis/spindle. Otherwise, the components are suppressed.

Note:
Bit 6 is only of significance if bit 5 is set, i.e. if the program test state has an effect on the following axis/spindle (see bits 3 and 5).

7 Reserved.

8 Reserved.

Note

The axis/spindle disable, which is set for the following axis/spindle via the NC/PLC interface signal DB31, ... DBX1.3, can be overwritten by the program test (DB21, ... DBX25.7 or DB21, ... DBX1.7) and SERUPRO states; this is also the case for every other axis/spindle.

Effect of bits 3/5 and 4/6

The effects of the different motion components on the following axis/spindle as a function of the associated axis/spindle disable are illustrated in the table below:

A/S disable FA	A/S disable LA_1	A/S disable LA_2	CPMVDI Bit 3/5	CPMVDI Bit 4/6	FA_{Total}	Meaning for FA
0	0	0	0	0	FA_{Cmd} + FA_{DEP1} + FA_{DEP2}	Real movement.
0	0	0	1	0	FA_{Cmd} + FA_{DEP1} + FA_{DEP2}	Real movement.
0	1	1	0	0	FA_{Cmd} + FA_{DEP1} + FA_{DEP2}	Simulated movement
0	0	1	0	0	-	Alarm 16773, different leading axis states with ref. to the A/S disable

A/S disable FA	A/S disable LA_1	A/S disable LA_2	CPMVDI Bit 3/5	CPMVDI Bit 4/6	FA_{Total}	Meaning for FA
1	0	0	0	0	FA_{Cmd} + FA_{DEP1} + FA_{DEP2}	Real movement, FA spindle disable has no effect.
1	0	1	1	0	FA_{Cmd} + FA_{DEP1} + FA_{DEP2}	Simulated movement, FA spindle disable has an effect.
1	0	1	1	1	FA_{Cmd} + FA_{DEP2}	Simulated movement, as bit 4 is set, FA_{DEP1} is suppressed.
0	1	1	0	1		Alarm 16694 bit 4/6 = 1 is only supported if bit 3/5 = 1.
1	0	0	0	1		Alarm 16694 bit 4/6 = 1 is only supported if bit 3/5 = 1.

A/S disable:	Axis/spindle disable This refers to the resulting internal state of the axis/spindle disable. The spindle disable, which is set via the NC/PLC interface signal DB31, … DBX1.3 (axis/spindle disable), can be overwritten by states such as program test (DB21, … DBX25.7 or DB21, … DBX1.7) and SERUPRO, thus generating an axis/spindle state which differs from the NC/PLC interface signal.
FA:	Following axis
LA_1:	Leading axis 1
LA_2:	Leading axis 2
FA_{Total}:	Total setpoint value of the following axis
FA_{Cmd}:	Independent motion component of the following axis
FA_{DEP1}:	Dependent motion component of leading axis 1
FA_{DEP2}:	Dependent motion component of leading axis 2
Real movement:	Real movement means that positioning movements are transferred to the position control.
Simulated movement:	Simulated movement means that no positioning movements are transferred to the position control. The real machine axis remains stationary. This corresponds to the state of an activated axis/spindle disable or program test.

Note

The states in the columns for leading axes 1 and 2 also apply if there are several leading axes/ spindles, which have the same state with reference to the axis/spindle disable.

10.5.5.19 Alarm suppression (CPMALARM)

The CP keyword `CPMALARM` can be used to suppress coupling-related alarms.

Programming

Syntax: `CPMALARM[FAx]=` <value>

Designation: **C**ou**p**ling **M**ode **Alarm**

Functionality: `CPMALARM` is a bit-coded CP keyword for suppressing special coupling-related alarm outputs.

The bit combination operators `B_OR`, `B_AND`, `B_NOT`, and `B_XOR` can be used to set individual bits.

Bit	Value	Meaning
0	= 1	Alarm 16772 is suppressed.
1	= 1	Alarm 16773 is suppressed.
2	= 1	Alarm 16774 is suppressed.
3	= 1	Alarm 22012 is suppressed.
4	= 1	Alarm 22013 is suppressed.
5	= 1	Alarm 22014 is suppressed.
6	= 1	Alarm 22015 is suppressed.
7	= 1	Alarm 22016 is suppressed.
8	= 1	Alarm 22025 is suppressed.
9	= 1	Alarm 22026 is suppressed.
10	= 1	Alarm 22040 is suppressed for the cyclic check in case of previously activated position control for following and leading spindles.
11	= 1	Alarm 16771 is suppressed.
12 - 31		Reserved.

The default values correspond to the settings in the machine data:

- MD11410 $MN_SUPPRESS_ALARM_MASK (mask for suppressing special alarm outputs)
- MD11415 $MN_SUPPRESS_ALARM_MASK_2 (suppress alarm outputs)

Example

Program code	Comment
`CPMALARM[X2]='H300'`	`; The 22025 and 22026 alarms are suppressed for the coupling of the X2 following axis.`

10.5.6 Coupling cascading

Coupling cascades

The coupling modules can be connected in series. The following axis/spindle of a coupling module then becomes the leading axis/spindle of another coupling module. This results in a coupling cascade.

Multiple coupling cascades in series is also possible. The internal computation sequence of the individual coupling modules is performed so that there is no position offset in the coupling relationship. This also applies for a cross-channel cascading.

Example:

Two new coupling modules are created. For the coupling module with following axis X2, the leading axis X1 is defined. For the coupling module with following axis X2, the leading axis X2 and A1 are defined.

Programming
`CPDEF=(X2) CPLA[X2]=(X1) CPDEF=(A2) CPLA[A2]=(X2) CPLA[A2]=(A1)`

Supplementary conditions

- The availability of cascading is option-based (see Section "Requirements (Page 510)").
- Cascades between couplings of existing coupling functions and couplings of generic couplings are not possible.
- A ring coupling is not permitted. It is rejected with alarm 16778:
  "Ring coupling with following axis FAx and leading axis LAx not allowed"
  (A ring coupling occurs when a following axis is also a leading axis of its own coupling module or a leading axis in a series-connected coupling module).

10.5.7 Compatibility

10.5.7.1 Adaptive cycles

Adaptive cycles

The provision of adaptive cycles as fixed component of the NC software ensures a syntactic and functional compatibility to coupling calls of existing coupling types (coupled motion, master value coupling, electronic gearbox and synchronous spindle). This means that as long as the manufacturer/user does not need new coupling characteristics, it is not necessary to modify present coupling calls and any dependent application components (e.g. PLC evaluation of coupling signals).

Assignment to existing coupling commands

The number of adaptive cycles corresponds to the number of existing coupling commands. The assignment is as follows:

Coupling commands	Adaptive cycle
TRAILON	cycle700
TRAILOF	cycle701
LEADON	cycle702
LEADOF	cycle703
COUPDEF	cycle704
COUPON	cycle705
COUPONC	cycle706
COUPOF	cycle707
COUPOFS	cycle708
COUPDEL	cycle709
COUPRES	cycle710
EGDEF	cycle711
EGON	cycle712
EGONSYN	cycle713
EGONSYNE	cycle714
EGOFC	cycle715
EGOFS	cycle716
EGDEL	cycle717

Storage location

Adaptive cycles are stored in the directory "CST".

User-specific adaptive cycles

If necessary (functional completion) the user can copy an adaptive cycle to the directory "CMA" or "CUS" and apply changes there. When reading adaptive cycles, the sequence CUS → CMA → CST is observed and cycle variants are taken over on a first found basis, i.e. the adaptive cycles copied into the directory CMA / CUS by the user are selected on a priority basis.

Note

When upgrading the NC software, a log file is saved in the "CST" directory (Changelog), indicating necessary changes of the adaptive cycles.

10.5.7.2 Coupling types (CPSETTYPE)

Coupling types

If presetting of coupling types (coupled motion, master value coupling, electronic gearbox and synchronized spindle) is required, when creating the coupling module (`CPON`/`CPLON` or `CPDEF`/`CPLDEF`), the keyword `CPSETTYPE` needs to be used also.

Programming

Syntax:	`CPSETTYPE[FAx]=` <value>
Designation:	**C**ou**p**ling **Set Type**
Functionality:	Defines the presettings of coupling characteristics (coupling type).
Value:	Type: STRING

Range of values:

"CP"	Freely programmable
"TRAIL"	Coupling type "Coupled motion"
"LEAD"	Coupling type "Master Value Coupling"
"EG"	Coupling type "Electronic gearbox"
"COUP"	Coupling type "Synchronized spindle"

Default value: "CP"

Example:

Programming	Comment
`CPLON[X2]=(X1) CPSETTYPE[X2]="LEAD"`	`; Creates a coupling module for following axis X2 with leading axis X1 and activates the coupling module. Coupling properties are set such that they correspond to the existing master value coupling type.`

Default settings

Presettings of programmable coupling characteristics for various coupling types can be found in the following table:

Keyword	Coupling type				
	Default (CP)	**Coupled motion (TRAIL)**	**Master value coupling (LEAD)**	**Electronic gear (EG)**	**Synchronous spindle (COUP)**
CPDEF		-	-		
CPDEL		-	-		
CPLDEF		-	-		
CPLDEL		-	-		
CPON					
CPOF					
CPLON					
CPLOF					
CPLNUM	1.0	1.0	-	1.0	1.0
CPLDEN	1.0	1.0	-	1.0	1.0
CPLCTID	Not set	-	0	Not set	-
CPLSETVAL	CMDPOS	CMDPOS	CMDPOS	CMDPOS	CMDPOS
CPFRS	BCS	BCS	BCS	BCS	MCS
CPBC	NOC	NOC	NOC	FINE	IPOSTOP
CPFPOS + CPON	Not set	-	-	Not set	Not set
CPFPOS + CPOF	Not set	-	-	-	Not set
CPFMSON	CFAST	CFAST	CCOARSE	NRGT	CFAST
CPFMON	STOP	VL1): STOP HL2): CONT	VL1): CONT HL2): CONT	STOP	CONT
CPFMOF	STOP	VL1): STOP HL2): STOP	VL1): STOP HL2): STOP	STOP	CONT
CPLPOS + CPON	Not set	-	-	Not set	-

Keyword		Coupling type				
		Default (CP)	Coupled motion (TRAIL)	Master value coupling (LEAD)	Electronic gear (EG)	Synchronous spindle (COUP)
CPMRESET		NONE	MD20110	MD20110	MD20110	MD20110
CPMSTART		NONE	MD20112	MD20112	MD20112	MD20112
CPMPRT		NONE	MD20112 / MD22620[3)]	MD20112 / MD22620[3)]	MD20112 / MD22620[3)]	MD20112 / MD22620[3)]
CPLINTR		0.0	0.0	0.0	0.0	0.0
CPLINSC		1.0	1.0	1.0	1.0	1.0
CPLOUTTR		0.0	0.0	0.0	0.0	0.0
CPLOUTSC		1.0	1.0	1.0	1.0	1.0
CPSYNCOP		MD37200	MD37200	MD37200	MD37200	MD37200
CPSYNFIP		MD37210	MD37210	MD37210	MD37210	MD37210
CPSYNCOP2		MD37202	MD37202	MD37202	MD37202	MD37202
CPSYNFIP2		MD37212	MD37212	MD37212	MD37212	MD37212
CPSYNCOV		MD37220	MD37220	MD37220	MD37220	MD37220
CPSYNFIV		MD37230	MD37230	MD37230	MD37230	MD37230
CPMBRAKE						
	Bit 0	1	-	-	-	-
CPMVDI						
	Bit 3	0	0	0	0	0
	Bit 4	1	1	1	1	1
	Bit 5	0	0	0	0	0
	Bit 6	1	1	1	1	1
CPMALARM		MD11410 MD11415	MD11410 MD11415	MD11410 MD11415	MD11410 MD11415	MD11410 MD11415

Legend:

1) Pre-processing

2) Main run

3) depends additionally on MD22621

- not relevant or not allowed

Additional properties

Value ranges or availability of additional properties of a set coupling type (`CPSETTYPE`) can be found in the following table:

	Default (CP)	Coupled motion (TRAIL)	Master value coupling (LEAD)	Electronic gear (EG)	Synchronous spindle (COUP)
Number of leading axes	≦ 5	≦ 2	1	≦ 5	1
Following axis type	Axis/spindle	Axis/spindle	Axis/spindle	Axis/spindle	Spindle
Defining/deleting coupling module	CPDEF/CPDEL or CPON/CPOF	CPON/CPOF	CPON/CPOF	CPDEF/CPDEL	CPDEF/CPDEL
Defining/deleting leading axis	CPLDEF/CPLDEL or CPLON/CPLOF	CPLON/CPLOF	CPLON/CPLOF	CPLDEF/CPLDEL	CPLDEF/CPLDEL
Cascading	+	+	+	+	-
Dynamic observation of the leading spindle	-	-	-	-	+
Implicit selection/deselection of state control[1)]	-	-	-	-	+
Legend: [1)] also refer to: Function Manual, Extended Functions; Synchronous Spindle (S3) - not relevant or not allowed					

Availability of the specified characteristics depends on the available version (see Section "Requirements (Page 510)").

Example:

The coupled motion coupling type (`CPSETTYPE="TRAIL"`) allows a maximum of two leading axes and cascading. However, this is not available in the basic version, but requires the CP-EXPERT option.

Supplementary conditions

- `CPSETTYPE` can be programmed in synchronous actions.
- If the coupling type (`CPSETTYPE`) is set, certain coupling characteristics are preset and cannot be changed. Subsequent change attempts with keywords cause an error and are rejected with an alarm:

CPSETTYPE=	TRAIL	LEAD	EG	COUP
CPDEF	Alarm 16686	Alarm 16686		
CPDEL	Alarm 16686	Alarm 16686		

CPSETTYPE=	TRAIL	LEAD	EG	COUP
CPLDEF				
CPLDEL				
CPON			Alarm 16686	Alarm 16686
CPLON				
CPOF			Alarm 16686	Alarm 16686
CPLOF				
CPRES	Alarm 16686	Alarm 16686	Alarm 16686	
CPLNUM	Alarm 16686	Alarm 16686		
CPLDEN	Alarm 16686	Alarm 16686		
CPLCTID	Alarm 16686			Alarm 16686
CPLSETVAL	Alarm 16686 with CMDVEL	Alarm 16686 with CMDVEL	Alarm 16686 with CMDVEL	
CPFRS				Alarm 16686 with BCS
CPBC	Alarm 16686	Alarm 16686		
CPFPOS + CPON	Alarm 16686	Alarm 16686		
CPFPOS + CPOF	Alarm 16686	Alarm 16686	Alarm 16686	
CPFMSON	Alarm 16686	Alarm 16686		Alarm 16686
CPFMON	Alarm 16686	Alarm 16686 with ADD/STOP	Alarm 16686	Alarm 16686 with STOP
CPFMOF	Alarm 16686	Alarm 16686 with ADD/STOP	Alarm 16686 with ADD	Alarm 16686 with ADD
CPLPOS + CPON	Alarm 16686	Alarm 16686		Alarm 16686
CPMRESET	Alarm 16686	Alarm 16686	Alarm 16686	Alarm 16686
CPMSTART	Alarm 16686	Alarm 16686	Alarm 16686	Alarm 16686
CPMPRT	Alarm 16686	Alarm 16686	Alarm 16686	Alarm 16686
CPMBRAKE	Alarm 16686	Alarm 16686	Alarm 16686	Alarm 16686
Number of leading axes ($\sum$ LA)	Alarm 16672 with $\sum$ LA > 2	Alarm 16672 with $\sum$ LA > 1	Alarm 16672 with $\sum$ LA > 5	Alarm 16672 with $\sum$ LA > 1
Following axis type				Alarm 14092 with axis

10.5.7.3 Projected coupling (CPRES)

If the coupling type "Synchronous spindle" is set, (see `CPSETTYPE`), the coupling properties contained in machine data can be activated instead of the programmed coupling properties.

References:
Functions Manual Extension Functions; Synchronous spindles (S3);
Chapter "Programming of synchronous spindle couplings"

Programming

Syntax:	`CPRES=` (<following spindle>)
Designation:	**C**ou**p**ling **Res**tore
Functionality:	Activates projected data of the synchronous spindle coupling to following spindle FAx.
Following spindle:	Type: AXIS
	Range of values: All defined spindle names in the channel

Example:

Programming	**Comment**
`CPLON[S2]=(S1) CPSETTYPE[S2]="COUP"`	`; Creates a coupling module for following spindle S2 with leading spindle S1 and activates the coupling module. Coupling properties are set such that they correspond to the existing synchronous spindle coupling type.`
`...`	
`CPRES=(S2)`	`; Activates projected data of the synchronous spindle coupling to following spindle S2.`

Boundary conditions

- `CPRES` is only allowed when the coupling type "Synchronous spindle" (`CPSETTYPE="COUP")` is set.
- Application of `CPRES` to an already active coupling results in a new synchronization.
- Applying `CPRES` to an undefined coupling module does not result in any action.

10.5.8 Cross-channel coupling, axis replacement

The following and leading axes must be known to the calling channel.

Following axis

The following axis is requested for replacement in the channel when programming a CP keyword in the part program, depending on the axis replacement projection (MD30552) with the language command `GETD`.

Axis change of the following axis after activating the coupling module is only permitted in the channel. Changing from channel to main run and vice versa is still possible, however a change across channel boundaries is not. Supplementary conditions and properties still apply to axis change. The axis replacement via channel axes is released again after deactivating the coupling module.

Reference:
Function Manual, Extended Functions; Mode Groups, Channels, Axis Replacement (K5)

Leading axes

Axis change of leading axes can be performed independently of the state of the coupling.

10.5.9 Behavior with rotary axes

Rotary axes as leading or following axes

It is possible to couple rotary axes to a linear axis and vice versa. Note that a direct assignment of degrees to mm must be performed using the coupling rule.

Example:

A = Rotary axis, X = Linear axis

Programming	**Comment**
`N10 G0 A0 X0`	`; Traverse motion: X = 0 mm, A = 0 degrees`
`N20 CPON=(A) CPLA[A]=(X) CPLNUM[A,X]=2`	`; A coupling module for rotary axis A with linear axis X as leading axis is created and activated. The coupling value is 2.`
`N30 X100`	`; Traverse motion: X = 100 mm, A = 200 degrees (= 100*2)`

Modulo reduced rotary axes as leading axes

With modulo reduced rotary axes as leading axes, the input variable is not reduced during the reduction of the leading axis. The non-reduced position is still taken as the input variable, i.e. the traversed distance is considered.

Example:

A = Modulo reduced rotary axis, X = Linear axis

Programming	Comment
`N10 G0 A0 X0`	; Traverse motion: X = 0 degrees, X = 0 mm
`N20 CPON=(X) CPLA[X]=(A) CPLNUM[X,A]=0.5`	; A coupling module for linear axis X with rotary axis A as leading axis is created and activated. The coupling value is 0.5.
`N30 A200`	; Traverse motion: A = 200 degrees, X = 100 mm (= 200*0.5)
`N40 A=IC(200)`	; A traverses through 200 degrees in a positive direction to 400 degrees, Display A = 40. X traverses through 100 mm to 200.
`N50 A=IC(100)`	; A traverses from 40 degrees to 140 degrees, X traverses through additional 50mm to 250.
`N60 A=ACP(80)`	; A traverses in the positive direction to 50 degrees, the traversing path is 300 degrees in the positive direction. X traverses correspondingly by 150 mm in the positive direction. The end position is therefore X = 400.

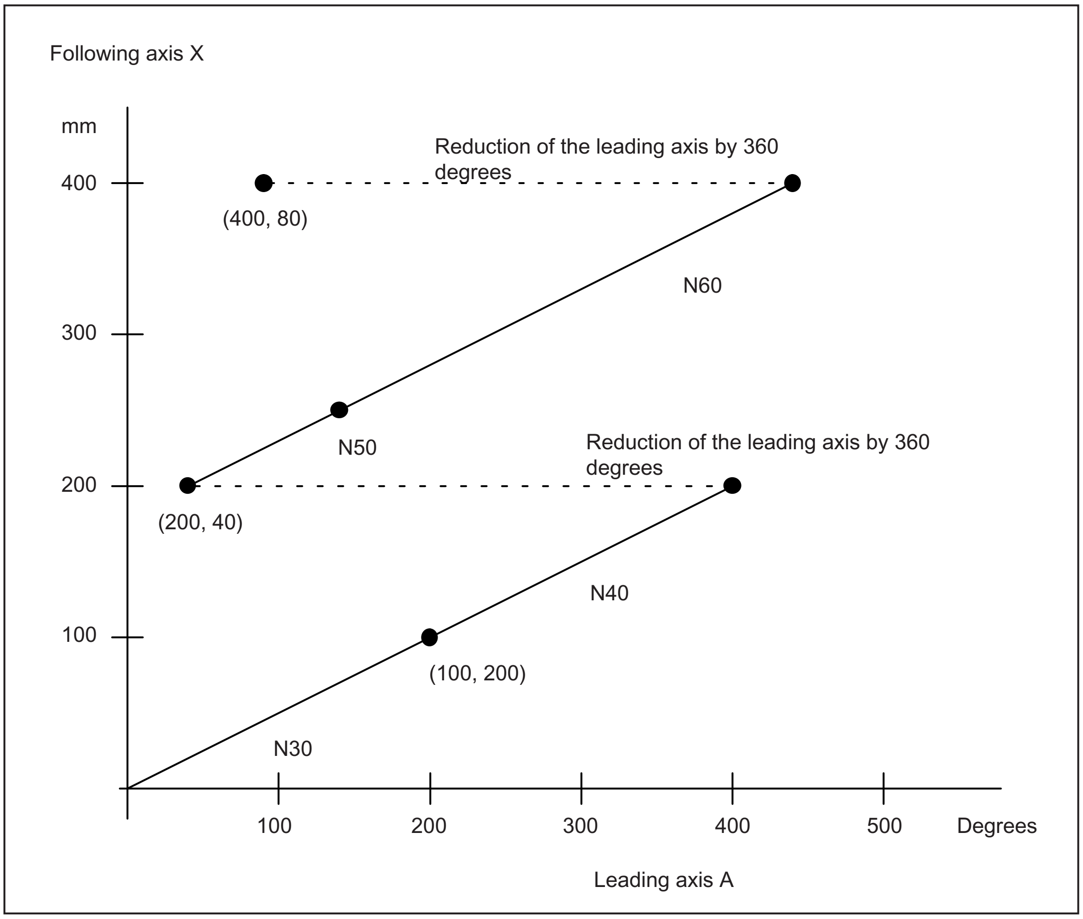


Figure 10-12 Example: Modulo reduced rotary axis to linear axis
(...) Position indication for X, A

10.5.10 Behavior during POWER ON, ...

Power on

No coupling is active at power ON. Coupling modules are not available.

RESET

The behavior on RESET can be set separately for each coupling module (see `CPMRESET`). The coupling can be activated, deactivated or the current state can be retained.

Mode change

The coupling remains active during a mode change. The coupling is suppressed (not deselected!) only in JOG-REF mode when referencing a following axis.

Reference point approach

`G74` of the following axis is not possible with an active coupling. An alarm is output.

If the JOG-REF mode is selected and the following axis is traversed, the coupling is suppressed. The coupling is only performed after JOG-REF mode is cancelled.

SERUPRO

The SERUPRO procedure will simulate the generic coupling and provide values for a restart.

With axial couplings, the simulation always assumes a setpoint coupling, which means, when there is an actual value coupling, this is switched to setpoint coupling during the SERUPRO procedure. This can mean that the simulation is not performed correctly.

Further deviations from the real procedure can occur due to increased simulation speed and canceled axis dynamics limitations.

10.5.11 CP SW limit monitoring

10.5.11.1 Function

What is the purpose of the function?

The "CP-SW limit monitoring" function improves the braking response of a following axis or a following spindle in axis operation when approaching a software limit switch.

Advantages:

- This avoids that the software limit switch is passed.
- If possible, synchronous operation of the coupling is kept.

Availability

The "CP SW limit monitoring" function can only be activated for following axes from:

- Generic couplings, type "Freely programmable (CPSETTYPE[FAx] = "CP")
- Couplings (generic coupling with CPSETTYPE[FAx] not equal to "CP", coupled motion, electronic gearbox, master value coupling or synchronous spindle) with a maximum of one active leading axis/spindle

In all other cases (and when "CP SW limit monitoring" is not activated) the previous function of the software limit switch monitoring is active (see Chapter "Limit switch monitoring" in the Function Manual, Basic Functions, A3: Axis Monitoring, Protection Areas).

Monitoring and setting the brake

The "CP-SW limit monitoring" function checks, in every IPO cycle, as to whether the movement of the following axis/spindle can be enabled for the following IPO cycle, so that the axis can always stop in plenty of time before the software limit switch. If this is not the case, for the following axis, a "CP-SW limit stop" is set; this signifies unconditional direction-specific braking along the contour.

Transferring the brake to the axes

Setting the brake means that the coupling-independent motion component (CMD and CORR component) of the following axis is stopped. On the other hand, the coupling-dependent motion component (DEP component) of the following axis can only be braked by braking the leading axes.

For couplings (generic coupling with CPSETTYPE[FAx] not equal to "CP", coupled motion, electronic gearbox, master value coupling or synchronous spindle) with a maximum of one active leading axis/spindle, the brake is also set for the (only) active leading axis in the collision direction (depending on the coupling factor).

For generic couplings, type "freely programmable (CPSETTYPE[FAx] = "CP"), the brake is only transferred to all of the active leading axes, if the coupling property CPMBRAKE[FAx] Bit 0 = 1 has been programmed (see "Reaction to stop signals and commands (CPMBRAKE) (Page 553)").

The collision direction depends on the coupling factor for the specific leading axis: With a negative coupling factor, the collision direction reverses, there is no transfer for a coupling factor of "zero". For non-linear couplings (e.g. curve table), the coupling factor is derived from the gradient, determined as linear approximation.

Note

A brake set for a following axis can only influence its leading axes as long as the coupling is active.

Braking behavior

The normal acceleration ramp is used when braking the axes towards the software limit switch along the contour, maintaining the coupling definition.

When braking, the stop state 75 "brake request" and situation-dependent, the higher priority stop states 22 "wait: spindle enable missing", 12 "wait for axis/spindle release" or 71 "wait for enable, transformation axis" are displayed.

If the following axis is stationary as result of the "CP-SW limit stop", and cannot approach the software limit switch any closer, then the following alarm is displayed:

10625 "%?C{channel %1: %}block %3 following axis/spindle %2 with CP-SW limit stop %4"

This means that up to the "final" stopping position, the situation can be somewhat relieved by moving the following axis in the opposite direction.

Note

For the "CP-SW limit monitoring", coupling involves maintaining synchronous operation. As a consequence, it cannot be guaranteed that the axis stops at precisely the correct position, if the coupling rule is to be maintained. If the brake is to be transferred to the leading axes, and the interpolators brake as required, then the leading axis comes to a standstill in front of the software limit switch. The maximum distance to the software limit switch is then obtained from the actual maximum acceleration rate of the following axes multiplied by the square of the interpolation cycle time.

Retraction

The user is responsible for retraction.

For diagnostic purposes, there are:

- the stop state 75 "brake request"
  (if an axis has a travel command, but stops or may not traverse as a result of a braking request)
- and the system variables:
  $AA_BRAKE_STATE (actual brake status)
  $AA_BRAKE_CONDB (context-sensitive conditions for the interpolator stop in the BCS)
  $AA_BRAKE_CONDM (context-sensitive conditions for the interpolator stop in the MCS)
  including the OPI variables aaBrakeState, aaBrakeCondB and aaBrakeCondM derived from these.

Example for retraction:

1. acknowledge alarm 10625.
2. Switch to the JOG mode.
3. Traverse the leading axis using the traversing keys so that the following axis moves away from the software limit switch.

10.5.11.2 Parameterization

Activation

The "CP SW limit monitoring" function is activated on an axis-for-axis basis using machine data:

MD30455 $MA_MISC_FUNCTION_MASK (axis functions)

Bit	Value	Meaning
11	0	CP SW limit monitoring is not active.
	1	CP SW limit monitoring is active.

10.5.11.3 Programming

Transferring the brake to the leading axes

For generic couplings, type "freely programmable" (CPSETTYPE[FAx] = "CP"), by programming the coupling property CPMBRAKE (see "Reaction to stop signals and commands (CPMBRAKE) (Page 553)") it can be set as to whether the brake of the following axis, initiated using the "CP-SW limit monitoring" function, should also be transferred to the leading axes.

10.5.11.4 Boundary conditions

Possible fault sources

If the axes do not brake as required, this could be due to the following reason:

At least one leading axis does not react to the brake, because it is also a leading axis in its own right (cascade), and the brake is not transferred to its leading axes. A brake is only transferred to the leading axes (recursive), if the following applies for the following axes involved: CPSETTYPE[FAx] = "CP" and CPMBRAKE[Fax] bit 0 = 1

Even if all of the axes brake as required, the contour can be violated as a result of different acceleration rates and/or coupling factors if these values have not been appropriately configured and/or programmed.

Even if all of the previous preconditions are fulfilled, there are still some reasons why it cannot be avoided that a following axis passes a software limit switch:

- The actual maximum acceleration of a leading axis changes.
- The actual maximum acceleration of the following axis becomes smaller.
- The velocity of the following axis when switching in the coupling or when the CP-SW limit monitoring responds is already so high that with the actual maximum acceleration it is no longer possible to stop the axis in the appropriate time.

If the axis cannot be stopped in the appropriate time, then as before, the previous software limit switch monitoring responds, and stops the following axis at the software limit switch.

Braking response for transformations

If the axis to be braked is the output of a transformation, and it has a (MCS) travel command in the collision direction, then the brake is transferred to all input axes of the transformation in both directions, and a path stop is executed for this transformation. However, the following exceptions apply: It does not apply to the independent axes of the transformation. For TRANSMIT and TRAANG, these are the axes specified in machine data $MC_TRAFO_AXES_IN_*[2]. For the transformations, braking is only transferred to their dependent input axes.

DRF offset

Setting a CP-SW limit stop can cause a DRF movement to be canceled, as shown in the following example:

The basic motion is path motion with extremely low velocity in the positive direction. A larger DRF correction motion applies to this in the opposite direction. This results in a travel command in the negative direction. If these movements are now stopped using a CP-SW limit stop, then for standard machine data, this means that DRF motion is canceled, and only path motion is continued after the stop is withdrawn.

NCU link

The following restriction must be taken into account for cross-NCU couplings based on lead-link axis and axis container:

As the leading axis can be interpolated on another NCU, it can take between one and two IPO cycles for the brake to be transferred to the NCU of the leading axis.

10.5.11.5 Examples

Example 1: Generic coupling type "freely programmable" (CPSETTYPE[FAx] = "CP")

Configuration:

MD26110 $MA_POS_LIMIT_PLUS[AX2]=15 ;	Position of the 1st software limit switch of the following axis for the traversing range limit in the positive direction
MD30455 $MA_MISC_FUNCTION_MASK[AX2] = 'H800' ;	Bit 11 = 1 (CP-SW limit monitoring is activated)

Programming:

```
; start position of X and Y
N100 G0 X0 Y0
; coupling activation FA=Y LA=X
; set with CPMBRAKE bit 0  (transfer of the brake to the leading axis)
N120 CPON=Y CPSETTYPE="CP" CPLA=X CPLDEN=2 CPMBRAKE B_OR='H1'
N140 G4 F2

; traverse leading axis X
N200 G1 X=1000 F1000
```

Example 2: Couplings with a maximum of one active leading axis/spindle

With the following calls, the CP-SW limit monitoring is executed – including the transfer of the brake to the leading axis – if MD30455 MA_MISC_FUNCTION_MASK[AX2] bit 11 is set:

```
TRAILON(Y,X,0.5) ;    Definition and switching in the coupling of the coupled motion
                      axis Y two leading axis X.
```

```
EGDEF(Y,X,1) ;               Definition of an EG axis group with setpoint coupling
                             from X to Y (following axis).
EGON( Y,"FINE",X,1,2) ;      Activating coupling.
```

10.5.12 Disturbance characteristic

10.5.12.1 Rapid stop

Function

The rapid stop stops the axis / spindle without ramp, i.e. the velocity setpoint value is specified as zero. This default applies the brakes at the current limit. The servo enable is retained.

The rapid stop is set at:

- Stop A and Stop C (Safety Integrated)
- Alarms with rapid stop as configured braking behavior
- Reaching the hardware limit switch and rapid stop as configured braking behavior: MD36600 $MA_BRAKE_MODE_CHOICE = 1

Switchover to actual-value coupling.

The actual values of the leading spindle are used to calculate the setpoint values as soon as the rapid stop of the leading spindle is reported to a generic coupling.

The changeover to actual value coupling takes place smoothly and remains active till the servo enable as well as the pulse enable is available again to the leading spindle and no more position offset takes place. The setpoint value calculation is programmed as with `CPLSETVAL` only if these conditions are fulfilled.

Note

A rapid stop that was initiated on reaching the hardware limit switch does not changeover the actual value coupling.

Response of the following spindle

If a rapid stop is detected for a leading spindle and the following spindle does not execute any rapid stop by itself, then the following spindle tries to follow the dynamics of the movement of the leading spindle defined within its framework. As position synchronization is generated, there may be oscillations in the following axis in relation to the position to be approached.

The start of a rapid stop for a leading axis/spindle is detected across NCUs.

Note

A simultaneous rapid stop of the leading and following spindle is executed in the synchronized spindle coupling type (`CPSETTYPE="COUP"`) during a servo alarm.

10.5.13 Tracking the deviation from synchronism

10.5.13.1 Fundamentals

Deviation from synchronism

Workpiece machining operations which are to be carried out both on the face front and the face rear require a workpiece transfer to another workpiece receptacle (e.g. a counterspindle chuck).

When workpieces are transferred from front to rear machining, a position offset may result from the closing of the workpiece receptacle. This could be down to square-edged workpieces or due to the generation of an angular momentum when the workpiece receptacle (chuck) is closed quickly during a movement. Depending on the resistance of the workpiece, the tension can be detected by means of an increase in the current consumption of both the motors involved in the coupling and/or by means of the workpiece being subjected to torsion.

This could lead to the following NC/PLC interface signals being reset, according to the synchronism tolerance which has been set, and the magnitude of the offset:

DB31, ... DBX98.1 (coarse synchronous operation) and/or

DB31, ... DBX98.0 (fine synchronous operation)

For the setpoint coupling, the position and velocity setpoints are calculated precisely in accordance with the programmed coupling rule and output to the Control Units. If identical drives and a rigid workpiece are used, this will lead to a regulative deviation at the leading and following spindle, half due to a setpoint difference and half due to an actual value difference.

Function

The "track the deviation from synchronism" function serves to detect the position offset which has been imposed on the actual value and to correct the following spindle when calculating the setpoint.

Requirement

A coupling closed via the part/chuck must be in place in order to use this function.

Versions

There are two different options for determining the deviation from synchronism:

1. The deviation from synchronous operation is determined by the NC (see "Measuring the deviation from synchronism (Page 577)").
2. The deviation value is already known and entered by the user directly (see "Entering the deviation from synchronism directly (Page 580)").

In both cases, the deviation value is then incorporated into the setpoint value calculation for the following spindle, as a correction value.

Availability

The "track the deviation from synchronism" function was developed for machine couplings (`CPFRS="machine"`). This means that it is also available for the "synchronous spindle" coupling type (`CPSETTYPE="COUP"`).

Like the other higher-level movements, the availability of the function (e.g. speed difference) depends on the option (see "Requirements (Page 510)").

10.5.13.2 Measuring the deviation from synchronism

The controller measures the difference between the setpoint positions and actual positions when the following spindle is operating in synchronism. This results in a correction value, which is saved in a system variable.

Requirements

The following requirements must be met to enable the controller to calculate the correction value:

- Requirements if the set coupling type is "synchronous spindle" (`CPSETTYPE="COUP"`):
  - The coupling has precisely one leading spindle (requirement is met if `CPSETTYPE="COUP"`).
  - The coupling factor (quotient from `CPLNUM` and `CPLDEN`) is 1 or -1.
  - The following value is derived from the setpoint position ("DV") or the actual position ("AV") of the leading spindle.
  - Setpoint synchronism must be achieved:
    DB31, ... DBX99.4 (synchronization running) = 0
  - Setpoint synchronism must not decline again.
  - No overlaid movement (DB31, ... DBX98.4 = 0) must be present.
  - A dynamics limit is required for the leading spindle, in order to exclude the possibility of the following spindle being subjected to excessive demands.
- Requirements for a free generic coupling with `CPFRS="machine"`:
  - The configured spindles are coupled.
  - The coupling has precisely one leading spindle.
  - The coupling factor (quotient from `CPLNUM` and `CPLDEN`) is 1 or -1.
  - The following value is derived from the setpoint position (`CPLSETVAL="CMDPOS"`) or the actual position (`CPLSETVAL="ACTPOS"`) of the leading spindle.
  - Setpoint synchronism must be achieved:
    DB31, ... DBX99.4 (synchronization running) = 0
  - Setpoint synchronism must not decline again.
  - No overlaid movement (DB31, ... DBX98.4 = 0) must be present.
  - A dynamics limit is required for the leading spindle, in order to exclude the possibility of the following spindle being subjected to excessive demands.

Note

Dynamics limit for the leading spindle

The "dynamics limit for the leading spindle" property is specified automatically when the "synchronous spindle" coupling type (`CPSETTYPE="COUP"`) is set. In the case of other coupling types, it is the particular responsibility of the user/machine manufacturer to provide suitable measures to ensure that the following spindle cannot become dynamically overloaded.

Activation

Measuring and tracking of the deviation from synchronism are activated by setting the following NC/PLC interface signal to "1":

DB31, ... DBX31.6 (track synchronism)

The signal only has an effect on the following spindle.

Note

In the following cases, signal DB31, ... DBX31.6 (track synchronism) is ignored:

- Axis/spindle disable is active (DB31, ... DBX1.3 = 1).
- Program test is selected.
- SERUPRO is active.

If one of these situations arises when the "track the deviation from synchronism" function is already active, the function will be deactivated.

Time when measurement is performed

The time when the measurement is performed and the correction value is calculated depends on the bit 7 setting made in the following item of machine data:

MD30455 $MA_MISC_FUNCTION_MASK (axis functions)

Bit	Value	Meaning
7	0	The correction value is calculated continuously, as long as the NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) is set and setpoint synchronism is active (cyclic calculation).
	1	The correction value is only calculated at the time when the NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) changes from 0 to 1 (edge evaluation).

Note

If a temporal extension is taken into account when relieving the tension between the leading and the following spindle, bit 7 should be set to 0. The interface signal is then state-controlled.

The time required to relieve the tension can depend on various factors (e.g. K_V factor of position control, accelerating power of the motors) and must be determined by way of experiment.

Measuring sequence

The NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) can only become effective once setpoint synchronism has been achieved:

DB31, ... DBX99.4 (synchronization running) = 0

The actual values of the following spindle are read within the interpolation cycle and the difference between them and the setpoint position is calculated, either for as long as signal DB31, ... DBX31.6 is activated, or just once on that signal's rising edge (the frequency will depend on the setting of bit 7 in MD30455; see the section titled "Time when measurement is performed").

The correction value is the difference between the setpoint and actual-value synchronism positions. This value is saved for the corresponding following spindle in the following system variable:

$AA_COUP_CORR[S<n>] (following spindle: correction value for synchronous spindle coupling)

Note

You must ensure that the velocity of the leading and following axes is kept as constant as possible and that no acceleration jump occurs for the duration of the measurement.

Example

When the coupling of the synchronous spindle [S2] is activated, a position offset of 77 degrees is also programmed:

```
CPON=S2 ... CPFPOS[S2]=AC(77)
```

When the workpiece receptacle is closed, this results in a mechanical position offset, which leads to an actual-value position offset of 81 degrees.

When the "track the deviation from synchronism" function is activated (DB31, ... DBX31.6 = 1) and setpoint synchronism has been achieved (DB31, ... DBX99.4 = 0), the actual-value position offset ($VA_COUP_OFFS[S2] = 81) is compared with the setpoint position offset ($AA_COUP_OFFS[S2] = 77). This results in a correction value of 4 degrees, which is saved in system variable $AA_COUP_CORR[S2].

10.5.13.3 Entering the deviation from synchronism directly

If the deviation value is known, it can be written directly to the system variable $AA_COUP_CORR for the corresponding following spindle. This is executed via a part program or synchronized action.

Note

Please note that the system variable can only be written once the mechanical coupling has been created. Otherwise a new offset may arise when closing the chuck.

Requirements

To enable the system variable $AA_COUP_CORR to be written from the part program or synchronized actions, a generic machine coupling must have been activated at least once for the corresponding following spindle since the most recent controller power-up was performed.

10.5.13.4 Synchronism correction

If correction value $AA_COUP_CORR[S<n>] is a value other than zero and a generic machine coupling has been active for following spindle S<n> (by means of `CPFRS="machine"` or `CPSETTYPE="COUP"`), the following NC/PLC interface signal is set:

DB31, ... DBX103.0 (synchronism correction is taken into account)

The correction value is incorporated into the setpoint value calculation for the following spindle, in the coupling module. Resetting the setpoint by the coupling offset relieves the tension between the leading and following spindles.

The synchronism signals are produced by comparing the actual values with the corrected setpoints. Once a correction process has been undertaken, the synchronism signals should be present again:

DB31, ... DBX98.1 (coarse synchronism) and/or

DB31, ... DBX98.0 (fine synchronism)

The correction value can be implemented and the synchronism signals produced as well, since the whole point of the "tracking the deviation from synchronism" function is to improve synchronism when tension is present. When implementing this value, the accelerating power is restricted to no more than 10% of the maximum acceleration and velocity.

When $AA_COUP_CORR[S<n>] has been implemented in full, the following NC/PLC interface signal is set:

DB31, ... DBX99.2 (synchronism correction implemented)

This still applies even if $AA_COUP_CORR[S<n>] is zero and no correction needs to be implemented.

When synchronism correction is complete, the NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) must be reset to "0" in order to restore the rigidity of the coupling.

The correction value is not changed again once signal DB31, ... DBX31.6 has been reset or once the coupling has been deactivated (with `CPOF`). The system variable $AA_COUP_CORR[S<n>] then returns a constant value.

The correction value is taken into account until it is reset by setting system variable $AA_COUP_CORR[S<n>] to "0", which must be done, at the very latest, once the workpiece is removed from the spindle.

Note

The setpoint correction by means of the system variable $AA_COUP_CORR[S<n>] impacts on all subsequent following spindle programming in the same way as a position offset, similar to a DRF offset in the machine.

10.5.13.5 Diagnostics for synchronism correction

The current value of $AA_COUP_CORR (correction value for tracking the deviation in synchronism) is displayed in the "Axis/Spindle Service" window, under the "Position offset for the leading axis/spindle setpoint" line, for the purposes of diagnostics.

System variable $AA_COUP_CORR_DIST ($AA_COUP_CORR distance-to-go) can be used to determine how much of the correction value is still be implemented.

10.5.13.6 Resetting synchronism correction

Versions

Synchronism correction can be reset in the following ways:

- Writing value "0" to variable $AA_COUP_CORR[S<n>].
  Synchronism correction is suppressed via a ramp with reduced accelerating power (just as when a correction value is implemented).
- Resetting synchronism correction via the PLC.
  On the rising edge of the NC/PLC interface signal:
  DB31, ... DBX31.7 (reset synchronism correction)
  , the variable $AA_COUP_CORR[S<n>] is set to zero and synchronism correction is reset as follows:
  - If the spindle is in speed control mode, the correction movement is stopped. The existing synchronism correction is then transferred to the setpoint position.
  - In all other cases, the synchronism correction that has already been implemented is reset in exactly the same way as when variable $AA_COUP_CORR[S<n>] is set to zero.

Requirements

A requirement for resetting synchronism correction is that the correction value is not currently being calculated (see Section "Measuring the deviation from synchronism (Page 577)").

End of the reset procedure

When the reset procedure is complete, the following NC/PLC interface signal is set:

DB31, ... DBX99.2 (synchronism correction implemented)

If the NC/PLC interface signal DB31, ... DBX103.0 (synchronism correction is taken into account) is also reset, so too can the NC/PLC interface signal DB31, ... DBX31.7 (reset synchronism correction) be reset.

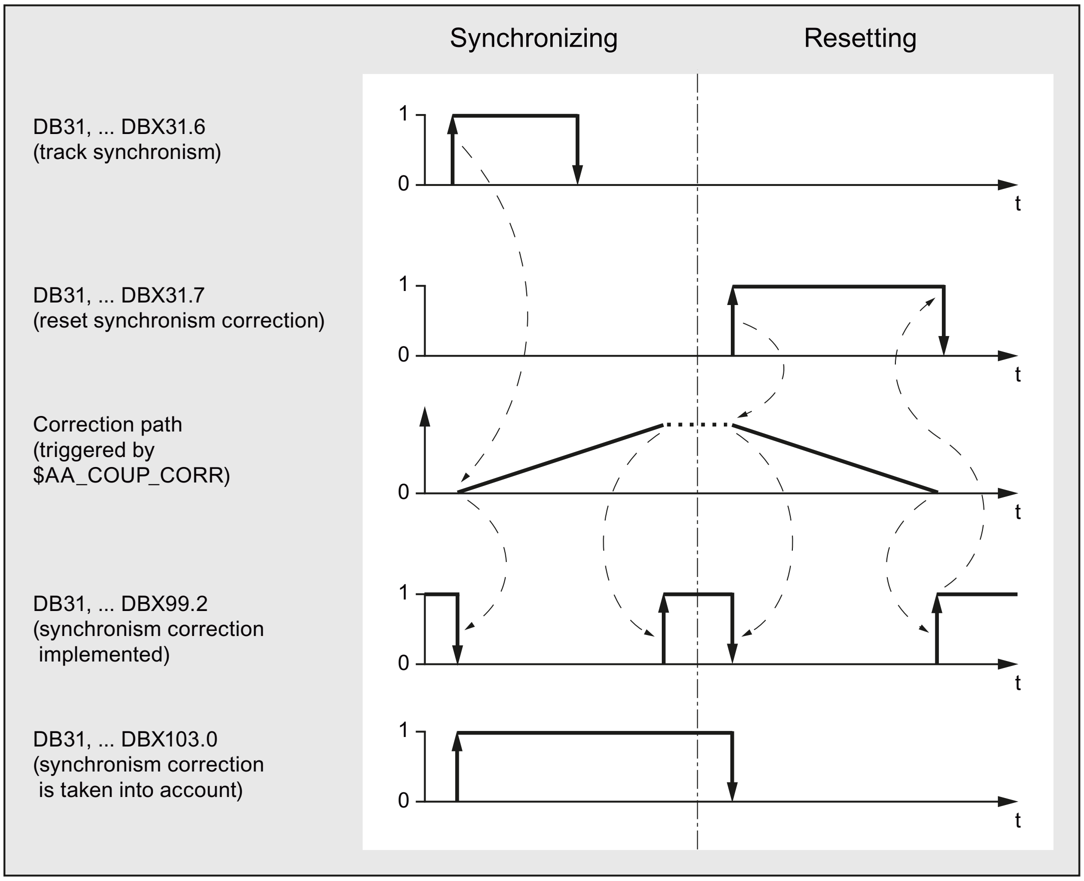


Figure 10-13 Time diagram for synchronizing and resetting synchronism correction

Note

If the correction path has not been traversed in full and the NC/PLC interface signal DB31, ... DBX31.7 (reset synchronism correction) has not been reset, writing to variable $AA_COUP_CORR[S<n>] will not have any effect.

10.5.13.7 Limitations and constraints

Several following spindles

If a leading spindle has several following spindles, each of these following spindles can be processed with the axial NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) separately from one another.

Writing variable $AA_COUP_CORR

System variable $AA_COUP_CORR is only written from the part program or synchronized actions when a generic machine coupling has been activated for the corresponding axis/spindle at least once.

Correction value

If the correction value $AA_COUP_CORR is being written via a part program/synchronized action, as well as being determined due to the "track the deviation from synchronism" function being activated (DB31, ... DBX31.6 = 1), the most recent event to occur is always the one that takes effect.

Resetting synchronism correction

Writing correction value $AA_COUP_CORR (via a part program or synchronized action or when performing a calculation) has no effect when synchronism correction is being reset.

Response to channel/mode group reset

Synchronism correction is not reset in the event of a channel/mode group reset, it is retained instead.

Response to search for reference and zero mark synchronization

If a search for reference or zero mark synchronization procedure is performed for spindles, synchronism correction is reset automatically.

The system variable $AA_COUP_CORR must not be set during the search for reference/zero mark synchronization and, as a result, no measurement may be taken for the deviation from synchronism either.

Furthermore, the synchronism correction must have been implemented in full before the search for reference/zero mark synchronization is started.

Response to an interruption

If an interruption occurs (e.g. emergency stop), synchronism correction is reset automatically, the existing synchronism correction is transferred to the setpoint position, and the NC/PLC interface signal DB31, ... DBX99.2 (synchronism correction implemented) is set.

If, once the interruption has been dealt with, the generic machine coupling remains active and the NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) is set, the following applies:

- If the signal is level-triggered (MD30455 $MA_MISC_FUNCTION_MASK, bit 7 = 0), the deviation from synchronism is measured and written to $AA_COUP_CORR; the setpoints are corrected accordingly.
- If the signal is edge-triggered (MD30455 $MA_MISC_FUNCTION_MASK, bit 7 = 1) and the deviation from synchronism is to be measured, a new rising edge of the NC/PLC interface signal DB31, ... DBX31.6 (track synchronism) needs to be executed.

10.5.14 Examples

10.5.14.1 Programming examples

Direct switch on/off with one leading axis

A coupling module is created and activated with following axis X2 and leading axis X1. The coupling factor is 2.

```
CPON=(X2) CPLA[X2]=(X1) CPLNUM[X2,X1]=2
...
CPOF=(X2)                           ; The coupling is deactivated and the created
                                      coupling module is deleted with CPOF.
```

Direct switch on/off with two leading axes

A coupling module is created and activated with following axis X2 and leading axes X1 and Z. The coupling factor regarding leading axis X1 is 2, the coupling factor regarding leading axis Z is 3.

```
CPON=(X2) CPLA=(X1) CPLNUM[X2,X1]=2 CPLA=(Z) CPLNUM[X2,Z]=3
...
CPOF=(X2)                           ; The coupling is deactivated and the created
                                      coupling module is deleted with CPOF.
```

Selective switch-off with two leading axes

A coupling module is created and activated with following axis Y and leading axes X and Z. The coupling factor regarding leading axis X is 2, the coupling factor regarding leading axis Z is 1.2.

```
CPON=(X2) CPLA[X2]=(X1) CPLNUM[X2,X1]=2 CPLA[X2]=(Z) CPLNUM[X2,Z]=1.2
...
CPOF=(X2) CPLA[X1]=(Z)              ; The coupling with leading axis Z is deactiva-
                                      ted with CPOF, the coupling with leading axis
                                      X1 is retained. The created coupling module
                                      remains created.
```

Selective switch-on/off with three leading axes

A coupling module is created and activated with following axis X2 and leading axes X1, Z and A.

```
N10 CPDEF=(X2) CPLA[X2]=(X1) CPLA[X2]=(Z) CPLA[X2]=(A)
N20 CPON=(X2)                             ; All leading axes become active, i.e. all
                                            contribute a position component accord-
                                            ing to the coupling rule (coupling com-
                                            ponent) to axis X2.
```

```
N30 CPOF=(X2)                          ; All leading axes are deactivated.
N40 CPLON[X2]=(X1)                     ; Leading axis X1 is activated, only this
                                         axis supplies a coupling component.
                                         Leading axes Z and A remain deactivated.
N50 CPLON[X2]=(A)                      ; Leading axis X1 remains active, leading
                                         axis A is deactivated, X1 and A contrib-
                                         ute coupling components (→ selective
                                         switch-on is additive, the condition of
                                         the other leading axes is retained).
N60 CPLON[X2]=(Z) CPLOF[X2]=(A)        ; Leading axis Z is activated, leading ax-
                                         is A deactivated. Leading axes X 1and Z
                                         are now active.
N70 CPLOF[X2]=(X1)                     ; Leading axis X1 is deactivated. Leading
                                         axis Z remains active.
```

Definition/deletion of a coupling module

With `CPDEF` a coupling module is created and activated with following axis X2 and leading axes X1 and Z. The coupling is not activated. Following axis X2 does not follow the coupling rule!

```
CPDEF=(X2) CPLA[X2]=(X1) CPLNUM[X2,X1]=2 CPLA[X2]=(Z) CPLNUM[X2,Z]=3
...
```

Activation can be done with `CPON`, deactivation with `CPOF`.

After the deactivation of the coupling relationship to all leading axes, the coupling object can be deleted. Reserved memory is released:

```
CPDEL=(X2)
```

10.5.14.2 Adapt adaptive cycle

Target

Coupled motion in the machine co-ordinate system must be possible with the existing coupling command `TRAILON`. The adaptive cycle for `TRAILON` is supplemented with the coupling characteristic "Co-ordinate reference" (`CPFRS`).

Procedure

1. Copy adaptive cycle 700 from directory "CST" to directory "CMA".
2. Supplement cycle 700 with the following entry:
   CPFRS[_FA]="MCS"
3. Comment to cycle changes (e.g. user version number and change date).
4. Save cycle.

```
;*CHANGE : 07.01.02.00 Mar 08, 2006
;$PATH=/_N_CST_DIR/_N_CYCLE700_SPF
;USER V1.1 Mar 22, 2006
;classic TRAILON(FA,LA,Factor)
PROC CYCLE700(AXIS _CPF=NO_AXIS, AXIS _CPL=NO_AXIS,REAL _CPLF=1) IPTRLOCK SBLOF DISPLOF ICYCOF
;*IF _CPLF==0 GOTOF _CPOF
CPLON[_CPF]=(_CPL) CPSETTYPE[_CPF]="TRAIL" CPLNUM[_CPF,_CPL]=(_CPLF) CPFRS[_CPF]="MCS"
RET
_CPOF:
IF ($P_TECCYCLE==TRUE)
  IF ($AA_CPSETTYPE[_CPF]<>"TRAIL") GOTOF _EXIT
ELSE
  IF ($PA_CPSETTYPE[_CPF]<>"TRAIL") GOTOF _EXIT
ENDIF
IF _CPL==NO_AXIS
  CPSETTYPE[_CPF]="TRAIL" CPOF=(_CPF)
ELSE
  CPSETTYPE[_CPF]="TRAIL" CPLOF[_CPF]=(_CPL)
ENDIF
_EXIT: RET

;* SysError 500918 therefore IF command commented
```

Figure 10-14 Cycle 700 after adaption. Changes are indicated by a colored bar.

10.6 Dynamic response of following axis

10.6.1 Parameterized dynamic limits

The dynamics of the following axis is limited with the following machine data values:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

MD32300 $MA_MAX_AX_ACCEL (Maximum axis acceleration)

10.6.2 Programmed dynamic limits

10.6.2.1 Programming (VELOLIMA, ACCLIMA)

Reducing or increasing dynamics limits

The dynamic limits of the following axis (FA) specified through MD32000 and MD32300 can be reduced or increased from the **part program**:

Command	Meaning
`VELOLIMA[FA]`	Reducing or increasing the maximum **Axis velocity**
`ACCLIMA[FA]`	Reducing or increasing the maximum **Axis acceleration**

The values specified during the programming of `VELOLIMA[FA]` and `ACCLIMA[FA]` are **process values**. They define the proportion with which the parameterized dynamic limits (MD32000 and MD32300) are to be considered:

Range of values	Meaning
1 ≤ value < 100	effects a **Reduction** in the dynamic limit
100 ≤ value < 200	effects an **increase** in the dynamic limit

The dynamic limits for the velocity and acceleration of the following axis are then calculated as follows:

Maximum axis velocity = MD32000 $MA_MAX_AX_VELO * VELOLIMA[FA]

Maximum axis acceleration = MD32300 $MA_MAX_AX_ACCEL * ACCLIMA[FA]

Note

The reduction / increase is effected on the overall dynamics of the axis, i.e., on the sum of the axis component from overlay and coupling.

Programming in synchronized actions

The possibility of programming `VELOLIMA[FA]` and `ACCLIMA[FA]` in **synchronized actions** depends on the coupling type:.

Coupling type	Part program	Synchronized actions
Tangential correction	x	
Coupled motion	x	x
Master value coupling	x	x
Electronic gearbox	x	
Synchronous spindle	x	
Generic coupling	x	x

Synchronization between following and leading axes

The acceleration characteristics set and the dynamics offsets set change the duration for synchronization between following and leading axes during acceleration operations as follows:

Dynamic offset	Activation
Dynamic reduction	Prolongs the synchronism difference. The monitoring from leading to following value may exceed the allowed range for extended periods.
Dynamic increase	Shortens the synchronism difference. The monitoring of leading and following values may exceed the allowed range for short periods.

Note

The user must restore the technological synchronization between machining and the synchronism difference.

Acceleration mode

Only `BRISKA`is available for the following axis, i.e., abrupt axis acceleration. Acceleration modes `SOFTA` and `DRIVEA`are not available for the following axes described.

Furthermore, it is also possible to configure the positions controller as a PI controller.

CAUTION

Application faults

This option can only be used in conjunction with servo trace and with the appropriate technical knowledge of the control.

References:

- CNC Commissioning Manual: NC, PLC, Drive
- Function Manual, Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)

POWER ON

During POWER ON the values of `VELOLIMA` and `ACCLIMA` are initialized to 100%.

Mode change

The dynamic offsets remain valid only on transition from AUTO → JOG mode.

RESET

The validities of the (`VELOLIMA` and `ACCLIMA`) dynamic offsets after RESET depend on the setting in the channel-specific machine data:

MD22410 $MC_F_VALUES_ACTIVE_AFTER_RESET (F Function is active even after RESET)

Value	Meaning
0	The values of `VELOLIMA[FA]` and `ACCLIMA[FA]` are set to 100% after RESET.
1	The last programmed values of `VELOLIMA[FA]` and `ACCLIMA[FA]` are also active after RESET.

This response also applies for dynamics offsets set using static synchronized actions. If this is not the case even when MD22410 = 0, then the IDS synchronized action should trigger a repeat or continuous writing of the dynamic offset.

References:
Function Manual, Synchronized Actions

10.6.2.2 Examples

Electronic gearbox

Axis 4 is coupled to X via an electronic gearbox coupling. The acceleration capability of the following axis is limited to 70% of maximum acceleration. The maximum permissible velocity is limited to 50% of maximum velocity. After POWER ON, the maximum permissible velocity is set to 100% again.

```
...
N120 ACCLIMA[AX4]=70
N130 VELOLIMA[AX4]=50                    ; reduced speed
N150 EGON(AX4,"FINE",X,1,2)
N200 VELOLIMA[AX4]=100                   ; full speed
...
```

Master value coupling

Axis 4 is coupled to X via a master value coupling. The acceleration capability of the following axis is limited to 80% of maximum acceleration.

```
...
N120 ACCLIMA[AX4]=80                     ; 80%
N130 LEADON(AX4,X,2)                     ; Activate coupling
...
```

Master value coupling with synchronized action

Axis 4 is coupled to X via a master value coupling. The acceleration response is limited to position 80% by static synchronized action 2 from position 100.

```
...
N120 IDS=2 WHENEVER $AA_IM[AX4] > 100 DO ACCLIMA[AX4]=80
N130 LEADON(AX4,X,2)
...
```

10.6.2.3 System variables

For geometry axis, channel axis, machine axis and spindle axis, the following readable system variables are available in the part program and synchronous actions:

Identifier	Data type	Description	Unit
Preprocessing			
$PA_ACCLIMA[n]	REAL	Acceleration offset set with ACCLIMA[Ax]	%
$PA_VELOLIMA[n]	REAL	Velocity offset set with VELOLIMA[Ax]	%
Main run			
$AA_ACCLIMA[n]	REAL	Acceleration offset set with ACCLIMA[Ax]	%
$AA_VELOLIMA[n]	REAL	Velocity offset set with VELOLIMA[Ax]	%

Note

Reading the main run variables, implicitly triggers a preprocessing stop.

10.7 General supplementary conditions

Note

Drive optimization

At a SINAMICS S120 drive unit, a maximum of 3 drives can be optimized or measured at the same time (speed controller optimization/function generator). Therefore, for a coupling with more than 3 coupled drives at the same time, we recommend that these are distributed over several drive units.

Note

Block search with active coupling

For an active coupling, it is recommended to only use block search type 5, "Block search via program test" (SERUPRO) for a block search.

10.8 Data lists

10.8.1 Machine data

10.8.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11410	SUPPRESS_ALARM_MASK	Screen form for suppressing special alarm outputs
11415	SUPPRESS_ALARM_MASK_2	Suppress alarm outputs
11660	NUM_EG	Number of possible electronic gears
11750	NCK_LEAD_FUNCTION_MASK	Functions for master value coupling
11752	NCK_TRAIL_FUNCTION_MASK	couple motion functions
18400	MM_NUM_CURVE_TABS	Number of curve tables (SRAM)
18402	MM_NUM_CURVE_SEGMENTS	Number of curve segments (SRAM)
18403	MM_NUM_CURVE_SEG_LIN	Number of linear curve segments (SRAM)
18404	MM_NUM_CURVE_POLYNOMS	Number of curve table polynomials (SRAM)
18406	MM_NUM_CURVE_TABS_DRAM	Number of curve tables in DRAM
18408	MM_NUM_CURVE_SEGMENTS_DRAM	Number of curve segments in DRAM
18409	MM_NUM_CURVE_SEG_LIN_DRAM	Number of linear curve segments (DRAM)
18410	MM_NUM_CURVE_POLYNOMS_DRAM	Number of curve polynomials in DRAM
18450	MM_NUM_CP_MODULES	Maximum number of allowed CP coupling modules
18452	MM_NUM_CP_MODUL_LEAD	Maximum number of allowed CP master values

10.8.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RE-SET/part program end
20112	START_MODE_MASK	Definition of control basic setting after run-up and RE-SET
22620	START_MODE_MASK_PRT	Definition of the control basic settings for special start
22621	ENABLE_START_MODE_MASK_PRT	Activation of MD22620
20900	CTAB_ENABLE_NO_LEADMOTION	Curve tables with jump of following axis
20905	CTAB_DEFAULT_MEMORY_TYPE	Default memory type for curve tables
21300	COUPLE_AXIS_1	Projection synchronous spindle pair

10.8.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30130	CTRLOUT_TYPE	Setpoint output type
30132	IS_VIRTUAL_AX	Axis is virtual axis
30455	MISC_FUNCTION_MASK	Axis functions
35040	SPIND_ACTIVE_AFTER_RESET	Own spindle RESET
37160	LEAD_FUNCTION_MASK	Functions for master value coupling
37200	COUPLE_POS_TOL_COARSE	Threshold value for "Coarse synchronous operation"
37202	COUPLE_POS_TOL_COARSE_2	Second synchronous operation monitoring: Threshold value for "Coarse synchronous operation"
37210	COUPLE_POS_TOL_FINE	Threshold value for "Fine synchronous operation"
37212	COUPLE_POS_TOL_FINE_2	Second synchronous operation monitoring: Threshold value for "Fine synchronous operation"
37220	COUPLE_VELO_TOL_COARSE	Velocity tolerance "coarse"
37230	COUPLE_VELO_TOL_FINE	Velocity tolerance "fine"
37500	ESR_REACTION	Reaction definition with extended stop and retract
37550	EG_VEL_WARNING,	Threshold value for speed warning threshold
37560	EG_ACC_TOL	Threshold value for the signal "Axis is accelerating"

10.8.2 Setting data

10.8.2.1 Axis/spindle-specific setting data

Number	Identifier: $SC_	Description
43100	LEAD_TYPE	Definition of master value type
43102	LEAD_OFFSET_IN_POS	Offset of master value with coupling to this axis
43104	LEAD_SCALE_IN_POS	Scaling of master value with coupling to this axis
43106	LEAD_OFFSET_OUT_POS	Offset of the function value of the curve table
43108	LEAD_SCALE_OUT_POS	Scaling of the function value of the curve table

10.8.3 System variables

Electronic gear (EG) and master value coupling

Identifier	Meaning
$AA_EG_ACTIVE	Coupling for leading axis b is active, i.e. switched on
$AA_EG_AX	Name for nth leading axis

Identifier	Meaning
$AA_EG_DENOM	Numerator of the coupling factor for leading axis b
$AA_EG_NUMERA	Numerator of the coupling factor for leading axis b
$AA_EG_NUMLA	Number of leading axes defined with EGDEF
$AA_EG_SYN	Synchronized position of leading axis b
$AA_EG_SYNFA	Synchronous position of following axis a
$AA_EG_TYPE	Type of coupling for leading axis b
$AA_IN_SYNC[FA]	Synchronization status of the following axis
$AA_LEAD_P	Current leading position value (modulo reduced).
$AA_LEAD_P_TURN	current leading value - position component lost as a result of modulo reduction.
$AA_LEAD_SP	simulated master value - position MCS
$AA_LEAD_SV	Simulated master value - velocity
$AA_LEAD_V	Current leading velocity value
$AA_SYNC	Coupling status of the following axis for leading value coupling.
$P_EG_BC	Block change criterion for EG activation calls: EGON, EGONSYN. WAITC = immediate synchronism fine or coarse and setpoint synchronous operation.
$VA_EG_SYNCDIFF	Synchronism difference

Generic coupling

Identifier	Meaning
$AA_ACCLIMA	Acceleration correction (main run) set with ACCLIMA
$AA_COUP_ACT	Coupling type of a following axis/spindle
$AA_COUP_CORR	Following spindle - correction value for tracking the deviation in synchronism
$AA_COUP_CORR_DIST	$AA_COUP_CORR distance-to-go
$AA_COUP_OFFS	Setpoint position offset
$AA_CPACTFA	Name of active following axis
$AA_CPACTLA	Name of active leading axis
$AA_CPBC	Block change criterion
$AA_CPDEFLA	Name of defined leading axis
$AA_CPFACT	Coupling type of the following axis/spindle.
$AA_CPFCMDPT	Axis setpoint position for all coupling components
$AA_CPFCMDVT	Axis setpoint speed for all coupling components
$AA_CPFMOF	Behavior of the following axis at switch-off
$AA_CPFMON	Behavior of the following axis at switching on
$AA_CPFMSON	Synchronization mode
$AA_CPFRS	Coupling reference system
$AA_CPLCMDP	Axis setpoint component of the leading axis
$AA_CPLCMDV	Axis setpoint speed component of the leading axis
$AA_CPLCTID	Table number of the active curve table
$AA_CPLDEN	Denominator of the coupling factor
$AA_CPLNUM	Numerator of the coupling factor

Identifier	Meaning
$AA_CPLSETVAL	Coupling reference of the leading axis
$AA_CPLSTATE	State of the coupling
$AA_CPSYNCOP	Threshold value of position synchronism "Coarse" (main run)
$AA_CPSYNCOV	Threshold value of velocity synchronism "Coarse" (main run)
$AA_CPSYNFIP	Threshold value of position synchronism "Fine" (main run)
$AA_CPSYNFIV	Threshold value of velocity synchronism "Fine" (main run)
$AA_CPLINSC	Scaling factor of the input value of a leading axis (main run)
$AA_CPLINTR	Offset value of the input value of a leading axis (main run)
$AA_CPLOUTSC	Scaling factor of the output value of the coupling (main run)
$AA_CPLOUTTR	Offset value of the output value of the coupling (main run)
$AA_CPLTYPE	Type of coupling
$AA_CPMRESET	State of the coupling after RESET
$AA_CPMSTART	State of the coupling after program start
$AA_CPNACTFA	Number of active following axes
$AA_CPNACTLA	Number of active leading axes
$AA_CPNDEFLA	Number of defined leading axes
$AA_CPSETTYPE	Preset coupling type
$AA_EG_ACTIVE	Coupling for leading axis b is active
$AA_EG_AX	Name for nth leading axis
$AA_EG_BC	Block change criterion
$AA_EG_DENOM	Denominator of the coupling factor
$AA_EG_NUMERA	Numerator of the coupling factor
$AA_EG_NUM_LA	Number of leading axes defined with EGDEF
$AA_EG_SYN	Synchronized position of the leading axis
$AA_EG_SYNFA	Synchronized position of the following axis
$AA_EG_TYPE	Type of coupling
$AA_IN_SYNC[FA]	Synchronization status of the following axis
$AA_JERKLIMA	Jerk correction set with JERKLIMA (main run)
$AA_LEAD_SP	Simulated master value - position with LEAD
$AA_LEAD_SV	Simulated master value - speed with LEAD
$AA_LEAD_P_TURN	Current leading value - position component lost as a result of modulo reduction.
$AA_LEAD_P	Current leading value - position (modulo reduced)
$AA_LEAD_V	Current leading velocity value
$AA_SYNC	Coupling status of following axis
$AA_SYNCDIFF[FA]	Synchronism difference of the setpoint
$AA_SYNCDIFF_STAT[FA]	Status of the synchronism difference of the setpoint
$AA_TYP/TYPE	Axis type
$AA_VELOLIMA	Velocity correction set with VELOLIMA (main run)
$PA_ACCLIMA	Acceleration correction set with ACCLIMA (pre-processing)
$PA_CPFACT	Coupling type of the following axis/spindle.
$PA_CPFPOSSTAT	Validity of synchronous and stop position

Identifier	Meaning
$PA_CPSYNCOP	Threshold value of position synchronism "Coarse" (pre-processing)
$PA_CPSYNCOV	Threshold value of velocity synchronism "Coarse" (pre-processing)
$PA_CPSYNFIP	Threshold value of position synchronism "Fine" (pre-processing)
$PA_CPSYNFIV	Threshold value of velocity synchronism "Fine" (pre-processing)
$PA_CPLINSC	Scaling factor of the input value of a leading axis (pre-processing)
$PA_CPLINTR	Offset value of the input value of a leading axis (pre-processing)
$PA_CPLOUTSC	Scaling factor of the output value of the coupling (pre-processing)
$PA_CPLOUTTR	Offset value of the output value of the coupling (pre-processing)
$PA_CPSETTYPE	Preset coupling type
$PA_JERKLIMA	Jerk correction set with JERKLIMA (pre-processing)
$PA_VELOLIMA	Velocity correction set with VELOLIMA (pre-processing)
$VA_COUP_OFFS[S2]	Actual-value position offset of the synchronous spindle
$VA_EG_SYNCDIFF	Synchronism difference
$VA_EG_SYNCDIFF_S	Synchronism difference with sign
$VA_SYNCDIFF[FA]	Synchronism difference of the actual value
$VA_SYNCDIFF_STAT[FA]	Status of the synchronism difference
$P_COUP_OFFS[S2]	Programmed position offset of the synchronous spindle
$P_EG_BC	Block change criterion

Dynamics of following axis

Identifier	Meaning
$AA_ACCLIMA	Main run acceleration correction set with ACCLIMA
$AA_VELOLIMA	Main run speed correction set with VELOLIMA
$PA_ACCLIMA	Preprocessing acceleration correction set with ACCLIMA
$PA_VELOLIMA	Preprocessing speed correction set with VELOLIMA

10.8.4 Signals

10.8.4.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override	DB31, ... DBX0.0-7	DB380x.DBB0
Axis disable	DB31, ... DBX1.3	DB380x.DBX1.3
Controller enable	DB31, ... DBX2.1	DB380x.DBX2.1
Activate handwheel	DB31, ... DBX4.0-2	DB380x.DBX4.0/1
Feed stop	DB31, ... DBX4.3	DB380x.DBX4.3
Enable following axis overlay	DB31, ... DBX26.4	DB380x.DBX5002.4
Synchronize following spindle	DB31, ... DBX31.4	DB380x.DBX5007.4

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Disable synchronization	DB31, ... DBX31.5	DB380x.DBX5007.5
Track synchronism	DB31, ... DBX31.6	DB380x.DBX5007.6
Reset synchronism correction	DB31, ... DBX31.7	DB380x.DBX5007.7

10.8.4.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Limiting of differential speed	DB31, ... DBX83.1	DB390x.DBX2001.1
Spindle in setpoint range, differential speed	DB31, ... DBX83.5	DB390x.DBX2001.5
Speed limit exceeded, total speed	DB31, ... DBX83.6	DB390x.DBX2001.6
Actual direction of rotation clockwise, total speed	DB31, ... DBX83.7	DB390x.DBX2001.7
Synchronous mode	DB31, ... DBX84.4	DB390x.DBX2002.4
Synchronism fine	DB31, ... DBX98.0	DB390x.DBX5002.0
Synchronism coarse	DB31, ... DBX98.1	DB390x.DBX5002.1
Actual value coupling	DB31, ... DBX98.2	DB390x.DBX5002.2
Overlaid movement	DB31, ... DBX98.4	DB390x.DBX5002.4
Velocity warning threshold	DB31, ... DBX98.5	DB390x.DBX5002.5
Acceleration warning threshold	DB31, ... DBX98.6	DB390x.DBX5002.6
Leading spindle active	DB31, ... DBX99.0	DB390x.DBX5003.0
Following spindle active	DB31, ... DBX99.1	DB390x.DBX5003.1
Synchronism correction implemented	DB31, ... DBX99.2	DB390x.DBX5003.2
Following axis accelerated	DB31, ... DBX99.3	DB390x.DBX5003.3
Synchronization in progress	DB31, ... DBX99.4	DB390x.DBX5003.4
Maximum velocity reached	DB31, ... DBX99.5	DB390x.DBX5003.5
Maximum acceleration reached	DB31, ... DBX99.6	DB390x.DBX5003.6
Synchronism correction is taken into account	DB31, ... DBX103.0	DB390x.DBX5007.0
Synchronism 2 fine	DB31, ... DBX103.4	DB390x.DBX5007.4
Synchronism 2 coarse	DB31, ... DBX103.5	DB390x.DBX5007.5

R3: Extended stop and retract 11

11.1 Brief description

The extended stop and retract function - subsequently called ESR - offers the possibility of flexibly responding when a fault situation occurs as a function of the process:

- **Extended stop**
  Assuming that the specific fault situation permits it, all of the axes, enabled for extended stopping, are stopped in an orderly fashion.
- **Retract**
  The tool currently in use is retracted from the workpiece as quickly as possible.
- **Generator operation (SINAMICS drive function "Vdc control")**
  If a parameterizable value of the DC link voltage is fallen below, e.g. because the line voltage fails, the electrical energy required for retraction is generated by recovering the braking energy of the drive intended for this purpose (generator operation).

ESR and active couplings

While stopping and retracting, active couplings are kept for a parameterizable time.

Retracting along a path

A straight line can be programmed as retraction path, as an alternative to purely axial retraction.

11.2 Control-managed ESR - 840D sl only

11.2.1 Extended stop and retract (ESR)

Using the Extended stop/retract (ESR)" function, axes that have been enabled for the function are stopped and retracted in a defined, delayed fashion. This is done to quickly separate the tool and workpiece in certain programmable system states.

In order that the energy required for the retraction motion is available in the drives involved – even when the power fails – then one or several drives can be parameterized as "generators" using the SINAMICS S120 "Vdc control" drive function. Even when the power fails, their kinetic energy is used to maintain the DC link voltage in order to permit NC controlled retraction motion.

Note

Detailed information on the SINAMICS S120 drive function "Vdc control" can be found in:

References
Function Manual SINAMICS S120 Drive Functions

NC-controlled responses

The function provides the following NC-controlled responses:

- "Extended stop"
  Programmable, defined, delayed path-related stopping of traversing motion
- "Retraction"
  Fastest possible retraction away from the machining plane to a safe retraction position to separate the tool and workpiece

The responses are independent of each other. Retraction operations and temporary continuation of axis couplings before stopping can be configured so that they are executed in parallel from a time perspective. In this case, an axis in generator mode (Vdc control) can maintain the DC link voltage.

Interaction of NC-controlled responses

The NC controlled responses are initiated via the **channel**-specific system variable $AC_ESR_TRIGGER.

Using $AC_ESR_TRIGGER, interpolatory stopping along the path or contour is possible. The NC-controlled retraction is performed in synchronism by the retraction axes in the channel.

For ESR-enabled axes, only precisely one channel may be assigned and it is not possible to switch between channels.

For NC-controlled stopping, an existing traversing motion as well as an active electronic coupling is maintained over an adjustable time (MD21380 $MC_ESR_DELAY_TIME1) even if there is an alarm with motion stop. After the parameterized time has expired, the axis is braked down to standstill along the programmed path.

In order to perform retraction outside the AUTOMATIC mode as well, triggering of this function is linked to the system variable $AC_ESR_TRIGGER. Retraction initiated via $AC_ESR_TRIGGER is locked, in order to prevent multiple retractions.

11.2.2 Drive-independent reactions

Generator operation

Generator operation is a drive function. Using the "Vdc control" function, the SINAMICS S120 drive unit can monitor the DC link group for undervoltage. When an adjustable voltage value is fallen below, then the drive intended for the purpose is switched into generator operation. The kinetic energy of the drive is used to buffer the DC link voltage. This allows the axes that are still moving to be stopped and retracted in an ordering fashion on the NC side.

Note

At the instant that it is switched into generator operation, if an axis is in closed-loop position control, then additional alarms can occur.

References

For detailed information on the SINAMICS S120 drive function "Vdc control", see:

Function Manual SINAMICS S120 Drive Functions

11.2.3 Power failure detection and bridging

DC link voltage limit values

The DC link is monitored against the limit values shown in the following diagram:

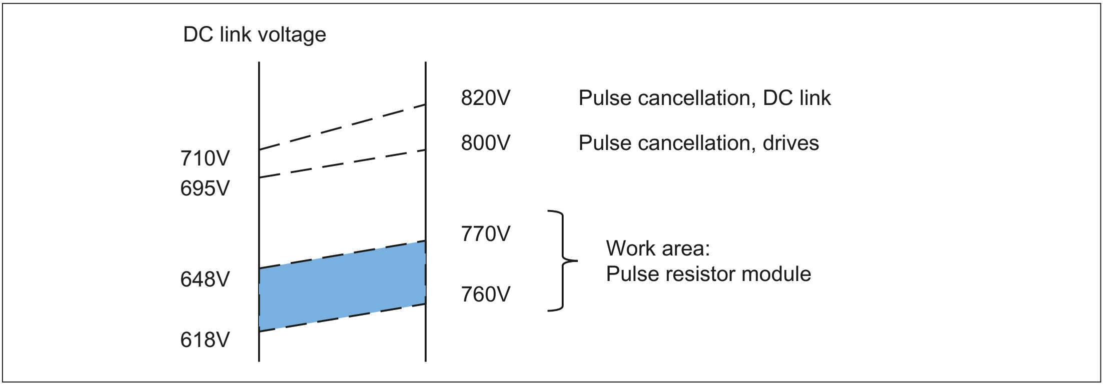


Figure 11-1 DC link voltage limit values

The drive and DC link pulses are cancelled at specific voltage levels, which means that the drives coast-down. If this behavior is not desired, the excess energy can be discharged using a resistor module. The operating range of the resistor module (shown highlighted in the diagram) lies below the critical voltage level.

Note

The pulse power of the resistor module is greater than the infeed power.

Monitoring the intermediate circuit minimum voltage limit

The DC link voltage can be monitored against a limit value that can be parameterized in the drive:

- p1248 (DC link voltage – lower voltage threshold)

When the limit value is fallen below, the following signal is set in the message word (MELDW) of the PROFIdrive telegram:

- MELDW.Bit4 = 1 (VDC_min controller is active (Vdc link < p1248))

The signal can be used in the NC to initiate ESR responses.

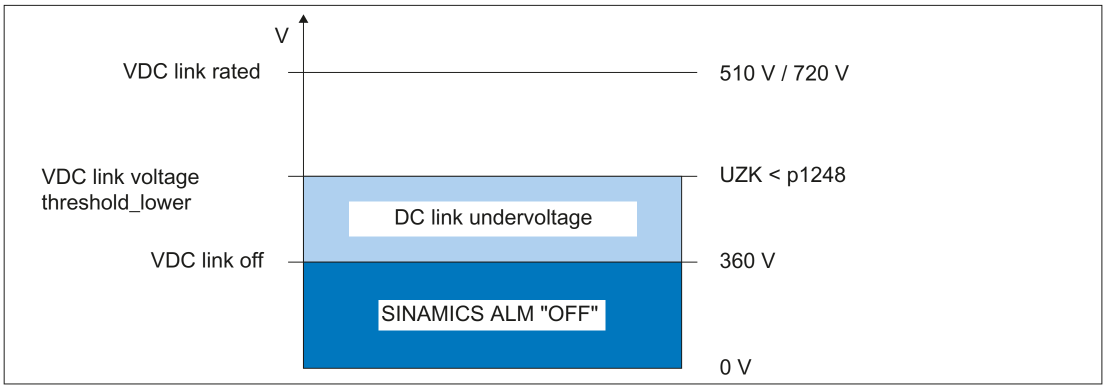


Figure 11-2 Monitoring the DC link voltage

11.2.4 NC-controlled extended stop

Axis-specific parameterization

The NC-controlled extended stopping for a machine axis is activated using machine data:

MD37500 $MA_ESR_REACTION[<axis>] = **22**

Note

Path axes

If NC-controlled extended stopping is parameterized for a **path axis**, then the corresponding behavior is also transferred to all other **path axes** of the channel.

Leading and following axis

During the extended stopping, a following axis of an electronic gearbox follows the leading axes according to the parameterized/programmed motion principle. That is why independent braking is not possible for the following axis during the times specified in MD21380 $MC_DELAY_TIME1 and MD21381 $MC_ESR_DELAY_TIME2. MD21380 $MC_DELAY_TIME1MD21381 $MC_ESR_DELAY_TIME2 A prerequsite for correct functioning of ESR is that all necessary enable signals must be set and remain set.

Channel-specific parameterization

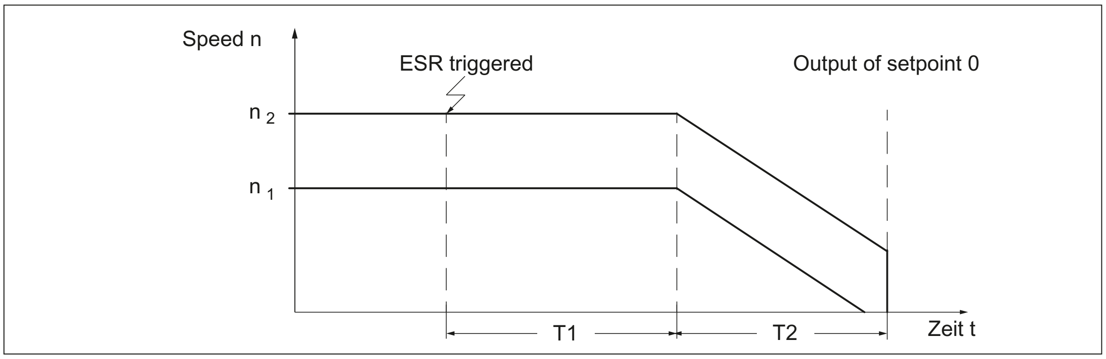
The timing within a channel is parameterized using the following machine data:

- MD21380 $MC_ESR_DELAY_TIME1[<axis>] (delay time ESR axes)
- MD21381 $MC_ESR_DELAY_TIME2[<axis>] (ESR time for interpolatory braking)

Time sequence

After initiating ESR, the axis continues to traverse with the actual speed. After a delay time has expired (..._TIME1), the axis is braked, interpolating.

The set time (..._TIME 2) is the maximum time available for interpolatory braking. After this time has expired, a setpoint of 0 is output for the axis, and the follow-up mode is activated.



T1	MD21380 $MC_ESR_DELAY_TIME1 (delay time, ESR axes)
T2	MD21381 $MC_ESR_DELAY_TIME2 (ESR time for interpolatory braking)
n_1	Actual speed, which could be braked to a standstill within T2
n_2	Actual speed, which could not be braked to a standstill within T2 ⇒ fast braking

Note

For safety reasons, the sum of T1 and T2 should not exceed a maximum value of approx. 1 second.

Precondition

To ensure that ESR is carried out, at least one of the axes involved in the movement must be parameterized as an NC-controlled retraction or stopping axis: MD37500 $MA_ESR_REACTION > 20

For axes that are not parameterized as NC controlled retraction or stopping axis, fast braking with subsequent tracking is realized immediately so that extended stopping starts ($AC_ESR_TRIGGER = 1).

Processing of all commands, especially those that result in an axis stop (e.g. Reset, Stop, Stopall), as well as the standard alarm responses STOPBYALARM and NOREADY, is delayed by the sum of the parameterized times:

Delay time = MD21380 $MC_ESR_DELAY_TIME1 + MD21381 $MC_ESR_DELAY_TIME2

An NC controlled stop is also active in conjunction with the "Electronic gearbox" function (see Chapter "Electronic gear (EG) (Page 482)"). It contains the (selective) switchover of the electronic gearbox to actual value coupling if there is a fault on the leading axes, and also maintains traversing motion and the enable signals during the delay time for ESR axes: MD21380 $MC_ESR_DELAY_TIME1

11.2.5 Retract

11.2.5.1 Function

Rapid lift to the position defined with `POLF` is programmed using modal command `LFPOS`.

Retraction motion parameterized with `LFPOS` or `POLF` for the axes programmed with `POLFMASK` or `POLFMLIN` replaces the traversing motion programmed for these axes in the NC program.

Regarding retraction motion, the following applies:

- The axes specified with `POLFMASK` independently traverse to the positions specified with `POLF`.
- The axes specified with `POLFMLIN` traverse to the positions specified with `POLF` in a **linear** relationship.

The extended retraction (i.e. `LIFTFAST`/`LFPOS` initiated through $AC_ESR_TRIGGER) **cannot be interrupted** and can only be terminated prematurely using EMERGENCY OFF.

Velocity and acceleration limits for the axes involved in the retraction are monitored during the retraction motion. The retracting movement takes place with BRISK, i.e. without jerk limitation.

Initiating the retraction

Retraction motion `LIFTFAST` is initiated in a channel, if, in this channel there is an axis for which retraction motion is enabled ($AA_ESR_ENABLE == 1) - and the trigger for fast retraction ($AC_ESR_TRIGGER = 1) is set.

Requirement

- The retraction position must be programmed in the NC program.
- The enable signals must be set and remain set for retraction motion.

11.2.5.2 Parameterization: Machine data

Activation

NC-controlled retraction for an axis is activated using machine data:

MD37500 $MA_ESR_REACTION = **21**

Delay times

Two different retraction times can be parameterized for the retraction:

- MD21380 $MC_ESR_DELAY_TIME1 = <delay time: Traversing motion>
- MD21381 $MC_ESR_DELAY_TIME2 = <delay time: interpolatory braking>

To perform retraction motion, the sum of the two delay times are available. After the total delay time has expired, fast braking is initiated for the retraction axes with a setpoint of 0 - with subsequent switchover into follow-up (tracking) operation.

Mirroring

The machine data is used to set, whether, for active "Mirroring" function (DB31, … DBX97.2), the retraction direction should be mirrored for "Fast retraction from the contour" (`LIFTFAST`):

MD21202 $MC_LIFTFAST_WITH_MIRROR = TRUE

The mirroring of the retraction direction only refers to the direction components perpendicular to the tool direction.

Stop behavior

The stop response for lift motion is set on a channel-for-general basis using the following machine data:

MD21204 $MC_LIFTFAST_STOP_COND, <bit> = <value>

Bit	Val-ue	Description
0	Response to the axis-specific NC/PLC interface signal DB31 DBB4.3 (feed stop) or context-sensitive interpolator stop.	
	0	Retraction motion is stopped for axis-specific feed stop or context-sensitive interpolator stop
	1	Retraction motion is not stopped for axis-specific feed stop or context-sensitive interpolator stop

Bit	Val-ue	Description
1	Response to the channel-spec. NC/PLC interface signal DB21 DBB6.0 (feed disable)	
	0	Retraction motion stop for feed disable in the channel
	1	No retraction motion stop for feed disable in the channel

Note

Influence of the interface signals

- The axis-specific NC/PLC interface signal DB31 DBB4.3 (feed stop) influences the complete retraction motion. All axis motion defined using `POLFMASK` and `POLFMLIN` are stopped.
- The NC/PLC interface signal, DB21 DBB7.3 (NC stop) has no effect on the retraction movement.

See also

Lift fast with linear relation of axes (Page 637)

11.2.5.3 Parameterization: System variable

Axis-specific function enable

An axis is enabled for retraction on an axis-for-axis basis using system variable:

$AA_ESR_ENABLE[<axis>] = 1

Channel-specific initiation

Retraction motion is initiated on a channel-for-channel basis using system variable:

$AC_ESR_TRIGGER = 1

Retraction is then realized in all channel axes for which the following applies:

$AA_ESR_ENABLE[<axis>] == 1 **AND** MD37500 $MA_ESR_REACTION[axis] == 21 **OR** 22

11.2.5.4 Programming (POLF, POLFA, POLFMASK, POLFMLIN)

Syntax

```
POLF(<axis>)=<position>
POLFA(<axis>,<type>,<position>)
POLFA(<axis>,<type>)
POLFMASK(<axis_1>,<axis_2>,...)
POLFMLIN(<axis_1>,<axis_2>,...)
```

Meaning

`POLF`:	Address for specifying the target position of the retraction axis `POLF` is modal.			
	`<axis>`:	Channel axis name of the retraction axis		
	`<position>`:	Retraction position		
		Type:	REAL	
		The geometry axis is retracted in the WCS. For all other channel axes in the MCS. With the same identifiers for geometry and channel/machine axis, retraction is in the WCS.		
`POLFA`:	Predefined subprogram call for the specification of the retraction position of single axes			
	`<axis>`:	Channel axis name of the retraction axis		
	`<type>`:	Position specification mode		
		Type:	INT	
		Value:	0	Mark position value as invalid
			1	Position value is absolute
			2	Position value is incremental (distance)
		Note: If an axis is not a single axis or if the type is missing or type=0, then a corresponding alarm is output.		
	`<position>`:	Retraction position (see above) (**optional**)		
		Note: The position value is also accepted with type=0. Only this value is marked as invalid and has to be reprogrammed for retraction.		
`POLFMASK`:	Predefined subprogram call for selection of the axes that are to be retracted after tripping of rapid lift **independently of one another**.			
	`<axis_1>,…`:	Names of the axes that are to be traversed to their positions defined with `POLF` during rapid lift. All the axes specified must be in the same coordinate system.		
	`POLFMASK()` without specification of an axis deactivates the rapid lift for all axes that have been retracted independently of one another.			
`POLFMLIN`:	Predefined subprogram call for selection of the axes that are to be retracted after tripping of rapid lift **in linear relation**.			
	`<axis_1>,…`:	See above.		
	`POLFMLIN()` without specification of an axis deactivates the rapid lift for all axes that have been retracted in linear relation.			

Note

Before rapid retraction to a fixed position can be enabled via `POLFMASK` or `POLFMLIN`, a position must have been programmed with `POLF` for the selected axes.

Note

If axes are enabled one after the other with `POLFMASK`, `POLFMLIN` or `POLFMLIN`, `POLFMASK`, then the last definition always applies for the particular axis.

Note

The positions programmed with `POLF` and the activation by `POLFMASK` or `POLFMLIN` are deleted when the part program is started. This means that the user must reprogram the values for `POLF` and the selected axes in `POLFMASK` or `POLFMLIN` in each part program.

Note

If, when using the abbreviated form `POLFA` only the type is changed, then the user must ensure that either the retraction position or the retraction path contains a practical and sensible value. In particular, the retraction position and the retraction path have to be set again after a warm restart.

Example: Retracting an individual axis

Program code	Comment
`MD37500 $MA_ESR_REACTION[AX1]=21`	`; NC-controlled retraction.`
`...`	
`$AA_ESR_ENABLE[AX1]=1`	
`POLFA(AX1,1,20.0)`	`; AX1 is assigned the axial retraction position 20.0 (absolute).`
`$AA_ESR_TRIGGER[AX1]=1`	`; Retraction starts from here.`

11.2.5.5 Boundary conditions

General supplementary conditions

Exceptions

Retraction and/or rapid lift is **not** executed for the following axes:

- Axes, which are not permanently assigned a channel
- Axes, which are in the open-loop speed-controlled mode (spindles)
- Axes, which are interpolated as positioning spindles (SPOS/SPOSA)

Modulo rotary axes

Modulo rotary axes respond to rapid lift as follows:

- For incremental programming of the target position, the latter is approached without modulo offset.
- With absolute programming, the target position is approached time-optimized with the use of modulo offsets. This is almost identical to positioning via the shortest path.

Utilizing the axis dynamics

The retraction movement is interpolated linearly, using the maximum acceleration and speed of the axes involved in `POLFLIN`.

Parallel retraction

Only **one linear** retraction is permitted in each **channel**. This means that multiple axis groups, which approach their retraction positions linearly, cannot be created in the channel.

In **parallel** with linear retraction, additional axes can also use `POLFMASK` for independent axis-specific retraction movement to their programmed retraction positions.

Concurrent retraction

If axes are used in both `POLFMASK` and in `POLFMIN`, it should be noted that the last state programmed is always active for retracting movement. This means that an axis previously activated with `POLFMIN` is removed from the linear relationship following programming in `POLFMASK` and the retracting movement would then take place as an independent movement (see the examples in Chapter "Lift fast with linear relation of axes (Page 637)").

Effectiveness of retraction parameters

The parameters valid at the **triggering time** are decisive for the retraction movement. If one of these parameters (`POLF`, `POLFMASK`, `POLFMLIN`, Frame, etc.) changes during the retracting movement (e.g. due to a block change), this change does not affect the retraction movement that has already started.

Supplementary conditions: Path axes

Alternative extended stopping

If, for a path axis ESR_REACTION=21 (NC-controlled retraction) is configured and enabled with $AC_ESR_ENABLE=1, then ESR_REACTION=22 (extended stopping) is active for all path axes for which no ESR_REACTION=21 is configured or enabled.

Rapid retraction/lift without retraction motion enabled

If, for an axis, ESR_REACTION=21 is configured and enabled with $AC_ESR_ENABLE=1, but e.g. no retraction motion is enabled with `POLFMASK`, then, for this axis, ESR_REACTION=22 is active (extended stopping).

Behavior for different enable signals

"NC-controlled retraction" is configured for path axes X and Y:

- MD37500 $MA_ESR_REACTION[X] = 21
- MD37500, $MA_ESR_REACTION[Y] = 21

1. "Extended stop and retract" and retraction motion is enabled for both path axes:
   - $AA_ESR_ENABLE(X) = 1
   - $AA_ESR_ENABLE(Y) = 1
   - POLFMASK(X,Y)

   For rapid lift, X and Y execute the programmed retraction motion.
2. "Extended stop and retract" is only enabled for one path axis, but retraction motion is enabled for both path axes:
   - $AA_ESR_ENABLE(X) = 0
   - $AA_ESR_ENABLE(Y) = 1
   - POLFMASK(X,Y)

   For rapid lift, for axis X, due to the path interrelationship, in spite of the fact that it is not enabled, "extended stopping" corresponding to ESR_REACTION = 22 is executed.
   Axis Y executes the programmed retraction motion.
3. However, for both path axes, "Extended stop and retract" is enabled – but retraction motion only for one path axis:
   - $AA_ESR_ENABLE(X) = 1
   - $AA_ESR_ENABLE(Y) = 1
   - POLFMASK(Y)

   For rapid lift, for axis X, due to the path interrelationship, in spite of the missing enable signal, retraction motion "extended stopping" corresponding to ESR_REACTION = 22 is executed.
   Axis Y executes the programmed retraction motion.

11.2.6 Trigger sources

A distinction must be made between the ESR trigger sources on a user-for-user basis by evaluating system variables. This is the reason that all system variables are available, which can be read in **synchronized actions**.

General trigger sources

- Digital inputs of the NCU module or the internal control image of digital outputs that can be read back: $A_IN, $A_OUT
- Channel status: $AC_STAT
- NC/PLC interface signals

- System variables written from the PLC: $A_DBB, $A_DBW, $A_DBD
  It is **not** recommended to use these system variables, written by the PLC for **time-critical** signals, as in this case, the PLC cycle time is included in the total response time. Nevertheless, it is an appropriate way for the PLC to influence the sequence or release of the extended stop and retract function. However, linking-in PLC states in such a fashion does make sense if these are **exclusively** input from the PLC (e.g. Emergency-Stop, reset button, stop button)
- Group signals from alarms: $AC_ALARM_STAT

Axial trigger sources

- $VA_SYNCDIFF[<following axis>] (synchronism difference, on the actual value side)
- $AA_ESR_STAT (axial return signal word: ESR status)

Alarm responses

When ESR is active, the alarm responses NOREADY and STOPBYALARM are delayed by one IPO cycle. The self-clearing alarm is displayed to indicate this delay:
Alarm 21600 "Monitoring for ESR active".

Note

The display of this alarm can be suppressed by:

MD11410 $MN_SUPPRESS_ALARM_MASK, bit16 = 1

11.2.7 Logic gating functions: Source and reaction linking

The flexible logic operation possibilities of the **static synchronized actions** can be used to trigger specific reactions based on sources. Linking all relevant sources using static synchronized actions is the responsibility of the user/machine manufacturer. They can selectively evaluate the source system variables as a whole or by means of bit masks, and then make a logic operation with their desired reactions. The static synchronous actions are effective in all operating modes. A detailed description of how to use synchronous actions can be found in:

References:

- Function Manual, Synchronized Actions
- Programming Manual Work Preparation (Synchronized Actions, System Variables)

Linking of axial sources with global or channel specific sources can be configured variably with the aid of $AA_TYP (axis type).

11.2.8 Activation

Option

The function "Extended stop and retract" is an option.

Axis-specific function enable ($AA_ESR_ENABLE)

The axis-specific function enable is realized using the system variable:

$AA_ESR_ENABLE[<axis>] = 1

Axis-specific enable for extended stopping

An axis is enabled for extended stopping with:

MD37500 $MA_ESR_REACTION[axis] = 22

Axis-specific enable for retraction

An axis is enabled for retraction using:

MD37500 $MA_ESR_REACTION[axis] = 21

Channel-specific trigger ($AC_ESR_TRIGGER)

ESR is triggered on a channel-for-channel basis by setting the following system variable:
$AC_ESR_TRIGGER = 1

ESR is then realized in all channel axes for which the following applies:

$AA_ESR_ENABLE[<axis>] == 1 **AND** MD37500 $MA_ESR_REACTION[axis] == 21 **OR** 22

11.2.9 Configuring aids for ESR

Voltage failure

For the following modules, when the line voltage fails, the power supply must be maintained using suitable measures at least until it is ensured that the axes have come to a stop:

- SINUMERIK 840D sl NCU 7x0
- SINUMERIK NCU I/O modules
- SIMATIC PLC I/O modules
- SINAMICS drive system S120 (booksize)

DC link energy

The energy available in the DC link of the drive units when the line supply fails is calculated as follows:

$E = 1/2 * C * (\text{VDC link}_{alarm}^2 - \text{VDC link}_{min}^2)$

E:	Energy in Wattseconds [Ws]
C:	Total capacity of intermediate circuit in Farad [F]
VDClink_{alarm}:	DC link voltage below which an undervoltage is identified. SINAMICS Parameter: r0296 + p0279
VDClink_{min}:	Lower limit of the DC link voltage for safe and reliable operation, taking into account the motor-specific EMF. **Note** $\text{VDC link}_{min} > \text{VDC link}_{off}$

This energy is available for a minimum time of t_{min}:

$t_P = E / P_n * \eta$

E:	Energy in Wattseconds [Ws]
t_P:	Buffer time in milliseconds [ms]
P_n:	Power in kilowatt [kW]
η:	Efficiency of the drive unit

Example

Assumptions:

- P_n = 16 kW, infeed (ALM) with 3-ph. 380 VAC
- $C = C_{is} - 20\% = 20{,}000\ \mu F - 20\% = 16{,}000\ \mu F = 16 * 10^{-3}$ F, for safety, a 20% lower effectively available capacity is assumed.
- VDC link_{alarm} = 550 V
- VDC link_{min} = 350 V

$E = 1/2 * 16*10^{-3}\ F * ((550\ V)^2 - (350\ V)^2) = 1440\ Ws$

This energy is available for a time t_{min}, e.g. for a retraction:

- E = 1440 Ws
- P_n = 16 kW
- η = 0.9 (assumption)

$t_{Pmin} = 1440\ Ws / 16\ kW * 0.9 = 81\ ms$

The following table shows a summary of the values for various SINAMICS infeeds (ALMs). Whereby, the nominal (C_{max}) and minimum (C_{min}) capacitance are taken into account.

The total capacitance of the DC link available is made up, taking into account the respective maximum capacitance C_{max} (charge limit), from the capacitances of the infeed (ALM), the motor modules as well as any capacitor modules.

The capacitance specified in the table for the minimum energy content (E_{min} for C_{min}) takes into account a component tolerance of -20% (worst case).

Note

It is recommended that the SIZER configuration tool be used to determine the total capacitance of the DC link available.

Table 11-1 SINAMICS infeeds (ALMs): Nominal and minimum buffer times

ALM P_{max} [kW]	Max. capacitance C_{max}[µF]	Energy content E_{max} for C_{max} [Ws]	Energy content E_{min} for C_{min} [Ws]	Backup time t_{Pmax} for P_{max} [ms]	Buffer time t_{Pmin} for P_{max} [ms]
16	20000	1800	1440	101,25	81,00
36	20000	1800	1440	45,00	36,00
55	20000	1800	1440	29,46	23,56
80	20000	1800	1440	20,25	16,20
120	20000	1800	1440	13,50	10,80

Energy balance

When configuring retraction motion, it is always necessary to consider the energy flow (balance) to find out whether an additional capacitor module or a generator axis (with correspondingly dimensioned flywheel effect) is required - or not.

Stopping as energy supply

From approx. the third interpolator clock cycle, the speed setpoints of the configured stopping or retraction axes change. After this time the braking phase starts (if no drive independent shutdown is projected for this axis).

As soon as the braking process is initiated, the energy released in this manner is available for retraction motion. An energy balance must be used to ensure that the kinetic energy of the braking axes is sufficient for retraction.

The energy balance shows the maximum setting for the interpolator clock cycle time, which will allow a safe emergency retraction to be executed.

Example

With a 16 kW unit under maximum load and minimum intermediate circuit capacity it should be possible to execute an emergency retraction without generator operation. Interpolator clock cycle time may theoretically not be more than 4.86 ms, i.e. in this case a maximum of 4 ms could be set.

If necessary, a more powerful NCU must be chosen in order to achieve optimal conditions.

Generator operation

For cases in which the DC link energy is not sufficient for safe and reliable retraction (minimum 3 IPO cycles), generator operation can be configured for a drive using the SINAMICS "Vdc control" function. In this case, the mechanical energy of an axis is fed back into the DC link.

The energy stored in an axis can be calculated as follows:

$E = 1/2 * J_{tot} * \omega^2$

J_{tot}	Total moment of inertia [kg*m^2]
ω	Angular velocity at the time of switchover to generator operation [s^{-1}]

This energy is fed back into the DC link with an efficiency of approx. 90%:

When using infeeds with a high power rating (55, 80, 120 kW) for generator operation, it is recommended to use a dedicated axis with additional flywheel effect. For this axis, after the drive has accelerated to the rated speed, energy is only required to compensate for friction losses.

However, in principle, any axis can be used for generator operation, if this is not directly involved in extended stopping and retraction or in couplings that must be specifically maintained.

In order to prevent that the DC link voltage does not become too high when using generator operation (limit value: pulse cancelation, drive) and as a consequence that the pulses are canceled for the axes, adequately dimensioned pulsed resistor modules must be used or the SINAMICS function "Vdc control, overvoltage monitoring".

11.2.10 Control system response

11.2.10.1 Axis behavior depending on the enable signals

Systematic

If an axis is configured as NC-controlled retraction axis:

MD37500, $MA_ESR_REACTION[*<axis>*] = 21

The following enable signals must be available to execute retraction motion in the case of ESR:

- Axial ESR enable using system variable: $AA_ESR_ ENABLE[<axis>] = 1
- Axial retraction enable using a part program command: `POLFMASK(<axis>)`

If one of the two enable signals is not available, then in the case of ESR, the axis responds corresponding to the following table:

Enable signals	Response in the case of ESR
• $AA_ESR_ ENABLE[<axis>] = 1 • No enable via `POLFMASK`	For the axis, implicit NC-controlled stopping becomes active, corresponding to MD37500, $MA_ESR_REACTION[<axis>] = 22.
• $AA_ESR_ ENABLE[<axis>] = 0 • Enable via `POLFMASK(<axis>)`	Axis is a path axis: For the axis, the ESR response is implicitly changed to NC-controlled stopping. The response corresponds then to a configuration of MD37500, $MA_ESR_REACTION[<axis>] = 22
	Axis is a special axis: The axis is stopped with rapid stop.

Examples for axes with path relationship

For the subsequent examples, two path axes X and Y are assumed, which are configured as NC-controlled retraction axes and are programmed for the retraction positions:

- MD37500, $MA_ESR_REACTION[X] = 21
- MD37500, $MA_ESR_REACTION[Y] = 21
- `POLF[X]=<retraction position> LFPOS`
- `POLF[Y]=<retraction position> LFPOS`

Initial situation	Response in the case of ESR
• $AA_ESR_ ENABLE[X] = 1 • $AA_ESR_ ENABLE[Y] = 1 • `POLFMASK(X,Y)` • Program traversing motion: Y	Retraction motion in X and Y.
• $AA_ESR_ ENABLE[X] = 0 • $AA_ESR_ ENABLE[Y] = 1 • `POLFMASK(X,Y)` • Program traversing motion: X and Y	X does not have ESR enable. As X is a path axis, for X, the ESR response is implicitly changed to 22 (stopping). This is the reason that X continues to traverse corresponding to the programmed traversing motion. Y executes retraction motion. After the delay time for ESR axes expires, MD21380, $MC_ESR_DELAY_TIME1, X is stopped with the path acceleration.
• $AA_ESR_ ENABLE[X] = 1 • $AA_ESR_ ENABLE[Y] = 1 • `POLFMASK(Y)` • Program traversing motion: X	X has no retraction enable (`POLFMASK`). X does not execute any retraction motion. As X is a path axis, for X, the ESR response is implicitly changed to 22 (stopping). This is the reason that X continues to traverse corresponding to the programmed traversing motion. Y executes retraction motion. After the delay time for ESR axes expires, MD21380, $MC_ESR_DELAY_TIME1, X is stopped with the path acceleration.
• $AA_ESR_ ENABLE[X] = 1 • $AA_ESR_ ENABLE[Y] = 1 • `POLFMASK(X)` • Program traversing motion: X and Y	Y has not been enabled for retraction (`POLFMASK`). see lines above, syntax Y does not execute retraction motion. As Y is a path axis, for Y, the ESR response is implicitly changed to 22 (stopping). This is the reason that Y continues to traverse corresponding to the programmed traversing motion. X executes retraction motion. After the delay time for ESR axes expires, MD21380, $MC_ESR_DELAY_TIME1, Y is stopped with the path acceleration.

Examples for axes without path relationship

For the subsequent examples, path axes X and a command axis B are assumed, which are configured as NC-controlled retraction axis:

- MD37500, $MA_ESR_REACTION[X] = 21
- MD37500, $MA_ESR_REACTION[B] = 21

Initial situation	Response in the case of ESR
• $AA_ESR_ ENABLE[X] = 1 • $AA_ESR_ ENABLE[B] = 0 • `POLFMASK(X)` • Program traversing motion: X and B	B is not enabled for ESR. As B is not a path axis, B is immediately stopped with a rapid stop. X executes retraction motion.
• $AA_ESR_ ENABLE[X] = 1 • $AA_ESR_ ENABLE[B] = 1 • `POLFMASK(X)` • Program traversing motion: X and B	B is not enabled for retraction (`POLFMASK`). However, as B is enabled for ESR, for B, the ESR response is implicitly changed to 22 (stopping). This is the reason that B continues to traverse corresponding to the programmed traversing motion. Y executes retraction motion. After the delay time for ESR axes expires, MD21380, $MC_ESR_DE-LAY_TIME1, B is stopped with axial acceleration.

11.2.10.2 POWER OFF/POWER ON

If the user-specific retraction logic is programmed within motion synchronized actions, then it must be observed that this is still not active when the control boots (POWER ON). A retraction logic, which should be active after POWER ON, must therefore lie in an ASUB started from the PLC or must be called event-controlled.

References

For detailed information about event-driven program calls see:

Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1), Section: Eventcontrolled program calls

11.2.10.3 Operating mode change, NC Stop, reset

Static synchronized actions (IDS keyword) can be used for user-specific retraction logic as the following system states or state change have no influence on this:

- Operating mode change, e.g. DB11 DBX0.0 (AUTOMATIC)
- NC stop, e.g. DB21, ... DBX7.3 (NC stop)
- Reset, e.g. DB21, ... DBX7.7 (reset)

Note

NC stop and command axes

Traversing command axes is interrupted for an NC stop.

Note

POLF/POLFMASK reset and program commands

For reset, the programmed absolute retraction positions (POLF) and the enable signals for the retraction axes (POLFMASK) are not deleted.

11.2.10.4 Part program start, NC start

In order that a defined initial state is available when a part program starts, the programmed absolute retraction positions and the enable signals of the retraction axes are deleted when the part program starts.

Note

Retraction positions and enable signals

When a part program starts, the absolute retraction positions (`POLF`) and the enable signals of the retraction axes (`POLFMASK`) must be reprogrammed corresponding to the actual requirements.

11.2.10.5 Alarm behavior

The alarm response depends on the axis in which a fault occurs:

- Fault in an axis outside the axis grouping of an electronic gear:
  This axis switches off "normally". Extended stopping and retraction continue to operate as before - or are initiated by such a fault.
- Error in a leading axis (LA):
  selective switchover to actual value linkage already during stop, otherwise as previously.
- Error in a following axis (FA):
  - Carry out retract: Retraction axis may not be a following axis, that is, no conflict.
  - Carry out stop: The following axis may react with uncontrollable behavior. Saving the workpiece/tool must be left to the retraction; however, the stop should not disrupt the process any further.

- Error in the retraction axis: There is no retraction.
- Emergency Stop
  An Emergency Stop is not a fault from a control system point of view, rather the response is the same as for any other control signal. For safety reasons, Emergency Stop interrupts the interpolation and all traversing motion, and also cancels the electronic coupling by withdrawing the controller enable signals.
  In applications where the coupling and traversing movements must remain valid after Emergency Stop, the **PLC** must **delay** the Emergency Stop long enough for the required NC or drive-end reactions to terminate.
  The following interface signal is available as feedback signal to the PLC:
  IS "ESR reaction not initiated" DB31, ... DBX98.7

If an alarm with cross-channel NOREADY reaction is issued during the active phase of the ESR (i.e. NOREADY | NCKREACTIONVIEW | BAGREACTIONVIEW), then ESR is triggered in **all** channels.

11.2.10.6 Block search, REPOS

Extended stop and retract does not affect block search or REPOS motions.

11.3 ESR executed autonomously in the drive

11.3.1 Fundamentals

Function

Drive-autonomous extended stop and retract (ESR) enables the fast separation of workpiece and tool independent of the higher-level control (NC).

For this purpose, the following axis-specific functions can be configured in the drive:

- Generator operation
- Extended stop
- Retract

The drive-autonomous responses are automatically initiated in error situations. The triggering of the drive-autonomous responses can also be realized user-specific via the part programs or synchronized actions from the higher-level control system.

As the stopping and retraction motion of the drive-autonomous ESR are purely axis-specific, in contrast to control-managed ESR, couplings are not taken into account.

ESR executed autonomously in the drive can also be used in combination with control-managed ESR. This also allows stopping and retraction motion in the drives if they are no longer specified from the control, e.g. as a result of a communication failure.

Note

DRIVE-CLiQ

For possibly occurring DRIVE-CLiQ errors, in order to keep the reaction to the functions of the drive-autonomous ESR as low as possible, within the scope of ESR, we recommend that the motor modules involved are not connected through a line-type topology, but instead are directly connected to the Control Unit (CU).

Link axes

The "Drive-autonomous extended stop and retract (ESR)" function cannot be used in conjunction with link axes.

Requirements

The following conditions must be fulfilled:

- SINAMICS and SINUMERIK software version: ≥ V4.4
- Servo drive object, PROFIdrive telegram: 102 - 199
- Servo drive object, function module "ESR" active
  The function module "Extended stop and retract (ESR)" must be activated via the drive wizard of SINUMERIK Operate. For a description of the drive wizard, see: SINUMERIK Operate online help.
- Servo drive object, generator operation (Vdc control) active in a drive
- Control unit drive object, PROFIdrive telegram: 390, 391
- The 24 V power supply for the modules is buffered via CSM or UPS

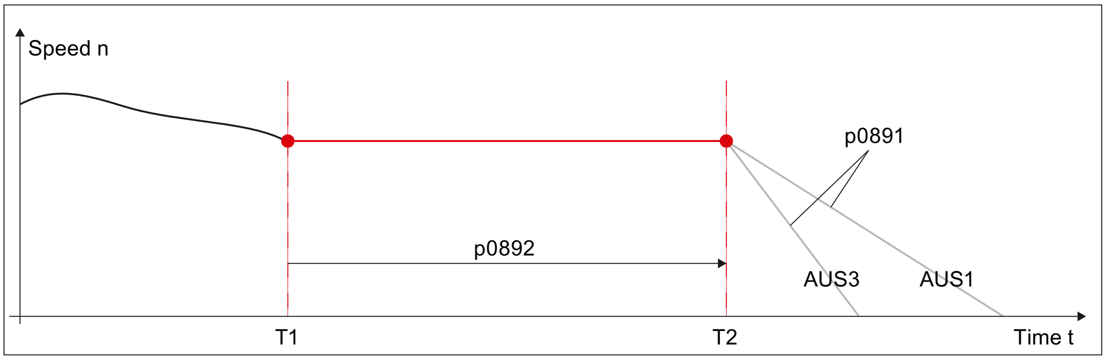
11.3.2 Configuring stopping in the drive

Drive integrated shutdown is configured via the following drive parameters:

Parameter	Description	
p0888	ESR: Configuration	
	Value	**Meaning**
	1	Extended stopping (function integrated in the drive)

Parameter	Description	
p0891	ESR: Off ramp	
	Value	**Meaning**
	0	OFF3 (default) The drive is braked along the OFF3 down ramp by immediately entering n_set = 0 (p1135: OFF3 ramp-down time).
	1	OFF1 By immediately entering n_set = 0, the drive is braked along the ramp generator down ramp (p1121: ramp-function generator ramp-down time).

Parameter	Description
p0892	ESR: Timer
	The drive travels at a constant velocity for the configured time with the speed setpoint that was present when the fault occurred.



T1	Time at which stopping was triggered
T2	Instant in time after the time that was configured in p0892 has expired
OFF3/OFF1	Braking ramp as a function of p0891

Figure 11-3 Response to drive-integrated stopping

Feedback signal

The stop status is returned to the control (see Section "Feedback of the ESR status (Page 625)").

References

For a detailed description of drive parameters, refer to:

SINAMICS S120/S150 Parameter Manual

11.3.3 Configuring retraction in the drive

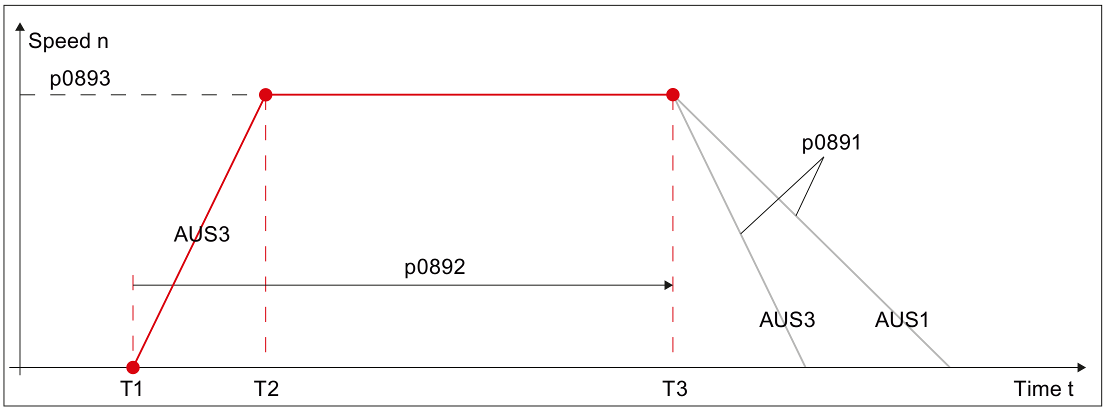
Drive integrated retraction is configured using the following drive parameters:

Parameter	Description	
p0888	ESR: Configuration	
	Value	**Meaning**
	2	Extended retraction (function integrated in the drive)

Parameter	Description	
p0891	ESR: Off ramp	
	Value	**Meaning**
	0	OFF3 (default) The drive is braked along the OFF3 down ramp by immediately entering n_set = 0 (p1135: OFF3 ramp-down time).
	1	OFF1 By immediately entering n_set = 0, the drive is braked along the ramp generator down ramp (p1121: ramp-function generator ramp-down time).

Parameter	Description
p0892	ESR: Timer
	The parameter specifies the total time that elapses to reach the speed specified in p0893 followed by constant velocity travel. This is followed by an OFF1 or OFF3 ramp depending on the parameterization in p0891.

Parameter	Description
p0893	ESR: Speed
	The parameter specifies the speed (retraction speed) that is reached when initiating retraction via an OFF3 ramp.



T1	Instant in time that retraction was initiated
T2	Instant in time when the retraction speed specified in p0893 is reached
T3	Instant in time after the time that was configured in p0892 has expired

Figure 11-4 Behavior for drive-integrated retraction

Feedback signal

The retract status is returned to the control (see Section "Feedback of the ESR status (Page 625)").

References

For a detailed description of drive parameters, refer to:

SINAMICS S120/S150 Parameter Manual

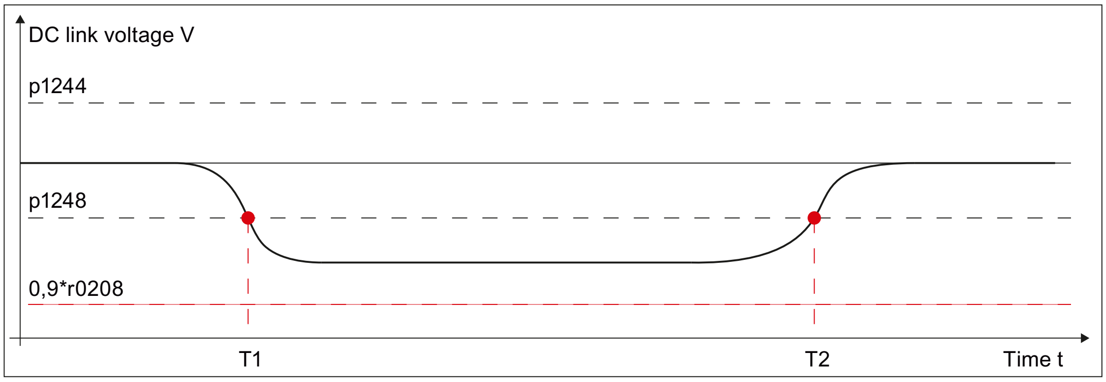
11.3.4 Configuring generator operation in the drive

The generator mode for the drive-autonomous ESR is configured using the following drive parameters:

Parameter	Description	
p0888	ESR: Configuration	
	Value	**Meaning**
	3	Generator operation (Vdc controller)

Parameter	Description	
p1240	Vdc controller or Vdc monitoring configuration	
	Value	**Meaning**
	2	When reaching the lower DC link voltage threshold (p1248) - for ESR - the motor is braked in order to utilize its kinetic energy to buffer the DC link.

Parameter	Description
p1248	Lower DC link voltage threshold
	Setting the lower threshold for the DC link voltage. For p1240 = 2, this threshold is used as setpoint limit for the Vdc_min controller.



p1244	Upper DC link voltage threshold
p1248	Lower DC link voltage threshold
r0208	Rated power unit line supply voltage. The pulses are canceled if the DC link voltage drops below 0.9*r0208
T1	Instant in time at which generator operation was triggered
T2	Instant in time at which generator operation was exited

Figure 11-5 Configuring_drive_integrated_generator_operation

Generator minimum speed

The lower limit of the motor speed of the generator axis is configured using drive parameter p2161:

Parameter	Description
p2161	Speed threshold value 3 Speed threshold value for the message: |n_act| < speed threshold value 3

Feedback signal

The generator operation status is returned to the control (see Section "Feedback of the ESR status (Page 625)").

References

A detailed description of drive parameters and the Vdc control can be found in:

- SINAMICS S120/S150 List Manual
- SINAMICS S120 Function Manual; Section "Servo control" > "Vdc control"

11.3.5 ESR is enabled via system variable

The ESR response of an axis, configured via drive parameter, must be programmed on a user-specific basis in a part program/synchronized action using the following axis-specific system variable:

$AA_ESR_ENABLE[<axis>]	
Value	**Meaning**
1	ESR responses enabled in the drive
0	ESR responses inhibited in the drive

11.3.6 Triggering ESR via system variable

Triggering ESR responses, configured via drive parameter and enabled via $AA_ESR_ENABLE[<axis>], must be programmed on a user-specific basis in a part program/synchronized action via the following NC-specific system variable:

$AN_ESR_TRIGGER	
Value	**Meaning**
1	trigger drive-integrated ESR reactions that have been released
0	do not trigger drive-integrated ESR reactions that have been released

Note

The trigger signal must be active in the part program/synchronized action for at least 2 IPO cycles to ensure that the PROFIdrive telegram is transferred to the drive.

Mapping in the drive unit

System variable $AN_ESR_TRIGGER is mapped to drive parameter p0890.0. Interconnecting drive parameter p0890.0 with the corresponding information in telegram 390 or 391 to the Control Unit is realized automatically when the drive unit commissions the ESR reactions.

References

For a detailed description of drive parameters, refer to:

SINAMICS S120/S150 List Manual

11.3.7 Feedback of the ESR status

The drive signals back the actual ESR status to the control using the message word (MELDW) of the cyclic PROFIdrive telegram. In the PLC, the status can be read via system variables and NC/PLC interface signals.

System variable

The system variables can be evaluated in the part program / synchronized action, for example, in order to trigger the drive-autonomous ESR (see Section "Triggering ESR via system variable (Page 625)") or to acknowledge the ESR reactions triggered in the drive (see Section "Acknowledge ESR reactions (Page 626)").

Interrelationship between signals

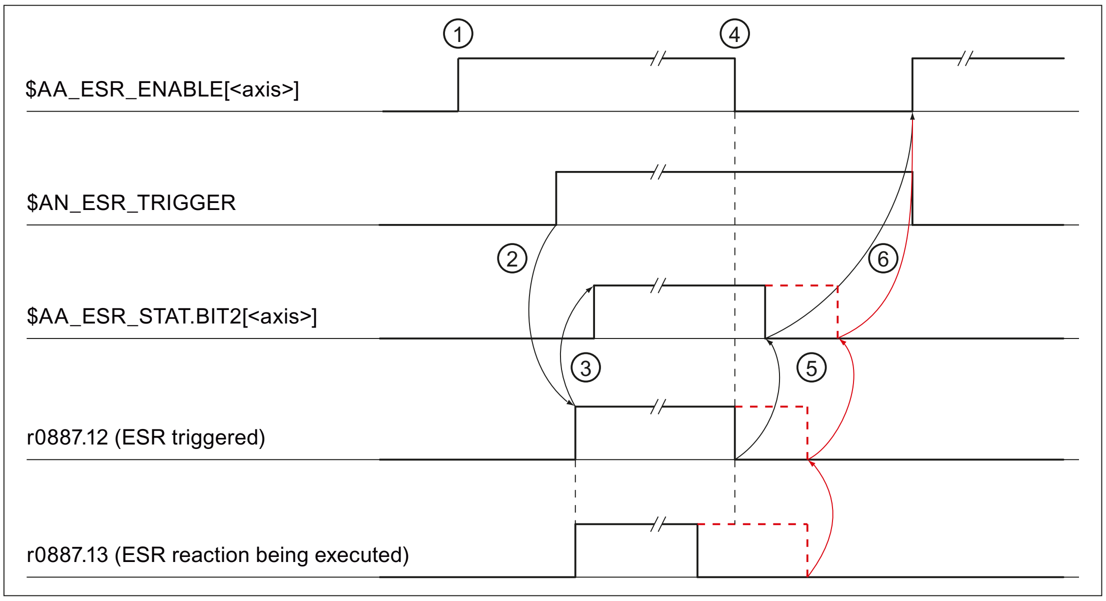
MELDW	System variable	NC/PLC interface signal	Meaning
MELDW.Bit 9 (r0887.12)	$AA_ESR_STAT.Bit 2	DB31, … DBX95.2	ESR: Reaction triggered or generator operation active (r0887.12)
MELDW.Bit 4 (r0056.15)	$AA_ESR_STAT.Bit 3	DB31, … DBX95.1	ESR: DC link undervoltage (p1248)
MELDW.Bit 2 (r2199.0)	$AA_ESR_STAT.Bit 4	DB31, … DBX95.3	ESR, generator operation: Speed lower than minimum (p2161)

11.3.8 Acknowledge ESR reactions

Acknowledging axis-specific drive-independent ESR reactions must be programmed on a user-specific basis in a part program/synchronized action. After the ESR reaction in the drive has been completed, an edge change must be generated in the system variables $AA_ESR_ENABLE: 1→0→1. In order that the drive-integrated ESR reactions can be retriggered, the system variable $AN_ESR_TRIGGER must also be reset to a value of 0.

System variable and drive parameters

The following diagram shows the relationship between system variables and drive parameters when triggering and acknowledging ESR reactions.



① NC: Enabling the ESR reaction via $AA_ESR_ENABLE = 1 (axis-specific)

② NC: Triggering ESR responses via $AN_ESR_TRIGGER = 1

③ Drive: The triggering of the ESR reaction has been detected (CU_STW.2 == 1) ⇒
r0887.12 = 1 AND r0887.13 = 1
Feedback signal to the NC via $AA_ESR_STAT.BIT2 = 1 (axis-specific)

④ NC: Acknowledging the ESR reaction via $AA_ESR_ENABLE = 0 (axis-specific)

Black: At instant in time ④, the ESR reaction no longer runs (r0887.13 == 0) ⇒

⑤ Drive: Reset "ESR triggered": r0887.12 = 0
Feedback signal to the NC via $AA_ESR_STAT.BIT2 = 0 (axis-specific)

⑥ NC: The ESR reaction can be re-enabled ⇒
$AA_ESR_ENABLE = 1 AND $AN_ESR_TRIGGER = 0

Red: At instant in time ④, the ESR reaction still runs (r0887.13 == 1) ⇒

⑤ Drive: "ESR initiated" is only reset (r0887.12 = 0), if the ESR response in the drive has been completed: r0887.13 == 0
Feedback signal to the NC via $AA_ESR_STAT.BIT2 = 0 (axis-specific)

⑥ See ⑥ above

Figure 11-6 Sequence: Trigger and acknowledge drive-integrated ESR reactions

11.3.9 Configuring ESR in the part program

Configuring drive-autonomous ESR functions "stop" and "retract" can be changed using the commands from the part program described in the following.

11.3.9.1 Stopping (ESRS)

Syntax

```
ESRS(<access_1>,<stopping time_1>[,...,<axis_n>,<stopping time_n>])
```

Meaning

`ESRS`:	Using the function, drive parameters can be changed regarding the drive-autonomous "stop" ESR function. Special situations: • Must be alone in the block • Triggers a preprocessing stop • Cannot be used in synchronized actions • A maximum of five axes can be programmed per function call; n = 5	
`<axis_n>`:	For the specified axis, writes to drive parameter p0888 (configuration): p0888 = 1	
	Data type:	AXIS
	Range of values:	Channel axis name
`<stopping time_n>`:	For the axis specified under <Axis_n>, writes to drive parameter p0892 (timer): P0892 = <stopping time_n>	
	Unit:	s
	Data type:	REAL
	Range of values:	0.00 - 20.00

11.3.9.2 Retraction (ESRR)

Syntax

```
ESRR(<axis_1>,<retraction distance_1>,<retraction
velocity_1>[,...,<axis_n>,<retraction distance_n>,<retraction
velocity_n>])
```

Meaning

`ESRR`:	Using the function, drive parameters can be changed regarding the drive-autonomous "retract" ESR function. Special situations: • Must be alone in the block • Triggers a preprocessing stop • Cannot be used in synchronized actions • A maximum of five axes can be programmed per function call; n = 5	
`<axis_n>`:	For the specified axis, writes to drive parameter p0888 (configuration): p0888 = 2	
	Data type:	AXIS
	Range of values:	Channel axis name
`<retraction velocity_n>`:	For the axis specified under <Axis_n>, writes to drive parameter p0893 (retraction speed): p0893 = (<retraction velocity> based on the currently effective total transmission ratio converted into the corresponding retraction speed)	
	Unit:	mm/min, inch/min, degrees/min (depending on the unit of the axis)
	Data type:	REAL
	Range of values:	0.00 - MAX
`<retraction distance_n>`:	For the axis specified under <Axis_n>, writes to drive parameter p0892 (retraction time): p0892 = <retraction distance_n> / <retraction velocity _n>	
	Unit:	mm, inches, degrees (depending on the unit of the axis)
	Data type:	REAL
	Range of values:	MIN - MAX

Note

Retraction speed

It should be ensured that the drive can reach the programmed retraction speed (see parameters <retraction velocity_n>) within the retraction time (see parameter <retraction distance _n>). The settings in the following drive parameters should be checked:

- p1082[0...n] (maximum speed)
- p1135[0...n] (OFF3 ramp-down time)

Retraction path

The conversion of the retraction path into a retraction duration for drive parameter p0892 is performed under the assumption that the retraction motion is triggered at axis standstill (see "Figure 11-5 Configuring_drive_integrated_generator_operation (Page 624)").

If retraction motion is triggered while the axis is traversing, the actually traversed retraction distance differs from the programmed retraction distance due to the fact that traversing motion is superimposed on the retraction motion.

Dependencies

The programmed values for the retraction path and the retraction velocity refer to the load side. Before writing to the drive parameters these are converted over to the motor side. The transmission ratio effective in the NC at the execution time is applicable for the conversion. If the drive-autonomous ESR for an axis is enabled ($AA_ESR_ENABLE[<axis>] == 1), then it is not permissible to change the axial transmission ratio.

Note

Changing the transmission ratio

If the drive-autonomous ESR is not enabled for an axis ($AA_ESR_ENABLE[<axis>] == 0), then the axial transmission ratio can be changed. After the change, it is the sole responsibility of the user/machine manufacturer to appropriately adapt the retraction distance and the retraction velocity.

See also

Configuring generator operation in the drive (Page 623)

11.3.9.3 Boundary conditions

Consistent parameter change

In order that the drive parameters are consistently effective, in the part program the drive-autonomous ESR enable signals must be withdrawn before the functions `ESRS(...)` and `ESRR(...)`.

Example:

Program code	Comment
N100 $AA_ESR_ENABLE[X1]=0	; Withdraw ESR enable signals
N110 $AA_ESR_ENABLE[Z1]=0	
N120 $AA_ESR_ENABLE[Y1]=0	
N130 ESRR(X1,20.0,10000.0,Z1,-15.0,20000.0)	; Change ESR parameters
N140 ESRS(Y1,0.5,B1,0.5)	
N150 $AA_ESR_ENABLE[X1]=1	; Set ESR enable signals
N160 $AA_ESR_ENABLE[Z1]=1	
N170 $AA_ESR_ENABLE[Y1]=1	

When executing the functions `ESRS(...)` and `ESRR(...)` further execution of the part program is stopped until writing to the drive parameters has been acknowledged. During this time, the following message is used as feedback signal for the user: "Wait for drive parameters from <Function>" is displayed.

Block search with calculation

During the block search with calculation, the functions `ESRS(...)` and `ESRR( ... )` are collected and executed in the action block.

Block search with calculation in "Program test" (SERUPRO) mode

During SERUPRO, the functions `ESRS(...)` and `ESRR( ... )` are executed immediately.

Reset behavior

The parameter values written using the functions `ESRS(...)` and `ESRR(...)` are overwritten with the parameter values saved in the drive when the drive runs up or for a drive warm restart.

Drive parameters can be saved from the HMI user interface or in the drive. From the HMI user interface:

- Operating area "Commissioning" > "Machine data" > "Drive MD" > "Save/Reset" > "Save" > "Drive object"**or "**drive unit" **or "** drive system"

Control power-up (POWER ON)

If the drive-autonomous ESR should already be active after the control has powered up (POWER ON), then the programming and enable must be realized within one of the ASUBs started from the PLC – or using the PROGEVENT mechanism.

References

For detailed information about event-controlled program calls, refer to:

Basic Functions Function Manual, Mode Group, Channel, Program Operation, Reset Response (K1);
Chapter: Event-driven program calls

11.3.10 ESR and Safety Integrated (840D sl)

Delay times

In order that drive-integrated ESR reactions in conjunction with Safety Integrated are executed before the safety reactions associated with STOP E, STOP F or communication failure, then the corresponding delay times regarding safety reactions must be configured in the control and in the drive.

Delay times are configured in the controller using the following machine data:

- MD36954 $MA_SAFE_STOP_SWITCH_TIME_E (transition time, STOP E to safe standstill)
- MD36955 $MA_SAFE_STOP_SWITCH_TIME_F (transition time, STOP F to STOP B)
- MD36961 $MA_SAFE_VELO_STOP_MODE (stop response, safely-reduced speed)
- MD36963 $MA_SAFE_VELO_STOP_REACTION (stop response, safely-reduced speed)
- MD10089 $MN_SAFE_PULSE_DIS_TIME_BUSFAIL (wait time pulse cancellation when the bus fails)

Delay times in the drive are configured using the following drive parameters:

- p9554 (transition time from STOP E to "Safe Operating Stop" (SOS))
- p9555 (transition time STOP F to STOP B)

- p9561 (SG stop response)
- p9563 (SG-specific stop response)
- p9580 (wait time after which the pulses are safely cancelled after bus failure)
- p9697 (delay time for the pulse suppression after bus failure)
- p9897 (delay time for the pulse suppression after bus failure)

Note

The drive parameters are automatically aligned with the corresponding NC machine data using the Safety Integrated commissioning function "Copy SI data".

Trigger drive-integrated ESR reactions

The feedback signal of the safety stop reaction – currently active in the drive – sent to the control is realized using the following axis-specific system variable:

- $VA_STOPSI[<axis>] (actual stop reaction)

By evaluating the system variable in a part program/synchronized action, a user-specific response can be realized to the particular stop reaction of the axis, e.g. by triggering the drive-integrated and/or control-managed ESR reactions using the system variable $AN_ESR_TRIGGER and/or $AC_ESR_TRIGGER or $AA_ESR_TRIGGER.

References

A detailed description for configuring and the interaction of drive-integrated ESR responses and Safety Integrated functions is given in:

References

Function Manual SINUMERIK Safety Integrated

11.3.11 ESR and Safety Integrated (828D)

No feedback signal of the safety stop reaction

Within the scope of SINUMERIK 828D, the safety stop reaction currently active in the drive is **not** signaled back to the control. Therefore, in conjunction with Safety Integrated, only drive-autonomous ESR functions can be used.

The ESR reactions triggered from the drive can be determined on the control side using the axis-specific system variable $AA_ESR_STAT (see Chapter "Feedback of the ESR status (Page 625)") and additional ESR reactions can be triggered on a user-specific basis via $AN_ESR_TRIGGER.

ESR reactions that are not influenced by safety reactions

In order that drive-autonomous ESR reactions in conjunction with Safety Integrated are not influenced by safety reactions as a result of STOP E, STOP F executed in parallel or communication failure, then the appropriate delay times must be configured in the drive regarding safety reactions **and** the corresponding safety trigger sources to trigger the drive-autonomous ESR reactions.

Delay times

Delay times in the drive are configured using the following drive parameters:

- p9554 (transition time from STOP E to "Safe Operating Stop" (SOS))
- p9555 (transition time STOP F to STOP B)
- p9580 (wait time after which the pulses are safely cancelled after bus failure)
- p9697 (delay time for the pulse suppression after bus failure)

Safety trigger sources

Using a BICO interconnection of drive parameter p0890 (BI: ESR trigger) ESR reactions can also be triggered with the safety reactions STOP E, STOP F and communication failure:

Parameter	BICO interconnection with:	Effect: Trigger ESR reactions for
p0890[1]	r9721[15]	Safety STOP E
p0890[2]	r9723[1]	Safety STOP F
p0890[3]	r9723[2]	Safety communication failure

References

A detailed description of the drive parameters and the BICO technology can be found in:

- SINAMICS S120/S150 List Manual
- SINAMICS S120 Function Manual; Chapter "Fundamentals of drive technology" > "BICO technology: Interconnecting signals"

11.4 Boundary conditions

Transformation

If NC-controlled extended stopping and retraction is activated for **an**axis involved in an active transformation, then extended stopping and retraction must also be activated for **all other** axes involved in the transformation. Otherwise, when activating extended stopping and retraction, it is possible that the transformation group will ungrouped.

Drive components

In order that the configured stopping or retraction responses can be executed, the drive components involved must be fully functional. Axis-specific servo or drive alarms, which indicate the failure of one of these components, are also implicitly signaling that the configured stopping or retraction response of the axis involved is no longer available or only partially.

Motion-synchronous actions

Motion-synchronous actions are executed in the interpolator clock cycle. Increasing the interpolator clock cycle, e.g. due to a high number of active motion-synchronous actions, results in a coarser time grid for evaluating trigger conditions and triggering responses for extended stopping and retraction.

11.5 Examples

11.5.1 NC-controlled reactions

Task

The A axis is to operate as the generator drive, while the X axis should retract 10 mm at maximum velocity in the event of a fault. The Y and Z axes should stop after a delay of 100 ms so that the X retraction axis has sufficient time to cancel the mechanical coupling.

The example is performed using NC-controlled reactions.

Only the essential parts of the entire functionality are displayed.

Requirements

The following functions must be available:

- Extended stopping and retraction (ESR)
- Static synchronized actions
- Asynchronous subprograms (ASUBs)

Settings

Machine data

Axis-specific operating mode of the ESR

- MD37500 $MA_ESR_REACTION[X] = 21 (NC-controlled retraction axis)
- MD37500 $MA_ESR_REACTION[Y] = 22 (NC-controlled stop axis)
- MD37500 $MA_ESR_REACTION[Z] = 22 (NC-controlled stop axis)

Configuration: NC-controlled extended stop

- MD21380 $MC_ESR_DELAY_TIME1 = 0.1 (duration of the path interpolation in seconds)
- MD21381 $MC_ESR_DELAY_TIME2 = 0.04 (braking duration in seconds)

Programming

Program code	Comment
`LFPOS`	`Axis-specific lift to a position`
`POLF[X]=IC(10)`	`Retraction position, incremental`
`POLFMASK(X)`	`Enable retraction`
`; Enable "Extended stop and retract"`	
`$AA_ESR_ENABLE[X]=1`	`; X axis`
`$AA_ESR_ENABLE[Y]=1`	`; Y axis`
`$AA_ESR_ENABLE[Z]=1`	`; Z axis`

Trigger conditions and static synchronized actions

Example 1

Trigger condition is the occurrence of alarms, which activate the follow-up (tracking) mode:

Program code
```
IDS=02 WHENEVER ($AC_ALARM_STAT B_AND 'H2000') > 0 DO $AC_ESR_TRIGGER=1
```

Example 2

Trigger condition is when the ELG synchronous monitoring responds, if, e.g. Y is defined as ELG following axis and the max. permissible synchronism difference is 100 mm:

Program code
```
IDS=03 WHENEVER $VA_EG_SYNCDIFF[Y] > 0.1 DO $AC_ESR_TRIGGER=1
```

11.5.2 Retraction while thread cutting

During thread cutting, for a fault/interruption, axis X should be retracted to the position specified under `POLF` as response. Axis Z continues to traverse until it is stopped as normal.

Program code	Comment
`N10 G0 G90 X200 Z0 S200 M3`	`;`
`N20 G0 G90 X170`	`;`
`N30 POLF[X]=210`	`; Retraction position axis X, absolute`
`N40 LFPOS`	`; Retraction via POLF/POLFMASK ON`
`N50 POLFMASK(X)`	`; Enable retraction, axis X`
`N60 LFON`	`; Fast retraction for thread cutting ON`
	`; Thread cutting`
`N70 G33 X100 I10`	`; Retraction response, axial: X (abs.)`
`N80 X130 Z-45 K10`	`; dto.`

Program code	Comment
N90 X155 Z-128 K10	; dto.
N100 X145 Z-168 K10	; dto.
N110 X120 I10	; dto.
N120 G0 Z0 LFOF	; Fast retraction for thread cutting OFF
N130 POLFMASK()	; Disable retraction of all axes

11.5.3 Rapid lift using ASUB and fast input

Activating ASUB using the LIFTFAST program command (rapid lift) via fast input 1:

Program code	Comment
N10 SETINT (1) PRIO=1 ABHEB_Y LIFTFAST	; ASUB activation, fast input 1
N20 LFPOS	; Retraction via POLF/POLFMASK ON
N30 POLF[X]=19.5 POLF[Y]=33.3	; Retraction positions axis X Y, absolute
N40 POLF[Z]=100	; Retraction position axis Z, absolute
N50 X0 Y0 G0	;
N60 POLFMASK(X, Y)	; Enable retraction, axis X Y
N70 Z100 G1 F1000	; Retraction response, axial: X, Y (abs.)
N80 POLFMASK(Z)	; Disable retraction, axis X Y and
	; Enable retraction, axis Z
N90 Y10	; Retraction response, axial: Z (abs.)
N100 POLFMASK()	; Disable retraction of all axes

11.5.4 Rapid lift, absolute and incremental

Retraction to absolute positions and through an incremental distance:

Program code	Comment
N10 $AA_ESR_ENABLE[X]=1	; Enable retraction, axis X
N20 $AA_ESR_ENABLE[Z]=1	; Enable retraction, axis Z
N30 $AA_ESR_ENABLE[Y]=1	; Enable retraction, axis Y
N40 LFPOS	; Retraction via POLF/POLFMASK ON
N50 POLF[X]=IC(3.0) POLF[Y]=-4.0	; Retraction position axis X, incremental
	; Retraction position axis Y, absolute
N60 POLF[Z]=100	; Retraction position axis Z, absolute
N70 G0 X0 Y0 Z0	;
N80 POLFMASK(X, Y)	; Enable retraction, axis X Y
N90 Z100 G1 F1000	; Retraction response, axial: X (inc.), Y (abs.)
N100 POLF[X]=10	; Retraction position axis X, absolute
N110 Y0 G1 F1000	; Retraction response, axial: X, Y (abs.)
N120 POLFMASK(Z)	; Disable retraction, axis X Y and
	; Enable retraction, axis Z
N130 Y10	; Retraction response, axial: Z (abs.)

Program code	Comment
N140 POLFMASK()	; Disable retraction of all axes

11.5.5 Lift fast with linear relation of axes

Retracting in a linear relationship, absolute and incremental

Program code	Comment
N10 $AA_ESR_ENABLE[X]=1	; Enable retraction, axis X
N20 $AA_ESR_ENABLE[Y]=1	; Enable retraction, axis Y
N30 $AA_ESR_ENABLE[Z]=1	; Enable retraction, axis Z
N40 LFPOS	; Retraction via POLF/POLFMASK ON
N50 POLF[X]=IC(3.0) POLF[Z]=-4.0	; Retraction position axis X, incremental
	; Retraction position axis Z, absolute
N60 POLF[Y]=100	; Retraction position axis Z, absolute
N70 X0 Y0 Z0 G0	; ---
N80 POLFMLIN(X, Y)	; Enable retraction, axis X Y, linear
	; Relationship
N85 POLFMASK(Z)	; Enable retraction, axis Z
N90 Z100 G1 F1000	; Retraction response:
	; - linear relation: X (inc.), Y (abs.)
	; - axial: Z (abs.)
N95 POLF[X]=10	; Retraction position axis X, absolute
N100 Y0 G1 F1000	; Retraction response:
	; - linear relation: X (inc.), Y (abs.)
	; - axial: Z (abs.)
N110 POLFMLIN()	; Disable retraction, axis X Y
N120 Y10	; Retraction response, axial: Z (abs.)
N130 POLFMASK()	; Disable retraction of all axes

11.6 Data lists

11.6.1 Machine data

11.6.1.1 Channelspecific machine data

Number	Identifier: $MC_	Description
21204	LIFTFAST_STOP_COND	Stop characteristics for rapid lift
21380	ESR_DELAY_TIME1	Delay time (STOPBYALARM, NOREAD) for ESR axes
21381	ESR_DELAY_TIME2	Time for interpolatory braking of ESR axes

11.6.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
37500	ESR_REACTION	Reaction definition with extended stop and retract

11.6.2 System variables

Identifier	Meaning
$A_DBB	Read/write data byte from/to PLC
$A_IN	Digital input
$A_OUT	Digital output
$AA_ESR_ENABLE[<axis>]	Enable "Extended stop and retract"
$AA_TYP[<axis>]	Axis type
$AA_ESR_TRIGGER[<axis>]	Initiate ESR for PLC controlled axis (single axis)
$AC_ALARM_STAT	Alarm status in the channel
$AC_ESR_TRIGGER	Trigger ESR, channel-specific
$AC_STAT	Channel status

11.6.3 Signals

11.6.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate disable	DB21, ... DBX6.0	

11.6.3.2 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feed stop	DB31, ... DBX4.3	

11.6.3.3 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
ESR: DC link undervoltage (p1248)	DB31, ... DBX95.1	DB390x.DBX4003.1
ESR: Reaction triggered / generator operation active (r0887.12)	DB31, ... DBX95.2	DB390x.DBX4003.2
ESR, generator operation: Speed lower than minimum (p2161)	DB31, ... DBX95.3	DB390x.DBX4003.3

S9: Setpoint exchange - 840D sl only

12

12.1 Brief description

Function

The "Setpoint switchover" function is required if only one motor is to be used to drive several axes/spindles. For example, for milling heads, where the spindle motor is to be used to drive the tool as well as to orientate the milling head.

Requirements

The following conditions must be fulfilled so that the setpoint can be switched over:

- All of the axes involved in the setpoint switchover are at a standstill
- No sign-of-life error and no errors are active on PROFIBUS
- None of the following functions in the axis with drive control are active:
  - Reference point approach
  - Measuring
  - Travel to fixed stop
  - Function generator
  - Star/delta switchover
  - Parameter set switchover

Position control loop

The drive train, and therefore also the position control loop, are disconnected while the setpoint is being switched over. In order to avoid instability, the setpoint is only switched over at standstill - and once all controller enable signals have been cleared (deleted).

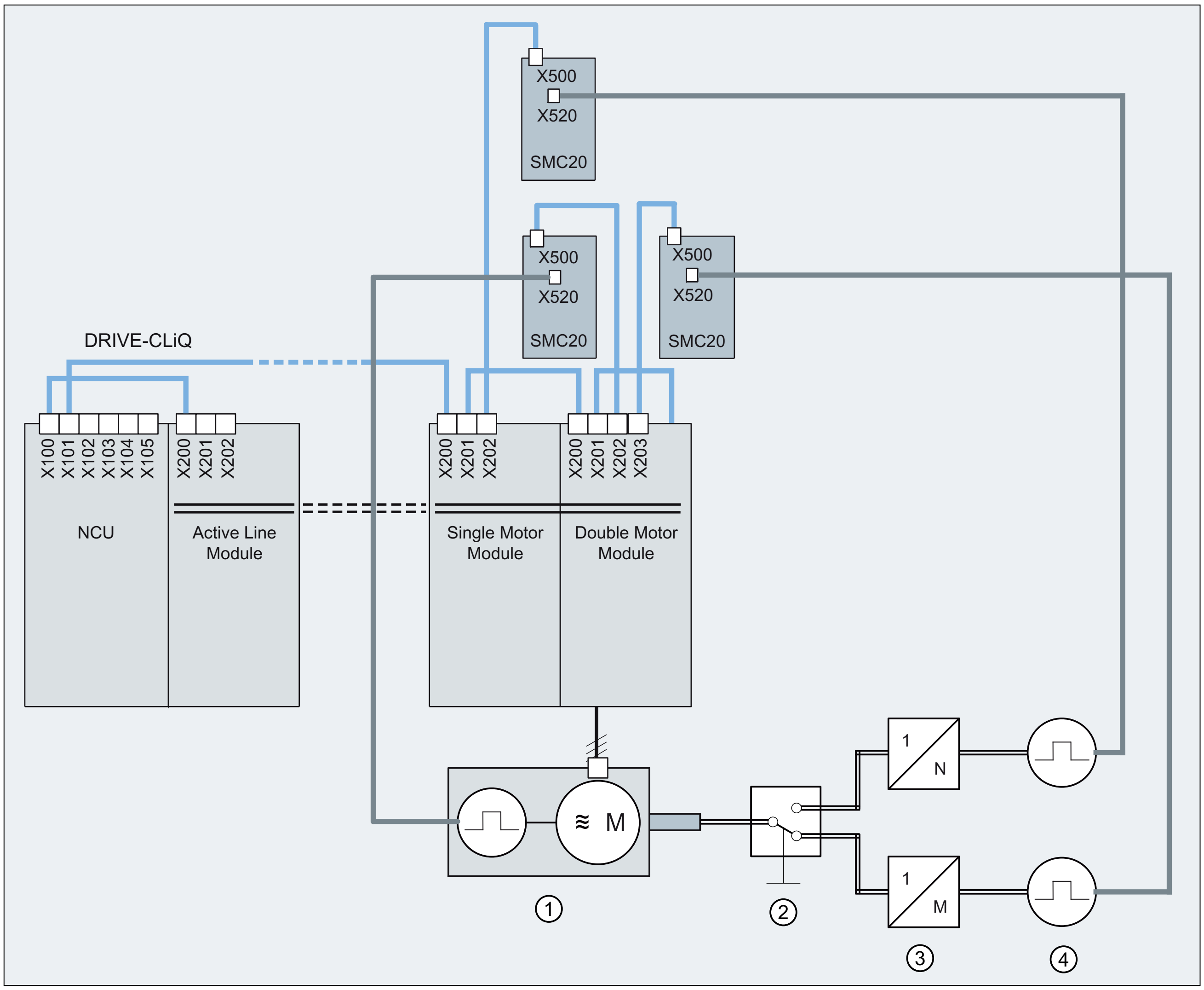
When using a single drive, only one control loop can be closed at the same instant in time. Axes without drive control are operated with open position controller and tracked positions.

Note

Vertical axes

The controller enable signals for all of the axes involved are automatically withdrawn by the control for the duration of the switchover. Axes without drive control are not in closed-loop control. Therefore, a brake control must be implemented for vertical axes.

Example



① Motor with encoder
② Mechanical switchover device
③ Gearbox 1
④ Encoder 1 (e.g. for the milling head)
⑤ Gearbox 2
⑥ Encoder 2 (e.g. for the spindle)

Figure 12-1 Setpoint switchover with 2 axes

Replacing the technology function "setpoint changeover" (TE5)

The "setpoint changeover" function replaces the technology function "setpoint changeover" (TE5). Migration of the technology function to the functionality of the "setpoint changeover" function means that you must make changes in the machine data and in the PLC user program:

- Machine data MD63750 is no longer required.
- The meanings of various NC/PLC interface signals have changed.
- Alarms 70451 and 70452 are no longer applicable.
- A setpoint changeover with simulated axes (MD30130 = 0) is no longer supported.
- The known restrictions relating to the technology card function are no longer applicable.

12.2 Startup

Machine data

Setpoint assignment

The same setpoint channel of a drive is assigned a number of times to define the axes participating in the setpoint switchover. For this, the following machine data must be pre-assigned with the same logical drive number for each axis:

MD30110 $MA_CTRLOUT_MODULE_NR (setpoint assignment: module number)

Note

Alarm 26018 is output if the option is missing.

Encoder assignment

The encoder is assigned in the machine data on an axis-for-axis basis:

MD30230 $MA_ENC_INPUT_NR(actual value assignment: Input on drive module/measuring circuit module)

NC/PLC interface signals

Note

Access rights

Despite assignment of a drive to several machine axes, the use of NC/PLC interface signals remains unchanged. In the PLC user program this requires an explicit coordination of the access operations to the NC/PLC interface signal of the individual machine axes involved in the setpoint switchover.

Request for drive control

The request to assume drive control is realized via:

DB31, ... DBX24.5 (setpoint switchover: request drive control)

Status of the drive control

The current status of drive control is displayed via:

DB31, ... DBX96.5 (setpoint switchover: drive control active)

Status signals

After drive control is assumed for the first time for one of the machine axes involved in the setpoint switchover (DB31, ... DBX24.5 = 1 (setpoint switchover: switchover)) the following status signals are always displayed the same for all of the machine axes involved:

DB31, ... DBB92 - DBB95

Control signals

The control signals are only active in the machine axis that currently has the drive control (DB31, ... DBX96.5 == 1 (setpoint switchover active)):

DB31, ... DBB20 to DBB21

Controller enable

The controller enable (DB31, ... DBX2.1) is only active in the machine axis that currently has drive control (DB31, ... DBX96.5 == 1 (status setpoint switchover))

Follow-up mode

Machine axes **without** drive control, involved in the setpoint switchover, are always switched into the follow-up mode in the control itself (DB31, ... DBX61.3 == 0).

Note

Only one axis with drive control

At any one point in time, only one axis must have drive control DB31, ... DBX24.5 == 1 and DB31, ... DBX96.5 == 1.

Special cases

Several simultaneous requests to take over

No switchover takes place for several simultaneous takeover requests. The last axis used keeps drive control. The same effect applies if there are no takeover requests.

No takeover request when powering up

If there is no takeover request when the control powers up, then the **control** assigns the drive control to the first machine axis with the same logical drive number.

```
MD30110 $MA_CTRLOUT_MODULE_NR[0,AX1] = 1

MD30110 $MA_CTRLOUT_MODULE_NR[0,AX2] = 2

MD30110 $MA_CTRLOUT_MODULE_NR[0,AX3] = 3
```

```
MD30110 $MA_CTRLOUT_MODULE_NR[0,AX4] = 4 ; setpoint switchover: 1st axis
MD30110 $MA_CTRLOUT_MODULE_NR[0,AX5] = 4 ; setpoint switchover: 2nd axis
```

In order to be able to traverse the axis using the drive, the NC/PLC interface signal to request drive control (DB31, ... DBX24.5) must be set.

12.3 Flow diagram

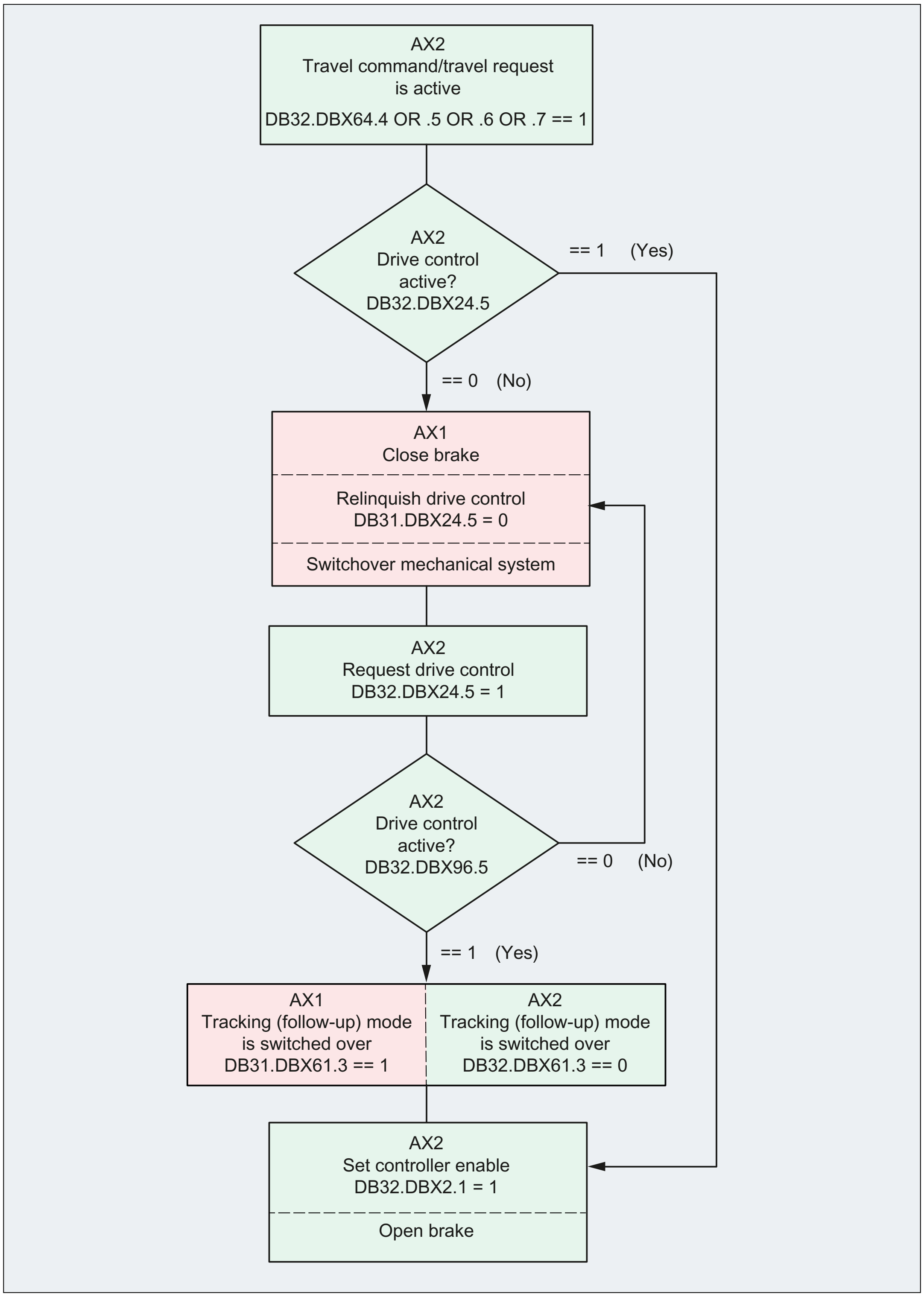


Figure 12-2 Sequence when switching over the setpoint from machine axes AX1 to AX2

12.4 Supplementary conditions

Alarms

Only axes with active drive control display drive alarms.

Position control loop

The drive train, and therefore also the position control loop, are disconnected while the setpoint is being switched over. In order to avoid instability, the setpoint is only switched over at standstill - and once all controller enable signals have been cleared (deleted).

When using a single drive, only one control loop can be closed at the same instant in time. Axes without drive control are operated with open position controller and tracked positions.

Reference points

The use of load-side encoders does not affect the axis-specific reference points of a setpoint switchover.

However, the mechanical reference to the load can be lost following setpoint switchover for a load-side position derived solely from the motor encoder. These types of axis must be referenced again after every setpoint switchover.

Parameter set switchover

A setpoint exchange is not performed if one of the following states is present:

- A parameter set changeover in one of the two machine axes has not completed.
- The parameter sets of the two machine axes are not the same.

Note

If, due to secondary conditions, a setpoint switchover is not performed, then a message is **not** displayed.

"Parking" state

If an axis is in the "Parking" state, this state can only be canceled if this axis assumes drive control.

Service display drive

The "Drive service display" HMI diagnostics screen does **not** take into account the changes in assignments between machine axes and the drive.

Commissioning via SinuCom NC

The "SinuCom NC" commissioning tool can only be used to commission the setpoint switchover via the expert list.

Safety Integrated (only 840D sl)

A detailed description of the supplementary conditions for setpoint switchover in conjunction with Safety Integrated is available in:

Reference:
SINUMERIK Safety Integrated Manual

12.5 Data lists

12.5.1 Machine data

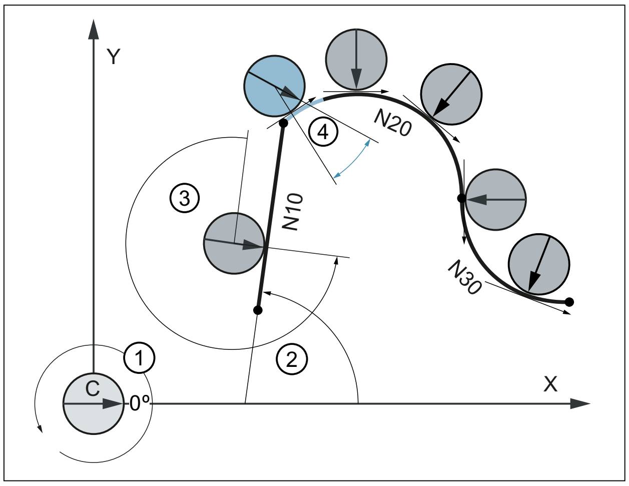
12.5.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
30130	CTRLOUT_TYPE	Output type of setpoint
30200	NUM_ENCS	Number of encoders
30220	ENC_MODULE_NR	Actual-value assignment: Drive number / measurement circuit number
30230	ENC_INPUT_NR	Actual-value assignment: Input on drive module/measuring circuit module

T3: Tangential control - 840D sl only

13

Using the "Tangential control" function, a rotary axis is coupled as following axis to two geometry axes as leading axes, so that the alignment of the following axis is a function of the path tangent of the leading axes.



① Initial position and positive direction of rotation of the following/rotary axes C

② Angle of the path tangent in machining plane X/Y

③ Offset angle = 270° or -90°

④ Rounding clearance and present angular deviation

The path-defining leading axes, geometry axes X and Y and rotary axis C the following axis of the tangential coupling can be seen in the diagram. The rotary axis is tracked with a programmable offset angle of 270° with respect to the path tangent. At the block transition from N10 to N20, the contour has a discontinuous transition or corner. As a result of a dynamic response that is too low, the following axis cannot follow the path tangent over part of the path (blue). However, the dynamic response of the following axis is sufficient so that it can precisely follow circular blocks N20 and N30 further along the contour.

Properties

The tangential coupling sets itself apart as a result of the following features:

- The leading axes of the tangential coupling must be two geometry axes.
- The following axis of the tangential coupling must be a rotary axis.
- The tangential coupling can be defined, activated, deactivated and deleted in the NC program.
- Tangential control is defined for the basic and the workpiece coordinate system.
- The positive direction of the path tangent is the same as the traversing direction of the path.
- For an offset angle of 0° between the rotary axis and the path tangent, shows the 0° direction.
- The axes of the tangential coupling must be the channel axes of the same channel.

- The position of the following axis can act as the input value for an additional transformation.
- The tangential coupling is only active in the AUTOMATIC and MDI modes.

Application examples

- Tangential positioning a rotatable tool during nibbling
- Tracking the workpiece alignment for a belt saw.
- Positioning a dressing tool on a grinding wheel
- Positioning a cutting wheel for cutting glass or paper
- Tangentially feeding a wire for 5-axis welding.

Corner behavior:

If the contour described by the leading axes has a discontinuous block transition or corner, then one of the following corner behaviors can be selected.

- The dynamic response of the rotary axis has no effect on the leading axes
- The dynamic response of the rotary axis is considered in the path planning of the leading axes together with programmable parameters "rounding clearance" and "angular tolerance".
- The leading axes are stopped before the corner, and in an automatically generated intermediate block, the following axis is realigned.
  The corner angle is detected depending on the value of machine data MD37400 $MA_EPS_TLIFT_TANG_STEP (tangential angle for corner detection).

13.1 Commissioning

Machine data

Corner detection TLIFT()

In conjunction with the corner detection function TLIFT() (Page 653), a discontinuity in the programmed path of the leading axes is identified as a corner if the change in the tangential angle for the following axis is greater at this location than the value specified in the machine data:

MD37400 $MA_EPS_TLIFT_TANG_STEP[<following axis>] = <tangent change angle>

Parameterizable offset angle

A parameterizable offset angle is specified for the following axis using the machine data:

Maschinendatum MD37402 $MA_TANG_OFFSET[<following axis>] = <parameterizable offset angle>

The offset angle of the following axis is the angle between the zero position of the following axis and the path tangent of the programmed path of the leading axes when tangential coupling is active.

The active offset angle is the sum of the offset angle parameterized in the machine data and the offset angle programmed with TANGON() (Page 654) when tangential coupling is activated:

Block search settings

Using the machine data, the velocity is selected with which the tangential axis traverses to its position when tangential coupling is activated during or after a block search:

- Rapid traverse velocity (G0)
- Programmed velocity (G1 F....)
- Setting data SD42121 $SC_AX_ADJUST_FEED ≠ 0

MD11450 $MN_SEARCH_RUN_MODE, bit 7

See also: Supplementary conditions (Page 658)

System variables

$AC_BLOCKTYPE block type

System variable $AC_BLOCKTYPE can be used to determine as to whether the actual block is an intermediate block generated by TLIFT(). The actual block is an intermediate block generated by TLIFT() if the following applies:

$AC_BLOCKTYPE == 6

13.2 Programming

13.2.1 Defining coupling (TANG)

Via the predefined procedure TANG(...), a tangential coupling between a rotary axis is defined as the following axis and two geometry axes as the leading axes. The following axis is continuously aligned with the path tangent of the leading axes.

Note

Coupling factor

A coupling factor of 1 does not have to be programmed explicitly.
The direction of the tangential axis is rotated using the coupling factor of -1.

Syntax

```
TANG(<following axis>, <leading axis_1>, <leading axis_2>, <coupling
factor>, <coordinate system>, <optimization>)
```

Meaning

`TANG(...)`:	Define tangential coupling		
`<following axis>`:	Axis name of the following axis (rotary axis)		
	Data type:	AXIS	
	Range of values:	Channel axis names	
`<leading axis_1>` `<leading axis_2>`:	Axis names of the leading axes (geometry axes) 1)		
	Data type:	AXIS	
	Range of values:	Geometry axis names of the channel	
`<coupling factor>`:	Factor n of the angle change of the following axis for changing the path tangent of the leading axes: Angle change$_{\text{following axis}}$ = angle change$_{\text{path tangent}}$ * n		
	Data type:	REAL	
	Default value:	1.0	
`<coordinate system>`:	Active coordinate system 2)		
	Data type:	CHAR	
	Value:	`"B"`:	Basic coordinate system (default value)
		`"W"`:	Workpiece coordinate system (not available)
`<optimization>`:	Optimization type		
	Data type:	CHAR	
	Value:	`"S"`:	Standard (default value) The dynamic response of the rotary axis has no effect on the leading axes If the dynamic response of the rotary axis is greater than required for tracking, this method is sufficiently precise. If the dynamic response of the rotary axis is not great enough to follow the change in the path tangent, the orientation of the rotary axis will deviate from the target orientation along an undefined rounding clearance.
		`"P"`:	The dynamic response of the rotary axis is considered in the path planning of the leading axes. For this purpose, on activation of the tangential coupling with TANGON(), two additional parameters must be specified: • Rounding clearance • Angular tolerance See Section "Activating the coupling (TANGON) (Page 654)" **Note** With kinematic transformations, we recommend using optimization method "P."
Note Default values do not have to be programmed explicitly.			

[1] **Note**
As the leading axes for tangential coupling, the geometry axes must be used that travel along the programmed path in the machine coordinate system (MCS) with reference to the initial position of the machine. For example, if swivel cycle CYCLE800 is used on a milling machine with a swivel head, depending on how the cycle is configured, interpolation will be performed in the WCS, e.g. with the geometry axes X and Y. The tangential coupling, however, must be defined with the geometry axes as the leading axes, which travel along the programmed path in the MCS. For this purpose, the geometry axes in the **non-swiveled** condition of the machine must be used as the leading axes.
[2] **Note**
The basic coordinate system (BCS) must not be rotated with respect to the MCS. For example, if the BCS is rotated with the ROT command or with the swivel cycle CYCLE800, the tangential control is no longer correct.

13.2.2 Activating intermediate block generation (TLIFT)

If the tangent change of the following axis at any position along the programmed path of the leading axes exceeds the limit parameterized in machine data MD37400 $MA_EPS_TLIFT_TANG_STEP, further path planning will depend on the set behavior at corners. Without use of the predefined procedure TLIFT(...), the path is traversed in accordance with the rounding behavior programmed in connection with TANG(...) (Page 651) and TANGON(...) (Page 654).

Activating intermediate block generation

If TLIFT(...) is programmed after TANG(...), an intermediate block automatically generated by the control is inserted at this point when a corner is detected during preprocessing.

When the program is executed, the leading axes are stopped when the intermediate block is reached. In the intermediate block, the following axis is rotated with maximum axis dynamics toward the path tangent of the following block. The leading axes are then traversed further on the programmed path.

Deactivating intermediate block generation

To deactivate intermediate block generation, the tangential coupling must be defined again using TANG(...), but without subsequent activation of intermediate block generation by means of TLIFT(...).

Syntax

```
TLIFT(<following axis>)
```

Meaning

`TLIFT(...)`:	Activate corner detection with intermediate block calculation	
`<following axis>`:	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Range of values:	Channel axis names

Speed of rotation of the following axis

Path axis

If the following axis had already been traversed as a path axis before tangential coupling was activated, the rotational movement is performed in the intermediate block as a path axis.

If you specify the reference radius with `FGREF[<axis>]=0.001`, the rotational movement will be performed with the parameterized maximum axis velocity:

MD32000 $MA_MAX_AX_VELO[<following axis>]

Positioning axis

If the following axis had not yet been traversed as a path axis before tangential coupling was activated, the rotation is performed in the intermediate block as a positioning axis.

The rotational movement is performed with the parameterized positioning axis velocity:

MD32060 $MA_POS_AX_VELO[<following axis>]

13.2.3 Activating the coupling (TANGON)

Via the predefined procedure TANGON(...), a tangential coupling previously defined with TANG(...) (Page 651) is activated. The following axis is then continuously aligned with the path tangent during subsequent travel.

Angle of the following axis

The angle of the following axis with respect to the path tangent depends on the transformation ratio specified in TANG(...), the offset angle parameterized in the machine data MD37402 $MA_TANG_OFFSET, and the offset angle specified for TANGON(...), which is applied additively.

Optimization "P"

If the value "P" was specified as the optimization parameter in the definition of the tangential coupling (TANG(...)), the parameter "rounding clearance" and optionally the parameter "angular tolerance" must be set when coupling is activated.

If the value 0 is specified as the angular tolerance, only the parameter "rounding clearance" will be active.

If a value greater than 0 is specified as the angular tolerance, the active rounding clearance results from the minimum of the parameterized rounding clearance and the rounding clearance based on the parameterized angular tolerance.

If the dynamic response of the following axis is not sufficient to follow the parameterized conditions, the path velocity of the leading axes will be reduced accordingly.

Syntax

```
TANGON(<following axis>, <offset angle>, <rounding clearance>,
<angular tolerance>)
```

Meaning

`TANGON(...)`:	Activate tangential coupling	
`<following axis>`:	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Range of values:	Channel axis names
`<offset angle>`:	Offset angle of following axis with respect to the path tangent The reference point is the zero point of the rotary axis.	
	Data type:	REAL
`<rounding clearance>`:	Maximum permissible rounding clearance If the rounding clearance is increased due to the dynamic conditions, the path velocity of the leading axes is reduced.	
	Data type:	REAL
`<angular tolerance>`:	Maximum permissible tolerance with respect to the specified angle between the following axis zero setting and the path tangent	
	Data type:	REAL

13.2.4 Deactivating the coupling (TANGOF)

Via the predefined procedure TANGOF(...), a tangential coupling defined with TANG(...) (Page 651) and activated with TANGON(...) (Page 654) is deactivated. The following axis is then no longer aligned with the path tangent of the leading axis. However, the coupling of the following axis to the leading axes is retained even after deactivation, which prevents the following functions, for example:

- Plane change
- Geometry axis switchover
- Definition of a new tangential coupling for the following axis

Final cancellation of the connection of the coupling of the following axis to the leading axes is not completed until the coupling has been deleted with TANGDEL(...) (Page 656).

Programming

```
TANGOF(<following axis>)
```

Meaning

`TANGOF(...)`:	Deactivate a tangential coupling	
`<following axis>`:	Axis name of the following axis (rotary axis)	
	Data type:	AXIS
	Range of values:	Channel axis names

13.2.5 Deleting a coupling (TANGDEL)

A tangential coupling defined with TANG(...) (Page 651) will be retained even after deactivation of the tangential coupling with TANGOF(...) (Page 655). The existing tangential coupling then continues to prevent, for example, the following functions:

- Plane change
- Geometry axis switchover
- Definition of a new tangential coupling for the following axis

With the predefined procedure TANGDEL(...), the existing tangential coupling is deleted after the tangential coupling has been deactivated with TANGOF(...).

Syntax

```
TANGDEL(<following axis>)
```

Meaning

`TANGDEL(...)`:	Delete a tangential coupling defined with TANG()	
	Effective:	Non-modal
`<following axis>`:	Axis name of the following axis whose tangential coupling is to be deleted	
	Data type:	AXIS
	Range of values:	Channel axis names

Examples

Leading axis change

Before a new tangential coupling can be defined with another leading axis for the following axis, the existing tangential coupling must first be deleted.

```
Program code                 Comment
N10 TANG(A, X, Y, 1)         ; Define tangential couping for following axis A: A to
                             X and Y
N20 TANGON(A)                ; Activate tangential coupling for following axis A
N30 X10 Y20
...
N80 TANGOF(A)                ; Deactivate tangential coupling for following axis A
N90 TANGDEL(A)               ; Delete tangential coupling for following axis A
...
N120 TANG(A, X, Z)           ; Define new tangential coupling for following axis A
N130 TANGON(A)               ; Activate new tangential coupling for following axis A
...
```

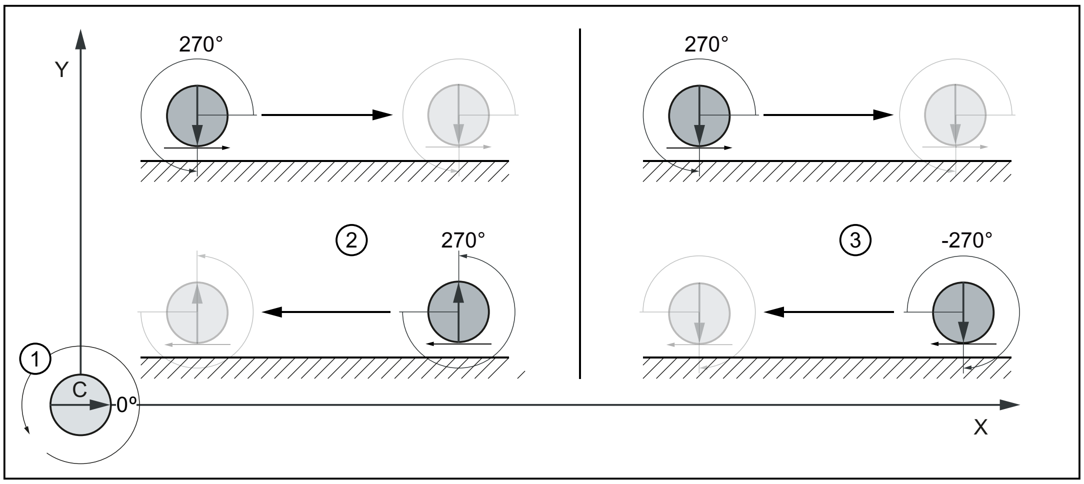
Geometry axis switchover

Before geometry axis switchover can be performed for an existing coupling, the coupling must first be deleted.

Program code	Comment
`N10 GEOAX(2,Y1)`	`; 2nd geometry axis = machine axis Y1`
`N20 TANG(A, X, Y)`	`; Define tangential coupling for following axis A`
`N30 TANGON(A, 90)`	`; Activate tangential coupling for following axis A`
`N40 G2 F8000 X0 Y0 I0 J50`	`; Motion block`
`N50 TANGOF(A)`	`; Deactivate tangential coupling for following axis A`
`N60` **`TANGDEL`**`(A)`	`; Delete tangential coupling for following axis A`
`N70` **`GEOAX`**`(2, Y2)`	`; 2nd geometry axis = machine axis Y2`
`N80 TANG(A, X, Y)`	`; Define new tangential coupling for following axis A`
`N90 TANGON(A, 90)`	`; Activate new tangential coupling for following axis A`
`...`	

13.3 Limit angle

For alternating traversing motion, the direction of the path tangent jumps through 180° at the reversal point. The alignment of the following axis also jumps through 180 °. This behavior is generally not practical (see Fig., ②). Instead, for return motion, the following axis should be traversed with the same alignment as for the forward motion (see Fig., ③).



① Initial position and positive direction of rotation of the following/rotary axis C
② Forward and reverse motion with the same offset angle 270°
③ Forward and reverse motion with alternating offset angles, 270° and -270°

Avoiding reversal jump

Using the working area limitation G25 (lower limit) and G26 (upper limit), for the following axis, a minimum and a maximum value must be entered for the position values referred to the basic coordinate system. After this, the working area limitation for the following axis should be activated/deactivated depending on the machining situation using the WALIMON / WALIMOF commands. At the **path reversal instant** , the working area limitation must be**active**.

Effect

If the following axis traverses with the actual offset angle with the axis position outside the active working area limitation, then the control system checks whether the following axis, when using the negative offset angle, would move in a permissible working area. If this is the case, then the following axis is traversed with the negative offset angle.

Reference

Programming Guide Fundamentals; Chapter "Supplementary commands" > "Working area limitation" > "Working area limitation in the BCS (G25/G26, WALIMON, WALIMOF)"

13.4 Supplementary conditions

Block search with active tangential coupling

If tangential control is active, it is recommended that for a block search only optimization type "P" and block search type 2 ("Block search with calculation at the contour"), programmable under TANG() (Page 651), or 5 ("Block search with calculation in the program test mode (SERUPRO)") are used.

Explicit positioning of the following axis

If the following axis explicitly traverses an active tangential coupling, then the traversed distance is additive to the programmed offset angle.

13.5 Examples

Example 1: Define and activate tangential coupling

Program code	Comment
`; Define tangential coupling:`	
`;  - Following axis C, leading axes X and Y`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "P"`	
`N10 TANG(C,X,Y,1,"B","P")`	`; Simplified: TANG(C,X,Y,,,"P")`
`; Activate tangential coupling:`	
`;  - Offset angle: 90°`	

Program code	Comment
`N20 TANGON(C,90)`	
`...`	

Example 2: Plane change

Program code	Comment
`; Define tangential coupling:`	
`;  - Following axis A, leading axes X and Y`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "S"`	
`N10 TANG(A,X,Y)`	
`; Activate tangential coupling:`	
`;  - Offset angle: 0°`	
`N20 TANGON(A)`	
`N30 X10 Y20`	`; Traverse leading axes`
`...`	
`N80 TANGOF(A)`	`; Deactivate tangential coupling`
`N90 TANGDEL(A)`	`; Delete tangential coupling`
`...`	
`; Define NEW tangential coupling:`	
`;  - Following axis A, leading axes X and Z`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "S"`	
`TANG(A,X,Z)`	
`TANGON(A)`	
`...`	

Example 3: Geometry axis switchover

Program code	Comment
`N10 GEOAX(2,Y1)`	`; 2nd geometry axis:` **`Y1`**
`; Define tangential coupling:`	
`;  - Following axis A, leading axes X and Y`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "S"`	
`N20 TANG(A,X,Y)`	
`; Activate tangential coupling:`	
`;  - Offset angle: 90°`	
`N30 TANGON(A,90)`	
`N40 G2 F8000 X0 Y0 I0 J50`	`; Traverse circle`

Program code	**Comment**
`N50 TANGOF(A)`	`; Deactivate tangential coupling`
`N60 TANGDEL(A)`	`; Delete tangential coupling`
`N70 GEOAX(2,Y2)`	`; 2nd geometry axis:` **`Y2`**
`; Define NEW tangential coupling:`	
`;  - Following axis A, leading axes X and Y`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "S"`	
`N80 TANG(A,X,Y)`	
`; Activate tangential coupling:`	
`;  - Offset angle: 90°`	
`N90 TANGON(A,90)`	
`...`	

Example 4: Optimization type "P"

Program code	**Comment**
`...`	
`N100 G0 C0`	
`N110 G1 X1000 Y500 F50000`	
`N120 TRAORI`	`; Activate transformation`
`N130 G642`	`; Activate smoothing`
`N171 TRANS X50 Y50`	`; Work offset WCS`
`; Define tangential coupling:`	
`;  - Following axis C, leading axes X and Y`	
`;  - Coupling factor: 1.0`	
`;  - Coordinate system: BCS`	
`;  - Optimization type: "P"`	
`N180 TANG(C,X,Y,1,,"P")`	
`; Activate tangential coupling:`	
`;  - Offset angle: 0°`	
`;  - Rounding clearance 5.0 mm`	
`;  - Angular tolerance 2.0°`	
`N190 TANGON(C,0,5.0,2.0)`	
`N210 G1 X1310 Y500`	
`N215 G1 X1420 Y500`	
`N220 G3 X1500 Y580 I=AC(1420) J=AC(580)`	
`N230 G1 X1500 Y760`	
`N240 G3 X1360 Y900 I=AC(1360) J=AC(760)`	
`N250 G1 X1000 Y900`	
`N280 TANGOF(C)`	`; Deactivate tangential coupling`
`N290 TRAFOOF`	`; Deactivate transformation`
`...`	

Example 5: Corner in space

```
Program code                                   ; Comment
; Define tangential coupling:
;  - Following axis A, leading axes X and Y
;  - Coupling factor: 1.0
;  - Coordinate system: BCS
;  - Optimization type: "S" (standard)
TANG(A,X,Y,1.0,"B")
TLIFT(A)                                       ; Activate intermediate block generation
G1 G641 X0 Y0 Z0 A0                            ; Go to initial position
; Activate tangential coupling:
;  - Offset angle: 0°
TANGON(A,0)
N4 X10                                         ; Traverse leading axis X
N5 Z10                                         ; Traverse geometry axis Z
N6 Y10                                         ; Traverse leading axis Y
M30
```

By traversing the geometry axes in blocks N4 and N6, a discontinuous characteristic is obtained, i.e. a corner in the contour of the leading axes of the tangential coupling. By programming TLIFT(), an intermediate block is inserted in front of N6, in which following axis A is rotated through 90° in the tangential direction of the Y axis. Only then is the Y axis traversed in N6.

13.6 Data lists

13.6.1 Machine data

13.6.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11450	SEARCH_RUN_MODE	Block search parameterization

13.6.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
37400	EPS_TLIFT_TANG_STEP	Tangential angle for corner recognition
37402	TANG_OFFSET	Default angle for tangential follow-up control

13.6.2 Setting data

13.6.2.1 Channel-specific setting data

Number	Identifier $SC_	Description
42121	AX_ADJUST_FEED	Path feed in adjustment motion of the tangential axes

13.6.3 System variables

Identifier	Description
$AC_BLOCKTYPE	Current block is an intermediate block generated by `TLIFT`

T4: Automatic post optimization with AST (option) - only 840D sl

14

14.1 Function

Note

The "Automatic post optimization with AST" function is an option for SINUMERIK 840D sl that requires a license.

With the "Automatic post optimization with AST" function, axes with a changed mechanical system can be post optimized from the part program.

SIEMENS provides predefined cycles for the functions of the Automatic Servo Tuning (AST) (post optimization) (see "Programming (Page 666)"). They can be used by machine manufacturers to create their own tuning cycles. The machine operator must only use the machine manufacturer's cycle.

Note

Storage location for optimization cycles

In order that the machine operator can call the optimization cycles generated by the machine manufacturer based on AST cycles, these must be saved in the manufacturer cycle directory (_N_CMA_DIR) of the NC memory.

Application

General problem description: Dependency on the loading

After changing the load - and therefore the moment of inertia, for all directly-driven axes, the current setpoint filters must be adapted to ensure that the speed control loop remains stable.

Such axes are:

- Tables with torque motors
- Linear axes with linear motors that support different masses
- Directly driven main spindles with different clamping fixtures (lathe chuck)

All conventionally driven axes (servomotor with ballscrew spindle and possibly with gearbox) do not require that the speed controller is post optimized (retuned).

Note

If, after the load has changed, the actual parameter data set no longer fits, in order to traverse directly-driven axis reliably without any instability, then parameter values must be adapted **before** setting the controller enable (DB31, ... DBX2.1 = 1). Automatic post optimization with AST is only possible after the controller has been enabled.

The following measures are possible:

- The active parameter data set is set so that it is suitable for all load conditions - and can be used without adaptation ("moderately" adjusted parameter data set).
- A "moderately" adjusted parameter data set is adapted using automatic post optimization with AST, and is then subsequently activated.
- The required parameter values are assigned by the PLC (FB3, PI services, …) and/or manufacturer cycle **before**setting the controller enable.

Typical applications

For typical applications, see Chapter "Examples (Page 688)".

Setting up optimization cycles/troubleshooting

We recommend that an optimization cycle that has been newly created or modified is executed and tested before the machine is shipped.

The reasons for this are as follows:

- For programming errors - or if the appropriate preconditions are not available (e.g. XML file with optimization strategy is missing), then the program is canceled. Based on the alarm that is output, the program line with the error can be identified and the error subsequently resolved. As soon as the optimization cycle can be executed without any errors, program stops should no longer occur for the end user.
- If the user interface-based automatic servo optimization (AST) had not been previously used on the machine, and the machine data is not set for the current recommendation for precontrol (e.g. MD33000 $MA_FIPO_TYPE = 3, MD32620 $MA_FFW_TYPE = 4), then the first time that the AST cycles are executed results in this machine data being set, and a power on (NC reset) or a channel reset is necessary.

Reference:

For information on user interface-based automatic servo optimization, see:

- CNC Commissioning Manual: NC, PLC, Drive

14.2 Commissioning

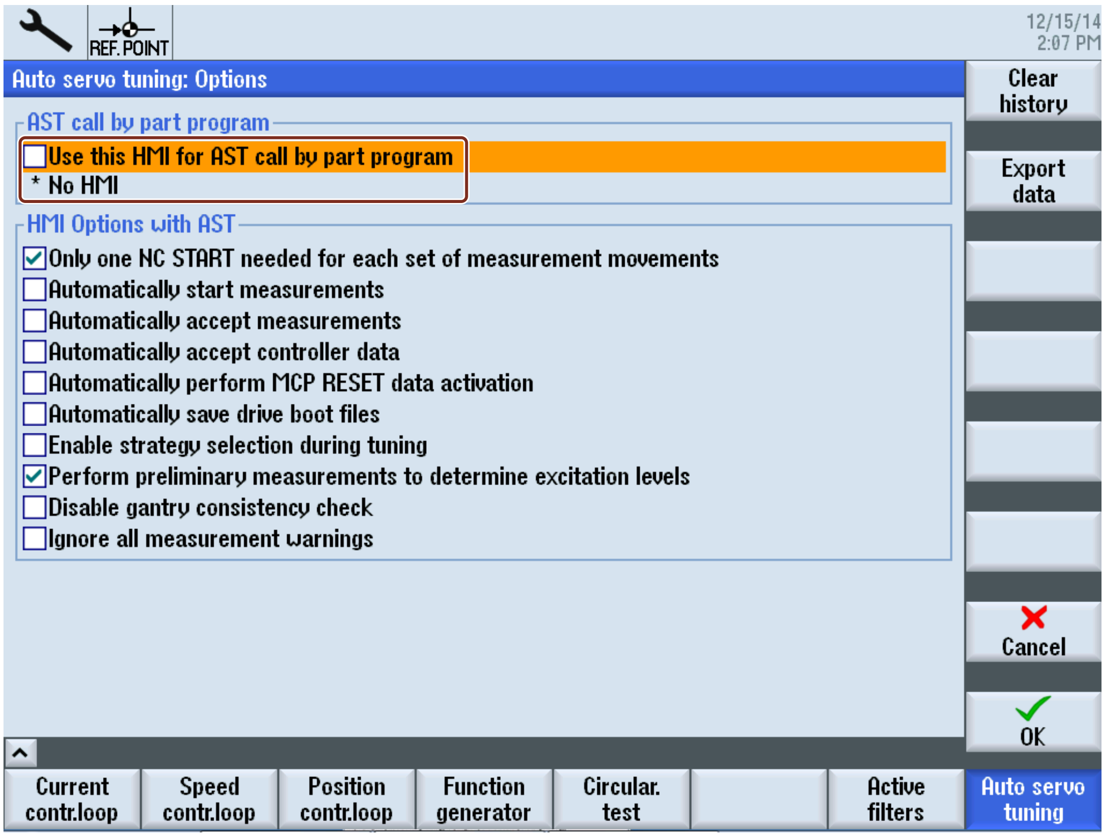
System requirements

- SINUMERIK 840D sl with SINUMERIK Operate
- CNC software as of version 4.7 SP1
- "Automatic post optimization with AST" option is set.

Machines with several HMI components

The "Automatic post optimization with AST" function can always be assigned to an HMI.

In order to change the actual assignment, in the HMI intended for the function, window ≫Automat. servo optimization: options≪ must be called, and option ≫Assign this HMI for "Call AST from the part program" ≪ selected.



Auto servo tuning: Options

AST call by part program

- [ ] Use this HMI for AST call by part program
- * No HMI

HMI Options with AST

- [✓] Only one NC START needed for each set of measurement movements
- [ ] Automatically start measurements
- [ ] Automatically accept measurements
- [ ] Automatically accept controller data
- [ ] Automatically perform MCP RESET data activation
- [ ] Automatically save drive boot files
- [ ] Enable strategy selection during tuning
- [✓] Perform preliminary measurements to determine excitation levels
- [ ] Disable gantry consistency check
- [ ] Ignore all measurement warnings

14.3 Programming

14.3.1 Overview

Predefined cycles for automatic post optimization with AST

SIEMENS provides machinery builders (OEMs) the following cycles so that they can program their own optimization cycles:

- CYCLE751 - Open/execute/close optimization session (Page 667)
- CYCLE752 - Add axis to an optimization session (Page 668)
- CYCLE753 - Select optimization mode (Page 671)
- CYCLE754 - Add/remove data set (Page 672)
- CYCLE755 - Backup/restore data (Page 673)
- CYCLE756 - Activate optimization results (Page 674)
- CYCLE757 - Save optimization data (Page 675)
- CYCLE758 - Change a parameter value (Page 679)
- CYCLE759 - Read parameter value (Page 681)

Internally, each of these cycles calls CYCLE750, which contains the MMC command with the actual function call.

Data handling

Archived in standard directories

Automatic servo optimization (AST) uses three standard directories in the file system to save specific data:

Directory path	To archive the following files
/user/sinumerik/nck/data/optimization	• XML files with axis and/or path-specific optimization data, which are automatically generated for the user interface-based automatic servo optimization. • XML files with axis and/or path-specific optimization data, which are automatically generated for automatic post optimization with AST using CYCLE757.
/user/sinumerik/nck/data/optimization/restore	• Backup files that are generated by the automatic post optimization with AST using CYCLE755.
/user/sinumerik/nck/data/optimization/data	• CSV files that are generated by the automatic post optimization with AST using CYCLE757.

Addressing using relative path data

The starting point for addressing an optimization file with relative path data is the data-specific standard directory in the file system. When addressing an XML file with axis-specific optimization data, the relative path would be specified, e.g. starting at standard directory */card/user/sinumerik/nck/data/optimization*.

The relative path is specified, starting without a forward slash ("/").

Subdirectories can be specified.

Example:

data/my_ast_ax1.xml

(corresponds to absolute path */user/sinumerik/nck/data/optimization/data/my_ast_ax1.xml*)

Addressing using absolute path data

Using absolute path data, optimization files outside the standard directories can be addressed.

When addressing an optimization file by specifying an absolute path, the complete file address is specified.

For Linux, the complete address comprises the following elements:

- Directory path
- File name

Example:

- */user/sinumerik/hmi/log/optimization/data/my_ast_ax1.xml*

Addressing files in external memories

When addressing optimization files in an external memory, the drive name must be specified at the beginning of the path.

All symbolic device names configured in *card/user/sinumerik/hmi/cfg/logdrive.ini* are permissible.

Examples:

- *LOCAL_DRIVE:/…* (local drive)
- *//DEV_5:/…* (user-defined drive)

14.3.2 CYCLE751 - Open/execute/close optimization session

Syntax

```
CYCLE751(<S_I_SESSIONCOMMAND>)
```

Parameter

No.	Parameters		
1	`<S_I_SESSIONCOMMAND>`		
	Data type:	INT	
	Value:	`0`	Not defined
		`1`	Open new optimization session **Note:** Multiple sessions cannot be open concurrently. Opening a new session closes the current session automatically.
		`2`	Close current optimization session
		`3`	Execute optimization functions of the current session **with** immediate transfer of the optimization results The scope of the optimization functions (measurement of the speed control loop, measurement of the position control loop, optimization of the path interpolation) to be executed depends on the axis-specific optimization strategies set from the user interface. The optimization results are transferred immediately (update the axis and drive parameters, update the optimization files).
		`4`	Execute optimization functions of the current session **without** immediate transfer of the optimization results Unlike 3, the optimization results are not transferred immediately. Transfer can be realized later in the optimization session using CYCLE756.

14.3.3 CYCLE752 - Add axis to an optimization session

After opening the optimization session, CYCLE752 specifies which axes should be reoptimized. If a reoptimization of the path interpolation is provided, CYCLE752 also specifies the axes required for the comparison of the path interpolation.

Syntax

```
CYCLE752(<S_I_AXIS>, <S_I_ACTIONREQUEST>, <S_B_ISPATHMEMBER>,
<S_SZ_FILENAME>)
```

Parameters

No.	Parameters	
1	`<S_I_AXIS>`	
	Meaning:	Machine axis number Specifies the axis to be added to the optimization session. **Note:** For gantry or master-slave couplings, only the leading or master axis of the coupling group should be specified.
	Data type:	INT

No.	Parameters			
2	`<S_I_ACTIONREQUEST>`			
	Meaning:	Specifies whether the stored optimization data (optimization strategy and optimization results) should be used or the axis with AST default settings reoptimized for the axis to be added. **Note:** If the axis has already been optimized and the optimization data has been saved in an optimization file, this data is normally accessed first. CYCLE753 can then specify the further procedure.		
	Data type:	INT		
	Value:	`0`	Not defined	
		`1`	Reoptimization with AST default settings The axis should be remeasured and reoptimized with the AST default settings. The selected AST settings are **not** considered on the user interface.	
		`2`	Optimization data from a user-defined optimization file The stored optimization data from the specified user-defined optimization file (see <S_SZ_FIL-ENAME> parameter) is used for the axis. **Note:** This setting is only possible if the axis has already been optimized once, and the results have been saved to a user-defined optimization file.	
		`3`	Optimization data from the standard optimization file The stored optimization data from the axis-specific standard optimization file is used for the axis. The selected AST settings are not considered on the user interface. **Note:** For the user interface-based automatic servo optimization, the standard optimization files are automatically generated to backup optimization information and data (optimization strategy, measuring and optimization results) and saved in the file system under */user/sinumerik/nck/data/optimization*. There is a standard optimization file for every axis – and a standard optimization file for the path interpolation.	
		For example, the setting <S_I_ACTIONREQUEST> = 2 or 3 is used in the following cases: • The axis should be post-optimized using the saved strategy and measurement settings. To do this, the saved optimization data is loaded with CYCLE752 and the deployed optimization mode defined with CYCLE753. • On completion of the reoptimization of the individual axes, the path interpolation must be reoptimized. • Saved optimization results (values for NC and drive parameters) should be restored.		
3	`<S_B_ISPATHMEMBER>`			
	Meaning:	Consideration of the axis for the last optimization step (optimization of the path interpolation)		
	Data type:	BOOL		
	Value:	`0` (`FALSE`)	Do **not** consider the axis for the optimization of the path interpolation	
		`1` (`TRUE`)	Consider the axis for the optimization of the path interpolation All axes, which are added using CYCLE752(, , **TRUE**,), are taken into account when optimizing path interpolation. This is realized within the scope of the optimization, e.g. started using CYCLE751(3).	

No.	Parameters	
4	`<S_SZ_FILENAME>`	
	Meaning:	Address of the user-defined, axis-specific optimization file (only relevant for <S_I_ACTIONREQUEST> = 2!) If the file was generated via the program-based automatic post-optimization with AST using CYCLE757, then as default setting, the file is in directory */user/sinumerik/nck/data/optimization* and it is not necessary to specify a directory path. However, the more frequent use case is that the file is generated via the user interface-based automatic servo optimization, and was saved in directory */user/sinumerik/hmi/log/optimization*. In addition to the file name, the address must also include the directory path (unless the file was subsequently copied to the standard directory */user/sinumerik/nck/data/optimization*). See also "Example 6: Measuring an axis without optimization (Page 703)".
	Data type:	STRING [100]

Note

Gantry/master-slave coupling

For gantry or master-slave couplings, with CYCLE752, only the leading or master axis of the coupling group is specified. The AST software automatically adds the coupled following or slave axes.

AST then automatically measures all axes of a coupling group – and optimizes them with identical controller settings. The position controller gains are set the same, the integral time in the speed controller is selected the same and the effective speed controller gain (relative to the overall moment of inertia) is selected the same. The current setpoint filters can be optimized differently.

Coupled axes must be referenced (homed) before the automatic optimization!

Note

Premeasurement

The decision for performing a premeasurement depends on the available axis optimization data.

- Cases in which the control unit performs a premeasurement for determining the measuring parameters:
  - The optimization strategy envisages a premeasurement; data from an earlier premeasurement is **not** available.
  - The optimization data does **not** specify any optimization strategy.
- Cases in which no premeasurement is performed:
  - The optimization strategy envisages a premeasurement; data from an earlier premeasurement is available.
  - The optimization strategy does **not** envisage any premeasurement (the optimization strategy takes precedence over the "Perform premeasurement for determining the excitation" option selected from the user interface).

14.3.4 CYCLE753 - Select optimization mode

CYCLE753 must be programmed when different strategy settings should act for the axis than those stored as standard or selected from the user interface. For example, this the case when an axis with user-defined strategy and measuring settings from a stored optimization file should be remeasured and reoptimized.

Syntax

```
CYCLE753(<S_I_AXIS>, <S_I_ACTION>, <S_B_SAVEBOOTFILES>)
```

Parameters

No.	Parameters			
1	`<S_I_AXIS>`			
	Meaning:	Machine axis number		
	Data type:	INT		
2	`<S_I_ACTION>`			
	Meaning:	Specifies the optimization mode for the specified axis.		
	Data type:	INT		
	Value:	`0`	Not defined	
		`1`	Match the path interpolation The axis (path axis) should not be remeasured and reoptimized, but rather considered and adapted only for the optimization of the path interpolation. The stored results from the optimization file are used for the axis. The optimization of the path interpolation means that the axis-specific machine data and drive parameters can change.	
		`2`	Reoptimization with a different optimization strategy based on stored measurements The axis should be reoptimized based on the results from an earlier measurement and currently valid optimization strategy changed from the user interface or CYCLE758. This mode is used, for example, to eliminate known periodic disturbance frequencies (see "Example 5: Reoptimize the speed control loop to eliminate known periodic disturbance frequencies. (Page 701)").	
		`3`	Measurement and reoptimization (→ **Mode for changed mechanics/loading!**) The mechanics/loading has changed. The axis should be remeasured and reoptimized using the user-defined strategy and measuring settings from an optimization file. The strategy and measuring settings can be adapted from the user interface or with CYCLE758.	
3	`<S_B_SAVEBOOTFILES>`			
	Meaning:	Save updated drive parameter values as persistent data		
	Data type:	BOOL		
	Value:	`0` (`FALSE`)	No	
		`1` (`TRUE`)	Yes (default)	

14.3.5 CYCLE754 - Add/remove data set

If the optimization results rather than the current data set should overwrite a specific axis or drive data set, the data set to be overwritten must be added to the data set list with CYCLE754. This list involves an invisible internal function list used to define data sets to be optimized. As default setting, only the currently selected data set is included in it. CYCLE754 must also be used to remove a data set from the list.

Note

Adding or removing an axis or drive data set using CYCLE754 only refers to the data set list, i.e an axis or drive data set removed from the list still remains in the control or in the drive.

Syntax

```
CYCLE754(<S_I_AXIS>, <S_I_ACTIONREQUEST>, <S_I_DATASETTYPE>,
<S_I_INDEX>)
```

Parameters

No.	Parameters		
1	`<S_I_AXIS>`		
	Meaning:	Machine axis number	
	Data type:	INT	
2	`<S_I_ACTIONREQUEST>`		
	Meaning:	Specifies whether the specified axis or drive data set should be added to or removed from the data set list.	
	Data type:	INT	
	Value:	0	Not defined
		1	Add a data set
		2	Remove a data set
		3	Remove all data sets from the list **Note:** If the list is empty, the optimization results can be written only to the active data set.
3	`<S_I_DATASETTYPE>`		
	Meaning:	Data set type	
	Data type:	INT	
	Value:	0	Not defined
		1	Axis data set (servo parameter set)
		2	Drive data set (DDS)

No.	Parameters			
4	`<S_I_INDEX>`			
	Meaning:	Data set index Specifies the data set to be added and into which the optimization results should be written or should be removed from the list. **Note:** The following applies to the **axis**data set index: <n> = parameter set number **- 1** Example: Index 0 ≙ parameter set 1		
	Data type:	INT		
	Range of values:	`-1`, `0`, `1`, ... `<MAX_INDEX>`		
		`-1`	active data set	

14.3.6 CYCLE755 - Backup/restore data

Using CYCLE755, backup files can be created in the file system that permit the backup of the actual data, and therefore facilitate specific data restoration at any time during the optimization session. Data restoration is also called using CYCLE755. Further, CYCLE755 is used to query the existence of a backup file as well as to delete backup files.

Note

The backup files are no longer required after the optimization session has been successfully completed. This is the reason that these should be deleted again at the end of the optimization session with CYCLE755.

Typical examples for using CYCLE755 include:

- Undoing data changes, which are only to be temporarily valid.
- Restoring data after an interrupted optimization session.
  A program that queries the existence of a backup file, and for a positive result, restores the data from this file, can be used to return to a defined initial state after an optimization session has been interrupted.

Syntax

```
CYCLE755(<S_I_ACTIONREQUEST>, <S_SZ_NAME>, <S_SZ_GUD_BOOL_NAME>)
```

Parameters

No.	Parameters		
1	`<S_I_ACTIONREQUEST>`		
	Data type:	INT	
	Value:	`0`	Not defined
		`1`	Backup data for a later restore This setting saves the data values at the time of the cycle call in the specified file. The following parameters define the content of the data backup: • the currently valid optimization strategy • the axes currently added to the optimization session • the current list of parameter sets to be written
		`2`	Restore data This setting restores the data from the specified file. Requirements: • The optimization session is still active. • The axis and data set selection at the time of the backup is still available.
		`3`	Delete backup file
		`4`	Querying the existence of a backup file The result of the query is saved to a channel-global user variable (GUD), type BOOL (see parameter <S_SZ_GUD_BOOL_NAME>).
2	`<S_SZ_NAME>`		
	Meaning:	Name of backup file The file name can be freely selected with the following restriction: it is not permissible that the sequence of letters "AST" is located at the beginning of the file name. Specifying a path (absolute or relative) is only required if the file is not to be saved to the standard directory */user/sinumerik/nck/data/optimization/restore*.	
	Data type:	STRING [100]	
3	`<S_SZ_GUD_BOOL_NAME>`		
	Meaning:	Name of the GUD, in which the result of the file query should be saved (only relevant, if <S_I_ACTIONREQUEST> = 4!) **Note:** The specified GUD must be defined in a definition file as a channel-global variable, type BOOL: DEF CHAN BOOL	
	Data type:	STRING [50]	

14.3.7 CYCLE756 - Activate optimization results

CYCLE756 can be used to activate the optimization results at a specific time / under specific conditions. The conditions must be queried in the manufacturer cycle. CYCLE756 can also specify which control parameters should be active: original data, axis-optimum data or path-optimum data.

Syntax

```
CYCLE756(<S_I_AXIS>, <S_I_REGULATOR_ROLE>, <S_B_SAVEBOOTFILES>)
```

Parameters

No.	Parameters		
1	`<S_I_AXIS>`		
	Meaning:	Machine axis number	
	Data type:	INT	
2	`<S_I_REGULATOR_ROLE>`		
	Meaning:	Specifies which optimization results should act for the specified axis in the control unit and in the drive. **Note:** If the axis is part of a gantry group or a master-slave coupling, then optimization factors are included in the optimization results, which would be necessary to adapt the dynamic response of all axes of the coupling group.	
	Data type:	INT	
	Value:	`0`	Not defined
		`1`	The original data should act for the specified axis.
		`2`	The current optimization results **without** optimization of the path interpolation should become active for the specified axis (axis-optimum data)
		`3`	The current optimization results **with** optimization of the path interpolation should become active for the specified axis (path-optimum data).
3	`<S_B_SAVEBOOTFILES>`		
	Meaning:	Save updated drive parameter values as persistent data	
	Data type:	BOOL	
	Value:	`0` (`FALSE`)	No
		`1` (`TRUE`)	Yes (default)

Application

Some situations for which CYCLE756 is useful:

- CYCLE751(4) executes the optimization functions of the current session without transferring the optimization results immediately. The optimization results can be activated later during the optimization session with CYCLE756. Prior to the activation, the data can be read and tested with CYCLE758/759.
- CYCLE752 loads the saved optimization data from an optimization file. CYCLE756 activates the loaded data.

14.3.8 CYCLE757 - Save optimization data

Contrary to the user interface-based automatic servo optimization, where optimization data are automatically saved (standard optimization files), saving optimization data for program-based automatic post optimization with AST must be explicitly called by programming CYCLE757.

The following user-defined optimization files can be created in CYCLE757:

- Optimization files in the XML format:
  – for the axis-specific optimization data
  – for the data to optimize path interpolation
- Optimization files in the CSV format with frequency, amplitude and phase of the following frequency responses:
  – measured speed-controlled system
  – measured mechanical frequency response
  – calculated closed speed controller
  – calculated closed position controller
  – measured closed speed controller
  – measured closed position controller

If a path is not specified, the files are saved in the file system in the following standard directories:

- XML files: */user/sinumerik/nck/data/optimization*
- CSV files: */user/sinumerik/nck/data/optimization/data*

NOTICE

Data loss

If CYCLE757 is not programmed, measurement data as well as parameters for assessment and offline optimization are no longer available after the optimization session has been closed. However, the optimization (depending on the strategy) is written to the axis and drive parameters.

Remedy: Save optimization data to be backed up before closing the optimization session with CYCLE757.

Syntax

```
CYCLE757(<S_I_AXIS>, <S_SZ_FILENAME>, <S_I_CONTENT_TYPE>,
<S_I_FORMATFILTER>, <S_I_SUPPINFO>)
```

Parameters

No.	Parameters	
1	`<S_I_AXIS>`	
	Meaning:	Machine axis number
	Data type:	INT

No.	Parameters	
2	`<S_SZ_FILENAME>`	
	Meaning:	Name of the optimization file to be created (**with** file extension ".xml" or ".csv") The file name can be freely selected with the following restriction: it is not permissible that the sequence of letters "AST" is located at the beginning of the file name. Specifying a path (absolute or relative) is only required if the file is not to be saved to the standard directory. Examples: ● AXIS.xml ● PATH.xml ● SPEEDCTRL_PLANT.csv ● POSCTRL_MECHRESP.csv Special case: ● Empty string (""); only valid, if <S_I_CONTENT_TYPE> = 1 or 2 Entering an empty string means that the data of the program-based automatic post optimization are written to the standard optimization file for the axis (<S_I_CONTENT_TYPE> = 1) or for the path interpolation (<S_I_CONTENT_TYPE> = 2). **Notice:** The standard optimization files contain the data of the user interface-based automatic servo optimization. By programming <S_SZ_FILENAME> = "", these are overwritten by the data of the program-based automatic post optimization.
	Data type:	STRING [100]

No.	Parameters		
3	`<S_I_CONTENT_TYPE>`		
	Meaning:	Specifies which optimization data should be saved in the created optimization file.	
	Data type:	INT	
	Value:	`0`	Not defined
		`1`	Creates an XML file in which the axis-specific optimization data for the specified axis is saved. The following data is included: • Measuring conditions • Measurement results • Optimization strategy • Optimization results **Note:** If the axis is part of a gantry group or a master-slave coupling, then the optimization information for the axis, includes the optimization information for all other axes of the coupling group.
		`2`	Creates an XML file in which the data for optimization of the path interpolation is saved. The following data is included: • List of all involved axes. • Strategy for optimization of the path interpolation
		`3`	Creates a CSV file, in which the data for the frequency characteristic of the measured speed controlled system is to be saved. The data is stored as a 3-column table: • Column 1: Frequency [Hz] • Column 2: Linear amplitude • Column 3: Phase [rad]
		`4`	Creates a CSV file in which the data for the measured mechanical frequency response is saved. The data is stored as a 3-column table: • Column 1: Frequency [Hz] • Column 2: Linear amplitude • Column 3: Phase [rad]
		`5`	Creates a CSV file, in which the data for the frequency response of the calculated closed speed controller is saved. The data is stored as a 3-column table: Column 1: Frequency [Hz] Column 2: Linear amplitude Column 3: Phase [rad]
		`6`	Creates a CSV file, in which the data for the frequency response of the calculated closed position controller is saved. The data is stored as a 3-column table: Column 1: Frequency [Hz] Column 2: Linear amplitude Column 3: Phase [rad]
		`7`	Creates a CSV file, in which the data for the frequency response of the measured closed speed controller is saved. The data is stored as a 3-column table: Column 1: Frequency [Hz]

No.	Parameters		
			Column 2: Linear amplitude Column 3: Phase [rad]
		`8`	Creates a CSV file, in which the data for the frequency response of the measured closed position controller is saved. The data is stored as a 3-column table: Column 1: Frequency [Hz] Column 2: Linear amplitude Column 3: Phase [rad]
4	`<S_I_FORMATFILTER>`		
	Meaning:	Reserved	
	Data type:	INT	
5	`<S_I_SUPPINFO>`		
	Meaning:	Reserved	
	Data type:	INT	

14.3.9 CYCLE758 - Change a parameter value

CYCLE758 changes individual strategy and measuring settings prior to the optimization or the optimization results after the optimization.

Syntax

```
CYCLE758(<S_I_AXIS>, <S_I_PARAMID>, <S_I_MEASTYPE>, <S_I_MEASINDEX>,
<S_SZ_NEWVALUE>)
```

Parameters

No.	Parameters	
1	`<S_I_AXIS>`	
	Meaning:	Machine axis number (not relevant for parameters that affect the optimization of the path interpolation!)
	Data type:	INT
2	`<S_I_PARAMID>`	
	Meaning:	ID number of the parameter whose value should be changed Parameter IDs, see "List of the parameters for the automatic servo optimization (Page 682)".
	Data type:	INT

No.	Parameters		
3	`<S_I_MEASTYPE>`		
	Meaning:	Measuring type to which the parameter refers (relevant only for measuring parameters!) Uniquely identifies measuring parameters with the same ID number.	
	Data type:	INT	
	Value:	`1`	Mechanical frequency response
		`2`	Closed position control loop
		`3`	Closed speed control loop
		4	Speed-controlled system
		5	Speed setpoint step change
4	`<S_I_MEASINDEX>`		
	Meaning:	Measurement index to which the parameter refers (relevant only for measuring parameters!) Uniquely identifies measuring parameters with the same ID number that refer to the same measuring type. It must be specified only when multiple measurements of a measuring type exist.	
	Data type:	INT	
5	`<S_SZ_NEWVALUE>`		
	Meaning:	New parameter value	
	Data type:	STRING [20]	
	Value:	The value should be specified as a character string in inverted commas. The character string can comprise a maximum of 20 characters. Values are only permissible that correspond to the data type of the parameter to be changed - and that lie within the valid range. Examples: 1. The parameter to be changed is of data type BOOL (e.g. <S_I_PARAMID> = 157). Value range: 0 (= FALSE), 1 (= TRUE) For <S_SZ_NEWVALUE>, the following permissible character string is obtained: "true", "false", "TRUE", "FALSE", "0" or "1" Inadmissible character strings would be, for example: 0, 1, true, false, "2", "my_bool" 2. The parameter to be changed is of data type REAL (e.g. <S_I_PARAMID> = 34). In this particular case, with <S_SZ_NEWVALUE> theoretically, every real number within the value range ± (~ $2.2*10^{-308}$ … ~ $1.8*10^{+308}$) can be specified. Permissible character strings are, for example: "14.1", ".0023" Inadmissible character strings would be, for example "true", 14.1, "my_real", "R1"	

Note

Gantry/master-slave coupling

If the measurement parameters of a specific following or slave axis must be changed, then in CYCLE758 the corresponding number of this axis must be specified (**not** the number of the leading or master axis, as when adding coupled axes with CYCLE752 (Page 668)).

14.3.10 CYCLE759 - Read parameter value

CYCLE759 reads the strategy and measuring parameters and the optimized controller parameters. This is useful for checking the file loaded with CYCLE752 prior to the optimization or reading the results after the optimization (e.g. determined total inertia, determined gain).

Syntax

```
CYCLE759(<S_I_AXIS>, <S_I_PARAMID>, <S_I_MEASTYPE>, <S_I_MEASINDEX>,
<S_SZ_GUDRESULT>)
```

Parameters

No.	Parameters		
1	`<S_I_AXIS>`		
	Meaning:	Machine axis number (not relevant for parameters that affect the optimization of the path interpolation!)	
	Data type:	INT	
2	`<S_I_PARAMID>`		
	Meaning:	ID number of the parameter whose value should be read	
		Parameter IDs, see "List of the parameters for the automatic servo optimization (Page 682)".	
	Data type:	INT	
3	`<S_I_MEASTYPE>`		
	Meaning:	Measuring type to which the parameter refers (relevant only for measuring parameters!)	
		Uniquely identifies measuring parameters with the same ID number.	
	Data type:	INT	
	Value:	1	Mechanical frequency response
		2	Closed position control loop
		3	Closed speed control loop
		4	Speed-controlled system
		5	Speed setpoint step change
4	`<S_I_MEASINDEX>`		
	Meaning:	Measurement index to which the parameter refers (relevant only for measuring parameters!)	
		Uniquely identifies measuring parameters with the same ID number that refer to the same measuring type. It must be specified only when multiple measurements of a measuring type exist.	
	Data type:	INT	
5	`<S_SZ_GUDRESULT>`		
	Meaning:	Name of the global user variable (GUD) into which the parameter value should be copied.	
		Note: The specified GUD must be defined in a definition file. The data type of the GUD must match the data type of the parameter.	
	Data type:	STRING [20]	

14.3.11 List of the parameters for the automatic servo optimization

The ID of a parameter for the automatic servo optimization whose value should be changed with CYCLE758 or read with CYCLE759 can be taken from the following table:

ID	Meaning	Unit	Data type	Value range	Write with CYCLE758	Read with CYCLE759
Measuring conditions						
001	Start position of the axis during a measurement	mm or degrees	REAL		-	+
002	Direction sequence during a measurement • 1 = plus • 2 = minus • 3 = plus, then minus • 4 = minus, then plus		INT	1, 2, 3, 4	+	+
003	Number of measurement repetitions for determining the frequency range averaging		INT	0 to	+	+
004	Number of time range windows in each time range averaging		INT	1 to	+	+
005	Settling time: The wait time after the PRBS excitation before the first measuring window is acquired	s	REAL	0 to	+	+
006	Measurement bandwidth This value only determines the measurement bandwidth (i.e. excitation and sampling frequency). This allows the frequency resolution to be influenced.	Hz	REAL	Float: 0 ... fs/2	+	+
007	Time for performing an individual measurement (without repetition, ramp-up and settling time)	s	REAL		-	+
008	Time required to bring the speed offset to the setpoint	s	REAL		-	+
009	Estimated travelled distance during the maximum traversing motion in the repetition sequence	m or rad	REAL		-	+
010	Estimated total time of the axis motion during the measurement	s	REAL		-	+
011	Specifies the measurement type: • 1 = frequency response of the mechanical parts • 2 = closed position control loop • 3 = closed speed control loop • 4 = speed-controlled system • 5 = speed setpoint increment		INT		-	+
012	Specifies whether the axis motion lies within the traverse range limits		BOOL	0 (= FALSE) 1 (= TRUE)	-	+

ID	Meaning	Unit	Data type	Value range	Write with CYCLE758	Read with CYCLE759
013	Specifies whether a measurement is performed with the maximum bandwidth		BOOL	0 (= FALSE) 1 (= TRUE)	-	+
021	Amplitude of the excitation	Nm, N, m/s, rad/s, m, rad (dependent on the measurement type1))	REAL	Dynamic	+	+
022	Velocity offset	m/s or rad/s (depending on measurement type1))	REAL	Dynamic	+	+
Speed control (controller)						
031	Proportional gain for the speed control loop	Nms/rad or Ns/m	REAL		+	+
032	Integral time for the speed control loop	s	REAL		+	+
033	Motor moment of inertia	kgm^2 or kg	REAL		-	+
034	Total moment of inertia of the axis expressed in the units of the motor	kgm^2 or kg	REAL		+	+
035	Specifies whether the reference model is activated for the speed control		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
036	Frequency of the PT2 reference model	Hz	REAL		+	+
037	Damping of the PT2 reference model		REAL		+	+
038	Ratio of the Kp proportional gain to the moment of inertia that represents the 1/s bandwidth of the idealized PT1-equivalent (proportional-controlled) systems	1/s	REAL		-	+
Position control (controller)						
101	Position control loop gain	1000/min	REAL		+	+
102	Specifies whether DSC (dynamic stiffness control) is active		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
103	Specifies the precontrol type: • 0 = none • 1 = speed precontrol • 2 = speed and torque precontrol		INT	0, 1, 2	+	+
104	Total moment of inertia of the axis as parameter for the torque precontrol	kg m^2 or kg	REAL		+	+

ID	Meaning	Unit	Data type	Value range	Write with CYCLE758	Read with CYCLE759
105	Delay of the speed precontrol in the drive (e.g. p1429)	s	REAL		+	+
106	Equivalent time constant of the speed control loop as parameter for the speed precontrol	s	REAL		+	+
107	Adapted parameter value 106 for considering the following effects: • Inconsistent delays • Changes carried out by the NC	s	REAL		+	+
108	Equivalent time constant of the current control loop as parameter for the torque precontrol	s	REAL		+	+
109	Adapted parameter value 108 for considering the following effects: • Inconsistent delays • Changes carried out by the NC	s	REAL		+	+
110	PT1 time constant of the axis to adapt the dynamic response (MD32910 $MA_DYN_MATCH_TIME)	s	REAL		+	+
111	Time constant used for the dynamic response adaptation for the axis-specific setpoint phase filter (MD32895 $MA_DESVAL_DELAY_TIME)	s	REAL		+	+
112	Calculated total time constant for the positioning response, including precontrol and all prefilters. This time constant is only intended to provide users with information. You can use it as a target time constant for the position controller strategy for another axis. By forcing the same total time constant, interpolation fits already after axis optimization.	s	REAL		-	+
Position control (strategy)						
151	Adaptation factor (reduction) for the optimized maximum Kv value		REAL	0.1 ... 1.0	+	+
152	Not defined					
153	Specifies the additional method for maximizing the Kv value: • 0 = none (standard) • 1 = PT1 speed filter 2 • 2 = damping of the optimum speed controller • 3 = PT1 speed filter 1		INT	0 ... 3	+	+
154	DSC		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
155	Select precontrol type: • 0 = none • 1 = speed precontrol • 2 = speed and torque precontrol		INT	0, 1, 2	+	+

ID	Meaning	Unit	Data type	Value range	Write with CYCLE758	Read with CYCLE759
156	Maximum permissible Kv value	1000/min	REAL	0.1 ... 99.0	+	+
157	Specifies whether the precontrol should always be active for the axis		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
158	Setpoint for the equivalent time of the position controller overall response (including precontrol and setpoint filter)	s	REAL		+	+
159	Indicates whether an attempt should be made to attain the nominal equivalent time		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
160	Specifies the jerk filter technology to be selected: • 0 = keep the existing jerk filter settings in the machine data • 1 = deactivates the jerk filter • 2 = use jerk filter, type 2 (= sliding average value generation)		INT	0 ... 2	+	+
161	Time constant (MD32410 $MA_AX_JERK_TIME) for type 2 jerk filter (parameter 160 = 2)	s	REAL	0 … 0.5	+	+
Speed control (strategy)						
201	Setpoint for the amplitude reserve for the optimization	dB	REAL		+	+
202	Setpoint for the phase reserve for the optimization	deg	REAL		+	+
203	Indicates whether the reference model should be activated		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
204	Specifies the permissible minimum value for the integral time of the speed control loop	s	REAL	0.001 ... 1.000	+	+
205	Specifies whether the design of band-stop filters should deploy a constant reduction (for the case that significantly more poles are available as filter)		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
206	Optimization aggressiveness of the speed controller A higher value produces a better optimization result coupled with a reduction of the robustness		REAL	0 ... 1	+	+

ID	Meaning	Unit	Data type	Value range	Write with CYCLE758	Read with CYCLE759
207	Kp upper limit based on the bandwidth of a PT1-equivalent proportionally-controlled system Specifying a maximum speed controller bandwidth is an additional rule to limit the Kp. In so doing, the dynamic response of the speed control is limited in the form of a "Control bandwidth" (speed controller response in the frequency domain). Just as before, amplitude and phase reserves must be maintained. As long as the amplitude and phase reserve permit a higher Kp, Kp is calculated under the following assumptions: • The total moment of inertia is a rigid mass. • The speed control loop can be approximated as a first order (PT1) lowpass filter.	Hz	REAL		+	+
208	Indicates whether a gain filter for eliminating frequency-specific periodic disturbances should be designed and deployed		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
209	Frequency of the optional gain filter for eliminating frequency-specific periodic disturbances	Hz	REAL		+	+
Optimization of path interpolation (strategy)						
251	Indicates whether an attempt should be made to achieve an agreement with the effective Kp value		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
252	Indicates whether an attempt should be made to achieve an agreement with the Tn integral time		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
253	The maximum proportion with which the Kp value may be reduced to order to achieve an agreement with the effective Kp value		REAL	0.1 ... 1.0	+	+
254	Specifies whether non-agreements of the Kv value between axes are permitted		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
255	Indicates whether the axes without precontrol or with speed precontrol must have matched reference models		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
256	Specifies the strategy for the path compensation: • 0 = adapt the equivalent time of the control loop with controller parameters and/or MD32800/MD32810 • 1 = use MD32895 (time constant for the axial setpoint phase filter) for adaptation		INT	0, 1	+	+
257	Specifies whether the precontrol should always be active for all axes in the group		BOOL	0 (= FALSE) 1 (= TRUE)	+	+
258	Specifies whether the equivalent times should be identical for all axes		BOOL	0 (= FALSE) 1 (= TRUE)	+	+

$^{1)}$ The parameter value unit depends on the measurement type:

- The following measurements were carried out in the drive. This is the reason that the units refer to the motor side. The resulting axis velocity is obtained, taking into account the gearbox ratios or spindle pitch. The velocity specified here is only equal to the axis velocity for linear motors and torque motors (without gearbox). Here, it should be noted that the axis velocity is specified in the AST screen forms, e.g. mm/min. Here, the numerical values cannot be taken from the screen forms.

	Unit			
	Parameter 021: Amplitude of the excitation		Parameter 022: Velocity offset	
Measurement type	Rotary motor	Linear motor	Rotary motor	Linear motor
Speed-controlled system	Nm	N	rad/s	m/s
Closed speed control loop	rad/s	m/s	rad/s	m/s

- The following measurement were carried out via the control. Here, the units refer to the axis, i.e. to the load side. It may be necessary to convert to SI units.

	Unit			
	Parameter 021: Amplitude of the excitation		Parameter 022: Velocity offset	
Measurement type	Rotary axis	Linear axis	Rotary axis	Linear axis
Mechanical frequency response	rad/s	m/s	rad/s	m/s
Closed position control loop	rad	m	rad/s	m/s

14.4 Diagnostics

If an error occurs when executing cycles, then an appropriate HMI alarm is output (number range 150001 to 150004) as well as NC alarm 61840. Based on the block number specified in alarm 61840, the user can define which NC block with a CYCLE75x call caused the error.

Reference:
For alarm descriptions, refer to the Diagnostics Manual Alarms.

14.5 Supplementary conditions

"Automatic post optimization with AST" for active coupling

The following table lists which coupling function support the "Automatic post optimization with AST":

Coupling function	"Automatic post optimization with AST" can be used?
Gantry axes	+
Master-slave coupling	+
Coupled motion	-

Coupling function	"Automatic post optimization with AST" can be used?
Leading value coupling	-
Electronic gear	-
Synchronous spindle	-

14.6 Examples

14.6.1 Example 1: Measuring an axis with AST default settings

The rotary axis (here: 4th axis) driven with a torque motor depends greatly on the inertia of the clamped mass. If the load inertia changes, the axis must be measured again. No controller values for AST should be overwritten; the optimization, however, is determined with new measurements. The results, such as the complete inertia, can be read with CYCLE759. The newly calculated values also indicate whether the controlled system has changed. In specific cases, switching can be made to predefined controller data sets or the acceleration changed. The AST default setting includes the speed and position controller optimization with "normal" optimization objective. The determined controller data is, however, not activated in this example.

Programming

Definition of global user variables (GUD)

To read out the complete moment of inertia, a new type CHAN REAL variable must be defined in file MGUD.DEF (it is possible that this file must be newly created):

```
; Definition of GUD to read out the total moment of inertia:
DEF CHAN REAL _AST_R_ESTINERTIA
```

Optimization cycle

Note

In the following program example, those cycle calls required for the measurement are highlighted **bold**.

Any cycle calls not highlighted are optional.

```
DEF INT myaxiswithnewload=4
; Switch to a safe data set.
; Traverse axis to the safe position / home position.
; Is the optimization strategy set from the user interface?
; Is a destination data set (DDS) available that may be overwritten?

; Open the optimization session.
```

```
CYCLE751(1)

; Optimize axis 4: Optimization with a selected strategy based on a new
measurement.
; Optimization with default values (the selection from the AST user
interface is not used).
; No path interpolation, no values are written by AST.
CYCLE752(myaxiswithnewload,1,false)

; Save the original state in a file.
CYCLE755(1,"restorepoint1")

; Start the optimization without activating the results.
CYCLE751(4)

; Save the optimization data in an XML file.
CYCLE757(myaxiswithnewload,"axis_retuned.xml",1,,)

; Export the speed-controlled system.
CYCLE757(myaxiswithnewload,"speedctrlplant.csv",3,,)

; Fetch the inertia.
CYCLE759(myaxiswithnewload,34,,,"_AST_R_ESTINERTIA")

; If required check the optimization results, and then activate.
; Do not activate the results.
; Use predefined parameter sets.

; Close the optimization session.
CYCLE751(2)

; Switch to the newly optimized DDS or select the predefined DDS depending
on the load inertia.
; Limit the acceleration depending on the new inertia.

M17
```

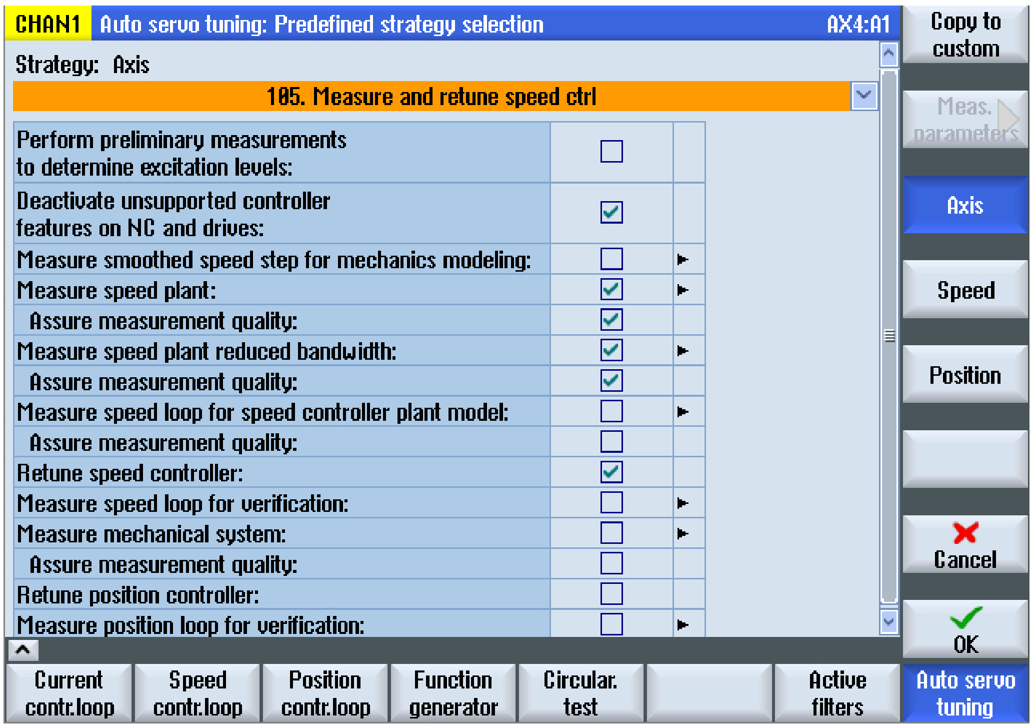
14.6.2 Example 2: Reoptimize the speed controller of an axis

The controller setting of the rotary axis (here: 4th axis), which is driven by a torque motor, depends on the clamping equipment and the load moment of inertia. If the clamping equipment or the load inertia changes, the axis must be measured again. A new controller parameter set must be determined (a DDS will be overwritten), otherwise the axis would be unstable because of the new resonances. New current setpoint filters must probably by set. Only the speed controller is reoptimized. Because the position control remains identical and the precontrol should not be changed, a similar bandwidth of the speed controller should be attained again. For this purpose, a target bandwidth can be selected in the speed controller optimization strategy. This is set to 100 Hz for the "normal compensation" optimization objective. If the new load inertia and the clamping make many filters necessary and the ratio from the load inertia / motor inertia is large, this objective may possibly not be achieved. In this case, a message is issued. And the axis machine data is still set correctly from the first optimization (initial commissioning). This means that the precontrol mode (MD32620 $MA_FFW_MODE), DSC (MD32640 $MA_STIFFNESS_CONTROL_ENABLE), position controller gain (MD32200 $MA_POSCTRL_GAIN) and equivalent time precontrol (MD32800 $MA_EQUIV_CURRCTRL_TIME or MD32810 $MA_EQUIV_SPEEDCTRL_TIME) are set correctly so that the loop interpolation is correct.

The results of the reoptimization, such as the complete inertia, can be read with CYCLE759. The newly optimized speed controller is written to another data set. The path interpolation is not considered here, because it is assumed that the precontrol data does not change significantly or another axis has a larger time constant in MD32800 $MA_EQUIV_CURRCTRL_TIME. The optimization strategy for axis and speed controllers, selected from the user interface, is deployed and was stored in the XML file for this axis (*/user/sinumerik/nck/data/optimization/AST_AX4_A1….xml*).

Preparation

- Select optimization strategy 105 from the user interface:



CHAN1 Auto servo tuning: Predefined strategy selection AX4:A1

Strategy: Axis

105. Measure and retune speed ctrl

Perform preliminary measurements to determine excitation levels:	☐	
Deactivate unsupported controller features on NC and drives:	✓	
Measure smoothed speed step for mechanics modeling:	☐	▸
Measure speed plant:	✓	▸
Assure measurement quality:	✓	
Measure speed plant reduced bandwidth:	✓	▸
Assure measurement quality:	✓	
Measure speed loop for speed controller plant model:	☐	▸
Assure measurement quality:	☐	
Retune speed controller:	✓	
Measure speed loop for verification:	☐	▸
Measure mechanical system:	☐	▸
Assure measurement quality:	☐	
Retune position controller:	☐	
Measure position loop for verification:	☐	▸

- If necessary, change the optimization objective (fast, moderate, robust) with the "Speed" softkey.

Programming

Definition of global user variables (GUD)

To read out the complete moment of inertia, a new type CHAN REAL variable must be defined in file MGUD.DEF (it is possible that this file must be newly created):

```
; Definition of GUD to read out the total moment of inertia:
DEF CHAN REAL _AST_R_ESTINERTIA
```

Optimization cycle

Note

In the following program example, those cycle calls required for the reoptimization are highlighted **bold**.

Any cycle calls not highlighted are optional.

```
DEF INT myaxiswithnewload=4
; Switch to a safe data set.
; Traverse axis to the safe position / home position.
; Is the optimization strategy set from the user interface?
; Is a destination data set (DDS) available that may be overwritten?

; Open the optimization session.
CYCLE751(1)

; Optimize axis 4: Optimization with a selected strategy based on a new
measurement.

; Add axis 4.
; Accept the optimization strategy from the standard optimization file (XML
file).
; The axis is not considered for an optimization of the path interpolation.
CYCLE752(myaxiswithnewload,3,false)

; Remove the current DDS from the data set list.
CYCLE754(myaxiswithnewload,2,2,-1)

; Add DDS3 to the data set list and overwrite after optimization.
CYCLE754(myaxiswithnewload,1,2,3)

; Save the original state in a file.
CYCLE755(1,"restorepoint1")

; Start the optimization by activating the results.
; CYCLE751(3)

; Start optimization without activation of the results, if necessary, check
the results first.
CYCLE751(4)

; Save the optimization data in an XML file.
CYCLE757(myaxiswithnewload,"axis_retuned.xml",1,,)

; Export the speed-controlled system.
CYCLE757(myaxiswithnewload,"speedctrlplant.csv",3,,)

; The inertia can be used to limit the acceleration:
; Fetch the inertia.
CYCLE759(myaxiswithnewload,34,,,"_AST_R_ESTINERTIA")

; If required check the optimization results, and then activate.

; Activate result:
; Activate axis-optimum results.
; Save boot files.
CYCLE756(myaxiswithnewload,2,1)

; Close the optimization session.
CYCLE751(2)

; Switch to the newly optimized DDS or limit the acceleration depending on
the new inertia.

M17
```

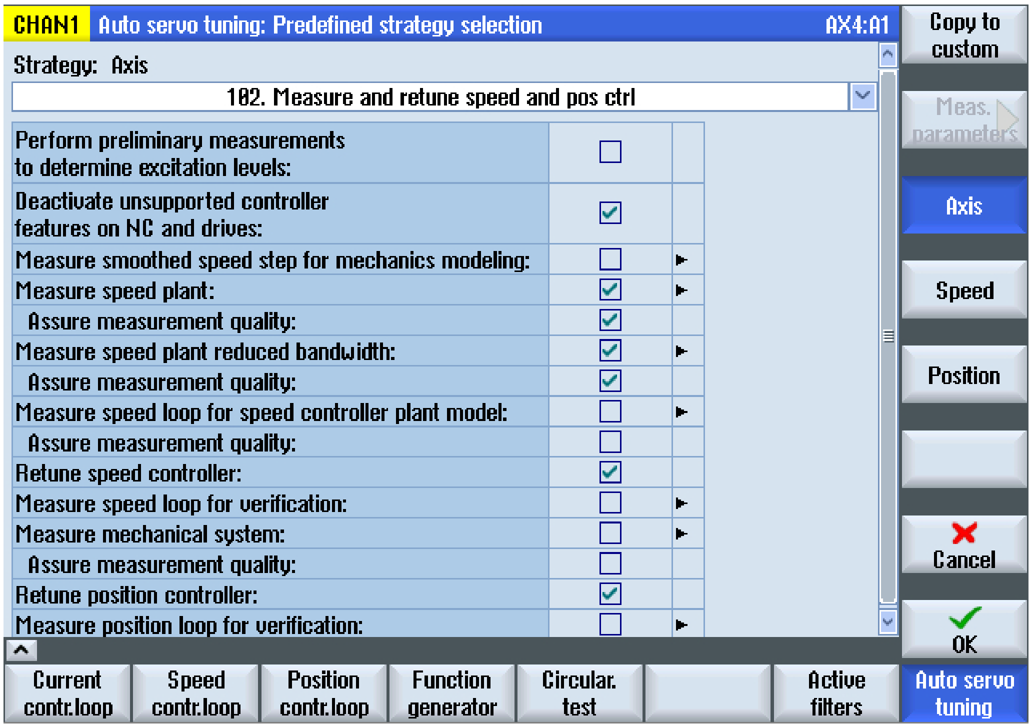
14.6.3 Example 3: Reoptimize the speed controller and the position controller of an axis

The controller setting of the rotary axis (here: 4th axis), which is driven by a torque motor, depends on the clamping equipment and the load moment of inertia. If the clamping equipment or the load inertia changes, the axis must be measured again. New current setpoint filters must probably by set. The speed controller must be reoptimized. Because it is not known whether the low-frequency behavior has changed, the position controller must also be reoptimized and the associated mechanical frequency response measured.

The results, such as the complete inertia, can be read with CYCLE759. The newly optimized speed controller is written to another data set. The newly optimized position controller can be written to any desired axis parameter set. The correct path interpolation is guaranteed by a target value for the (precontrolled) positioning behavior being specified in accordance with the previously optimized axes. The optimization strategy for axis and speed controllers, selected from the user interface, is deployed and was stored in the XML file for this axis (*/user/sinumerik/nck/data/optimization/AST_AX4_A1….xml*).

Preparation

- Select optimization strategy 102 from the user interface:



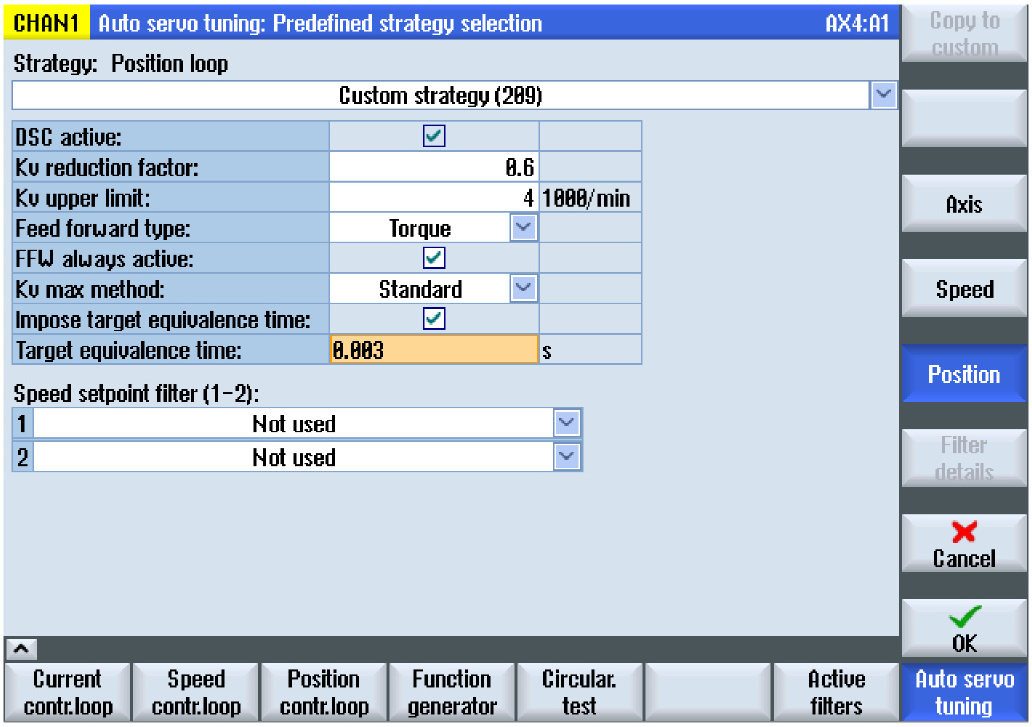
CHAN1 Auto servo tuning: Predefined strategy selection AX4:A1

Strategy: Axis

102. Measure and retune speed and pos ctrl

Perform preliminary measurements to determine excitation levels:	☐	
Deactivate unsupported controller features on NC and drives:	✓	
Measure smoothed speed step for mechanics modeling:	☐	▶
Measure speed plant:	✓	▶
Assure measurement quality:	✓	
Measure speed plant reduced bandwidth:	✓	▶
Assure measurement quality:	✓	
Measure speed loop for speed controller plant model:	☐	▶
Assure measurement quality:	☐	
Retune speed controller:	✓	
Measure speed loop for verification:	☐	▶
Measure mechanical system:	☐	▶
Assure measurement quality:	☐	
Retune position controller:	✓	
Measure position loop for verification:	☐	▶

- Press the "Position" softkey to display the window for adaptation of the position controller strategy and select "User-defined strategy 209":



CHAN1 Auto servo tuning: Predefined strategy selection AX4:A1

Strategy: Position loop

Custom strategy (209)

DSC active:	✓	
Kv reduction factor:	0.6	
Kv upper limit:	4	1000/min
Feed forward type:	Torque	
FFW always active:	✓	
Kv max method:	Standard	
Impose target equivalence time:	✓	
Target equivalence time:	0.003	s

Speed setpoint filter (1-2):

1	Not used
2	Not used

- Select the "Force equivalent time target value" option.

Note

The "Force equivalent time target value" option is set, when in addition to the speed controller, the position controller is also reoptimized, and despite an equivalent time of the precontrol specified, the path interpolation should not be reoptimized.

To ensure the correct path interpolation, an equivalent time can be specified. This should correspond to the equivalent time of the axes previously optimized on the path.

- Take the equivalent time (MD32800 $MA_EQUIV_CURRCTRL_TIME or MD32810 $MA_EQUIV_SPEEDCTRL_TIME depending on MD32620 $MA_FFW_MODE) from the other axes (here, 3 ms).
- The following parameters must be set in the servo parameter set:
  - Precontrol equivalent time (MD32800 $MA_EQUIV_CURRCTRL_TIME) = 3 ms
  - Position controller gain (MD32200 $MA_POSCTRL_GAIN)

Note

For the automatic operation, the servo parameter set with the appropriate path-optimum values must then be selected in the other axes (here: with equivalent time = 3 ms).

Programming

Definition of global user variables (GUD)

To read out the complete moment of inertia, a new type CHAN REAL variable must be defined in file MGUD.DEF (it is possible that this file must be newly created):

```
; Definition of GUD to read out the total moment of inertia:
DEF CHAN REAL _AST_R_ESTINERTIA
```

Optimization cycle

Note

In the following program example, those cycle calls required for the reoptimization are highlighted **bold**.

Any cycle calls not highlighted are optional.

```
DEF INT myaxiswithnewload=4
; Switch to a safe data set.
; Traverse axis to the safe position / home position.
; Strategy settings were made from the user interface: target value for the
equivalent time!
; Is a destination data set (DDS) available that may be overwritten?
; Open the optimization session.
CYCLE751(1)

; Optimize axis 4: Optimization with a selected strategy based on a new
measurement.
; Add axis 4.
; Accept the optimization strategy from the standard optimization file (XML
file).
; The axis is not considered for an optimization of the path interpolation.
CYCLE752(myaxiswithnewload,3,false)

; Remove the current DDS from the data set list.
CYCLE754(myaxiswithnewload,2,2,-1)

; Add DDS3 to the data set list and overwrite after optimization.
CYCLE754(myaxiswithnewload,1,2,3)

; Axis: Add parameter set 4 to the data set list and overwrite after
optimization.
CYCLE754(myaxiswithnewload,1,1,3)

; Save the original state in a file.
CYCLE755(1,"restorepoint1")

; Start the optimization by activating the results.
; CYCLE751(3)

; Start optimization without activation of the results, if necessary, check
the results first.
CYCLE751(4)

; Save the optimization data in an XML file.
CYCLE757(myaxiswithnewload,"axis_retuned.xml",1,,)

; Export the speed-controlled system.
CYCLE757(myaxiswithnewload,"speedctrlplant.csv",3,,)

; The inertia can be used to limit the acceleration:

; Fetch the inertia.
CYCLE759(myaxiswithnewload,34,,,"_AST_R_ESTINERTIA")

; If required check the optimization results, and then activate.

; Activate result:
; Activate axis-optimum results.
; Save boot files.
CYCLE756(myaxiswithnewload,2,1)

; Close the optimization session.
CYCLE751(2)

; Switch to the newly optimized DDS.
; Has Kv been optimized in the current parameter set?
; The equivalent time of the precontrol was forced by the strategy.
; For path interpolation, all axes must select the correct servo parameter
set.
; Limit the acceleration depending on the new inertia.

M17
```

14.6.4 Example 4: Reoptimization of the path interpolation

For a 5-axis machine, the rotary axis with torque motor (here: 4th axis) must position very accurate and participate in the path interpolation. The changes of the clamping situation and the inertia are completely unknown or so large that even large changes of the equivalent time (also power controller) are expected. The unknown equivalent time means that the path interpolation must be tested and compensated again after the axis optimization.

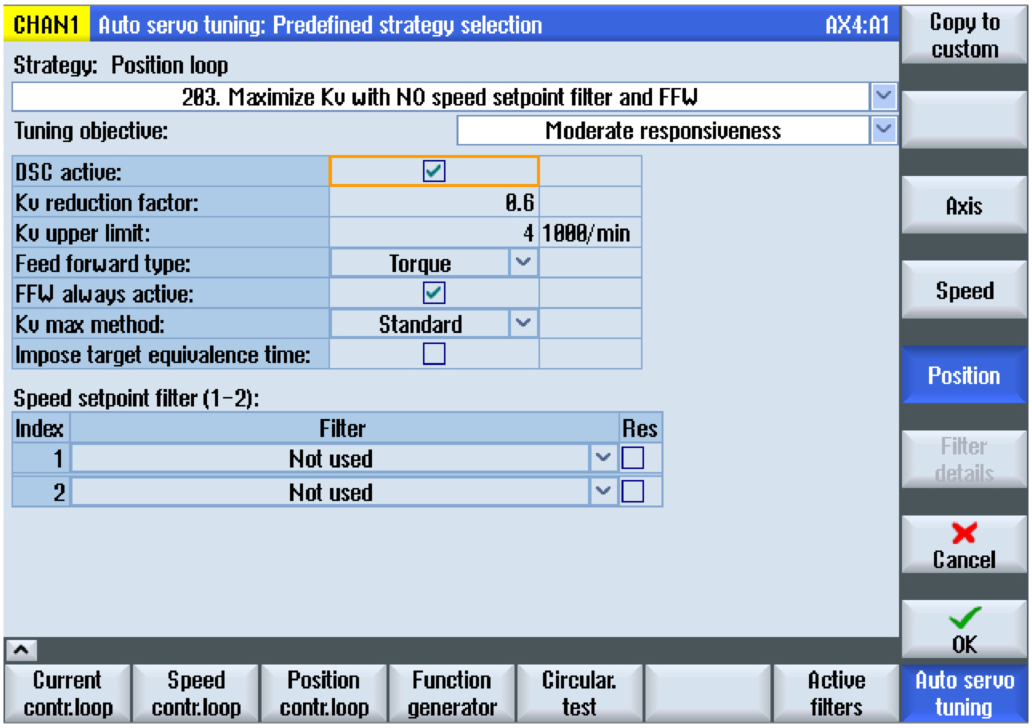
Only the rotary axis (workpiece-dependent) should be reoptimized. The other axes need to be adapted in the precontrol only when the dynamic response of the rotary axis has changed significantly. The rotary axis has several DDS. A safe data set that has a stable setting for clamping the workpieces is selected. It must be considered that the mass inertia to be clamped is not known. Consequently, the maximum inertia must be used for the calculation.

New current setpoint filters must probably be set for the rotary axis. The speed controller must be reoptimized. Because it is not known whether the low-frequency behavior has changed, the position controller must also be reoptimized and the associated mechanical frequency response measured.

The results, such as the complete inertia, can be read with CYCLE759. The newly optimized speed controller is written to another data set. The newly optimized position controller can be written to any desired axis parameter set. The correct path interpolation is guaranteed by all axes involved by being optimized by AST and now, after the reoptimization of the axis, the path interpolation optimization is performed again by AST. The optimization strategy for axis and speed controller, selected from the user interface, is deployed and was stored in the XML file for this axis (*/user/sinumerik/nck/data/optimization/AST_AX4_A1….xml*). The path optimization strategy set from the user interface is deployed and stored in the XML file for the optimization of the path interpolation.

Preparation

- No equivalent time is forced, namely, the predefined strategies for the position controller optimization are deployed:



CHAN1 Auto servo tuning: Predefined strategy selection AX4:A1

Strategy: Position loop

203. Maximize Kv with NO speed setpoint filter and FFW

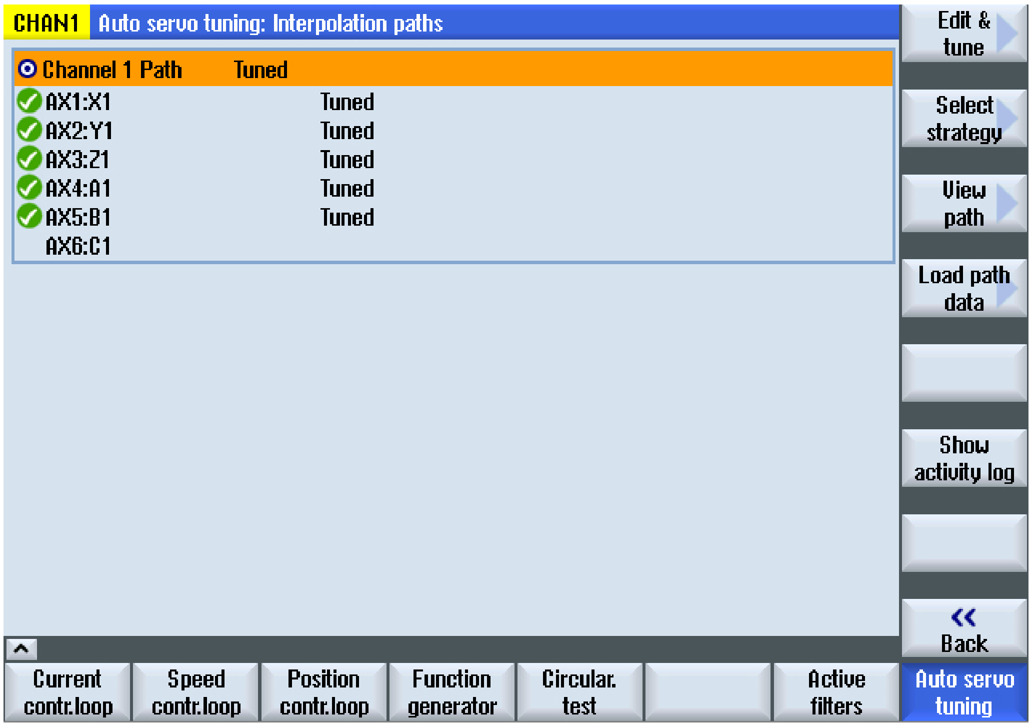
Tuning objective: Moderate responsiveness

DSC active:	✓	
Kv reduction factor:	0.6	
Kv upper limit:	4	1000/min
Feed forward type:	Torque	
FFW always active:	✓	
Kv max method:	Standard	
Impose target equivalence time:	☐	

Speed setpoint filter (1-2):

Index	Filter	Res
1	Not used	☐
2	Not used	☐

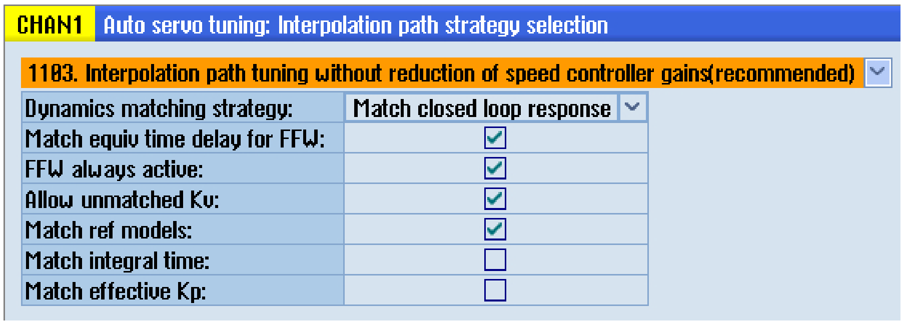
- For the path interpolation, the strategy (all equivalent times identical or use MD32895 $MA_DESVAL_DELAY_TIME) was already chosen for the first optimization.



CHAN1 Auto servo tuning: Interpolation paths

Channel 1 Path	Tuned
AX1:X1	Tuned
AX2:Y1	Tuned
AX3:Z1	Tuned
AX4:A1	Tuned
AX5:B1	Tuned
AX6:C1	

The following selection sets the balancing times (in MD32800 $MA_EQUIV_CURRCTRL_TIME or MD32810 $MA_EQUIV_SPEEDCTRL_TIME) of all involved axes to the largest time constant:


CHAN1 Auto servo tuning: Interpolation path strategy selection

1103. Interpolation path tuning without reduction of speed controller gains(recommended)

Dynamics matching strategy:	Match closed loop response
Match equiv time delay for FFW:	✓
FFW always active:	✓
Allow unmatched Kv:	✓
Match ref models:	✓
Match integral time:	☐
Match effective Kp:	☐

- The following questions must be answered:
  - Which axis parameter set may be reparameterized (for the path interpolation)?
  - May the first axis parameter set be overwritten?
  - Does the spindle need to be part of the path interpolation?
  - If the thread cutting should also run with this axis configuration, parameter set 2 for the axes and the gear stages in the spindle (parameter set 2 ... 6) may need to be optimized.
- **Save the original path data!**

Programming

Definition of global user variables (GUD)

To read out the complete moment of inertia, a new type CHAN REAL variable must be defined in file MGUD.DEF (it is possible that this file must be newly created):

```
; Definition of GUD to read out the total moment of inertia:
DEF CHAN REAL _AST_R_ESTINERTIA
```

Optimization cycle

Note

In the following program example, those cycle calls required for the reoptimization are highlighted **bold**.

Any cycle calls not highlighted are optional.

```
DEF INT myaxiswithnewload=4
; Switch to a safe data set.
; Traverse axis to the safe position / home position.
; Strategy settings were made from the user interface: target value for the
equivalent time!
; Is a destination data set (DDS) available that may be overwritten?
; Each axis was already optimized once with AST (using the screen forms).
; As a consequence, the standard file for CYCLE752 exists for each axis.

; Open the optimization session.
CYCLE751(1)

; Optimize axis 4, with path interpolation.
; Select strategy:
; Add axis 4.
; Accept the optimization strategy from the standard optimization file (XML
file).
; The axis is considered for an optimization of the path interpolation.
CYCLE752(myaxiswithnewload,3,true)

; Measure axis 4 again and reoptimize, save the boot files.
CYCLE753(myaxiswithnewload,3,1)

; Add axis 1 because of the path interpolation.
CYCLE752(1,3,true)
; Axis 1, only post-optimize path interpolation.
CYCLE753(1,1,true)

; Add axis 2 because of the path interpolation.
CYCLE752(2.3,true)
; Axis 2, only post-optimize path interpolation.
CYCLE753(2,1,true)

; Add axis 3 because of the path interpolation.
CYCLE752(3.3,true)
; Axis 3, only post-optimize path interpolation.
CYCLE753(3,1,true)

; Add axis 6 because of the path interpolation.
CYCLE752(6,3,true)
; Axis 6, only post-optimize path interpolation.
CYCLE753(6,1,true)

; Remove the current DDS from the data set list.
CYCLE754(myaxiswithnewload,2,2,-1)
; Add DDS3 to the data set list and overwrite after optimization.
CYCLE754(myaxiswithnewload,1,2,3)

; Axis: Add parameter set 4 to the data set list and overwrite after
optimization.
CYCLE754(myaxiswithnewload,1,1,3)
```

```
; Save the original state in a file.
CYCLE755(1,"restorepoint1")

; Start the optimization by activating the results.
CYCLE751(3)

; Save the optimization data in an XML file.
CYCLE757(myaxiswithnewload,"axis_retuned.xml",1,,)

; Save the measurement data of the speed controlled system to file.
CYCLE757(myaxiswithnewload,"speedctrlplant.csv",3,,)

; Save the result of the path optimization to file.
CYCLE757(1,"path_retuned.xml",2,0,0)

; Fetch the inertia.
CYCLE759(myaxiswithnewload,34,,,"_AST_R_ESTINERTIA")

; Close the optimization session.
CYCLE751(2)

; Switch to the newly optimized DDS.
; Kv has been optimized; switch to the appropriate parameter set.
; The path interpolation has been reoptimized.
; Limit the acceleration in axis 4 depending on the new inertia.

M17
```

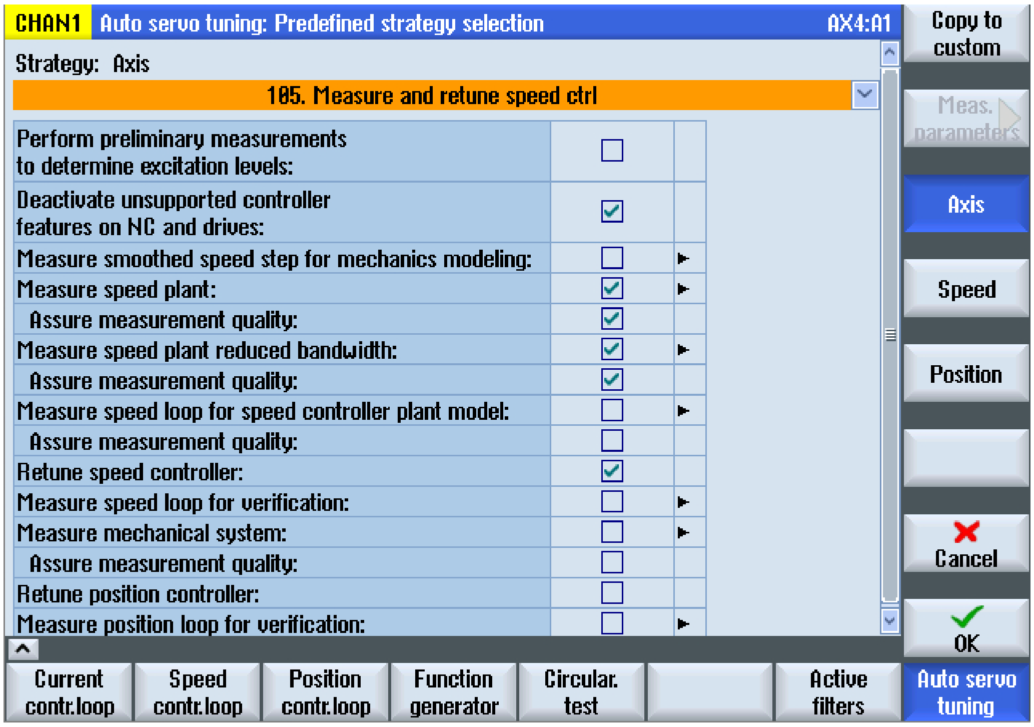
14.6.5 Example 5: Reoptimize the speed control loop to eliminate known periodic disturbance frequencies.

This example shows how the speed controller from the part program is reoptimized without new measurement. The objective of the reoptimization is to minimize vulnerability to known process-related periodic disturbance frequencies. The path interpolation is not reoptimized in this example because the optimization strategy for the position controller provides a dynamic customization by setting an equivalent time.

Preparation

In the simplest case, only a speed controller optimization is necessary:

- Select optimization strategy 105 from the user interface:



CHAN1 Auto servo tuning: Predefined strategy selection AX4:A1

Strategy: Axis

105. Measure and retune speed ctrl

Perform preliminary measurements to determine excitation levels:	☐	
Deactivate unsupported controller features on NC and drives:	✓	
Measure smoothed speed step for mechanics modeling:	☐	►
Measure speed plant:	✓	►
Assure measurement quality:	✓	
Measure speed plant reduced bandwidth:	✓	►
Assure measurement quality:	✓	
Measure speed loop for speed controller plant model:	☐	►
Assure measurement quality:	☐	
Retune speed controller:	✓	
Measure speed loop for verification:	☐	►
Measure mechanical system:	☐	►
Assure measurement quality:	☐	
Retune position controller:	☐	
Measure position loop for verification:	☐	►

Programming

```
DEF INT myAxis=4

; Open the optimization session.
CYCLE751(1)

; Add axis.
; The optimization data is loaded from the standard optimization file.
; The axis is not considered for an optimization of the path interpolation.
CYCLE752(myAxis,3,false,)

; The axis dynamic response is not measured again.
; The axis is reoptimized on the basis of the loaded measurement results
and the currently valid optimization strategy.
CYCLE753(myAxis,2,)

; Activate the use of the gain filter.
CYCLE758(myAxis,208,,,"true" )

; Gain filter: set the optimum frequency to 40 Hz.
```

```
CYCLE758(myAxis,209,,,"40.0" )

; Perform the optimization.
CYCLE751(3)

; Save the axis-specific optimization data for the offline analysis.
CYCLE757(myAxis,"AX1_TEMP_40HZ.XML",1,,)

; For known periodic disturbance frequencies (40 Hz):

; Deactivate the use of the gain filter.
CYCLE758(myAxis,208,,,"false" )

; Perform the optimization.
CYCLE751(3)

; Close the optimization session.
CYCLE751(2)

; For normal broadband disturbances:

M17
```

14.6.6 Example 6: Measuring an axis without optimization

By selecting a strategy, which does not carry out any optimization, then the axis can be automatically measured. This can be used to diagnose mechanical changes. These can involve:

- Load weight
- Component stiffness

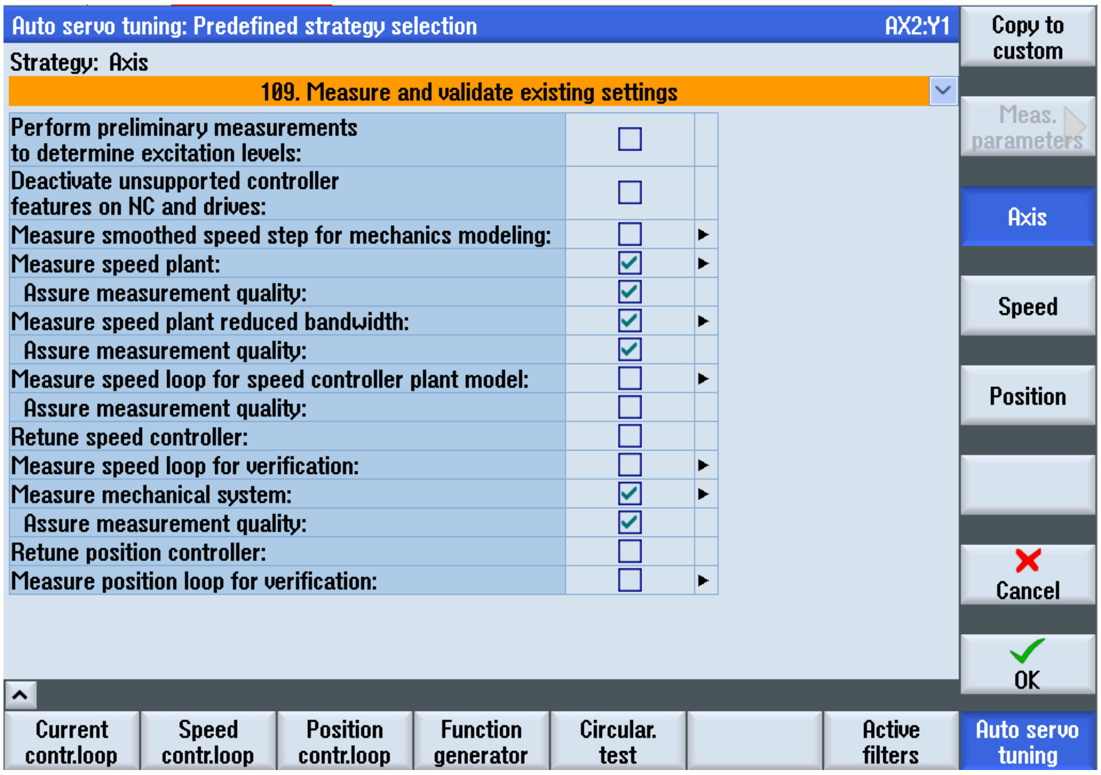
Using the automatic measurement, it can also be checked as to whether the control loop has the same response as when the system was originally delivered, for example. This AST strategy then supplies the actual measurement of the closed loop in the CSV format.

The following data can be automatically measured and exported:

- Speed controlled system
- Closed speed control loop (calculated and/or check measurement)
- Mechanical frequency response (if there are 2 encoders)
- Closed position control loop (calculated and/or check measurement of the position control loop)

Preparation

1. Select optimization strategy 109 from the user interface:



Auto servo tuning: Predefined strategy selection AX2:Y1

Strategy: Axis

109. Measure and validate existing settings

Perform preliminary measurements to determine excitation levels:	☐	
Deactivate unsupported controller features on NC and drives:	☐	
Measure smoothed speed step for mechanics modeling:	☐	▶
Measure speed plant:	✓	▶
Assure measurement quality:	✓	
Measure speed plant reduced bandwidth:	✓	▶
Assure measurement quality:	✓	
Measure speed loop for speed controller plant model:	☐	▶
Assure measurement quality:	☐	
Retune speed controller:	☐	
Measure speed loop for verification:	☐	▶
Measure mechanical system:	✓	▶
Assure measurement quality:	✓	
Retune position controller:	☐	
Measure position loop for verification:	☐	▶

2. Measure and check controller settings.
3. Accept measurement results.
4. Save data under a new name (e.g. RESULT_STRATEGY109_AX1.xml) as user-defined file (so that the standard optimization file is not overwritten).
   The user-defined file is saved as standard to directory */user/sinumerik/hmi/log/optimization*.

Programming

Example of a cycle to measure an axis without optimization:

```
; Release brake
DEF INT AxNr=1

; Open the optimization session.
CYCLE751(1)

; For axis 1, the saved data are used from the user-defined file
RESULT_STRATEGY109_AX1.xml.
CYCLE752(AxNr,2,0,"/user/sinumerik/hmi/log/optimization/
RESULT_STRATEGY109_AX1.xml")
```

```
; Start automatic measurement.
CYCLE751(4)

; Export the speed-controlled system in the CSV format.
CYCLE757(AxNr,"speedctrlplant.csv",3)

; Export the mechanical frequency response in the CSV format.
CYCLE757(AxNr,"mechresponse.csv",4)

; Export the speed control loop (calculated) in the CSV format.
CYCLE757(AxNr,"speedloopcalc.csv",5)

; Export the position control loop (calculated) in the CSV format.
CYCLE757(AxNr,"posloopcalc.csv",6)

; Export the speed control loop (measured) in the CSV format.
CYCLE757(AxNr,"speedloopmeas.csv",7)

; Export the position control loop (measured) in the CSV format.
CYCLE757(AxNr,"posloopmeas.csv",8)

; Close the optimization session.
CYCLE751(2)

M17
```

TE01: Installation and activation of loadable compile cycles

15

Contents

The following sections describe how technology and special functions in the form of individual loadable compile cycle files (*.ELF) are installed and activated in the control.

Technology functions available from Siemens in the form of compile cycles

- **1D/3D clearance control in position control cycle**
  Compile cycle: CCCLC.ELF
  References: Function Manual Special Functions; Clearance Control (TE1)
- **Handling transformation package**
  Compile cycle: CCRCTRA.ELF
  References: Function Manual Special Functions; Handling Transformation Package (TE4)
- **Setpoint exchange**
  Compile cycle: CCSETP.ELF
  References: Function Manual Special Functions; Setpoint Exchange (S9)
- **Axial coupling in the machine coordinate system (MCS coupling)**
  Compile cycle: CCMCSC.ELF
  References: Function Manual Special Functions; MCS Coupling (TE6)
- **Continue machining at the contour (retrace support)**
  Compile cycle: CCRESU
  References: Function Manual Special Functions; Retrace Support (TE7)
- **Fast laser switching signal**
  Compile cycle: CCHSLC.ELF
  References: Function Manual Special Functions; Cycle-Independent Path-Synchronous Signal Output (TE8)
- **Axis pair collision protection**
  Compile cycle: CCPROT.ELF
  References: Function Manual Special Functions; Axis Pair Collision Protection (TE9)

Compile cycle files (*.ELF) and CNC software versions

Compile cycle files (*.ELF) that have been generated for CNC software versions up to SW 4.4 cannot run on systems with CNC software versions as of SW 4.5.

Technology and special functions provided by Siemens in the form of compile cycles

New compile cycle files (*.ELF) for Siemens technology and special functions for CNC software versions as of SW 4.5 can be obtained from your regional Siemens sales office.

Technology and special functions provided by third parties in the form of compile cycles

Compile cycle files (*.ELF) that have been generated for CNC software versions up to SW 4.4 must be adapted and recompiled with the generation environment for CNC software versions as of SW 4.5. Details on the adaptation of the user-specific technology and special functions can be found in the the Open Architecture (OA) documentation.

Compile cycles

Compile cycles are functional expansions of the NC system software that can be created by the machine manufacturer and/or by Siemens and then imported in the control later.

As part of the open NC system architecture, compile cycles have comprehensive access to data and functions of the NC system level via defined software interfaces. In this way, compile cycles significantly extend the functionality of the NC.

Including a compile cycle in the NC system software is performed by loading the compile cycle into the file system of the NC. The compile cycle can be loaded at any time.

Siemens compile cycles

Siemens compile cycles are the technological functions supplied by Siemens.

When you order one of these technological functions, you get only the corresponding software license number. To obtain the compile cycle in the form of a loadable file (".ELF" extension for "**e**xecutable and **l**inking **f**ormat") please contact your regional Siemens sales partner.

Note

Compile cycles created by Siemens are options that require explicit activation and licensing.

References:
Ordering information in Catalog NC 60/61

15.1 Loading compile cycles

15.1.1 Loading a compile cycle with SINUMERIK Operate

Requirement

- The compile cycle to be transferred to the control must be saved on a storage medium which can be directly connected to the control, such as a USB-FlashDrive.
- Access authorization of protection level 1 "Machine manufacturer" (default password "SUNRISE") must be available.
- If there are no cursor keys on the operator panel, a mouse or keyboard must be connected in order to perform the operator actions.

Execution

Perform the following steps to load a compile cycle from a USB-FlashDrive, for example:

1. Insert the USB-FlashDrive in the USB interface on the front of the operator panel.
2. At the user interface, change into the "System data" area:
   Horizontal softkey bar: "Operating area switchover" > "Commissioning" > "System data"
3. Open the USB-FlashDrive: Directory "USB" (keyboard: "Enter" key).
4. Select the compile cycle and copy it:
   Vertical softkey bar: "Copy"
5. Within the "System data" area, switch into the directory "NC data" > "Compile cycles" and insert the compile cycle there:
   Vertical softkey bar: "Insert"
6. Initiate an NC reset to activate the compile cycle:
   Horizontal softkey bar: "Operating area switchover" > "Commissioning" > "Machine data"
   Vertical softkey bar: "Reset (po)"

15.1.2 Loading a compile cycle with HMI Advanced

Precondition

To transfer a compile cycle to the control, the following requirements must be met:

A storage medium (e.g. USB FlashDrive), which stores the compile cycle, is connected to the PCU.

Execution

Perform the following operation to load a compile cycle from a USB FlashDrive to the NC:

1. Insert the USB FlashDrive into the PCU 50/70.
2. Open the USB FlashDrive as the local drive:
   Operating area switchover > Services > Data admin > Local USB
   If the "Local USB" is displayed as disabled, it implies the USB FlashDrive was not detected. Wait until HMI Advanced has detected the USB FlashDrive.
3. Select the compile cycle and copy it:
   vertical "Copy" softkey
4. Go to the OEM directory of the NC card:
   ETC key (">") > "NC card" > Directory: "cc" or "Loadable Compile Cycles"
5. Add the compile cycle to the NC card's OEM directory:
   Vertical "Add" softkey
6. Initiate an NCK reset.

15.1.3 Loading a compile cycle from an external computer with WinSCP3

Precondition

To transfer a compile cycle to the control, the following requirements must be met:

- The external computer (programming device / PC) which the compile cycle is loaded onto is linked to the PCU via a network (TCP / IP).
- The program "WinSCP3" is installed on the external computer.
- The host name or the IP address of the PCU is known.
- The user name and the password to log onto the PCU are known.

Execution

Perform the following operations to load a compile cycle from an external computer into the NC:

1. Start the "WinSCP3" program on the external computer (programming device / PC)
2. Establish a connection to the PCU by selecting an appropriate profile or by entering the host name or IP address, the user name and the password.
3. Copy the compile cycle from the external computer into the following directory on the PCU: /oem/sinumerik/data/cc
   The file name of the compile cycle should have any capital letters in the directory of the PCU.
4. Initiate an NC reset.

15.2 Interface version compatibility

A SINUMERIK-specific interface is used for communication between the compile cycle and NC system software. The interface version used by the loaded compile cycle must be compatible with the interface version of the NC system software.

Interface versions

The relevant interface versions are displayed under:

- Interface version of the NC system software
  HMI Advanced:
  Diagnosis > Service Display > Version > NCU Version
  Display (excerpt)

  CC Interface Version:
  @NCKOPI . . . .@Interfaces=<*1. Position*>.<*2. Position*> . . . .
  Loaded Compile Cycles:
  . . . .

- Interface version of a compile cycle that has not yet been loaded
  HMI Advanced (excerpt):
  Services > <*Medium*> > Softkey: "Properties"
  Display:

  Contents: Loadable compile cycle
  Interface: . . . . @Interfaces=<*1. Position*>.<*2. Position*> . . . .

- Interface version of a loaded compile cycle
  HMI Advanced:
  Diagnosis > Service Display > Version > NCU Version
  Display (excerpt)

  CC Interface Version:
  @NCKOPI . . . .
  Loaded Compile Cycles:
  <*Identifier*> <*Version*> <*Generation Data*>
  CC start address
  _N_<*Identifier*><*Version*>IF<*1st Position*><*2nd Position*>_ELF . . .

  Example:
  _N_CLC407IF003001_ELF corresponds to interface version: 3.1

Dependencies

The following dependencies exist between the interface versions of a compile cycle and the NC system software:

- 1. Position of the interface version number
  The 1st position of the interface version number of a compile cycle and the NC system software must be **the same.**
- 2. Position of the interface version number
  The 2nd position of the interface version number of a compile cycle must be **less than or equal** to the 2nd position of the NC system software.

Note

If alarm 7200 is displayed after powering-up, this means **no** compile cycle has been loaded!

15.3 Software version of a compile cycle

The SW version of a compile cycle is displayed under:

HMI Advanced:

Diagnosis > Service Display > Version > NCU Version

Display (excerpt)

CC Interface Version:

@NCKOPI . . . .

Loaded Compile Cycles:

<*Identifier*> <*Version*> <*Generation Data*>

CC start address

_N_<*Identifier*><**Version**>IF<*1st digit*><*2nd digit*>_ELF . . .

Code=<*Address*> Data=<*Address*> . . .

Example:

_N_CLC**407**IF003001_ELF corresponds to software version 4.7

Note

The display of code and data range start addresses of a compile cycle are provided for diagnostics purposes only and have no significance in normal operation.

15.4 Activating the technological functions in the NC

Requirement

The corresponding option must be enabled before activating a technology function as described below.

If the option data has not been set, the following alarm appears every time the NC powers up - and the technology function will not be activated:

Alarm 7202 "XXX_ELF_option_bit_missing: <*bit number*>"

Function-specific machine data

Each technology function loaded by compile cycle creates a function-specific global NC machine data within the range of numbers from 60900 to 60999 in accordance with the following pattern:

$MN_CC_ACTIVE_IN_CHAN_<*identifier*>[n], with n = 0, 1

Example:

$MN_CC_ACTIVE_IN_CHAN_*MCSC*[0]

$MN_CC_ACTIVE_IN_CHAN_*MCSC*[1]

Activation for the 1st NC channel

The technology functions are activated in the first NC channel via:

$MN_CC_ACTIVE_IN_CHAN_<*identifier*>[0], bit0 = 1

The meaning of machine data bits Bit1 - Bit31 are described in the relevant function description (TE1 - REn).

After the NC powers up the next time, the activated technology functions are integrated into the system software.

CAUTION

SINUMERIK 840D sl

The following alarm is displayed when a bit is set for the first time in the function-specific NC machine data:
$MN_CC_ACTIVE_IN_CHAN_XXXX[0]
:

Alarm 4400 "MD modification causes reorganization of the buffered memory (data loss)"

And you are warned that all user data (part programs, tool data, etc.) will be deleted on the next run-up. If necessary, an archive should be created **after** setting the date and **before** initiating an NC RESET.

15.5 Function-specific startup

Further function-specific installation routines are described in the corresponding function description (TE1 - TEn).

15.6 Creating alarm texts

The alarm texts of the technology functions are stored in the system. If a new alarm is required, then the alarm texts can be supplemented accordingly. The general process depends on the existing interface.

15.6.1 Creating alarm texts with SINUMERIK Operate

The following alarms should be added to the alarm texts of the technology functions:

075999 0 0 "Channel %1 Block %2 Call parameter is invalid"

Proceed as follows

1. Please copy the "oem_alarms_deu.ts" file from the "/siemens/sinumerik/hmi/template/lng" directory to the "/oem/sinumerik/hmi/lng" directory.
2. Rename the file ("xxx_deu.ts").
3. Open the file in the editor and add the new alarm number and the new alarm text in German:
```
<message>
  <source>075999/NCK</source>
  <translation>
    Channel %1 block %2 call parameter is invalid
  </translation>
</message>
```

Note

Each alarm starts with the `<message>` tag and ends with the `</message>` tag. The `<source>` tag contains the alarm number and the source URL. The `<translation>` tag contains the alarm text.

4. To create an alarm text file in a foreign language, copy the just changed file and modify the language code in the filename (e.g., "xxx_eng.ts for English").
5. Open the foreign language alarm text file in the editor and record the translated alarm text in the <translation> tag.
6. Copy the example configuration file "oem_slaesvcadapconf.xml" from directory "/siemens/sinumerik/hmi/template/cfg" into directory "/oem/sinumerik/hmi/cfg".
7. Rename the file to "slaesvcadapconf.xml".
8. Open the "slaesvcadapconf.xml" file in the editor and enter the new base name (filename of the newly created alarm text file without language code and postfix), e.g.:
```
<BaseNames>
  <BaseName_02 type="QString" value="xxx"/>
</BaseNames>
```
9. Restart SINUMERIK Operate.

More information about creating alarm texts with SINUMERIK Operate can be found in:
References:
SINUMERIK 840D sl Base Software and Operating Software Commissioning Manual;
SINUMERIK Operate (IM9) Commissioning Manual, Section: Configuring machine data/alarms

15.6.2 Creating alarm texts with HMI Advanced

The following alarms should be added to the alarm texts of the technology functions:

075999 0 0 "Channel %1 Sentence %2 Call parameter is invalid"

Proceed as follows

1. Copy the "mbdde.ini" file from the "F:\mmc2" directory to the "F:\oem" directory.
2. Add the following two lines in the "mbdde.ini" file:
   [TextFiles]
   UserCZYK=F:\oem\alc_
3. Create the language-dependent "alc_XX.com" text files in the "F:\oem" directory, e.g.:
   "alc_GR.com" for German
   "alc_UK.com" for English
4. Insert the new alarm text in the language-dependent text files, e.g.:
   075999 0 0 "Channel %1 Sentence %2 Call parameter is invalid"
   in the text file in German.
5. Restart HMI Advanced.

For more information about creating alarm texts with HMI Advanced, please refer to:
References:
SINUMERIK 840Di sl/840D sl/840D Base Software and HMI Advanced Commissioning Manual; HMI-Advanced (IM4) Commissioning Manual, Section: Configuring user alarms

Note

HMI reinstallation

Retain the added alarm texts in the text files of the F:\oem even after a reinstallation of HMI.

15.7 Upgrading a compile cycle

To upgrade a compile cycle installed in the control, **under no circumstances** is it sufficient to just exchange the corresponding ELF file. If only the ELF file is exchanged, then this can result in undefined behavior of the NC software due to inconsistent data of the memory and data management.

Execution

Perform the following actions to upgrade a compile cycle:

1. Create an NC archive **without** compile cycles:
   Operating area: "Commissioning" > "ETC" key > "Commissioning archive" > "Creating a commissioning archive" > "OK" >
   – Selection: "NC data" selected
   – Selection: not selected "with compile cycles"!
2. Delete the old compile cycle:
   Operating area: "Commissioning" > "System data" in the directory: NC data> "Compile cycles" > File: <Identifier>.ELF
3. Download the new compile cycle (see Section "Loading compile cycles (Page 708)")

4. Initiate a power on reset with an NC general reset (NC commissioning switch: position "1"). See:
   References:
   Commissioning Manual IBN CNC: NC, PLC, Drive; Section: General tips > Separate NC and PLC general reset > NC general reset
5. Download the NC archive created under Point 1 into the NCU again.
   Operating area: "Commissioning" > "ETC" key > "Commissioning archive" > "Read-in commissioning archive" > "OK"

Note

Version check

To check the version of the newly loaded compile cycle (see Section "Software version of a compile cycle (Page 712)").

Several loaded compile cycles

If several compile cycles are loaded in the control, when upgrading a compile cycle according to the process described above, the other compile cycles remain unchanged.

Newly created data

If the new compile cycle creates new data, which were still not available in the previous version, then after subsequently booting the control, these are preassigned default values.

15.8 Deleting a compile cycle

If a loaded compile cycle is to be completely deleted in the control, it is not enough to only delete the corresponding ELF file. With this procedure, the following data is kept in the retentive memory of the control:

- Activation data: $MN_CC_ACTIVE_IN_CHAN_<identifier>[n]
- Function-specific machine data

Execution

Perform the following actions to delete a compile cycle:

1. Create an NC archive **without** compile cycles:
   Operating area: "Commissioning" > "ETC" key > "Commissioning archive" > "Creating a commissioning archive" > "OK" >
   - Selection: "NC data" selected
   - Selection: not selected "with compile cycles"!
2. Delete the compile cycle:
   :Operating area: "Commissioning" > "System data" in the directory: NC data> "Compile cycles" > File: select "<Identifier>.ELF" > "Delete"

3. Initiate a power on reset with an NC general reset (NC commissioning switch: position "1").
   See:
   References:
   Commissioning Manual IBN CNC: NC, PLC, Drive; Section: General tips > Separate NC and PLC general reset > NC general reset
4. Download the NC archive created under Point 1 into the NCU again.
   Operating area: "Commissioning" > "ETC" key > "Commissioning archive" > "Read-in commissioning archive" > "OK"

Note

Several loaded compile cycles

If several compile cycles are loaded in the control, when deleting a compile cycle according to the process described above, the other compile cycles remain unchanged.

Efficient memory usage

To use the memory resources in the NC as efficiently as possible, you should only load the compile cycles (ELF files) that are actually required into the machine.

15.9 Data lists

15.9.1 Machine data

15.9.1.1 NC-specific machine data

Number	Identifier: $MN_	Description

60900 + i
with:
i = 0, 1, 2, 3, ...
	CC_ACTIV_IN_CHAN_XXXX[n]
	with:
	XXXX = function code
	n = 0 or 1
		n = 0:
		Activating the technology function in NC channels
		n = 1:
		Additional functions within the technology function

TE02: Simulation of Compile Cycles (only HMI Advanced) 16

16.1 Brief description

16.1.1 Function

If, at the SINUMERIK user interface "HMI Advanced" part programs are simulated that use compile cycles, depending on the compile cycle being used, general specific conditions must be established. These are described in the subsequent chapters.

16.2 OEM transformations

When using OEM transformations, the simulation runtime environment has to be set.

Proceed as follows

1. On the computer, on which HMI Advanced is installed, in addition to the directory available as standard "<*HMI Advanced installation path*>/MMC2", create a new directory "<*HMI Advanced installation path*>/**OEM**".
2. In the "OEM" directory, create the file "**DPSIM.TEA**" with the following content:
```
$MN_NC_USER_CODE_CONF_NAME_TAB[196]="TRAORI"
$MN_NC_USER_CODE_CONF_NAME_TAB[197]="_TRAORI"
$MN_NC_USER_CODE_CONF_NAME_TAB[198]="TRACON"
$MN_NC_USER_CODE_CONF_NAME_TAB[199]="_TRACON"
CHANDATA(1)
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]=1
$MC_AXCONF_GEOAX_ASSIGN_TAB[1]=2
$MC_AXCONF_GEOAX_ASSIGN_TAB[2]=3
$MC_TRAFO_RESET_VALUE=0
; Make sure that transformation types 4096 - 4101 are deleted
$MC_TRAFO_TYPE_1=0
$MC_TRAFO_TYPE_2=0
$MC_TRAFO_TYPE_3=0
; Delete transformation chains with OEM transformations
$MC_TRACON_CHAIN_1[0]=0
$MC_TRACON_CHAIN_1[1]=0
; NOTICE! No spaces after M30
M30
```
3. In the "OEM" directory, create the file "**DPSIM.INI**" with the following content:
```
[PRELOAD]
CYCLES=1
CYCLEINTERFACE=0
```

4. Close the HMI application.
5. Launch the HMI application.
6. In the directory for the manufacturer cycles, create the file "**TRAORI.SPF**" with the following content:
   ```
   PROC TRAORI(INT II)
   RET
   ```
7. In the directory for the manufacturer cycles, create the file "**TRACON.SPF**" with the following content:
   ```
   PROC TRACON(INT II)
   RET
   ```

   Note

   The "TRAORI.SPF" and "TRACON.SPF" manufacturer cycles created under 6. and 7. must not be loaded onto the NC.

8. Start the simulation.
9. Run a data comparison for the cycles after the simulation has started up:
   HMI Advanced: **Data comparison > Compare cycles**

   Note

   At least the password for protection level 3 "End user: Service" is needed for the data comparison.

TE1: Clearance control - 840D sl only

17

17.1 Brief description

17.1.1 Brief description

Function

The "clearance control" technological function is used to maintain a one-dimensional (1D) or three-dimensional (3D) clearance required for technological reasons during a defined machining process. The clearance to be maintained may be e.g. the distance of a tool from the workpiece surface to be machined.

Function code

The code for the "clearance control" technological function for function-specific identifiers of program commands, machine data, etc. is:

CLC = Clearance Control

Function restriction

The "clearance control" technological function is only available in the first NC channel, even on controls with more than one NC channel.

This means:

- Only channel axes in the first NC channel may be used as clearance-controlled axes.
- CLC part program commands for the clearance control may only be used in part programs processed in the first NC channel.

Note

The "clearance control" technological function is only available in the **first** NC channel!

Availability

The "clearance control" technological function is only available for SINUMERIK 840D sl.

Compile cycle

The "clearance control" technological function is a compile cycle.

For a description of the system-specific availability and use of compile cycles (see Section "TE01: Installation and activation of loadable compile cycles (Page 707)").

17.1.2 Function description

Laser cutting technology is used as an example for the detailed description of the "clearance control" functionality.

Laser cutting

During laser cutting, a divergent parallel laser beam is directed across a fiber-optic cable or via a mirror to a light-collecting lens mounted on the laser machining head. The collecting lens focuses the laser beam at its focal point. Typical focal lengths are from 5 to 20 cm.

The position of the focal point in relation to the workpiece is an extremely critical process parameter in laser cutting operations and must be kept constant within a tolerance of ≤100 μm.

The distance between the focal point and the workpiece, which is also a key process variable, is usually measured by means of a high-speed capacitive clearance sensor. The analog output voltage of the clearance sensor is approximately proportional to the distance between the sensor and the workpiece surface.

The output voltage of the clearance sensor is transmitted as a digital input value via an analog I/O module to the control where, in the event of deviations from the setpoint clearance, it generates an additional velocity setpoint for the machining head motion axes.

System overview

An overview of the system components required for clearance control in conjunction with SINUMERIK 840D sl is provided in the following diagram.

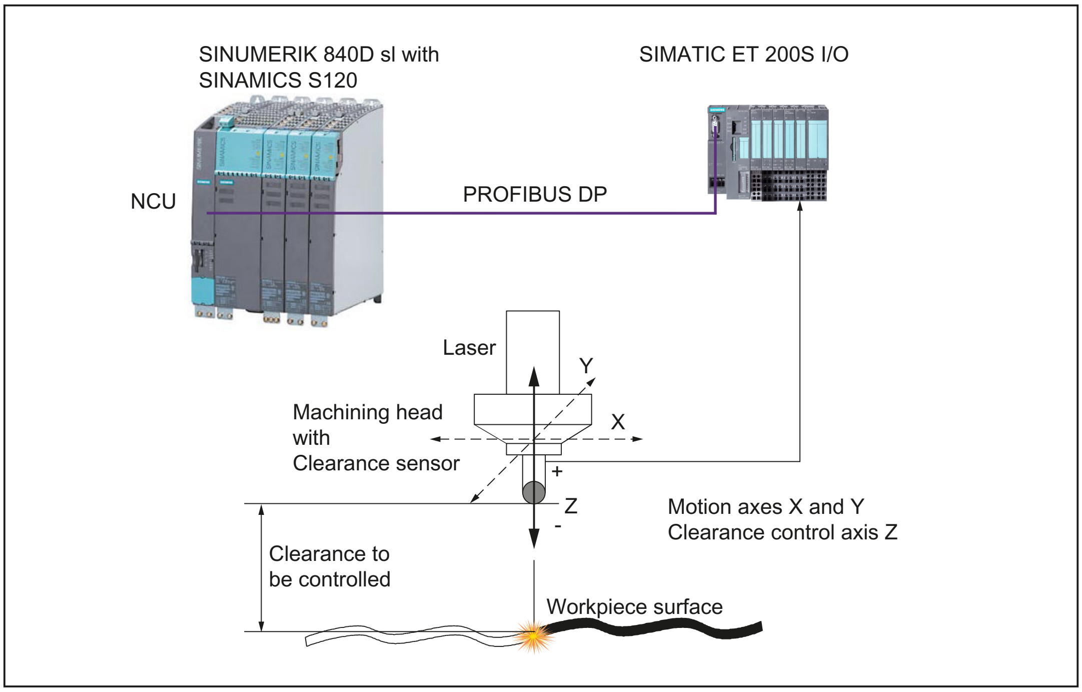


Figure 17-1 System components for clearance control with SINUMERIK 840D sl

1D- / 3D machining

Clearance control can be used for 1D and 3D machining with up to five interpolatory axes.

- 1D machining
  For 1D machining, only one axis is affected by the clearance control. For example, axis Z, as shown in the machine configuration described in the system overview (see previous diagram). Clearance control acts only in the direction of the Z axis.
- 3D machining
  Three linear axes are used to position the tool. One or two rotary axes are used for the orientation of the tool vector (5-axis machining). Up to three linear axes are controlled by the clearance control. Compensation motion can be specified in the directions of the following vectors:
  – Tool orientation vector (normal case)
  – Any programmable vector (compensation vector)

17.2 Clearance control

17.2.1 Control dynamics

Control loop gain K_V

The dynamic response of the closed control loop (sensor - open-loop control - axis) is determined by the maximum closed-loop control gain K_V.

The closed-loop control gain K_V is defined as:

$$K_V = \frac{\text{Velocity [m/min]}}{\text{Following error [mm]}}; \quad \text{in } \left[\frac{\text{[m/min]}}{\text{[mm]}}\right]$$

Clearance control characteristics

Clearance control is based on the two characteristics shown in the following diagram:

- Clearance sensor characteristic (sensor property)
- Clearance control characteristic (can be parameterized via machine data)

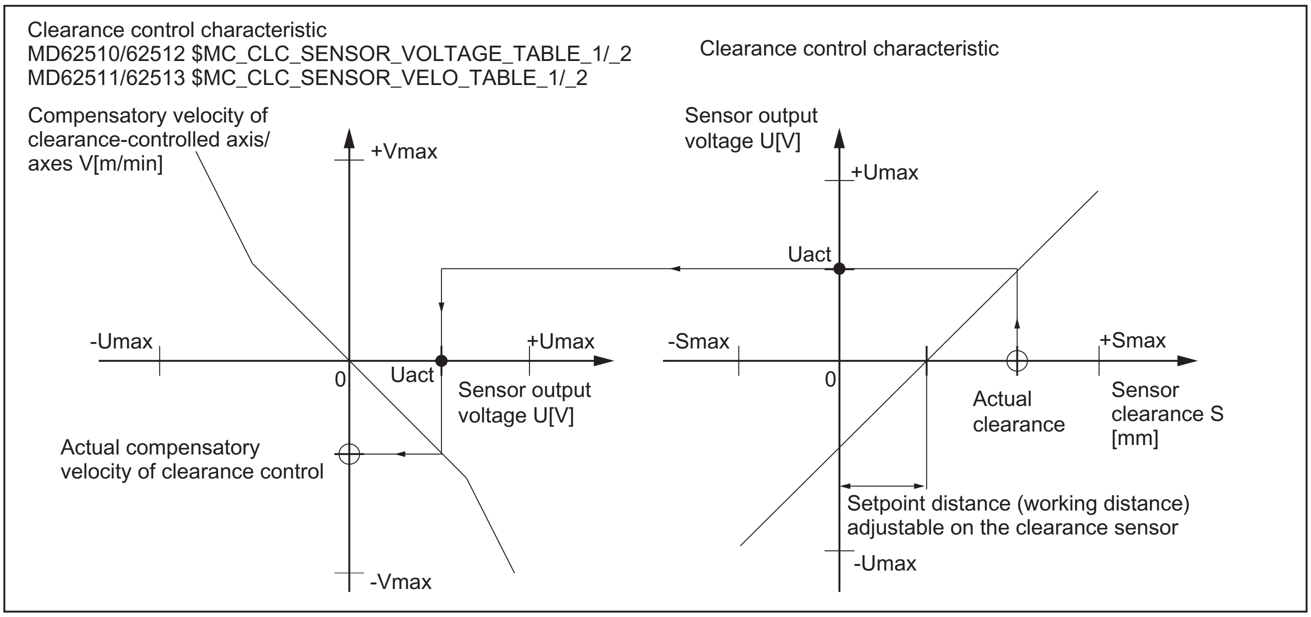

Clearance control characteristic
MD62510/62512 $MC_CLC_SENSOR_VOLTAGE_TABLE_1/_2
MD62511/62513 $MC_CLC_SENSOR_VELO_TABLE_1/_2
Clearance control characteristic
Compensatory velocity of clearance-controlled axis/ axes V[m/min]
+Vmax
-Umax
+Umax
0
Uact
Sensor output voltage U[V]
Actual compensatory velocity of clearance control
-Vmax
Sensor output voltage U[V]
+Umax
Uact
-Smax
+Smax
0
Actual clearance
Sensor clearance S [mm]
Setpoint distance (working distance) adjustable on the clearance sensor
-Umax

Figure 17-2 Correlation between characteristics: Clearance sensor and clearance control

- The clearance sensor measures the actual distance from the workpiece surface and returns as its output variable a voltage in [V], which is almost directly proportional to the distance.
- The clearance control function uses the parameterized voltage/velocity characteristic from the voltage provided by the clearance sensor to calculate a compensatory velocity for the clearance-controlled axes that is appropriate for the clearance.

From the point of view of the control, the unit for the closed-loop control gain is [(mm/min)/V]. In the same way as the setpoint clearance in standardized in [mm], values can only be standardized in [(mm/min)/mm] by using the sensor electronics.

Max. closed-loop control gain

The maximum achievable closed-loop control gain is determined by the following delay and reaction times of the overall system:

1. Reaction time of sensor
2. Delay time of A/D converter
3. Signal processing delay times/deadtimes
4. Reaction time of position controller
5. Reaction times of speed and current controllers
6. Time constants of motor and mechanical components

In practice, only items 3 and 4 are relevant.

The influencing variables together produce an effective time constant. A closed-loop control gain set too high based on this time constant will induce natural oscillations in the range of several hertz in the axis/axes to be controlled.

The objective when starting up the clearance control is to minimize important time constants so that the closed-loop control gain required by the process can be set without inducing natural oscillation of this type.

Dead time

In order to maximize the dynamics of the control response, clearance control takes place on the highest priority position controller level of the NC.

SINUMERIK 840D sl with I/O modules and drives connected via PROFIBUS DP produces a deadtime T_{dead} of:

T_{dead} = 2 * position controller cycle + 2 * speed controller cycle + input lead time T_i

17.2.2 Velocity feedforward control

Eliminating the delay time

The closed-loop control gain, K_V, set for the position controller, corresponds to a delay time Δt. The display time, Δt, is the time which elapses until the actual position of the axis to be controlled reaches the set position at a prescribed velocity of v.

With a delay time:

$$\Delta t = \frac{1}{K_v}$$

and a closed-loop control gain K_V in seconds:

$$[\frac{m/min}{mm}] = [\frac{1000\ mm/60\ s}{mm}] = 16{,}667\ [\frac{1}{s}]$$

for an assumed loop control gain $K_V = 4$, the corresponding delay time Δt is:

$$\Delta t = \frac{1}{4 * 16{,}667\ [\frac{1}{s}]} = 14{,}999\ ms$$

Optimizing the control response

If the control response of the axis is too rigid due to the velocity feedforward control, the control response can be optimized with the following axis-specific NC machine data:

- MD32410 $MA_AX_JERK_TIME (time constant for the axial jerk filter)
- MD32610 $MA_VELO_FFW_WEIGHT (feedforward control factor for speed)

The speed filters of the SINAMICS S120 drive provide additional damping capabilities:

- Parameter 1414 ff. (time constant, speed setpoint filter 1, 2)

Note

Every damping measure implemented contributes to increasing the overall time constant of the control loop!

Reference:
The complete description of the velocity feedforward control is described in:
Function Manual Expansion Functions; K3: Compensation, Section "Dynamic feedforward control (following error compensation)"

17.2.3 Control loop structure

The figures below provide an overview of how the clearance control function is embedded in the control loop structure of the NC position controller and the internal structure of the function.

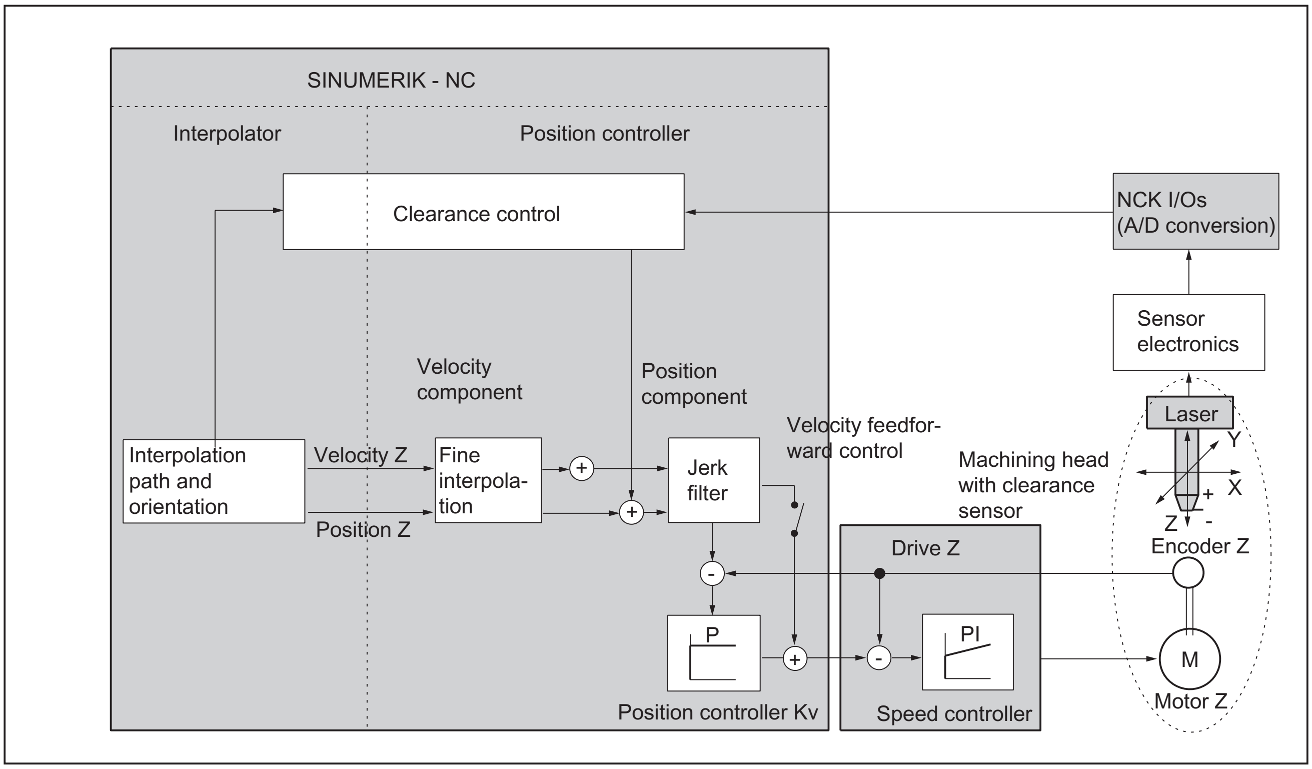


Figure 17-3 Control structure, position controller with clearance control (principle)

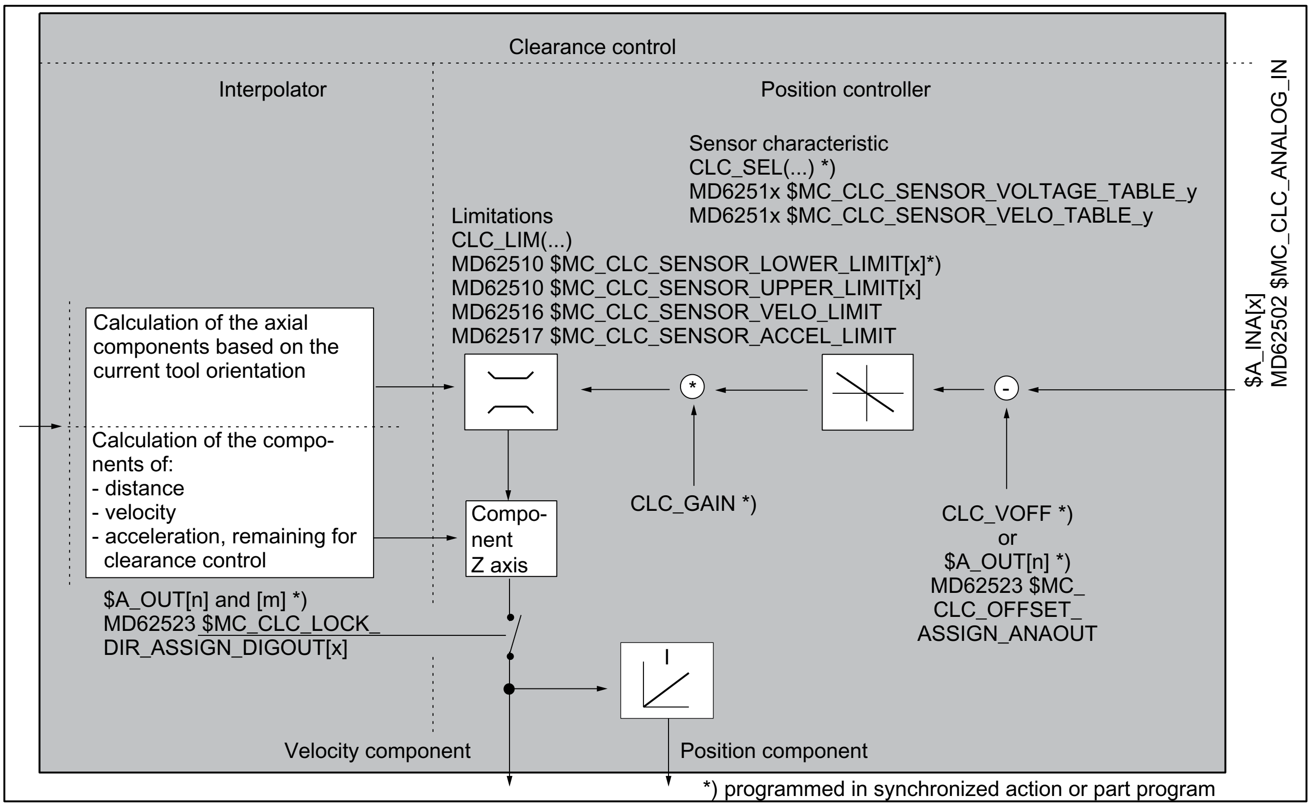

Clearance control
Interpolator
Position controller
Sensor characteristic
CLC_SEL(...) *)
MD6251x $MC_CLC_SENSOR_VOLTAGE_TABLE_y
MD6251x $MC_CLC_SENSOR_VELO_TABLE_y
Limitations
CLC_LIM(...)
MD62510 $MC_CLC_SENSOR_LOWER_LIMIT[x]*)
MD62510 $MC_CLC_SENSOR_UPPER_LIMIT[x]
MD62516 $MC_CLC_SENSOR_VELO_LIMIT
MD62517 $MC_CLC_SENSOR_ACCEL_LIMIT
$A_INA[x]
MD62502 $MC_CLC_ANALOG_IN
Calculation of the axial components based on the current tool orientation
Calculation of the components of:
- distance
- velocity
- acceleration, remaining for clearance control
Component Z axis
CLC_GAIN *)
CLC_VOFF *) or $A_OUT[n] *) MD62523 $MC_CLC_OFFSET_ASSIGN_ANAOUT
$A_OUT[n] and [m] *)
MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[x]
Velocity component
Position component
*) programmed in synchronized action or part program

Figure 17-4 Control structure, clearance control (principle)

17.2.4 Compensation vector

Standard compensation vector

The compensation vector of the clearance control and the tool orientation vector are normally identical. Consequently, the compensation movement of the clearance control is normally always in the direction of the tool orientation.

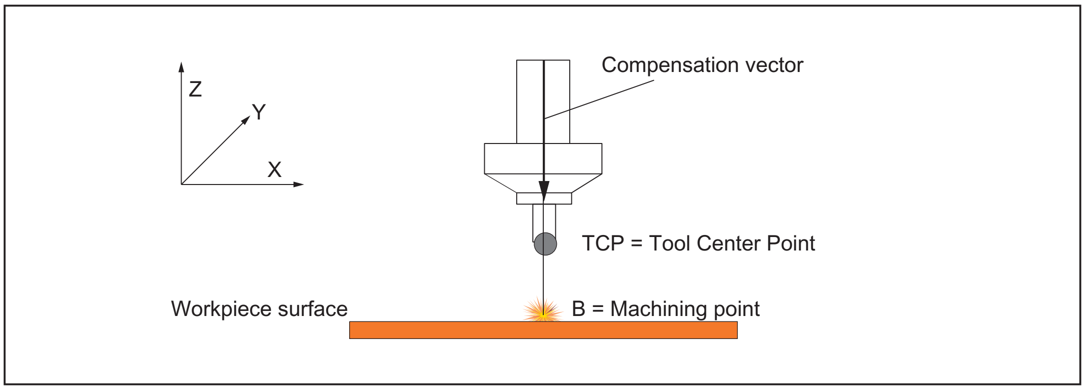


Figure 17-5 Clearance control with standard compensation vector

Note

In all the figures in this section, the traversing movement of the machining head needed in order to machine the workpiece is in the direction of the Y coordinate, i.e. perpendicular to the drawing plane.

As long as the tool orientation, and hence the compensation vector, is perpendicular to the workpiece surface, no disadvantage for the machining process results from the compensation movements of the clearance control.

If a tool setting angle is needed for technological reasons, with the result that the tool orientation is no longer perpendicular to the workpiece surface, the machining point on the workpiece surface is shifted during compensation movements of the clearance control along the standard compensation vector.

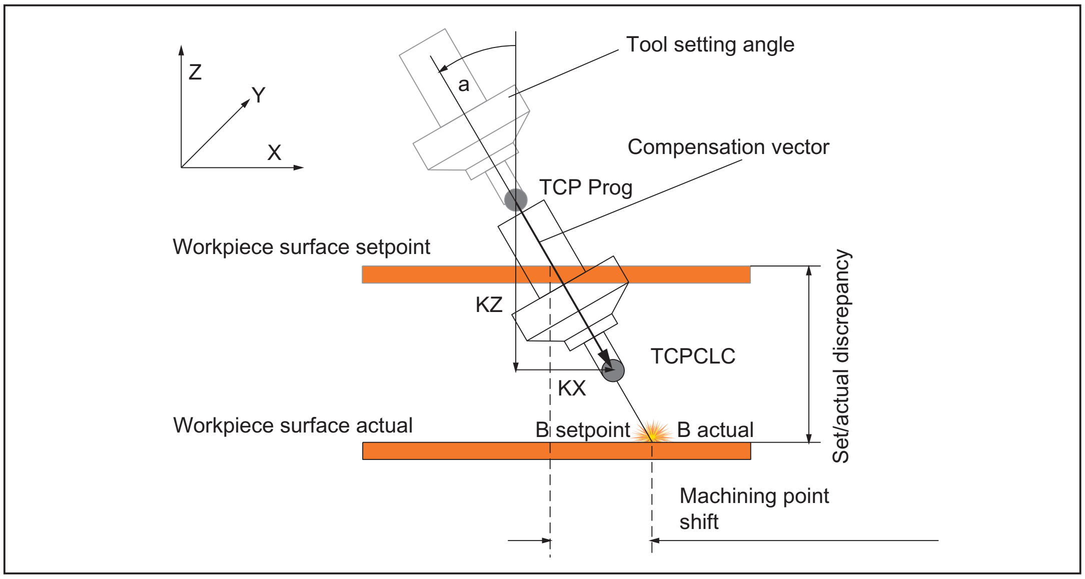


Figure 17-6 Standard compensation vector

The reason for the shift in the machining point is the X component (K_X) of the compensation vector parallel to the workpiece surface. The TCP of the tool, and thus the machining point B, is shifted by this amount.

Programmable compensation vector

When using the programmable compensation vector, the compensation movements of the clearance control are in the direction of the programmed vector, and not in the direction of the tool orientation.

The X component specified above (K_X) is omitted because the programmable compensation vector is defined perpendicular to the workpiece surface. This does not cause the machining point (B) to be shifted as a result of the compensation movement of the clearance control.

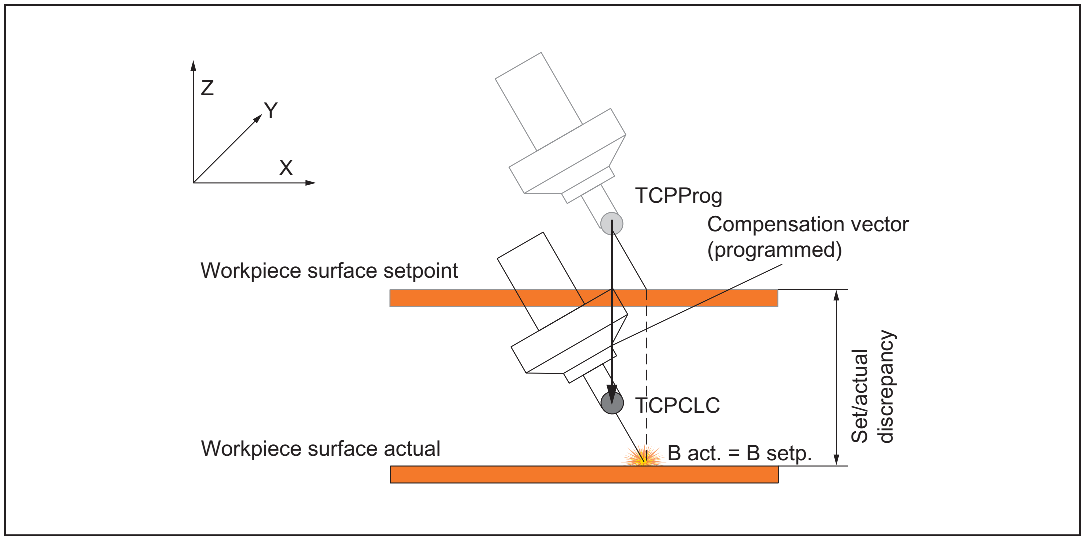


Figure 17-7 Programmable compensation vector

Changes in orientation

Based on the above observations, a different behavior also results when the orientation of the machining head is changed while the clearance control is active.

In the following diagram the normal case is shown on the left (compensation vector == tool orientation vector); and the case with the programmed compensation vector is shown on the right.

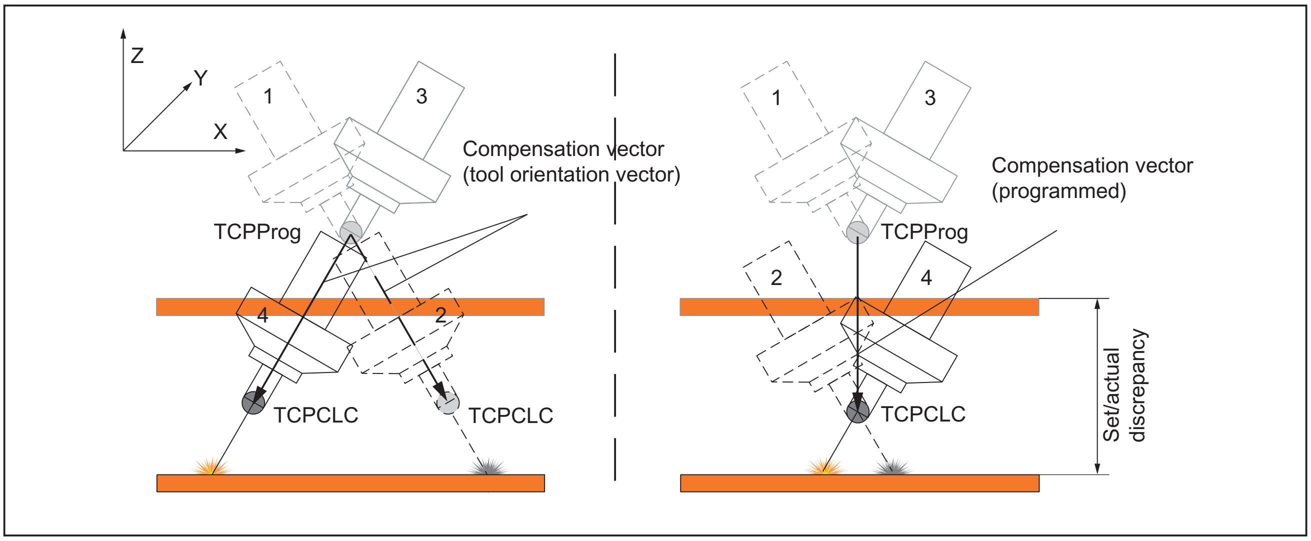

Z
Y
X
1
3
Compensation vector (tool orientation vector)
TCPProg
4
2
TCPCLC
TCPCLC
1
3
Compensation vector (programmed)
TCPProg
2
4
TCPCLC
Set/actual discrepancy

Figure 17-8 Change in orientation of the machining head

The meaning of the individual positions of the machining head is as follows:

1. Programmed position of the machining head
2. Actual position of the machining head with clearance control active before the orientation change

3. Programmed position of the machining head after the change in orientation
4. Actual position of the machining head with clearance control active after the orientation change

The machining head movement visible on the machine when the change in orientation takes place is direct from position 2 to position 4.

17.3 Technological features of clearance control

Clearance control is characterized by the following technological features:

- **Dynamic Response**
  The overlaid sensor motion uses the current residual dynamic response that is still in reserve after the programmed axis motion (velocity and acceleration). The proportion of residual acceleration that must be used can be set as a percentage in a machine data.
- **Sensor characteristic**
  The gain characteristic of a sensor can be defined with up to 10 interpolation points.
- **Sensors**
  Two sensors with different gain characteristics (e.g. a mechanical and a capacitive sensor) can be used simultaneously. The active sensor characteristic can be switched over block-synchronously by means of a language command in the part program.
- **Closed-loop control gain of clearance control**
  The closed-loop control gain configured in the NC machine data for clearance control can be changed block-synchronously by means of a language command in the part program.
- **Motion limitation**
  The lower and upper limits configured in the NC machine data for the axis movements induced by clearance control can be changed block-synchronously by means of a language command in the part program.
  An alarm appears when a limit is reached. The alarm response (stop all traversing movements or display only) can be parameterized. The current position offset can be frozen by means of a PLC signal.
- **Response on deactivation**
  The deactivation response of the clearance control function can be programmed either for synchronization with the current axis positions (no compensating movement) or for compensating axis movements to the last programmed axis positions (axis positions without clearance control).
- **Programmable clearance setpoint**
  An additional voltage value can be programmed in order to alter the setpoint distance set in the sensor electronics on a block-related basis.

- **Control options via the PLC interface**
  The following signals are available at the PLC interface:
  Status signals:
  - Closed-loop control active
  - Overlaying movement at standstill
  - Lower limit reached
  - Upper limit reached.

  Control signals:
  - Path override for sensor movement active
- **Status data of clearance control**
  Both the current values and the min/max values of the sensor signal and of the position offset are available as GUD and/or OPI variables.
- **Sensor signal**
  The sensor signal can be smoothed via a PT1 filter with adjustable time constant.

17.4 Sensor collision monitoring

Sensor signal

If the clearance sensor used has an additional "sensor collision" signal for detecting a collision between the sensor and the workpiece being machined, this signal can be made available to the clearance control function via a digital NC peripheral input.

As response to this signal, the clearance control executes retraction motion in all axes with clearance control. The retraction motion is executed independently of the current value of the velocity override with maximum traversing velocity in a positive control direction until the currently valid upper limit of the control range is reached. Path motion is simultaneously stopped.

After all traversing motion has been stopped, part program machining can be continued with NC START.

Parameter assignment

The digital peripheral input, which the "sensor collision" signal is wired to, is assigned to the clearance control function using the following machine data:

MD62504 $MC_CLC_SENSOR_TOUCHED_INPUT (digital peripheral input for "sensor collision" signal)

The digital peripheral input is specified by entering the input number in the same way as $A_IN/ $A_OUT digital I/O peripheral system variables are specified ($A_IN[*input number*]).

If a negative input number is entered, the "sensor collision" signal will be processed with internal inversion by the clearance control function (fail safe method).

17.5 Startup

Compile cycle

Before commissioning the technological function, ensure that the corresponding compile cycle has been loaded and activated (see Section "TE01: Installation and activation of loadable compile cycles (Page 707)").

17.5.1 Activating the technological function

The technological function is activated via the machine data:

MD60940 $MN_CC_ACTIVE_IN_CHAN_CLC[0], bit n = 1

n = channel number - 1; bit0 = channel 1, bit1 = channel 2, etc.

Note

The technological function can be activated for several channels simultaneously.

17.5.2 Configuring the memory

The technological function requires **additional** data in the NC-internal block memory. The values must be increased for the following memory configuring channel-specific machine data:

- MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS += **4** (number of block elements for compile cycles)
- MD28100 $MN_MM_NUM_CC_BLOCK_USER_MEM += **20** (size of block memory for compile cycles (DRAM) in KB)

17.5.3 Parameterizing the input signals

The following input signals must be parameterized in the machine data:

- Clearance sensor input voltage
  - 1 analog input
- "Sensor collision" input signal (optional)
  - 1 digital input

Analog input

The following machine data must be parameterized for the analog input:

- MD10300 $MN_FASTIO_ANA_NUM_INPUTS (number of active analog NC inputs)
- MD10362 $MN_HW_ASSIGN_ANA_FASTIN (per analog module) (hardware assignment for the fast analog NC inputs)
  Specifying the physical address activates the analog input module

Digital input

The following machine data must be parameterized for the digital input:

- MD10350 $MN_FASTIO_DIG_NUM_INPUTS (number of active digital NC input bytes)
- MD10366 $MN_HW_ASSIGN_DIG_FASTIN (per digital module) (hardware assignment for external digital NC inputs)
  Specifying the physical address activates the digital input module.

A complete description of the analog and digital inputs appears in:

References:

Function Manual, Extended Functions, Digital and Analog NC I/O (A4)

17.5.4 Parameters of the programmable compensation vector

Reference coordinate system

The programmable compensation vector specifies the direction in which the compensation movement of the clearance control takes place. The compensation vector always refers to the basic coordinate system (machine coordinate system).

The start coordinates [Xa, Ya, Za] of the compensation vector coincide with the origin of the basic coordinate system and are thus always [0, 0, 0].

The end coordinates [Xe, Ye, Ze] of the compensation vector are determined by the actual positions of 3 channel axes, known as the direction axes.

Direction axes

The direction axes must meet the following conditions:

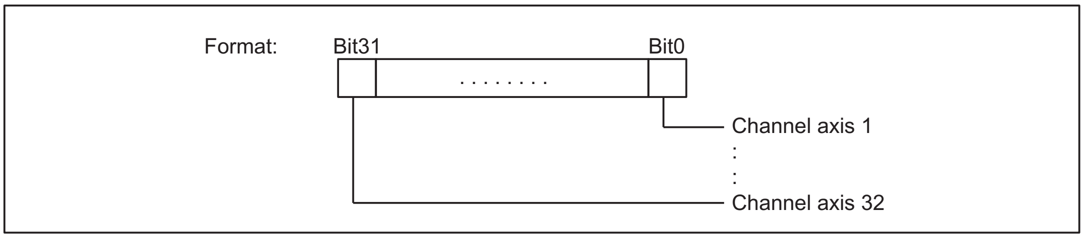
1. The direction axes must be channel axes of the channel in which the clearance control is activated.
2. The direction axes must be linear axes.
   Note:
   Since the direction axes are only used to interpolate the direction components, they do not need mechanical axes and can therefore be configured as simulation axes.
3. [mm] or [inch] must be selected as the unit of measurement for the direction axes.

4. The direction axes may not participate in an axis coupling, e.g. transformation, electronic gear, etc.
5. To ensure that the dynamic response of the path is not limited by the dynamic response of the direction axes, the following machine data for the direction axes must be set equal to or more than the corresponding values of the geometry axes of the channel:
   – MD32000 $MA_MAX_AX_VELO[x] (maximum axle velocity)
   – MD32200 $MA_POSCTRL_GAIN[x] (Kv factor)
   – MD32230 $MA_MAX_AX_ACCEL[x] (position controller structure configuration)
     x = axle number

The following machine data is used to specify which channel axis is the direction axis:

- MD62528 $MC_CLC_PROG_ORI_AX_MASK (progr. orientation vector: axis mask)

Each machine data bit corresponds to a channel axis.



- Coordinate X = channel axis corresponding to bit a
- Coordinate Y = channel axis corresponding to bit b
- Coordinate Z = channel axis corresponding to bit c
  with a < b < c

Current difference angle

The difference angle is the angle between the tool orientation vector and the compensation vector. If the current difference angle of the clearance control is to be output in a system variable $AC_PARAM[n], index n of the system variable should be entered in the following machine data:

MD65530 $MC_CLC_PROG_ORI_ANGLE_AC_PARAM ()

Permissible limit angle

The permissible limit angle specifies the maximum difference angle allowed between the tool orientation vector and the compensation vector. The limit angle is configured via the following machine data:

MD65520 $MC_CLC_PROG_ORI_MAX_ANGLE ()

17.5.5 Parameter settings for clearance control

The part program name

The following machine data must be parameterized for the declaration of the function-specific part program name CLC_GAIN and CLC_VOFF:

- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[0] = "OMA1" (list of re-configured NC codes)
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[1] = "**CLC_GAIN**"
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[2] = "OMA2"
- MD10712 $MN_NC_USER_CODE_CONF_NAME_TAB[3] = "**CLC_VOFF**"

1D/ 3D clearance control

The following machine data is used to select 1D or 3D clearance control:

- MD62500 $MC_CLC_AXNO = <n> (axis assignment for clearance control)
  - <n> > 0: 1D clearance control where <n> = axis number of the clearance-controlled channel axis
  - <n> = -1: 1. 5-axis transformation configured in channel
  - <n> = -2: 2. 5-axis transformation configured in channel

Input signals

The clearance sensor input signals parameterized above are made known to the clearance control function with the following machine data (see also Section "Parameterizing the input signals (Page 733)"):

- MD62502 $MN_CLC_ANALOG_IN = <n> (analog input for the clearance control)
  <n= input number, analog to the addressing of the system variables $A_INA[<n>]
- MD62504 $MN_CLC_SENSOR_TOUCHED_INPUT = <n> (assignment of an input bit for the "sensor collision" signal)
  <n= input number, analog to the addressing of the system variables $A_IN[<n>]

Exact stop

In order to be able to meet a programmed "Exact stop coarse/fine reached" block change condition (G601/G602), the traversing velocity induced by the clearance control function in the clearance-controlled axes must be lower than the standstill velocity tolerance at least for the duration of the standstill delay time.

The following machine data must be modified to optimize the block change time:

- MD36000 $MA_STOP_LIMIT_COARSE[<x>] (exact stop coarse)
- MD36010 $MA_STOP_LIMIT_FINE[<x>] (exact stop fine)
- MD36020 $MA_POSITIONING_TIME[<x>] (delay time exact stop fine)

- MD36040 $MA_STANDSTILL_DELAY_TIME[<x>] (standstill monitoring time delay)
- MD36060 $MA_STANDSTILL_VELO_TOL[<x>] ("axis/ spindle stopped" velocity/speed threshold)
  <x> = axis number of clearance-controlled machine axis

17.5.6 Starting up clearance control

Clearance sensor

The clearance sensor outputs should be connected to the I/O modules that were activated using the following machine data:

- MD10362 $MN_HW_ASSIGN_ANA_FASTIN (I/O address of the I/O module) (hardware assignment of fast analog NC inputs)
- MD10366 $MN_HW_ASSIGN_DIG_FASTIN (I/O address of the I/O module) (hardware assignment of external digital NC inputs)

(See also Section " "Supplementary conditions > I/O modules (Page 756)").

Test control direction

Proceed as follows to test the control direction of the clearance control function:

- Activating the clearance control function using the part program with CLC(1) (see Chapter "Activating and deactivating clearance control (CLC) (Page 739)")
- Generate an input voltage, e.g. via the following synchronized action:

Program code
```
N100 $AC_TIMER[1]=2.5
N110 ID = 1 EVERY $AC_TIMER[1] >= 2.5 DO $AC__TIMER[1]=0
N120 ID = 2 WHENEVER $AC_TIMER[1] < 2.0 DO $A_OUTA[6] = 100000.0 *
($AC_TIMER[1] - 1.0)
N130 ID = 3 WHENEVER $AC_TIMER[1] >= 2.0 DO $A_OUTA[6] = 0.0
```

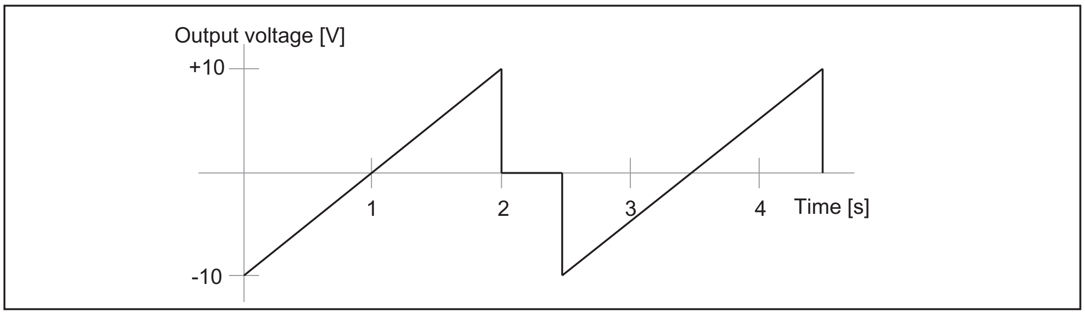


Figure 17-9 Output voltage for synchronized action

The voltage specification for the analog output $A_OUTA[6] used in the synchronized action is subtracted from the clearance sensor input voltage by the clearance control function and therefore has the opposite polarity to the input signal.

Set the following machine data to induce the clearance control function to use analog output 6 ($A_OUTA[6]) as an additional input overlaid on the sensor input:

MD62522 $MN_CLC_OFFSET_ASSIGN_ANAOUT = 6 (hardware assignment of external digital NC inputs)

Note

Before the clearance control function is activated for the first time, check that the entire working range enabled for clearance control is collision-free:

- MD62505 $MC_CLC_SENSOR_LOWER_LIMIT (lower clearance control motion limit)
- MD62506 $MC_CLC_SENSOR_UPPER_LIMIT (upper clearance control motion limit)

An incorrect control direction can be corrected using one of the following methods:

- Reversing the polarity of the analog input
- Changing the sign of all values in the following machine data:
  - MD62511 $MC_CLC_SENSOR_VELO_TABLE_1 (coordinate velocity of interpolation points sensor characteristic 1)
  - MD62513 $MC_CLC_SENSOR_VELO_TABLE_2 (coordinate velocity of interpolation points sensor characteristic 2)

Sensor signal

Signal quality

The quality of the analog input signal can be checked using the function-specific display data (see Chapter "Function-specific display data (Page 752)").

Input voltage range

The input voltage range for the measuring signal of the sensor can be adapted using the machine data for the weighting factor for the analog NC inputs:

MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[<analog input>] = <weighting factor>

The following machine data must be set in order that the weighting factor from the clearance control function is included in the calculation:

MD62508 $MC_CLC_SPECIAL_FEATURE_MASK, Bit 13 = 1

Function-specific alarm texts

To display function-specific alarm texts, these must first be incorporated in the appropriate HMI data management (see Chapter "Creating alarm texts (Page 713)").

Completion

A data backup is recommended once the start-up procedure has been completed.

References:

Commissioning Manual IBN CNC: NC, PLC, Drive

Note

A data backup is recommended once the start-up procedure has been completed.

17.6 Programming

17.6.1 Activating and deactivating clearance control (CLC)

Syntax

CLC(*Mode*)

Mode

- Format: Integer
- Value range: -1, 0, 1, 2, 3

CLC(...) is a procedure call and must therefore be programmed in a dedicated part program block.

Functionality

The following modes are available for activating/ deactivating clearance control:

- CLC(1)
  Activation of the clearance control with compensation vector in the direction of the tool orientation
  The evaluation of the sensor collision signal is deactivated.
- CLC(2)
  Activation of the clearance control with compensation vector in the direction of the tool orientation
  The evaluation of the sensor collision signal is activated.
- CLC(3)
  Activation of the clearance control with programmed compensation vector
  The evaluation of the sensor collision signal is deactivated.

- CLC(0)
  Deactivation of clearance control without canceling the position offset.
  If the clearance-controlled axes are still moving at the instant of deactivation due to the sensor signal, they are stopped. The workpiece coordinate system (WCS) is then synchronized with the corresponding standstill positions. An automatic preprocessing stop is executed.
- CLC(-1)
  Deactivation of clearance control with cancellation of the position offset
  If the clearance-controlled axes are still moving at the instant of deactivation due to the sensor signal, they are stopped. A position offset to the last programmed position is canceled automatically with the deactivation command.

RESET behavior

CLC(0) is executed implicitly on a reset (NC RESET or end of program).

Parameterizable RESET behavior

The reset response of a 1D clearance control function can be determined via the channel-specific NC OEM machine data:

- MD62524 $MC_CLC_ACTIVE_AFTER_RESET (reset response with active CLC)

⚠ CAUTION

Clearance control

The channel-specific NC-OEM machine data MD62524 is only effective in conjunction with a 1D clearance control function.

For 3D clearance control, CLC(0) is always effective in the event of a RESET.

The following response can be parameterized:

- MD62524 $MC_CLC_ACTIVE_AFTER_RESET = 0
  In the event of a reset, the clearance control function responds as it does to deactivation with CLC(0) (see "Functionality" section).
- MD62524 $MC_CLC_ACTIVE_AFTER_RESET = 1
  The current state of the clearance control function is retained.

Boundary conditions

Please note the following supplementary conditions:

Continuous-path mode

Activating/ deactivating clearance control (CLC(*mode*)) during active continuous-path mode (G64/G64x) will induce a drop in velocity for path motions. To avoid voltage drops of this type, clearance control must be activated before a path section with constant path velocity. During the corresponding path section, if necessary, clearance control can be blocked and then re-enabled via the programmed gain factor for clearance control (CLC_GAIN).

Block change with exact stop

If exact stop is active at the end of the block (G60/G09 with G601/G602) the block change may be delayed due to axis movements induced by the clearance control sensor signal.

Sensor collision monitoring

A digital input for an additional collision signal can be configured by the sensor using the following machine data:

MD62504 $MC_CLC_SENSOR_TOUCHED_INPUT (assignment of the input signal for the "sensor collision" signal)

This collision monitor can be activated and deactivated block-synchronously through alternate programming of CLC(1)/CLC(2).

As a reaction to the sensor collision signal, the clearance control moves, irrespective of the feedrate override setting, at maximum velocity in the plus direction until it reaches the currently valid upper limit. The path motion is stopped simultaneously.

NC-START can be used to resume processing.

3D clearance control and 5-axis transformation

If 3D clearance control is activated before the 5-axis transformation required for clearance control in the direction of the tool orientation has been activated, the clearance control function will be dependent on the active working plane (G17/G18/G19):

- `G17`: Direction of clearance control = Z
- `G18`: Direction of clearance control = Y
- G19: Direction of clearance control = X

Activation of 5-axis transformation

When 5-axis transformation is activated, the tool orientation specified by means of the rotary axis positions must tally with the control direction specified by the active working plane on activation of clearance control.

If the tool orientation of the 5-axis transformation and the control direction of the clearance control function do not tally, the following CLC alarm will appear:

- Alarm "75016 Channel *number* Block *number* CLC: Orientation changed with TRAFOOF"

Deactivation of 5-axis transformation

If 5-axis transformation is deactivated when clearance control is active, the last control direction before 5-axis transformation was deactivated is retained.

Tool radius compensation

3D clearance control can only be deactivated if no tool radius compensation is active in the channel at the time of deactivation (`G40`). If tool radius compensation is active (`G41`/`G42`), the following alarm appears:

- Alarm "75015 Channel *number* Block *number* CLC(0) with active TRC."

Compensation vector

Actual position of the direction axes

If the clearance control is activated with a programmable compensation vector at a position of 0 on all 3 direction axes, a compensation vector cannot be calculated from this information. The following alarm is then displayed:

- Alarm "75019 Channel *number*, error ID: 1, angle 0.0"

Referencing of the direction axes

The direction axes must be referenced before clearance control is activated with programmable compensation vector CLC(3).

Interface signals of the direction axes

The following interface signals must be set for all 3 direction axes by the PLC user program, before clearance control is activated with programmable compensation vector CLC(3).

- DBX31, … DBX1.5 = 1 (position measuring system 1)
- DBX31, … DBX2.1 = 1 (servo enable)
- DBX31, … DBX21.7 = 1 (pulse enable)
  x = axle number

Switchover of clearance control

Direct switchover of clearance control from CLC(1) or CLC(2) to CLC(3) or vice-versa is not possible. Such switchovers are ignored without a checkback message. If a switchover is necessary, the clearance control must first be deactivated with CLC(0) or CLC(-1) and then activated in the desired mode.

Interpolation of the compensation vector

If the compensation vector is required to follow a non-linear workpiece surface, such as an arc, with respect to its orientation, this can be achieved by programming the direction axes.

Example

Orientation of the compensation vector perpendicular to a semi-circular workpiece surface. The programming of the traversing movement is not considered.

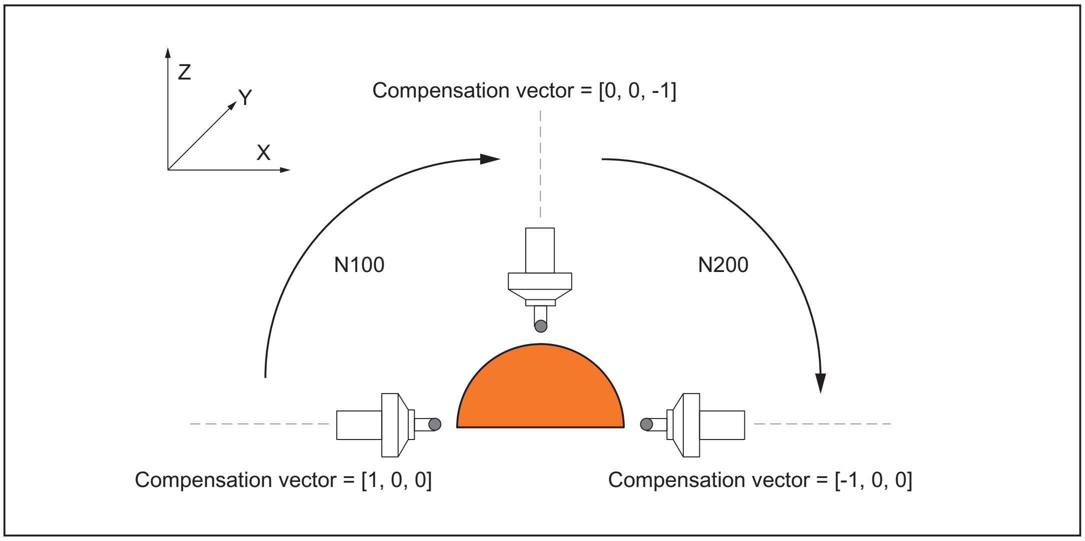

Z
Y
X
Compensation vector = [0, 0, -1]
N100
N200
Compensation vector = [1, 0, 0]
Compensation vector = [-1, 0, 0]

Figure 17-10 Interpolation of the compensation vector

The compensation vector must be oriented by programming the direction axes at [1, 0, 0] before part program block N100. In part program block N100, the end position of the compensation vector is oriented by programming the direction axes at [0, 0, -1].

The intermediate values are generated by path interpolation of all axes programmed in the part program block:

- Geometry axes for the movement of the machining head
- Direction axes of the compensation vector

It is necessary to break the movement down into part program blocks N100 and N200, because an antiparallel orientation of the compensation vector of [1, 0, 0] at the start of the movement and [-1, 0, 0] at the end of the movement (semi-circle) would otherwise result. In this case, the interpolator would interpolate only the X coordinate of the compensation vector, and the orientation of the compensation vector would remain unchanged.

Antiparallel orientation of the compensation vector

When an antiparallel orientation of the compensation vector is programmed in a part program block, the following alarm is displayed:

- Alarm "75018 Channel *number* Block *number* CLC in programmable direction, error ID: 1"

Note

Interpolation of the compensation vector

The interpolation of the compensation vector is not a genuine vector interpolation, as described above, but results from the interpolation of the actual positions of the direction axes.

Consequently, if the compensation vector changes due to the workpiece contour, the interpolation of the direction axes is included in the path interpolation of the geometry axes. In order to minimize the impact of the direction axes on the path interpolation, it is recommended to configure the dynamic response of the direction axes at least equal to or greater (by a factor of approx. 10) than the dynamic response of the geometry axes.

In the case of a re-orientation (rotation) of the compensation vector, it is also necessary to note the ratio between the programmed traversing path and the configured dynamic response of the direction axes. The ratio should be chosen such that the programmed traversing path is not traversed in one or a small number of interpolation cycles, due to the dynamic response of the axis. This causes heavy loads on the machine and, in certain circumstances, may trigger axial alarms and abort part program execution.

Example

Rotation of the compensation vector and thus the machining head through 90°:

- Initial orientation: Parallel to coordinate axis X
- Target orientation: Parallel to coordinate axis Y
  Bad programming of re-orientation:
- [1, 0, 0] → [0, 1, 0]
  Good programming of re-orientation:
- [100, 0, 0] → [0, 100, 0]

Rotation of the workpiece coordinate system

As described above, the compensation vector always refers to the basic coordinate system (machine coordinate system). If the workpiece coordinate system (rotation, mirroring) is transformed to machine a workpiece in such a way that the coordinate axes of both coordinate systems are no longer parallel with the same orientation, a corresponding transformation must be carried out for the compensation vector.

⚠ **CAUTION**

Unequal orientation

If the workpiece coordinate system is transformed such that the coordinate axes of the basic and workpiece coordinate systems are no longer parallel with the same orientation, it is the sole responsibility of the user to ensure that an appropriate transformation of the compensation vector is carried out.

17.6.2 Closed-loop control gain (CLC_GAIN)

Syntax

CLC_GAIN = *Factor*

Factor

- Format: Real
- Range of values: y 0.0

CLC_GAIN is an NC address and can therefore be written together with other instructions in a part program block.

When a negative factor is programmed, the absolute value is used without an alarm output.

Functionality

The current closed-loop control gain for clearance control is produced by the active characteristic specified via machine data:

- MD62510 $MC_CLC_SENSOR_VOLTAGE_TABLE1 (coordinate voltage of interpolation points sensor characteristic 1)
- MD62511 $MC_CLC_SENSOR_VELO_TABLE1 (coordinate velocity of interpolation points sensor characteristic 1)
  or
- MD62512 $MC_CLC_SENSOR_VOLTAGE_TABLE2 (coordinate voltage of interpolation points sensor characteristic 2)
- MD62513 $MC_CLC_SENSOR_VELO_TABLE2 (coordinate velocity of interpolation points sensor characteristic 2)

CLC_GAIN can be used to multiply the closed-loop control gain of the characteristic by a programmable factor.

NOTICE
Imprecise characteristic
Increasing the gain (CLC_GAIN > 1.0) may lead to oscillation in the controlled axes!

Instant of activation

The modified closed-loop control gain is effective in the part program block in which CLC_GAIN has been programmed or, if this block does not contain any executable instructions, in the next part program block with executable instructions.

Response to characteristic changeover

The programmed factor remains active even when the gain characteristic is changed over with CLC_SEL, i.e. it is immediately applied to the newly selected characteristic.

Response to CLC_GAIN=0.0

If the closed-loop control gain for clearance control is deactivated with CLC_GAIN=0.0, the CLC position offset present at the time of deactivation is retained and is not changed. This can be used for example when laser-cutting sheet steel to "skip over" sections of sheet that are not to be machined without foundering.

If the tool orientation is changed when 3D clearance control is active and the closed-loop control gain has been deactivated (CLC_GAIN=0.0), the CLC offset vector is rotated simultaneously. This generally induces an offset in the CLC operating point on the workpiece surface (see following diagram).

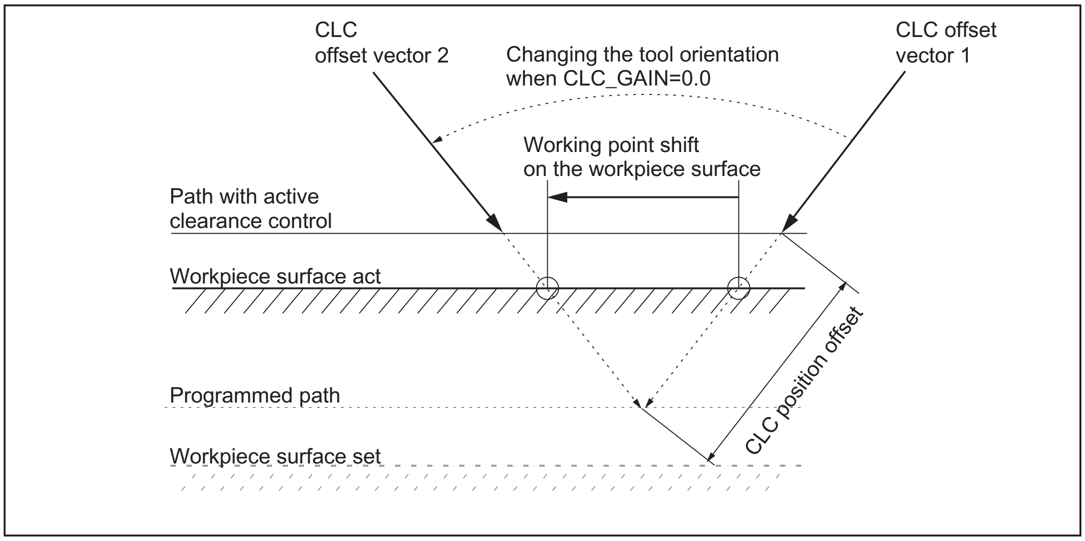

CLC offset vector 2
Changing the tool orientation when CLC_GAIN=0.0
CLC offset vector 1
Working point shift on the workpiece surface
Path with active clearance control
Workpiece surface act
CLC position offset
Programmed path
Workpiece surface set

Figure 17-11 Response of the CLC offset vector when CLC_GAIN=0.0

Reset

Within a part program, a modified gain factor must be reset by means of explicitly programming CLC_GAIN=1.0.

RESET behavior

CLC_GAIN=1.0 becomes effective after a power on reset, NC RESET or end of program.

17.6.3 Limiting the control range (CLC_LIM)

Syntax

CLC_LIM(*lower limit*, *upper limit*)

Lower limit, *upper limit*

Format and value range as machine data:

- MD62505 $MC_CLC_SENSOR_LOWER_LIMIT[n] (lower clearance control motion limit)
- MD62506 $MC_CLC_SENSOR_UPPER_LIMIT[n] (upper clearance control motion limit)

CLC_LIM(...) is a procedure call and must therefore be programmed in a dedicated part program block.

Functionality

The maximum control range for clearance control can be modified on a block-specific basis using CLC_LIM. The maximum programmable lower/upper limit is limited by the limit value preset in the relevant machine data:

- MD62505 $MC_CLC_SENSOR_LOWER_LIMIT[1] (lower clearance control motion limit)
- MD62506 $MC_CLC_SENSOR_UPPER_LIMIT[1] (upper clearance control motion limit)

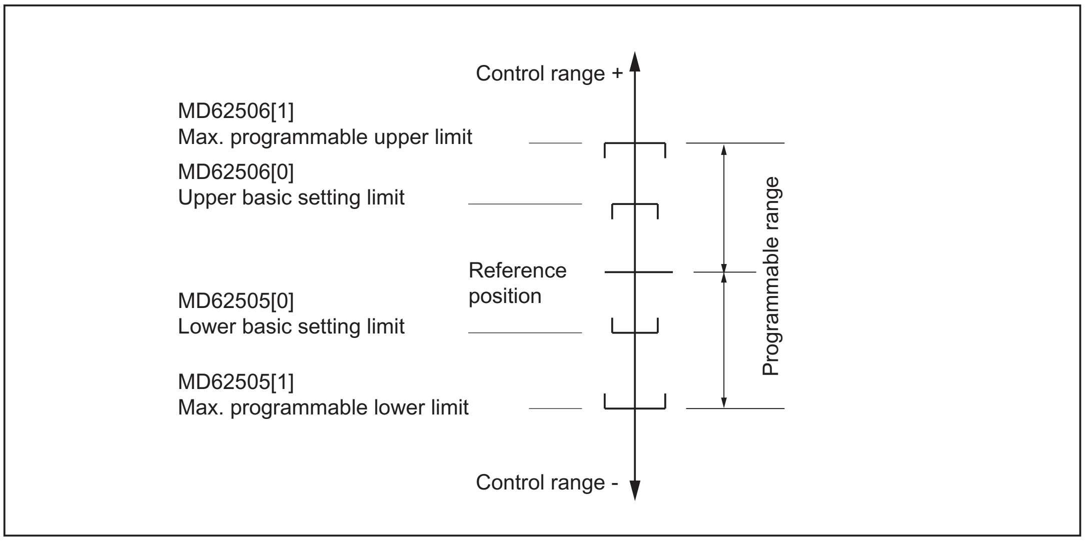

Control range +
MD62506[1]
Max. programmable upper limit
MD62506[0]
Upper basic setting limit
Reference position
MD62505[0]
Lower basic setting limit
MD62505[1]
Max. programmable lower limit
Programmable range
Control range -

Figure 17-12 Value range limits for lower and upper limit

The control range limit is effective in relation to the current programmed setpoint position of the axis. If the limits are changed so that the actual position is located outside the limit, the clearance control automatically effects travel back to the limit range.

Reset

Within a part program, a modified control range limit can be reset by explicitly programming CLC_LIM without a "CLC_LIM( )" argument. This reapplies the limits from the following machine data:

- MD62505 $MC_CLC_SENSOR_LOWER_LIMIT[0] (lower clearance control motion limit)
- MD62506 $MC_CLC_SENSOR_UPPER_LIMIT[0] (upper clearance control motion limit)

RESET response

The default setting from the above-mentioned machine data becomes effective after power on reset, NC RESET and end of program.

Error messages

The following programming errors are displayed with an alarm:

- Programming more than 2 arguments
  - CLC alarm "75005 Channel *number* Block *number* CLC_LIM: general programming error"
- Programming arguments outside the permissible limits
  - CLC alarm "750010 Channel *number* Block *number* CLC_LIM Value greater than MD limit"

17.6.4 Direction-dependent traversing motion disable

Syntax

$A_OUT[*number*] = *enabling signal*

Number

Number of the parameterized digital output (see "Parameter Assignment" section)

- Format: Integer
- Range of values: 1, 2, . . . max. number of digital outputs

Enabling signal

Invertable enable signal (see "Parameter Assignment" section)

- Format: Integer
- Range of values: 0, 1

System variable $A_OUT[n] can be set block-synchronously in the part program or asynchronously via synchronized actions.

Functionality

Parameterizable digital outputs (system variable $A_OUT) can be used for direction-dependent disabling of the traversing motion (manipulated variable) induced via clearance control. As long as e.g. the negative traversing direction is disabled, the clearance-controlled axes will only travel in a positive direction due to the sensor signal.

This can be used for example when laser-cutting sheet steel to "skip over" sections of sheet that are not to be machined without foundering.

Parameterization

The following machine data is used to parameterize the digital outputs:

- MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[n] (assignment of the digital outputs for disabling the CLC movement)
  n = 0 → Digital output for disabling the negative traversing direction
  n = 1 → Digital output for disabling the positive traversing direction

Example

The following digital outputs are to be used:

- $A_OUT[3] to disable the negative traversing direction
- $A_OUT[4] to disable the positive traversing direction

Parameter settings in the machine data:

- MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[0] = 3 (assignment of the digital outputs for disabling the CLC movement)
- MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[1] = 4

Effect:

- $A_OUT[3] = 0 → Negative traversing direction enabled
- $A_OUT[3] = 1 → Negative traversing direction disabled
- $A_OUT[4] = 0 → Positive traversing direction enabled
- $A_OUT[4] = 1 → Positive traversing direction disabled

Inversion of the evaluation

Enter the negative number of the digital output to evaluate the digital output signal with inversion:

Parameter settings in the machine data:

- MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[0] = -3 (assignment of the digital outputs for disabling the CLC movement)
- MD62523 $MC_CLC_LOCK_DIR_ASSIGN_DIGOUT[1] = -4

Effect:

- $A_OUT[3] = 0 → Negative traversing direction disabled
- $A_OUT[3] = 1 → Negative traversing direction enabled
- $A_OUT[4] = 0 → Positive traversing direction disabled
- $A_OUT[4] = 1 → Positive traversing direction enabled

17.6.5 Voltage offset, can be set on a block-specific basis (CLC_VOFF)

Syntax

CLC_VOFF = V*oltage offset*

Voltage offset

- Format: Real
- Unit: Volts
- Range of values: No restrictions

CLC_VOFF is an NC address and can therefore be written together with other instructions in a part program block.

Functionality

CLC_VOFF can be used to preset a constant voltage offset for clearance control, which is subtracted from the input voltage of the clearance sensor. The programmed voltage offset therefore changes the setpoint distance between the workpiece and the clearance sensor or offsets the operating point for clearance control.

The quantitative effect of the voltage offset is dependent on the additional parameters for clearance control and can therefore not be standardized in a generally valid format.

Instant of activation

The voltage offset is effective in the part program block in which CLC_VOFF has been programmed or, if this block does not contain any executable instructions, in the next part program block with executable instructions.

Reset

Within a part program, a voltage offset must be reset by means of explicitly programming CLC_VOFF=0.0.

RESET response

CLC_VOFF =0.0 becomes effective after a power on reset, NC RESET or end of program.

17.6.6 Voltage offset definable by synchronized action

Syntax

$A_OUTA[*number*] = V*oltage offset*

Number

Number of the parameterized analog output (see "Parameter Assignment" section)

- Format: Integer
- Range of values: 1, 2, . . .max. number of analog outputs

V*oltage offset*

Just like the voltage offset for CLC_VOFF (see Section "Voltage offset, can be set on a block-specific basis (CLC_VOFF) (Page 750)").

Functionality

A parameterizable output (system variable $A_OUTA) can be used to apply a voltage offset for clearance control, which, like CLC_OFF, is subtracted from the input voltage of the clearance sensor.

The voltage offset can be modified in the interpolator clock cycle by programming the analog output within a synchronized action.

Parameterization

The following machine data is used to parameterize the analog output:

MD62522 $MC_CLC_OFFSET_ASSIGN_ANAOUT (modification of the setpoint distance by means of sensor signal override)

Example

An external voltage Uext is present at analog input $A_INA[3], which is to be overlaid on the sensor voltage as a continuously variable voltage offset e.g. for test or start-up purposes. $A_OUTA[2] is used as an analog output for the clearance control voltage offset.

Parameter setting for the analog output for clearance control voltage offset:

MD62522 $MC_CLC_OFFSET_ASSIGN_ANAOUT = 2 (modification of the setpoint distance by means of sensor signal override)

The analog input $A_INA[3] is assigned to the clearance control analog output $A_OUTA[2] within a synchronized action:

ID=1 DO $A_OUTA[2] = $A_INA[3]

17.6.7 Selection of the active sensor characteristic (CLC_SEL)

Syntax

CLC_SEL(*characteristic number*)

Characteristic number

- Format: Integer
- Range of values: 1, 2

CLC_SEL(...) is a procedure call and must therefore be programmed in a dedicated part program block.

Characteristic number = 2 selects characteristic 2. Any other value selects characteristic 1 without alarm.

Functionality

CLC_SEL can be used to switch between the sensor characteristics defined in the machine data.

- Characteristic 1:
  - MD62510 $MC_CLC_SENSOR_VOLTAGE_TABLE_1 (coordinate voltage of interpolation points sensor characteristic 1)
  - MD62511 $MC_CLC_SENSOR_VELO_TABLE_1 (coordinate velocity of interpolation points sensor characteristic 1)
- Characteristic 2:
  - MD62512 $MC_CLC_SENSOR_VOLTAGE_TABLE_2 (coordinate voltage of interpolation points sensor characteristic 2)
  - MD62513 $MC_CLC_SENSOR_VELO_TABLE_2 (coordinate velocity of interpolation points sensor characteristic 2)

RESET response

Characteristic 1 becomes effective after a power on reset, NC RESET or end of program.

17.7 Function-specific display data

The "clearance control" technological function provides specific display data for supporting start-up and for service purposes.

Possible applications

Application options for display data include for example:

- Determination of form variances and transient control errors via the variables for the maximum and minimum position offset/sensor voltage.
- Determination of the voltage noise detected by the A/D converter via the variables for the maximum and minimum sensor input voltage. This requires a constant clearance between the clearance sensor and the workpiece surface and the deactivation of clearance control via CLC_GAIN = 0.0.

The minimum and maximum values are detected in the position controller cycle.

Types of variable

The display data is available both as channel-specific GUD (Global User Data) variables and as OPI variables.

17.7.1 Channel-specific GUD variables

The "clearance control" technological function provides the following channel-specific GUD variables for HMI applications:

- SINUMERIK HMI Advanced
- SINUMERIK Operate

Table 17-1 Channel-specific GUD variables

GUD variables	Description	Unit	Access
CLC_DISTANCE[0]	Current position offset	mm	read only
CLC_DISTANCE[1]	Absolute minimum of position offset	mm	read/write
CLC_DISTANCE[2]	Absolute maximum of position offset	mm	read/write
CLC_VOLTAGE[0]	Current sensor input voltage	V	read only
CLC_VOLTAGE[1]	Absolute minimum of sensor input voltage	V	read/write
CLC_VOLTAGE[2]	Absolute maximum of sensor input voltage	V	read/write

Once the technological function has been started up successfully, the GUD variables listed are not displayed automatically on the HMI interface.

SINUMERIK HMI Advanced

Proceed as follows to create and display the GUD variables in HMI Advanced.

1. Setting the password
   Enter the password for protection level 1: (machine manufacturer).
2. Activate the "definitions" display.
   Operating area switchover > Services > Data Selection
3. If no SGUD.DEF file is yet available:
   Operating area switchover > Services > Data admin > New...

- Name: SGUD
- Type: Global data/system
  Confirm with **OK**.
  This opens the file in the editor.

1. Edit the GUD variable definitions
   DEF CHAN REAL CLC_DISTANCE[3] ; Array of real, 3 elements
   DEF CHAN REAL CLC_VOLTAGE[3] ; Array of real, 3 elements
   M30
2. Save the file and close the editor.
3. Activate the SGUD.DEF file.

The GUD variables for clearance control are now displayed under:

Operating area switchover > Parameters > User data > Channel user data

SINUMERIK Operate

Proceed as follows to create and display the GUD variables for SINUMERIK Operate.

1. Setting the password
   Enter the password for protection level 1: (machine manufacturer).
2. If no SGUD.DEF file is yet available:
   Operating area switchover > Commissioning > System data > Open NC data directory: Set cursor to definitions **> New...**

- Name: SGUD
- Type: DEF
  Confirm with **OK**.
  This opens the file in the editor.

1. Edit the GUD variable definitions
   DEF CHAN REAL CLC_DISTANCE[3] ; Array of real, 3 elements
   DEF CHAN REAL CLC_VOLTAGE[3] ; Array of real, 3 elements
   M30
2. Save the file and close the editor.
3. Activate the SGUD.DEF file.

The GUD variables for clearance control are now displayed under:

Operating area switchover > Parameters > User variables > GUD channel

SINUMERIK NC

The newly created GUD variables, which are already being displayed, will only be detected by the clearance control function and supplied with up-to-date values following an NC POWER ON RESET.

Note

Once the GUD variables have been created, an NC POWER ON RESET must be carried out in order for the clearance control function to update the GUD variables.

17.7.2 OPI variable

The technology function "Clearance control" provides the following channel-specific OPI variables as display data for the HMI application:

OPI variable	Description	Unit	Access
CLC[0]	Current position offset	mm	read only
CLC[1]	Absolute minimum of position offset	mm	read/write
CLC[2]	Absolute maximum of position offset	mm	read/write
CLC[3]	Current sensor input voltage	V	read only
CLC[4]	Absolute minimum of sensor input voltage	V	read/write
CLC[5]	Absolute maximum of sensor input voltage	V	read/write
CLC[6]	1st component of the normalized tool orientation vector	-	read only
CLC[7]	2nd component of the normalized tool orientation vector	-	read only
CLC[8]	3rd component of the normalized tool orientation vector	-	read only

OPI variables still have to be registered or defined in the system in order that they can be accessed.

Defining OPI variables

The following operator actions must be carried out to define OPI variables.

1. Create the CLC-specific definition file: **CLC.NSK**
   Note:
   We recommend that you create the file in the \OEM directory rather than in the \MMC2 directory so that it is not overwritten when a new software version is installed.
2. Define the CLC-specific OPI variables.
   Add the following line to the CLC.NSK file:
   ```
   LINK("CLC" ,200,"2 1 1 1 1F# /NC 5 0 1",100)
   ```
3. Create/expand the user-specific definition file: **USER.NSK**
   (See item 1: Note)
4. The call of the CLC-specific definition file CLC.NSK must be added to the USER.NSK definition file. To do this, insert the following line:
   ```
   CALL(CLC.NSK)
   ```

Using LinkItem

In order to use the OPI variables in a DDE control, the "LinkItem" property of the DDE control must be set in accordance with the following example:

```
label1.LinkItem = "CLC[u1,1,9](" "!d%15.4lf" ")"
```

The format string can be modified if necessary.

The following code lines provide an example of how the variables supplied by means of NCDDE access can be distributed on a field of labels:

```
FOR i = 0 To 8
   label2.Caption[i] = Trim$(Mid$(label1.Caption, 1+15*i, 15))
NEXT
```

17.8 Function-specific alarm texts

For details of the procedure for creating function-specific alarm texts, see Section "Creating alarm texts (Page 713)".

17.9 Supplementary conditions

17.9.1 I/O modules

The analog output voltage of the distance sensor for the A/D conversion must be connected to the NC using an I/O module with an analog input.

Connection options

The SIMATIC ET 200S I/O for SINUMERIK 840D sl is connected via PROFIBUS DP. The clearance sensor is connected via an analog S7 I/O module.

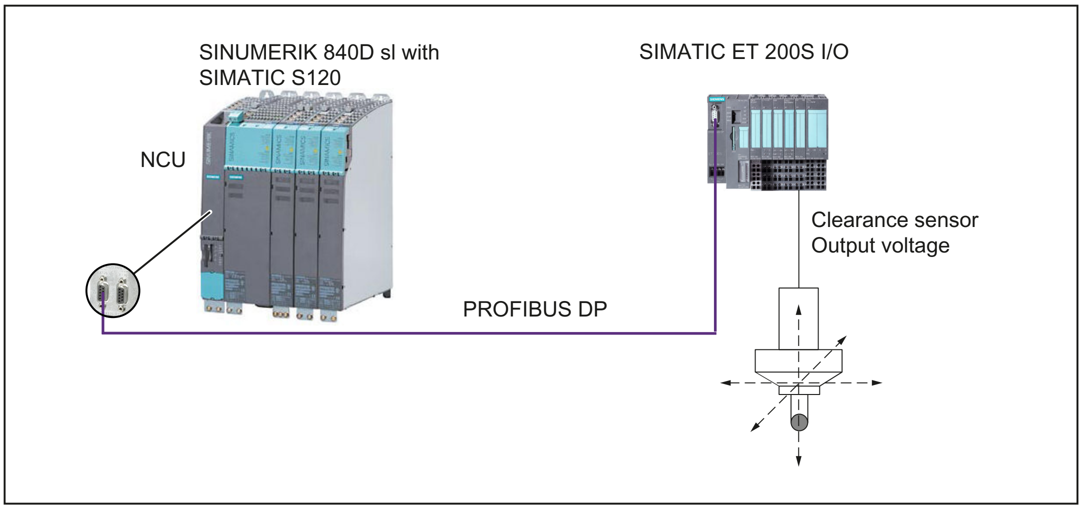


Figure 17-13 I/O modules connection for SINUMERIK 840D sl

Suitable I/O modules

As the A/D conversion time directly affects the deadtime of the clearance control servo loop, only one I/O module may be used with low conversion time.

Suitable SIMATIC S7 I/O modules for the clearance control are:

- Analog I/O module 2 AI, U, high-speed for ET 200S
- Analog I/O module 2 AO, U, high-speed for ET 200S

I/O module connection

The SIMATIC I/O devices of the Production Series ET200, e.g. ET200M, are brought into the S7 project as usual, and configured.

Note

To check whether a module selected from the hardware catalog complies with the module in the automation system, the following procedure is recommended:

1. Note the article numbers of all modules used in the automation system.
2. Select the corresponding module in the hardware catalog and compare the article number of the module used in the automation system with the article number that is displayed in the hardware catalog. Both article numbers must match one another.

17.9.1.1 External smoothing filters

If an external filter is to be interconnected to smooth the output voltage of the clearance sensor before the A/D conversion of the output voltage by the I/O module, please ensure that the resulting time constant is small in relation to the NC position controller cycle.

Note

It is better for the control if electromagnetic shielding is used to ensure a large signal-noise ratio than if smoothing filters are used in the signal path.

17.9.2 Function-specific supplementary conditions

Complete NC Stop

If in conjunction with an NC Stop, not only the programmed path motion but also the traversing movement of the clearance-controlled axes should be stopped, the following NC/PLC interface signals must be set:

- DB21, ... DBX7.3 = 1 (NC Stop)
- DB21, ... DBX7.4 = 1 (NC stop axes and spindles)

Followup

If a clearance-controlled axis is to be switched as an alarm response or via the corresponding interface signal from the NC/PLC in "follow-up" mode, setpoint output will cease for clearance control on this axis.

Travel without software limit switches

If the clearance-controlled axes are to travel without referencing (travel without software limit switches), values outside the used traversing range must still be parameterized for the axis-specific software limit switches:

- MD36100 $MA_POS_LIMIT_MINUS (1st software limit switch minus)
- MD36110 $MA_POS_LIMIT_PLUS (1st software limit switch plus)
- MD36120 $MA_POS_LIMIT_MINUS2 (2nd software limit switch minus)
- MD36130 $MA_POS_LIMIT_PLUS2 (2nd software limit switch plus)

Clearance control takes the machine data into account even if an axis has not been referenced.

Disabling digital/ analog inputs

Neither the analog input for the input voltage of the clearance sensor nor the digital input used by the clearance control in the context of the "Lift fast with position controller cycle" special function can be controlled (disabled) via the PLC:

DB10, DBB0 (inhibit digital NC inputs)

DB10, DBB146 (inhibit analog NC inputs)

See also the description of machine data:

- MD62508 $MC_CLC_SPECIAL_FEATURE_MASK, bit **4** and **5** (Special functions and operating modes of the clearance control)

Gantry axes: Only leading axes

Only one of the clearance-controlled axes may be configured as the master axis of a gantry grouping:

MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition)

The use of following axes in a gantry grouping is not permitted.

Displaying the axis position

The actual current axis position of a clearance-controlled axis as the sum of an interpolatory axis position and the current position offset of clearance control is not displayed in the main machine screen:

- SINUMERIK HMI Advanced:
  The actual current axis position is displayed in the Service screen: **Operating area switchover > Diagnosis > Service displays > Axis/spindle** displayed as the "actual position value".
- SINUMERIK Operate:
  The actual current axis position is displayed in the Service screen: **Operating area switchover > Diagnosis > Axis dialog > Service axis >** displayed as the "measuring system 1 and 2 actual position value".

No virtual axes

Clearance-controlled axes must not be parameterized as virtual axes:

MD30132 $MA_IS_VIRTUAL_AX[<axis>] (axis is virtual axis)

Computing time requirements

The additional computing time required for the "clearance control" technological function must be taken into account on control systems in which the cycle times set for the interpolator and position controller cycle have been substantially optimized in comparison with the default setting:

The additional required computing time results after the activation of the clearance control in the `CLC( ... )` part program. If the interpolation or position controller cycle is exceeded, the following alarm appears:

- Interrupt: "4240 Computing time overflow at IPO or position controller level, IP *point in part program*"

Execution of the part program is aborted.

1D clearance control function

Alarm "1016: System error, ID550010" can occur in the following situation:

- The clearance-controlled axis (e.g. Z axis) is parameterized as geometry axis
- Within any command sequence in which `STOPRE` is initiated implicitly or explicitly, the clearance control with CLC(0) is switched off

This is the reason that it is recommended to parameterize the clearance-controlled axis of a 1D clearance control (e.g. Z axis) so that it is no longer the geometry axis of the channel.

Parameter assignment: Machine data

- MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[<Z axis>] = 0
- MD20060 $MC_AXCONF_GEOAX_NAME_TAB[<Z axis>] = "NO_Z_AXIS"

Programming: Rotations around the Z axis

As the Z axis is no longer a geometry axis after reparameterization, for rotation around the Z axis, instead of the predefined function `CROT()` the predefined procedure`CRPL()` must be used:

```
CROT(Z,<angle>) → CRPL(1,<angle>)
```

17.10 Data lists

17.10.1 Machine data

17.10.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10300	FASTIO_ANA_NUM_INPUTS	Number of active analog NC inputs
10350	FASTIO_DIG_NUM_INPUTS	Number of active digital NC input bytes
10362	HW_ASSIGN_ANA_FASTIN	Hardware assignment of external analog NC inputs 0...7
10712	NC_USER_CODE_CONF_NAME_TAB	List of renamed NC identifiers

17.10.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of compile cycle block elements (DRAM)
28100	MM_NUM_CC_BLOCK_USER_MEM	Memory for compile cycle block elements (DRAM) in kB
28254	MM_NUM_AC_PARAM	Number of parameters for synchronized actions
Clearance control		
62500	CLC_AXNO	Axis assignment for clearance control
62502	CLC_ANALOG_IN	Analog input for clearance control function
62504	CLC_SENSOR_TOUCHED_INPUT	Input bit assignment for the "sensor collision" signal
62505	CLC_SENSOR_LOWER_LIMIT	Lower motion limit of clearance control
62506	CLC_SENSOR_UPPER_LIMIT	Upper motion limit of clearance control
62508	CLC_SPECIAL_FEATURE_MASK	Special functions and operating modes of the clear-ance control
62510	CLC_SENSOR_VOLTABE_TABLE_1	Coordinate voltage of interpolation points sensor char-acteristic 1
62511	CLC_SENSOR_VELO_TABLE_1	Coordinate velocity of interpolation points sensor char-acteristic 1
62512	CLC_SENSOR_VOLTAGE_TABLE_2	Coordinate voltage of interpolation points sensor char-acteristic 2
62513	CLC_SENSOR_VELO_TABLE_2	Coordinate velocity of interpolation points sensor char-acteristic 2
62516	CLC_SENSOR_VELO_LIMIT	Clearance control movement velocity
62516	CLC_SENSOR_ACCEL_LIMIT	Clearance control movement acceleration
62520	CLC_SENSOR_STOP_POS_TOL	Positional tolerance for status message "Clearance control zero speed"

Number	Identifier: $MC_	Description
62521	CLC_SENSOR_STOP_DWELL_TIME	Dwell time for status message "Clearance control zero speed"
62522	CLC_OFFSET_ASSIGN_ANAOUT	Modification of the setpoint distance by means of sensor signal override
62523	CLC_LOCK_DIR_ASSIGN_DIGOUT	Assignment of the digital outputs for disabling the CLC movement
62524	CLC_ACTIVE_AFTER_RESET	Clearance control remains active after RESET
62525	CLC_SENSOR_FILTER_TIME	PT1 filtering time constant of sensor signal
62528	CLC_PROG_ORI_AX_MASK	Programmed orientation vector: Axis mask
62529	CLC_PROG_ORI_MAX_ANGLE	Programmed orientation vector: Maximum difference angle
62530	CLC_PROG_ORI	Programmed orientation vector: Index of the $AC_PARAM variables for the output of the current difference angle

17.10.1.3 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
32070	CORR_VELO	Axis velocity for handwheel, external zero offsets, SA clearance control
32410	AX_JERK_TIME	Time constant for axis jerk filter
32610	VELO_FFW_WEIGHT	Feedforward control factor for velocity feedforward control
36000	STOP_LIMIT_COARSE	Exact stop coarse
36010	STOP_LIMIT_FINE	Exact stop fine
36040	STANDSTILL_DELAY_TIME	Delay time zero speed monitoring
36060	STANDSTILL_VELO_TOL	Axis/ spindle velocity stopped
36750	AA_OFF_MODE	Value calculation mode for axial position override

17.10.2 Drive parameters (SINAMICS S120)

Number	Short name	Long name
p1414[0...n]	n_soll_filt current	Speed setpoint filter activation 1, 2
p1415[0...n]	n_soll_filt 1 type	Speed setpoint filter 1 type
p1416[0...n]	n_soll_filt 1 T	Speed setpoint filter 1 time constant
p1417[0...n]	n_soll_filt 1 fn_n	Speed setpoint filter 1 denominator natural frequency
p1418[0...n]	n_soll_filt 1 D_n	Speed setpoint filter 1 denominator damping
p1419[0...n]	n_soll_filt 1 fn_z	Speed setpoint filter 1 numerator natural frequency
p1420[0...n]	n_soll_filt 1 D_z	Speed setpoint filter 1 numerator damping
p1421[0...n]	n_soll_filt 2 type	Speed setpoint filter 2 type
p1422[0...n]	n_soll_filt 2 T	Speed setpoint filter 2 time constant

Number	Short name	Long name
p1423[0...n]	n_soll_filt 2 fn_n	Speed setpoint filter 2 denominator natural frequency
p1424[0...n]	n_soll_filt 2 D_n	Speed setpoint filter 2 denominator damping
p1425[0...n]	n_soll_filt 2 fn_z	Speed setpoint filter 2 numerator natural frequency
p1426[0...n]	n_soll_filt 2 D_z	Speed setpoint filter 2 numerator damping

17.10.3 Signals

17.10.3.1 Signals to channel

DB number	Byte.bit	Description
21, ...	1.4	Stop CLC motion
21, ...	1.5	Feedrate override acts on CLC

17.10.3.2 Signals from channel

DB number	Byte.bit	Description
21, ...	37.3	CLC is active
21, ...	37.4-5	CLC motion has stopped
21, ...	37.4	CLC motion at lower motion limit
21, ...	37.5	CLC motion at upper motion limit

TE3: Speed/torque coupling, master-slave 18

18.1 Brief description

A master-slave coupling is a speed setpoint coupling between a master and any number of slave axes - performed at the position controller level - with and without torque equalization control. The coupling can be permanently switched on, dynamically switched on/off and reconfigured.

Note

The following restrictions apply for the SINUMERIK 828D with regard to the "Master-slave coupling":

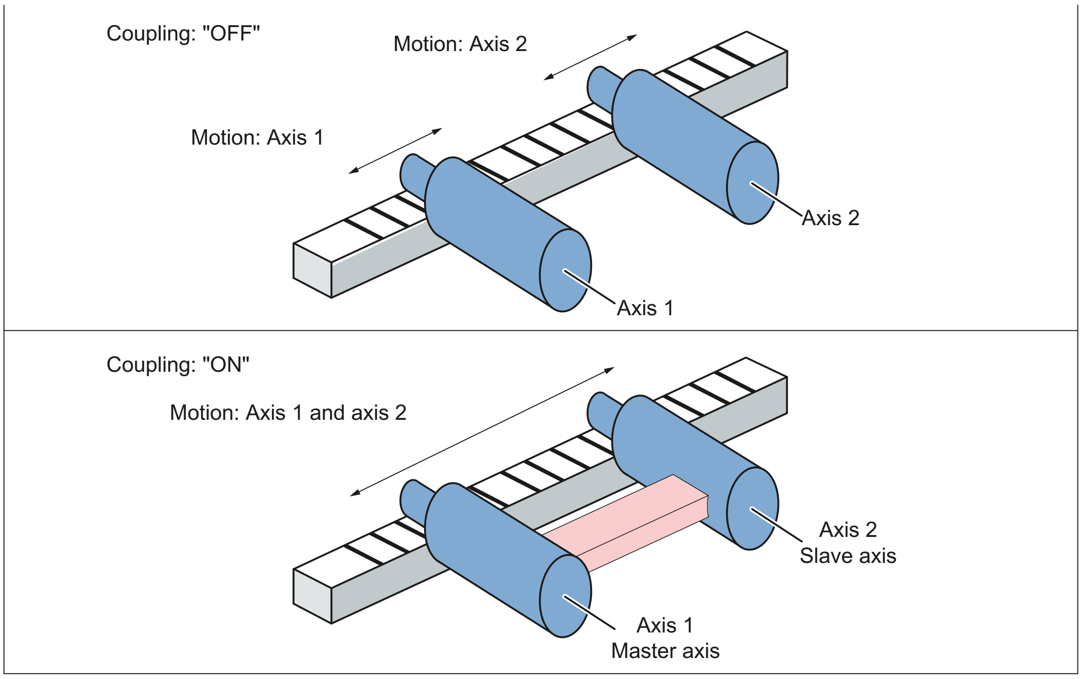
- Only **one** master-slave grouping is possible. (Exception: control variant "SW28x(1) CNC-SW Milling Adv. Export (me822) with max. **two** master-slave groupings).
- Only **one** slave axis can be coupled to the master axis.

Applications

Possible applications of a master-slave coupling include:

- Increase the power for mechanically coupled drives
- Compensating gear and gear tooth flank play by entering a pre-tensioning torque

Dynamic master-slave coupling of two axes



18.2 Coupling diagram

If the coupling is closed, the slave axis is traversed only with the load-side setpoint speed of the master axis. It is therefore only speed-controlled, not position-controlled. There is no differential position control between the master and slave axis.

The torque required between the master and slave axis is distributed using the torque equalization controller. When using different motors, this distribution can be adapted to the specific requirements using a weighting factor.

A tension can be established between the master and slave axes by entering a supplementary torque (tensioning torque).

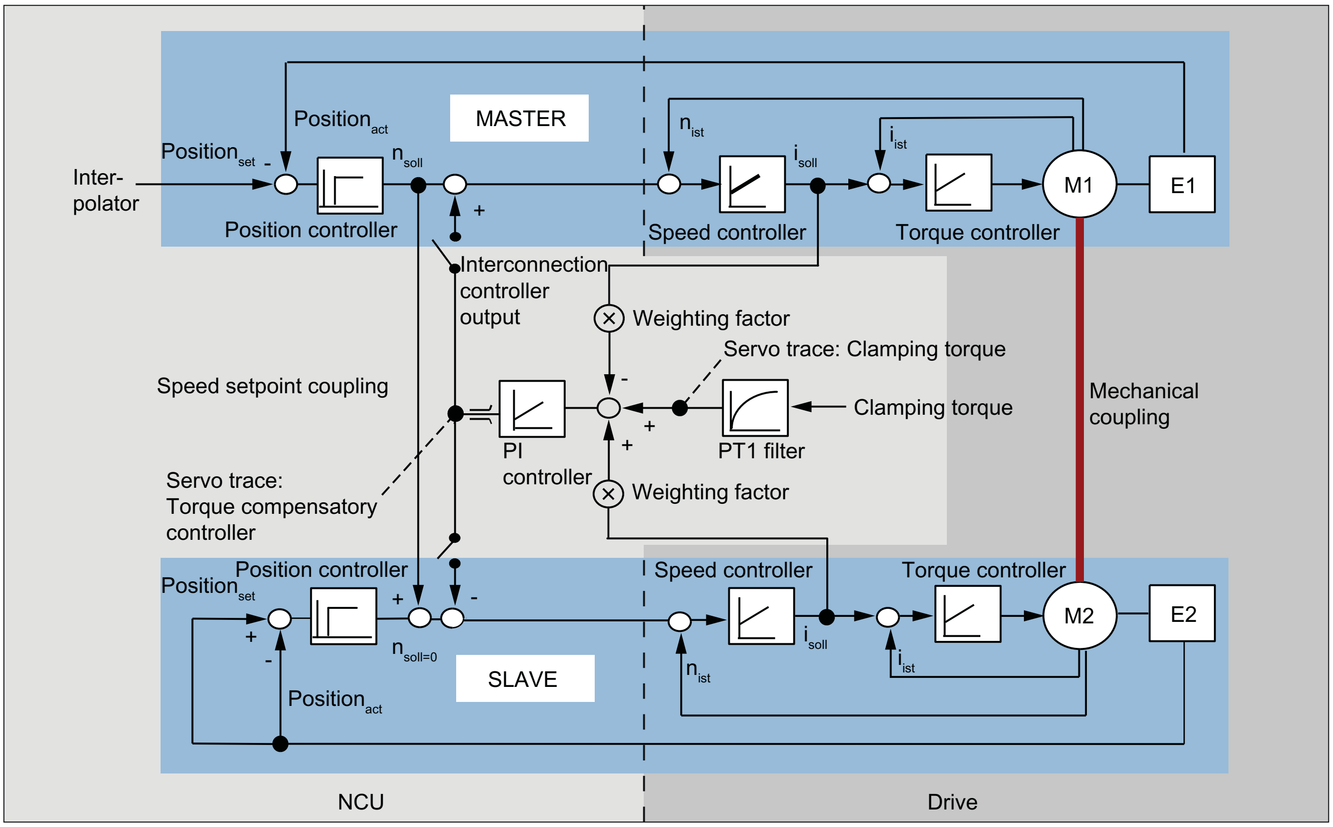


Figure 18-1 Control structure

18.3 Configuring a coupling

Static assignment

For a speed setpoint coupling and torque equalization control, the static assignment of master and slave axis is defined separately in the following machine data:

- Speed setpoint coupling
  MD37250 $MA_MS_ASSIGN_MASTER_SPEED_CMD[<slave axis>] =
  <machine axis number of the master axis for the speed setpoint coupling>
- Torque equalization control
  MD37252 $MA_MS_ASSIGN_MASTER_TORQUE_CTR[<slave axis>] =
  <machine axis number of the master or slave axis for torque equalization control>
  (see Section "Tension torque (Page 770)")

Dynamic assignment (only 840D sl)

The assignment of master and slave axes can be dynamically changed using the following program commands.

Defining assignments

Assignment of one or several slave axes to a master axis:

```
MASLDEF (<slave axis_1>, <slave axis_2>, ..., <master axis>)
```

In principle, any number of slave axes can be assigned. The assignment is kept even after an operating mode change, reset and end of part program.

Deleting assignments

Delete assignment of one or several slave axes to their associated master axis:

```
MASLDEL (<slave axis_1>, <slave axis_2>, ...)
```

An active coupling is first implicitly switched off before deleting the assignment.

Axis assignment for speed setpoint coupling and torque equalization control

For the speed setpoint coupling, the slave axis refers to the master axis specified when defining the assignment (`MASLDEF`).

The axis to which the slave axis refers for torque equalization control is defined in the following machine data:

MD37253 $MA_MS_FUNCTION_MASK[<slave axis>], bit 1 = <value>

<value>	Description
0	In the case of dynamic assignment (`MASLDEF`), the slave axis for the torque equalization control – as well as for speed setpoint coupling – refers to the master axis of the master-slave grouping.
1	In the case of dynamic assignment (`MASLDEF`), the slave axis for the torque equalization control refers to the axis, master or slave axis of the master-slave grouping specified in machine data: MD37252 $MA_MS_ASSIGN_MASTER_TORQUE_CTR<slave axis> = <axis>

(See Section "Tension torque (Page 770)")

Supplementary conditions

The following supplementary conditions must be observed for the dynamic assignment:

- When the coupling is switched on, a change to the assignment using `MASLDEF` has no effect. The change only becomes effective the next time that the coupling is switched off.
- The same master axis is always used for the speed setpoint coupling and torque equalization control.
- A plausibility check of the assignment is only made when the coupling is switched on.

User-specific standard assignment after reset (only 840D sl)

A user-specific standard assignment, which is always effective after a reset, can be defined in PROG_EVENT.SPF using the program commands `MASLDEF` and `MASLDEL`.

The following machine data should be set in order that PROG_EVENT.SPF is executed for a reset: MD20108 $MC_PROG_EVENT_MASK.Bit 2 = 1

Example: Dynamic change of the assignment (only 840D sl)

The assignment of slave axis AX3 is changed from master axis AX1 to master AX2. To do this, the coupling must be temporarily switched off (see Section "Closing/opening a coupling (Page 773)").

	Programming	Description
①	`MASLDEF (AX3, AX1)`	Assignment of slave axis AX3 to master axis AX1
②	`MASLON (AX3)`	Switching on the coupling
③	`MASLDEL (AX3)`	Switch off the coupling and cancel the assignment between AX3 and AX1
④	`MASLDEF (AX3, AX2)`	Assignment of slave axis AX3 to master axis AX2

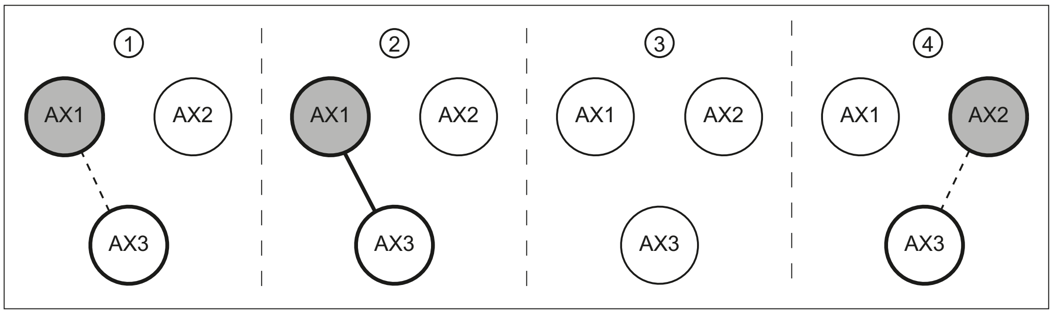


Figure 18-2 Alternating assignment of the slave axis

General supplementary conditions

Please note the following general supplementary conditions:

- A slave axis can only be assigned to one master axis
- A master axis can be assigned several slave axes
- A slave axis must not be a master axis or be in another master-slave relationship.

Note

Drive optimization

At a SINAMICS S120 drive unit, a maximum of three drives can be optimized or measured at the same time (speed controller optimization / function generator). Therefore, for a coupling with more than three coupled drives at the same time, we recommend that these are distributed over several drive units.

18.4 Torque compensatory controller

The torque compensatory controller (PI control) calculates a load-side initial speed setpoint. The additional speed setpoint can be differently entered via the following machine data:

MD37254 $MA_MS_TORQUE_CTRL_MODE[<slave axis>] = <value>

<value>	Input of the additional speed setpoint at:
0	Master and slave axis (default value)
1	Slave axis
2	Master axis
3	No additional setpoint injection

Note

Several slave axes

For a master-slave grouping with one master and several slave axes, the input of the additional speed setpoint to the master and slave axes - set as standard in MD37254 - can result in instability. For multiple slave axes, injection of an additional speed setpoint should only be applied in the slave axes:

MD37254 $MA_MS_TORQUE_CTRL_MODE = 1

SINAMICS S120: Current setpoint filter

Up to four current setpoint filters (p1656 - p1674, function block diagram [5710]) can be activated in the SINAMICS S120. In a master-slave group, it is recommended that, for the master and all of the slave axes, only filters with the same data be activated at any one time.

Scaling

Scaling of the machine data for the gain factor (P component) (MD37256 $MA_MS_TORQUE_CTRL_P_GAIN) and the speed setpoint limiting (MD37260 $MA_MS_MAX_CTRL_VELO) is defined in the following machine data:

MD37253 $MA_MS_FUNCTION_MASK[<slave axis>], bit 0 = <value>

<value>	Description
0	MD37256 and MD37260 are internally multiplied with the following factor: 1 / interpolator clock cycle 1)
1	MD37256 and MD37260 are accepted unchanged
1) The interpolator cycle is displayed with MD10071 $MN_IPO_CYCLE_TIME	

Note

We recommend not changing MD37256 and MD37260:

MD37253 $MA_MS_FUNCTION_MASK[<slave axis>], bit 0 = 1

Gain factor (P component)

The gain factor of the torque compensatory controller is set in the following machine data as a percentage of the ratio of the maximum load-side axis velocity of the slave axis (MD32000 $MA_MAX_AX_VELO) to its rated torque (SINAMICS S120: p2003):

MD37256 $MA_MS_TORQUE_CTRL_P_GAIN[<slave axis>]

Note: Scaling via MD37253 $MA_MS_FUNCTION_MASK[<slave axis>], bit 0

Integral time (I component)

The integral time of the torque compensatory controller is set in the following machine data:

MD37258 $MA_MS_TORQUE_CTRL_I_TIME[<slave axis>]

The I component is deactivated as standard.

Speed setpoint limiting

The speed setpoint calculated by the torque compensatory controller can be limited using the following machine data. The input is a percentage referred to the maximum, load-side axis velocity of the slave axis (MD32000 $MA_MAX_AX_VELO).

MD37260 $MA_MS_MAX_CTRL_VELO[<slave axis>]

Note: Scaling via MD37253 $MA_MS_FUNCTION_MASK[<slave axis>], bit 0

Deactivating the torque compensatory controller

If the following settings are made, the torque compensatory controller will be inactive:

- MD37254 $MA_MS_TORQUE_CTRL_MODE[<slave axis>] = 3
- MD37256 $MA_MS_TORQUE_CTRL_P_GAIN[<slave axis>] = 0

Torque weighting

The percentage of the torque generated by the slave axis of the total torque can be set using the torque weighting.

MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE[<slave axis>]

Using the torque weighting, it is possible to configure a different torque distribution between the master and slave axes for motors with different rated torques. For motors of the master and slave axes with the same rated torques, it makes sense to have a 50 % torque distribution (default setting).

The torque provided by the master axis is given by:

Torque provided by the master axis = 100 % - MD37268

Note

Mechanical coupling

When using the torque compensatory controller, it is absolutely necessary to have a mechanical coupling between the master and slave axis. Otherwise, the two drives involved could accelerate in an uncontrollable fashion.

Activation / Deactivation via the NC/PLC interface

Activation of the torque compensatory controller via the NC/PLC interface must be explicitly enabled:

MD37255 $MA_MS_TORQUE_CTRL_ACTIVATION[<slave axis>] = 1

Activation

The torque compensatory controller can be activated axis for axis via:

DB31, ... DBX24.4

Reading the actual state

The actual activation state of the torque compensatory controller can be read via:

DB31, ... DBX96.4

18.5 Tension torque

The tension torque is a supplementary torque which is switched to the active torque equalization controller. This means that a mechanical tension can be established between axes within a master-slave grouping. Establishing a tension is not only possible between the master and a slave axis, but also between two slave axes by declaring one of the slave axes the reference axis for the torque equalization controller.

Setting

The tension torque is entered as a percentage of the rated torque of the slave axis and is active immediately:

MD37264 $MA_MS_TENSION_TORQUE[<slave axis>] = <tension torque>

As can be seen from the structure of the torque equalization controller (Chapter "Torque compensatory controller (Page 767)"), the tension torque is entered via a PT1 filter. The filter time constant is set using the following machine data:

MD37266 $MA_MS_TENSION_TORQ_FILTER_TIME[<slave axis>] = <time constant>

A value for the filter time constant greater than 0 activates the filter.

NOTICE
No axis equalization
If a tension torque is entered without a mechanical coupling between the master and slave axis then this causes the axes to move.

Note

Minimum tension torque

The tension torque chosen must be high enough to ensure that the resulting torque does not drop below the minimum required tension even during acceleration.

Reducing the motor temperature rise

The tension torque can be reduced when the motor is at a standstill to reduce the motor temperature rise.

Example 1 (840D sl): Static coupling and establishing a tension in pairs

A master-slave application is parameterized so that one master axis is assigned three slave axes and a tension torque is entered for one axis pair. As a consequence, the master axis alone does not have to establish the tension torque with respect to all of the slave axes.

Assumption regarding machine axes:

- 1st to 4th machine axis: AX1, AX2, AX3, AX4

Static coupling for all slave axes

- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2] = 1
- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX3] = 1
- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX4] = 1

Axis	Reference axis of the speed setpoint coupling MD37250 = value		Reference axis of the torque equalization controller MD37252 = value		Input of the torque equalization contr. MD37254 = value	
	Value	Description	Value	Description	Value	Description
AX1	0	No reference axis	0	No reference axis	0	Master and slave
AX2	1	Master axis, AX1	1	Master axis, AX1	0	Master and slave
AX3	1	Master axis, AX1	0	No reference axis	3	No input
AX4	1	Master axis, AX1	3	2nd slave axis, AX3	0	Master and slave

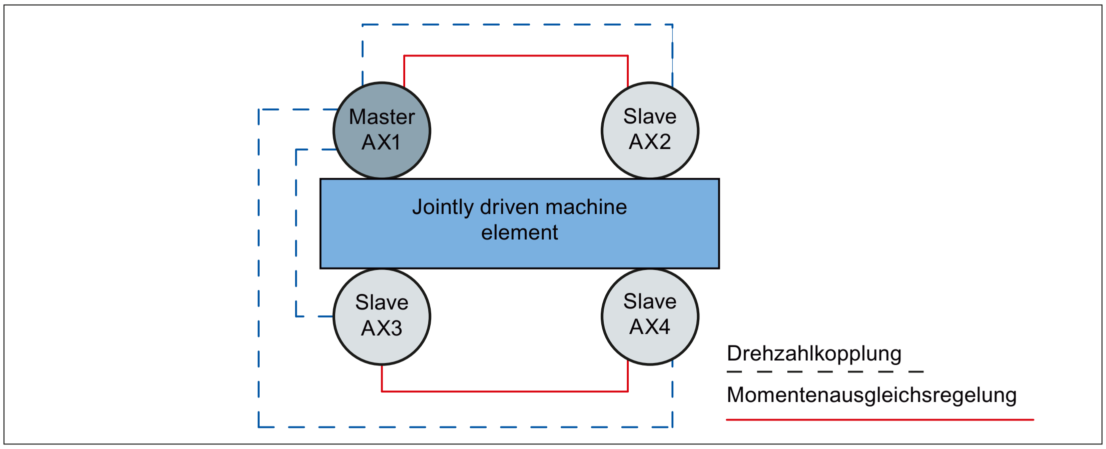


Figure 18-3 Example 1: Torque equalization control in pairs

Example 2 (840D sl): Dynamic coupling with 1x4 and 2x2 axes grouped in pairs

Assumption regarding machine axes:

- 1st to 4th machine axis: AX1, AX2, AX3, AX4

Dynamic coupling for all slave axes:

- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2] = 0
- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX3] = 0
- MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX4] = 0

For a dynamic coupling, the master axis specified when defining the coupling (`MASLDEF`) is, as standard, the reference axis for speed setpoint coupling and torque equalization control.

The definition of reference axis AX3 for the torque equalization control of the 3rd slave axis AX4 is required for the application "1x4 axes" where the following machine data is set: MD37253 $MA_MS_FUNCTION_MASK[AX4], Bit 1 = 1 (see part program)

Axis	Reference axis of the speed setpoint coupling MD37250 = value		Reference axis of the torque equalization controller MD37252 = value		Input of the torque equalization contr. MD37254 = value	
	Value	Description	Value	Description	Value	Description
AX1	0	No reference axis	0	No reference axis	0	Master and slave
AX2	0	No reference axis	0	No reference axis	0	Master and slave
AX3	0	No reference axis	0	No reference axis	3	No input
AX4	0	No reference axis	3	Machine axis AX3	0	Master and slave

```
Program code                              Comment
PROC MASL_SWITCH IPRTLOCK DISPLOF
  IF MASL_REQUEST==4                      ; Activate "1x4 axes"
    MASLOF(AX2, AX4)                      ; Switch off coupling
    MASLDEL(AX2, AX4)                     ; Delete coupling
    $MA_MS_FUNCTION_MASK[AX4]=            ; MD37253, bit1 = 1:
    $MA_MS_FUNCTION_MASK[AX4] B_OR 'B10'  ; Reference axis of the torque equalization controller
                                          ; from AX4, is AX3 (MD37252)
    NEWCONF                               ; Activate machine data change
    STOPRE
    MASLDEF(AX2, AX3, AX4, AX1)           ; Define assignment for "1x4 axes"
    MASLON(AX2, AX3, AX4)                 ; Switch on couplings
    MASL_ACTIVE=4 MASL_REQUEST=0          ; Feedback signal: "1x4 axes" switched-in
  ENDIF
  IF MASL_REQUEST==2                      ; Activate "2x2 axes"
    MASLOF(AX2, AX3, AX4)                 ; Switch off coupling
    MASLDEL(AX2, AX3, AX4)                ; Delete coupling
    $MA_MS_FUNCTION_MASK[AX4]=            ; MD37253, bit1 = 0:
    $MA_MS_FUNCTION_MASK[AX4] B_AND       ; Reference axis of the torque equalization controller
'HFFFD'                                   ; from AX4, is the master axis defined
                                          ; with MASLDEF
    NEWCONF
    STOPRE
    MASLDEF(AX2, AX1)                     ; Define assignment for the 1st "2x2 axes"
    MASLDEF(AX4, AX3)                     ; Define assignment for the 2nd "2x2 axes"
    MASLON(AX2, AX4)                      ; Switch on couplings
    MASL_ACTIVE=2 MASL_REQUEST=0          ; Feedback signal: "2x2 axes" switched on
  ENDIF
RET
```

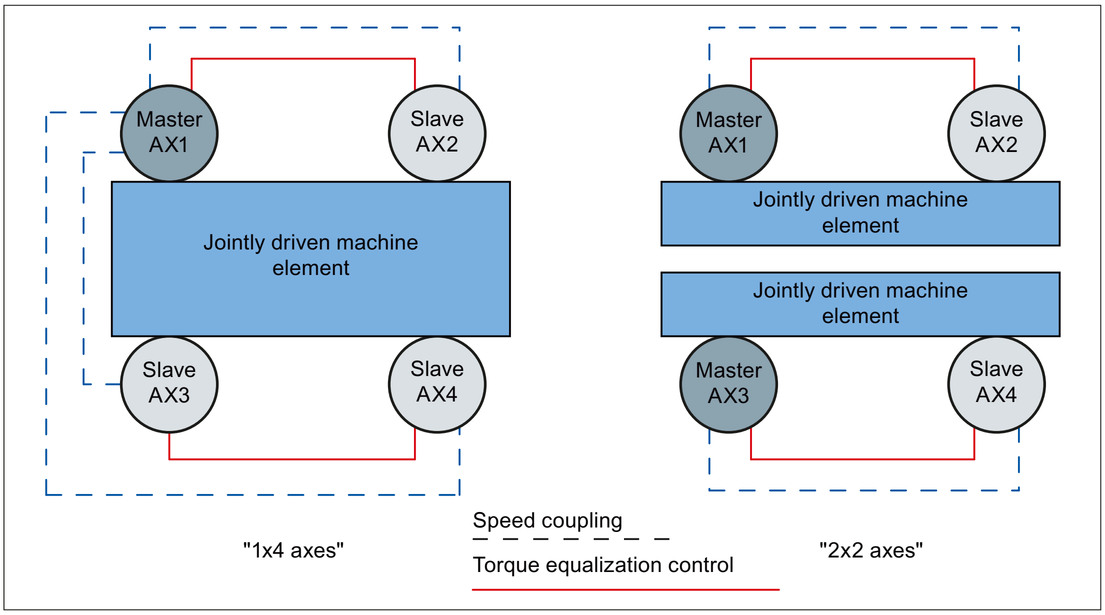

Master AX1
Slave AX2
Jointly driven machine element
Slave AX3
Slave AX4
Master AX1
Slave AX2
Jointly driven machine element
Jointly driven machine element
Master AX3
Slave AX4
"1x4 axes"
Speed coupling
Torque equalization control
"2x2 axes"

Figure 18-4 Example 2: Alternating coupling with 1x4 and 2x2 axes

18.6 Closing/opening a coupling

Default setting

After the control has booted, the following machine data defines whether the coupling is permanently switched on (static) or can be dynamically switched on/off and reconfigured:

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[<slave axis>] = <switch on mode>

Statically switching on a coupling

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[<slave axis>] = 1

After the control boots, the coupling is statically switched on.

By writing the machine data in the part program or synchronized action with subsequent warm restart `NEWCONF`, then a static coupling can also be switched off/on at a later time.

Example: Switching the static coupling for slave axis AX2 off and on

Program code	**Comment**
`N100 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2] = 0`	`; Coupling type: static -> dynamic`
`N110 NEWCONF`	`; Activate machine data.`
`N120 MASLOF(AX2)`	`; Switch off coupling.`
`....`	
`N200 PRESETON(AX2,...)`	`; E.g. actual value setting of the slave axis.`
`....`	
`N300 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2] = 1`	`; Coupling type: dynamic -> static, switch on coupling.`

Note

A statically switched on coupling can neither be switched on/off nor reconfigured using the master-slave-specific NC/PLC interface signals and/or program commands.

Dynamically switching a coupling on/off

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[<slave axis>] = 0

The coupling can be dynamically switched on and off and reconfigured.

The coupling can be switched on/off by:

- Machine data MD37262
  Writing the machine data to switch on (= 1) or switch off (= 0) in the part program or synchronized action. The change becomes active immediately.
- NC/PLC interface
  DB31, ... DBX24.7 = 1 or 0 (master-slave on or off)
- Program commands (only 840D sl)
  - Switch on: `MASLON (<slave axis_1>, <slave axis_2>, ...)`
  - Switch off: `MASLOF (<slave axis_1>, <slave axis_2>, ...)`
  - Switching off with braking of the slave spindles:
    `MASLOFS (<slave spindle_1>, <slave spindle_2>, ...)`
  - Switching off with deletion of the assignment:
    `MASLDEL (<slave axis_1>, <slave axis_2>, ...)`

Note

Changing the static assignment

The setting:

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[<slave axis>] = 0

Permits the coupling to be dynamically changed. This means that it is possible to define a new assignment using `MASLDEF`. The assignment parameterized in the machine data (see static assignment, Section "Configuring a coupling (Page 765)") takes effect only after the control has been powered up again (NC reset).

Coupling state

System variable

The actual coupling state of a slave axis can be read in the part program and synchronized action using the following system variable:

$AA_MASL_STAT[<slave axis>]

Value	Description
0	1) The coupling of the slave axis is not active. 2) The specific axis is not a slave axis
> 0	The coupling is active. <value> == machine axis number of the master axis

NC/PLC interface signal

The current coupling state of a slave axis can be read using the following axis-specific NC/PLC interface signal:

DB31, ... DBX96.7 (master/slave coupling active)

Setpoint state

Requests to switch on/off a master-slave coupling (NC/PLC interface signal and/or program commands) that consecutively follow one another mutually overwrite one another. The effective setpoint state results from the last chronological request (see Section "Response on activation/deactivation (Page 775)").

18.7 Response on activation/deactivation

On/off at standstill (axis)

A request to switch on/off the coupling only becomes effective when the master and slave axis are at a standstill (zero speed):

DB31, ... DBX61.4 == 1 (axis/spindle stationary)

In this case, the coupled axes must be in the closed-loop controlled mode.

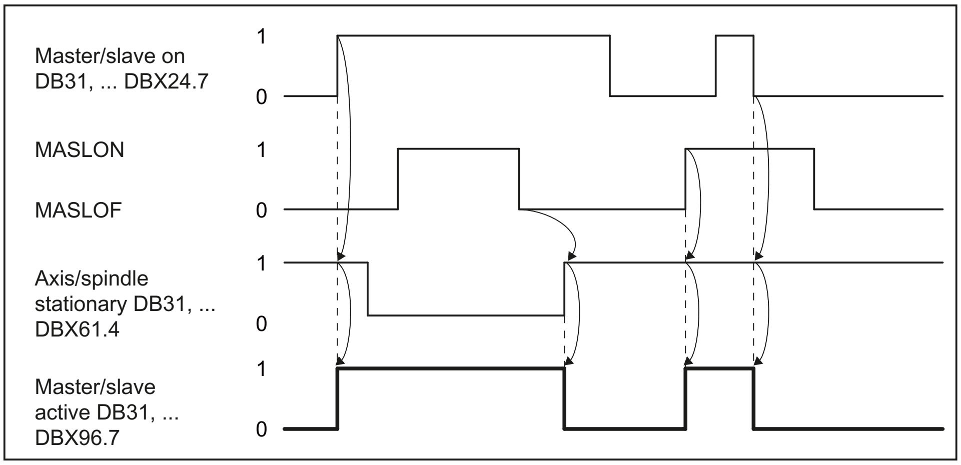

Master/slave on DB31, ... DBX24.7
1
0
MASLON
1
MASLOF
0
1
Axis/spindle stationary DB31, ... DBX61.4
0
1
Master/slave active DB31, ... DBX96.7
0

Figure 18-5 Activation procedure

When switching on the coupling using the program command `MASLON`, the system waits until the coupling is closed before the block is changed. The "Master-slave switchover active" message will be displayed on the user interface during this time.

Switching on/off during motion (spindle)

Note

Switching on/off during motion

During the motion, the coupling can be switched on/off only for **spindles** in **speed control mode**.

Switching on

When switching on during the motion, the coupling operation divides itself at different speeds in two phases.

- Phase 1
  The PLC user program must request the switch on of the coupling with:
  DB31, ... DBX24.7 = 1 (master/slave ON)
  The slave spindle accelerates or brakes along a ramp to the setpoint speed of the master spindle.
  When the setpoint speed is reached, the coupling is switched on and the NC/PLC interface signal is set:
  DB31, ... DBX96.7 == 1 (coupling active)
  If the master spindle is accelerated while the coupling is being switched on, then Phase 1 is extended corresponding to the dynamic difference between the master and the slave spindle.
- Phase 2
  The following synchronous signals are generated from the actual speed difference between the master and slave spindle(s):
  - DB31, ... DBX96.3 (speed tolerance, coarse)
  - DB31, ... DBX96.2 (speed tolerance, fine)
    The associated limits are set using the following machine data:
  - MD37270 $MA_MS_VELO_TOL_COARSE ("Tolerance coarse")

– MD37272 $MA_MS_VELO_TOL_FINE ("Tolerance fine").

Note

The "Speed tolerance coarse" signal can be used to implement a monitoring function on the PLC side that checks a coupled master-slave grouping for loss of speed synchronism.

The "Speed tolerance fine" signal can be used to directly derive the time taken to close the coupling mechanically and switch on the torque equalization controller.

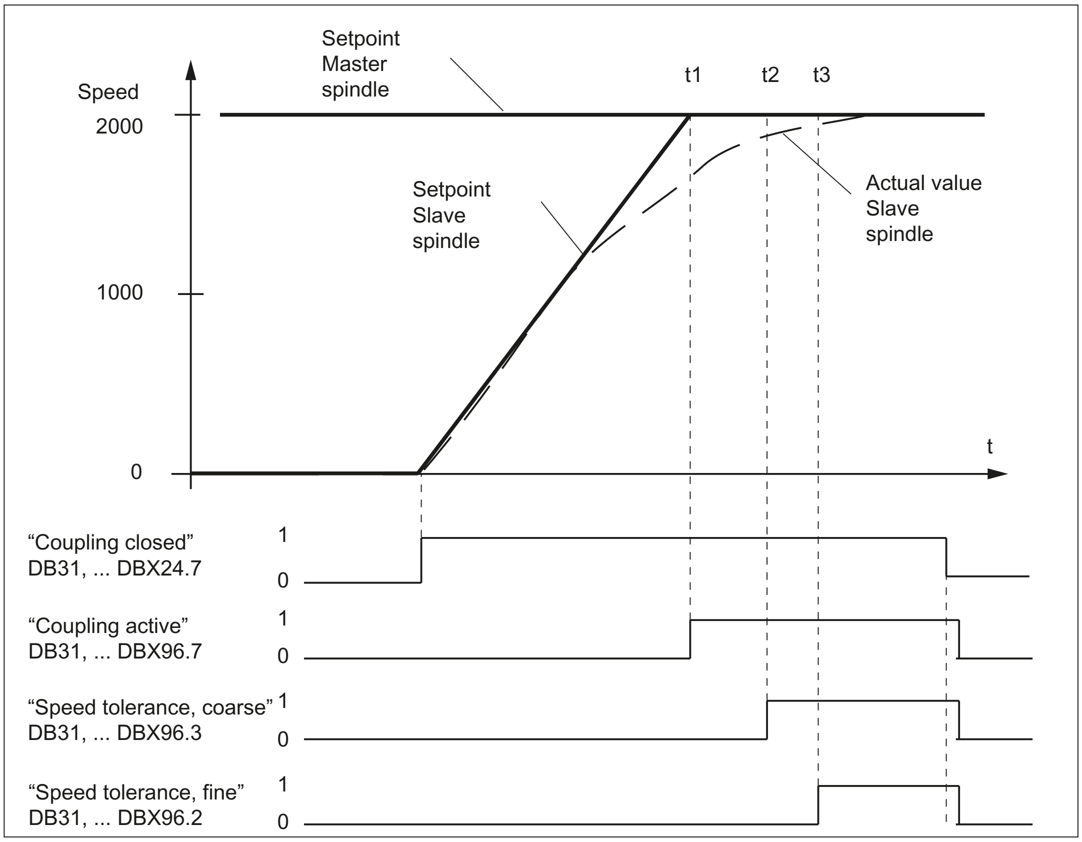


Figure 18-6 Coupling operation at different speeds

Switch off without braking

If the coupling is switched off with the `MASLOF` program command, the coupling will be switched off immediately for spindles in speed control mode. The slave spindles maintain their speeds at the time that the coupling is switched off until a new speed is programmed.

Switch off with braking

If the coupling is switched off with the `MASLOFS` program command, the coupling will be switched off immediately for spindles in speed control mode and the slave spindles braked.

Note

The implicit preprocessing stop is omitted for `MASLON` and `MASLOF`. The missing preprocessing stop means that the $P system variables of the slave spindles do not supply updated values until reprogrammed.

Coupling characteristics

The behavior regarding the program commands `MASLON`, `MASLOF`, `MASLOFS`, `MASLDEL` (only 840D sl) and the NC/PLC interface signal: DB31, ... DBX24.7 (master/slave on) is, for spindles in the open-loop speed controlled mode, set using the following machine data:

MD37263 $MA_MS_SPIND_COUPLING_MODE[<slave spindle>] = <value>

<value>	Description
0	Coupling and disconnection take place only at standstill. The current coupling state is retained until all axes involved have actually come to a stand-still. The `MASLOFS` and `MASLOF` operations are identical; the slave spindle is not decelerated automatically.
1	Coupling and disconnection takes place immediately and therefore during motion. During coupling, the slave spindles are accelerated automatically to the current speed of the master spindle. On disconnection, the slave spindles rotating at this time retain their speeds until next speed programming. However, a slave spindle disconnected with `MASLOFS` decelerates automat-ically.

18.8 Supplementary conditions

18.8.1 Functional boundary conditions

General

- Master and slave axes must be on the same NCU.
- A coupling is closed or opened independent of the channel state the next time that the axis is at a standstill.
- Slave axes can only be traversed via the master axis when the coupling is closed.
- A coupled slave axis cannot be switched further via the axis container.
- A requested axis interchange of coupled slave axes is not executed.

- When the coupling is switched-in (closed) via the slave axis, the master axis is braked automatically, if the channel axis is in the same channel: ⇒ asymmetrical behavior when opening and closing the coupling:
  – Activate: the leading axis is automatically braked
  – Deactivating: **no** automatic braking of the leading axis
- The setpoint position of a coupled slave axis corresponds to its actual position.
- For axes and spindles in the positioning mode, the coupling is only switched-in (closed) and switched-out (opened) at standstill (zero speed).
- The leading-following coupling must be deactivated prior to a gear change or a star-delta switchover.

Axial monitoring functions

- Except monitoring the speed setpoint and actual speed, axial monitoring functions like contour and standstill in the slave axis are inactive because of the missing position controller. The position control circuit parameters like gain factor, precontrol, balancing can thus be set differently for master and slave axes without the monitoring functions responding.
- In order to ensure the same braking behavior for all axes of a master-slave group, when the coupling is active, the same alarm reaction is applied to all of the axes of the master-slave group.
- The parameterization and evaluation of the standstill detection (e.g. braking ramp) is realized axially for each axis in the master-slave group. This can result in an asynchronous withdrawal of the controller enable for the coupled axes.
- When canceling error states, the slave axes do not reposition to the point of interruption.

Virtual axis

Master and/or slave axes must not be "virtual" axes:

MD30132 $MA_IS_VIRTUAL_AX (axis is virtual axis) = 0

Modulo rotary axes

- For slave axes, when the coupling is closed, the actual value displayed in the service display is not modulo 360°. Independent of the setting in machine data:
  MD30310 $MA_ROT_IS_MODULO[<slave axis>]

Spindles

- If a master-slave coupling is activated with spindles, the slave spindle is operated in the open-loop speed controlled mode. The actual value of the slave spindle is not displayed as modulo 360° in the service display. However, in the automatic basic display the actual value is displayed as modulo 360°.
- If the spindles are accelerated at the current limit, this may mean that no adjusting reserves are left over in the coupled state for the torque compensatory controller to use to distribute the torque between the master and the slave as required.
- The maximum master spindle chuck speed must be configured in the following machine data to be lower than or equal to that of the following spindles:
  MD35100 $MA_SPIND_VELO_LIMIT[<master spindle>]
- The axial velocity monitoring function should be adapted to the chuck speed:
  MD36200 $MA_AX_VELO_LIMIT[<master spindle>]

18.8.2 Axial NC/PLC interface signals

- In the brake control logic, the NC/PLC interface signal may longer be evaluated for slave axes: DB3x.DBX61.5 (position controller active
). When the coupling is closed, the signal is no longer set. Instead, the following NC/PLC interface signal should be used:
  DB31, ... DBX96.7 (master-slave coupling active).
- When requested, and for a closed coupling, the following NC/PLC interface signals are directly taken from the master axis for the slave axis:
  DB31, ... DBX2.1 (controller enable)
  DB31, ... DBX21.7 (pulse enable)
- When the controller enable is cancelled for the master axis:
  DB31, ... DBX2.1 (controller enable) = 0
  within the configured time of:
  MD36610 $MA_AX_EMERGENCY_STOP_TIME
  then this also results in interpolatory braking of the slave axis. The corresponding speed and current controller enable signals are only cancelled after the configured time has expired
  MD36620 $MA_SERVO_DISABLE_DELAY_TIME
- In order to facilitate a standard braking behavior for all coupled axes, it is recommended that the following parameters are set the same.
  - MD36620 $MA_SERVO_DISABLE_DELAY_TIME
  - p9560 (pulse suppression, shutdown speed)
  - p1228 (pulse suppression, delay time)

- If, for the master or slave axis, one of the following drive status signals is not set:
  DB31, ... DBX61.7 (current controller active) == 0 OR
  DB31, ... DBX61.6 (speed controller active) == 0
  then, when the slave axis is at a standstill, the status signal is reset:
  DB31, ... DBX96.7 (master/slave active) = 0
  As soon as the master and slave axis are again in closed-loop control:
  DB31, ... DBX61.7 (current controller active) == 1 AND
  DB31, ... DBX61.6 (speed controller active) == 1
  then for the slave axis, the status signal is again set:
  DB31, ... DBX96.7 (master/slave active) = 1
- The torque equalization controller can be activated via the NC/PLC interface using:
  DB31, ... DBX24.4 = 1 (torque equalization controller on)
  The status of the torque equalization controller is displayed in:
  DB31, ... DBX96.4 (master/slave equalization controller active)

Note

The slave axis is only closed-loop speed controlled when the coupling is closed. The NC/PLC interface signal: DB31, ... DBX61.5 (position controller active) is no longer set.

18.8.3 Interaction with other functions

Function generator

To calibrate the speed control loop with activated coupling, it is recommended that a low value should be set for the torque weighting of the slave axis (MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE). Traversing of a mechanical coupled-motion slave axis is not prevented by the torque compensatory controller.

Reference point approach

If the coupling is switched on, only the master axis can be referenced. Referencing of slave axes is suppressed. The referencing requirement does not have to be explicitly canceled for the slave axis in order to do this. The referencing status of coupled slave axes remains unchanged. The slave axis position is generally not the same as the master axis position. This difference in position is not significant. If the coupling is separated, each axis can be referenced separately as usual.

Position-related compensations

Position-related compensations of the slave axis, such as spindle pitch errors, backlash, temperature and sag offsets are not active because the position controller of the slave axis is not active.

Correct calculation of the backlash compensation requires that the backlash of the slave axis is always overtraveled by the motion of the master axis in coupled mode. Disconnecting the coupling during an axis reversal error will generate an incorrect actual value for the slave axis.

Dynamic stiffness control (DSC)

The "Dynamic stiffness control (DSC)" function must either be active or not active for all axes of a master-slave grouping.

MD32640 $MA_STIFFNESS_CONTROL_ENABLE

Speed/torque feedforward control (FFW)

The "Speed/torque feedforward control (FFW)" function does not have to be activated explicitly in the slave axis. The corresponding settings of the master axis are used. The speed feedforward value of the master axis is already incorporated in the speed setpoint of the slave axis. For an active torque feedforward control, the load-side torque feedforward control value of the master axis is also active in the slave drive.

The mechanical situation changes when the coupling is switched on. Axis-specific settings must be adapted accordingly. All coupled drives should have the same speed control dynamics.

Gantry

If one master-slave relationship is defined on each side of the gantry grouping to increase the gain, only the leading axis or following axis may be operated as master axis.

Travel to fixed stop (FXS)

The "Travel to fixed stop (FXS)" function can be programmed only in the master axis when a coupling is active and has a different effect on the master and slave axes:

- The programmed value is expressed as a percentage of the rated drive torque of the master axis. The master axis detects when the fixed stop has been reached.
- The programmed value is also active on the slave axis, but refers to the drive torque of the slave axis.

If the rated torque values of the master and slave axes are different, they can be adapted to each other through the following slave axis machine data:

MD37014 $MA_FIXED_STOP_TORQUE_FACTOR

Specifying a factor < 1 reduces the programmed clamping torque in the slave axis.

Please note the following supplementary conditions:

- Torque distribution between the master and slave axes is not possible during clamping as the torque compensatory controller is switched off during clamping operations.
- Status changes to the master-slave coupling have no effect during travel to fixed stop. Specification of a new status is only accepted when the fixed stop function has been completed.

Safety Integrated (new 840D sl)

As the slave axis is traversed via the master axis speed setpoint, the axis-specific setpoint limitation MD36933 $MA_SAFE_DES_VELO_LIMIT is inactive in the coupled slave axes. All safety monitoring functions remain active in the slave axes however.

Gear stage change with activated master-slave coupling

An automatic gear stage change in a coupled slave spindle is not possible and can only be implemented indirectly using the master spindle. The point in time at which the gear stage is changed is then derived from the master spindle. The oscillating motion of the coupled slave spindle is generated implicitly via the oscillating motion of the master spindle.

In contrast to the master spindle, the associated parameter block must be explicitly selected in the coupled slave spindle. To enable the parameter set specification, the following machine data must be set:

MD35590 $MA_PARAMSET_CHANGE_ENABLE = **2**

Note

For more information about gear stage change and parameter sets for changes in spindle mode, see:

References:

Function Manual, Basic Functions; Spindles (S1)

Axis container

If a coupled slave axis is configured as a container axis, alarm "4025 Switch axis container %3 not permitted: Master-slave active channel %1 Axis %2" is displayed for an axis container rotation.

Axis containers with changing master can be used to make the relevant spindle the master spindle following a rotation of the axis container. Both master and slave spindles can be container spindles.

For a coupling to be closed after axis container rotation using a different spindle in each case, the old coupling must be separated before the rotation, the configuration deleted and the new coupling configured and closed after the rotation.

Example of a cyclic coupling sequence (position = 3 / container = CT1)

Program code	Comment
`MASLDEF(AUX,SPI(3))`	`; S3 master for AUX`
`MASLON(AUX)`	`; Coupling in for AUX`
`M3=3 S3=4000`	`; Machining ...`
`MASLDEL(AUX)`	`; Delete configuration and release coupling`
`AXCTSWE(CT1)`	`; Axis container rotation`

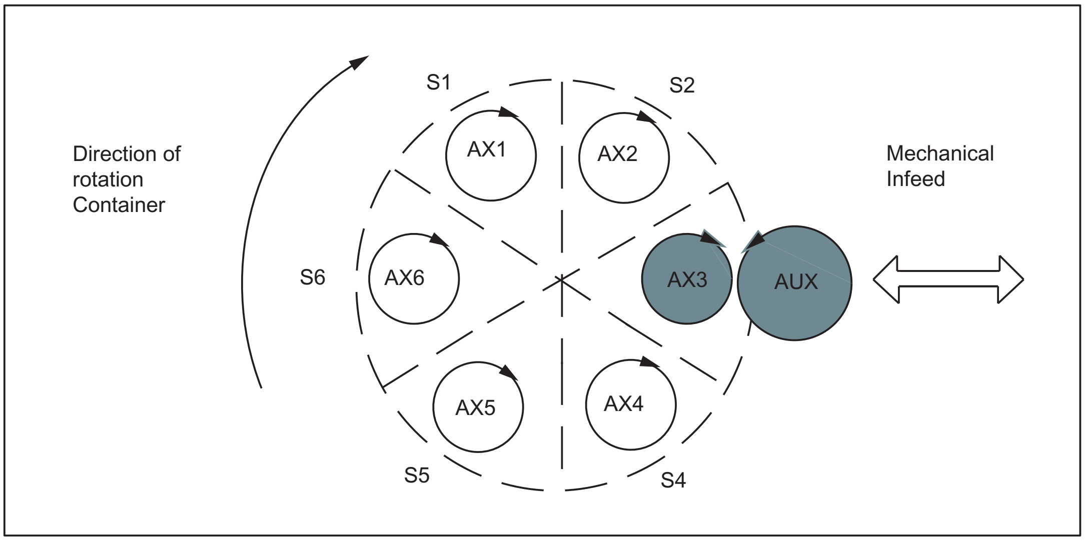


Figure 18-7 Coupling between container spindle S3 and auxiliary motor AUX (prior to rotation)

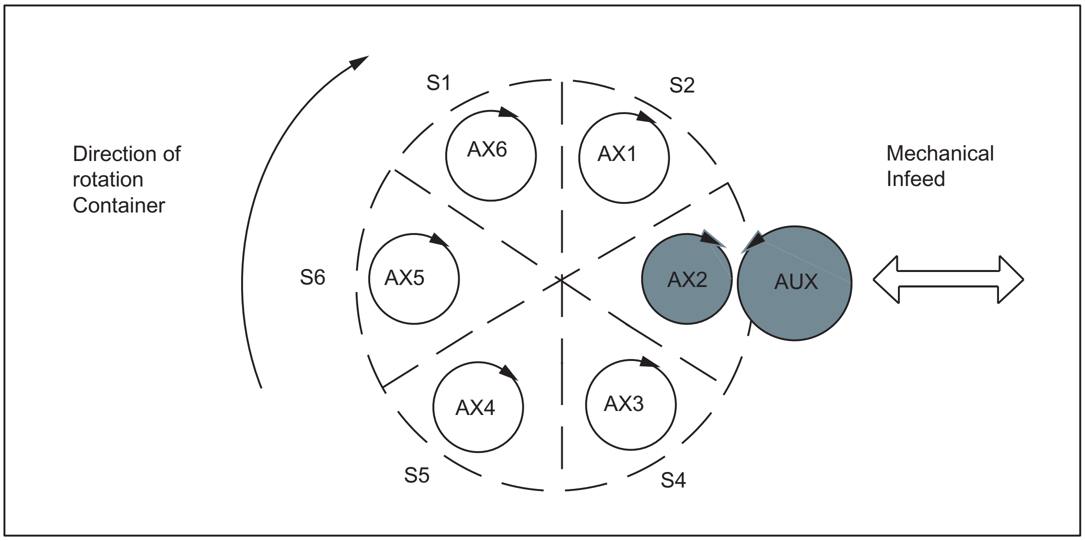


Figure 18-8 Coupling between container spindle S3 and auxiliary motor AUX (after to rotation)

Hardware and software limit switches

If the software or hardware limit switch is overtraveled by a slave axis, the master-slave grouping is stopped via the master axis. The relevant slave axis is shown in the alarm. The limit switch of the slave axis is overtraveled by the amount of the deceleration distance of the master axis.

The movement away of the slave axis from the limit switch must be performed via the traversing of the master axis. Switching off the coupling (`MASLOF`) and traversing the slave axis is not possible. The coupling cannot be switched off until the cause of the error has been removed.

NOTICE
Power off/on or warm restart (reset (po)) after the software or hardware limit switch has been passed - and the coupling switched out with MASLOF
If the software or hardware limit switch is overtraveled by a slave axis and an attempt is made to switch off the coupling with `MASLOF` while the slave axis is still behind the limit switch, the message "Waiting for coupled slave axis" is displayed. The coupling can only be switched off by means of a Power Off/On or warm restart (reset (po)).

Block search

Static coupling

The "Block search with calculation" function (SERUPRO) can be used without any restrictions in conjunction with a static master-slave coupling.

Dynamic coupling (only 840D sl)

For a dynamic coupling, the following restrictions must be observed with regard to the program commands `MASLON` and `MASLOF`:

- The coupled axes must be on the same NCU.
- The coupled axes must be in the same channel when the block search is executed.
- After the block search has been completed, the coupling state, the associated axis positions and speeds must be subsequently influenced by the user via the system ASUB "PROGEVENT.SPF". System variables are available for this purpose.
  - $P_PROG_EVENT: This variable provides information about the event which activated the subprogram. A value of 5 stands for block search.
  - $P_SEARCH_MASLC[slave axis name]: This variable stands for alteration of the coupling status during a block search.
  - $P_SEARCH_MASLD[slave axis name]: This variable indicates the positional offset calculated in the block search between the slave and master axes at the instant the coupling was closed.
  - $AA_MASL_STAT[slave axis name]: This variable indicates the current coupling state.

- The system ASUB "PROGEVENT.SPF" must be saved under the following path: /_N_CMA_DIR/_N_PROG_EVENT_SPF
- The following machine data should be parameterized so that PROGEVENT.SPF is started. NC-specific machine data:
  - MD11450 $MN_SEARCH_RUN_MODE = 'H02'
  - MD11602 $MN_ASUP_START_MASK = 'H01'
  - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100

  Channel-specific machine data:
  - MD20105 $MC_PROG_EVENT_IGN_REFP_LOCK = ’H3F’

Table 18-1 Example 1: PROGEVENT.SPF: Example 1

Program code	**Comment**
`N10 IF $P_PROG_EVENT==5`	`; Block search active`
`N20   IF (($P_SEARCH_MASLC[Y]<>0) AND ($AA_MASL_STAT[Y]<>0))`	`; The coupling state has changed during block search AND the current state is "coupled"`
`N30     MASLOF(Y)`	`; Disconnect coupling`
`N40     SUPA Y=$AA_IM[X]-$P_SEARCH_MASLD[Y]`	`; Traverse through the position offset using the slave axis`
`N50     MASLON(Y)`	`; Close coupling`
`N60   ENDIF`	
`N70 ENDIF`	
`N80 REPOSA`	

Table 18-2 Example 2: PROGEVENT.SPF

Program code	**Comment**
`N10 IF $P_PROG_EVENT==5`	`; Block search active`
`N20   IF (($P_SEARCH_MASLC[SPI(2)]<>0) AND ($AA_MASL_STAT[SPI(2)]==0))`	`; The state of the second spindle has changed during block search AND the current state is "disconnected"`
`N30     M2=$P_SEARCH_SDIR[2]`	`; Update direction of rotation`
`N40     S2=$P_SEARCH_S[2]`	`, Update speed`
`N50   ENDIF`	
`N60 ENDIF`	
`N70 REPOSA`	

For more application examples (see Chapter "Examples (Page 788)).

Note

For an activated coupling, it is recommended to only use block search type 5, "Block search via program test" (SERUPRO) for a block search.

More information about event-driven program calls and "block search using the program test" (SERUPRO) is included in:

References:

Function Manual, Basic Functions;

- Auxiliary function outputs to PLC (H2)
- Mode group, channel, program operation, reset response (K1)
- Spindles (S1)

18.9 Examples

18.9.1 Master-slave coupling between AX1=Master and AX2=Slave.

Configuration

Master-slave coupling between AX1=Master and AX2=Slave.

1. Machine axis number of master axis for speed setpoint coupling
   MD37250 $MA_MS_ASSIGN_MASTER_SPEED_CMD[AX2] = 1
2. Master axis with torque distribution identical to master axis with speed setpoint coupling
   MD37252 $MA_MS_ASSIGN_MASTER_TORQUE_CTR[AX2] = 0
3. Permanent coupling
   MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2] = 1
4. Torque is injected in both the master and slave axes
   MD37254 $MA_MS_TORQUE_CTRL_MODE[AX2] = 0
5. Torque distribution between the master and slave axes is 50% to 50%
   MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE[AX2] = 50
6. Parameters of the torque compensatory controller
   MD37256 $MA_MS_TORQUE_CTRL_P_GAIN[AX2] = 0.5
   MD37258 $MA_MS_TORQUE_CTRL_I_TIME[AX2] = 5.0

18.9.2 Close coupling via the PLC

This application allows you to close or separate a master-slave coupling between the machine axes AX1=Master axis and AX2=Slave axis during operation.

Preconditions

- One configured master axis
  MD37250 $MA_MS_ASSIGN_MASTER_SPEED_CMD ≠ 0
- Activation of a master-slave coupling via
  MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE=0
- The coupling is open.

Typical sequence of operations

Action	Effect/comment
• Approach coupling position	Each axis moves to the coupling position.
• Close coupling mechanically	Both axes are mechanically coupled to one another.
• Request to close the coupling	PLC interface signal "Master/slave on" DB32, ... DBX24.7 is set.
• Read back coupling state	When the axis is at a standstill, the coupled slave axis sets PLC interface signal "Master/slave active" DB32, ... DBX96.7 and clears "Position controller active" DB32, ... DBX61.5. Wait for checkback signal.
• Moving the master-slave group	The master axis is moved.

18.9.3 Closing/separating the coupling via the part program for the SINUMERIK 840D sl

This application allows you to close or separate a master-slave coupling between the machine axes AX1=Master axis and AX2=Slave via the part program.

Preconditions

- A configured master axis MD37250 0 0.
- Not a static master-slave coupling:
  MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE= 0
- The coupling is open.

Parts program

Program code	Comment
`N10 G0 AX1=0 AX2=0`	`; Approach the coupling position on an axis-for-axis basis`
`N20 MASLON (AX2)`	`; (mechanically connect the axes)`
	`; Close the coupling.`
`N30 AX1=100`	`; Move the master-slave group using the master axis.`

Program code	Comment
`N40 MASLOF (AX2)`	`; Open the coupling.`
	`; (Mechanically separate the axes)`
`N50 AX1=200 AX2=200`	`; Move the axes separately.`
`N60 M30`	

18.9.4 Release the mechanical brake

This application allows implementation of a brake control for machine axes AX1=Master axis and AX2=Slave axis in a master-slave coupling.

Preconditions

- Master-slave coupling is configured.
- Axes are stationary.
- No servo enable signals.

Typical sequence of operations

Action	Effect/comment
• Request to close the coupling	The following PLC interface signal is set: DB31, ... DBX24.7 (Master/Slave ON)
• Set controller enable	PLC interface signal "Servo enable" DB31, ... DBX2.1 is set for both axes.
• Interpreting the feedback	Link PLC interface signals of the master axis using AND: DB31, ... DBX61.7 (current controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX61.5 (position controller active) Link PLC interface signals of the saster axis using AND: DB31, ... DBX61.7 (current controller active) DB31, ... DBX61.6 (speed controller active) DB31, ... DBX96.7 (master/slave active)
• Release brakes	If the result of the AND operations on the master and slave axes is ≠ 0, the brake may be released.

18.10 Data lists

18.10.1 Machine data

18.10.1.1 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
37250	MS_ASSIGN_MASTER_SPEED_CMD	Leading axis for speed setpoint coupling
37252	MS_ASSIGN_MASTER_TORQUE_CTR	Leading axis for torque distribution
37254	MS_TORQUE_CTRL_MODE	Connection of torque control output
37255	MS_TORQUE_CTRL_ACTIVATION	Activate torque compensatory control
37256	MS_TORQUE_CTRL_P_GAIN	Gain factor of torque compensatory controller
37258	MS_TORQUE_CTRL_I_TIME	Reset time for torque compensatory controller
37260	MS_MAX_CTRL_VELO	Limitation of torque compensatory control
37262	MS_COUPLING_ALWAYS_ACTIVE	Permanent master-slave coupling
37263	MS_SPIND_COUPLING_MODE	Coupling characteristics of a spindle
37264	MS_TENSION_TORQUE	Master-slave tension torque
37268	MS_TORQUE_WEIGHT_SLAVE	Torque weighting of the slave axis
37270	MS_VELO_TOL_COARSE	Master-slave velocity tolerance "coarse"
37272	MS_VELO_TOL_FINE	Master-slave velocity tolerance "fine"
37274	MS_MOTION_DIR_REVERSE	Invert master-slave direction of travel

18.10.2 System variables

After a block search, the coupling status and associated axis positions can be adjusted subsequently by means of a system ASUB (asynchronous subroutine) "PROGEVENT.SPF". System variables $P_SEARCH_MASL_C, $P_SEARCH_MASL_D and $AA_MASL_STAT are available for this purpose; they can be used to alter the positional offset between the coupled axes and the coupling status:

Identifier	Meaning
$P_SEARCH_MASLC[following axis name]	This variable registers a change in the coupling state during the SERUPRO block search.
$P_SEARCH_MASLD[following axis name]	This variable indicates the positional offset between the slave and master axes at the instant the coupling was closed.
$AA_MASL_STAT[following axis name]	This variable indicates the current coupling state. Value ≠ 0: "Master-slave coupling active" In this case, it contains the current machine number of the master axis and, if the NCU link is active (several operating panel fronts and NCUs), also the NCU No. at the hundreds position. Example: 201 for Axis 1 on NCU2.

18.10.3 Signals

18.10.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Request master-slave torque compensatory controller On==1 / Off==0	DB31, ...DBX24.4	DB380x.DBX5000.4
Request master-slave coupling On==1 / Off==0	DB31, ...DBX24.7	DB380x.DBX5000.7

18.10.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
State master-slave speed difference "fine"	DB31, ... DBX96.2	DB390x.DBX5000.2
State master-slave speed difference "coarse"	DB31, ... DBX96.3	DB390x.DBX5000.3
State master-slave torque compensatory controller On==1 / Off==0	DB31, ... DBX96.4	DB390x.DBX5000.4
State master-slave coupling On==1 / Off==0	DB31, ... DBX96.7	DB390x.DBX5000.7

TE4: Handling transformation package - 840D sl only 19

19.1 Brief description

The handling transformation package has been designed for use on **manipulators** and **robots**. The package is a type of modular system, which enables the customer to configure the transformation for his machine by setting machine data (provided that the relevant kinematics are included in the handling transformation package).

Section structure

The "Handling transformation package" section deals with the following topics:

- The "Kinematic transformation (Page 794)" section describes the kinematic transformation environment.
- The "Definition of terms (Page 794)" section explains some basic terms.
- The "Configuration of the kinematic transformation (Page 797)" section explains the machine data required for the configuration.
- The "Kinematic descriptions (Page 811)" section uses configuring examples to illustrate the most commonly used 2-axis to 5-axis kinematics that can be configured with the "Handling transformation package".
- The "Tool orientation (Page 832)" to "Tool programming (Page 839)" sections handle the programming. describing orientation programming, the entry of tool parameters and transformation calls.

Abbreviations

FL	Flange coordinate system
HP	Wrist point coordinate system
IRO	Internal robot coordinate system
p_3, q_3, r_3	Coordinates of the last basic axis
RO	Robot or base center point coordinate system (= basic coordinate system)
WS	Workpiece coordinate system
T	Tool coordinate system
x_3, y_3, z_3	Coordinates of the first wrist axis

19.2 Kinematic transformation

Task of a transformation

The purpose of a transformation is to transform movements in the tool tip, which are programmed in a Cartesian coordinate system, into machine axis positions.

Fields of application

The handling transformation package described here has been designed to cover the largest possible number of kinematic transformations implemented solely via parameter settings in machine data. Currently, kinematics with two to maximum six axes to be included in the transformation can be configured with this package, corresponding to up to five spatial degrees of freedom. In this case, a maximum of 3 degrees are available for translation and 3 degrees for orientation, This package thus allows a tool (milling cutter, laser beam) to be oriented by a 5-axis machine 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 instructions for the real machine axes. The kinematic transformation requires information about the design (kinematics) of the machine, which are stored in machine data.

Kinematic categories

The handling transformation package is divided into two categories of kinematics, which can be selected via machine data:

MD62600 $MC_TRAFO6_KINCLASS (kinematic category)

- STANDARD: This category includes the most commonly used kinematics.
- SPECIAL: Special kinematics

19.3 Definition of terms

19.3.1 Units and directions

Lengths and angles

In the transformation machine data, all lengths are specified in millimeters or inches and, unless otherwise stated, all angles in degrees at intervals of [-180°, 180°].

Direction of rotation

In the case of angles, arrows in the drawings always indicate the mathematically positive direction of rotation.

19.3.2 Definition of positions and orientations using frames

In order to make a clear distinction from the term "frame" as it is used in the NC language, the following description explains the meaning of the term "frame" in relation to the handling transformation package.

Frame

A frame can be used to translate one coordinate system into another. In this respect, a distinction must be made between translation and rotation. Translation results in an offset; a rotation, a rotation of the coordinate system with respect to the initial system. Coordinates X, Y and Z are used as designator for a translation. They are defined so as to result in a legal coordinate system.

Translation

A translation is always specified in relation to the coordinate directions of the initial system. These directions are assigned to machine data as follows:

- X direction: ..._POS[**0**]
- Y direction: ..._POS[**1**]
- Z direction: ..._POS[**2**]

Rotation

A rotation is specified in RPY angles A, B and C (R=roll, P=pitch, Y=yaw). The positive direction of rotation is defined by the right hand rule, i.e. if the thumb on the right hand is pointing in the direction of the axis of rotation, then the fingers are pointing in the positive angular direction. Angles A and C are defined in the interval [-180; +180], and B in the interval [-90; +90].

The definitions of the RPY angles are as follows:

- Angle A: 1. rotation about the Z axis of the initial system
- B angle: 2. Rotation through the rotated Y axis
- C angle: 3. rotation about the twice rotated X axis

The RPY angles are assigned to machine data as follows:

- Angle A: ..._RPY[**0**]
- B angle: ..._RPY[**1**]
- C angle: ..._RPY[**2**]

The following diagram shows an example for a rotation through the RPY angle. The initial coordinate system X1, Y1, Z1 is transitioned into the coordinate system X4, Y4, Z4

1. Rotation through angle A around the Z1 axis
2. Rotation through angle B around the Y2 axis
3. Rotation through angle C around the X3 axis.

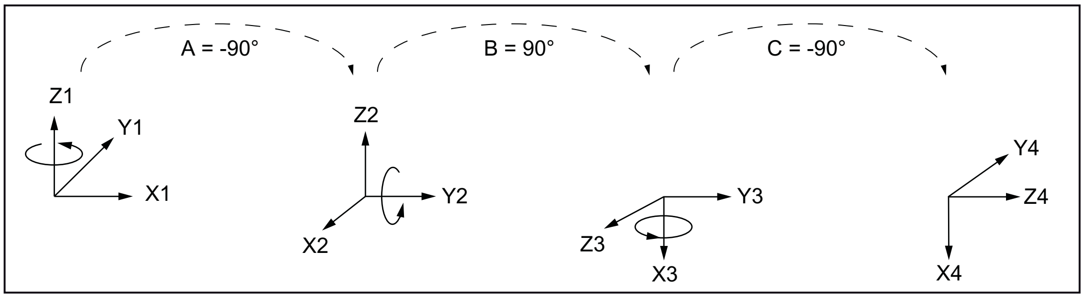


Figure 19-1 Example of rotation through RPY angles

19.3.3 Definition of a joint

Meaning

A sliding joint is implemented using a translatory axis, and a swivel joint, using a rotary axis.

The basic axis identifiers result from the arrangement and sequence of the individual joints. These are specified using letter codes (S, C, R, N).

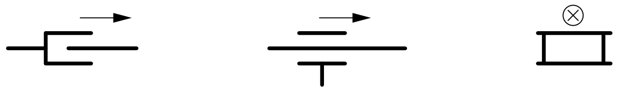
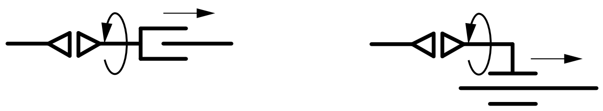
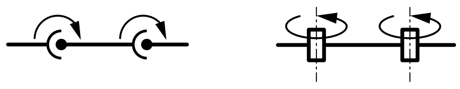
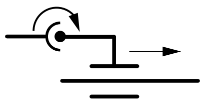
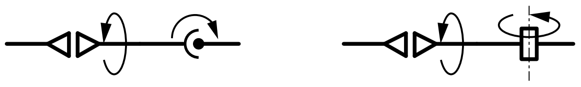
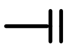
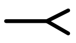
Identifier	Designation	Icons	
S	Sliding joint		
C	Sliding joint || swivel joint		
R	Swivel joint || swivel joint		
	Swivel joint ⊥ Sliding joint		
N	Swivel joint ⊥swivel joint		
FL	Flange for attaching the tool		→ Positive axis direction Positive axis direction: ⊗ - flowing toward the drawing plane ⊙ - flowing away from the drawing plane ⌒ Positive direction of rotation
T	Tool		|| Axis directions parallel to one another ⊥ Axis directions perpendicular to one another

Figure 19-2 Joint identifying letters

19.4 Configuration of a kinematic transformation

Meaning

In order to ensure that the kinematic transformation can convert the programmed values into axis motions, it must have access to some information about the mechanical construction of the machine. This information is stored in machine data:

- Axis assignments
- Geometry information

19.4.1 General machine data

MD24100 $MC_TRAFO_TYPE_1 (definition of transformation 1 in the channel)

The value 4100 must be entered in this data for the handling transformation package.

MD24110 $MC_TRAFO_AXES_IN_1 (axis assignment for transformation)

The axis assignment at the transformation input defines which transformation axis is mapped internally onto a channel axis. It is specified in machine data:

MD24110 $MC_TRAFO_AXES_IN_1

There is a predetermined axis sequence for the handling transformation package, The basic axes are entered in the first three components (index 0…2). The upper three components (index 3…5) are assigned to the manual axes.

- MD24110 $MC_TRAFO_AXES_IN_1[0] = 1 ; 1. Basic axis
- MD24110 $MC_TRAFO_AXES_IN_1[1] = 2 ; 2. Basic axis
- MD24110 $MC_TRAFO_AXES_IN_1[2] = 3 ; 3. Basic axis
- MD24110 $MC_TRAFO_AXES_IN_1[3] = 4 ; 1. Manual axis
- MD24110 $MC_TRAFO_AXES_IN_1[4] = 5 ; 2. Manual axis
- MD24110 $MC_TRAFO_AXES_IN_1[5] = 6 ; 3. Manual axis

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 (assignment between geometry axis and channel axis for transformation 1)

The translational degrees of freedom that are available for the transformation are entered via machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1

The 3 geometry axes normally correspond to Cartesian axis directions X, Y and Z.

In this case, the first three channel axis numbers must be transferred from MD24110 $MC_TRAFO_AXES_IN_1 to MD24120 $MC_TRAFO_GEO_AX_ASSIGN_TAB_1.

- MD24120 $MC_TRAFO_GEO_AX_ASSIGN_TAB_1[0] = 1
- MD24120 $MC_TRAFO_GEO_AX_ASSIGN_TAB_1[1] = 2
- MD24120 $MC_TRAFO_GEO_AX_ASSIGN_TAB_1[2] = 3

19.4.2 Parameterization using geometry data

Modular principle

The machine geometry is parameterized according to a type of modular principle. With this method, the machine is successively configured in geometry parameters from its base center point to the tool tip, thereby producing a closed kinematic loop. In this case, frames are used to describe the geometry. While the control is powering up, the configuration machine data is checked and alarms displayed when relevant. All axes of the mode group are tracked as alarm response. The alarms can be deleted using POWER ON.

As shown in the following diagram "Closed kinematic loop illustrated by the example of a robot", the kinematic transformation converts the tool operating point (tool coordinate system: X_{TO}, Y_{TO}, Z_{TO}), specified relative to the base coordinate system (BCS = robot coordinate system: X_{RO}, Y_{RO}, Z_{RO}), in machine axis values (MCS positions: A1, A2, A3, ...). The operating point (XWZ, YWZ, ZWZ) is specified in the part program in relation to the workpiece to be machined (workpiece coordinate system WCS: X_{WS}, Y_{WS}, Z_{WS}). The programmable frames make it possible to create an offset between the workpiece coordinate system (WCS) and the base coordinate system (BCS).

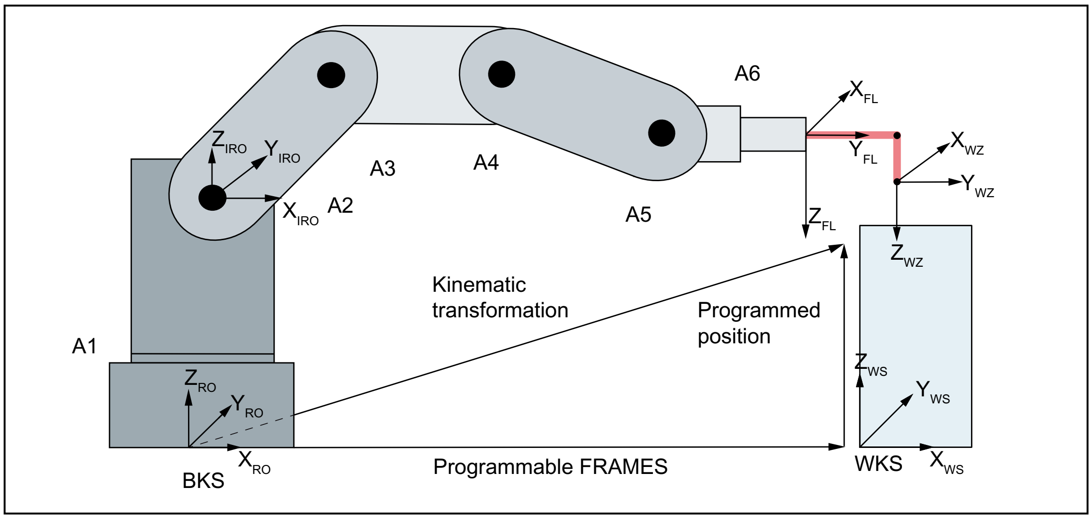


Figure 19-3 Closed kinematic loop illustrated by the example of a robot

Note

For more detailed information about coordinate systems, please see:

Reference:
Function Manual Basic Functions;

Chapter "K2, axes, coordinate systems, frames" > "Coordinate systems"

The following machine data is available for configuring kinematic transformations:

Frame between base center point and internal coordinate system

The frame T_IRO_RO links the base center point of the machine (BCS = RO) with the first internal coordinate system (IRO) determined by the transformation.

- MD62613 $MC_TRAFO6_TIRORO_RPY (frame between base center point and internal coordinate system (rotation component), n = 0...2)
- MD62612 $MC_TRAFO6_TIRORO_POS (frame between base center point and internal coordinate system (position component), n = 0...2)

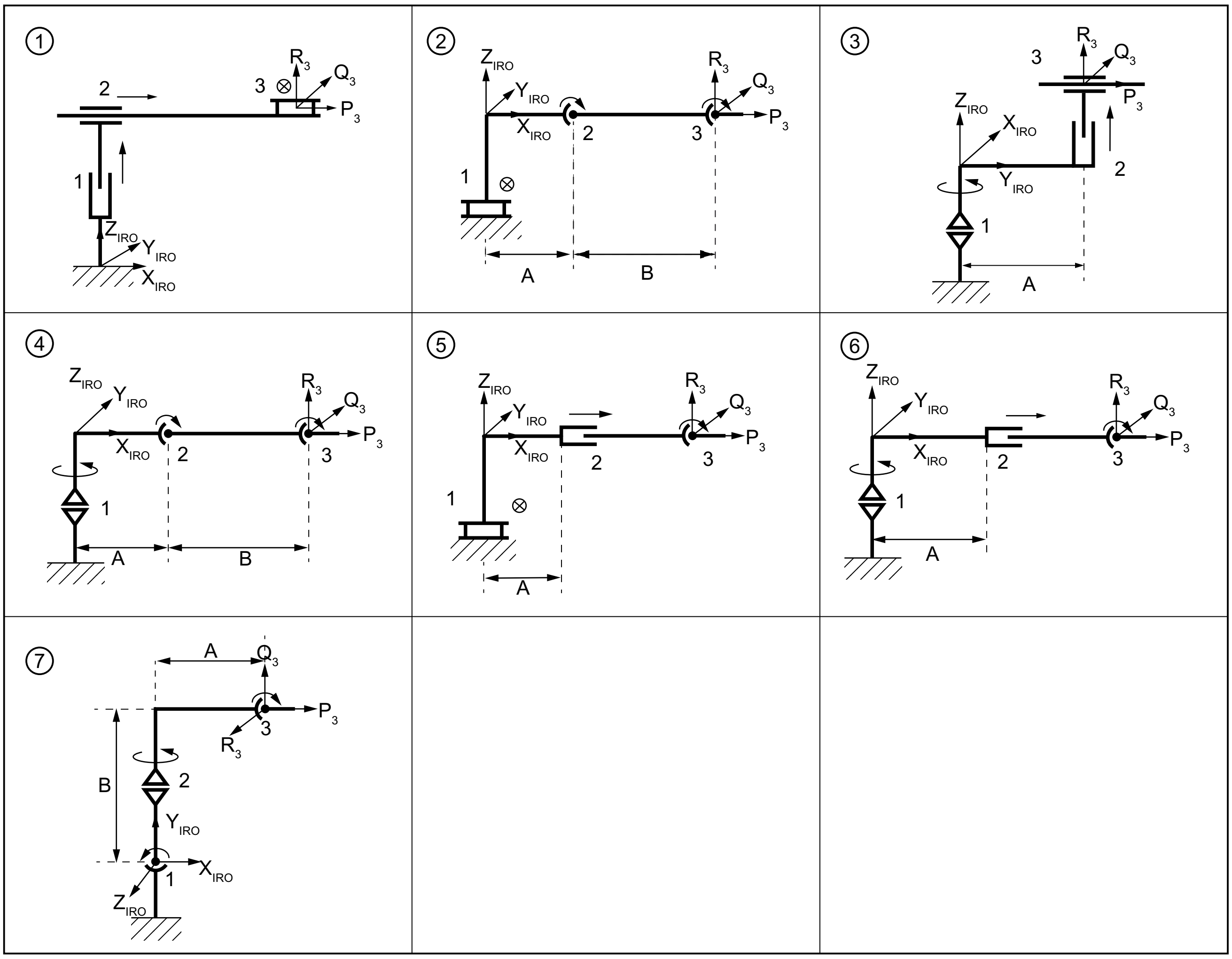
Basic axis arrangement

The type of basic axis arrangement is specified in machine data:

- MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)

The first three axes included in the transformation are generally referred to as the "basic axes". They must always be mutually parallel or perpendicular.

The handling transformation package contains the basic axis kinematics shown in the following diagram. Every basic axis arrangement has its own special identifier (see Chapter "Definition of a joint (Page 796)").



① SS: MD62603 = 1, gantry (3 linear axes, rectangular)

② CC: MD62603 = 2, SCARA (1 linear axis, 2 rotary axes (in parallel))

③ CS: MD62603 = 6, SCARA (2 linear axes, 1 rotary axis (spin axis))

④ NR: MD62603 = 3, articulated arm (3 rotary axes (2 axes in parallel))

⑤ SC: MD62603 = 4, SCARA (2 linear axes, 1 rotary axis (swivel axis))

⑥ RR: MD62603 = 5, articulated arm (1 linear axis, 2 rotary axes (vertical))

⑦ NN: MD62603 = 7, articulated arm (3 rotary axes)

Figure 19-4 Overview of basic axis configuration

Basic axis lengths A and B

The basic axis lengths A and B are specified with machine data:

- MD62607 $MC_TRAFO6_MAIN_LENGTH_AB (basic axis lengths A and B, n = 0...1)

As Fig. "Overview of basic axis configurations" illustrates, these are specially defined for each type of basic axis.

Position of the 4th axis

Whether the 4th axis is mounted parallel, anti-parallel or perpendicular to the last rotary basic axis is specified in machine data:

- MD62606 $MC_TRAFO6_A4PAR (axis 4 is parallel/anti-parallel to last basic axis)

Description of the wrist axis

Frame to attach the wrist axis

Frame T_X3_P3 links the last coordinate system of the basic axes with the first wrist axis coordinate system.

- MD62608 $MC_TRAFO6_TX3P3_POS (attachment of wrist axis (position component], n = 0...2)
- MD62609 $MC_TRAFO6_TX3P3_RPY (attachment of wrist axis (rotation component), n = 0...2)

Frame between wrist point and flange coordinate system

Frame T_FL_WP links the last hand wrist axis coordinate system with the flange coordinate system.

- MD62610 $MC_TRAFO6_TFLWP_POS (frame between wrist point and flange coordinate system (position component), n = 0...2)
- MD62611 $MC_TRAFO6_TFLWP_RPY (frame between wrist point and flange coordinate system (rotation component), n = 0...2)

This data is explained in more detail in the following sections.

Description of the wrist axis type

These parameters describe the wrist axis type.

- MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)
  The term wrist axes generally refers to axes four to six.

Wrist axes included in every transformation

The fourth axis and all further axes are generally referred to as "wrist axes". The handling transformation package can only identify wrist axes with rotary axes. The wrist axis identifier for three-axis wrists is entered in machine data:

- MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

When there are fewer than three wrist axes, the identifier for the inclined wrist axis or a central wrist axis is entered in the machine data:

- MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

The current software only supports the central and inclined wrist axis types.

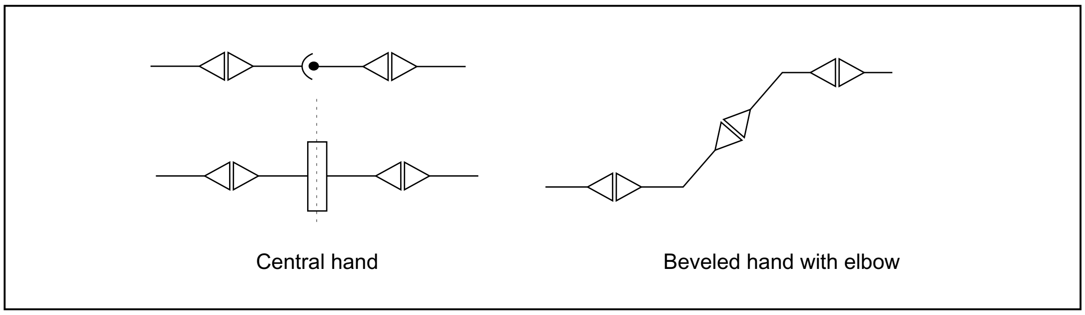


Figure 19-5 Overview of wrist axis configuration

Parameterization of wrist axes

With the following machine data, using a special frame type, the geometry of the wrist axis and/or the position of the coordinate system in the wrist axis with respect to one another is defined.

- MD62614 $MC_TRAFO6_DHPAR4_5A (parameter A for configuring the wrist axis, n = 0...1)
- MD62615 $MC_TRAFO6_DHPAR4_5D (parameter D for configuring the wrist axis, n = 0...1)
- MD62616 $MC_TRAFO6_DHPAR4_5ALPHA. (Parameter ALPHA for configuring the wrist axis, n = 0...1)

For this purpose, the machine data corresponds to certain frame components (see Section "Definition of positions and orientations using frames (Page 795)"):

- X component: MD62614 $MC_TRAFO6_DHPAR4_5A ≙ _POS[**0**] (x component)
- Z component: MD62615 $MC_TRAFO6_DHPAR4_5D ≙ _POS[**2**] (z component)
- C angle: MD62616 $MC_TRAFO6_DHPAR4_5ALPHA ≙ _RPY[**2**] (C angle)

The other components of the frame are zero.

Central wrist axis

On a central wrist axis, all wrist axes intersect at one point. All parameters must be set as shown in Table "Configuring data for a central wrist axis".

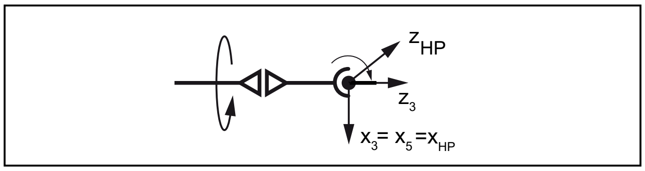


Figure 19-6 Central wrist axis, 5-axis machine

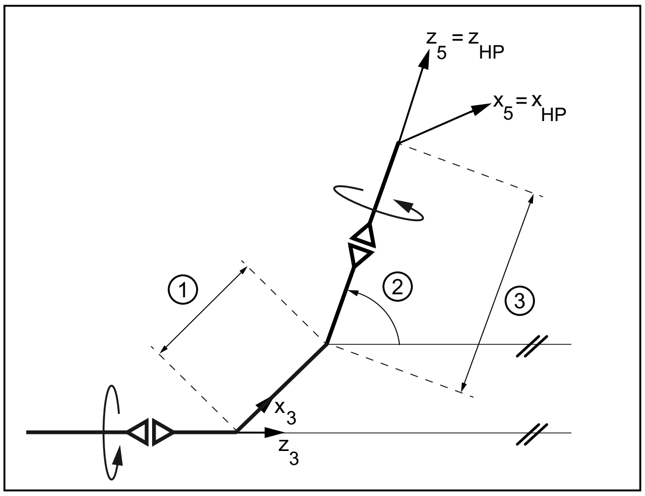
Table 19-1 Configuring data for a central wrist axis

Machine data	Value
MD62604 $MC_TRAFO6_WRIST_AXES	2
MD62614 $MC_TRAFO6_DHPAR4_5A	[0.0, 0.0]

Machine data	Value
MD62615 $MC_TRAFO6_DHPAR4_5D	[0.0, 0.0]
MD62616 $MC_TRAFO6_DHPAR4_5ALPHA	[-90.0, 90.0]

Inclined wrist axis

The inclined wrist axis differs from the central wrist axis in two respects, i.e. the axes do not intersect nor are they mutually perpendicular. Parameters a_4, d_5, and a_4 are available for this type of hand, as shown in Table "Configuring data for a central wrist axis".



① MD62614 $MC_TRAFO6_DHPAR4_5A[0]
② MD62616 $MC_TRAFO6_DHPAR4_5ALPHA[0]
③ MD62615 $MC_TRAFO6_DHPAR4_5D[1]

Figure 19-7 Inclined wrist axis, 5-axis machine

Table 19-2 Configuring data for an inclined wrist axis (5-axis

Machine data	Value
MD62604 $MC_TRAFO6_WRIST_AXES	6
MD62614 $MC_TRAFO6_DHPAR4_5A	[a4, 0.0]
MD62615 $MC_TRAFO6_DHPAR4_5D	[0.0, d5]
MD62616 $MC_TRAFO6_DHPAR4_5ALPHA	[α4, 0.0]

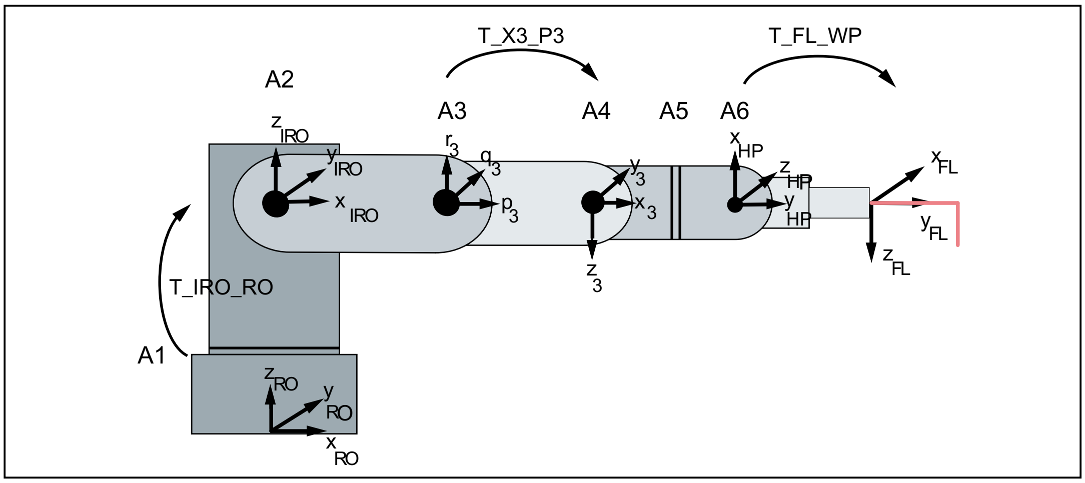


Figure 19-8 Link frames

Frame: T_IRO_RO

Frame T_IRO_RO provides the link between the base center point coordinate system (RO) defined by the user and the internal robot coordinate system (IRO). The internal robot coordinate system is predefined in the handling transformation package for each basic axis type and included in the kinematic diagrams for the basic axis arrangements. The base center point system is in the Cartesian zero point of the machine, corresponding to the basic coordinate system. If no `FRAMES` are programmed, the basic coordinate system equals the workpiece coordinate system.

Note

For more detailed information about `FRAMES`, please see:

References:
Programming Manual Fundamentals

Frame T_IRO_RO is not subject to any restrictions for 5-axis kinematics.

The following restrictions apply in relation to 4-axis kinematics:

- The first rotary axis must always be parallel/anti-parallel to one of the coordinate axes of the base center point coordinate system (RO).
- No further restrictions apply to type SS basic axes.
- In the case of type CC, CS or SC basic axes, no further restrictions apply provided that the 4th axis is parallel to the last rotary basic axis.
- With respect to all other basic axes, and basic axes of type CC, CS or SC if the 4th axis is perpendicular to the last rotary basic axis, the Z axis of RO must be parallel to the Z axis of IRO.

Frame: T_X3_P3

Frame T_X3_P3 describes the method used to attach the wrist axis to the basic axes. Frame T_X3_P3 is used to link the coordinate system of the last basic axis (p3_q3_r3 coordinate system) with the coordinate system of the first wrist axis (x3_y3_z3 coordinate system). The p3_q3_r3 coordinate system is included in the kinematic diagrams for the basic axis arrangements.

The z3 axis is always on the 4th axis.

Depending on the number of axes to be included in the transformation, frame T_X3_P3 is subject to certain restrictions relating to the wrist and basic axes:

- For 5-axis kinematics, frame T_X3_P3 can be freely selected in the following cases:
  - If the basic axes are of the SS type.
  - If the basic axes are type CC, CS, or SC, the transformation must either include a central wrist axis or the 4th axis must be positioned parallel to the last rotary basic axis.
  - If the basic axes are type NR or RR, the transformation must either include a central wrist axis or the 4th axis must be positioned in parallel to the last basic rotary axis and an X flange must intersect the 5th axis.
  - If the basic axes are of the NN type, the transformation must include a central wrist axis.
- For 4-axes, ensure that the Z3 axis is always parallel/anti-parallel or perpendicular to the last basic axis.

Frame: T_FL_WP

Frame T_FL_WP links the flange with the last internal coordinate system (wrist point coordinate system) preset by the handling transformation package.

For kinematics with less than 6 axes, this frame is subject to certain restrictions. These restrictions are explained by the relevant kinematics.

Other configuring data

Number of transformed axes

The number of axes included in the transformation is defined in machine data:

- MD62605 $MC_TRAFO6_NUM_AXES (number of transformed axes)

The number of transformed axes can currently lie between two and six axes.

Changing the axis sequence

Rearrangement of axes: MD62620

Note

With certain types of kinematics, it is possible to transpose axes without changing the behavior of the kinematic transformation. Machine data:

MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)

The axes on the machine are numbered consecutively from 1 to 6 and must be entered in the internal sequence in machine data:

MD62620 $MC_TRAFO6_AXIS_SEQ[0] ...[4]

All other axis-specific machine data refer to the sequence of axes on the machine.

Table 19-3 Changing the axis sequence

Basic axis kinematics	Options for changing axis sequence
SS, CC	Any
SC	1 and 2
CS	2 and 3

Example 1

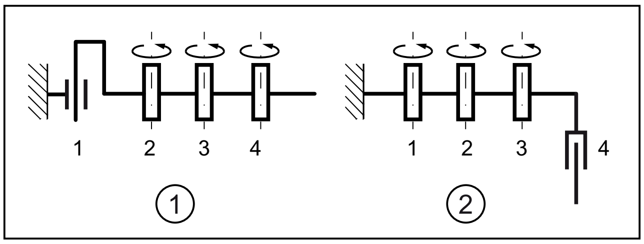
There are two kinematics, such as those shown in Fig. "Rearrangement of axes 1". Kinematic 1 is directly included in the handling transformation package. It corresponds to a CC kinematic with a wrist axis parallel to the last basic rotary axis.

Kinematic 2 is the same as kinematic 1 inasmuch as it is irrelevant for the resulting robot movement whether the translational axis is axis 1 or axis 4. In this instance, the data for kinematic 2 must be entered as follows in machine data:

- MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)

The input is as follows:

- MD62620 $MC_TRAFO6_AXIS_SEQ[0] = 4
- MD62620 $MC_TRAFO6_AXIS_SEQ[1] = 1
- MD62620 $MC_TRAFO6_AXIS_SEQ[2] = 2
- MD62620 $MC_TRAFO6_AXIS_SEQ[3] = 3



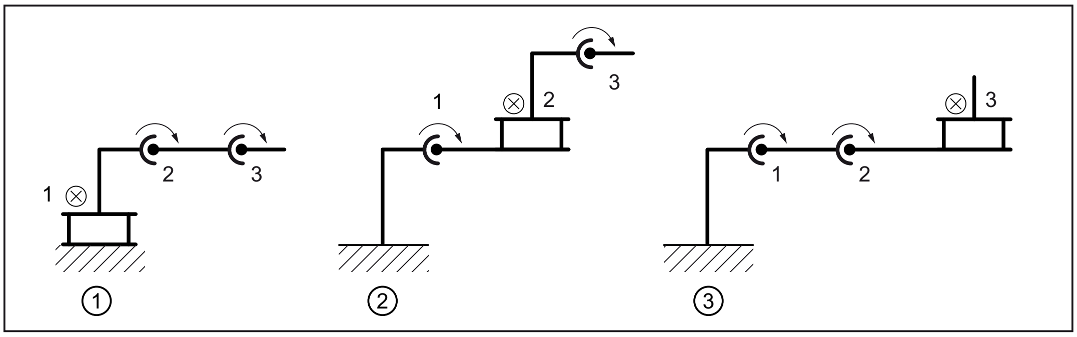
① Kinematics 1

② Kinematics 2

Figure 19-9 Rearrangement of axes 1

Example 2

This example involves a SCARA kinematic transformation as illustrated in Fig. "Rearrangement of axes 2", in which the axes can be freely transposed. Kinematic 1 is directly included in the handling transformation package. It corresponds to a CC kinematic. As regards the transposition of axes, it is irrelevant how many wrist axes are involved in the transformation.



① Kinematics 1: Axis sequence 1 2 3

② Kinematics 2: Axis sequence 2 1 3

③ Kinematics 3: Axis sequence 2 3 1

Figure 19-10 Rearrangement of axes 2

Changing the directions of axes

A rotational or offset direction is preset for each axis in the handling transformation package. This direction is not necessarily the same as the corresponding direction on the machine. The following machine data must be set to **-1** for the relevant axis if the direction is to be inverted, or otherwise to **+1**:

- MD62618 $MC_TRAFO6_AXES_DIR[ ] (matching of physical and mathematical directions of rotation [axis no.]: 0...5)

Adapting the zero points of the axes

Offset between mathematical and mechanical zero points

The mathematical zero points of axes are preset in the handling transformation package. However, the mathematical zero point does not always correspond to the mechanical zero point (calibration point) of axes. In order for the zero point of each axis to fit one another, the difference between the mathematical zero point and the alignment point for each axis must be entered in the following machine data:

- MD62617 $MC_TRAFO6_MAMES[ ] (offset between mathematical and mechanical zero points [axis no.]: 0...5)

The deviation to be entered corresponds to the difference between the mechanical zero point and the mathematically positive direction of rotation of the axis.

Example

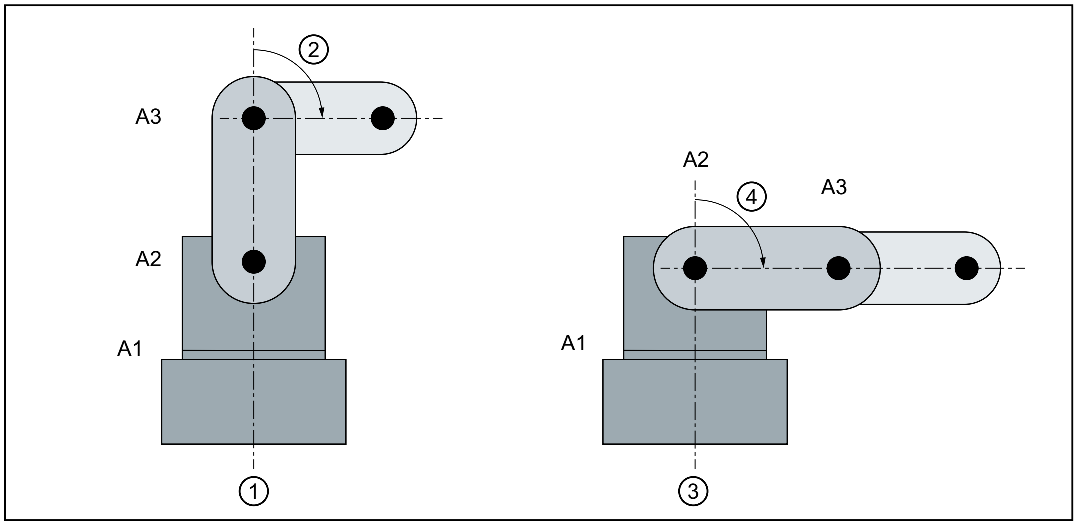
Jointed arm kinematics are shown in the following diagram.

In the mathematical zero position ②, axis 2 (A2) has a value of 90º. This value should be entered into the machine data for axis 2 (index 1):

- MD62617 $MC_TRAFO6_MAMES[**1**] = 90

Axis 3 (A3) is counted relative to the previous axis 2 (A2) in the kinematic chain, and in the mechanical zero position ① has the value: -90º:

- MD62617 $MC_TRAFO6_MAMES[**2**] = -90



① mechanical zero position (adjusted position):
② MD62617 $MC_TRAFO6_MAMES[**2**] = -90
③ mathematical zero position
④ MD62617 $MC_ TRAFO6_MAMES[**1**] = 90

Figure 19-11 Matching mathematical and mechanical zero points

Axis type for transformation

Which axis type is handled is specified in machine data:

- MD62601 $MC_TRAFO6_AXES_TYPE (axis type for transformation [axis no.]: 0...5)

The transformation package distinguishes between the following axis types:

- Linear axis: MD62601 = 1
- Rotary axis: MD62601 = 3

Velocity and acceleration values for override control that limits the dynamic response

⚠ CAUTION

Unnecessarily high limiting default values

Since software release **RCTRA 07.05.00**, RCTRA transformation for limiting the Cartesian velocities and acceleration levels, takes into account machine data **MD62629** to **MD62632**.

In order to avoid unnecessarily limiting the velocities and acceleration levels, this machine data must be adapted to the machine capabilities. Frequently, unnecessarily high limiting default values are set here, as - up until now - this data has not been used.

For traversing axes with active RCTRA transformation and override controls that limit the dynamic performance, dedicated substitute limit values have been introduced for velocities and acceleration levels for Cartesian motion components.

The override controller within the transformation is only active if motion is interpolated, whose dynamic limits are not secured - or not completely secured - through motion planning in the Look Ahead function of the control.

The velocity controller is active in the following cases:

- When traversing in the JOG mode with transformation
- For overlaid motion in realtime, e.g. via system variable $AA_OFF[X]
- For velocity limits as a result of Safety Integrated (SG level active)

Cartesian velocity

The velocities for individual translatory motion directions when traversing with override controller are specified using machine data:

- MD62629 $MC_TRAFO6_VELCP[i] (Cartesian velocity [No.]: 0...2)
  - Index i = 0 : X component of basic system
  - Index i = 1 : Y component of basic system
  - Index i = 2 : Z component of basic system

Cartesian acceleration rates

The acceleration levels for individual translatory motion directions when traversing with override controller are specified using machine data:

- MD62630 $MC_TRAFO6_ACCCP[i] (Cartesian acceleration levels [no.]: 0...2)
  - Index i = 0 : X component of basic system
  - Index i = 1 : Y component of basic system
  - Index i = 2 : Z component of basic system

Orientation angle velocities

The velocities for individual orientation directions when traversing with override controller are specified using machine data:

- MD62631 $MC_TRAFO6_VELORI[i] (orientation angle velocities [no.]: 0...2)
  - Index i = 0 : A angle
  - Index i = 1 : B angle
  - Index i = 2 : C angle

Orientation angle acceleration rates

The acceleration levels for individual orientation directions when traversing with override controller are specified using machine data:

- MD62632 $MC_TRAFO6_ACCORI[i] (orientation angle acceleration levels [no.]: 0...2)
  - Index i = 0 : A angle
  - Index i = 1 : B angle
  - Index i = 2 : C angle

Reduction factor for the velocity controller

A reduction factor for the velocity controller in the JOG mode can be specified with the machine data:

- MD62633 $MC_ROBX_DYN_LIM_REDUCE (reduction factor for the velocity controller)

Time constant for the velocity controller

A time constant for the PT1 filter of the velocity controller can be specified with the machine data:

- MD62634 $MC_ROBX_VEL_FILTER_TIME (time constant for the velocity controller)

19.5 Descriptions of kinematics

The following descriptions of kinematics for transformations involving 2 to 5 axes explain the general configuring procedure first before describing how the machine data need to be configured, using a configuring example for each kinematic type. These examples do not include all possible lengths and offsets. The direction data refer to the positive directions of traversal and rotation for the transformation. The axis positions correspond to their zero position for the relevant transformation.

19.5.1 3-axis kinematics

3-axis kinematics normally possess three translational degrees of freedom, but do not have a rotational degree of freedom for orientation. As a consequence, 3-axis kinematics only have basic axes.

Configuration

The procedure for configuring a 3-axis kinematic is as follows:

1. Enter "Standard" kinematic category in machine data:
   MD62600 $MC_TRAFO6_KINCLASS (kinematic category)
2. Set the number of axes for transformation in machine data:
   MD62605 $MC_TRAFO6_NUM_AXES = 3 (number of transformed axes)
3. Compare the basic axes with the basic axes contained in the handling transformation package. → Enter the basic axis identifier in machine data:
   MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)
4. If the axis sequence is not the same as the normal axis sequence, it must be corrected in machine data:
   MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)
5. As identifier for the wrist axes, the following machine data must be set to 1 (no wrist axis):
   MD62604 $MC_TRAFO6_WRIST_AXES = 1 (wrist axis identifier)
6. Enter the axis types for the transformation in machine data:
   MD62601 $MC_TRAFO6_AXES_TYPE (axis type for transformation)
7. Compare the directions of rotation of axes with the directions defined in the handling transformation package and correct in machine data:
   MD62618 $MC_TRAFO6_AXES_DIR (matching of physical and mathematical directions of rotation)
8. Enter the mechanical zero offset in machine data:
   MD62617 $MC_TRAFO6_MAMES (offset between mathematical and mechanical zero points)
9. Enter the basic axis lengths in machine data:
   MD62607 $MC_TRAFO6_MAIN_LENGTH_AB (basic axis lengths A and B)

10. Define frame T_IRO_RO and enter the offset in machine data:
    MD62612 $MC_TRAFO6_TIRORO_POS (frame between base center point and internal system (position component))
    Enter the rotation in machine data:
    MD62613 $MC_TRAFO6_TIRORO_RPY (frame between base center point and internal system (rotation component))
11. Determine the flange coordinate system. For this purpose, the p3_q3_r3 coordinate system must be regarded as the initial system. The offset is entered in machine data:
    MD62610 $MC_TRAFO6_TFLWP_POS (frame between wrist point and flange (position component))
    The rotation is entered in machine data:
    MD62611 $MC_TRAFO6_TFLWP_RPY (frame between wrist point and flange (rotation component))

SCARA kinematics

SCARA kinematics are characterized by the fact that they possess both translational and rotary axes. The kinematics have different names depending on how the basic axes are arranged with respect to one another, e.g.: CC, SC, CS kinematics (see Chapter "Definition of a joint (Page 796)").

3-axis CC kinematics

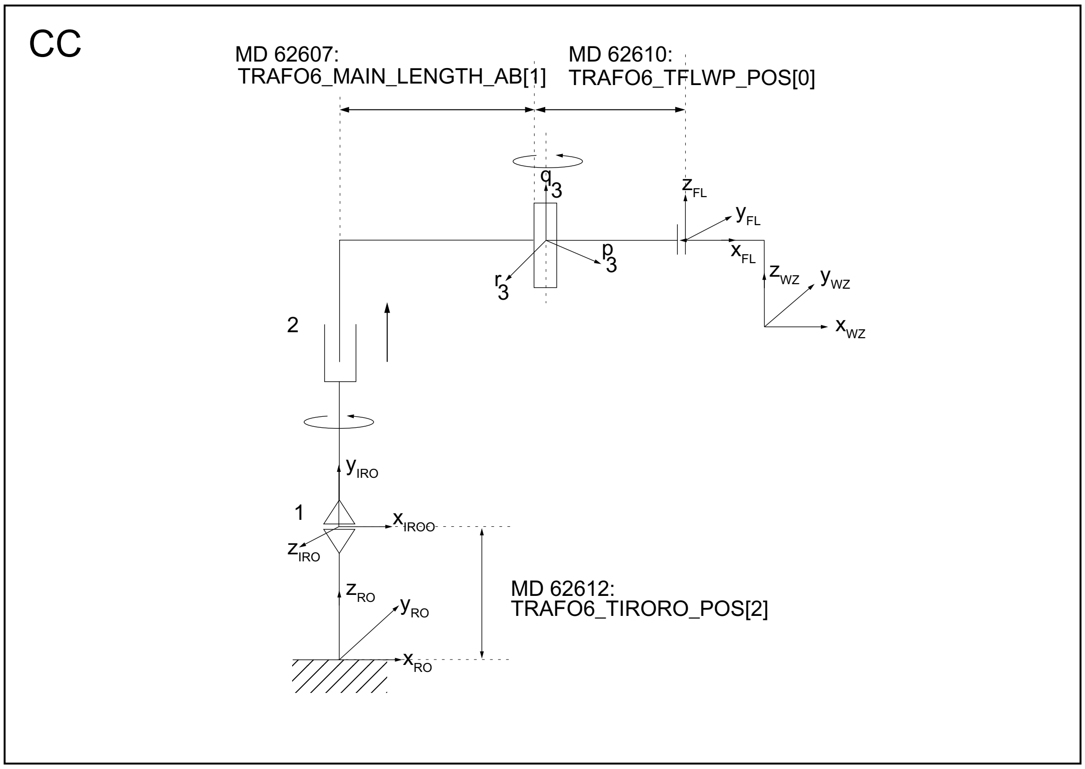


Figure 19-12 3-axis CC kinematics

Table 19-4 Configuration data for 3-axis CC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 1, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[2, 1, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[0.0, 300.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 90.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, -90.0]

3-axis SC kinematics

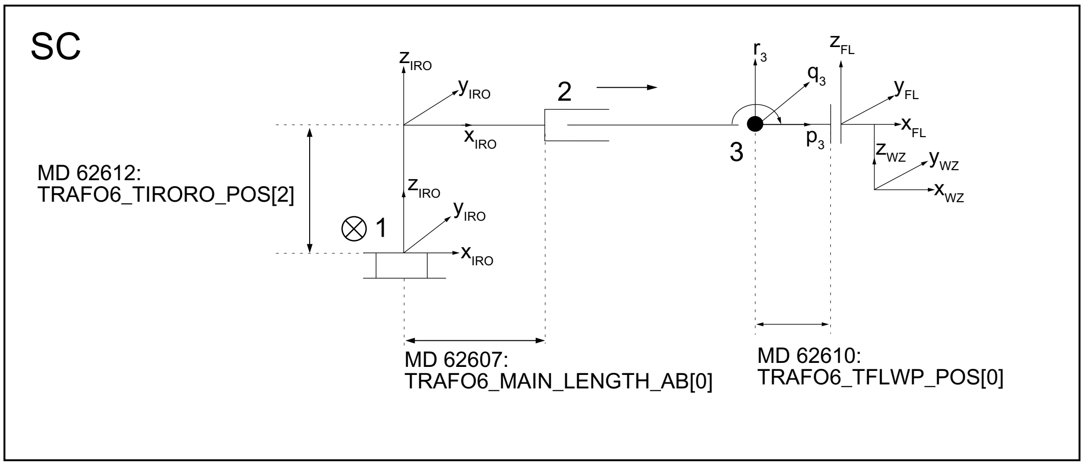


Figure 19-13 3-axis SC kinematics

Table 19-5 Configuration data for 3-axis SC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	4
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[1, 1, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]

Machine data	Value
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[500.0, 0.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[300.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

3-axis CS kinematic

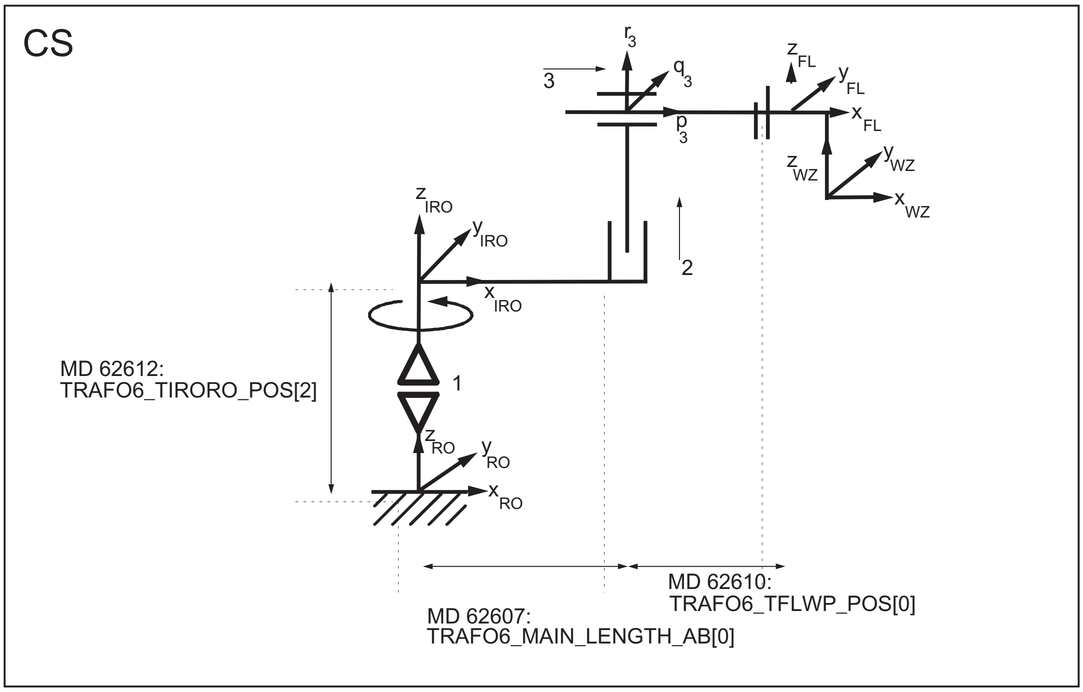


Figure 19-14 3-axis CS kinematic

Table 19-6 Configuration data for 3-axis CS kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	6
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 1, 1, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[500.0, 0.0]

Machine data	Value
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[300.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

Articulated-arm kinematics

3-axis NR kinematics

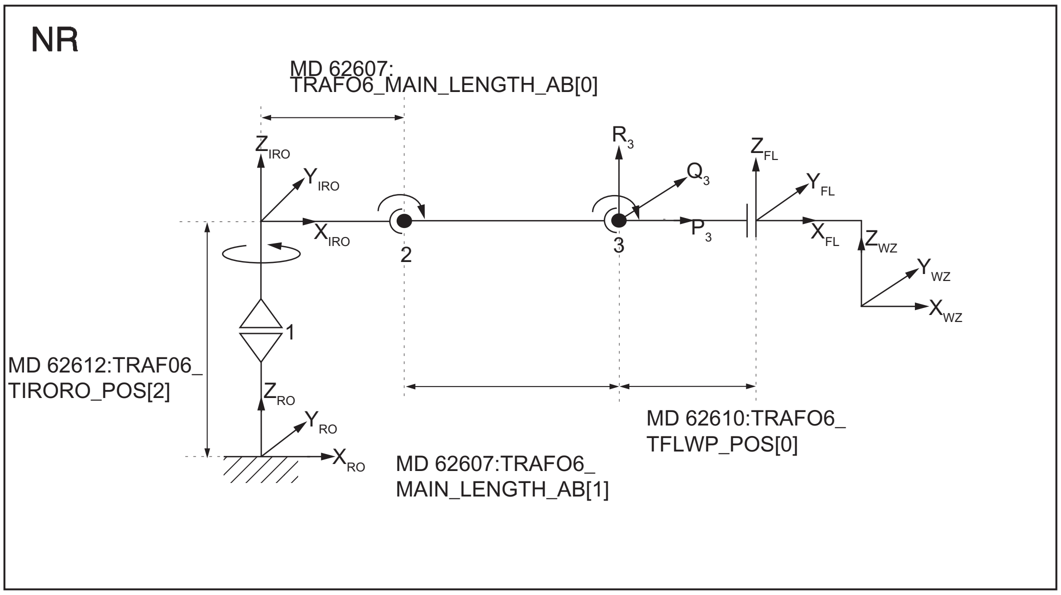


Figure 19-15 3-axis NR kinematics

Table 19-7 Configuration data 3-axis NR kinematic

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	3
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[300.0, 500.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]

Machine data	Value
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[300.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

3-axis RR kinematics

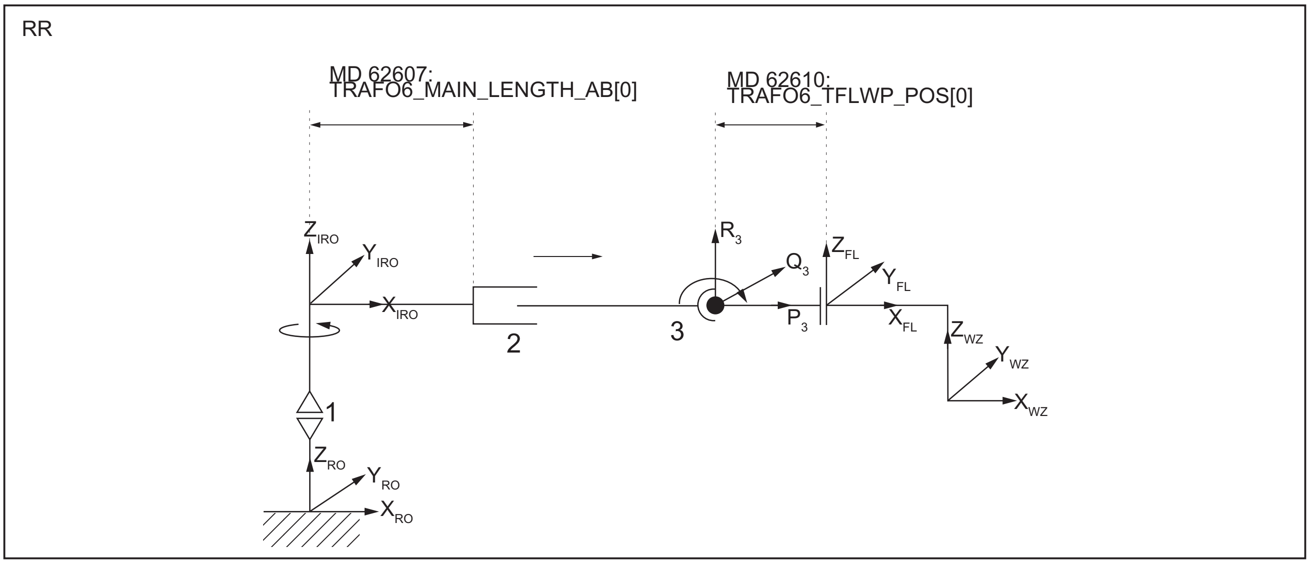


Figure 19-16 3-axis RR kinematics

Table 19-8 Configuration data for 3-axis RR kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	5
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 1, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[300.0, 0.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 300.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

3-axis NN kinematics

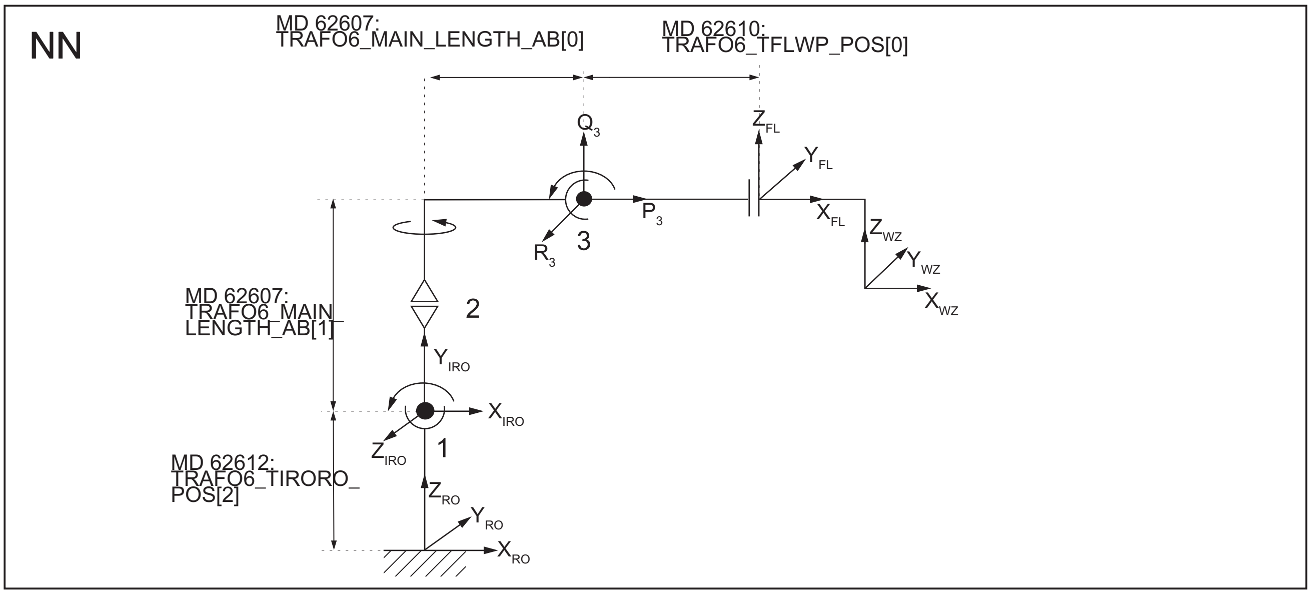

NN
MD 62607: TRAFO6_MAIN_LENGTH_AB[0]
MD 62610: TRAFO6_TFLWP_POS[0]
Q3
ZFL
YFL
XFL
P3
R3
3
ZWZ
YWZ
XWZ
MD 62607: TRAFO6_MAIN_LENGTH_AB[1]
2
YIRO
XIRO
ZIRO
1
MD 62612: TRAFO6_TIRORO_POS[2]
ZRO
YRO
XRO

Figure 19-17 3-axis NN kinematics

Table 19-9 Configuration data for 3-axis NN kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	7
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[300.0, 500.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 300.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 90.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPYFehler! Bookmark not defined.	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[400.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, -90.0]

19.5.2 4-axis kinematics

4-axis kinematics normally have 3 translational degrees of freedom and one rotational degree of freedom for orientation.

Restrictions

The following restrictions apply to 4-axis kinematics:

The frame T_FL_WP is subject to the following condition:

- MD62611 $MC_TRAFO6_TFLWP_RPY = [0.0, 90.0, 0.0] (frame between wrist point and flange (rotation component))
- X flange and X tool must be parallel to the 4th axis.
- Two successive basic axes must be parallel or orthogonal.
- The 4th axis must only be mounted in a parallel or orthogonal way to the last basic axis.

Configuration

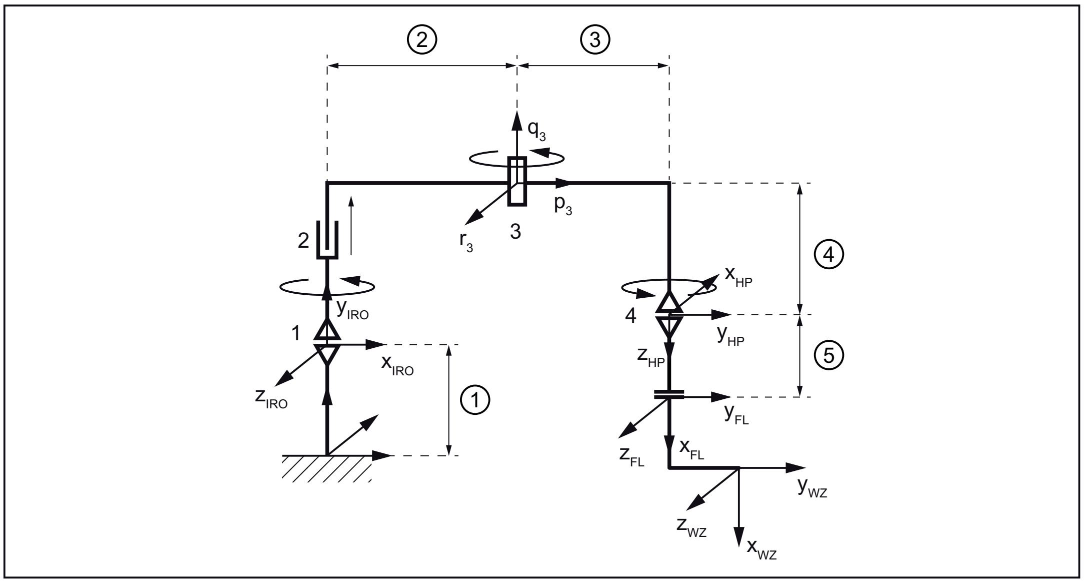
The procedure for configuring a 4-axis kinematic is as follows:

1. Enter "Standard" kinematic category in the machine data:
   MD62600 $MC_TRAFO6_KINCLASS (kinematic category)
2. Set the number of axes for transformation in the machine data:
   MD62605 $MC_TRAFO6_NUM_AXES=4 (number of transformed axes)
3. Compare the basic axes with the basic axes contained in the handling transformation package.
   – Enter the basic axis identifier in machine data:
     MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)
4. If the axis sequence is not the same as the normal axis sequence, it must be corrected in machine data:
   MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)
5. As identifier for the wrist axes, the following machine data must be set (no wrist axis):
   MD62604 $MC_TRAFO6_WRIST_AXES = 1 (wrist axis identifier)
6. Whether axis 4 runs parallel/anti-parallel to the last rotary basic axis is entered into the machine data:
   MD62606 $MC_TRAFO6_A4PAR (axis 4 is parallel/anti-parallel to last basic axis)
7. Enter the axis types for the transformation in machine data:
   MD62601 $MC_TRAFO6_AXES_TYPE (axis type for transformation)
8. Compare the directions of rotation of axes with the directions defined in the handling transformation package and correct in the machine data:
   MD62618 $MC_TRAFO6_AXES_DIR (matching of physical and mathematical directions of rotation)
9. Enter the mechanical zero offset in the machine data:
   MD62617 $MC_TRAFO6_MAMES (offset between mathematical and mechanical zero points)
10. Enter the basic axis lengths in the machine data:
    MD62607 $MC_TRAFO6_MAIN_LENGTH_AB (basic axis lengths A and B)

11. Define frame T_IRO_RO and enter the offset in the machine data:
    MD62612 $MC_TRAFO6_TIRORO_POS (frame between base center point and internal system (position component))
    Enter the rotation in the machine data:
    MD62613 $MC_TRAFO6_TIRORO_RPY (frame between base center point and internal system (rotation component))
12. Specification of frame T_X3_P3 to attach wrist axis. For this purpose, the p3_q3_r3 coordinate system must be regarded as the initial system. The offset is entered in the machine data:
    MD62608 $MC_TRAFO6_TX3P3_POS (attachment of wrist axis (position component))
    The rotation is entered in the machine data:
    MD62609 $MC_TRAFO6_TX3P3_RPY (attachment of wrist axis (rotation component))
    is entered.
13. Determine the flange coordinate system. For this purpose, the wrist point coordinate system must be regarded as the initial system. The offset is entered in the machine data:
    MD62610 $MC_TRAFO6_TFLWP_POS (frame between wrist point and flange (position component))
    The rotation is entered in the machine data:
    MD62611 $MC_TRAFO6_TFLWP_RPY (frame between wrist point and flange (rotation component))

SCARA kinematics

4-axis CC kinematics



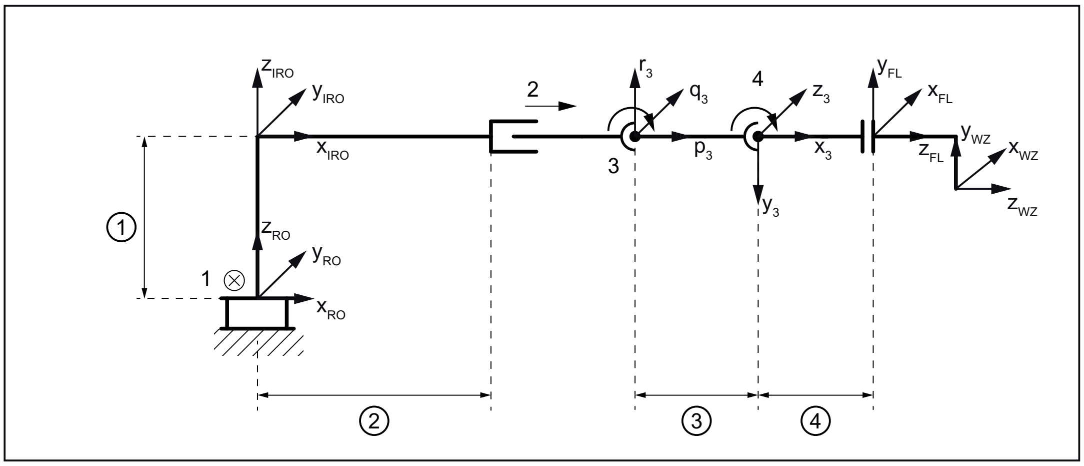
① MD62612 $MC_TRAFO6_TIRORO_POS[2]
② MD62607 $MC_TRAFO6_MAIN_LENGTH_AB[1]
③ MD62608 $MC_TRAFO6_TX3P3_POS[0]
④ MD62608 $MC_TRAFO6_TX3P3_POS[1]
⑤ MD62610 $MC_TRAFO6_TFLWP_POS[0]

Figure 19-18 4-axis CC kinematics

Table 19-10 Configuration data for 4-axis CC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	4
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62606 $MC_TRAFO6_A4PAR	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 1, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[2, 1, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR)	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[0.0, 300.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 90.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[300.0, 0.0, -200.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[-90.0, 90.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, 0.0, 200.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 0.0]

4-axis SC kinematics



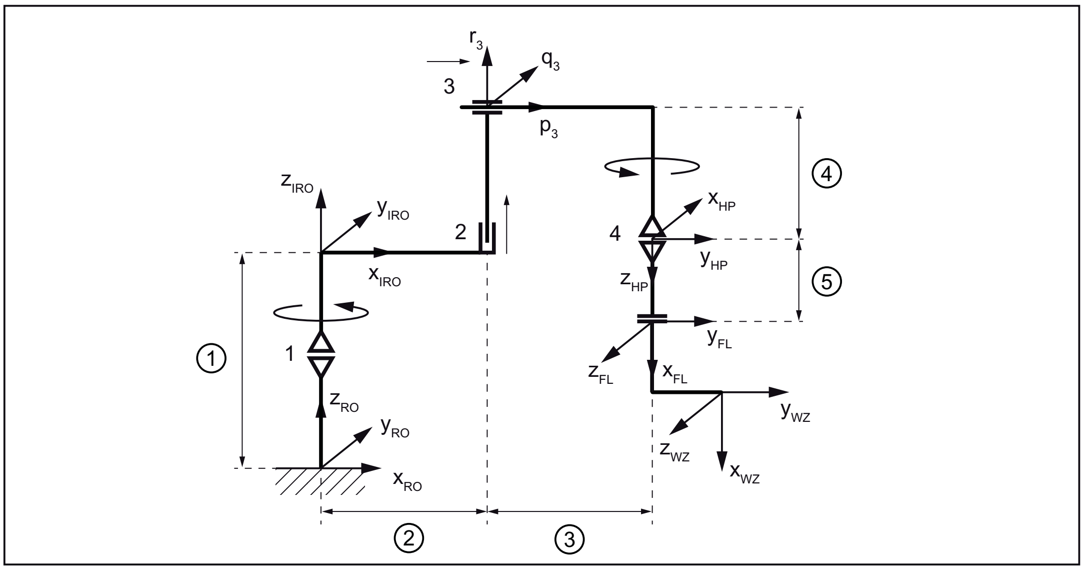
① MD62612 $MC_TRAFO6_TIRORO_POS[2]
② MD62607 $MC_TRAFO6_MAIN_LENGTH_AB[0]
③ MD62608 $MC_TRAFO6_TX3P3_POS[0]
④ MD62610 $MC_TRAFO6_TFLWP_POS[0]

Figure 19-19 4-axis SC kinematics

Table 19-11 Configuration data for 4-axis SC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	4
MD62603 $MC_TRAFO6_MAIN_AXES	4
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62606 $MC_TRAFO6_A4PAR	1
MD62601 $MC_TRAFO6_AXES_TYPE	[1, 1, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[300.0, 0.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 300.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[200.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, -90.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 180.0]

4-axis CS kinematic



① MD62612 $MC_TRAFO6_TIRORO_POS[2]

② MD62607 $MC_TRAFO6_MAIN_LENGTH_AB[0]

③ MD62608 $MC_TRAFO6_TX3P3_POS[0]

④ MD62608 $MC_TRAFO6_TX3P3_POS[2]

⑤ MD62610 $MC_TRAFO6_TFLWP_POS[2]

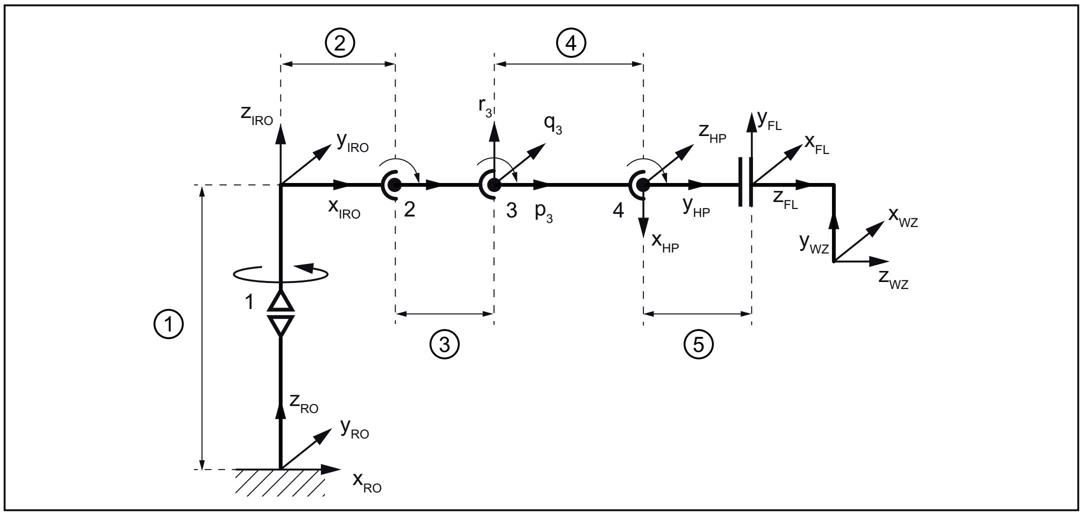
Figure 19-20 4-axis CS kinematic

Table 19-12 Configuration data for 4-axis CS kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	4
MD62603 $MC_TRAFO6_MAIN_AXES	6
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62606 $MC_TRAFO6_A4PAR	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 1, 1, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[400.0, 0.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 400.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[500.0, 0.0, -200.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[90.0, 0.0, 180.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, 0.0, 200.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 0.0]

Articulated-arm kinematics

4-axis NR kinematics



①	MD62612 $MC_TRAFO6_TIRORO_POS[2]
②	MD62607 $MC_TRAFO6_MAIN_LENGTH_AB[0]
③	MD62607 $MC_TRAFO6_MAIN_LENGTH_AB[1]
④	MD62608 $MC_TRAFO6_TX3P3_POS[0]
⑤	MD62610 $MC_TRAFO6_TFLWP_POS[0]

Figure 19-21 4-axis NR kinematics

Table 19-13 Configuration data 4-axis NR kinematic

Machine data	Value
MD62600 $MC_TRAFO6_ KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	4
MD62603 $MC_TRAFO6_MAIN_AXES	3
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62606 $MC_TRAFO6_A4PAR	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[300.0, 300.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[300.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, -90.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 180.0]

See also

Definition of a joint (Page 796)

19.5.3 5-axis kinematics

5-axis kinematics usually imply 3 degrees of freedom for translation and 2 for orientation.

Restrictions

The following restrictions apply to 5-axis kinematics:

1. There are restrictions for the flange coordinate system because the X flange axis must intersect the 5th axis, nevertheless, it must not be parallel to it.
2. The frame T_FL_WP is subject to the following condition as far as 5-axis articulated-arm kinematics are concerned:
   – MD62610 $MC_TRAFO6_TFLWP_POS = [0.0, 0.0, Z] (frame between wrist point and flange (position component))
   – MD62611 $MC_TRAFO6_TFLWP_RPY = [A, 0.0, 0.0] (frame between wrist point and flange (rotation component))
3. There are restrictions for the tool as far as 5-axis articulated-arm kinematics are concerned:
   – 4. Axis parallel to the 3rd axis: 2-dimensional tool is possible [X, 0.0, Z]
   – 4. Axis perpendicular to the 3rd axis: Only 1-dimensional tool is possible [X, 0.0, 0.0]

4. There are restrictions for the tool as far as 5-axis Scara kinematics are concerned:
   – 4. Axis perpendicular to the 3rd axis: 1-dimensional tool is possible [X, 0.0, 0.0]
5. Two successive basic axes must be parallel or orthogonal.
6. The 4th axis must only be mounted in a parallel or orthogonal way to the last basic axis.

Configuration

The procedure for configuring a 5-axis kinematic is as follows:

1. Enter "Standard" kinematic category in machine data:
   MD62600 $MC_TRAFO6_KINCLASS (kinematic category)
2. Set the number of axes for transformation in the machine data:
   MD62605 $MC_TRAFO6_NUM_AXES = 5 (number of transformed axes)
3. Compare the basic axes with the basic axes contained in the handling transformation package.
   – Enter the basic axis identifier in machine data:
     MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)
4. If the axis sequence is not the same as the normal axis sequence, it must be corrected in machine data:
   MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)
5. ID specification for the wrist axes. If axis 4 and 5 intersect, a central hand (ZEH) is present. In all other cases, the ID for beveled hand with elbow (BHE) must be entered in the machine data:
   MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)
6. Whether axis 4 runs parallel/anti-parallel to the last rotary basic axis must be entered into the machine data:
   MD62606 $MC_TRAFO6_A4PAR (axis 4 is parallel/anti-parallel to last basic axis)
7. Enter the axis types for the transformation in machine data:
   MD62601 $MC_TRAFO6_AXES_TYPE (axis type for transformation)
8. Compare the directions of rotation of axes with the directions defined in the handling transformation package and correct in machine data:
   MD62618 $MC_TRAFO6_AXES_DIR (matching of physical and mathematical directions of rotation)
9. Enter the mechanical zero offset in the machine data:
   MD62617 $MC_TRAFO6_MAMES (offset between mathematical and mechanical zero points)
10. Enter the basic axis lengths in machine data:
    MD62607 $MC_TRAFO6_MAIN_LENGTH_AB (basic axis lengths A and B)
11. Define frame T_IRO_RO and enter the offset in machine data:
    MD62612 $MC_TRAFO6_TIRORO_POS (frame between base center point and internal system (position component))
    Define frame T_IRO_RO and enter the rotation in machine data:
    MD62613 $MC_TRAFO6_TIRORO_RPY (frame between base center point and internal system (rotation component))

12. Specification of frame T_X3_P3 to attach hand. The offset is entered in machine data:
    MD62608 $MC_TRAFO6_TX3P3_POS (attachment of hand (position component))
    The rotation is entered in the machine data:
    MD62609 $MC_TRAFO6_TX3P3_RPY (attachment of hand (rotation component))
13. Specification of wrist axes parameters. For this purpose, only the parameters for axis 4 must be entered in the machine data:
    MD62614 $MC_TRAFO6_DHPAR4_5A[0] (parameter A for configuring the hand)
    MD62616 $MC_TRAFO6_DHPAR4_5ALPHA[0] (parameter A for configuring the hand)
    All other parameters must be set to 0.0.
14. Determine the flange coordinate system. For this purpose, the hand-point coordinate system must be regard as the initial system. The offset is entered in the machine data:
    MD62610 $MC_TRAFO6_TFLWP_POS (frame between wrist point and flange (position component))
    The rotation is entered in the machine data:
    MD62611 $MC_TRAFO6_TFLWP_RPY (frame between wrist point and flange (rotation component))

SCARA kinematics

5-axis CC kinematics

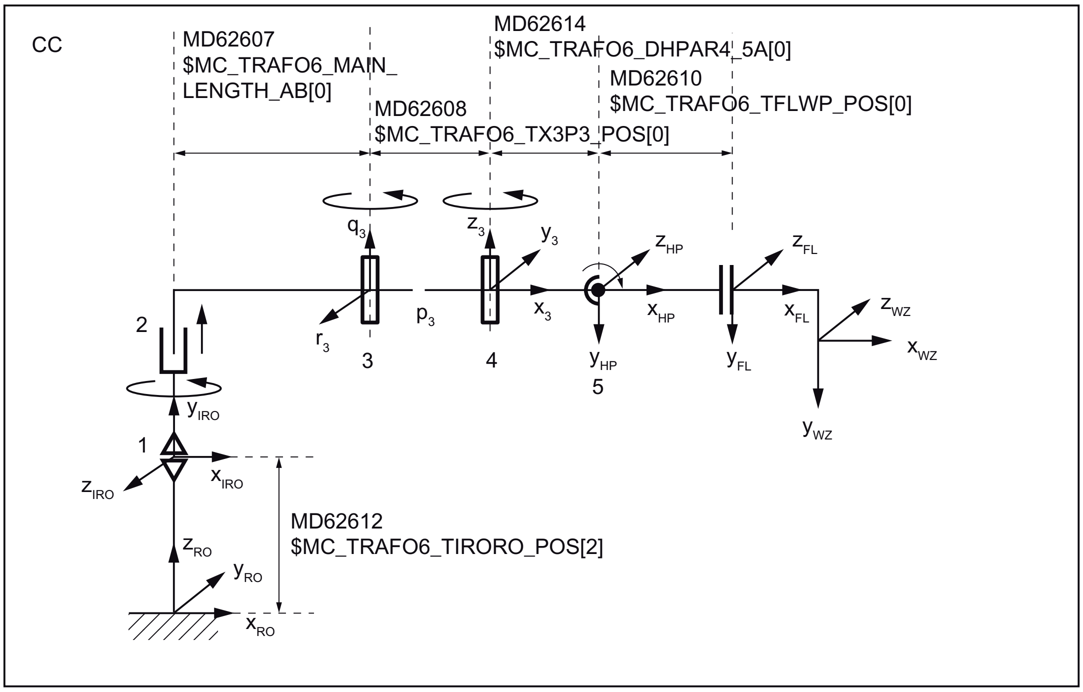


Figure 19-22 5-axis CC kinematics

Table 19-14 Configuration data for 5-axis CC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	5

Machine data	Value
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	5
MD62606 $MC_TRAFO6_A4PAR	1
MD62601 $MC_TRAFO6 _AXES_TYPE	[3, 1, 3, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[2, 1, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[0.0, 500.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 90.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[300.0, 0.0, -200.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, -90.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]
MD62614 $MC_TRAFO6_DHPAR4_5A	[200.0, 0.0]
MD62615 $MC_TRAFO6_DHPAR4_5D	[0.0, 0.0]
MD62616 $MC_TRAFO6_DHPAR4_5ALPHA	[-90.0, 0.0]

5-axis NR kinematics

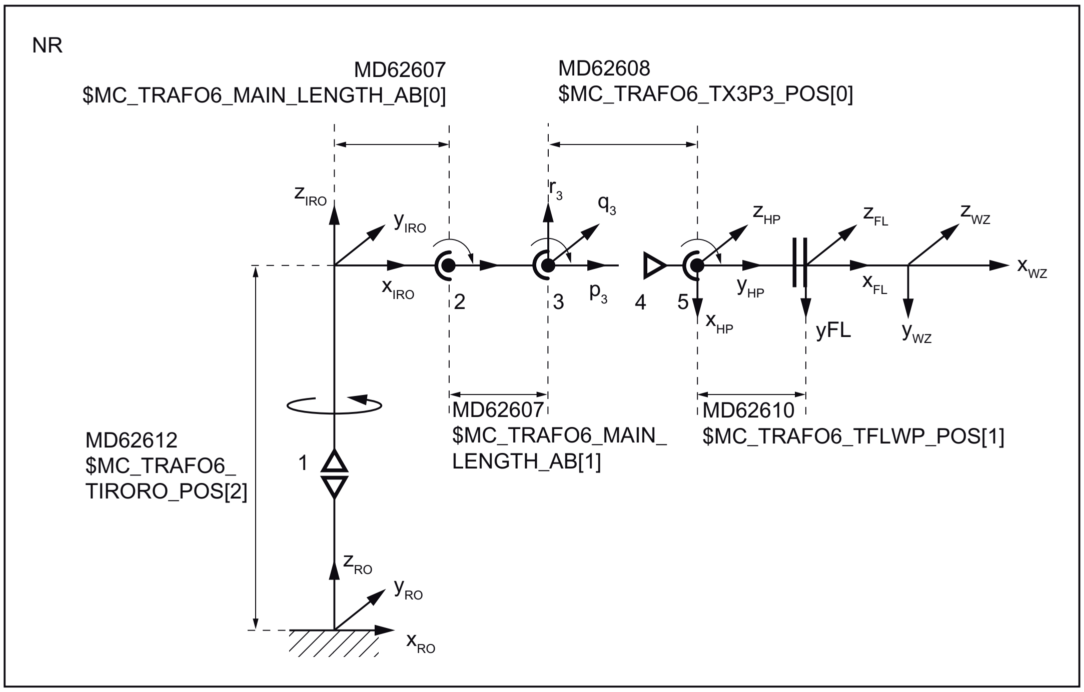


Figure 19-23 5-axis NR kinematics

Table 19-15 Configuration data 5-axis NR kinematic

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	5
MD62603 $MC_TRAFO6_MAIN_AXES	3
MD62604 $MC_TRAFO6_WRIST_AXES	2
MD62606 $MC_TRAFO6_A4PAR	0
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, 3, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[30.0, 300.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 500.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[500.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 90.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, -300.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[-90.0, 0.0, 0.0]
MD62614 $MC_TRAFO6_DHPAR4_5A	[0.0, 0.0]
MD62615 $MC_TRAFO6_DHPAR4_5D	[0.0, 0.0]
MD62616 $MC_TRAFO6_DHPAR4_5ALPHA	[-90.0, 0.0]

See also

Definition of a joint (Page 796)

19.5.4 6-axis kinematics

6-axis kinematics usually imply 3 degrees of freedom for translation and 3 more for orientation. This allows for the tool direction to be manipulated freely in space. Also, the tool can be rotated along its own axis to the machining surface or inclined with a tilting angle.

This kinematic type is supported by the software, but only for very specific solutions as the auxiliary method for this handling transformation package. A general, comprehensive and universally applicable software version is still not available for 6-axis kinematics.

19.5.5 Special kinematics

MD62602 $MC_TRAFO6_SPECIAL_KIN (special kinematic type)

Special kinematics are kinematics that are not directly included in the building block system of the Handling transformation package. They are frequently missing a degree of freedom or are characterized by mechanical links between the axes or with the tool. Set the following machine data for these kinematics:

MD62600 $MC_TRAFO6_KINCLASS = 2 (kinematic category)

Which special kinematic is handled is specified in machine data:

MD62602 $MC_TRAFO6_SPECIAL_KIN (special kinematic type)

Special 2-axis SC kinematic

This special kinematic is characterized by the fact that the tool is always maintained in the same orientation via a mechanical linkage. It implies two Cartesian degrees of protection. The identifier for this kinematic is machine data:

MD62602 $MC_TARFO6_SPECIAL_KIN = 3 (special kinematic type)

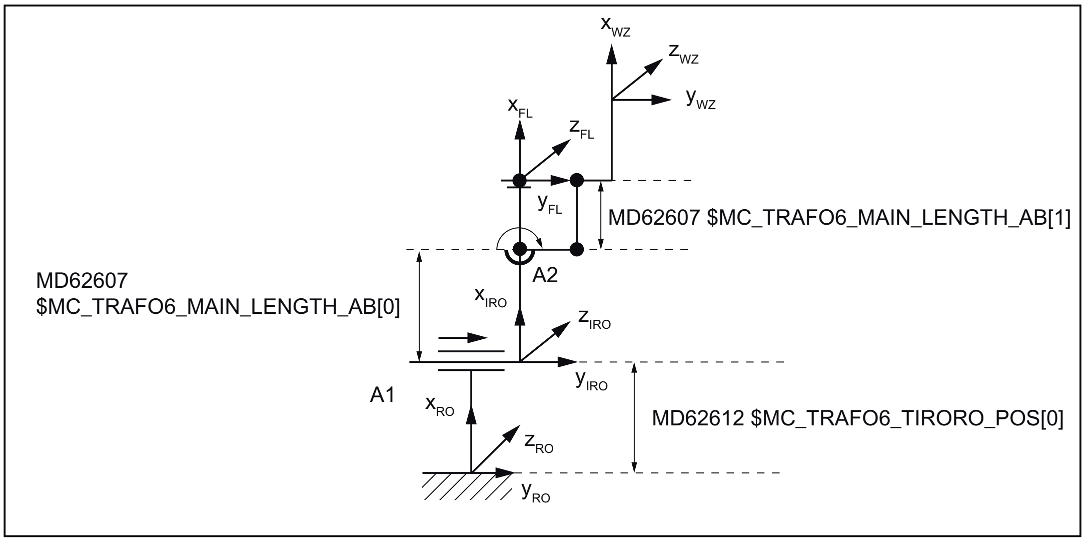


Figure 19-24 Special 2-axis SC kinematic

Table 19-16 Configuring data for a special 2-axis SC kinematic

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	2
MD62602 $MC_TRAFO6_SPECIAL_KIN	3
MD62605 $MC_TRAFO6_NUM_AXES	2
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[1, 3, 3, ...]

Machine data	Value
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[400.0, 500.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 300.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[0.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

Special 3-axis SC kinematic

The special kinematic has 2 Cartesian degrees of freedom and one degree of freedom for orientation. The identifier for this kinematic is:

MD62602 $MC_TRAFO6_SPECIAL_KIN = 4 (special kinematic type)

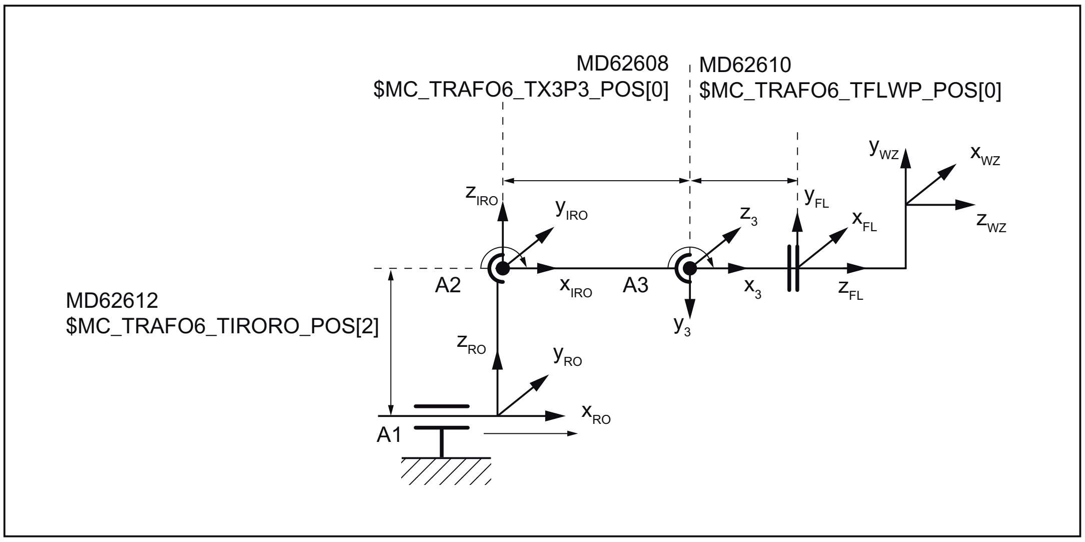


Figure 19-25 Special 3-axis SC kinematic

Table 19-17 Configuring data for a special 3-axis SC kinematic

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	2
MD62602 $MC_TRAFO6_SPECIAL_KIN	4
MD62605 $MC_TRAFO6_NUM_AXES	3
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[1, 3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]

Machine data	Value
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[0.0, 0.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[0.0, 0.0, 400.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[400.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, -90.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 180.0]

Special 4-axis SC kinematic

This special kinematic is characterized by the fact that axis 1 and axis 2 are mechanically coupled. This coupling ensures that axis 2 is maintained at a constant angle when axis 1 is swiveled. This kinematic also guarantees that axes 3 and 4 always remain perpendicular, irrespective of the positions of axes 1 and 2. The identifier for this kinematic is:

MD62602 $MC_TRAFO6_SPECIAL_KIN = 7 (special kinematic type)

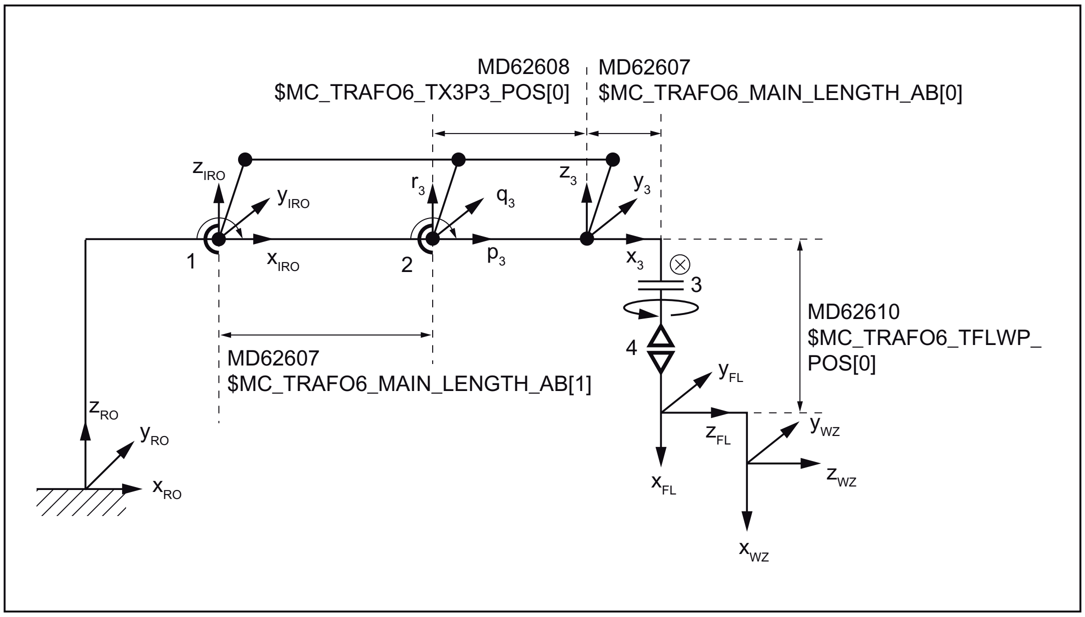


Figure 19-26 Special 4-axis SC kinematic

Table 19-18 Configuring data for a special 4-axis SC kinematic

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	2
MD62602 $MC_TRAFO6_SPECIAL_KIN	7
MD62605 $MC_TRAFO6_NUM_AXES	4
MD62603 $MC_TRAFO6_MAIN_AXES	2
MD62604 $MC_TRAFO6_WRIST_AXES	1

Machine data	Value
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, 1, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[100.0, 400.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[100.0, 0.0, 1000.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, 0.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[300.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, 0.0, -600.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 90.0, 0.0]

Special 2-axis NR kinematics

This special kinematic is characterized by the fact that axis 1 and axis 2 are mechanically coupled. Another special feature is the tool. With this kinematic, it maintains its orientation in space irrespective of the positions of the other axes.

The identifier for this kinematic is:

MD62602 $MC_TRAFO6_SPECIAL_KIN = 5 (special kinematic type)

Without a mechanical coupling between axis 1 and 2, it has the following identifier:

MD62602 $MC_TRAFO6_SPECIAL_KIN = 8 (special kinematic type)

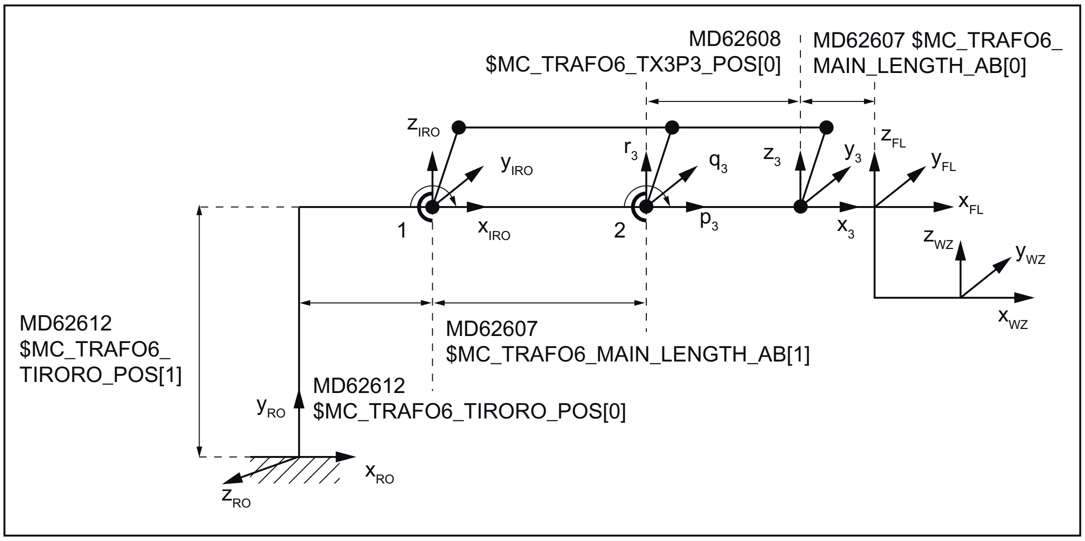


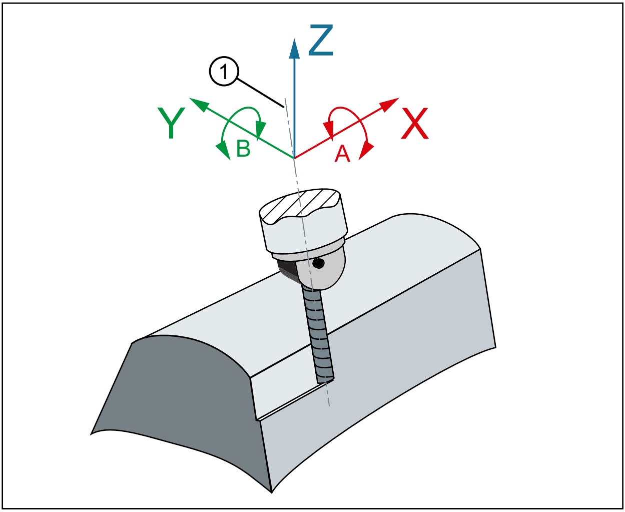
Figure 19-27 Special 2-axis NR kinematics

Table 19-19 Configuration data for special 2-axis NR kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	2
MD62602 $MC_TRAFO6_SPECIAL_KIN	5 (8)

Machine data	Value
MD62605 $MC_TRAFO6_NUM_AXES	2
MD62603 $MC_TRAFO6_MAIN_AXES	3
MD62604 $MC_TRAFO6_WRIST_AXES	1
MD62601 $MC_TRAFO6_AXES_TYPE	[3, 3, ...]
MD62620 $MC_TRAFO6_AXIS_SEQ	[1, 2, 3, 4, 5, 6]
MD62618 $MC_TRAFO6_AXES_DIR	[1, 1, 1, 1, 1, 1]
MD62617 $MC_TRAFO6_MAMES	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MD62607 $MC_TRAFO6_MAIN_LENGTH_AB	[100.0, 400.0]
MD62612 $MC_TRAFO6_TIRORO_POS	[100.0, 500.0, 0.0]
MD62613 $MC_TRAFO6_TIRORO_RPY	[0.0, 0.0, -90.0]
MD62608 $MC_TRAFO6_TX3P3_POS	[400.0, 0.0, 0.0]
MD62609 $MC_TRAFO6_TX3P3_RPY	[0.0, 0.0, 0.0]
MD62610 $MC_TRAFO6_TFLWP_POS	[0.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, 0.0, 0.0]

19.6 Tool orientation



① Tool axis

Figure 19-28 Peripheral milling with 5-axis transformation

Machine data

Identifiers of the Euler angles

The identifiers with which the Euler angles are programmed in the NC program can be set using:

MD10620 $MN_EULER_ANGLE_NAME_TAB (name of Euler angles)

Standard identifier: "A2", "B2", "C2"

Identifier of the direction vector components

The identifier with which the components of the direction vector are programmed in the NC program can be set using:

MD10640 $MN_DIR_VECTOR_NAME_TAB (name of normal 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.

Channel-specific input: Euler or RPY angle

Whether the tool orientation programming is realized in the channel with Euler angle or RPY angle can be set using:

MD21100 $MC_ORIENTATION_IS_EULER (angle definition for orientation programming)

G command for orientation interpolation

The type of interpolation for the orientation is specified using machine data:

MD21104 $MC_ORI_IPO_WITH_G_CODE = <value>

<value>	Meaning
FALSE	G commands for orientation interpolation: `ORIWKS` and `ORIMKS`
TRUE	G commands of the 51st G group for orientation interpolation: `ORIAXES`, `ORIVECT`, `ORIPLANE`, ...

Channel-specific initial setting for the tool orientation

G commands of the 25th G group: "Tool orientation reference" which takes effect after a channel or end of program reset, can be set via:

MD20150 $MC_GCODE_RESET_VALUES[24] (25. G group: tool orientation reference)

Programming

Function

The tool orientation can be programmed via:

- Directly programming the rotary axes
- Programming the Euler or RPY angle
- Programming the direction vector

The coordinate system in which motion is executed is defined using the `ORIWKS` and `ORIMKS` commands.

Syntax

```
ORIWKS
ORIMKS
```

Meaning

`ORIWKS`:	Tool orientation in the workpiece coordinate system (WCS) 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 extending from the start vector to the end vector. Transitioning a tool orientation at the start of a block to an orientation at the end of the block is only possible in the WCS.	
	Effective:	Modal
`ORIMKS`:	Tool orientation in the machine coordinate system (MCS) 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.	
	Effective:	Modal

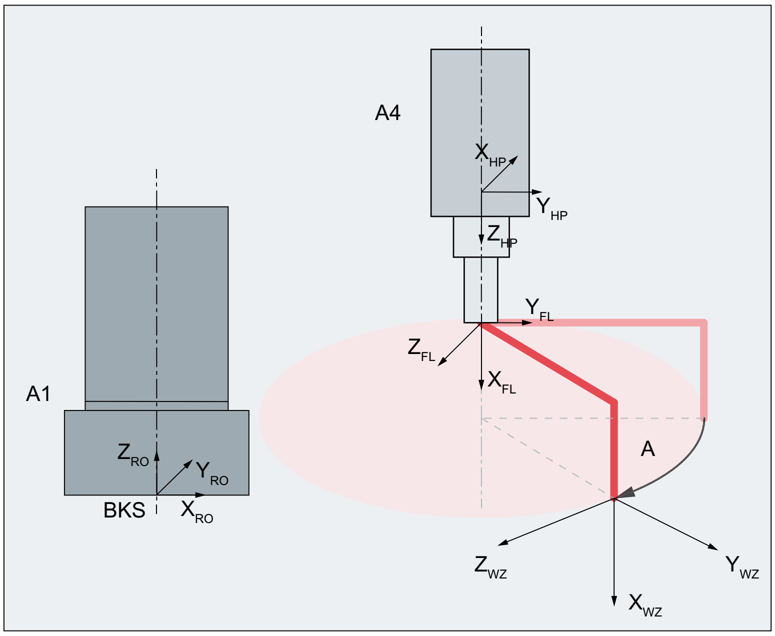
Note

- `ORIWKS` is the preferred approach for a program independent of a particular machine.
- It is not possible to program the tool orientation using Euler angles, RPY angles or direction vectors for kinematics involving fewer than 5 axes. For 4-axis kinematics with only one rotary axis, there is only one degree of freedom for orientation. This orientation angle can only be programmed with orientation axis angle "A".
- Transferring an orientation using `ORIMKS` is not allowed in the handling transformation package. With an active transformation, the machine axis angles are not programmed and traversed, but rather the "orientation angles" (RPY angles according to robotics definition, see Section "Definition of positions and orientations using frames (Page 795)").

19.6.1 Orientation programming for 4-axis kinematics

4-axis kinematics possess only one degree of freedom for orientation. When programming orientation using RPY angle, Euler angle or direction vector, it is not always possible to approach the specified orientation. If at all, this type of orientation programming is only practical for certain kinematics, where there is an invariance of the orientation angle with respect to the basis axes. This is the case for the Scara kinematics, for example.

For this reason, orientation programming is only permitted via "orientation angle" A for 4-axis kinematics. This angle corresponds to the RPY angle C of the robot definition. This means one rotation around the Z_{RO} axis, as illustrated in Fig. "Orientation angle for 4-axis kinematics".



A Virtual orientation angle

Figure 19-29 Orientation angle for 4-axis kinematics

Absolute calculation of the tool orientation for 4-axis kinematics

For 4-axis kinematics, the "virtual orientation angle" A is normally only defined within the range -180° < A < = + 180°. This means that, in the transition from 180° to 181°, angle A jumps to -179°. The following machine data must be set to avoid this step, and to ensure that angle A is continually controlled:

MD62643 $MC_TRAFO6_SPECIAL_FEATURE_MASK, bit 0 = 1

As a consequence, for the following 4-axis kinematics, angle A is controlled in absolute terms:

- CC kinematics
- SC kinematics
- CS kinematics

This is on condition that the 4th axis is installed parallel to the last rotary basic axis (MD62606 $MC_TRAFO6_A4PAR == 1).

Angle A is calculated from the sum of the rotary axes. Machine data MD62617 $MC_TRAFO6_MAMES and MD62618 $MC_TRAFO6_AXES_DIR are taken into account.

If, as a result of the rotations via MD62612 $MC_TRAFO6_TIRORO_RPY and MD62611 $MC_TRAFO6_TFLWP_RPY, additional rotations are obtained, then these must be compensated using MD62642 $MC_TRAFO6_C_ANGLE_OFFSET. In machine data MD62642, the angle, which is displayed as "virtual orientation angle" A at the mathematical zero position of the axes, must be entered with **inverted** sign.

19.6.2 Orientation programming for 5-axis kinematics

For 5-axis kinematics, when programming via orientation vector, it is assumed that the orientation vector corresponds to the x component of the tool.

When programming via orientation angle (RPY angle according to robotics definition), the x component of the tool is considered the initial point for rotations.

For this purpose, the vector in the x tool direction, as shown in Fig. "Orientation angle for 5-axis kinematic", is first rotated around the Z axis by the angle A and then around the rotated Y axis by the angle B. The rotation by the angle C is not possible for 5-axis kinematics because of the restricted degrees of freedom for the orientation.

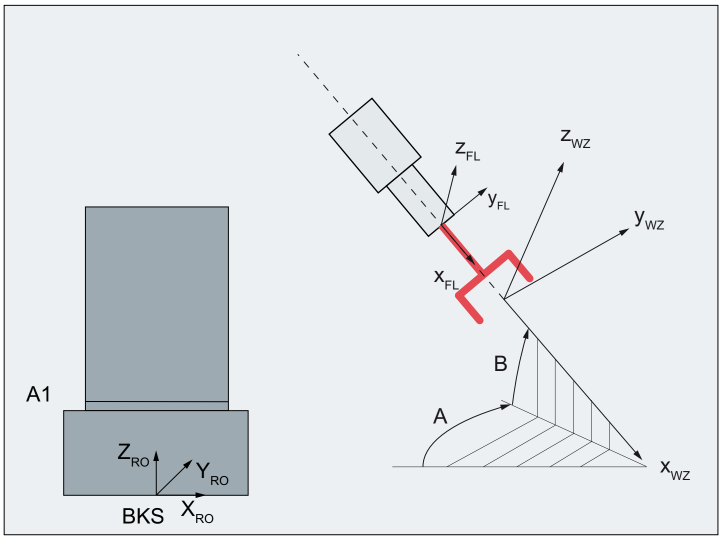


Figure 19-30 Orientation angle for 5-axis kinematic

Adaptation of the flange coordinate system

This machine data is used to set the flange coordinate system on the user side so that Z can be set as the tool direction for 5-axis kinematics:

MD62636 $MC_TRAFO6_TFL_EXT_RPY

Definition of the tool direction

In this machine data, set whether the tool direction will be calculated according to robotics conventions or NC conventions:

MD62637 $MC_TRAFO6_TOOL_DIR

Standard definition of the tool direction

The tool calculation normally used with machine tools can be set with machine data:

MD62637 $MC_TRAFO6_TOOL_DIR = 1

This machine data also has an effect on the rotation sequence of the virtual orientation axes. For this purpose, the vector in the Z tool direction is first turned through angle A at the X axis and then through angle B at the rotated Y axis. The rotation by the angle C is not possible for 5-axis kinematics because of the restricted degrees of freedom for the orientation.

Note

More information can be found in:

- Function Manual, Special Functions, Section "Orientation axes"
- Programming Manual, Job Planning; Section "Orientation axes".

19.7 Singular positions and how they are handled

The calculation of the machine axes to a preset position, i.e. position with orientation, is not always clear. Depending on the machine kinematic, there may be positions with an infinite number of solutions. These positions are called "singular".

Singular positions

- A singular position is, for example, characterized by the fact that the fifth axis is positioned at 0°. In this case, the singular position does not depend on a specified orientation. The fourth axis is not specified in this position, i.e., the fourth axis does not have any influence on the position or the orientation.
- A singular position also applies for articulated arm and SCARA kinematics if the third axis is at 0° or 180°. These positions are called leveling/diffraction singularity.
- Another singular position exists for articulated arm kinematics if the hand point is above the rotary axis of axis 1. This position is called overhead singularity.

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.

Behavior at pole

The unwanted behavior of fast compensating movements can be improved by reducing the velocity in the proximity of a pole. Traveling through the pole with active transformation is usually not possible.

19.8 Call and application of the transformation

Activating

The transformation is activated by means of the `TRAORI(1)` command.

Once the `TRAORI(1)` command has been executed and the transformation thus activated, the interface signal switches to "1":

DB21, … DBX33.6 (transformation active)

If the machine data have not been defined for an activated transformation grouping, the NC program stops and the control displays the alarm 14100 "Orientation transformation does not exist".

For more information, go to:

References:
Programming Manual, Production Planning,
Section: "5-axis machining"

Deactivating

The currently active transformation is deactivated by means of `TRAFOOF` or `TRAFOOF`().

Note

When the "Handling transformation package" transformation is deactivated, no preprocessing stop and no synchronization of the preprocessing with the main run is performed.

RESET/end of program

The control behavior in terms of transformation following run-up, end of program or `RESET` depends on machine data:

MD20110 $MC_RESET_MODE_MASK (definition of control basic setting na)

Bit 7:	Reset behavior of "active kinematic transformation"
Bit 7 = 0	For this reason, the initial setting is defined for active transformation after the end of part program or `RESET`, according to the following machine data: MD20140 $MC_TRAFO_RESET_VALUE (transformation data block run-up (re-set/part program end)) Meaning: 0: After `RESET` no transformation is active. 1 to 8: The set transformation is active according to machine data: MD24100 $MC_TRAFO_TYPE_1 (definition of transformation 1 in the channel) To machine data: MD24460 $MC_TRAFO_TYPE_8 (channel transformation 8)
Bit 7 = 1	The current setting for the active transformation remains unchanged after a `RESET` or end of part program.

19.9 Actual value display

MCS machine coordinate system

The machine axes are displayed in mm/inch and/or degrees in MCS display mode.

WCS workpiece coordinate system

If the transformation is active, the tool tip (TCP) is specified in mm/inch and the orientation by the RPY angles A, B and C in WCS display mode. The tool orientation results from turning a vector in the Z direction firstly with A around the Z axis, then with B around the new Y axis and lastly with C around the new X axis.

If the transformation is deactivated, the axes will be displayed with the channel axis names. Otherwise the geometry axis names will be displayed.

19.10 Tool programming

Meaning

The tool lengths are specified in relation to the flange coordinate system. Only 3-dimensional tool compensation is possible. Depending on which kinematic is involved, additional restrictions for the tool exist for 5- and 4-axis kinematics. For a kinematic as illustrated in Fig. "5-axis NR kinematic", only a 1-dimensional tool with length in the x direction is permitted.

The tool direction is dependent on the machine's initial setting, as specified with G commands `G17`, `G18` and `G19`. The tool lengths refer to the zero position specified by `G17`. These should not be changed in the program.

Example

An example of a 2-dimensional tool mounted on a 5-axis Scara is described below (see Fig. "5-axis CC kinematic"). Type 100 (cutting tool) should be specified as identification for the tool. The tool lengths result from the specifications shown in Fig. "Tool length programming". X-TOOL must be entered as tool length x and Y-TOOL must be entered as tool length y in the tool parameters.

$TC_DP1[1,1] = 100 ;	Cutting tool type
$TC_DP3[1,1] = 0.0 ;	(z) length compensation vector
$TC_DP4[1,1] = Y-TOOL ;	(y) length compensation vector
$TC_DP5[1,1] = X-TOOL ;	(x) length compensation vector

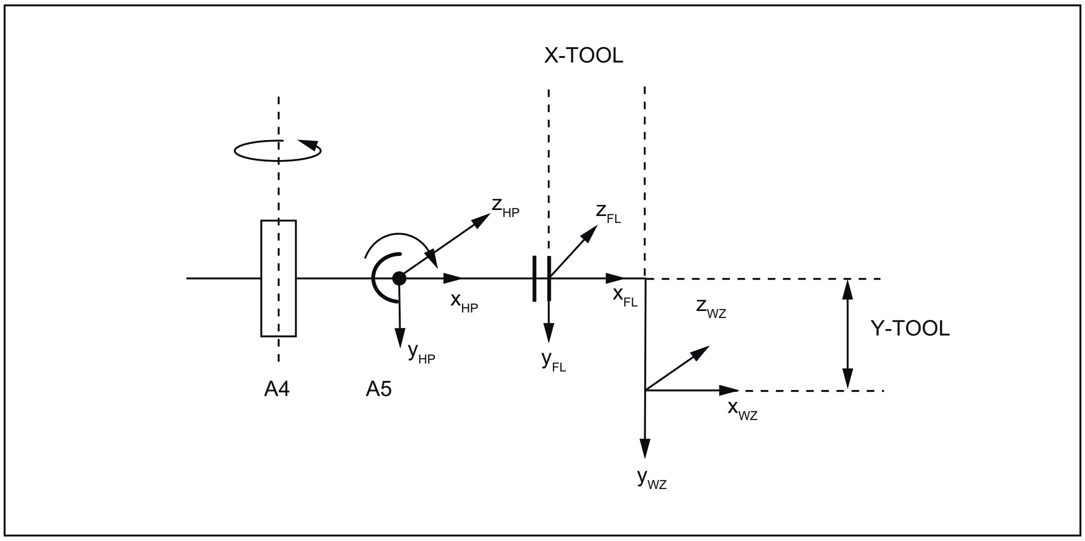


Figure 19-31 Tool length programming

19.11 Cartesian PTP travel with handling transformation package

It is possible to use the "Cartesian PTP travel" function with the handling transformation package (see Section "Cartesian PTP travel (Page 44)"). For this, the following machine data must be set:

MD24100 $MC_TRAFO_TYPE_1 = 4100 (definition of transformation 1 in the channel)

19.12 Startup

19.12.1 General commissioning

The handling transformation package is a compile cycle.

The general commissioning of a compile cycle is described in Chapter "TE01: Installation and activation of loadable compile cycles (Page 707)".

19.12.2 Function-specific commissioning

Option

The "Handling transformation package" function is an option for which a license must be purchased:

- Transformation Handling RCTRA, Order No.: 6FC5800-0AM31-0YB0

A detailed description of licensing, for instance the assignment between a license and hardware, is provided in:

References

Commissioning Manual, CNC Commissioning: NC, PLC, Drive

Section: "Licensing"

User-specific alarm texts

To create user-specific alarm texts in conjunction with compile cycles, see Chapter "Creating alarm texts (Page 713)"

Parameterizing transformation

1. Enter the transformation type into the machine data:
   - MD24100 $MC_TRAFO_TYPE_1 = **4100** (definition of transformation 1 in the channel)
2. Enter the assignment of the channel axes involved in the transformation in the machine data (axis numbers, starting at 1):
   - MD24110 $MC_TRAFO_AXES_IN_1[0 ... 5] (axis assignment for transformation)
3. Enter the geometry axes corresponding to the Cartesian degrees of freedom of the machine in the machine data:
   - MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0 ... 2] (assignment between geometry axis and channel axis for transformation 1)
4. Enter the kinematic identifier in the machine data:
   - MD62600 $MC_TRAFO6_KINCLASS (kinematic category)
5. Enter the identifier for special kinematics, if there are any, in machine data:
   - MD62602 $MC_TRAFO6_SPECIAL_KIN (special kinematic type)
6. Enter the number of axes in the machine data:
   - MD62605 $MC_TRAFO6_NUM_AXES (number of transformed axes)
7. If the travel direction of the involved axes is opposed to the transformation definition, then change the factory setting in the machine data:
   - MD62618 $MC_TRAFO6_AXES_DIR[ ] (matching of physical and mathematical directions of rotation)
8. Enter the data which define the basic axes:
   - MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)
   - MD62607 $MC_TRAFO6_MAIN_LENGTH_AB (basic axis lengths A and B)
9. Enter any changes to the axis sequence in the machine data:
   - MD62620 $MC_TRAFO6_AXIS_SEQ (rearrangement of axes)

10. Enter the data which define the wrist axis:
    - MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)
    - MD62614 $MC_TRAFO6_DHPAR4_5A (parameter A for configuring the wrist axis)
    - MD62615 $MC_TRAFO6_DHPAR4_5D (parameter D for configuring the wrist axis)
    - MD62616 $MC_TRAFO6_DHPAR4_5ALPHA (parameter ALPHA for configuring the wrist axis)
    - MD62606 $MC_TRAFO6_A4PAR (axis 4 is parallel/anti-parallel to last basic axis)
11. Enter the geometry parameters:
    - Frame T_IRO_RO
    - Frame T_X3_P3
    - Frame T_FL_WP
12. Enter the position in relation to the calibration point in the machine data:
    - MD62617 $MC_TRAFO6_MAMES (offset between mathematical and mechanical zero points)
13. Enter the Cartesian velocities and acceleration rates.

19.13 Supplementary conditions

Clearance control

The handling transformation package **cannot** be used together with the technology function: "**clearance control**".

Moving to fixed end stop

The handling transformation package **cannot**be used in conjunction with the "**Travel to fixed stop**".

Several transformations

The handling transformation package can be activated just once per channel in all channels.

Tool programming

Tools can only be parameterized by specifying tool lengths. It is not possible to program an orientation for the tool.

Programming of orientation

The programming possibilities of the orientation depend on the number of axes available on the machine:

- Number < 5:
  - Orientation axis angle
- Number = 5:
  - Orientation axis angle
  - Orientation vector

Singularities

A pole cannot be directly passed through when a transformation is active.

NOTICE

Traversing close to the pole

When traversing close to the pole, the axes involved can be overloaded. This is because the control does not automatically reduce or limit the feedrate, acceleration or jerk. As a consequence, the user is responsible in limiting the feedrate of the axes involved and/or the clearance to the pole.

19.14 Data lists

19.14.1 Machine data

19.14.1.1 General machine data

Number	Identifier: $MN_	Description
10620	EULER_ANGLE_NAME_TAB[n]	Name of Euler angle
19410	TRAFO_TYPE_MASK, bit 4	Option data for OEM transformation
60943	CC_ACTIVE_IN_CHAN_RCTR	Activation of the handling transformation for the corresponding channels
19610	TECHNO_EXTENSION_MASK	Option data for handling transformation

19.14.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system with automatic FRAME definition
24100	TRAFO_TYPE_1	Definition of transformation
24110	TRAFO_AXES_IN_1	Axis assignment for transformation 1
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Assignment of geometry axes to channel axes

19.14.1.3 Channel-specific machine data for compile cycles

Number	Identifier: $MC_	Description
62600	TRAFO6_KINCLASS	Kinematic category
62601	TRAFO6_AXES_TYPE	Axis type for transformation
62602	TRAFO6_SPECIAL_KIN	Special kinematic type
62603	TRAFO6_MAIN_AXES	Basic axis identifier
62604	TRAFO6_WRIST_AXES	Wrist axis identifier
62605	TRAFO6_NUM_AXES	Number of transformed axes
62606	TRAFO6_A4PAR	Axis 4 is parallel/anti-parallel to last basic axis
62607	TRAFO6_MAIN_LENGTH_AB	Basic axis lengths A and B
62608	TRAFO6_TX3P3_POS	Attachment of hand (position component)
62609	TRAFO6_TX3P3_RPY	Attachment of hand (rotation component)
62610	TRAFO6_TFLWP_POS	Frame between wrist point and flange (position component)
62611	TRAFO6_TFLWP_RPY	Frame between wrist point and flange (rotation component)
62612	TRAFO6_TIRORO_POS	Frame between base center point and internal system (position component)
62613	TRAFO6_TIRORO_RPY	Frame between base center point and internal system (rotation component)
62614	TRAFO6_DHPAR4_5A	Parameter A for configuring the hand
62615	TRAFO6_DHPAR4_5D	Parameter D for configuring the hand
62616	TRAFO6_DHPAR4_5ALPHA	Parameter ALPHA for configuring the hand
62617	TRAFO6_MAMES	Offset between mathematical and mechanical zero points
62618	TRAFO6_AXES_DIR	Matching of physical and mathematical directions of rotation
62619	TRAFO6_DIS_WRP	Mean distance between wrist point and singularity (**not used**)
62620	TRAFO6_AXIS_SEQ	Rearrangement of axes
62621	TRAFO6_SPIN_ON	Configuration includes triangular or trapezoidal spindles
62622	TRAFO6_SPIND_AXIS	Axis that is controlled by triangular spindle

Number	Identifier: $MC_	Description
62623	TRAFO6_SPINDLE_RAD_G	Radius G for triangular spindle
62624	TRAFO6_SPINDLE_RAD_H	Radius H for triangular spindle
62625	TRAFO6_SPINDLE_SIGN	Sign for triangular spindle
62626	TRAFO6_SPINDLE_BETA	Angular offset for triangular spindle
62627	TRAFO6_TRP_SPIND_AXIS	Axes driven via trapezoidal connection
62628	TRAFO6_TRP_SPIND_LEN	Trapezoid lengths
62629	TRAFO6_VELCP	Cartesian velocities (substitute limit values for the velocity controller)
62630	TRAFO6_ACCCP	Cartesian accelerations (substitute limit values for the velocity controller)
62631	TRAFO6_VELORI	Orientation angle velocities (substitute limit values for the velocity controller)
62632	TRAFO6_ACCORI	Orientation angle accelerations (substitute limit values for the velocity controller)
62634	TRAFO6_DYN_LIM_REDUCE	Reduction factor for velocity controller during the JOG travel
62635	TRAFO6_VEL_FILTER_TIME	Time constant for PT1 filter when the velocity controller is active
62636	TRAFO6_TFL_EXT_RPY	Adaptation of the flange coordinate system
62637	TRAFO6_TOOL_DIR	Definition of the tool direction
62642	TRAFO6_C_ANGLE_OFFSET	Offset angle for C angle calculation in the BCS
62643	SPECIAL_FEATURE_MASK	Function mask

19.14.2 Signals

19.14.2.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate PTP traversing	DB21, … .DBX29.4	-

19.14.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Transformation active	DB21, ...DBX33.6	DB330x.DBX1.6
Number of the active G commands of the G group 25 (ORIWKS, ORIMKS, ORIPATH)	DB21, ...DBB232	-
PTP traversing active	DB21, … .DBX317.6	-

TE6: MCS coupling - 840D sl only 20

20.1 Brief description

If there are two or more separately traversable machining heads on a machine tool and a transformation is required for machining, the orientation axes of the machining heads cannot be coupled via the standard coupling functions `COUPON`, `TRAILON`. The couplings are established in the workpiece coordinate system (WCS). However, under the conditions mentioned above the coupling of the orientation axes must be performed in the machine coordinate system (MCS).

The compile cycle "MCS coupling" can be used to parameterize couplings in the machine coordinate system (MCS) with coupling factors "1" and "-1" for a master axis, referred to as CC_Master in the following, and several slave axes, referred to as CC_Slave in the following.

The couplings can be switched on and off via commands in the part program.

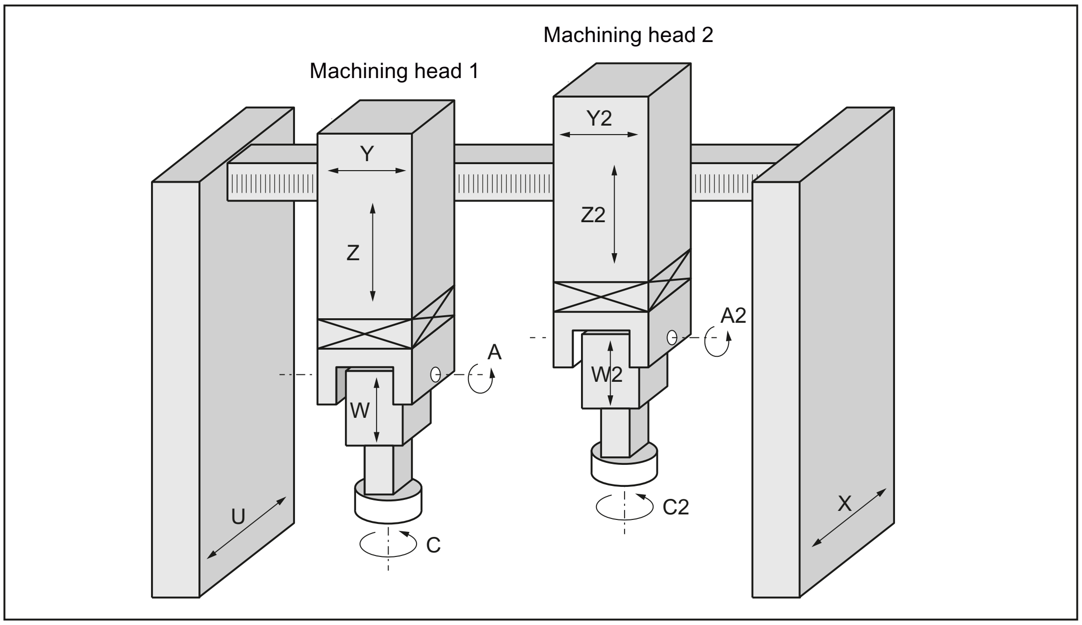


Figure 20-1 Application example: Double slide gantry-type milling machine

Master and slave axes

A CC_Master can be master for several CC_Slaves. A CC_Slave cannot be a CC_Master at the same time.

The following functions are not possible for a CC_Slave:

- To be a PLC axis
- To be a command axis
- To be traversed separately from the CC_Master in JOG mode

Tolerance window

When the coupling is active, the actual values of the CC_Master and CC_Slave are monitored for compliance with a parameterizable tolerance window.

Programming a CC-Slave

If a CC_Slave is programmed in a channel, either an alarm is displayed or the axis is requested for the channel via an implicitly triggered axis interchange (`GET`), depending on the setting in machine data:

MD30552 $MA_AUTO_GET_TYPE = <GET type> (automatic GET for axis)

Switching the coupling on/off

A coupling is switched on and off via OEM-specific language commands and can be active in all operating modes.

Collision protection

In order to protect the machining heads against collision in decoupled operation or in mirrored coupling, a collision protection can be parameterized. The activation is performed via machine data or the NC/PLC interface. The assignment of the protected pairs is not dependent on the CC_Master and CC_Slave pairs.

20.2 Description of MCS coupling functions

20.2.1 Defining coupling pairs

A CC_SLAVE axis is matched to its CC_Master axis via the following axial machine data:

MD63540 $MA_CC_MASTER_AXIS (specifies the CC_Master axis assigned to a CC_Slave axis)

The axes involved in the coupling can only be changed when the coupling is not active (open).

A CC_Slave axis is displayed in axial VDI-Out byte:

DB31, ... DBX97.0 (MCS coupling: slave axis)

Precondition

- The CC_Master and CC_Slave axes must be either both rotary axes or both linear axes.
- Spindles cannot be coupled by this function.
- Neither CC_Master nor CC_Slave axis are permitted to be a interchanged axis ($MA_MASTER_CHAN[AXn]=0)

20.2.2 Switching the coupling ON/OFF

Activating the coupling

- Activating the 1:1 coupling. Monitoring of the tolerance window is active.
  CC_COPON([<axis1>][<axis2>][<axis3>][<axis4>][<axis5>])
- Activating the 1:-1 coupling (mirror). Monitoring of the tolerance window is not active.
  CC_COPONM([<axis1>][<axis2>][<axis3>][<axis4>][<axis5>])

Axis name

An axis can be specified via:

- Machine axis name
- Channel axis name
- Geometry axis name

Boundary conditions

- A programmed axis must be involved in a coupling.
- CC_Master axes, CC_Slave axes or both can be programmed simultaneously. All defined couplings are switched on with CC_COPON or CC_COPONM.
- An alarm is output if an axis not involved in a coupling is programmed.

NC/PLC interface signals

All NC/PLC interface signals of a coupling refer to the CC_Slave axes.

- DB31, … DBX97.1 (MCS coupling: coupling active)
- DB31, ... DBX97.2 (MCS coupling: mirroring active)
- DB31, …DBX24.2 (MCS coupling: deactivate or do not permit)
  If the coupling is suppressed, an alarm is not output for the CC_Slave axis.

Deactivating the coupling

CC_COPOFF([<axis1>][<axis2>][<axis3>][<axis4>][<axis5>])

As for switching on the coupling with the difference that no alarm is output if an axis is programmed that is not involved in a coupling.

An existing coupling can also be switched off by the axial NC/PLC interface signal of the CC_Slave axis.

Boundary condition

A coupling can only be switched on or off when the axes are at standstill.

20.2.3 Tolerance window

A monitoring window is specified via axial machine data:

MD63541 $MA_CC_POSITION_TOL (monitoring window)

The absolute difference between the actual values of CC_Slave axis and CC_Master axis must never be greater than this value. Alarm 70010 is output if values are exceeded.

Monitoring is not active

- if the machine data is set to 0,
- if the coupling is switched off (open),
- if 'axis/spindle inhibit' is set for one of the axes,
- if an axis is in the 'follow-up mode',
- for the 1:-1 coupling.

If the offset stored at the instant of coupling activation changes when 1:1 coupling is active, the change is indicated at the NC => PLC VDI-SS:

DB31, ... DBX97.3 (MCS coupling: offset change)

Note

The offset can change.
- if the SW limit monitor was active for one axis during the main run,
- if one axis has been switched to follow-up mode,
- if collision protection was active for one axis.

20.2.4 Memory configuration: Block memory

The technological function requires **additional** data in the NC-internal block memory. The values must be increased for the following memory configuring channel-specific machine data:

- MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS += **1** (number of block elements for compile cycles)
- MD28100 $MN_MM_NUM_CC_BLOCK_USER_MEM += **1** (size of block memory for compile cycles (DRAM) in KB)

20.3 Description of collision protection

20.3.1 Defining protection pairs

A ProtecSlave axis (PSlave) is matched to its ProtecMaster (PMaster) axis via the following axial machine data:

MD63542 $MA_CC_PROTECT_MASTER (specifies the PMaster axis assigned to a PSlave axis)

The protection pairs can thus be defined independently of the coupling pairs. A PSlave axis may act as the PMaster axis for another axis. The axes must be either both rotary axes or both linear axes.

20.3.2 Switching collision protection ON / OFF

The minimum clearance between PSlave and PMaster is provided in the axial machine data of the PSlave axis:

MD63544 $MA_CC_COLLISION_WIN (collision protection window)

No collision protection is implemented if the value entered here is less than 0. The 0 position offset between PSlave and PMaster is provided in the axial machine data (PSlave axis):

MD63545 $MA_CC_OFFSET_MASTER (zero point offset between PSlave and PMaster)

Before the collision protection is switched on, the monitoring function for each individual axis must be enabled in the following machine data:

MD63543 $MA_CC_PROTECT_OPTIONS

In the same machine data for the PSlave axis, a setting is entered to specify whether the collision protection must be active continuously or whether it is activated via VDI interface signal (PLC => NC):

DB31, … DBX24.3

If collision protection is active, the setpoint positions of the PSlave and PMaster in the next IPO cycle are extrapolated and monitored in the IPO clock cycle using the current setpoint position and current velocity.

If the axes violate the minimum clearance, they are braked at the configured maximum acceleration rate:

MD32300 $MA_MAX_AX_ACCEL (Axis acceleration)

Or at a 20% faster acceleration rate, defined in machine data:

MD63543 $MA_CC_PROTECT_OPTIONS

An alarm is output as soon as the axes reach zero speed.

⚠ WARNING

Risk of collision when starting

If the axes are forced to brake, the positions displayed in the workpiece coordinate system are incorrect!

These are not re-synchronized again until a system RESET.

If the axes are already violating the minimum clearance when collision protection is activated, they can only be traversed in one direction (retraction direction). The retraction direction is configured in machine data:

MD63543 $MA_CC_PROTECT_OPTIONS

The collision protection status is optionally displayed in axial VDI-Out byte of the PSlave:

DB31, ... DBX66.0 (MCS coupling: collision protection active)

- DB31, … DBX66.0=1 → collision protection active
- DB31, … DBX66.0=0 → collision protection inactive

This output is activated via Bit7 in machine data of the PSlave axis:

MD63543 $MA_CC_PROTECT_OPTIONS

20.3.3 Configuring example

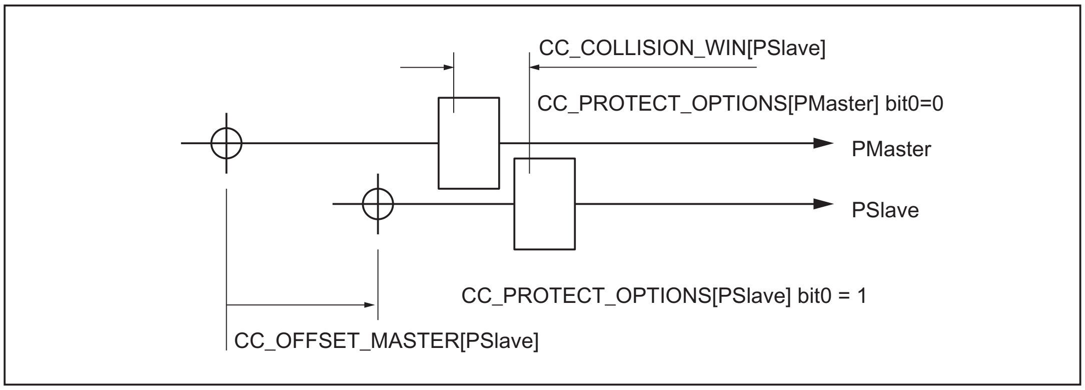

CC_COLLISION_WIN[PSlave]
CC_PROTECT_OPTIONS[PMaster] bit0=0
PMaster
PSlave
CC_PROTECT_OPTIONS[PSlave] bit0 = 1
CC_OFFSET_MASTER[PSlave]

Figure 20-2 Configuring example

Note

Since the collision protection function extrapolates the target positions from the "current velocity + maximum acceleration (or +20%)", the monitoring alarm may be activated unexpectedly at reduced acceleration rates:

Example:

PMaster = X, PSlave = X2, $MA_CC_COLLISION_WIN = 10mm

Starting point in part program: X=0.0 X2=20.0

```
N50 G0 X100 X2=90
```

; the monitoring alarm is activated because X and X2 are interpolating together: for this reason, the acceleration rate of X2 < maximum acceleration.

Remedy:

- `N50 G0 POS[X]=100 POS[X2]=90`
- or switch the monitoring function off.

20.4 User-specific configurations

Parking the machining head

In this context, "parking" means that the relevant machining head is not involved in workpiece machining. All of the axes are in closed-loop position control and are at an exact stop.

The coupling should be active, even if only one machining head is being used for production! This is essential primarily if only the second head (Y2....) is being for machining. "Axis/spindle inhibit" must then be set axially (PLC -> NC) for the "parked" head.

Note

When an axis/spindle inhibit is active, a part program can be executed if this axis is not operating under position control.

Spindle functionalities

Since an MCS coupling cannot be activated for spindles, other types of solutions should be configured for these.

- Positioning the spindle (`SPOS`= ...)
  Instead, a cycle is called from `SPOS`. `SPOS` is called for all active spindles in this cycle.
- Speed setpoint
  Speed and direction of rotation setpoints can be detected via synchronized actions or PLC and transferred to all other active spindles.
- Synchronous spindle function

20.5 Special operating states

Reset

The couplings can remain active after a `RESET`.

Reorg

No non-standard functionalities.

Block search

During a block search, the last block containing an OEM-specific language command is always stored and then output with the last action block. This feature is illustrated in the following examples. The output positions of the axes are always 0.

Example 1:

```
N01 M3 S1000
```

```
N02 G01 F1000 X10 Y10
N03 CC_COPON( X, Y)
TARGET:
```

If this program is started normally, axes X and Z traverse to `X10 Z10` in the decoupled state. After block search to `TARGET:` axes X and Y traverse to this position in the coupled state!

Example 2:

```
N01 M3 S1000
N02 CC_COPON( X)
N03 G01 F1000 X100 Y50
N04 CC_COPOFF( X)
N05 CC_COPON( Y)
N06 Y100
N10 CC_COPOFF()
TARGET:
```

After block search to `TARGET:` the axes traverse to `X100 Y100` in the decoupled state.

Example 3:

```
N01 CC_COPON( X, Y, Z)
N02 ...
...
N10 CC_COPOFF( Z)
TARGET:
```

After block search to `TARGET:` no coupling is active!

Single block

There are no nonstandard functionalities.

20.6 Boundary conditions

Validity

The function is configured only for the first channel.

Braking behavior

Braking behavior at the SW limit with path axes

The programmable acceleration factor `ACC` for braking at the SW limit corresponds to the path axes.

The axes in an MCS coupling are principal axes that are referred to as geometry axes due to their geometric arrangement.

Braking geometry axes using synchronized actions

The faster deceleration capacity as required for path axes can be implemented for geometry axes as follows using a synchronized action:
`ACC[x2]=190`

Axial acceleration limitation with G0

The following machine data is not considered by the MCS coupling:

MD32434 $MA_G00_ACCEL_FACTOR (scaling of the acceleration limitation with G0)

A value deviating from the standard value affects the braking ramp up to the software limit switch.

20.7 Data lists

20.7.1 Machine data

20.7.1.1 Channelspecific machine data

Number	Identifier: $MC_	Description
28090	NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles.
28100	NUM_CC_BLOCK_USER_MEM	Total size of usable block memory for compile cycles

20.7.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
63540	CC_MASTER_AXIS	Specifies the CC_Master axis assigned to a CC_Slave axis.
63541	CC_POSITION_TOL	Monitoring window
63542	CC_PROTEC_MASTER	Specifies the PMaster axis assigned to a PSlave axis.
63543	CC_PROTEC_OPTIONS	
63544	CC_COLLISION_WIN	Collision protection window
63545	CC_OFFSET_MASTER	Zero point offset between PSlave and PMaster

20.7.2 Signals

20.7.2.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
MCS coupling: deactivate or do not permit	DB31, …DBX24.2	-
MCS coupling: Activate collision protection	DB31, ... DBX24.3	-

20.7.2.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
MCS coupling: Collision protection active	DB31, ... DBX66.0	-
MCS coupling: Following axis	DB31, …DBX97.0	-
MCS coupling: Coupling active	DB31, …DBX97.1	-
MCS coupling: Mirroring active	DB31, …DBX97.2	-
MCS coupling: Offset change	DB31, …DBX97.3	-

TE7: Continue machining at the contour (retrace support) - 840D sl only 21

21.1 Brief description

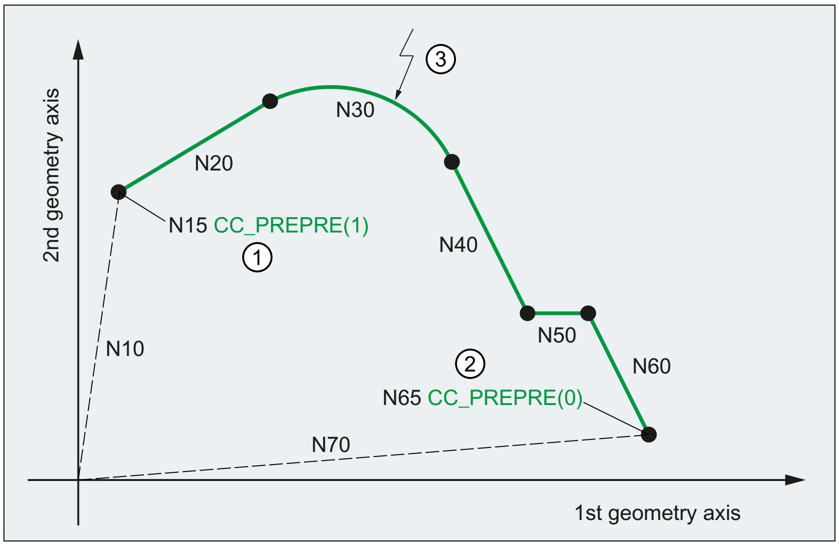
Function

The "Continue machining - Retrace support (RESU)" technological function supports the retracing of uncompleted 2-dimensional machining processes such as laser cutting, water jet cutting, etc.

In the event of a fault during the machining process, e.g. loss of the laser, RESU can be used even by machine operators who do not have specific knowledge of the active part program to interrupt machining and travel back along the contour from the interruption point to a program continuation point necessary for machining purposes.

After reaching the "continue machining" point, the machine operator triggers the re-machining. As part of the retrace process, an implicit block search takes place along the contour with calculation followed by repositioning on the contour and automatic retracing of the part program machining process.

The possibility of retrace support within a machining program is realized by programming RESU start and end points using the CC_PREPRE(...) procedure. The program continuation point can be arbitrarily selected within a contour range defined in this way.



① RESU start point or start of the RESU-capable contour range
② RESU end point or end of the RESU-capable contour range
③ Example of a program continuation point

Figure 21-1 Programmed contour with program continuation and interruption points

Precise retracing of contours is possible along all programmed contours comprising straight and circular elements. During retracing, other contour elements such as splines or automatically inserted non-linear contour elements (circle, parable, etc. e.g. through tool radius compensation) are mapped as straight lines between the start and end points of the corresponding contour element, thereby preventing precise retracing of contours.

Function code

The code for the "Continue Machining - Retrace support" technological function for function-specific identifiers of program commands, machine data, etc. is:

RESU (= REtrace SUpport)

Restrictions

The use of the "Continue Machining Retrace support" technological function is subject to the following restrictions:

- **The technological function is available only in the 1st channel of the NC.**
- Program continuation or reverse travel is only possible for part program blocks that contain traversing blocks in the configured RESU working plane (e.g. 1st and 2nd geometry axes of the channel, see Section "Definition of the RESU working plane (Page 865)").

References

The "Continue Machining - Retrace support" technological function is a compile cycle. For the handling of compile cycles (see Section "TE01: Installation and activation of loadable compile cycles (Page 707)").

21.2 Function description

21.2.1 Function

Block search with calculation on contour

To be able to resume interrupted machining at a specific point in a part program, a block search can be carried out using the "Block search with calculation on contour" standard function. However, detailed knowledge of the part program is required to be able to enter the block number of the part program block required for the block search (i.e. the number of the block the search needs to locate).

Continue machining - retrace support

The "Continue machining - Retrace support" technological function supports the continuation of the machining operation by means of an implicit block search with calculation on the contour without the machine operator needing to know the part program block required.

Continue machining might be required for example in a laser cutting application if the laser is lost during the machining operation and machining needs to resume at the point at which it was interrupted.

RESU supports continue machining (retrace support) using the following automatically running sub-functions:

- Function-specific reverse travel along the contour to the required program continuation point
- Automatic identification of the part program block associated with the program continuation point
- Block search with calculation on the contour for the part program block identified
- Repositioning on the contour at the program continuation point
- Continuation of part program machining

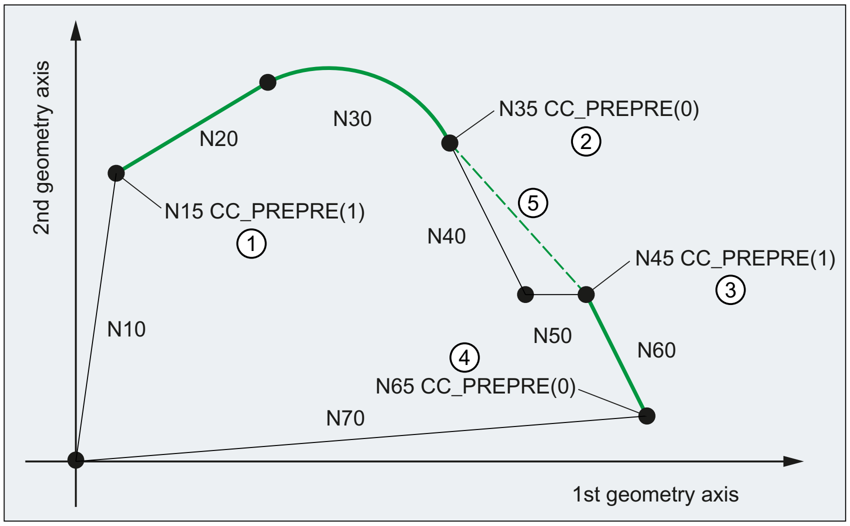
To approach the required program continuation point exactly, it is possible to switch several times between reverse and forward travel along the contour during the continue machining process.

RESU-capable contour ranges

RESU is activated by programming the function-specific part program command `CC_PREPRE (1)`. Only the contour range lying between the RESU start (`CC_PREPRE(1)`) and the interruption point (NC stop) is retraceable in the sense of RESU.

Once RESU has been launched, all part program blocks in which traversing movements are programmed are logged by RESU for possible subsequent reverse travel. Contour ranges for which continuing machining is irrelevant can be excluded from the log using RESU stop `CC_PREPRE (0)`.

Contour ranges, which are not logged are bridged by straight lines between the start and end points during reverse/forward travel.



①	RESU start point 1 or start of the RESU-capable contour range 1
②	RESU end point 1 or end of the RESU-capable contour range 1
③	RESU start point 2 or start of the RESU-capable contour range 2
④	RESU end point 2 or end of the RESU-capable contour range 2
⑤	RESU straight-line as equivalent contour of the **non** RESU-capable contour range

Figure 21-2 RESU-capable contour ranges

21.2.2 Definition of terms

Interruption point

The interruption point is the point of the contour at which the traversing movement comes to a standstill following an NC stop and reverse travel is activated.

Program continuation point

The program continuation point is the point of the contour at which reverse travel terminates and program continuation is activated.

Retraceable contour range

Retraceable contour ranges consist of traversing blocks in the configured RESU working plane (e.g. 1st and 2nd geometry axis of the channel), that are programmed in the part program between the RESU start command `CC_PREPRE(1)` and the RESU stop command `CC_PREPRE(0)` (see "Figure 21-2 RESU-capable contour ranges (Page 860)").

21.2.3 Functional sequence (principle)

The principle sequence of the RESU function between the interruption point, program continuation point and continuation of part program processing is described below.

Requirements

A part program with traversing blocks in the configured RESU working plane (Page 865) as well as the command for the RESU start has been started in the 1st channel.

Functional sequence

1. Interrupt part program processing:
   The part program processing / traversing movement may be interrupted any number of traversing blocks after RESU start by NC stop.
2. Select reverse travel:
   The selection of reverse travel is realized using the NC/PLC interface signal:
   DB21, … DBX0.1 = 1
3. Reverse travel:
   The contour is traversed in the RESU working plane (Page 865) in the reverse direction with NC start. Instead of the current machining program, RESU selects the automatically generated RESU main program (Page 872).
4. End reverse travel:
   Once the required program continuation point on the contour has been reached, reverse travel is ended using NC stop.
5. Select forward travel (optional):
   For forward travel, reverse travel must be deselected via NC/PLC interface signal:
   DB21, … DBX0.1 = 0
6. Forward travel (optional):
   The contour is traversed in the RESU working plane (Page 865) in the forward direction with NC start.
7. End forward travel (optional):
   Once the required program continuation point on the contour has been reached, forward travel is ended using NC stop.

8. Retrace support:
   Continue machining is initiated via PLC interface signal:
   DB21, … DBX0.2 = 1 (start retrace support)
   For retrace support, RESU automatically selects the original machining program and launches a block search with calculation as far as the program continuation point.
9. Continuation of part program machining:
   Part program processing continues at the program continuation point in accordance with the "Block search with calculation" standard function when two NC start commands occur one after the other.
   The first NC start command processes the action blocks. Retrace support ASUP CC_RESU_BS_ASUP.SPF (Page 875) is initiated when the last action block is reached:
   DB21, … DBX32.6 == 1 (last action block active)
   The second NC start command processes the approach block before part program processing is resumed.

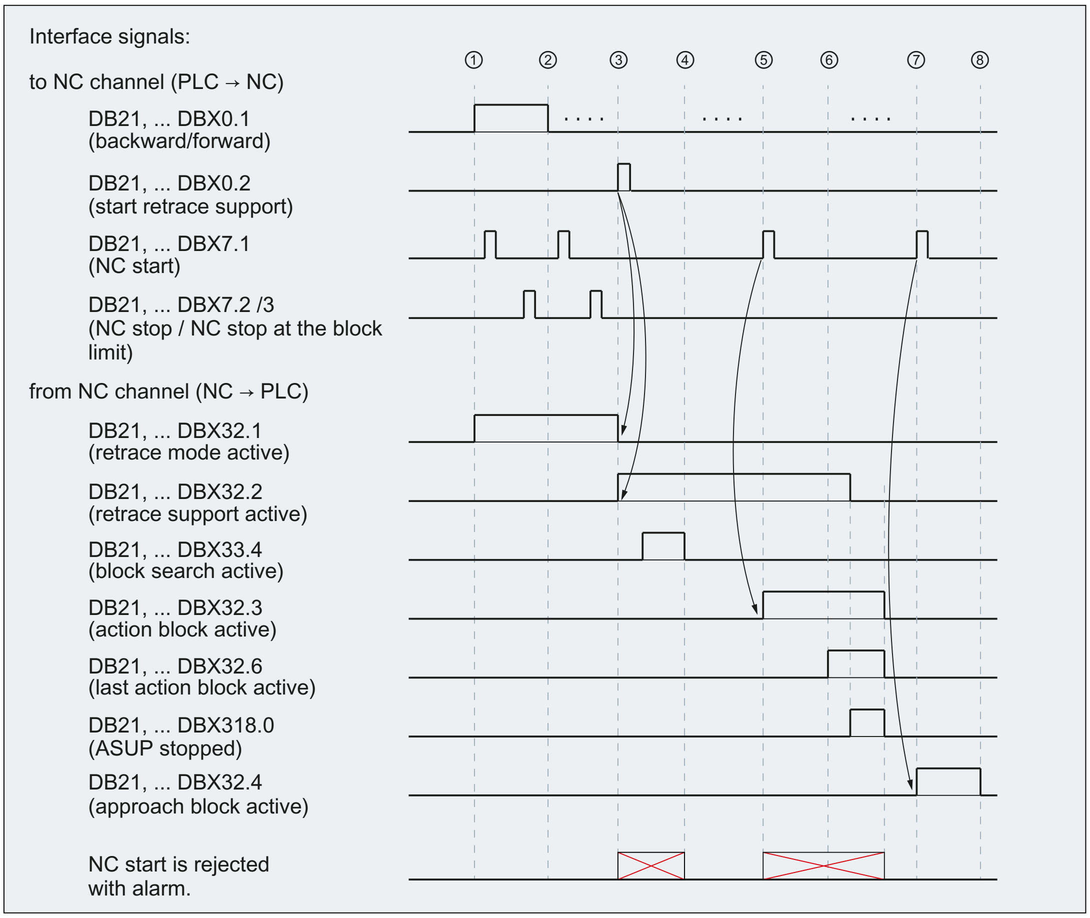
Note

Points 3 to 8 can be repeated as often as required.

Following retrace support, a new reverse travel is possible up to a maximum of the last program continuation point.

Signal characteristic of the NC/PLC interface signals

The principle sequence of the RESU function is illustrated in the following figure as a signal characteristic of the NC/PLC interface signals involved:


Interface signals:
to NC channel (PLC → NC)
DB21, ... DBX0.1 (backward/forward)
DB21, ... DBX0.2 (start retrace support)
DB21, ... DBX7.1 (NC start)
DB21, ... DBX7.2 /3 (NC stop / NC stop at the block limit)
from NC channel (NC → PLC)
DB21, ... DBX32.1 (retrace mode active)
DB21, ... DBX32.2 (retrace support active)
DB21, ... DBX33.4 (block search active)
DB21, ... DBX32.3 (action block active)
DB21, ... DBX32.6 (last action block active)
DB21, ... DBX318.0 (ASUP stopped)
DB21, ... DBX32.4 (approach block active)
NC start is rejected with alarm.
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

① Reverse travel is started.
② Forward travel is started (optional).
③ Continue machining is started (block search).
④ Search destination (target block) was found.
⑤ 1st NC START → Action blocks are output
⑥ Last action block is activated.
The RESU ASUP CC_RESU_BS_ASUP.SPF (Page 875) is triggered when the last action block is activated.
⑦ 2nd NC start → the approach block to the program continuation point is executed.
⑧ Part program processing (target block) resumed

Figure 21-3 Signal chart

21.2.4 Maximum retraceable contour area

In multiple machining continuation within a contour area, the reverse travel on the contour is always possible only up to the last machining continuation point (**W**). In first-time reverse travel after RESU start, reverse travel up to the start of the contour range is possible.

This response must be illustrated via the following graph. For the sake of simplicity, the interruption point (**U**) is always the same:

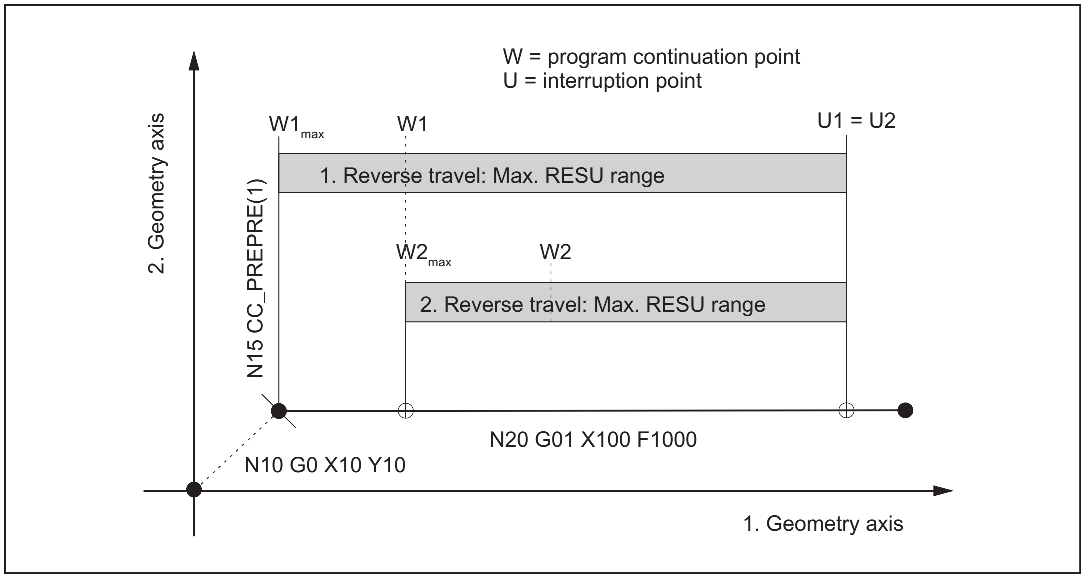


Figure 21-4 Maximum Retraceable contour range

1st reverse travel

In first-time reverse travel, reverse travel is possible maximum up to the start of the first contour element (`N20`) after RESU start (`N15`) ($W1_{max}$).

If reverse travel takes place up to the machining continuation point W1, W1 defines the maximum RESU range for a possible further reverse travel after machining continuation and forward travel.

2nd reverse travel

A renewed reverse travel can now be undertaken maximum only up to the last machining continuation point $W2_{max =}$ W1.

If reverse travel goes as far back as program continuation point W2, the maximum RESU range is restricted further.

21.3 Startup

21.3.1 Activation

Before commissioning the technological function, ensure that the corresponding compile cycle has been loaded and activated (see also Section "TE01: Installation and activation of loadable compile cycles (Page 707)").

Activation

The "Continue machining - Retrace support" technology function is activated via the following machine data:

MD60900+i $MN_CC_ACTIVE_IN_CHAN_RESU[0], bit 0 = 1

Note

The "Continue machining - Retrace support" technological function is available only in the **1st channel** of NC.

21.3.2 Definition of the RESU working plane

Machining continuation/reverse travel is possible only for part program blocks that contain traversing blocks in the configured RESU working plane.

The RESU working plane is defined with the following machine data:

MD62580 $MC_RESU_WORKING_PLANE

Value	Meaning
1	The RESU working plane is formed by the **1st** and **2nd** Geometry axes of the 1st channel (for `G17`).
2	The RESU working plane is formed by the **1st** and **3rd** Geometry axes of the 1st channel (for `G18`).
3	The RESU working plane is formed by the **2nd** and **3rd** Geometry axes of the 1st channel (for `G19`).

21.3.3 Memory configuration: Block memory

The technological function requires **additional** data in the NC-internal block memory. The values must be increased for the following memory configuring channel-specific machine data:

- MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS += **4** (number of block elements for compile cycles)
- MD28100 $MN_MM_NUM_CC_BLOCK_USER_MEM += **20** (size of block memory for compile cycles (DRAM) in KB)

21.3.4 Memory configuration: Heap memory

Memory requirements

RESU requires compile cycles heap memory for the following function-specific buffers:

- Block buffer
  The larger the block buffer (see "Figure 21-6 RESU-specific part programs (Page 871)"), the larger the number of part program blocks that can be traversed backwards.
  32 bytes are required per part program block.
  The block buffer can be parameterized directly.
- Block search buffer
  The block search buffer contains the information required for processing subprogram searches in the context of RESU.
  180 bytes are required for each subprogram. The block search buffer requires at least 2880 bytes (16 subprogram calls with 180 bytes each).
  The block search buffer cannot be parameterized directly.
  A function-specific GUD variable displays the size of the block search buffer (for creation of GUD variables, see Section "Channel-specific GUD variables (Page 880)").

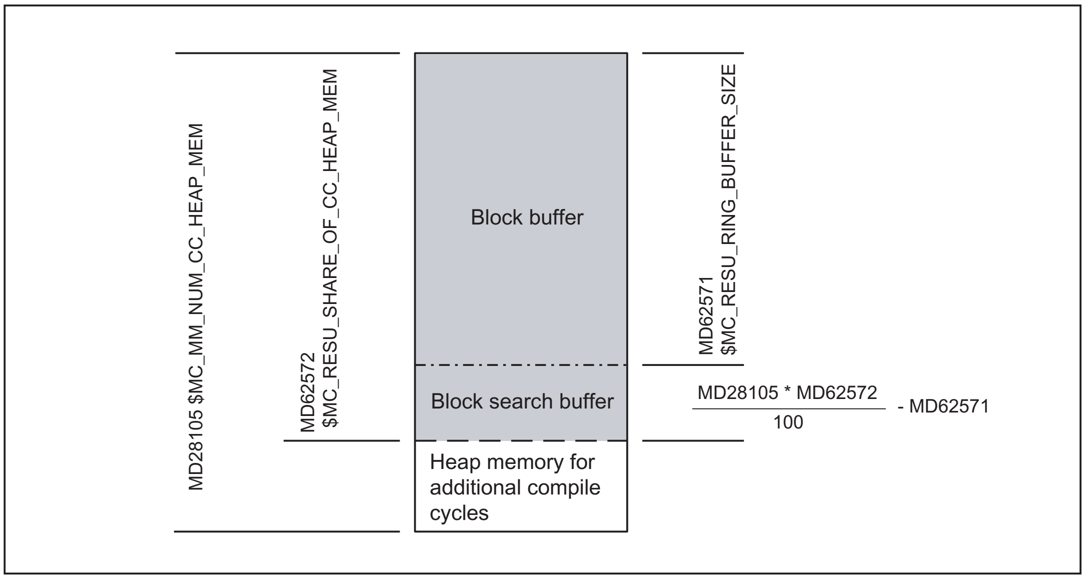

MD28105 $MC_MM_NUM_CC_HEAP_MEM
MD62572 $MC_RESU_SHARE_OF_CC_HEAP_MEM
Block buffer
Block search buffer
Heap memory for additional compile cycles
MD62571 $MC_RESU_RING_BUFFER_SIZE
MD28105 * MD62572 / 100 - MD62571

Figure 21-5 Division of the heap memory for compile cycles

Memory configuration

Size of the compile cycle heap memory

The size of the heap memory in KB that can be used by the user for compile cycles is defined via the memory configuring channel-specific machine data:

MD28105 $MC_MM_NUM_CC_HEAP_MEM

For RESU, the already existing machine data value (x) is adjusted as follows:

MD28105 $MC_MM_NUM_CC_HEAP_MEM = x + **50**

Size of the block buffer

The size of the block buffer is adjusted via the machine data:

MD62571 $MC_RESU_RING_BUFFER_SIZE

Default setting:

MD62571 $MC_RESU_RING_BUFFER_SIZE = **1000**

RESU portion of the total heap memory

The RESU portion of the total heap memory that can be used for compile cycles is set via the machine data:

MD62572 $MC_RESU_SHARE_OF_CC_HEAP_MEM

Default setting:

MD62572 $MC_RESU_SHARE_OF_CC_HEAP_MEM = 100

Error messages

The block search buffer requires at least 2880 bytes (corresponding to 16 subprogram calls with 180 bytes each). Otherwise, the following alarm will be generated during NC power-up:

Alarm 75600 "Channel 1 Retrace Support: Incorrect MD configuration, error no. 5"

If the block search buffer is not big enough during operation, the following alarm appears:

Alarm 75606 "Channel 1 retraceable contour shortened"

21.3.5 RESU main program memory area

Memory configuration

The storage location of the RESU main program CC_RESU.MPF can be set with the following machine data (see Section "Main program (CC_RESU.MPF) (Page 872)"):

MD62574 $MC_RESU_SPECIAL_FEATURE_MASK (additional RESU features)

Bit	Value	Meaning
1	0	The RESU main program is stored in the **dynamic NC memory** (default).
	1	The RESU main program is stored in the **static NC memory** (default).

Storage in the dynamic NC memory (default)

If the RESU main program is created in the dynamic NC memory, the memory area available to the user for file storage must be increased as follows:

MD18351 $MN_MM_DRAM_FILE_MEM_SIZE = x[1)] + **100**

[1)] already available machine data value

Storage in the static NC memory

If the RESU main program is created in the static memory area of the NC, it is retained even after a POWER OFF. However, since RESU regenerates the RESU main program every time the retrace support function is used, this parameter setting is not recommended.

21.3.6 Storage of the RESU subroutines

Storage as user or manufacturer cycles

The following RESU-specific subroutines can be stored as user cycles or manufacturer cycles:

- INI program: CC_RESU_INI.SPF
- END program CC_RESU_END.SPF
- Retrace support ASUB CC_RESU_BS_ASUP.SPF
- RESU ASUB CC_RESU_BS_ASUP.SPF

The setting is done using machine data:

MD62574 $MC_RESU_SPECIAL_FEATURE_MASK (additional RESU features)

Bit	Value	Meaning
2	0	The RESU-specific subroutines are stored as **user cycles** (default).
	1	The RESU-specific subroutines are stored as **manufacturer cycles**.

Series commissioning

Due to the default setting of MD62574 Bit 2, the RESU-specific subroutines along with their contents are stored as user cycles the first time the NC is powered up after the activation of the technological function.

If the setting is then changed to specify that the RESU-specific subroutines are to be stored as manufacturer cycles, the RESU-specific subroutines already created as user cycles are retained even after a new power-up and must be deleted.

As support for series commissioning, the RESU-specific subroutines present as user cycles can be deleted without prompting when the NC is powered up:

MD62574 $MC_RESU_SPECIAL_FEATURE_MASK, Bit 3 = 1

21.3.7 ASUB enable

Note

A requirement for using ASUBs is that the "Cross-mode actions" option must be available.

The following machine data must be set for the start enable for the RESU-specific ASUB CC_RESU_ASUP.SPF while the channel is in the NC STOP state:

MD11602 $MN_ASUP_START_MASK, bit 0 = 1 (ignore stop reason for ASUB)

MD11604 $MN_ASUP_START_PRIO_LEVEL = 1 (priorities from which MD11602 is effective)

21.3.8 PLC user program

Requirements

The following functionality is necessary for the sequential coordination of the RESU function in the PLC user program:

IF	DB21, … DBX32.2	"Retrace support active" == 1
THEN	DB21, … DBX0.1	"Forward/Reverse" = 0
	DB21, … DBX0.2	"Start retrace support" = 0

IF	DB11, … DBX0.7	"Mode group reset" == 1
OR	DB21, … DBX7.7	"Reset" == 1
THEN	DB21, … DBX0.1	"Forward/Reverse" = 0
	DB21, … DBX0.2	"Start retrace support" = 0

The following signals should be reset for safety reasons:

IF	DB21, … DBX0.2	"Start retrace support" == 1
THEN	DB21, … DBX0.1	"Forward/Reverse" = 0

IF	DB21, … DBX0.1	"Forward/Reverse" == 1
THEN	DB21, … DBX0.2	"Start retrace support" = 0

Program example

The following program extract implements the changes described above:

```
U      DB21, … DBX32.2          // IF         "Retrace support active" == 1
R      DB21, … DBX0.1           // THEN       "Forward/Reverse" = 0
R      DB21, … DBX0.2           //            "Start retrace support" = 0
O      DB11, … DBX0.7           // IF         "Mode group reset" == 1
O      DB21, … DBX7.7           // OR         "Reset" == 1
R      DB21, … DBX0.1           // THEN       "Forward/Reverse" = 0
R      DB21, … DBX0.2           //            "Start retrace support" = 0
U      DB21, … DBX0.2           // IF         "Start retrace support" == 1
R      DB21, … DBX0.1           // THEN       "Forward/Reverse" = 0
```

```
U      DB21, … DBX0.1           // IF         "Forward/Reverse" == 1
R      DB21, … DBX0.2           // THEN       "Start retrace support" = 0
```

21.4 Programming

21.4.1 RESU Start/Stop/Reset (CC_PREPRE)

Using procedure `CC_PREPRE` (**Prep**are **Re**trace) the "Start", "Stop" and "Reset" modes can be activated for retracing:

Syntax

```
CC_PREPRE(<mode>)
```

Meaning

`CC_PREPRE`:	Retrace mode	
	Alone in the block:	Yes
`<mode>`:	Mode	
	Data type:	INT
	Value range:	-1, 0, 1
	Value	**Meaning**
	-1	Starts logging the traversing blocks. The information required for traversing backward is logged on a block-specific basis in a RESU-internal block buffer. The traversing information is related to the two geometry axes of the RESU working plane e.g. the 1st and 2nd geometry axes of the channel: MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[x]; with x = 0 and 1 Or, if transformation is active: MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB[x]; with x = 0 and 1
	0	Stops logging the traversing blocks. Thus irrelevant contour areas can be excluded from the log. These excluded contour areas are bridged by a straight line between the starting point and end point when traversing backward.
	1	Deactivates logging of the traversing blocks and deletes the function-internal block buffer. Contour areas located before the instant of deactivation of the part program are therefore no longer available to RESU.

RESET response

For the following reset events, `CC_PREPRE(-1)` is implicitly executed:

- Power on (warm restart)
- Channel reset
- End of program reset (M30)

21.5 RESU-specific part programs

21.5.1 Overview

The following programs are automatically generated within the scope of the retrace support functionality (continue machining). User-specific content can be incorporated in the programs listed.

Function	Program name	Can be changed
Main program	CC_RESU.MPF	No
INI program	CC_RESU_INI.SPF	Yes
END program	CC_RESU_END.SPF	Yes
Continue machining ASUP	CC_RESU_BS_ASUP.SPF	Yes
RESU ASUP	CC_RESU_ASUP.SPF	No

Internal structure

The following diagram provides an overview of the internal structure of the technological function and the relationship between the various part programs.

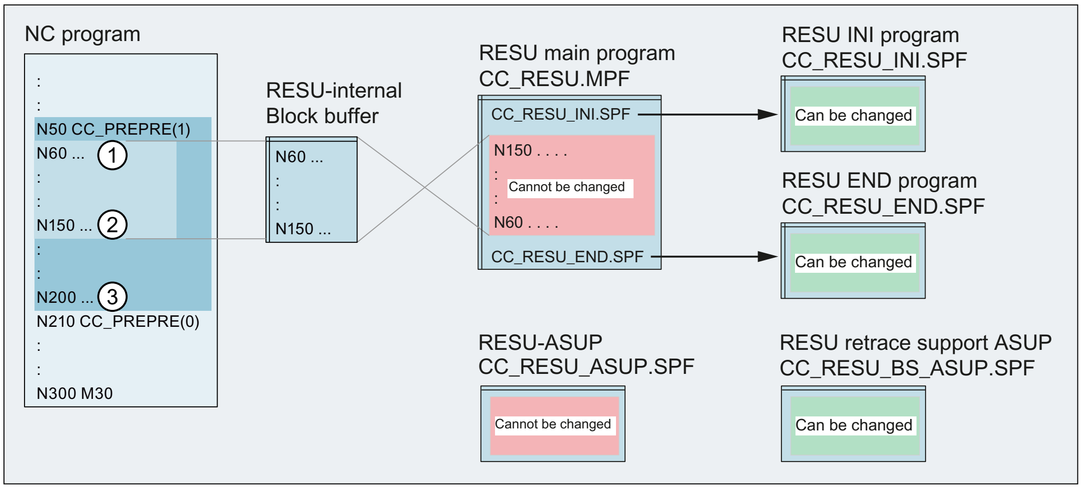


Figure 21-6 RESU-specific part programs

21.5.2 Main program (CC_RESU.MPF)

Function

In addition to the calls for the RESU-specific subprograms, the RESU main program "CC_RESU.MPF" contains the traversing blocks generated from the traversing blocks logged in the block buffer for reverse/forward travel along the contour. The program is always regenerated by the RESU function if, once the part program has been interrupted, the status of the following interface signal changes:

DB21, … DBX0.1 (Reverse / Forward)

Note

CC_RESU.MPF must not be changed. User-specific adjustments are to be made in the corresponding RESU-specific subprograms.

Error messages

By default, RESU generates traversing blocks for the entire retraceable contour range logged in the block buffer. If there is not enough memory for all traversing blocks to be generated in the parameterized memory area of the RESU main program (see Section "RESU main program memory area (Page 867)"), RESU reduces the number of generated traversing blocks.

The lacking memory and/or reduction in the number of the generated traversing blocks is indicated by an alarm:

RESU alarm 75608 "Channel *number* NC memory limit reached, RAM type *type*"

If the RESU main program is created in the static user memory, the following system alarm appears at the same time as the RESU alarm:

Alarm 6500 "NC memory full"

Note

If the number of traversing blocks generated is reduced due to insufficient memory, the entire retraceable contour can still be retraced for retrace support. To do this, proceed as follows:

- Reverse travel up to the end of the RESU main program
- Two-time change of the interface signal:
  DB21, … DBX0.1 (Reverse/Forward)

Using the current position as a new interruption point enables RESU to generate a new RESU main program.

Subsequently, travel is possible as far as the end of the retraceable contour range or, if the limits have changed, as far as the starting point of the last traversing block that can be generated.

The procedure described can be repeated as many times as required both for reverse and forward travel.

21.5.3 INI program (CC_RESU_INI.SPF)

Function

The RESU-specific subprogram "CC_RESU_INI.SPF" contains the defaults required for the reverse travel:

• Metric input system:	`G71`
• Absolute dimensions:	`G90`
• To switch off the adjustable zero-point offsets / frames (see Chapter "Frames (Page 885)"):	`G500`
• To switch off the active tool offsets (see Chapter "Tool offsets (Page 885)"):	`T0`
• Switching off the tool radius offset:	`G40`
• Traversing velocity:	`F200`

Program structure

CC_RESU_INI.SPF has the following content by default:

```
PROC CC_RESU_INI
```

```
        G71 G90 G500 T0 G40 F200
        ;system frames that are present are deactivated
        ;actual value and scratching
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H01'
        $P_SETFRAME = ctrans()
        endif
        ;external zero point offset
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H02'
        $P_EXTFRAME = ctrans()
        endif
        ;tool carrier
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H04'
        PAROTOF
        endif
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H08'
        TOROTOF
        endif
        ;workpiece reference points
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H10'
        $P_WPFRAME = ctrans()
        endif
        ;cycles
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H20'
        $P_CYCFRAME = ctrans()
        endif
        ;transformations
        if $MC_MM_SYSTEM_FRAME_MASK B_AND 'H40'
        $P_TRAFRAME = ctrans()
        endif
        ; bit mask for global basic frames
        $P_NCBFRMASK = 0
        ;bit mask for channel-specific basic frames
        $P_CHBFRMASK = 0
        ;programmable frame
        $P_PFRAME = ctrans()
M17
```

CAUTION

Program changes

By changing the content of the RESU-specific subprogram "CC_RESU_INI.SPF", the user (machine manufacturer) takes over the responsibility for the correct sequence of the technological function.

Note

CC_RESU_INI.SPF may be changed.

CC_RESU_INI.SPF must not contain any RESU part program commands. `CC_PREPRE(x)`.

21.5.4 END program (CC_RESU_END.SPF)

Function

The task of the RESU-specific subroutine "CC_RESU_END.SPF" is to stop reverse travel once the end of the retraceable contour is reached. If the RESU function is parameterized appropriately, this scenario will not arise under normal circumstances.

Program structure

CC_RESU_END.SPF has the following content by default:

```
PROC CC_RESU_END
        M0
M17
```

⚠ CAUTION

Program changes

By changing the content of the RESU-specific subroutine "CC_RESU_END.SPF", the user (machine manufacturer) takes over the responsibility for the correct sequence of the technological function.

Note

CC_RESU_END.SPF may be changed.

CC_RESU_END.SPF must not contain any RESU part program commands. `CC_PREPRE(x)`.

21.5.5 Retrace support ASUB (CC_RESU_BS_ASUP.SPF)

Function

The NC is forced to approach the current path point during machining continuation with the help of the RESU-specific ASUB "CC_RESU_BS_ASUP.SPF":

• Reapproach next point on path:	`RMN`
• Approach along line on all axes:	`REPOSA`

Program structure

CC_RESU_BS_ASUP.SPF has the following content by default:

```
PROC CC_RESU_BS_ASUP SAVE
        RMN
        REPOSA
```

```
M17
```

⚠ CAUTION

Program changes

By changing the content of the RESU-specific subroutine "CC_RESU_BS_ASUP.SPF", the user (machine manufacturer) takes over the responsibility for the correct sequence of the technological function.

Note

CC_RESU_BS_ASUP.SPF may be changed.

User-specific modifications must be inserted before the part program block `RMN`.

21.5.6 RESU ASUB (CC_RESU_ASUP.SPF)

Function

The RESU-specific ASUB "CC_RESU_ASUP.SPF" is required internally for the function. The ASUB is initiated if the following RESU interface signal is switched over in the NC stop state:

DB21, … DBX0.1 (Forward/Reverse)

Program structure

CC_RESU_ASUP.SPF has the following content:

PROC CC_RESU_ASUP	
	; siemens system asub - do not change G4 F0.001 M0 REPOSA
M17	

Note

CC_RESU_ASUP.SPF must not be changed.

21.6 Retrace support

21.6.1 General information

Retrace support refers to the entire operation from initiation of retracing through the interface signal DB21, … DBX0.2 = 1 (start machining continuation) up to to the continuation of the part program processing of the programmed contour.

Requirement

In order for retrace support to function, the retrace mode, launched by means of the request for reverse travel, must be active in the channel:

DB21, … DBX32.1 = 1 (retrace mode active)

(See Section "Functional sequence (principle) (Page 861)")

Subfunctions

The two essential subfunctions of retrace support are the standard NC functions:

- Block search with calculation on contour
- Repositioning on the contour via shortest route (`REPOS RMN`)

21.6.2 Block search with calculation on contour

Function

The block search with calculation on the contour initiated implicitly by RESU within the framework of the retrace support has the following tasks:

- Set the program pointer on the part program block on which repositioning was done with the help of Reverse / Forward travel
- Calculates the axis positions on the basis of the programmed traversing blocks from the start of the part program to the target block
- Collates the instructions programmed from the start of the part program to the target block, which are executed in the action block. These include:
  – Auxiliary functions
  – Tool change
  – Spindle functions
  – Feedrate programming

All part program instructions which are not executed in the action block but are required for retrace support in the part program must be entered manually in the RESU-specific retrace support ASUB CC_RESU_BS_ASUP.SPF, e.g.:

- Synchronized actions
- M functions

References

The complete description of the Block search is available in:
Basic Functions Function Manual; Mode Group, Channel, Program Mode (K1), Program test

21.6.3 Reposition

Function

Following the end of the last action block (last traversing block before repositioning), NC Start outputs the approach block for repositioning all channel axes programmed in the part program as far as the target block.

Geometry axes

In the approach block, the geometry axes of the RESU working plane (e.g. 1st and 2nd geometry axes of the channel) traverse the shortest route along the contour to the program continuation point.

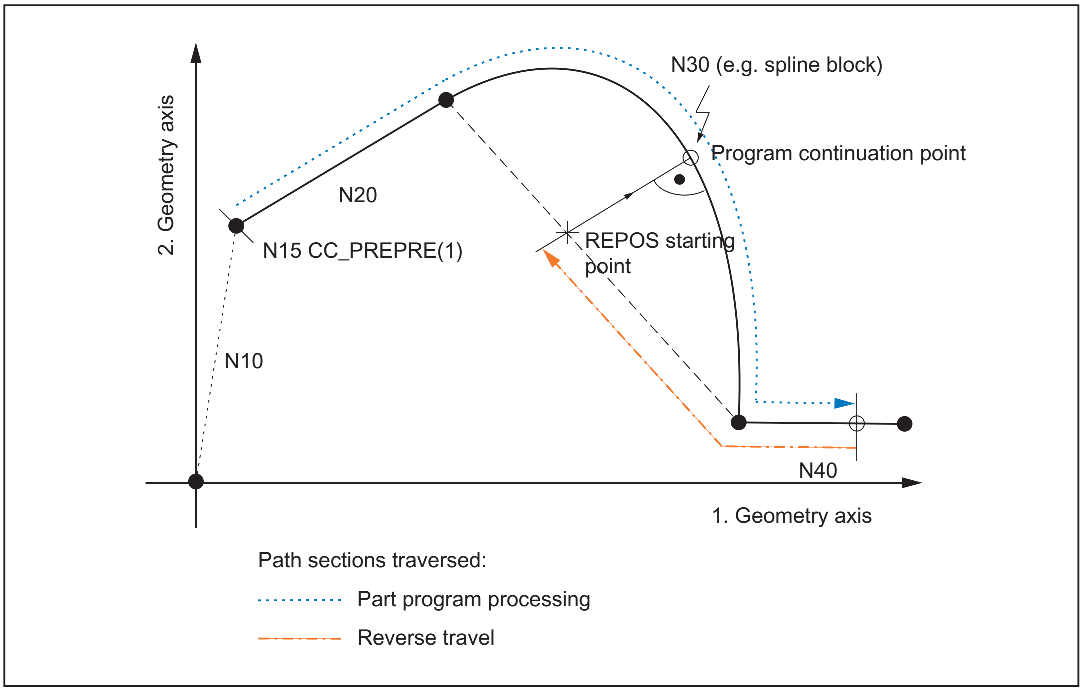


Figure 21-7 Retraceable contour ranges and REPOS

Channel axes

All other channel axes programmed in the part program travel to the relevant position calculated in the block search.

21.6.4 Temporal conditions concerning NC start

NC start should be initiated twice by the machine operator within the framework of continue machining (see Section "Functional sequence (principle) (Page 861)").

The following conditions must be met:

- In NC start for outputting the action blocks:
  - The block search must be completed:
    DB21, … DBX33.4 = 0 (block search active)
- In NC start for outputting the approach blocks:
  - The RESU ASUB "CC_RESU_BS_ASUP" must be completed:
    DB21, … DBX318.0 = 1 (ASUB stopped)

21.6.5 Block search from last main block

The block search with calculation on the contour executed within the framework of the machining continuation via the use of the most powerful NCU in very large part programs can itself lead to computation times of several minutes up to the reaching of the target block.

A significant reduction of this waiting period is possible through the use of the "Block search from the last main block".

Functionality

For retrace support with block search from the last main block, the search for the target block takes place in 2 stages:

1. Block search without calculation from start of machining program to last main block before target block. Subprograms are ignored during this search, i.e. it takes place exclusively in the main program.
2. Block search with calculation on contour from main block to target block. This block search does not ignore subprograms.

Requirement

To enable a search from last main block for retrace support, at least one main block must be programmed after the RESU start `CC_PREPRE(1)`.

Main block

All instructions required for processing the subsequent section of the part program must be programmed in one main block.

The main blocks are to be designated with a Main Block No. consisting of the sign ":" and a positive whole number (block number).

References:
Programming Manual, Fundamentals; Fundamentals of NC Programming, Language Elements of the Programming Language

Activation

Activation of the block search from the last main block is performed using the RESU-specific machine data:

MD62575 $MC_RESU_SPECIAL_FEATURE_MASK_2, bit 0 (additional RESU features)

Bit	Value	Meaning
0	0	Retrace support is performed using block search with calculation on contour.
	1	Retrace support is performed using block search from the last main block.

Supplementary conditions

In order that a new retrace support operation can take place following a retrace support operation with block search from last main block, the RESU start `CC_PREPRE(1)` must be programmed in the retrace support ASUB "CC_RESU_BS_ASUP.SPF".

Programming example:

```
PROC CC_RESU_BS_ASUP SAVE
        ; (synchronized actions, M functions, etc. required for retrace support
        )
        CC_PREPRE(1)
        RMN
        REPOSA
M17
```

21.7 Function-specific display data

21.7.1 Channel-specific GUD variables

As display data for the size of the block search buffer, RESU provides the following channel-specific GUD variables:

GUD variable	Meaning	Value	Access
CLC_RESU_LENGTH_BS_BUFFER	Size of block search buffer	Byte	read only

After the startup of the technological function, the GUD variable is not displayed automatically on the operator panel.

Creating and displaying the GUD variable

Perform the following steps to create and display the GUD variables in the operator panel:

1. Set password:
   The password of protection level 1 (machine manufacturer) is to be input.
2. Activate the "Definitions" display
3. The file is recreated if an "SGUD.DEF" file is still not available:
   Name: SGUD
   Type: Global data /system
   Confirm entry with **OK**.
   This opens the file in the editor.
4. Edit the GUD variable definitions:
   ```
   DEF CHAN REAL CLC_RESU_LENGTH_BS_BUFFER
   M30
   ```
5. Save the file and close the editor.
6. Activate the "SGUD.DEF" file.

The GUD variable is not displayed on the operator panel.

Note

The new GUD variable, which is already being displayed, will be detected by the RESU function and supplied with an up-to-date value only following an NC POWER ON Reset. As a consequence, an NC POWER ON must be initiated after the creation.

References

The exact procedure to be following while creating and displaying GUD variables depends on the software version of the existing operator panel and is described in the:

- Operating Manual

21.8 Function-specific alarm texts

For details of the procedure for creating function-specific alarm texts, see Section "Creating alarm texts (Page 713)".

21.9 Boundary conditions

21.9.1 Function-specific supplementary conditions

21.9.1.1 Continue machining within subprograms

Subprogram call outside or inside a program loop

Clear retrace support within subprograms depends on whether the subprogram call is made outside or inside a program loop:

- Outside
  Clear retrace support is possible if a subprogram is called outside a program loop.
- Inside
  Clear retrace support may not be possible if a subprogram is called inside a program loop (see Section "Continue machining within program loops (Page 882)").

Number of passes P

Subprogram repetitions using number of passes P are taken into account for retrace support. This means that retrace support is performed in the part program with the correct reference to the part program block and number of passes P to the program continuation point of the contour.

21.9.1.2 Continue machining within program loops

In NC high-level language, program loops can be programmed using:

- `LOOP` `ENDLOOP`
- `FOR` `ENDFOR`
- `WHILE` `ENDWHILE`
- `REPEAT` `UNTIL`
- `CASE/IF-ELSE-ENDIF` in conjunction with `GOTOB`

⚠ WARNING

Risk of collision

Significant contour deviations can occur during the machining if the programmed continuation point at the programmed contour is the result of a loop run that is not equal to the first loop run.

If retrace support is performed within program loops, the retrace support is always effective in the first loop run.

21.9.1.3 Machining continuation on full circles

In full circles, the block starting and end points coincide at one contour point. As no clear differentiation is possible in this case, one always starts from the block start point during machining continuation on this kind of a contour point. The first part program block following retrace support is then the circular block.

A contour point just before the block end point of the circular block must be selected to avoid traversing the circular block after the machining continuation.

21.9.1.4 Automatically generated contour elements

The automatic generation of non-linear / non-circular contour elements by the NC takes place, for example, when programming the following NC functions in the part program:

- `RND`
- `G641/G642`
- Tool radius compensation

For reverse / forward travel as part of RESU these contour elements can be replaced by straight lines between the start and end of the block.

21.9.2 Supplementary conditions for standard functions

21.9.2.1 Axis replacement

As long as RESU is active, the two geometry axes of the RESU working plane (e.g. the 1st and 2nd geometry axes of the channel) must not be transferred to another channel via axis replacement (`RELEASE(x)/GET(x)`).

The RESU activity:

- Starts:
  – with the part program command `CC_PREPRE(1)`
- Ends with:
  – the program end
  or
  – with the part program command `CC_PREPRE(-1)`

21.9.2.2 Traversing movements of the channel axes

Other channel axes, except the two geometry axes of the RESU working plane, are not considered by RESU.

If traversing movements in other channel axes are required for machining continuation or reverse travel, these can either be undertaken by the machined operator manually, or programmed as travel block in the RESU-specific subroutine "CC_RESU_INI.SPF".

⚠ WARNING

Risk of collision when traversing

The machine operator must ensure that collision of the associated traversing movements does not take place during the entire machining continuation operation within the framework of the technological function of RESU.

21.9.2.3 Block numbers

The following RESU-specific subroutines and their subroutines must not contain block numbers:

- CC_RESU_INI.SPF
- CC_RESU_END.SPF

The following alarm appears in the event of an error:

Alarm 75604 "Reverse travel not possible, error no. *number*"

21.9.2.4 Block search

Block search with calculation

RESU is subject to the following constraints in the context of the "block search with calculation (on contour/at end of block") standard function:

- The last block of the RESU part program command `CC_PREPRE(x)` run during the block search is effective in the target block.
- The retraceable contour range starts with the `REPOS` approach block.

Block search without calculation

RESU part program commands `CC_PREPRE(x)` have**no** effect in the "block searches without calculation" function.

21.9.2.5 Transformations

RESU can also be used for active kinematic transformation (e.g. 5-axis transformation) subject to restrictions, as the traversing movements of the two geometry axes of the RESU working plane are recorded in the basic coordinate system (BCS) and therefore before the transformation (see also Section "TE4: Handling transformation package - 840D sl only (Page 793)").

Transformation changeover

While RESU is active, no transformation changes are permitted to take place and transformation must not be activated/deactivated.

The RESU activity:

- Starts:
  - with the part program command `CC_PREPRE(1)`
- Ends with:
  - the program end
    or
  - with the part program command `CC_PREPRE(-1)`

References

A full description of the transformations is available in:
Function Manual, Extended Functions; Kinematic Transformation (M1)

21.9.2.6 Compensations

RESU can be used in interaction with compensations, because the traversing movements of the two geometry axes of the RESU working plane is recorded in the basic coordinate system (BCS) and therefore before the compensation.

A full description of the compensations can be found in:
References:
Function Manual, Extended Functions; Compensations (K3)

21.9.2.7 Frames

RESU can be used in conjunction with frames.

However, as the traversing movements of the two geometry axes of the RESU working plane are recorded in the basic coordinate system (BCS) and therefore after the frames have been taken into account, the frame offsets must be deactivated during retrace support (reverse / forward travel).

The frame offsets are deactivated during retrace support via the standard default settings of the RESU-specific subprogram "CC_RESU_INI.SPF" (see Section: "INI program (CC_RESU_INI.SPF) (Page 873)").

A full description of the frames can be found in:
Reference:
Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2)

21.9.2.8 Tool offsets

RESU can be used in conjunction with tool offsets.

However, as the traversing movements of the two geometry axes of the RESU working plane are recorded in the basic coordinate system (BCS) and therefore after the tool offsets have been taken into account, the tool offsets must be deactivated during retrace support (reverse / forward travel).

The tool offsets are deactivated during retrace support via the standard default settings of the RESU-specific subprogram "CC_RESU_INI.SPF" (see Section "INI program (CC_RESU_INI.SPF) (Page 873)").

Specific instances of tool radius compensation, e.g. compensation on outside corners `G450 DISC=x` may generate contour deviations between the contour traversed during retrace support and the contour programmed in the machining program.

Contour deviations are always generated if tool radius compensation produces contour elements that are non-linear or circular. For example, `G450DISC=x`, where x > 0 produces parabolic or hyperbolic contour elements.

A full description of the tool radius compensation can be found in:
Reference:
Function Manual, Basic Functions; Tool Compensation (W1)

21.10 Data lists

21.10.1 Machine data

21.10.1.1 General machine data

Number	Identifier: $MN_	Meaning
11602	ASUP_START_MASK	Ignore stop reasons if an ASUB is running.
11604	ASUP_START_PRIO_LEVEL	Defines the ASUB priority from which MD11602 is effective.
18351	MM_DRAM_FILE_MEM_SIZE	Size of the part program memory in DRAM (in kByte)

21.10.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment 'geometry/channel axis'
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Assignment of geometry - channel axes for transformation 1
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles (CC)
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for CC in KB
28105	MM_NUM_CC_HEAP_MEM	Heap memory in KB for CC applications (DRAM)
62571	RESU_RING_BUFFER_SIZE	Size of ring buffer (RESU-internal block buffer)
62572	RESU_SHARE_OF_CC_HEAP_MEM	RESU portion of the total CC heap memory
62573	RESU_INFO_SA_VAR_INDEX	Indices of the synchronized action variables

Number	Identifier: $MC_	Description
62574	RESU_SPECIAL_FEATURE_MASK	Additional RESU features
62575	RESU_SPECIAL_FEATURE_MASK_2	Additional RESU properties
62580	RESU_WORKING_PLANE	Selection of the RESU working plane

21.10.2 Signals

21.10.2.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
RESU: Backward/forward	DB21, ...DBX0.1	-
RESU: Start retrace support	DB21, ...DBX0.2	-

21.10.2.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
RESU: Retrace mode active	DB21, ...DBX32.1	-
RESU: Retrace support active	DB21, ...DBX32.2	-

TE8: Cycle-independent path-synchronous switching signal output - 840D sl only 22

22.1 Brief description

Function

The "Cycle-independent path synchronized switching signal output" technological function serves the purpose of switching time-critical, position-based machining processes on and off quickly, e.g. high speed laser cutting (HSLC; High Speed Laser Cutting).

The switching signal output can be block-related or path length-related:

- Block-related switching signal output
  The switching signal output and therefore, the activation / deactivation of the machining is undertaken independently of the status changes:
  - Rapid travel mode `G00` active / inactive
  - Programmed feed threshold fell short / exceeded
- Path length-related switching signal output
  The switching signal output and therefore, the activation / deactivation of the machining takes place in a continuous change and is monitored along the traversed path.
  This enables a regular machining without the individual switching positions having to be programmed explicitly.

The activation or selection of the mentioned options leads to the monitoring of the output of the digital signal which is programmed with a part program command.

I/Os

Only the on-board I/Os of the NC module can be used as the digital I/O via which the switching signal is output: The switching signal can only be output via one of the 4 on-board digital outputs on the NCU module.

Restrictions

The use of the "Cycle-independent path synchronized switching signal output" technological function is subject to the following restriction:

- **The technological function is available only in one channel of NC.**

References

The "cycle-independent path-synchronous switching signal output" technological function is a compile cycle. For the handling of compile cycles, see Section "TE01: Installation and activation of loadable compile cycles (Page 707)".

22.2 Functional description

22.2.1 General information

Note

The functionality is described with examples, with the help of the "High speed laser cutting technology (HSLC, High Speed Laser Cutting).

22.2.2 Calculating the switching positions

22.2.2.1 Block-related switching signal output

Switching criteria

During high-speed laser cutting, e.g. as used to manufacture perforated sheets, it is absolutely essential to switch the laser beam on/off exactly at the programmed setpoint positions during the machining process.

In order to minimize programming overheads, the switching positions of the technology function are calculated using the velocity of the geometry axes programmed in the part program block.

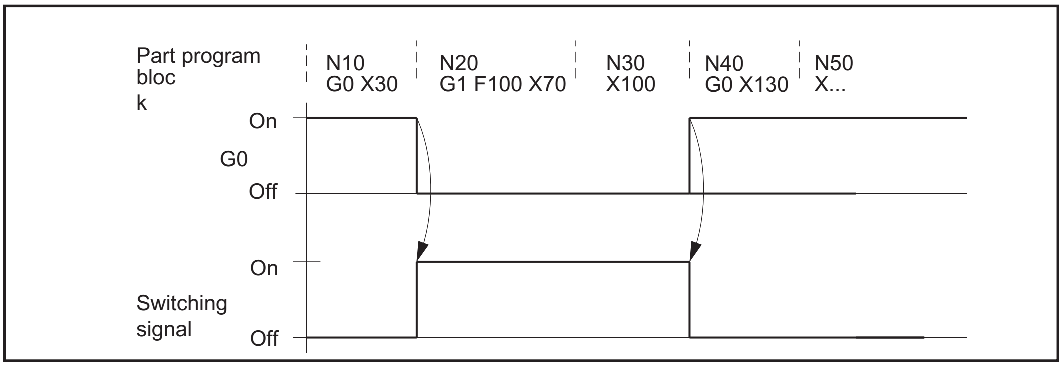
The following criteria define the setpoint position programmed in the part program block (end of block position) as a switching position:

1. `G0` edge change
2. Overshooting/undershooting a freely programmable velocity threshold

G0 edge change as switching criterion

If `G0` (rapid traverse) is **active** in a part program block (programmed or modal), the switching signal is switched **off**. On the other hand: If `G0` (rapid traverse) is **not active** in a part program block, the switching signal is switched **on**. The `G0` edge change marks the programmed end of block position of the previous block as the switching position.

Example:



The following block positions function as switching positions:

- Position X30 for `G0`-edge change from `N10` to `N20`
- Position X100 for `G0`-edge change from `N30` to `N40`

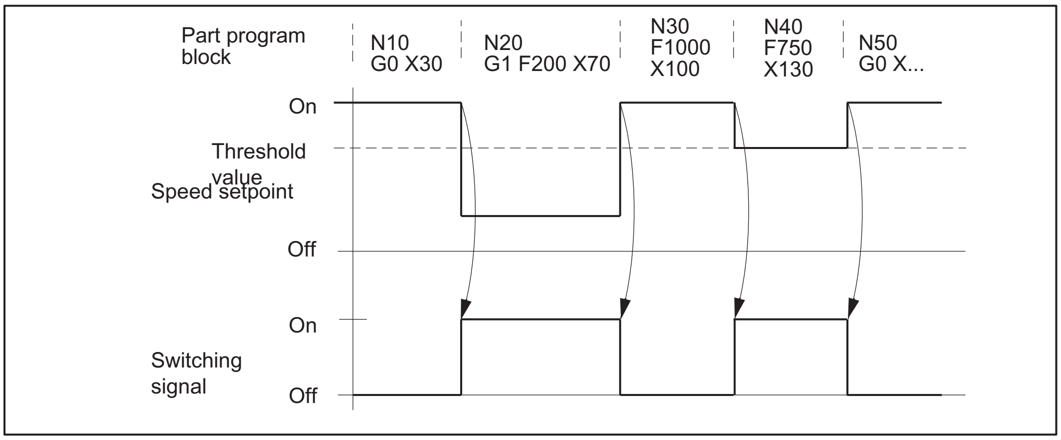
Freely programmable velocity threshold value as switching criterion

A freely programmable velocity threshold value is used to define the setpoint velocity programmed in the part program block at and above which the switching signal is activated/ deactivated.

- If the setpoint velocity programmed in the part program block is **above** the programmed threshold, the switching signal is switched **off**.
- If the setpoint velocity is **at** or **below** the threshold value, the switching signal is switched **on**.

The edge change marks the programmed end of block position of the previous block as the switching position.

Example:



The following block positions function as switching positions:

- Position X30 for edge change from `N10` to `N20`
- Position X70 for edge change from `N20` to `N30`

Note

`G0` always deactivates the switching signal, regardless of the threshold value.

22.2.2.2 Path length-related switching signal output

Programmable paths as the switching criterion

In path-related switching signal output, the switching positions are defined with the help of the two freely programmable paths s_1 and s_2.

Functional sequence

The path length-related switching signal output starts with a switch on signal at the beginning of the first traversing block after activation with `CC_FAST_CONT` (refer to ① in the screen).

The processing is active till a deactivation signal is set after the traversing of a programmable path s_1 (refer to ② in screen). This way the processing is interrupted till a switch on signal is set again after the traversing of a programmable path s_2 (refer to ③ in screen).

The changeover between activation and deactivation signal, and therefore between processing and interruption phase is path length dependent at the end of either of the stretch sections s_1 and s_2. This enables a continuous and regular machining without the individual switching positions having to be programmed explicitly.

The path length-related switching signal output ends with the start of the first traversing block (or of another executable block) after deactivation with `CC_FASTOFF` (refer to ④ in screen).

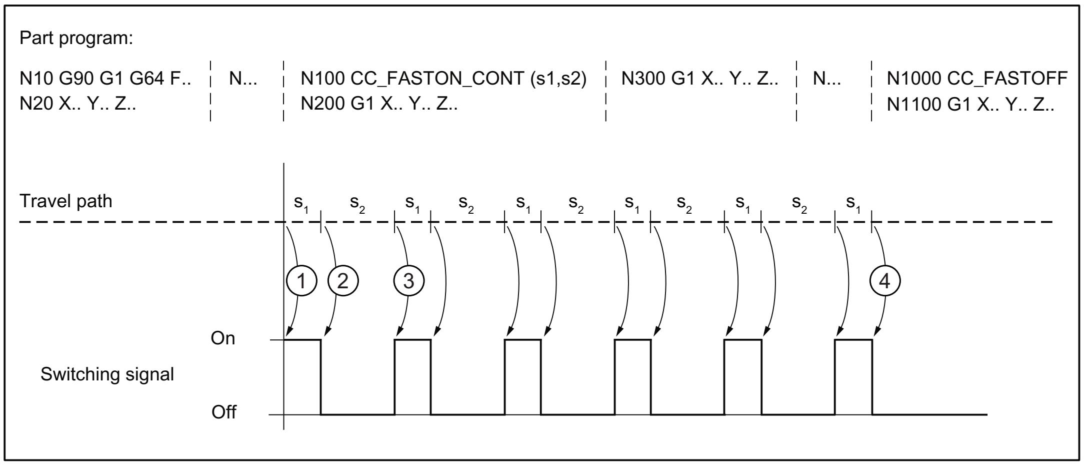


Figure 22-1 Path length-related switching signal output

22.2.3 Calculating the switching instants

In order for the switching to be as precise as possible at the switching positions calculated, the control calculates the positional difference between the actual position of the geometry axes involved and the switching difference in every position controller cycle.

If the positional difference is less than 1.5 position controller cycles, the control converts it into a temporal difference taking into account the current path velocity and acceleration rate of the geometry axes.

With the temporal difference specified, a hardware timer is started, which triggers the switching signal at exactly the instant calculated in advance regardless of the position controller cycle.

22.2.4 Switching frequency and switching position distance

Maximum switching frequency

The maximum switching frequency is: 1 signal edge change per IPO cycle

---

Note

Special case: IPO cycle time = position controller cycle time

In this case, the maximum switching frequency is:

1 signal edge change per **2** IPO cycles

Minimum switching position clearance

The minimum possible distance between the switching positions placed one above the other depends on:

- The duration of an IPO cycle
- The feed rate

The theoretically possible minimum distance can be determined from these dimensions as follows:

```
Minimum switching position clearance  =  Programmed feed rate * IPO cycle time
```

Example:

For IPO cycle times of 2 ms and position controller cycle times of 1 ms as well as a feed rate of 20000 mm/min, the theoretically possible minimum distance between switching positions one above the other is limited to:

20000 mm/min * 2 ms = 0.667 mm

Value below minimum switching position distance

For path length-related switching signal output, the value may fall below the minimum switching position distance, e.g. due to:

- Increase in feed rate
- Decrease in programmable switching position distance s_1 and s_2

The following reactions take place if the value falls below the minimum:

- Alarm 75501 "Channel %1 HSLC: CC_FASTON_CONT speed too high" is displayed.
- The switching signal at the current switching position is omitted. To maintain the machining rhythm, the function also suppresses the switching signal at the following switching position. The position of all the following switching positions is not affected by this. In the further course of the path, one can witness alternating output and suppression of two switching signals following each other.

22.2.5 Approaching switching position

If in block-related switching signal output a switching position is not reached exactly, e.g. in continuous-path mode and travel in more than one geometry axis, then switching takes place at the instant at which the positional difference between the actual position of the geometry axes involved and the programmed switching position increases again.

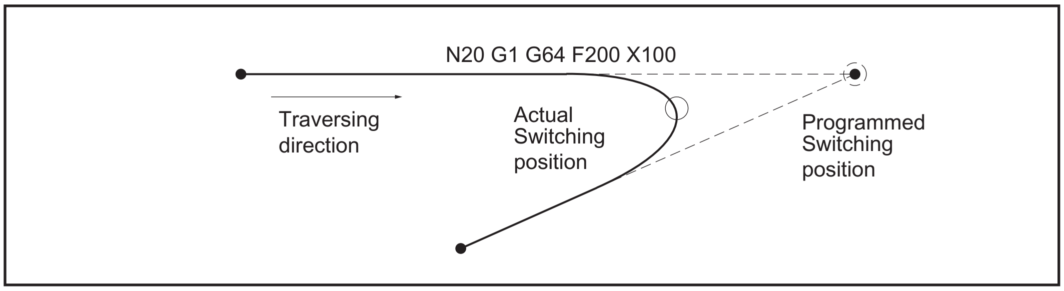


Figure 22-2 Switching position offset in path control mode

22.2.6 Programmed switching position offset

Programmed switching position offset

For block-related switching signal output, a positional offset of the switching position can be programmed :

- Offset distance **negative** = lead
  With a negative offset distance, the switching position is offset **before** the set point position programmed in the part program block.
  If an excessively large negative offset distance is programmed, i.e. the setpoint has already been exceeded by the time the edge is detected, the signal is switched immediately.
- Offset distance **positive** = follow-up
  With a positive offset distance, the switching position is offset **behind** the set point position programmed in the part program block.

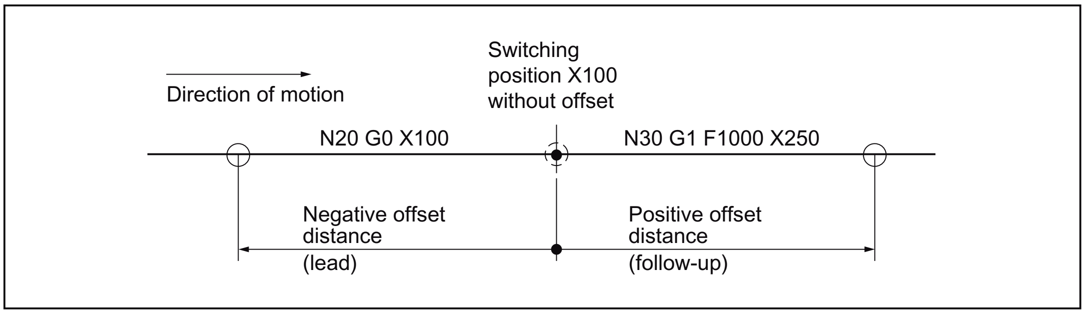


Figure 22-3 Programmed switching position offset

Path reference

The offset distance is a positional specification that refers to the programmed path. This way one can start from a simplified linear motion. Path curves are not taken into account.

Response to single block and G60

Due to the internal motion logic, negative offset distances (lead) have no effect when used with the following standard functions:

- Single Block
- Exact stop at block end (`G60`)

22.2.7 Response to part program interruption

Following an interruption in the part program (NC-STOP) and subsequent change to JOG mode, the technological function is deactivated or switching signals cease to be output.

The technology function is reactivated or switching signals are output again only after switching to the AUTOMATIC mode and continuing the part program (NC-START).

22.3 Startup

22.3.1 Activation

Before commissioning the technological function, ensure that the corresponding compile cycle has been loaded and activated (see also Section "TE01: Installation and activation of loadable compile cycles (Page 707)").

Activation

The "Cycle-independent path synchronized switching signal output" technology function is activated via the following machine data:

MD60948 $MN_CC_ACTIVE_IN_CHAN_HSLC[0], Bit 0 = 1

Note

The "Cycle-independent path-synchronized switching signal output" is available only in **one** channel of NC.

22.3.2 Memory configuration

The technological function requires **additional** data in the NC-internal block memory. The values must be increased for the following memory configuring channel-specific machine data:

- MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS += **1** (number of block elements for compile cycles)
- MD28100 $MN_MM_NUM_CC_BLOCK_USER_MEM += **10** (size of block memory for compile cycles (DRAM) in KB)

22.3.3 Parameterizing the digital on-board outputs

Parameter assignment

A digital output from the local I/O is required for the switching signal.

For this, at least 1 digital output byte must be defined through the following machine data:

MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS ≥ 1 (number of active digital output bytes)

References

A full description of the parameterization of a digital output can be found in:

- Function Manual, Extended Functions, Digital and Analog NC I/O (A4)

22.3.4 Parameterizing the switching signal

Output number of the switching signal

Once the compile cycle has started up, the following function-specific machine data appears in the channel-specific machine data:

MD62560 $MC_FASTON_NUM_DIG_OUTPUT (number of the digital output of the switching signal)

The number n of the on-board digital output through which the switching signal is to be output, must be entered in it:

n = 1, 2, 3 or 4

Disable

Entering the number of the digital output n = 0 deactivates the function. **No** message or alarm is output.

Effect on other output signals

The hardware-timer-controlled output of the switching signal at the parameterized output delays the signal output for the other digital on-board outputs, e.g. due to synchronized actions, by 2 IPO cycles.

22.3.5 Parameterization of the geometry axes

Default setting

Machines for high-speed laser cutting normally have two geometry axes that are configured in the following two machine data:

MD20050_$MC_AXCONF_GEOAX_ASSIGN_TAB[0]

MD20050_$MC_AXCONF_GEOAX_ASSIGN_TAB[1]

The calculation of the switching instants is derived from these two geometry axes.

Note

The configured axis selection for calculating the switching instants can be changed by redefining the first and second geometry axes in the part program with the help of program instructions `GEOAX(1,`*<axis name>*`)` and `GEOAX(2,`*<axis name>*`)`.

Important:
For the function change to be considered correctly, it should occur before the interpretation of the `CC_FASTON` command.

Changing the default setting

For a deviating machine configuration (e.g. definition of a third geometry axis), the default setting can be adjusted via the following machine data:

MD60948 $MN_CC_ACTIVE_IN_CHAN_HSCL[1]

Value	Meaning
$MN_CC_ACTIVE_IN_CHAN_HSCL[1]='H3'	Default setting. The calculation of the switching instants is derived from the **first and second** geometry axis.
$MN_CC_ACTIVE_IN_CHAN_HSCL[1]='H7'	The calculation of the switching instants is derived from the **first, second and third** geometry axis.

Any change is effective only after the NC has powered up the next time.

22.4 Programming

22.4.1 Activating the block-related switching signal output (CC_FASTON)

Syntax

```
CC_FASTON (DIFFON, DIFFOFF [,FEEDTOSWITCH])
```

`CC_FASTON()` is a procedure call and must therefore be programmed in a dedicated part program block.

Parameter

The parameters for the `CC_FASTON()` procedure have the following meaning:

Parameter	Meaning
<DIFFON>	Length* of the offset distance for setting the switching signal.
<DIFFOFF>	Length* of the offset distance for deactivating the switching signal.
<FEEDTOSWITCH>	This parameter is optional. If the parameter is not specified in the procedure call, the `G0` edge change is used as the switching criterion. If the parameter is specified in the procedure call, it contains as a switching criterion the velocity threshold value, which, when undershot or exceeded, activates or deactivates the switching signal accordingly.
* The basic unit (inch or mm) depends on the current dimensions programming (`G70` / `G71` / `G700` / `G710`).	

Programming example

Programming	Comment
`DEF REAL DIFFON= -0.08`	`; Length* of the offset distance for activating the switching signal = -0.08`
`DEF REAL DIFFOFF= 0.08`	`; Length* of the offset distance for deactivating the switching signal = 0.08`
`DEF REAL FEEDTOSWITCH= 20000`	`; Speed threshold value = 20000`
`CC_FASTON(DIFFON,DIFFOFF,FEEDTOSWITCH)`	`; Activating block-related switching signal output (with speed threshold value as switching criterion)`

Changing parameters

The parameters for the `CC_FASTON()` procedure can be modified at any time during the execution of the part program. To do this, enter the procedure call again with the new parameter values. The switching criterion (`G0` edge change / velocity threshold value) may also be changed.

Reset response

A reset (NC RESET or end of program) deactivates the function.

22.4.2 Activating the path length-related switching signal output (CC_FASTON_CONT)

Syntax

```
CC_FASTON_CONT (PATH_DISTANCE_ON, PATH_DISTANCE_OFF)
```

`CC_FASTON_CONT()` is a procedure call and must therefore be programmed in a dedicated part program block.

Parameter

The parameters for the `CC_FASTON_CONT()` procedure have the following meaning:

Parameter	Meaning
< PATH_DISTANCE_ON >	Length* of the stretch section with machining (s_1).
< PATH_DISTANCE_OFF>	Length* of the stretch section without machining (s_2).
* The basic unit (inch or mm) depends on the current dimensions programming (`G70` / `G71` / `G700` / `G710`).	

Programming example

Programming	Comment
`DEF REAL PATH_DISTANCE_ON = 0.5`	`; Length* of the stretch section with machining = 0.5`
`DEF REAL PATH_DISTANCE_OFF = 1.0`	`; Length* of the stretch section without machining = 1.0`
`CC_FASTON_CONT (PATH_DISTANCE_ON, PATH_DISTANCE_OFF)`	`; Activating path length-related switching signal output.`

Changing parameters

The parameters for the `CC_FASTON_CONT( )` procedure can be modified at any time during the execution of the part program. For this, the procedure call must be specified again with the new parameter values.

Reset response

A reset (NC RESET or end of program) deactivates the function.

22.4.3 Deactivation (CC_FASTOFF)

Syntax

`CC_FASTOFF`

`CC_FASTOFF` is a procedure call and must therefore be programmed in a dedicated part program block.

Functionality

The `CC_FASTOFF` procedure call deactivates the "Cycle independent, path synchronized switching signal output" function.

22.5 Function-specific alarm texts

For details of the procedure for creating function-specific alarm texts, see Section "Creating alarm texts (Page 713)".

22.6 Supplementary conditions

22.6.1 Block search

Switching signal output for block search

If a block search is on a part program block which lies after a `CC_FASTON()` procedure call for activating the technology function, then the switching signal is activated with the next traversing motion. One of the specific consequences of this is to initiate travel along the contour from the start position of the geometry axes back to the program continuation point with an activated switching signal.

Example

Normal process:

In the normal process of the part program machining, the switching signal is activated for the first time at the beginning of part program block `N60`.

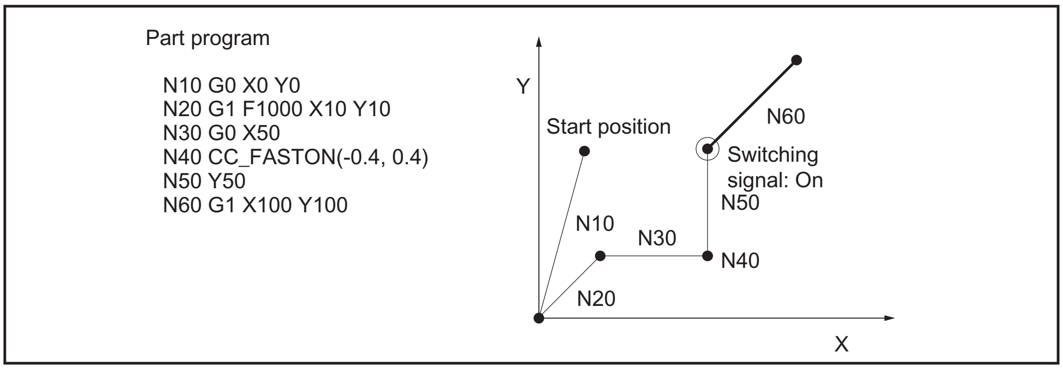


Figure 22-4 Switching signal for part program machining operation

Process sequence after block search:

If a block search is executed for the block end point of part program block `N60` the switching signal is activated on reaching the start position of the geometry axes.

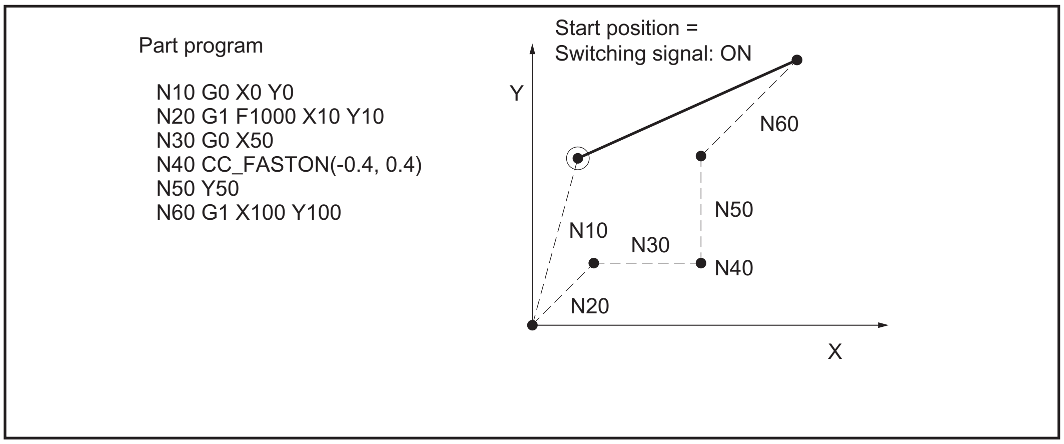


Figure 22-5 Switching signal after block search

Suppressing the switching signal output

The user (machine manufacturer) must take appropriate measures, e.g. disable the switching signal, in order to suppress the activation of the switching signal in the REPOS block in the constellation described above.

Note

It is the sole responsibility of the user (machine manufacturer) to suppress the output of the switching signal during repositioning, e.g. after a block search.

References

You will find a description of the block search in:

Function Manual, Basic Functions, Mode Group, Channel, Program Operation, Reset Behavior (K1)

22.6.2 Transformations

The function will only run correctly with deactivated transformation. There is no monitoring function.

For a description of the transformations (see Section "TE4: Handling transformation package - 840D sl only (Page 793)").

References:
Function Manual Extended Functions; Kinematic Transformation (M1)

22.6.3 Compensations

The following compensations are considered while calculating the switching positions:

- Temperature compensation
- Sag compensation

A description of the compensations can be found in:
Reference:
Function Manual, Extended Functions; Compensations (K3)

22.6.4 Tool radius compensation (TRC)

As part of tool radius compensation, control-internal part program blocks (compensation blocks) are inserted into the part program. With reference to the switching signal output, a compensation block is always added to the next programmed part program block.

A description of the tool radius compensation can be found in:
Reference:
Function Manual, Basic Functions; Tool Compensations (W1), Section: Tool radius compensation

22.6.5 Continuous-path mode

Continuous-path mode

Although the `CC_FASTON()`, `CC_FASTON_CONT()` and `CC_FASTOFF` procedure calls must be programmed in dedicated part program blocks, this will not lead to a drop in velocity while continuous-path mode is active (`G64`, `G641`, ...).

Continuous-path mode (ADIS)

If in continuous-path mode with programmable rounding response (`G641 ADIS`) a part program block is inserted in the part program, then the originally programmed switching position is not reached and the switching signal output takes place nevertheless in the center of the rounding block.

References

You will find a description of the continuous-path mode in:

Function Manual, Basic Functions, Continuouspath Mode, Exact Stop and Look Ahead (B1)

22.6.6 Software cams

Because the hardware timer is also used for the "software cam" function, it is not possible to use the "clock-independent switching signal output" function with software cams at the same time.

The following alarm appears in the event of an error:

Alarm 75500 "Channel *Channel No.*, wrong configuration of function: clock-independent switching signal output"

A description of the software cams can be found in:
Reference:
Function Manual, Extended Functions; Software cams, Position switching signals (N3)

22.7 Data lists

22.7.1 Machine data

22.7.1.1 General machine data

Number	Identifier: $MN_	Description
10360	FASTO_NUM_DIG_OUTPUTS	Number of digital output bytes

22.7.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20050	AXCONF_GEOAX_ASSIGN_TAB	Assignment of geometry axis to channel axis
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for CC
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for CC
62560	FASTON_NUM_DIG_OUTPUT	Number of the on-board digital output for the switching signal

TE9: Axis pair collision protection 23

23.1 Brief description

Note

Compile cycle

Before commissioning the function, ensure that the corresponding compile cycle has been loaded and activated (see Chapter "TE01: Installation and activation of loadable compile cycles (Page 707)").

Function

The "axis pair collision protection" function enables machine axes, which are arranged on the same guide element of a machine, to be monitored in pairs to ensure that no collisions occur and that the maximum distance between the two axes is not exceeded.

Function code

The code for function-specific identifiers of machine data, system variables etc., is:

PROTECT (axial collision PROTECTion)

Maximum number of axis pairs

A maximum of **20** axis pairs can be parameterized.

23.2 Functional description

The "axis pair collision protection" function is a protection function for machine axes which are arranged in a machine tool in such a way (on the same guide rail, for example) that incorrect operation or programming could cause them to collide with one another.

The machine axes are always monitored in pairs, i.e. parameters always need to be assigned for two machine axes in each case, which are then monitored in relation to one another. The machine axes being monitored may be located in different machine coordinate systems.

Collision protection

The function uses the current actual positions and actual velocities, as well as the offset for the machine coordinate systems and the axis-specific brake acceleration values, to calculate the distance between the standstill positions of the machine axes, on a cyclical basis. If the resulting distance is shorter than the protection window that has been set, the machine axes are decelerated to a standstill. This ensures that the minimum distance defined using the protection window is not undershot.

Distance monitoring

If the offset vector is selected accordingly, the function can also be used to monitor the maximum distance between the two machine axes (maximum distance vector).

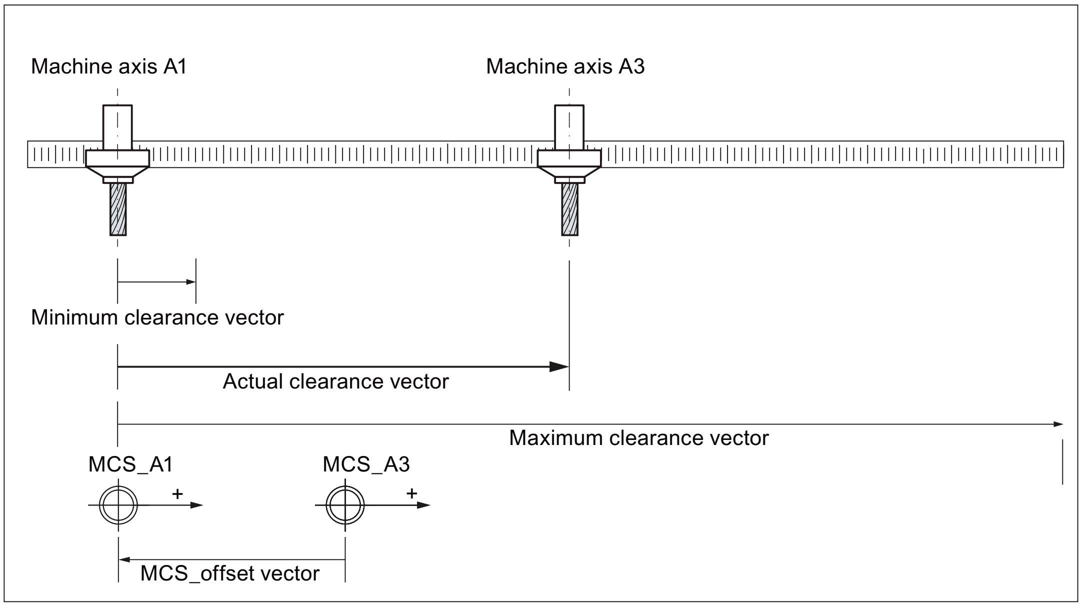


Figure 23-1 Basic design

Monitoring status

The actual status of an axis pair can be read out of _PROTECT_STATUS (Page 912), optionally defined global user variables in the NC program (GUD).

23.3 Startup

23.3.1 Enabling the technology function (option)

The function is an option, which must be assigned to the hardware via the license management.

6FC5800-0AN06-0YB0, "RMCC/PROT axis collision avoidance"

For test purposes, the function can be enabled by setting the option data:

MD19610 $ON_TECHNO_EXTENSION_MASK[2], BIT4 = 1

23.3.2 Activating the technology function

Activation rule

The function must be activated on a channel-for-channel basis for the following NC channels:

- Independent of which channels are assigned to the machine axes to be monitored, always in the 1st channel of the NC
- In all channels that are assigned to the machine axes to be monitored (by the function) by parameterizing the machine data
- In all channels that are assigned at a later point in time to the machine axes to be monitored (by the function), e.g. as a result of axis replacement

Activation

The function is activated on a channel-for-channel basis using machine data:

MD60972 $MN_CC_ACTIVE_IN_CHAN_PROT[0], bit n = 1

with n = 0, 1, 2, ..., corresponding to the (n+1)th channel of the NC

See also

TE01: Installation and activation of loadable compile cycles (Page 707)

23.3.3 Activating the supplementary functions

The supplementary functions are activated for specific axis pairs using option data:

MD61535 $MN_CC_PROTECT_OPTIONS[<a>]

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

Bit	Value	Meaning
0	1	Axis pair-specific activation/deactivation of the function "Axis pair collision protection" via the NC/PLC interface signal (Page 913)" **Note** After activating this supplementary function, the protection function is in the monitoring status (Page 912) == 1 (selected, but still not active)

23.3.4 Definition of an axis pair

A machine axis pair to be monitored is defined on an axis pair-specific basis in machine data:

MD61516 $MN_CC_PROTECT_PAIRS[<a>] = <yyxx>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

<yyxx>	Meaning
xx	1st and 2nd decimal place ⇒ axis number of the 1st machine axis
yy	3rd and 4th decimal place ⇒ axis number of the 2nd machine axis

Example

Definition of the 1st axis pair:

- 1st axis: 4th machine axis
- 2nd axis: 12th machine axis

MD61516 $MN_CC_PROTECT_PAIRS[0] = 1204

23.3.5 Retraction direction

The direction of travel for retracting the corresponding machine axis is set using machine data:

MD61517 $MN_CC_PROTECT_SAFE_DIR[<a>] = <yyxx>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

<yyxx>	Meaning
xx	Retraction direction for the 1st axis of the axis pair
yy	Retraction direction for the 2nd axis of the axis pair
Retraction in the positive direction of travel of the machine axis: xx or yy > 0 Retraction in the negative direction of travel of the machine axis: xx or yy = 0	

Note

Changing the retraction direction

Changing the retraction direction in machine data MD61517 $MN_CC_PROTECT_SAFE_DIR[<**axis pair**>] may only be carried out if the protection function for the axis pair is not active (MD61516 $MN_CC_PROTECT_PAIRS[<**axis pair**>] == 0).

23.3.6 Offset for the machine coordinate systems

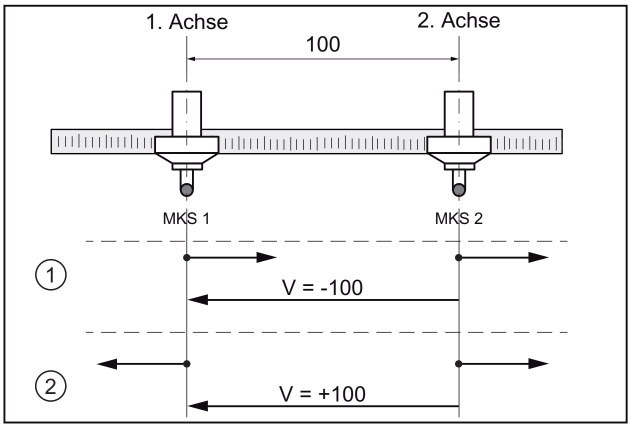
If the machine axes of the axis pair are located in different machine coordinate systems, then the appropriate offset vector must be specified in the following machine data:

MD61518 $MN_CC_PROTECT_OFFSET[<a>] = <offset vector>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

The offset vector should be specified as vector from the origin of the machine coordinate system of the 2nd axis of the axis pair to the origin of the machine coordinate system of the 1st axis, referred to the machine coordinate system of the 1st axis.

If both machine axes are located in the same machine coordinate system, an offset vector of 0 must be specified.



① Case 1: Equidirectional MCS orientation
② Case 2: Counterdirectional MCS orientation
V Offset vector

Note

Changing the offset vector

Changing the offset vector in machine data MD61518 $MN_CC_PROTECT_OFFSET[<**axis pair**>] may only be carried out if the protection function for the axis pair is not active (MD61516 $MN_CC_PROTECT_PAIRS[<**axis pair**>] == 0).

23.3.7 Protection window

The protection window and/or the minimum clearance is defined which the axes of the axis pair may not fall below using the machine data:

MD61519 $MN_CC_PROTECT_WINDOW[<a>] = <minimum clearance>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

When the clearance approaches the minimum clearance, the axes are braked with the function-specific acceleration (Page 911).

The protection window can be dynamically extended, e.g. in an NC P program using theprotection window extension (Page 910).

Note

Changing the protection window

Changing the protection window in machine data MD61519 $MN_CC_PROTECT_WINDOW[<**axis pair**>] may also be carried out if the protection function for the axis pair is **active** (MD61516 $MN_CC_PROTECT_PAIRS[<**axis pair**>] ≠ 0).

23.3.8 Orientation

The orientation of the axes of the axis pair to one another is specified in the following machine data:

MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[<a>] = <value>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

<value>	Meaning
0	Equidirectional orientation
1	Counterdirectional orientation

Note

Changing the orientation

Changing the orientation in machine data MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[<**axis pair**>] may only be carried out if the protection function for the axis pair is not active (MD61516 $MN_CC_PROTECT_PAIRS[<**axis pair**>] == 0).

23.3.9 Protection window extension

The protection window (Page 909) can be dynamically **extended** using machine data:

MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[<a>] = <extension>

with a = 0, 1, 2, ... (maximum number of axis pairs - 1) corresponding axis pair 1, 2, 3, ...

The resulting effective protection window of an axis pair is as follows:

Effective protection window[<**axis pair**>] =

MD61519 $MN_CC_PROTECT_WINDOW[<**axis pair**>] +

MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[<**axis pair**>]

It is **not** possible to reduce the protection window by entering a negative value.

Note

Changing the protection window extension

Changing the protection window extension in machine data MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[<axis pair>] may also be carried out when the protection function is active, e.g. from the NC program - and can be activated by initiating "Activate machine data".

23.3.10 Activating the protection function

Static activation using machine data

The protection function for an axis pair is statically activated as soon as the following preconditions are fulfilled:

- Both machine axes of the axis pair have been referenced
- Valid machine axes have been parameterized for the axis pair:
  MD61516 $MN_CC_PROTECT_PAIRS[<axis pair>] = <valid machine axis pair>
- The supplementary function "Axis pair-specific activation/deactivation via axis-specific NC/PLC interface signals" is not active:
  MD61535 $MN_CC_PROTECT_OPTIONS[<axis pair>], bit 0 = 0

Dynamic activation via NC/PLC interface signal

The protection function for an axis pair is dynamically activated as soon as the following preconditions are fulfilled:

- Both machine axes of the axis pair have been referenced
- Valid machine axes have been parameterized for the axis pair:
  MD61516 $MN_CC_PROTECT_PAIRS[<axis pair>] = <valid machine axis pair>
- The supplementary function "Axis pair-specific activation/deactivation via axis-specific NC/PLC interface signals" is active:
  MD61535 $MN_CC_PROTECT_OPTIONS[<axis pair>], bit 0 = 1
- The axis-specific NC/PLC interface signal to activate the protection function is set for one of the two machine axes of the axis pair:
  DB31, ...DBX24.3 == 1

Clearance less than the protection window

If, at the time when the protection function is activated, the distance between the two machine axes is less than the minimum distance of the protection window which has been parameterized, then the machine operator must retract the machine axes. In this particular state, the control only allows traversing motion in the parameterized retraction direction (Page 908) of the machine axes.

Monitoring status

The actual monitoring status of an axis pair can be read using global user variable _PROTECT_STATUS (Page 912).

23.3.11 Axis-specific acceleration

The acceleration with which the two machine axes of the axis pair are braked by the protection function when a critical distance is approached is set using:

MD63514 $MA_CC_PROTECT_ACCEL[<axis>] = <acceleration>

with <axis>: Machine axis name, e.g. AX1, AX2, ...

Note

Without jerk limitation

The braking acceleration set using MD63514 $MA_CC_PROTECT_ACCEL acts without jerk limiting.

Priority of the function-specific acceleration

The protection function only uses the function-specific acceleration of the machine axes from MD63514 $MA_CC_PROTECT_ACCEL to calculate the time at which the axis should be braked. The protection function does not take the actual acceleration of the machine axis in the channel into account.

Note

Path reference

If machine axes being monitored by the protection function are traversed by a channel with a path reference to other axes, this path reference is lost as soon as the protection function causes the axis grouping to decelerate. The machine axes being monitored by the protection function are decelerated using their function-specific acceleration values from machine data item MD63514 $MA_CC_PROTECT_ACCEL. The remaining axes in the axis grouping are decelerated using the current path acceleration value of the channel.

23.3.12 Monitoring status (GUD)

The actual status of one axis pair is displayed using the global user variable _PROTECT_STATUS.

The variable is not available in the default setting. When required, it must be defined in the GUD.DEF definition window.

Definition

```
DEF NCK INT _PROTECT_STATUS[ <number of parameterized axis pairs> ]
```

with <number of parameterized axis pairs> = 1, 2, 3, ... (maximum number of axis pairs)

Range of values

Value	Meaning: The monitoring of the axis pair is ...
0	Not active
1	selected, but not yet active
2	active, the axes are currently **not** being braked
3	active, the axes are currently being braked
4	deselected, however still active

23.3.13 PLC interface: Axis-specific braking operations

Within the general system variable field $A_DBD, a double word (four bytes) can be used to define an axes-specific braking interface. When the axes of the axis pair critically approach one another, the actual braking of the machine axes is displayed via the braking interface.

MD61534 $MN_CC_PROTECT_A_DBD_INDEX = <value>

<value>	Meaning
-1	Deactivation of the output.
≥ 0	Braking interface index within the system variable field: $A_DBD[<index>] with index = 0, 4, 8, 12, ...

Note

Double word index

The starting index can be specified in the machine data, byte-by-byte (0, 1, 2, ...). System variable $A_DBD is accessed from the PLC using a double word. This means that a starting index, which is not located at a double word boundary (0, 4, 8, ...), is **rounded off** to the next double word boundary (0, 4, 8, 12, ...): Index = (index DIV 4) * 4

Braking interface

Each bit of the braking interface is assigned a machine axis:

Bit n → (n+1)th machine axis, with n = 0, 1, 2, ...

Bit	31	30	29	28	...	3	2	1	0
MA 1)	---	31	30	29	...	4	3	2	1
1) Machine axis number									

Bit n	Meaning
0	Machine axis (n+1) is not braked
1	Machine axis (n+1) is braked

References

Block FC21, function 3 is available to read the braking interface from the PLC user program.

Function Manual Basic Functions; Chapter "Basic PLC program for SINUMERIK 840D sl" > "Block descriptions" > "FC21: Transfer data exchange NC/PLC" > "Function 3, 4: Fast data exchange PLC-NC"

23.3.14 PLC interface: Axis pair-specific activation of the protection function

If the supplementary function "Activation/deactivation of function "Axis pair collision protection" via NC/PLC interface signal (Page 907)" is active, using the axis-specific interface signal, the protection function for the axis pair can be activated and deactivated from the PLC user program:

DB31, ...DBX24.3 (activate collision protection)

Activating

The protection function is activated if the axis-specific interface signal for one of the two machine axes of the axis pair is set.

Deactivating

The protection function is deactivated if the axis-specific interface signal for both machine axes of the axis pair is reset.

23.4 Limitations and constraints

23.4.1 Axes

Same axis types

Both machine axes of an axis pair must be of the **same** axis type:

- Linear axis:
  - MD30300 $MA_IS_ROT_AX = 0
  - MD30310 $MA_ROT_IS_MODULO = 0
- Rotary axis:
  - MD30300 $MA_IS_ROT_AX = 1
  - MD30310 $MA_ROT_IS_MODULO = 0

Modulo rotary axes

None of the machine axes of an axis pair may be a **modulo rotary axis**:

- MD30300 $MA_IS_ROT_AX = 1 (rotary axis)
- MD30310 $MA_ROT_IS_MODULO = 1 (error: modulo rotary axis !)

23.4.2 Axis container

If the assignment of the machine axes to be monitored changes dynamically during the course of a machining process – for example when using axis containers – the function must be deactivated, its parameters reassigned, and then reactivated prior to the change (e.g. axis container rotation) being made.

Example

The protection function is to monitor logical machine axes 1 and 13. These relate to slots 1 and 2 of axis container CT1. The associated real machine axes are AX1 and AX13.

In an axis container rotation, the axis container makes a transition by one step, which causes the real machine axes to be changed.

Machine axis AX13 is retracted in the positive direction of travel. Machine axis AX1 is retracted in the negative direction of travel.

Assigning parameters for the NC
Logical machine axes: axis numbers 1 and 13

- MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB [**0**] = "CT1_SL1" (log. mach. axis **1**)
- MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB [**12**] = "CT1_SL2" (log. mach. axis **13**)

Axis container CT1, slot 1 and slot 2

- MD12750 $MN_AXCT_NAME_TAB[0] = "CT1"
- MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[**0**] = "AX1" (slot **1**)
- MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[**1**] = "AX13" (slot **2**)

Real machine axes: machine axis names AX1 and AX13

- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[x] = "AX1"
- MD10000 $MN_AXCONF_MACHAX_NAME_TAB[y] = "AX13"

Parameterizing the "axis pair collision protection" function before **axis container rotation**

- MD61516 $MN_CC_PROTECT_PAIRS[0] = 01 13
- MD61517 $MN_CC_PROTECT_SAVE_DIR[0] = 01 00

Performing the axis container rotation

1. Deactivate the protection function
   MD61516 $MN_CC_PROTECT_PAIRS[0] = 00 00
2. Trigger a reset in the 1st NC channel in order to adopt the change made to the machine data
   MD60972 $MN_CC_ACTIVE_IN_CHAN_PROT[0], BITx, ...
3. Execute the axis container rotation
   AXCTSWED(CT1)

Re-parameterizing the "axis pair collision protection" function after **axis container rotation**:

- MD61516 $MN_CC_PROTECT_PAIRS[0] = 13 01
- MD61517 $MN_CC_PROTECT_SAFE_DIR[0] = 01 00

or

- MD61516 $MN_CC_PROTECT_PAIRS[0] = 01 13
- MD61517 $MN_CC_PROTECT_SAFE_DIR[0] = 00 01

Trigger a "reset" in the 1st NC channel in order to adopt the change made to the machine data.

23.4.3 Link axes

If the axes of an axis pair are link axes, i.e. the setpoints of machine axes from channels of various NCUs are generated via the "NCU link" function, then the axes can neither be monitored with respect to one another nor protected.

23.4.4 Interpolatory couplings

Assumption

1. A machine axis is part of an interpolatory coupling, e.g.:
   - Generic coupling (CP)
   - Coupled motion(TRAIL)
   - Master value coupling (LEAD)
   - Electronic gear (EG)
   - Synchronous spindle(COUP)
2. The machine axis is **not** traversed in the **first** channel of the NC.
3. The machine axis is monitored using the "axis pair protection" function.

Effect

If the machine axis is not traversed in the first channel of the NC, then the components, created from the interpolatory couplings for the position and velocity setpoint are only available after a **deadtime** of **one interpolator clock cycle** of the "axis pair collision protection" function. Therefore, the machine axes are monitored, offset by these components. The absolute value of the components is independent of the interpolator clock cycle and the actual velocity and acceleration of the machine axis.

23.5 Examples

23.5.1 Collision protection

The figure shows the arrangement of 3 machine axes and the offset and orientation of the machine coordinate systems (machine).

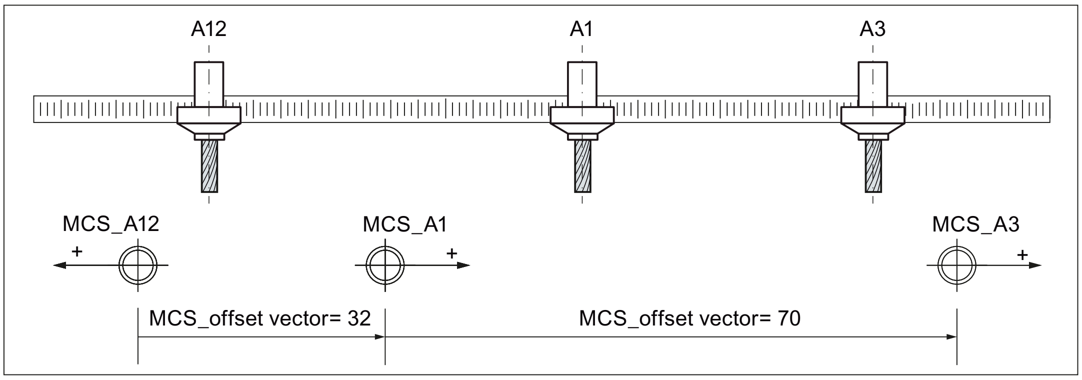


Figure 23-2 Collision protection for 2 axis pairs

Parameter assignment: Protection function 1

Axis pair: 1st machine axis A3, 2nd machine axis A1

- MD61516 $MN_CC_PROTECT_PAIRS[0] = 01 03

Retraction direction: A1 in negative direction, A3 in positive direction

- MD61517 $MN_CC_PROTECT_SAFE_DIR[0] = 00 01

Offset vector from machine coordinate system machine_A1 to machine_A3 with reference to machine_A3

- MD61518 $MN_CC_PROTECT_OFFSET[0] = 70.0

Example protection window, 10.0 mm

- MD61519 $MN_CC_PROTECT_WINDOW[0] = 10.0

Orientation of the machine coordinate systems to one another: same direction

- MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[0] = 0

Protection window extension: none

- MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[0] = 0.0

Parameter assignment: Protection function 2

Axis pair: 1st machine axis A1, 2nd machine axis A12

- MD61516 $MN_CC_PROTECT_PAIRS[1] = 12 01

Retraction direction: A12 in positive direction, A1 in positive direction

- MD61517 $MN_CC_PROTECT_SAFE_DIR[1] = 01 01

Offset vector from machine coordinate system machine_A12 to machine_A1 with reference to machine_A1

- MD61518 $MN_CC_PROTECT_OFFSET[1] = 32.0

Example protection window, 5.0 mm

- MD61519 $MN_CC_PROTECT_WINDOW[1] = 5.0

Orientation of the machine coordinate systems to one another: opposite direction

- MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[1] = 1

Protection window extension: by 5.0 mm to give a total of 10.0 mm

- MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[1] = 5.0

23.5.2 Collision protection and distance limiter

The figure shows the arrangement of two machine axes, the offset and orientation of the machine coordinate systems (machine), and the minimum and maximum distance vectors.

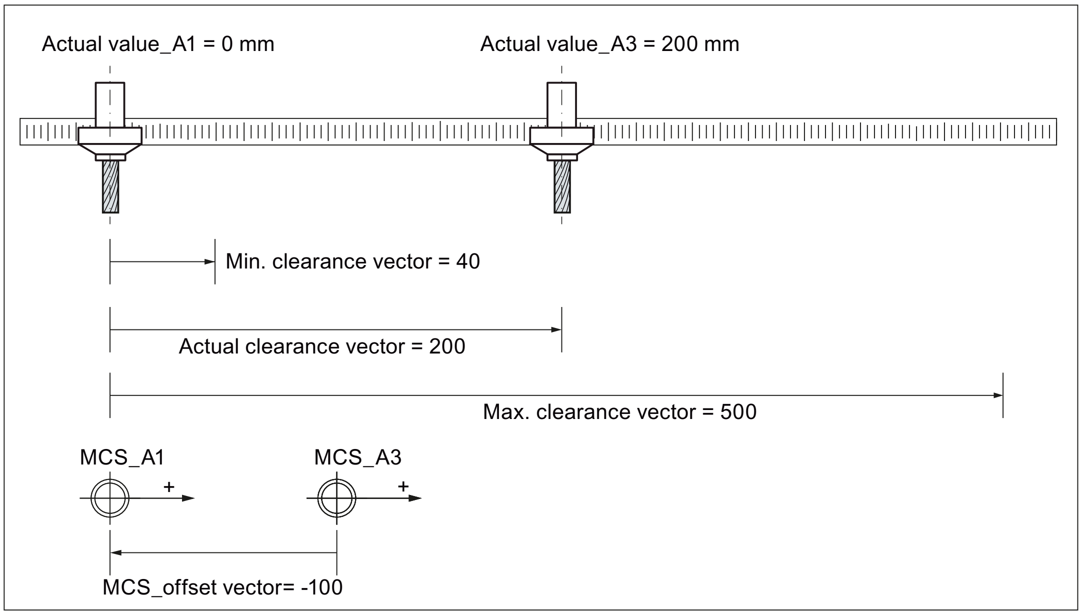


Figure 23-3 Collision protection and distance limiter for one axis pair

Parameter assignment: Protection function 1 - Collision protection

Axis pair: 1st machine axis A1, 2nd machine axis A3

- MD61516 $MN_CC_PROTECT_PAIRS[0] = 03 01

Retraction direction: A1 in negative direction, A3 in positive direction

- MD61517 $MN_CC_PROTECT_SAFE_DIR[0] = 01 00

Offset vector from machine coordinate system machine_A3 to machine_A1 with reference to machine_A1

- MD61518 $MN_CC_PROTECT_OFFSET[0] = -100.0

Example protection window, 40.0 mm

- MD61519 $MN_CC_PROTECT_WINDOW[0] = 40.0

Orientation of the machine coordinate systems to one another: same direction

- MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[0] = 0

Protection window extension: none

- MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[0] = 0.0

Parameter assignment: Protection function 2 - Distance limiter

Axis pair: 1st machine axis A1, 2nd machine axis A3

- MD61516 $MN_CC_PROTECT_PAIRS[1] = 03 01

Retraction direction: A1 in positive direction, A3 in negative direction

- MD61517 $MN_CC_PROTECT_SAFE_DIR[1] = 00 01

Offset vector = "offset vector from machine coordinate system machine_A3 to machine_A1 with reference to machine_A1" - "maximum distance vector with reference to machine_A1"

Note

Maximum distance vector

The maximum distance vector from the 1st machine axis to the 2nd machine axis is the vector from the origin of the machine coordinate system for the 1st machine axis to the maximum permissible position of the 2nd machine axis, with reference to the machine coordinate system for the 1st machine axis.

- MD61518 $MN_CC_PROTECT_OFFSET[1] = -100.0 - 500.0 = 400.0

Example protection window, 20.0 mm

- MD61519 $MN_CC_PROTECT_WINDOW[1] = 20.0

Orientation of the machine coordinate systems to one another: same direction

- MD61532 $MN_CC_PROTECT_DIR_IS_REVERSE[1] = 0

Protection window extension: none

- MD61533 $MN_CC_PROTECT_WINDOW_EXTENSION[1] = 0.0

If machine axis A1 has a value of 0, the settings above will limit the traversing range of machine axis A3 to between -60.0 and 380.0, with reference to machine_A3.

23.6 Data lists

23.6.1 Option data

Number	Identifier: $ON_	Description
19610	TECHNO_EXTENSION_MASK[6]	Enable the technology function via BIT4 = 1

23.6.2 Machine data

23.6.2.1 NC-specific machine data

Number	Identifier: $MN_	Description
60972	CC_ACTIVE_IN_CHAN_PROT[0]	Channel-specific activation of the technology function
61516	CC_PROTECT_PAIRS[<a>]	Axis pair-specific definition of both machine axes
61517	CC_PROTECT_SAFE_DIR[<a>]	Axis pair-specific definition of the retraction direction
61518	CC_PROTECT_OFFSET[<a>]	Axis pair-specific definition of the offset of both machine coordinate systems
61519	CC_PROTECT_WINDOW[<a>]	Axis pair-specific definition of the protection window or minimum clearance
61532	CC_PROTECT_DIR_IS_REVERSE[<a>]	Axis pair-specific definition of the orientation of the machine coordinate systems of both machine axes to one another
61533	CC_PROTECT_WINDOW_EXTENSION[<a>]	Axis pair-specific definition of the protection window extension
61534	CC_PROTECT_A_DBD_INDEX	Braking interface window within the system variable field $A_DBD
61535	CC_PROTECT_OPTIONS[<a>]	Axis pair-specific activation of supplementary functions

23.6.2.2 Axis/Spindle-specific machine data

Number	Identifier: $MA_	Description
61514	CC_PROTECT_ACCEL	Protection function-specific brake acceleration

23.6.3 Signals

23.6.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate collision protection	DB31, ...DBX24.3	-

23.6.4 User data

23.6.4.1 Global user data (GUD)

Identifier	Description
_PROTECT_STATUS[n]	Status of the protection function (optional)

V2: Preprocessing 24

24.1 Brief description

Preprocessing

The programs stored in the directories for standard and user cycles can be preprocessed to reduce runtimes.

Preprocessing is activated via machine data.

Standard and user cycles are preprocessed when the power is switched on, i.e. as an internal control function, the part program is translated (compiled) into a binary intermediate code optimized for processing purposes.

All program errors that can be corrected by means of a compensation block are detected during preprocessing. In addition, when the program includes branches and check structures, a check is made to ensure that the branch destinations are present and that structures are nested correctly.

The full scope of control functionality is available:

- Override influence
- Reactions to data and signals that are input by the PLC or the operator
- Current block display
- The programs can be processed in single block mode (SBL1 and SBL2). Block searches can be executed. The compilation cannot be stored; it is concealed from the user and regenerated every time the power is switched on.

Preprocessing can be used:

- To optimize the runtimes of part programs with high-level language components (branches, check structures, motion-synchronous actions)
- CPU time intensive part programs (e.g. stock removal cycles)
- Faster processing of time-critical sections (e.g. program continuation after preprocessing stop during rapid deletion of distance-to-go, or return stroke, or in the tool change cycle).

General information

Preprocessing standard and user cycles is possible. The processing time of part programs can then be reduced without restricting the control functionality.

The standard and user cycles are preprocessed when machine data is set accordingly:

MD10700 $MN_PREPROCESSING_LEVEL (program preprocessing level)

Preprocessing is carried out program-specifically. It is possible to mix preprocessed part programs and part programs interpreted in ASCII format. Preprocessing serves to reduce incidental times.

Memory is required for preprocessing cycles. You can optimize your memory in two ways:

- The program to be executed can be shortened with the command `DISPLOF` (display off).
- MD10700 $MN_PREPROCESSING_LEVEL has been expanded by bit 2 and 3. This allows selective cycle preprocessing of the individual directories (e.g. user cycles).

MD10700 $MN_PREPROCESSING_LEVEL has been expanded by bit 4. This allows you to select preprocessing for user cycles from the _N_CMA_DIR directory.

MD10700 $MN_PREPROCESSING_LEVEL has been expanded by bit 5. This allows selective preprocessing of the specific individual user cycles that have the command `PREPRO` after the PROC instruction.

Pretranslated cycles are stored in the dynamic NC memory by default. MD10700 $MN_PREPROCESSING_LEVEL has been expanded by bit 6. This allows specifying that the compiled programs that are now stored in the dynamic NC memory and no longer have enough space can be stored in static NC memory.

Functionality

The programs stored in the directories for standard and user cycles are preprocessed when the power is switched on, i.e. the part program is translated (compiled) into an intermediate binary code optimized for processing purposes. The compilation is processed when called.

Runtime optimization

The preprocessing function is primarily suited for optimizing the runtimes of part programs with high-level language components (branches, check structures, motion-synchronous actions).

While branches and check structures are invalidated by a search through all blocks (block start) when part programs are interpreted in ASCII format (active as default), a branch is made directly to the destination block in a preprocessed part program.

The runtime differences between branches and check structures are thus eliminated.

Example of a preprocessing runtime:

Runtime reduction by 30% with active compressor

```
DEF INT COUNTER

Target: G1 G91 COMPON

G1 X0.001 Y0.001 Z0.001 F100000

COUNTER=COUNTER +1

COUNTER=COUNTER -1

COUNTER=COUNTER +1

IF COUNTER<= 100000 GOTOB TARGET
```

CPU time intensive programs and programs with symbolic names are processed faster.

Runtime-critical sections (e.g. continuation of processing after deletion of distance-to-go or preprocessing stop in cycles) can be processed faster.

If the interrupt routine is available as a preprocessed cycle, processing can be continued more rapidly after the program interrupt.

24.2 Program handling

Activation/Deactivation

Cycles are preprocessed on POWER ON if the following machine data is set:

MD10700 $MN_PREPROCESSING_LEVEL, bit 1 (program preprocessing level)

Bit	Value	Meaning
0		No preprocessing
	0	Call description of cycles is not known by default. Cycles, like normal subprograms, have to be stated as external before the cycle call. This is a sensible setting when no cycles with call parameters are used.
	1	During control power-up, the call description of the cycles is generated. All user cycles (_N_CUS_DIR directory) and Siemens cycles (_N_CST_DIR directory) with transfer parameters can be called up without external statement. Changes to the cycle-call interface do not take effect until the next POWER ON. The following machine data must be set: MD18170 $MN_MM_NUM_MAX_FUNC_NAMES (number of auxiliary actions) MD18180 $MN_MM_NUM_MAX_FUNC_PARAM (number of auxiliary parameters for cycles)
1	1	During control power-up, all cycles are preprocessed into a compilation optimized for processing. All user cycles (_N_CUS_DIR directory) and standard cycles (_N_CST_DIR directory) are processed at high speed. Program changes to cycle programs do not take effect until the next POWER ON.
2	1	During control power-up, the standard cycles in the _N_CST_DIR directory are pre-processed in a compilation optimized for processing.
3	1	During control power-up, the user cycles in the _N_CUS_DIR directory are prepro-cessed in a compilation optimized for processing.
4	1	Preprocessing of user cycles from the directory _N_CMA_DIR
5	1	Preprocessing the user cycles with the `PREPRO` command in the PROC instruction line. Unmarked files of directories marked with bits 1-4 are not preprocessed. If bit 0, then control of the preprocessing is carried out in accordance with the speci-fications of bits 0-4.
6	0	The compilation is stored in the dynamic NC memory as long as free memory is still available. If sufficient memory is not available, preprocessing is aborted. The size of the dynamic NC memory is obtained with this machine data: MD18351 $MN_MM_DRAM_FILE_MEM_SIZE.

The areas occupied in the dynamic NC memory by the compilation are visible to the user.

Bit combinations are permissible.

Compiling

In the directories for standard cycles: _N_CST_DIR, _N_CMA_DIR and user cycles: Subprograms (_SPF file extension) located in _N_CUS_DIR, and, if necessary, the subprograms marked with PREPRO are compiled. The name of the compilation corresponds to the original cycle with extension _CYC.

Note

Program changes to precompiled programs do not take effect until the next POWER ON.

Access rights

The preprocessed program can only be executed, but not read or written. The compilation cannot be modified or archived. The original cycles _SPF files are not deleted.

The compilation is not changed when the ASCII cycle is altered, i.e. changes do not take effect until after the next POWER ON.

Memory requirements

The memory requirement for compiled cycles is approximately factor 2 in addition to the ASCII part program.

The memory requirements for variables defined in the part programs are defined by the following machine data:

MD28020 $MC_MM_NUM_LUD_NAMES_TOTAL (number of local user variables)

MD28010 $MC_MM_NUM_REORG_LUD_MODULES (number of modules for local user variables with REORG)

MD28040 $MC_MM_LUD_VALUES_MEM (memory size for local user variables)

MD18242 $MC_MM_MAX_SIZE_OF_LUD_VALUE (memory block size for LUD/GUD values)

References:
Function Manual, Extended Functions; Memory Configuration (S7)

While preprocessing is in progress, the amount of memory required is the same as if the preprocessed program were called on the first subprogram level.

When programs are preprocessed after POWER ON, a name is counted for each branch destination/label as if it were a variable. These names must be taken into account in the following machine data:

MD28020 $MC_MM_NUM_LUD_NAMES_TOTAL (number of local user variables)

Example:

Program code	**Comment**
`PROC NAMES`	`; 1 name`
`DEF INT VARIABLE, ARRAY[2]`	`; 2 names`
`START:`	`; 1 name, only for preprocessing`
`FOR VARIABLE = 1 TO 9`	`; 1 name, only for preprocessing`

Program code	Comment
`G1 F10 X=VARIABLE*10-56/86EX4+4*SIN(VARIABLE/3)`	
`ENDFOR`	`; 1 name, only for preprocessing`
`M17`	

In order to execute this program normally, the following machine data must specify at least 3 names:

MD28020 $MC_MM_NUM_LUD_NAMES_TOTAL

Six names are required to compile this program after POWER ON.

Preprocessed programs/cycles are stored in the dynamic NC memory. The space required for each program must be flashed over unmodified as outlined above. Adjustment of the memory mapping in the static NC memory is required only if bit 6 = 1 is set in the following machine data:

MD10700 $MN_PREPROCESSING_LEVEL (program processing level)

In this case, the program compilations for which there is insufficient space in the dynamic NC memory are stored in the static NC memory.

Examples for appropriate machine data settings can be found under "Examples" in the Subsection "Preprocessing in the dynamic NC memory".

24.3 Program call

Overview

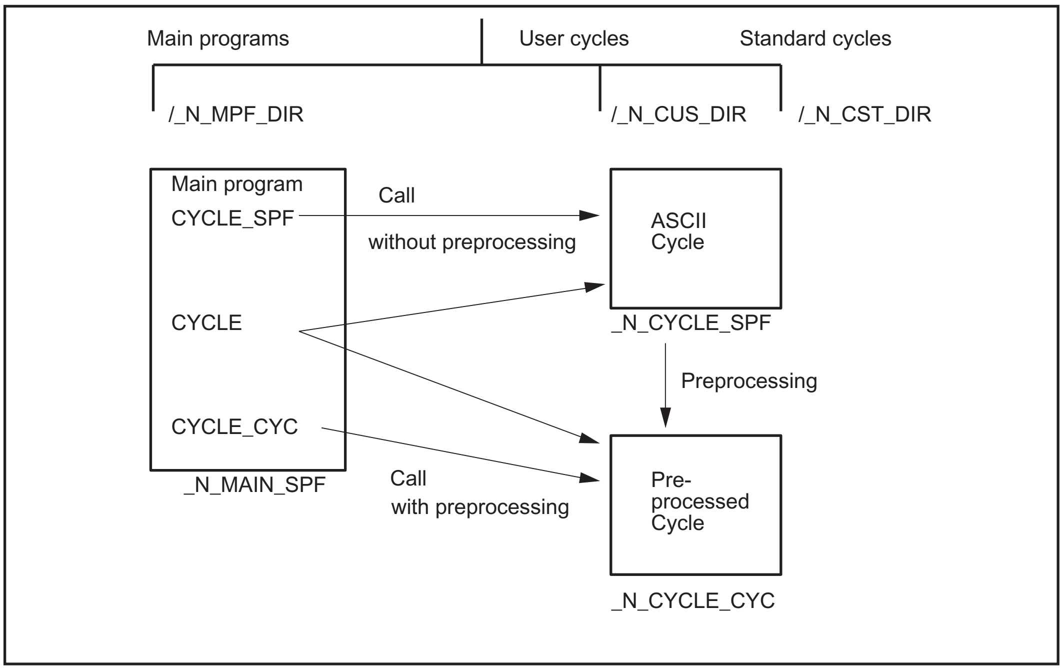


Figure 24-1 Generation and call of preprocessed cycles without parameters

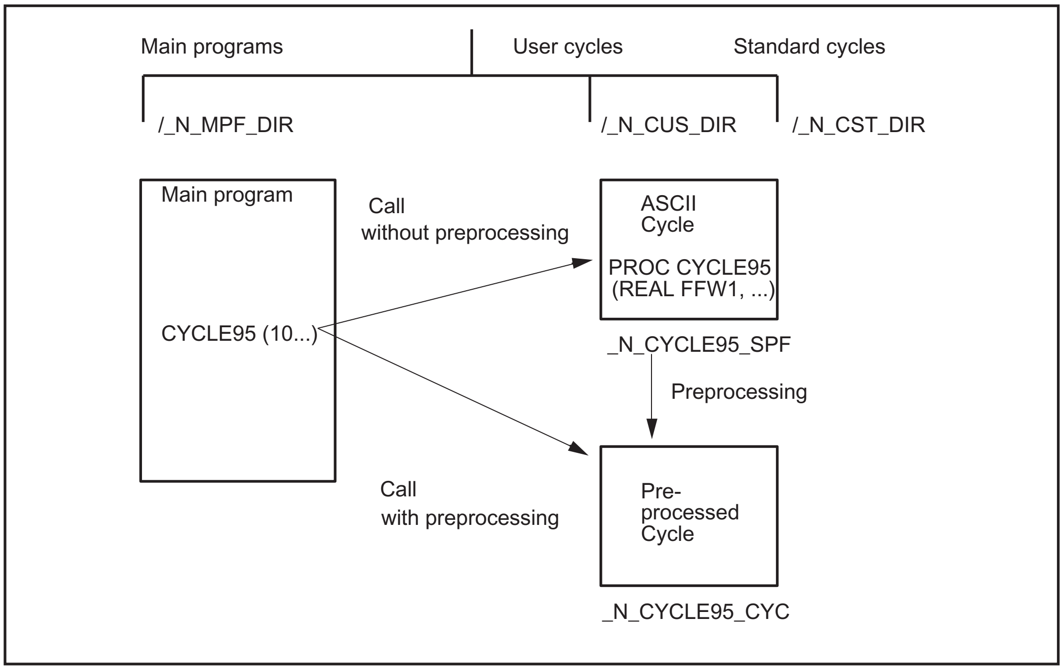


Figure 24-2 Generation and call of preprocessed cycles with parameters

Call

- Compiled cycle: A compiled cycle is called in exactly the same way as a normal subroutine.
  Example: CYCLE
- Preprocessing is activated: The compiled cycle is called instead of the ASCII cycle.
  - If the subroutine is called explicitly with extension _SPF, then the ASCII cycle is called even if a compilation is available.
    Example: CYCLE_SPF; ASCII cycle call
  - If the subroutine is called explicitly with extension _CYC, then the preprocessed cycle is called (assuming that it is available). An error message is output if no compilation is available.
    Example: CYCLE_CYC; calls a precompiled cycle
  - If bit 5 is set and a file that is not marked with `PREPRO` is called explicitly with the extension _CYC, an error message is issued with Alarm 14011.
- If a subroutine is called without explicit extension, an attempt is first made to load the program. If this is not possible (not marked with `PREPRO`), an attempt is made to load the SPF program.

- The change to an external language mode with G291 is rejected and an alarm issued. When calling a precompiled cycle, an explicit change is made into the Siemens language mode.
- When the subroutine is called, it is checked whether the compiled file is older than the cycle. If so, the compile file is deleted and an alarm issued. The user must preprocess the cycles again.

Note

Only cycles without parameters may be called with the extension _SPF or _CYC.

It is not permissible that PUDs are used in cycles that have been preprocessed. The PUDs are created in the calling main program. At the time of compilation after POWER ON, this data is not known to the cycles.

The actual program display shows whether the current ASCII cycle or the compilation has been called (extension _SPF or _CYC).

Call condition

All cycles in the cycle directories must be compiled before preprocessing is activated. Non-compiled cycles in _N_CUS_DIR and _N_CST_DIR which were loaded after POWER ON, for example, can only be called through explicit specification of extension _SPF.

If preprocessing is active and bit 5 is set, all programs that do not start with the PREPRO PROC instruction are not precompiled.

Syntax check

All program errors, which can be corrected using a correction block, are already identified at the time of preprocessing. In addition, when the program includes branches and check structures, a check is made to ensure that the branch destinations are present and that structures are nested correctly.

Branch destinations and labels must be unique in the program.

After the errors detected during preprocessing have been corrected, preprocessing must be started again by means of an NC POWER ON.

24.4 Constraints

Availability of the "preprocessing" function

The function is an option ("Program pre-processing"), which must be assigned to the hardware via the license management.

Language scope

The full vocabulary of the NC language is available in the part program.

There are no restrictions on the calculation of measured process variables and in the reaction to signals from the process and other channels (override, deletion of distance-to-go, motion-synchronous actions, channel coordination, interrupt processing, etc.).

Axis name

Part programs are compiled independently of channels. For this reason, the geometry and channel names set in the following machine data must be **identical** in all channels if they are **used directly in the precompiled cycles**:

MD20060 $MC_AXCONF_GEOAX_NAME_TAB (name of the geometry axis in the channel)

MD20080 $MC_AXCONF_CHANAX_NAME_TAB (name of the channel axis in the channel)

Generally speaking, axis names are not used directly in machining cycles since cycles are written as follows:

- independently of channels and
- independently of the axis names defined on the machine.

The axes to be traversed are addressed indirectly via machine data or transferred as parameters:

- Indirect axis programming:
  - IF $AA_IM[**AXNAME($MC_AXCONF_CHANAX_NAME_TAB[4])**] > 5
    ; This branch will pass through if the actual value of the 5th channel axis
    ; with reference to the machine coordinate system is greater than 5.
  - G1 **AX[AXNAME($MC-AXCONF-GEOAX-NAME-TAB[0])**] = 10
    F1000 G90.
    ; Traverse the 1st geometry axis to the value 10.
    ENDIF
- Transfer of axis to be traversed from the main program:
  - Cycle definition
    PROC DRILL(AXIS DRILL_AXIS)
    WHILE $AA_IW[DRILL_AXIS] > -10
    G1 G91 F250 AX[DRILL_AXIS] = -1
    ENDWHILE
  - Call from the main program
    DRILL(Z)

24.5 Examples

24.5.1 Preprocessing individual files

Program code	Comment
`PROC UP1 PREPRO`	`; Preprocessing if bit 5 = 1`
	`; in PREPROCESSING_LEVEL`
`N1000 DEF INT COUNTER`	
`N1010 TARGET: G1 G91 COMPON`	
`N1020 G1 X0.001 Y0.001 Z0.001 F100000`	
`N1030 COUNTER=COUNTER+1`	
`N1040 COUNTER=COUNTER-1`	
`N1050 COUNTER=COUNTER+1`	
`N1060 IF COUNTER<=10 GOTOB TARGET`	
`N1070 M30`	
`PROC UP2`	
`N2000 DEF INT VARIABLE, ARRAY[2]`	
`N2010 IF $AN_NCK_Version < 3.4`	
`N2020 SETAL(61000)`	
`N2030 ENDIF`	
`N2040 START:`	
`N2050 FOR VARIABLE = 1 TO 5`	
`N2060 G1 F1000 X=VARIABLE*10-56/86EX4+4*SIN(VARIABLE/3)`	
`N2070 ENDFOR`	
`N2080 M17`	
`PROC MAIN`	
`N10 G0 X0 Y0 Z0`	
`N20 UP1`	
`N30 G0 X10 Y10 Z10`	
`N40 UP2`	
`N50 G0 X100 Y100`	
`N60 UP3`	
`N70 G0 X10 Y10`	
`N80 M30`	

Sample constellations:

a) Bit 5 = 1

MD10700 $MN_PREPROCESSING_LEVEL=45 ; bit 0, 2, 3, 5

Subprogram UP1 is pretranslated, and the call description is generated.

Subprogram UP2 is not pretranslated, but the call description is generated.

b) Bit 5 = 0

MD10700 $MN_PREPROCESSING_LEVEL=13 ; bit 0, 2, 3,

Both subprograms are pretranslated, and the call description is generated.

c) Example of an **invalid** subprogram with activated compiling:

Program code	Comment
`PROC SUB1 PREPRO`	
`G291`	`; ← Alarm when compiling, G291 not possible`
`G0 X0 Y0 Z0`	
`M17`	

24.5.2 Preprocessing in the dynamic NC memory

Machine data for preprocessing only in the dynamic NC memory with selective selection:

Program code	Comment
	`; Bit 5 = 1 Selective program selection`
	`; Bit 6 =0 no switch to`
	`; static NC memory if`
	`; dynamic NC memory is full`
`N30 $MN_MM_DRAM_FILE_MEM_SIZE = 800`	`; Reserve space`
`N40 $MN_PREPROCESSING_LEVEL = 63`	`; Bit 0-5 = 1`
`M17`	

Machine data for preprocessing in the dynamic NC memory with the option of using the static NC memory and selective selection:

Program code	Comment
	`; Bit 5 = 1 Selective program selection`
	`; Bit 6 = 1 switch to static`
	`; NC memory if dynamic NC`
	`; memory full`
`N30 $MN_MM_DRAM_FILE_MEM_SIZE = 800`	`; Reserve space`
`N40 $MN_PREPROCESSING_LEVEL = 127`	`; Bit 0-6 = 1`
`M17`	

24.6 Data lists

24.6.1 Machine data

24.6.1.1 General machine data

Number	Identifier: $MN_	Description
10700	PREPROCESSING_LEVEL	Program preprocessing level
18242	MM_MAX_SIZE_OF_LUD_VALUE	Maximum LUD-variable array size

24.6.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG (DRAM)
28020	MM_NUM_LUD_NAMES_PER_PROG	Number of local user variables (DRAM)
28040	MM_LUD_VALUES_MEM	Memory size for local user variables (DRAM)

W5: 3D tool radius compensation - 840D sl only 25

25.1 Function

25.1.1 Introduction

3D circumferential milling and 3D face milling

3D tool radius compensation (3D-TRC) is used to process contours with tools that can be controlled in their orientation independently of the tool path and shape.

SINUMERIK provides the 3D-TRC in different variants, which are used in combination with the following manufacturing processes:

- Circumferential milling (Page 936) for structural components
- Face milling (Page 942) for free-form surfaces

Tool orientation

For the 3D-TRC, a distinction must be made between the two scenarios of a tool with a fixed orientation and a tool with a changeable orientation.

The tool radius compensation with a fixed tool for circumferential milling is designated as a 2½D-TRC. This is the conventional tool radius compensation. Circumferential milling with a functionality that goes beyond this scope of performance is not possible for the compensation with a fixed tool. For circumferential milling, there is only a 3D-TRC if the tool orientation is changeable. This means, in addition to the 3 degrees of freedom for positioning the tool (generally 3 linear axes), two additional degrees of freedom (2 rotary axes) are also needed for setting the tool orientation (5-axis machining).

During face milling, the tool orientation can remain constant or be variable. The advantages of variable orientation are:

- better approximation of the end contour
- increased chip performance
- more freedom in selecting the shape of the tool
- a wider range of surfaces that can be processed (undercuts)

Note

5D-WRK

The 3D-TRC is also called 5D-TRC below, because 5 degrees of freedom are available in this case for defining the position of the tool in space relative to the workpiece.

Parameter assignment

The machine and setting data set for the 2D-TRC are also in effect for the 3D-TRC.

Reference:
Function Manual Basic Functions; Chapter "W1: tool offset"

In addition to this, there is special system data that is only relevant to 3D-TRC. See Chapter "Parameter assignment (Page 950)" regarding settings of this data.

Activation/deactivation

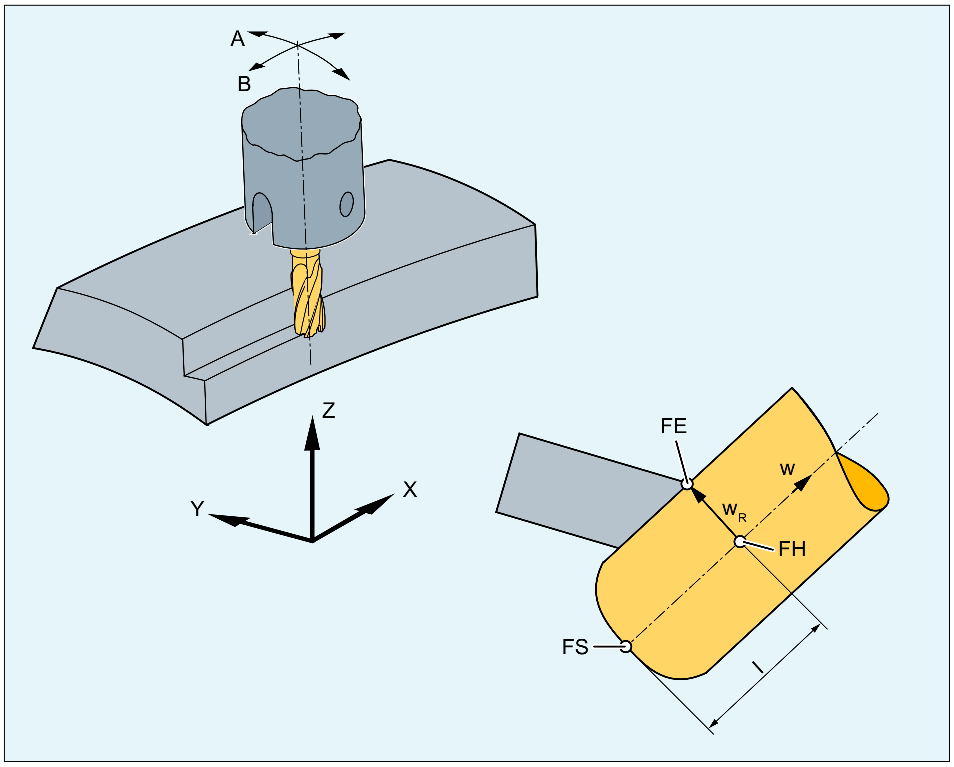
The selection of the 3D-TRC variant is made in the NC program via G commands of Group 22 (tool offset type). The actual activation and deactivation are done via the G commands of Group 7 (tool radius compensation).

See Chapter "Programming (Page 951)".

25.1.2 Circumferential milling

The 3D circumferential milling is used for machining contours with tools having an orientation that is not constant during processing. Processing is done using the peripheral surface of the tool. If the tool is cylindrical, ruled surfaces (e.g. cones, cylinders, etc.) are created during this processing. Other forms of milling machines can also be used, however (e.g. form cutters). The tool radius compensation needed for this type of processing is an extension of the conventional 2½D-TRC.

The ratios for 3D circumferential milling are shown in the following figure:



FE:	Milling tool machining point
FS:	Milling tool tip
FH:	Milling tool reference point
l:	Insertion depth
w:	Tool vector
w_r:	Vector from the milling tool reference point to the milling tool machining point with length of shank radius R

Figure 25-1 Circumferential milling

Insertion depth

The insertion depth of the milling tool is the distance of the milling tool reference point from the tip of the tool.

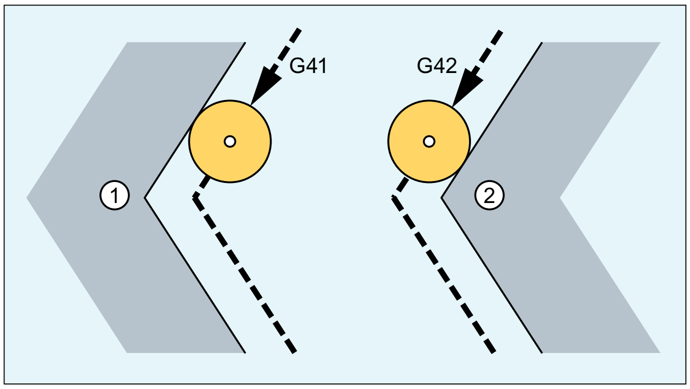
The milling tool reference point is the vertical projection of the milling tool machining point on the programmed path to the longitudinal axis of the tool.

The position of the machining point on the peripheral surface of the tool is set with the insertion depth.

Via the address ISD (InSertion Depth), the insertion depth of the tool for circumferential milling and **active** 3D tool radius compensation can be changed (see Chapter "Programming (Page 951)").

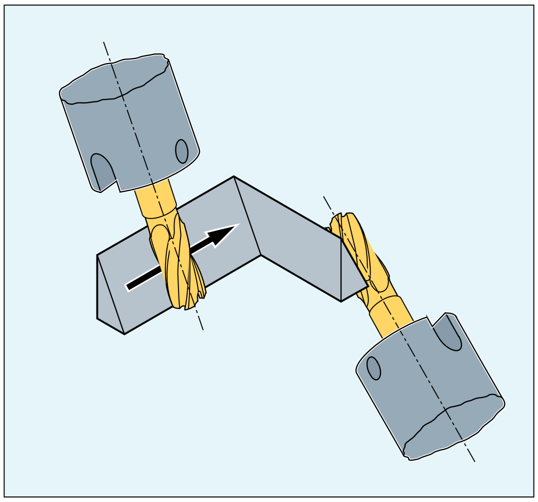
25.1.2.1 Corners in circumferential milling

Inside and outside corners are handled separately. The terms inside corner and outside corner are dependent on the tool orientation.



①	Inside corner
②	Outside corner

When the orientation changes at a corner, for example, the corner type may change while machining is in progress:



Whenever this occurs, the machining operation is aborted with an error message.

25.1.2.2 Behavior at outer corners

At outside corners during circumferential milling with 3D-TRC analogous to the conditions in the 2½D-TRC, the G commands of Group 18 (corner behavior tool offset) are evaluated:

- G450: Transition circle (tool travels around workpiece corners on a circular path)
  Outside corners are treated like circles with a radius of 0, where the circular plane stretches from the end tangent of the first block and the start tangent of the second block. Thus, an orientation change during the block transition is also possible. Thus, a block is always inserted at an outside corner.
  In contrast to the solution for the 2½D-TRC, the contour element inserted at an outside corner is always a circle with a radius of 0, which is affected by the tool radius compensation like on any other programmed path. It is not possible to insert conical sections in place of the circles. The address DISC therefore has no relevance in this case and is not evaluated.
- G451: Intersection of equidistant paths (tool backs off from the workpiece corner)
  The intersection is determined by extending the offset curves of the two participating blocks and defining the intersection of the two blocks at the corner in the plane perpendicular to the tool orientation.
  If an intersection is found in the plane perpendicular to the tool, this does not mean that the curves also intersect in space. Rather the curves in the direction of the tool longitudinal axis are considered, which are generally a certain distance apart. The positional offset is eliminated over the entire block length in direction of the tool.
  The way this offset is processed in tool direction at outside corners is the same as for inside corners.

For nearly tangential transitions, the behavior for an active G450 is the same as for G451 (limit angle can be set via MD20210 $MC_CUTCOM_CORNER_LIMIT). Conversely, if G451 is active, a circle is also inserted (procedure as for G450) if there is no intersection or if the corner angle exceeds a specific value (MD20230 $MC_CUTCOM_CURVE_INSERT_LIMIT).

Behavior during orientation changes

The intersection procedure (G451) is not used when at least one block containing a change to the tool orientation was inserted between the traversing blocks in question. In this case a circle is always inserted at the corner.

The G commands of Group 27 (too offset for orientation change at outside corners) can be used to define whether orientation changes, which have been programmed between the two traversing blocks that form the corner, are executed before the beginning of the inserted circle set or at the same time.

- ORIC: Orientation changes at outside corners are superimposed on the circle block to be inserted.
- ORID: Orientation changes are performed before the circle block.

If two or more blocks with orientation changes (e.g. A2=... B2=... C2=...) are programmed between two traversing blocks and ORIC is active, then the circular block to be inserted is divided between these intermediate blocks corresponding to the amount of the individual angle changes. Further programmed intermediate blocks without traversing and orientation motions are executed at the programmed positions.

If ORID is active, all inserted blocks (both those with and without orientation movement) are executed at the end of the first of the two traversing blocks. The tangent in the end point of the first traversing block is used for the offset calculation. The circle block with constant orientation is inserted immediately before the second traversing block.

See also "Example 1: Orientation change at outside corner during 3D circumferential milling (Page 966)".

Intermediate blocks without relevant traversing information

Intermediate blocks without relevant traversing information (neither tool orientation nor position of geometry axes are changed) are permissible. The intersection procedure is applied to the adjacent blocks as if these intermediate blocks did not exist. In the same manner, tool direction motions in the tool direction may also be programmed in intermediate blocks.

25.1.2.3 Behavior at inside corners

For the 3D-TRC, only adjacent traversing blocks are taken into consideration for the calculation of the intersecting point.

If the orientation is not changed at the block limit, then the contour need only be considered in the plane vertical to the tool axis. In this case, the tool cross-section is a circle which touches the two contours. The geometric relations in this plane are identical to those for $2^1/_2$D-TRC.

Behavior during orientation changes

If the orientation changes on a block transition, the tool moves in the inside corner so that it is constantly in contact with the two blocks forming the corner.

Path segments must be long enough to ensure that the contact points of the tool do not cross the block limits into other blocks when the orientation changes at an inside corner.

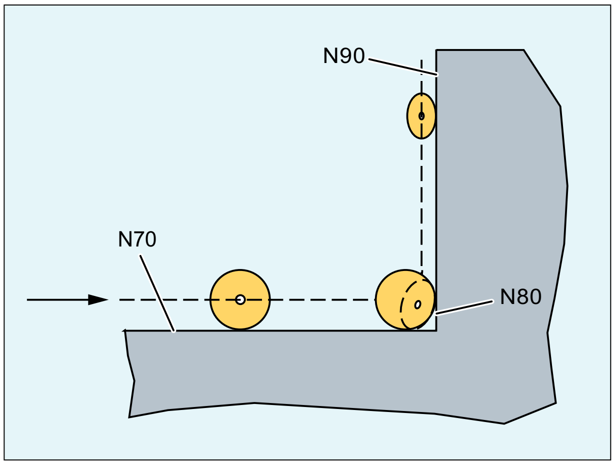


Figure 25-2 Orientation changes at inside corner

When the orientation changes in a block that is one of the two blocks forming the inside corner, then it is no longer possible to adhere to the programmed relationship between path position and associated orientation. This is because the orientation must reach its end value even though the path end position is not reached. This response is identical to the response of synchronized axes with $2^1/_2$ D tool radius compensation.

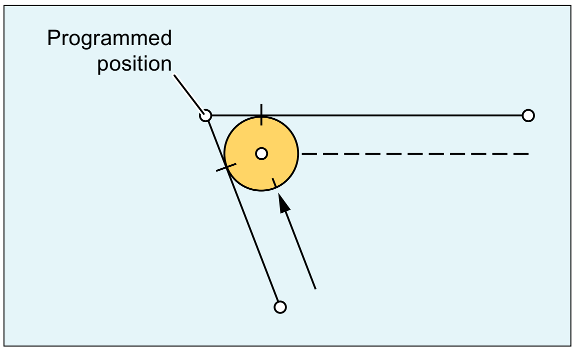


Figure 25-3 Path end position and change in orientation at inside corners

See also "Example 2: Orientation change at inside corner during 3D circumferential milling (Page 969)".

Change in insertion depth

Generally speaking, the contour elements that form an inside corner are not positioned on the plane perpendicular to the tool. This means that the contact points between the two blocks and the tool are at different distances from the tool tip.

This means: The insertion depth (ISD) changes abruptly from the 1st to the 2nd block at an inside corner.

To ensure that this difference in depth is not an abrupt step change, it is distributed continuously among the blocks involved during interpolation. The depth-compensating motion is executed in the current tool direction.

This solution prevents the contour from being violated by cylindrical tools if the length of the tool prevents the cutter contact point on the lateral surface of the cutter leaving the range in which machining is possible.

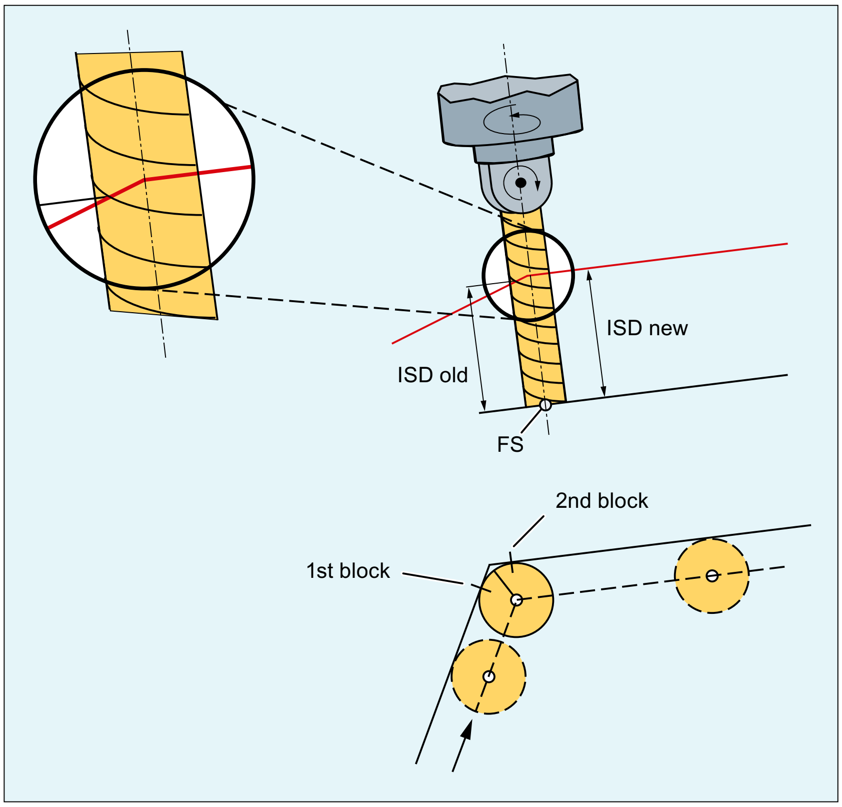


Figure 25-4 Change in insertion depth

25.1.2.4 Monitoring of path curvature

The path curvature is monitored for an unchangeable orientation in the following way:

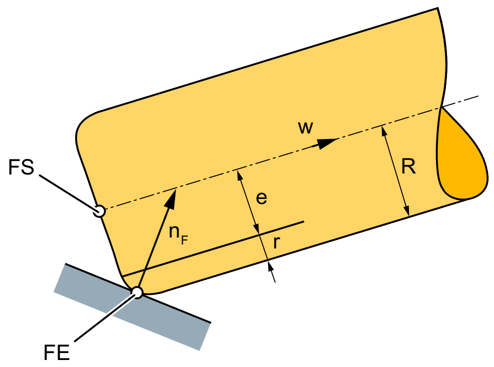
1. The contour in each block is projected on the plane that is orthogonal to the tool orientation.
2. Then the curvature of the projection is analyzed. Areas with too much curvature (tool does not fit in the curve) are cut out.
3. The remaining offset contours overlap each other and thus form the global offset contour.

25.1.3 Face milling

Face milling is used to process curved surfaces of any kind. In this case, the longitudinal axis of the tool and the surface normal vector are more or less parallel. In contrast, the longitudinal axis of the tool and the surface normal vector of the surface to be machined in a 3D circumferential milling operation are at right angles to one another.

Information about the surface to be machined is absolutely essential for face milling operations, i.e. a description of the linear path in space is not sufficient. The tool shape must also be known in order to implement the tool offset (the term "Tool radius compensation" is not appropriate in this case).

The conditions for face milling are shown in the following figure:

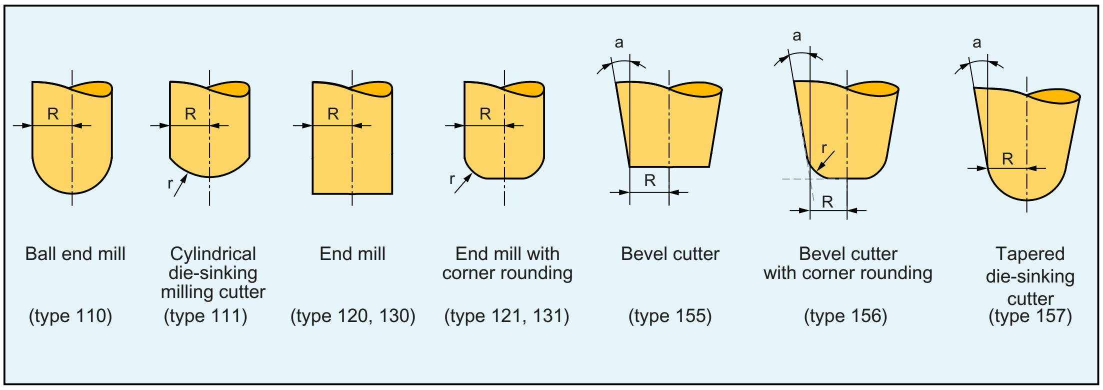


FE:	Milling tool machining point
FS:	Milling tool tip
n_F:	Surface normal vector
w:	Tool vector
R:	Shaft radius
r:	Corner radius
e:	Difference between the shank radius R and corner radius r

Figure 25-5 Face milling with a torus cutter

25.1.3.1 Tool shapes and tool data for face milling

An overview of the tool shapes, which may be used for face milling operations and relevant tool data are listed in the following. The shape of the tool shaft is not taken into account. Therefore, the tool types 120 and 155, for example, have an identical effect.


R
r
a
Ball end mill (type 110)
Cylindrical die-sinking milling cutter (type 111)
End mill (type 120, 130)
End mill with corner rounding (type 121, 131)
Bevel cutter (type 155)
Bevel cutter with corner rounding (type 156)
Tapered die-sinking cutter (type 157)

R	= shank radius (tool radius)
r	= corner radius
a	= angle between the tool longitudinal axis and upper end of the torus surface

Cutter type	Type No.	R	r	a
Ball end mill	110	> 0	-	-
Cylindrical die-sinking milling cutter	111	> 0	> R	-
End mill, angle head cutter	120, 130	> 0	-	-
End mill, angle head cutter with corner rounding	121, 131	> r	> 0	-

Cutter type	Type No.	R	r	a
Bevel cutter	155	> 0	-	> 0
Bevel cutter with corner rounding	156	> 0	> 0	> 0
Tapered die-sinking cutter	157	> 0	-	> 0

- : is not evaluated.

Tool data	Tool parameters	
Tool dimensions	**Geometry**	**Wear**
R	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

Note

If, in the NC program, a type number is specified that differs from that in the diagram, then the system automatically uses tool type 110 (ball end mill).

Note

An alarm is output if the tool data limit values are violated.

Geometry and wear values of tool data have an additive effect, i.e. the tool form that is actually used for the offset calculation is always described by the sum values.

In the CUT3DFD function (differential tool), only the wear value is considered as an offset value. The geometry values are only necessary in this case for describing the tool form.

Tool length offset

The tool tip (intersecting point between the tool longitudinal axis and the workpiece surface) is considered the reference point for the length offset.

Tool change

A new tool with changed dimensions (R, r, a) or other form can only be specified during the first activation of the tool offset (i.e. during the transition from G40 to G41 or G42) or for an already active offset only for new programming of G41 or G42. This rule does not apply to any other tool data, e.g. tool lengths, so that tools can be loaded without reprogramming G41 or G42.

25.1.3.2 Face milling with a specified surface normal vector

Expanded options for orientation programming are provided for 3D face milling.

The tool offset for face milling cannot be calculated simply by specifying the path (e.g. of a line in space). The surface to be machined must also be known. The necessary information about the surface is provided to the controller with the surface normal vector.

The surface normal vector at the block beginning is programmed with A4, B4, C4, and the vector at the block end with A5, B5, C5. Components of the surface normal vectors which are not programmed are set to zero. The length of a vector programmed in this way is irrelevant.

A vector of zero length (all three components are zero) is ignored, i.e. the direction programmed beforehand remains valid, no alarm is generated.

If only the start vector is programmed (A4, B4, C4) in a block, then the programmed surface normal vector remains constant over the entire block. If only the end vector is programmed (A5, B5, C5), then large-circle interpolation is used to interpolate between the end value of the preceding block and the programmed end value. If both start and end vectors are programmed, interpolation according to the large circle principle is also performed between the two directions. The fact that the start vector may be reprogrammed in a block means that the direction of the surface normal vector can change irregularly on a block transition. Irregular transitions of the surface normal vector always occur in cases where there is no tangential transition between the surfaces (planes) involved, i.e. if they form an edge.

Once a surface normal vector is programmed, it remains valid until a new vector is programmed. In the basic position, the surface normal vector is set to equal the vector in the Z direction. This basic setting direction is independent of the active plane (G17 - G19). If ORIWKS is active, surface normal vectors refer to the active frame, i.e. when the frame is rotated, the vectors rotate simultaneously. This applies both to programmed orientations and to those that have been derived from the active plane. If ORIWKS is active, the surface normal vectors are brought along during a frame change. An orientation modified as the result of frame rotations is not returned to its original state on switchover from ORIWKS to ORIMKS.

It must be noted that the programmed surface normal vectors may not necessarily be the same as those used internally. This is always the case if the programmed surface normal vector is not vertical on the path tangent. A new surface normal vector is then formed, which lies within the plane spanned by the path tangent and programmed surface normal vector, but which is vertical on the path tangent vector. This orthogonality is required, because the path tangent vector and the surface normal vectors (must) always be vertical to one another in a real surface. However, since the two variables can be programmed independently of one another, they can contain information that is mutually contradictory. Orthogonalization ensures that the information contained in the path tangent vector has priority over the data in the surface normal vector. An alarm is output if the angle between the path tangent vector and the programmed surface normal vector undershoots the limit value, which is defined in the following machine data:

MD21084 $MC_CUTCOM_PLANE_PATH_LIMIT (minimal angle between surface normal vector and path tangent vector)

If a block is shortened (inside corner), then the interpolation range of the surface normal vector is reduced accordingly, i.e. the end value of the surface normal vector is not reached as it would be with other interpolation quantities such as, for example, the position of an additional synchronized axis.

In addition to the usual methods of programming orientation, it is also possible to refer the tool orientation to the surface normal vector and path tangent vector using the addresses LEAD (lead or camber angle) and TILT (side angle). The interpretation of the angle data depends on the setting in MD21094 $MC_ORIPATH_MODE (see Chapter "Path-relative orientation (ORIPATH, ORIPATHS, ORIROTC) (Page 99)"). The specification of the angle relative to the surface normal is only an expanded option of orientation programming at the end of the block. It does not imply that the lead and side angles reach their programmed values before the path end point is reached.

The resulting tool orientation is determined from the path tangent, surface normal vector, lead angle and side angle at the end of the block. This orientation is always implemented by the end of the block, particularly in cases where the block is shortened (at an inside corner). If the omitted path section is not a straight line in a plane, the lead and side angles generally deviate from their programmed values at the path end point. This is because the orientation has changed relative to the surface normal vector or path tangent vector when the absolute orientation of the tool is the same as at the original path end point.

25.1.3.3 Compensation on path

A special case must be examined with respect to face milling operations, i.e. that the machining point on the tool surface moves around. This may be the case on a torus cutter whenever surface normal vector n_F and tool vector w become collinear (i.e. the tool is at exact right angles to the surface) since it is not a single point on the tool that corresponds to this direction, but the entire circular surface on the tool end face. The contact point is therefore not defined for this orientation. A path point in which tool longitudinal axis and surface normal are parallel is therefore referred to below as a singular point or a singularity.

The above case is also meaningful in practical terms, e.g. in cases where a convex surface, which may have a vertical surface normal (e.g. hemisphere), must be machined with a perpendicular tool (e.g. face milling with constant orientation). The machining point on the contour remains fixed, but the machine must be moved to bring the machining point from one side of the tool to the other.

The problem described is only a borderline case (lead angle β = 0 and side angle y = 0). If the lead angle β = 0 and the side angle y has a low value, then the tool must be moved very rapidly (in borderline case in steps) to keep the machining point resulting from the milling conditions close to the arc-line forming the end face (see the following figure).

In principle, it is resolved by the smoothing of the surface normal and the smoothing of the contour.

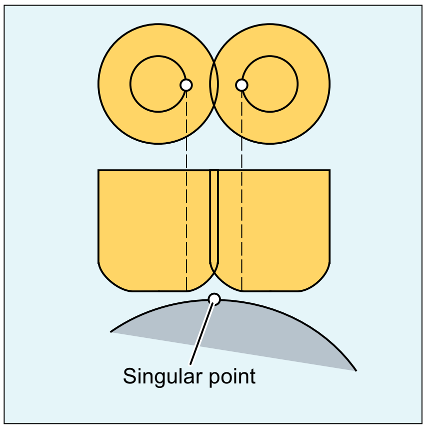


Figure 25-6 Switching of the processing point on the tool surface in the vicinity of a point in which the surface normal vector and the tool orientation are parallel.

Singularities do not only occur in isolated points, but can also occur over entire curves. This is the case, for example, if the curve to be interpolated is a plane curve (i.e. a curve with a constant osculating plane) and the tool is constantly aligned in parallel to the binormal vector (perpendicular to the osculating plane). A simple example of this is an arc in the X-Y plane, which is processed with a tool that is aligned parallel to the Z axis. For paths of this type, the tool offset is reduced to a tool length compensation. I.e. the tool is positioned in such a way that its tip lies on the programmed path.

For a jump-like transition between singular and non-singular curves, linear blocks must be inserted, just as in the handling of individual points, so that the processing point on the tool can migrate from the tool tip to the periphery (at outside corners or convex surfaces) or the paths must be shortened to prevent contour violations (at inside corners or concave surfaces).

It is therefore necessary to activate both the smoothing of the contours ("Top Surface" function) and the smoothing of the surface normal to ensure that the transitions run smoothly.

25.1.3.4 Corners in face milling

Two surfaces that do not tangentially transition into one another form an edge. The paths defined on the surfaces form a corner. This corner is a point of the edge.

The type of corner (inside or outside corner) is defined by the surface normals of the participating surfaces and the paths defined on them.

The surface normals of the two surfaces forming the edge may point in opposite directions of the overall surface (the front edge of one surface is continued on the rear edge of the second surface), see also the following Figure. Transitions of this type are not allowed and are rejected with an alarm.

The scalar product of the surface normal vector and (possibly variable) tool orientation on one corner/path must be positive at each point, i.e. it is not permissible to machine from the rear face of the surface. If this condition is not fulfilled, an alarm is output. The permissible ranges of validity of tool orientation for inside and outside corners are illustrated in the following Figure. These ranges are further restricted by the condition that the angle between the surfaces to be machined and the "steepest" surface line of the tool surface must not be lower than a particular machine data setting. The "steepest" surface line is a line at angle a to the tool longitudinal axis (this line is in the same direction as the tool longitudinal axis on cylindrical tools). This restriction must be imposed to ensure that the contact point on the tool does not leave the permissible range.

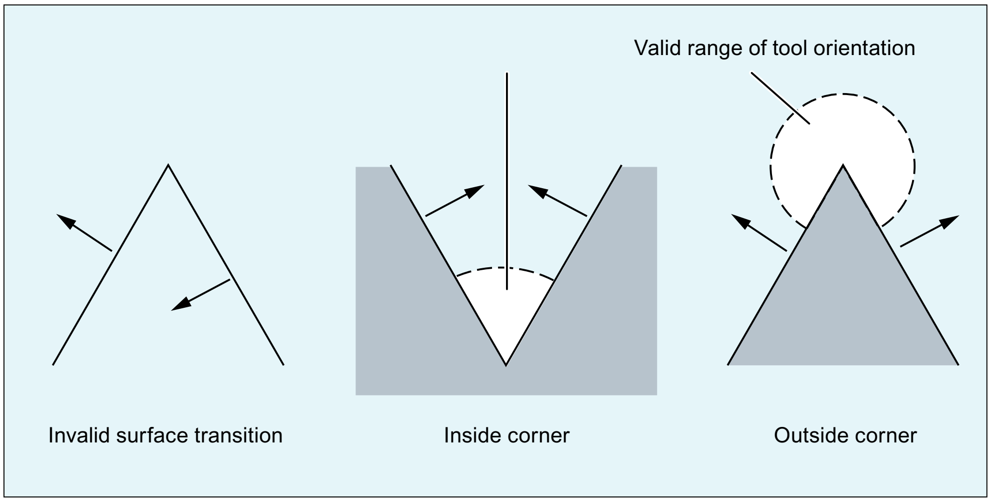


Figure 25-7 Corners in face milling

It is possible to insert blocks that contain no motion commands (e.g. auxiliary function outputs) and/or that include motions of geometry axes not involved in the path between two blocks which contain a path definition. In particular, orientation changes can also be programmed in such blocks. The only exception applies to the activation and deactivation of the 3D tool radius compensation function: Intermediate bocks with orientation changes are not permitted between the activation block and the first corrected block or between the last corrected block and the deactivation block. Other intermediate block are allowed, however.

25.1.3.5 Behavior at outer corners

In face milling, outside corners are treated like circles with a radius of 0, where the circular plane stretches from the end tangent of the first block and the start tangent of the second block. Thus, an orientation change during the block transition is also possible. Thus, a circle is always inserted as a contour element at an outside corner. The intersecting point process is not available for face milling.

Behavior during orientation changes

The G commands of Group 27 (too offset for orientation change at outside corners) can be used to define whether orientation changes, which have been programmed between the two traversing blocks that form the corner, are executed before the beginning of the inserted circle set or at the same time.

- ORIC: Orientation changes at outside corners are superimposed on the circle block to be inserted.
- ORID: Orientation changes are performed before the circle block.

If two or more blocks with orientation changes (e.g. A2=... B2=... C2=...) are programmed between two traversing blocks and ORIC is active, then the circular block to be inserted is divided between these intermediate blocks corresponding to the amount of the individual angle changes. Further programmed intermediate blocks without traversing and orientation motions are executed at the programmed positions.

If ORID is active, all inserted blocks (both those with and without orientation movement) are executed at the end of the first of the two traversing blocks. The tangent in the end point of the first traversing block is used for the offset calculation. The circle block with constant orientation is inserted immediately before the second traversing block.

See also "Example 1: Orientation change at outside corner during 3D circumferential milling (Page 966)".

25.1.3.6 Behavior at inside corners

The position of the tool in which it is in contact with the two surfaces forming the corner must be determined at an inside corner. The contact points must be on the paths defined on both surfaces. It is not usually possible to solve this problem exactly since, when the tool is moved along the path on the first surface, it normally touches a point on the second surface which is not on the path.

For this reason, the tool is not moved along the path on the first surface, but deviates from the path in such a way as to ensure that the distance between the contact points and the relevant contours in the position in which the tool contacts both surfaces is minimal, see also the following Figure.

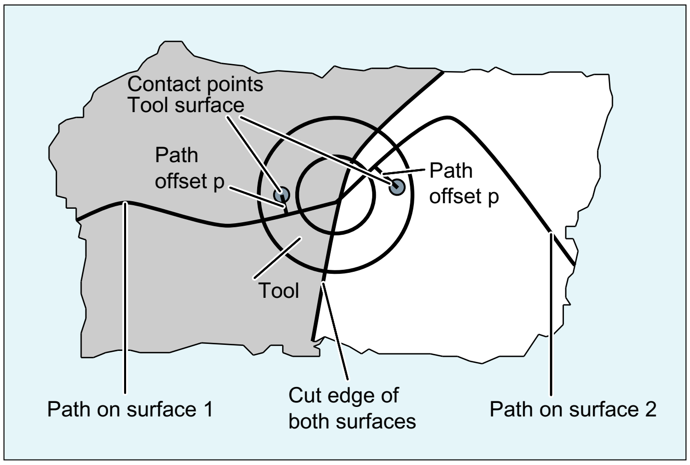


Figure 25-8 Inside corner with face milling (view in direction of longitudinal axis of tool)

Note

The amount by which the contact points deviate from the programmed contour will generally be small since the explanatory example shown in the Figure, in which the machining point "changes" the cutter side at an inside corner (the value of the angular difference Ψ about the tool longitudinal axis between the two contact points on the tool surface is approximately 180°) is more likely to be the exception (see also the following Figure, on the right). The angle Ψ will normally stay almost constant so that the distance between the contact points on the tool surface will be relatively small (see also the following Figure, on the left).

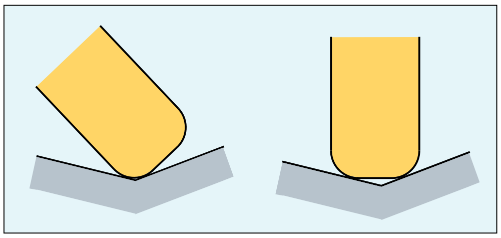


Figure 25-9 Machining at inside corners

The difference between the programmed point on the path and the point actually to be approached (path offset p) is eliminated linearly over the entire block length. Differences resulting from inside corners at the block start and block end are overlaid. The current difference in a path point is always perpendicular to the path and in the surface defined by the surface normal vector.

If the tool orientation at an inside corner is not constant, the change in orientation is implemented in the same way as described in Subsection "Behavior at inside corners" for 3D circumferential milling, i.e. the tool is moved in the corner so that it contacts the two adjacent surfaces at the block start, block end and at two points $^1/_3$ and $^2/_3$ of the change in orientation. A 3rd-degree polynomial is used to interpolate between these 4 points.

A variable tool orientation in a block that is shortened owing to an inside corner is also treated in the same way as described in Subsection "Behavior at inside corners" for 3D milling (Page 940), i.e. the entire change in orientation is executed in the shortened block. Consequently, the functional relationship between path tangent, surface normal and tool orientation also changes. This results in new, previously nonexistent singularities or impermissible side angles (at points which are virtually singular) occurring in the shortened block. If this type of situation is detected during processing of an inside corner, the machining operation is aborted with an alarm. No block division takes place at the singular points since the compensatory motions this would involve frequently cause contour violations and the change in machining side on the tool is not generally intended or even foreseen by the user. The alarm is also output during examination of an inside corner if the singularity occurs in the second of the two blocks without the transition to the next block being considered. The system does not therefore detect that a block of this type will form an inside corner in conjunction with the following block and that the singularity would be eliminated again by the second block reduction.

Remedy: Create new/different NC program, e.g. with a smaller tolerance.

The surface normal vector $\mathbf{n}_F$ is not affected by the reduction of a block. This means that in contrast to the tool orientation, the change in orientation that may need to be executed for this vector will not be imaged onto the reduced traversing interval. This is necessary because a surface other than that programmed would be machined. Unlike the tool orientation, no problems arise as the result of an abrupt change in the surface normal vector at a block transition since it does not reflect any axis motions.

Analogously to 3D circumferential milling, (see Section "Behavior at inside corners (Page 940)"), the two traversing blocks that form an inside corner must contain contact points. There is no evaluation of several traversing blocks (i.e. no bottleneck detection), CDON/CDOF are not evaluated. If no contact point can be found, the machining operation is aborted with an alarm (risk of collision).

25.1.3.7 Monitoring of path curvature

The path curvature is only monitored for circumferential milling. As a rule, it is therefore not detected for CUT3DF* functions when an attempt is made to process a concave surface which is so intensively curved that machining is not possible with the tool being used. A possible exception are blocks that are split owing to a singularity. The transition between the two partial blocks created after the split is then treated like an inside corner. Apart from these special cases, it is the responsibility of the user to only use tools that make it possible to process every part of the contour without contour violation. It could be helpful to use the CUT3DFD function for this, where the differential tool is only minimally different from the original tool.

25.2 Commissioning

25.2.1 Parameter assignment

Minimal angle between path tangent and tool orientation

For the 3D-TRC, the angle between the path tangent and the tool orientation cannot undershoot a specific limit angle.

This angle is defined in the machine data:

MD21080 $MC_CUTCOM_PARALLEL_ORI_LIMIT

The lower the value of MD21080, the higher the computation effort generally is, which is needed to check adherence to the cited conditions. Exceptions apply to linear blocks with constant orientation.

Minimal angle between surface normal vector and tool orientation

In 3D face milling, the following machine data specifies the minimum angle which the surface normal vector and tool orientation must form in each point of the path if you are working with a side angle that is not equal to zero and the tool is not a cherry:

MD21082 $MC_CUTCOM_PLANE_ORI_LIMIT

If this value is undershot, processing is canceled with an alarm.

The lower the value of MD21082, the higher the computation effort generally is, which is needed to check adherence to the cited conditions.

MD21082 has no effect in linear blocks with constant orientation. In this case, angles of any size are permitted, even if the side angle is not equal to zero.

Minimal angle between surface normal vector and path tangent vector

During 3D face milling, the following machine data specifies the minimum angle which the surface normal vector and path tangent vector must form in each point of the path:

MD21084 $MC_CUTCOM_PLANE_PATH_LIMIT

If this value is undershot, processing is canceled with an alarm.

The lower the value of MD21084, the higher the computation effort generally is, which is needed to check adherence to the cited conditions.

Interpolation of the surface normals via polynomials

In order to use smoothing of the surface normal in 3D face milling, the function "Interpolation of the surface normal via polynomials" must be enabled:

MD28291 $MC_MM_SMOOTH_SURFACE_NORMALS = TRUE

If MD28291 is not set, processing is canceled with an alarm.

25.3 Programming

25.3.1 Selecting 3D tool radius compensation for 3D circumferential milling (CUT3DC, CUT3DCD, ISD)

The 3D tool radius compensation (3D TRC) for the 3D circumferential milling without taking limitation surfaces into account is selected with the modally effective G command CUT3DC or CUT3DCD.

The actual activation is performed with G41 or G42 The tool radius compensation is deactivated with G40.

Syntax

```
G41/G42 ORIC/ORID ISD=... CUT3DC/CUT3DCD CDOF2 X... Y... Z...
...
G40 X... Y... Z...
```

Meaning

`CUT3DC`:	3D TRC for circumferential milling (only when 5-axis transformation is active)	
`CUT3DCD`:	3D TRC for circumferential milling referred to a differential tool (only when 5-axis transformation is active) The radius difference is specified by the tool parameter $TC_DP15.	
`G41/G42 X... Y... Z...`:	Activate tool radius compensation	
	`G41`:	Tool radius compensation left of the contour
	`G42`:	Tool radius compensation right of the contour
	Note: The activation must be performed in a linear block (G0/G1).	
`CDOF2`:	Deactivate collision detection for 3D circumferential milling	
`ORIC/ORID`:	The behavior for orientation changes at outside corners is specified via the G commands `ORIC` and `ORID`.	
	`ORIC`:	Orientation changes at outside corners are superimposed on the circle block to be inserted.
	`ORID`:	Orientation changes at outside corners are executed before the circle block to be inserted.
`ISD=<Value>`:	With the `ISD` address, the insertion depth of the tool can be changed for circumferential milling and **active** 3D tool radius compensation.	
	`<Value>`:	Length of the insertion depth
`G40 X... Y... Z...`:	Deactivate tool radius compensation **Note:** The deactivation must be performed in a linear block (G0/G1) with geometry axis movements.	

Note

The G commands for selecting the 3D TRC are evaluated in the approach block, i.e. typically in the block that contains G41 or G42.

G41 or G42 can also be programmed in blocks without traversing movement in geometry axes relevant for the compensation. In this case, the approach block is the first traversing block following such a block.

A change of the 3D TRC variant with active tool radius compensation is ignored without alarm.

Example

Program code	Comment
	; Definition of tool D1:
`$TC_DP1[1,1]=120`	; Type (end mill)
`$TC_DP3[1,1]=20`	; Length compensation vector
`$TC_DP6[1,1]=8`	; Radius
`N10 X0 Y0 Z0 T1 D1 F12000`	; Selection of the tool.
`N20 TRAORI(1)`	; Activation of the transformation.
`N30 G42 ORIC ISD=10` **`CUT3DC`** `G64 X30`	; Activation of the 3D circumferential milling,
	; continuous orientation changes at outside corners,
	; insertion depth: 10 mm
`N40 ORIWKS A30 B15`	; Orientation change at a corner through specification of axis positions.
`N50 Y20 A3=1 C3=1`	; Traversing block with orientation change,
	; specification of the orientation with direction vector.
`N60 X50 Y30`	; Traversing block with constant orientation.
`N70 Y50 A3=0.5 B3=1 C3=5`	; Traversing block with orientation change.
`N80 M63`	; Block without traversing information.
`N90 X0 ISD=20`	; Traversing block with change of the insertion depth.
`N100 G40 Y0`	; Deactivation of the tool radius compensation.
`N110 M30`	

Further information

Path and orientation

The type of circumferential milling used here is implemented by defining a path (guide line) and the associated orientation. In this type of machining, the shape of the tool on the path is not relevant. Only the radius at the tool intervention point is decisive.

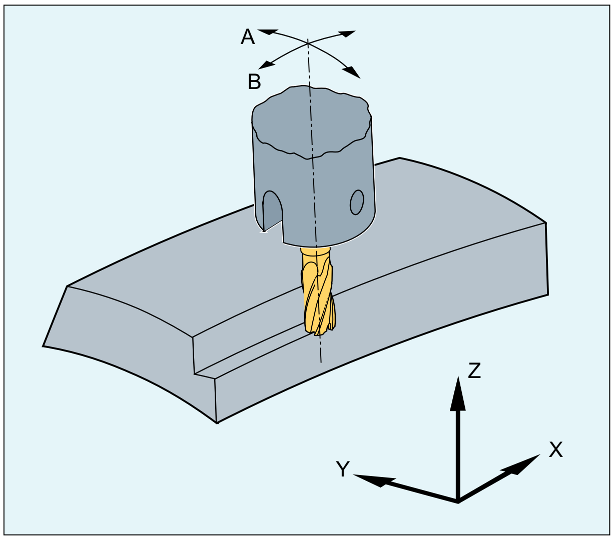


Figure 25-10 Circumferential milling

Approach behavior

The approach behavior is always NORM for the 3D variant of the tool radius compensation.

Behavior at outside corners

The G commands of group 18 (corner behavior, tool offset) are evaluated at the outside corners for circumferential milling with 3D TRC in the same way as for the conditions for the 2½D TRC:

- G450: Transition circle (tool travels round workpiece corners on a circular path)
  In contrast to the solution for the 2½D TRC, the inserted contour element at an outside corner is always a circle with a 0 radius, on which the tool radius compensation acts in the same way as on any other programmed path. It is not possible to insert conics instead of circles. In this case, the DISC address has no significance and is therefore not evaluated.
- G451: Intersection of equidistant paths (tool backs off from the workpiece corner)
  The intersection is determined by extending the offset curves of the two participating blocks and defining the intersection of the two blocks at the corner in the plane perpendicular to the tool orientation.

The intersection procedure (G451) is not used when at least one block containing a change to the tool orientation was inserted between the relevant traversing blocks. In this case, a circle is always inserted at the corner.

Behavior for changes in orientation at outside corners

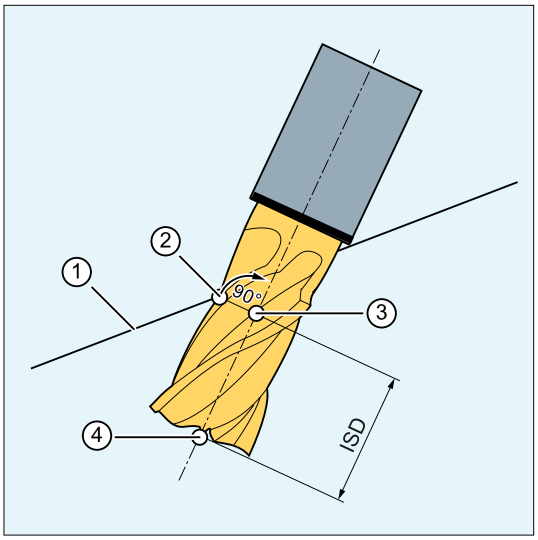
The ORIC and ORID G commands are used to determine whether changes in orientation programmed between two blocks forming the corner are executed before the inserted circle block (ORID) is processed or at the same time (ORIC).

Insertion depth

The insertion depth of the milling tool is the distance of the milling tool reference point from the tip of the tool.

The milling tool reference point is the vertical projection of the milling tool machining point on the programmed path to the longitudinal axis of the tool.

The position of the machining point on the peripheral surface of the tool is set with the insertion depth.



①	Programmed path
②	Milling tool machining point
③	Milling tool reference point
④	Milling tool tip
ISD	Insertion depth (InSertion Depth)

Figure 25-11 Insertion depth

Tool radius compensation referred to a differential tool

3D TRC for circumferential milling referred to a differential tool is activated via the CUT3DCD command. It should be applied if the programmed contour refers to the center-point path of a standard tool, and a tool other than a differential tool is used for machining. When calculating the 3D tool radius compensation, only the wear value of the radius of the active tool ($TC_DP15) and any programmed tool offsets OFFN and TOFFR are taken into account. The basic radius ($TC_DP6) of the active tool is **not** taken into account.

Pocket milling with inclined side walls for circumferential milling with CUT3DC

In this 3D tool radius compensation, a deviation of the mill radius is compensated by infeed toward the surface normals to be machined. The plane, in which the milling tool face is located, remains unchanged if the insertion depth ISD has remained the same. For example, a milling tool with a smaller radius than a standard tool would not reach the pocket base, which is also the limitation surface. For automatic tool infeed, this limitation surface must be known to the control, see Section "3D circumferential milling taking into account a limitation surface (CUT3DCC, CUT3DCCD) (Page 961)".

Advanced Surface / Top Surface

Note

When applying tool radius compensation CUT3DCD in combination with the "Advanced Surface" or "Top Surface" option (requiring a license), the setting recommendations regarding "Advanced Surface" / "Top Surface" must be observed.

Special test programs are provided in the SIOS portal for checking the set data:

- Test programs for Advanced Surface (https://support.industry.siemens.com/cs/ww/en/view/78956392)
- Test programs for Top Surface (https://support.industry.siemens.com/cs/ww/en/view/109738423)

25.3.2 Selecting 3D tool radius compensation for the 3D face milling (CUT3DF, CUT3DFS, CUT3DFF, CUT3DFD)

The 3D tool radius compensation (3D TRC) for the 3D face milling is selected with the modally effective G command CUT3DF, CUT3DFS, CUT3DFF or CUT3DFD.

The actual activation is performed with G41 or G42

With 3D face milling, the surface normals of the plane to be machined must be defined in order to calculate the tool radius compensation. This must be performed in the block with G41 or G42 via the A4, B4, C4 and A5, B5, C5 addresses.

The tool radius compensation is deactivated with G40.

Syntax

```
G41/G42 ORIC/ORID CUT3DF/CUT3DFS/CUT3DFF/CUT3DFD X... Y... Z... A4=... B4=...
C4=... A5=... B5=... C5=...
...
G40 X... Y... Z...
```

Meaning

`CUT3DFS`:	3D TRC for face milling with constant orientation. The tool orientation is defined by G17 - G19 and is not influenced by frames.
`CUT3DFF`:	3D TRC for face milling with constant orientation. The tool orientation is the direction defined by G17 - G19 and, in some case, rotated by a frame.
`CUT3DF`:	3D TRC for face milling with orientation change (only with active 5-axes transformation)

`CUT3DFD`:	3D TRC for face milling with orientation change referred to a differential tool (only with active 5-axis transformation) The radius difference is specified by the tool parameter $TC_DP15. **Note**: CUT3DFD is only possible in combination with "Smoothing of surface normals in 3D face milling". This is activated by calling the "Top Surface" function (requires as license) via CYCLE832(...).	
`G41/G42 X... Y... Z...`:	Activate tool offset The behavior with G41 and with G42 is identical for 3D face milling. **Note:** The activation must be performed in a linear block (G0/G1).	
`A4/5=... B4/5=... C4/5=...`:	Definition of the surface normals of the plane to be machined	
	`A4=... B4=... C4=...`:	Definition at start of block
	`A5=... B5=... C5=...`:	Definition at end of block
`ORIC/ORID`:	The behavior for orientation changes at outside corners is specified via the G commands `ORIC` and `ORID`.	
	`ORIC`:	Orientation changes at outside corners are superimposed on the circle block to be inserted.
	`ORID`:	Orientation changes are performed before the circle block.
`G40 X... Y... Z...`:	Deactivate tool radius compensation **Note:** The deactivation must be performed in a linear block (G0/G1) with geometry axis movements.	

Note

G41 or G42 can also be programmed in blocks without traversing movement in geometry axes relevant for the compensation. In this case, the approach block is the first traversing block following such a block.

A change of the 3D TRC variant with active tool radius compensation is ignored without alarm.

Examples

Example 1: 3D face milling with CUT3DF

Program code	**Comment**
`N10`	`; Definition of tool D1:`
`N20 $TC_DP1[1,1]=121`	`; Tool type (toroidal miller)`
`N30 $TC_DP3[1,1]=20`	`; Length compensation`
`N40 $TC_DP6[1,1]=5`	`; Radius`
`N50 $TC_DP7[1,1]=3`	`; Rounding radius`
`N60`	
`N70`	

Program code	Comment
N80 X0 Y0 Z0 A0 B0 C0 G17 T1 D1 F12000	; Selection of the tool.
N90 TRAORI(1)	; Select orientation transformation.
N100 B4=-1 C4=1	; Definition of the plane.
N110 **G41** ORID **CUT3DF** G64 X10 Y0 Z0	; Activate tool offset.
N120 X30	
N130 Y20 A4=1 C4=1	; Outside corner, new plane definition.
N140 B3=1 C3=5	; Change in orientation with ORID.
N150 B3=1 C3=1	; Change in orientation with ORID.
N160 X-10 A5=1 C5=2 ORIC	
N170 A3=-2 C3=1	; Change in orientation with ORIC.
N180 A3=-1 C3=1	; Change in orientation with ORIC.
N190 Y-10 A4=-1 C4=3	; New plane definition.
N200 X-20 Y-20 Z10	; Inside corner with previous block.
N210 X-30 Y10 A4=1 C4=1	; Inside corner, new plane definition.
N220 A3=1 B3=0.5 C3=1.7	; Change in orientation with ORIC.
N230 X-20 Y30 A4=1 B4=-2 C4=3 ORID	
N240 A3 = 0.5 B3=-0.5 C3=1	; Change in orientation.
N250 X0 Y30 C4=1	; Path movement, new plane,
	; orientation with relative programming.
N260 BSPLINE X20 Z15	; Spline begin, relative programming of the orien-tation
N270 X30 Y25 Z18	; remains active during spline.
N280 X40 Y20 Z13	
N290 X45 Y0 PW=2 Z8	
N300 Y-20	
N310 G2 ORIMKS A30 B45 I-20 X25 Y-40 Z0	; Helix, orientation with axis programming.
N320 G1 X0 A3=-0.123 B3=0.456 C3=2.789 B4=-1 C4=5 B5=-1 C5=2	; Path movement, orientation, non-constant plane.
N330 X-20 G40	; Deactivation of the tool radius compensation.
N340 M30	

Example 2: NC program (section) generated from a CAD system with CUT3DFD

Program code	Comment
N01 G710	
N03 T="12"	
N06 S5305 M03	
N07 G642	
; Approaching the starting position in the MCS taking the tool length into account.	
G00 G90 X-250.62787 Y-38.37944 A=DC(253.12719) B-12.49543	
G00 G90 Z251.80052	
; End of positioning in the MCS.	
;	

Program code	Comment
TRAORI(1)	; Select orientation transformation.
G500	
D1	
CYCLE832(0.01, _TOP_SURFACE_SMOOTH_ON + _ORI_FINISH, 1)	; Call CYCLE832 with: ; Contour tolerance = 0.01 mm, ; Processing type: Top Surface with smoothing, ; Finishing with input of an orientation tolerance, ; Orientation tolerance = 1 degree
CUT3DFD	
N08 G90 G94	
N09 G00 X-269.21195 Y128.32027 Z1.18577 A3=-.216361688 B3=.934284397 C3=-.283373051	; The blocks N09 to N10 are the fast portion of the approach movement to the workpiece with constant orientation.
N10 G00 X-251.90301 Y53.57752 Z23.85561	
N11 G01 X-247.57578 Y34.89183 Z29.52308 F50000.00000	; In the blocks N11 to N21, the slow portion of the approach movement is realized. The tool is already close to the workpiece; furthermore, the mold making settings (e.g. COMPSURF) from the CYCLE832 are now active (due to active G01). The path of this so-called transient phase for the mold-making behavior should be about 1000 times the contour tolerance (10 mm in this example).
N12 X-247.69126 Y33.82182 Z24.78219 F1061.00000	
N13 X-247.76560 Y33.13299 Z21.73022	
N14 X-247.82755 Y32.55897 Z19.18691	
N15 X-247.87918 Y32.08062 Z17.06748	
N16 X-247.92220 Y31.68200 Z15.30129	
N17 X-247.95805 Y31.34981 Z13.82947 A3=-.216361686 B3=.934284391 C3=-.283373071	
N18 X-247.98792 Y31.07299 Z12.60295 A3=-.216360662 B3=.934280801 C3=-.283385691	
N19 X-248.01282 Y30.84230 Z11.58085 A3=-.216336015 B3=.934194446 C3=-.283689030	
N20 X-248.03357 Y30.65006 Z10.72910 A3=-.216233089 B3=.933833626 C3=-.284952647	
N21 X-248.05086 Y30.48986 Z10.01931 A5=-.060687572 B5=.974940255 C5=-.214029243 A3=-.215712821 B3=.932005189 C3=-.291263295	
N22 **G41** X-248.06237 Y30.32400 Z9.36695 A5=-.060431854 B5=.973045457 C5=-.222554556 A3=-.214974689 B3=.929398552 C3=-.300007025 F1061.03295	; From N22, the surface normal is completely defined over the entire block for the first time (i.e. surface normal at the end of the previous block N21 is present, thus the surface normal at the beginning of the block N22 and the surface normal at the end of the block N22). Thus the prerequisite for switching on the tool offset with G41/G42 is fulfilled.

Program code	Comment

N23 X-248.07130 Y30.15119 Z8.71082
A5=-.060165696 B5=.971048883 C5=-.231179920
A3=-.214177198 B3=.926684940 C3=-.308841625	

N24 X-248.07829 Y29.97126 Z8.05094
A5=-.059884286 B5=.968941717 C5=-.239928784
A3=-.213318480 B3=.923853466 C3=-.317789237	

N25 X-248.08317 Y29.78487 Z7.38844
A5=-.059584206 B5=.966718449 C5=-.248807482
A3=-.212397895 B3=.920898045 C3=-.326854594	

N26 X-248.08578 Y29.59254 Z6.72679
A5=-.059263963 B5=.964380907 C5=-.257793037
A3=-.211418355 B3=.917822366 C3=-.336012474	

...	

Note

For 3D face milling with CUT3DFD, the definition of the surface normal is required for activating the tool offset with G41/G42. Programming of G41/G42 without definition of the surface normals leads to the output of an alarm.

Further information

3D face milling

For this type of 3D milling, you require the line-by-line description of the 3D paths on the workpiece surface. The tool shape and dimensions are taken into account in the calculations, which are normally performed in CAM. The postprocessor writes to the part program - in addition to NC blocks - the tool orientations (for active 5-axis transformation) and the G command for the required 3D tool offset. This means that the machine operator has the possibility of using tools that are slightly smaller - deviating from the tool used to calculate the NC paths.

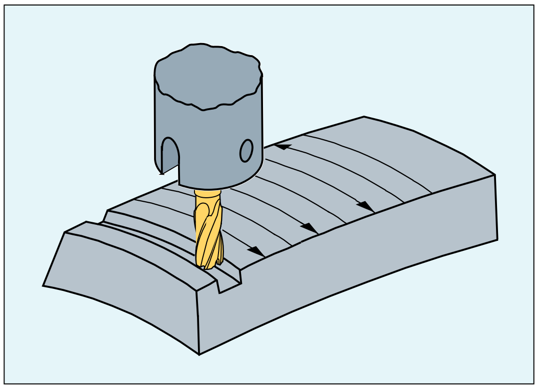


Figure 25-12 Face milling

Approach behavior

The approach behavior is always NORM for the 3D variant of the tool radius compensation.

Behavior at outside corners

Outside corners are treated as circles with radius 0 for face milling, whereby the circle plane extends from the end tangent of the first block to the start tangent of the second block. In this way, the orientation can be changed during block transition. A circle is therefore always inserted as contour element at an outside corner. The intersection procedure is not available with face milling.

Behavior for changes in orientation at outside corners

The ORIC and ORID G commands are used to determine whether changes in orientation programmed between two blocks forming the corner are executed before the inserted circle block (ORID) is processed or at the same time (ORIC).

Tool radius compensation referred to a differential tool

3D tool radius compensation referring to a differential tool is selected using the CUT3DFD command. It should be applied if the programmed contour refers to the center-point path of a standard tool, and a tool other than a differential tool is used for machining. When calculating the 3D tool radius compensation, only the wear value of the radius of the active tool ($TC_DP15) and any programmed tool offsets OFFN and TOFFR are taken into account. The basic radius ($TC_DP6) of the active tool is **not** taken into account.

3D face milling with CUT3DFD is only possible in combination with "Smoothing of surface normals in 3D face milling". This is activated by calling the "Top Surface" function (requires as license) via CYCLE832(...). Activation must take place **before** the tool offset is activated with G41/G42; not directly before tool intervention, but rather one path length before that, which corresponds to approx. 1000 times the contour tolerance (e.g. 1000 x 0.01 mm = 10 mm). Deactivation must be executed in reverse order: First switch off the tool offset with G40, then deactivate with e.g. CUT2D (or similar) after a path length which corresponds with approximately 1000 time the contour tolerance.

In order to be able to use the "Smoothing of surface normals in 3D face milling", the "Interpolation of surface normals via polynomials" function must also be enabled:

MD28291 $MC_MM_SMOOTH_SURFACE_NORMALS = TRUE

Note

For 3D face milling with CUT3DFD in combination with "Top Surface", the setting recommendations regarding "Top Surface" must be observed.

Special test programs are provided (https://support.industry.siemens.com/cs/ww/en/view/109738423) in the SIOS portal for checking the set data.

25.3.3 3D circumferential milling taking into account a limitation surface (CUT3DCC, CUT3DCCD)

In 3D circumferential milling with a continuous or constant change in tool orientation, the tool center-point path is frequently programmed for a defined standard tool. Because in practice suitable standard tools are often not available, a tool that does not deviate too much from a standard tool (≤ 5%) can be used.

CUT3DCCD takes account of a limitation surface for a real differential tool that the programmed standard tool would define. The NC program defines the center-point path of a standard tool.

CUT3DCC with the use of cylindrical tools takes account of a limitation surface that the programmed standard tool would have reached. The NC program defines the contour on the machining surface.

The surface normal vector of the limitation surface is specified with A4, B4, C4 and A5, B5, C5 for 3D face milling.

Syntax

```
G41/G42 CUT3DCCD/CUT3DCC CDOF2 X... Y... Z... A4=... B4=... C4=... A5=... B5=...
C5=...
...
G40 X... Y... Z...
```

Meaning

`CUT3DCCD`:	3D TRC for the circumferential milling taking into account a limitation surface with a differential tool on the tool center-point path: Infeed to the limitation surface	
`CUT3DCC`:	3D TRC for the circumferential milling taking into account a limitation surface with 3D radius compensation: Contour on the machining surface	
`G41/G42 X... Y... Z...`:	Activate tool radius compensation	
	`G41`:	Tool radius compensation left of the contour
	`G42`:	Tool radius compensation right of the contour
	Note: The activation must be performed in a linear block (G0/G1).	
`CDOF2`:	Deactivate collision detection for 3D circumferential milling	
`A4/5=... B4/5=... C4/5=...`:	Definition of the surface normals of the limitation surface	
	`A4=... B4=... C4=...`:	Definition at start of block
	`A5=... B5=... C5=...`:	Definition at end of block
`G40 X... Y... Z...`:	Deactivate tool radius compensation **Note:** The deactivation must be performed in a linear block (G0/G1) with geometry axis movements.	

Note

The G commands for selecting the 3D TRC are evaluated in the approach block, i.e. typically in the block that contains G41 or G42.

G41 or G42 can also be programmed in blocks without traversing movement in geometry axes relevant for the compensation. In this case, the approach block is the first traversing block following such a block.

A change of the 3D TRC variant with active tool radius compensation is ignored without alarm.

Example

Program code	Comment
`N10 $TC_DP1[1,1]=120`	`; Cylindrical milling tool`
`N20 $TC_DP6[1,1]=10`	
`N30 $TC_DP15[1,1]=-3`	
`...`	
`; Processing with cylindrical milling tool and CUT3DCCD`	
`N110 TRAORI`	`; Activation of the transformation.`
`N120 A4=0 B4=0 C4=1`	`; Definition of the surface normal of the limitation surface at the start of the block.`
`N130 X0 Y0 Z0 A0 C0 T1 D1 F20000`	
`N140 X10 Y0 Z0 G41` **`CUT3DCCD`** `CDOF2 G64`	`; Activate 3D circumferential milling, taking into account the limitation surface + switching off collision detection.`
`N150 X20`	
`N160 X30 A45`	`; Obtuse angle ==> no infeed`
`N170 X40 A-45`	`; Acute angle ==> infeed`
`N180 X55`	
`N190 Y10 Z10`	`; Movement in the tool direction.`
`N200 Y20`	
`N210 C45`	`; Pure change in orientation.`
`N220 Y30 C90`	
`N230 A5=-1 B5=0 C5=2 Y40`	`; Change of the surface.`
`N240 Y50 G40`	`; Deactivation of the tool radius compensation.`
`...`	

Further information

Tool type

The tool type (tool parameter $TC_DP1) is evaluated. Only milling tools with cylindrical shank (cylinder or end mill, toroidal miller and, in the limit case, cylindrical die mill) are permitted. This corresponds to the tool types 1 - 399, with the exception of the numbers 111 and 155 to 157.

Standard tools with corner rounding

Corner rounding with a standard tool is defined by the tool parameter $TC_DP7. Tool parameter $TC_DP16 describes the deviation of the corner rounding of the real tool compared with the standard tool.

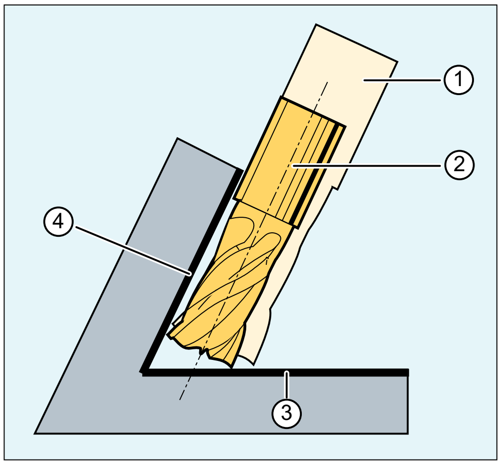
Example: Toroidal miller with reduced radius compared to the standard tool

Tool type	Shaft radius (R)	Corner radius (r)
Standard tool with corner rounding	R = $TC_DP6	r = $TC_DP7
Real tool with corner rounding Tool types 121 and 131 toroidal miller (end mill with corner rounding)	R' = $TC_DP6 + $TC_DP15 + OFFN	r' = $TC_DP7 + $TC_DP16

In this example, both $TC_DP15 + OFFN and $TC_DP16 are negative.

3D TRC with CUT3DCCD: Tool center-point path with infeed up to the limitation surface

If a tool with a smaller radius than the appropriate standard tool is used, machining is continued using a milling tool, which is fed in in the longitudinal direction until it reaches the bottom (base) of the pocket. The tool removes as much material from the corner formed by the machining surface and limitation surface. This involves a machining type combining circumferential and face milling. Analogous to a tool with reduced radius, for a tool with increased radius, the infeed is in the opposite direction.



① Standard tool
② Tool with smaller radius infeed up to the limitation surface
③ Limitation surface
④ Machining surface

Contrary to all other tool offsets of G group 22, tool parameter $TC_DP6 specified for CUT3DCCD has no relevance for the tool radius and does not influence the resulting compensation. The compensation offset results from the sum of the wear value of the tool radius (tool parameter $TC_DP15) and a tool offset OFFN programmed to calculate the perpendicular offset to the limitation surface.

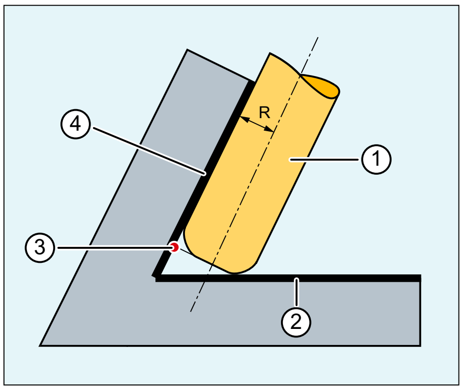
The generated part program does not specify whether the surface to be machined is to the right or left of the path. It is therefore assumed that the radius is a positive value and the wear value of the original tool is a negative value. A negative wear value always describes a tool with a reduced diameter.

Using cylindrical tools

When cylindrical tools are used, infeed is only necessary if the machining surface and the surface of limitation form an acute angle (less than 90 degrees). If a toroidal miller (end mill with corner rounding) is used, tool infeed in the longitudinal direction is required for both acute and obtuse angles.

3D TRC with CUT3DCC: Contour on the machining surface

If CUT3DCC is active with a toroidal miller, the programmed path refers to a fictitious cylindrical milling tool having the same diameter. The resulting path reference point is shown in the following diagram for a toroidal miller.



①	Toroidal miller
②	Limitation surface
③	Path reference point
④	Machining surface
R	Shaft radius (tool radius)

The angle between the machining and limitation surfaces may change from an acute to an obtuse angle and vice versa even within the same block.

The tool actually being used may either be larger or smaller than the standard tool. However, the resulting corner radius must not be negative and the sign of the resulting tool radius must be kept.

For CUT3DCC, the NC part program refers to the contour on the machining surface. As for conventional tool radius compensation, the total tool radius is used that comprises the following components:

- Tool radius (tool parameter $TC_DP6)
- Wear value (tool parameter $TC_DP15)
- A tool offset OFFN programmed to calculate the perpendicular offset to the limitation surface

The position of the limitation surface is defined from the following difference:

Dimensions of the standard tool - tool radius (tool parameter $TC_DP6)

Advanced Surface / Top Surface

Note

When applying tool radius compensation CUT3DCC / CUT3DCCD in combination with the "Advanced Surface" or "Top Surface" function (requiring a license), the setting recommendations regarding "Advanced Surface" / "Top Surface" must be observed.

Special test programs are provided in the SIOS portal for checking the set data:

- Test programs for Advanced Surface (https://support.industry.siemens.com/cs/ww/en/view/78956392)
- Test programs for Top Surface (https://support.industry.siemens.com/cs/ww/en/view/109738423)

25.4 Constraints

Boundary conditions for face milling

Functions with high-order geometry information

Functions with high-order geometry information (e.g. polynomials or spline interpolation) cannot be used.

Top Surface

3D face milling with CUT3DFD is only possible in conjunction with the "Top Surface" function, which requires a license.

Bottleneck detection

A "bottleneck detection" is not possible, and therefore the user is solely responsible.

Traversing movements in the orientation direction of the tool

For traversing motion in the orientation direction of the tool, and alarm is displayed and machining stopped.

25.5 Examples

25.5.1 Example 1: Orientation change at outside corner during 3D circumferential milling

Orientation change at outside corner with ORIC active

Program code	Comment
N10 A0 B0 X0 Y0 Z0 F5000	

Program code	Comment
`N20 T1 D1`	`; Selection of the tool (radius=5).`
`N30 TRAORI(1)`	`; Activation of transformation.`
`N40 CUT3DC`	`; Selection of 3D TRC for circumferential milling.`
`N50 ORIC`	`; Orientation changes at outside corners are superimposed on the circle block to be inserted.`
`N60 G42 X10 Y10`	`; Activation of TRC.`
`N70 X60`	
`N80 A3=1 B3=0 C3=1`	`; change in orientation of external corner generated from N70 and N90`
`N90 Y60`	
`N100 X10`	
`N110 G40 X0 Y0`	`; Deactivation of TRC.`
`N120 M30`	

The circular motion and change in orientation are executed in parallel in block N80 (ORIC active):

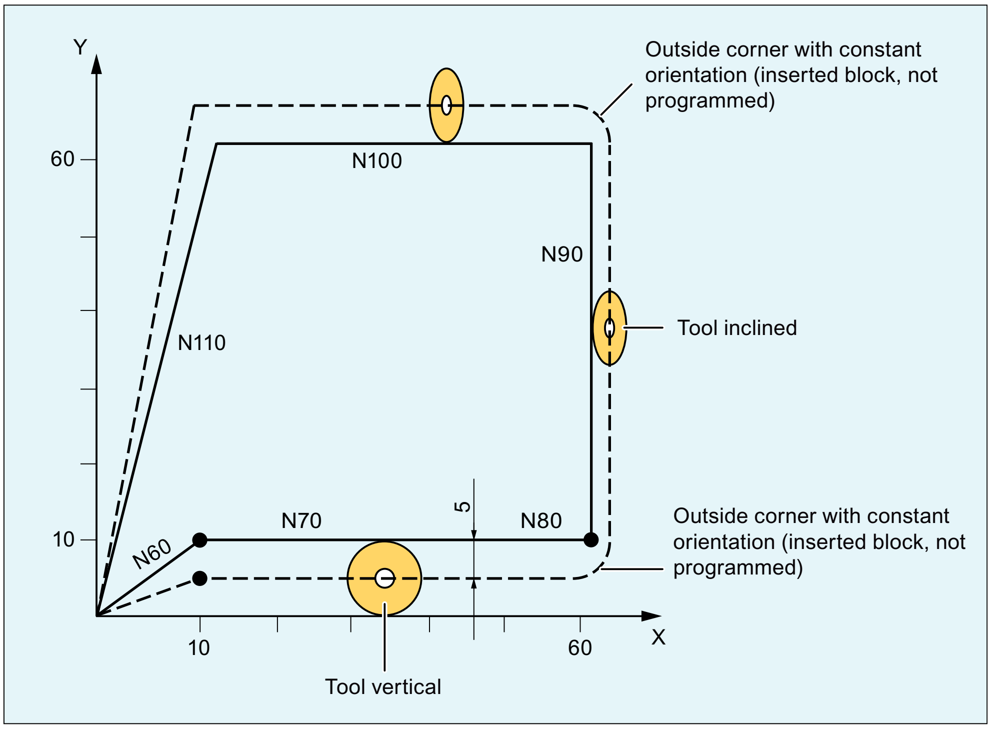

Y
60
10
X
10
60
N60
N70
N80
N90
N100
N110
5
Outside corner with constant orientation (inserted block, not programmed)
Tool inclined
Outside corner with constant orientation (inserted block, not programmed)
Tool vertical

Figure 25-13 ORIC: Change in orientation and path movement in parallel

Special case:

Intermediate blocks without traversing and orientation motions are executed at the programmed positions, e.g. auxiliary functions.

Example:

Program code	Comment
`N70 X60`	

Program code	Comment
N75 M20	; auxiliary function call
N80 A3=1 B3=0 C3=1	; change in orientation of external corner generated from N70 and N90
N90 Y60	

Blocks N75 and N80 are executed after N70. The circle block is then executed with the current orientation.

Orientation change at outside corner with ORID active

Program code	Comment
N10 A0 B0 X0 Y0 Z0 F5000	
N20 T1 D1	; Selection of the tool (radius=5).
N30 TRAORI(1)	; Activation of transformation.
N40 CUT3DC	; Selection of 3D TRC for circumferential milling.
N50 ORID	; Orientation changes at outside corners are performed on the circle block to be inserted.
N60 G42 X10 Y10	; Activation of TRC.
N70 X60	
N80 A3=1 B3=0 C3=1	; change in orientation of external corner generated from N70 and N90
N90 Y60	
N100 X10	
N110 G40 X0 Y0	; Deactivation of TRC.
N120 M30	

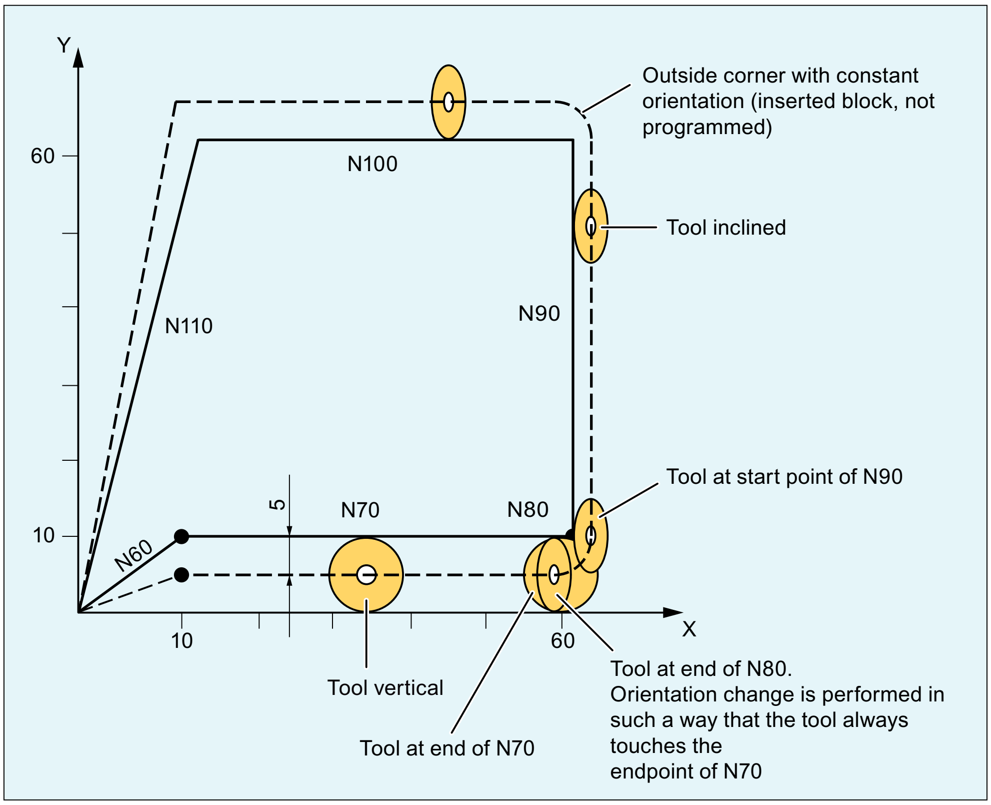
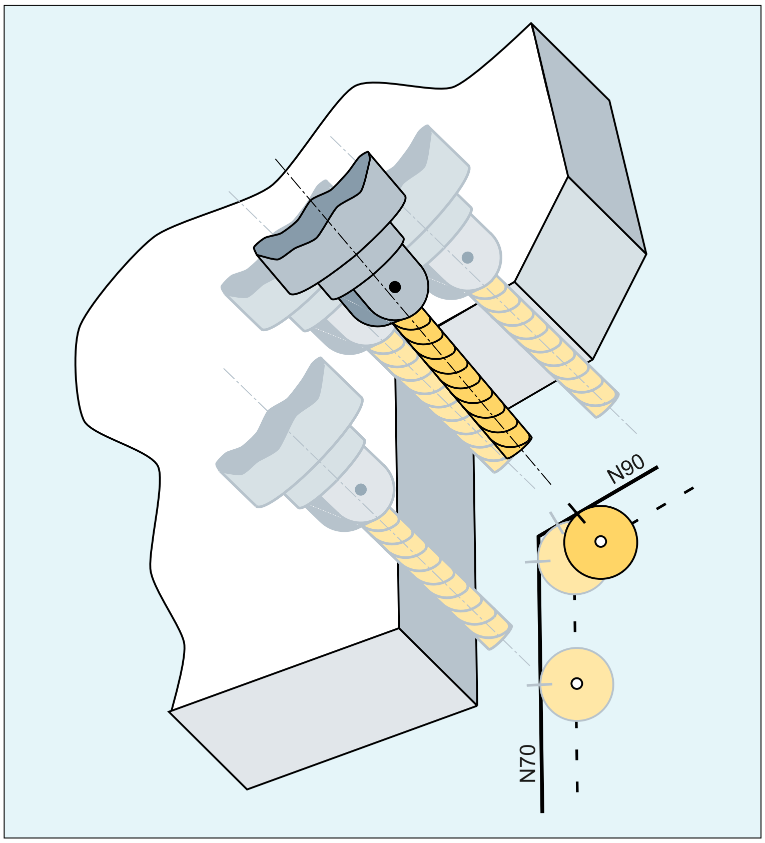

Y
60
10
X
10
60
5
N60
N70
N80
N90
N100
N110
Outside corner with constant orientation (inserted block, not programmed)
Tool inclined
Tool at start point of N90
Tool vertical
Tool at end of N70
Tool at end of N80.
Orientation change is performed in such a way that the tool always touches the endpoint of N70

Figure 25-14 ORID: Change in orientation and path movement consecutively

25.5.2 Example 2: Orientation change at inside corner during 3D circumferential milling

Program code	Comment
`N10 A0 B0 X0 Y0 Z0 F5000`	
`N20 T1 D1`	`; Selection of the tool (radius=5).`
`N30 TRAORI(1)`	`; Activation of transformation.`
`N40 CUT3DC`	`; Selection of 3D TRC for circumferential milling.`
`N50 ORID`	
`N60 G42 X10 Y10 G451`	`; Activation of TRC.`
`N70 Y60`	
`N80 A3=1 B3=0 C3=1`	`; change in orientation of inside corner generated from N70 and N90.`
`N90 X60 Y90`	
`N100 G40 X... Y...`	
`...`	
`N190 CDOF`	
`N200 M30`	



25.6 Data lists

25.6.1 General machine data

Number	Identifier: $MN_	Description
18094	MM_NUM_CC_TDA_PARAM	Number of TDA data
18096	MM_NUM_CC_TOA_PARAM	Number of TOA data, which can be set up per tool and evaluated by the CC
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool offsets per TOA module
18110	MM_NUM_TOA_MODULES	Number of TOA modules

25.6.2 Channelspecific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of the control basic setting after power-up and RESET / part program end
20120	TOOL_RESET_VALUE	Definition of the tool from which the tool length compensation is selected during run-up and upon RESET or part program end, dependent upon MD 20110.
20130	CUTTING_EDGE_RESET_VALUE	Definition of tool cutting edge for which tool length compensation is selected during power-up or on reset or parts program end as a function of MD20110
20140	TRAFO_RESET_VALUE	Definition of transformation block which is selected during power-up and or RESET or parts program end as a function of MD20110
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Behavior of the TRC at outside corners
20230	CUTCOM_CURVE_INSERT_LIMIT	Minimum value for intersection calculation with tool radius compensation
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Number of blocks without traversing motion for tool radius compensation
20270	CUTTING_EDGE_DEFAULT	Selected cutting edge after tool change
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid motions
21080	CUTCOM_PARALLEL_ORI_LIMIT	Minimal angle between path tangent and tool orientation for 3D-TRC
21082	CUTCOM_PLANE_ORI_LIMIT	Minimal angle between surface normal vector and tool orientation (for 3D face milling)
21084	CUTCOM_PLANE_PATH_LIMIT	Minimal angle between surface normal vector and path tangent vector (for 3D face milling)
22550	TOOL_CHANGE_MODE	New tool offset with M function
22560	TOOL_CHANGE_M_CODE	M function for tool change
28291	MM_SMOOTH_SURFACE_NORMALS	Interpolation of the surface normals via polynomials (for 3D face milling)

W6: Path length evaluation - 840D sl only 26

26.1 Brief description

Function

With the "Path length evaluation" function, the NC specific machine axis data is made available as the system and OPI variables, with whose help it is possible to assess the strain on the machine axes and thereby make an evaluation on the state of the machine's maintenance.

Recorded data

The following data is recorded:

- Total traverse path
- Total travel time
- Total travel count
- Total traverse path at high axis speeds
- Total travel time at high axis speeds
- Travel count at high axis speeds
- Total sum of jerks
- Axis travel time with jerk
- Travel count with jerk

Evaluation

When the function is activated, the selected control data is automatically sent and made available via the system and OPI variables for evaluation in the part program or synchronized actions, as with user-specified HMI functions.

Meaning

The data remains saved in the control, so that it may continue to be used after POWER OFF / ON. Consequently, the actual value of a data item represents the sum of all measured values since the function was activated.

26.2 Data

The following data is available:

System variable [1]	OPI variable	Meaning
$AA_TRAVEL_DIST	aaTravelDist	Total traverse path: sum of all set position changes in MCS [2] in [mm] or [deg.].
$AA_TRAVEL_TIME	aaTravelTime	Total travel time: sum of IPO clock cycles from set position changes in MCS [2] in [s] (solution: 1 IPO clock cycle)
$AA_TRAVEL_COUNT	aaTravelCount	Total travel count: a complete machine axis trip is defined by the following succession of states, as based on set position: standstill > traversing > standstill
$AA_TRAVEL_DIST_HS	aaTravelDistHS	Total traversing distance at high axis velocities [3]
$AA_TRAVEL_TIME_HS	aaTravelTimeHS	Total traversing time at high axis velocities [3]
$AA_TRAVEL_COUNT_HS	aaTravelCountHS	Total number of traversing operations at high axis velocities [3]
$AA_JERK_TOT	aaJerkTotal	Total sum of axis jerks: Sum of all jerk setpoints in [m/s^3] or [deg./ s^3].
$AA_JERK_TIME	aaJerkTime	Total travel time with jerk: sum of IPO clock cycles from jerk setpoint changes in [s] (solution: 1 IPO clock cycle)
$AA_JERK_COUNT	aaJerkCount	Total number of traversing operations with jerk
[1] System variables can be read from part programs and synchronized actions [2]) MCS: Machine coordinate system [3] Actual machine axis velocity ≥ 80% of the maximum parameterized axis velocity (MD32000 MAX_AX_VELO)		

26.3 Parameterization

26.3.1 General activation

The function is generally activated via the NC-specific machine data:

MD18860 $MN_MM_MAINTENANCE_MON (Activate recording of maintenance data)

26.3.2 Data groups

The data has been collected into data groups.

The data groups are activated via the axis-specific machine data:

MD33060 $MA_MAINTENANCE_DATA (configuration to record maintenance data)

Bit	Value	Activation of the following data: System variable / OPI variable
0	1	• Total traverse path: $AA_TRAVEL_DIST / aaTravelDist • Total travel time: $AA_TRAVEL_TIME / aaTravelTime • Total travel count: $AA_TRAVEL_COUNT / aaTravelCount
1	1	• Total traversing distance for high velocities: $AA_TRAVEL_DIST_HS / aaTravelDistHS • Total traversing time for high velocities: $AA_TRAVEL_TIME_HS / aaTravelTimeHS • Total number of traversing operations at high velocities: $AA_TRAVEL_COUNT_HS / aaTravelCountHS
2	1	• Total sum of jerks: $AA_JERK_TOT / aaJerkTotal • Travel time with jerk: $AA_JERK_TIME / aaJerkTime • Total travel count with jerk: $AA_JERK_COUNT / aaJerkCount

26.4 Examples

26.4.1 Traversal per part program

Three geometry axes AX1, AX2 and AX3 exist in a machine. For geometry axis AX1, the part program-driven total traverse path, total travel time and travel count should be calculated.

Parameter assignment

Activation of the overall function:

MD18860 $MN_MM_MAINTENANCE_MON = TRUE

Group activation: "Total travel distance, total travel time and number of travel operations" for geometry axis AX1:

MD33060 $MA_MAINTENANCE_DATA[AX1] = 1

Programming

To determine the values referred to the part program, the actual value of the system variables at the beginning of the part program must be saved in a calculation variable. The difference is acquired at the end of the part program.

Part program (extract):

```
Program code                                  Comment
...
                                            ; Current values:
```

Program code	**Comment**
`R10=$AA_TRAVEL_DIST[AX1]`	`; Total traversing distance AX1`
`R11=$AA_TRAVEL_TIME[AX1]`	`; Total traversing time AX1`
`R12=$AA_TRAVEL_COUNT[AX1]`	`; Number of traversing operations AX1`
`...`	`; Traversing motion of axes`
	`; Generating the difference:`
`R10=R10-$AA_TRAVEL_DIST[AX1]`	`; Traversing distance AX1 in the part program`
`R11=R11-$AA_TRAVEL_TIME[AX1]`	`; Traversing time AX1 in the part program`
`R12=R12-$AA_TRAVEL_COUNT[AX1]`	`; Number of traversing operations AX1 in the part program`

26.5 Data lists

26.5.1 Machine data

26.5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18860	MM_MAINTENANCE_MON	Activate recording of maintenance data

26.5.1.2 Axis/spindlespecific machine data

Number	Identifier: $MA_	Description
33060	MAINTENANCE_DATA	Configuration, recording maintenance data

Z3: NC/PLC interface signals 27

From Edition 05/2017, a detailed description of the NC/PLC interface signals is provided in the NC Variables and Interface Signals List Manual.

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

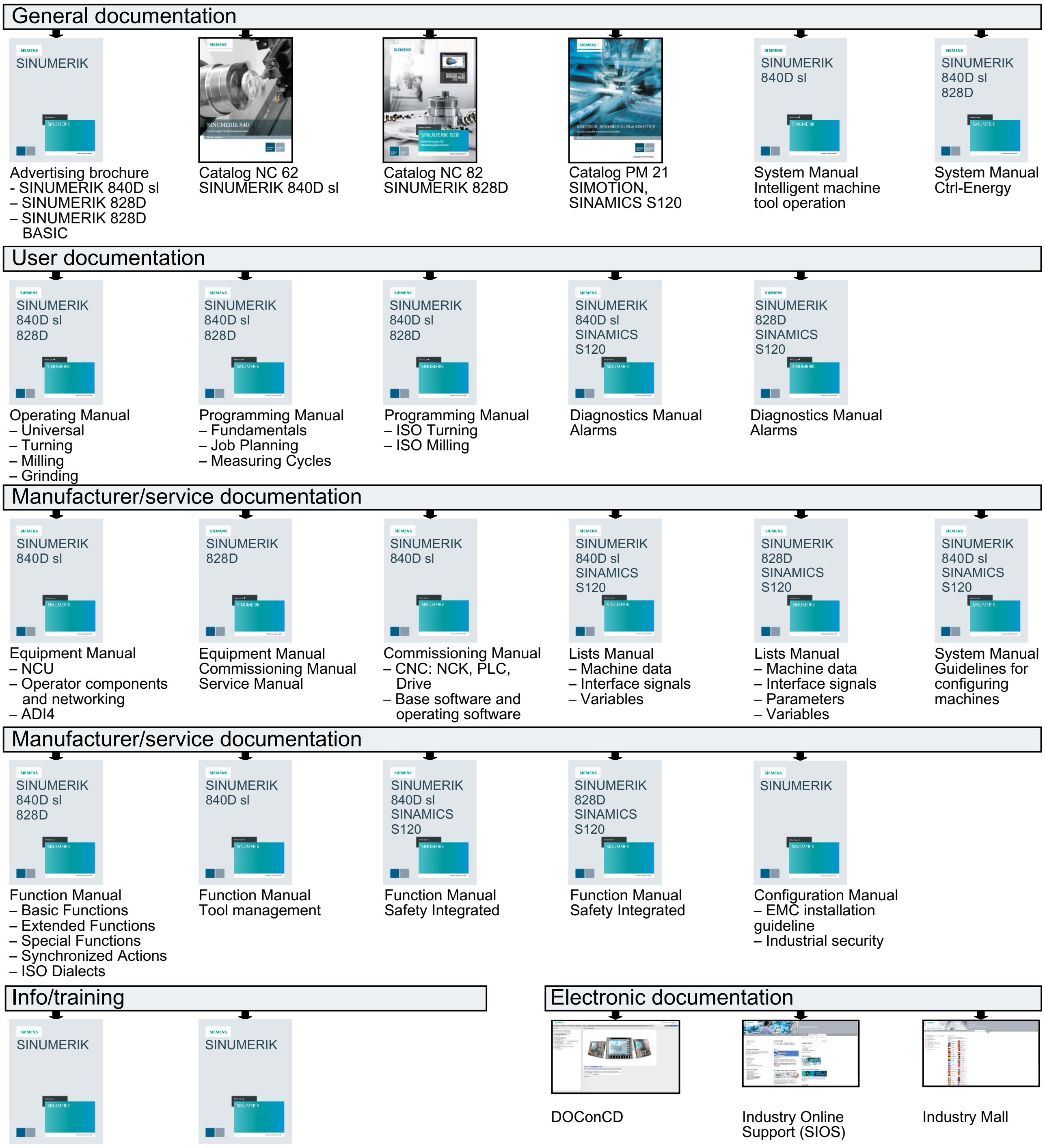
V	
VDI	Internal communication interface between NC and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]
VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VI	Voltage Input
VO	Voltage Output
FDD	Feed Drive

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)

A.2 Overview



General documentation

- Advertising brochure
  - SINUMERIK 840D sl
  - SINUMERIK 828D
  - SINUMERIK 828D BASIC
- Catalog NC 62 SINUMERIK 840D sl
- Catalog NC 82 SINUMERIK 828D
- Catalog PM 21 SIMOTION, SINAMICS S120
- System Manual Intelligent machine tool operation
- System Manual Ctrl-Energy

User documentation

- Operating Manual
  - Universal
  - Turning
  - Milling
  - Grinding
- Programming Manual
  - Fundamentals
  - Job Planning
  - Measuring Cycles
- Programming Manual
  - ISO Turning
  - ISO Milling
- Diagnostics Manual Alarms
- Diagnostics Manual Alarms

Manufacturer/service documentation

- Equipment Manual
  - NCU
  - Operator components and networking
  - ADI4
- Equipment Manual Commissioning Manual Service Manual
- Commissioning Manual
  - CNC: NCK, PLC, Drive
  - Base software and operating software
- Lists Manual
  - Machine data
  - Interface signals
  - Variables
- Lists Manual
  - Machine data
  - Interface signals
  - Parameters
  - Variables
- System Manual Guidelines for configuring machines

Manufacturer/service documentation

- Function Manual
  - Basic Functions
  - Extended Functions
  - Special Functions
  - Synchronized Actions
  - ISO Dialects
- Function Manual Tool management
- Function Manual Safety Integrated
- Function Manual Safety Integrated
- Configuration Manual
  - EMC installation guideline
  - Industrial security

Info/training

- Training documentation
  - Simple milling with ShopMill
  - Simple turning with ShopTurn
- Manual Tool and mold making

Electronic documentation

- DOConCD
- Industry Online Support (SIOS)
- Industry Mall

Glossary

Absolute dimensions

A destination for an axis motion is defined by a dimension that refers to the origin of the currently valid coordinate system. See → Incremental dimension

Acceleration with jerk limitation

In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a certain operand or operand range, e.g. input, output, etc.

Alarms

All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the deletion criterion. Alarms and messages are displayed separately.

1. Alarms and messages in the part program:
   Alarms and messages can be displayed in plain text directly from the part program.
2. Alarms and messages from the PLC:
   Alarms and messages for the machine can be displayed in plain text from the PLC program. No additional function block packages are required for this purpose.

Archiving

Reading out of files and/or directories on an **external** memory device.

Asynchronous subprogram

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

Automatic

Operating mode of the controller (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.

Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: Interpolating path axes
- Auxiliary axes: Non-interpolating feed and positioning axes with an axis-specific feedrate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis name

Axis name

To ensure clear identification, all channel and → machine axes of the control system must be designated with unique names in the channel and control system. The → geometry axes are called X, Y, Z. The rotary axes rotating around the geometry axes → are called A, B, C.

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Basic axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic Coordinate System

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference lies in the → axis names.

Baud rate

Rate of data transfer (bits/s).

Blank

Workpiece as it is before it is machined.

Block

"Block" is the term given to any files required for creating and processing programs.

Block search

For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting

Loading the system program after power ON.

C axis

Axis around which the tool spindle describes a controlled rotational and positioning motion.

C spline

The C spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feedrate, and the workpiece is thereby machined.

CNC

See → NC

Computerized Numerical Control: includes the components → NC, → PLC, HMI, → COM.

CNC

See → NC

Computerized Numerical Control: includes the components → NC, → PLC, HMI, → COM.

COM

Component of the NC for the implementation and coordination of communication.

Compensation axis

Axis with a setpoint or actual value modified by the compensation value

Compensation table

Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → PLC

CU

Transformation ratio

Curvature

The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r).

Cycles

Protected subprograms for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data blocks contain data definitions for global user data. This data can be initialized directly when it is defined.

Data word

Two-byte data unit within a → data block.

Diagnostics

1. Operating area of the control.
2. The control has a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and pitch values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (`G70`/`G71`), the control is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental work offset in Automatic mode in conjunction with an electronic handwheel.

Drive

The drive is the unit of the CNC that performs the speed and torque control based on the settings of the NC.

Dynamic feedforward control

Inaccuracies in the → contour due to following errors can be practically eliminated using dynamic, acceleration-dependent feedforward control. This results in excellent machining accuracy even at high → path velocities. Feedforward control can be selected and deselected on an axis-specific basis via the → part program.

Editor

The editor makes it possible to create, edit, extend, join, and import programs / texts / program blocks.

Exact stop

When an exact stop statement is programmed, the position specified in a block is approached exactly and, if necessary, very slowly. To reduce the approach time, → exact stop limits are defined for rapid traverse and feed.

Exact stop limit

When all path axes reach their exact stop limits, the control responds as if it had reached its precise destination point. A block advance of the → part program occurs.

External work offset

Work offset specified by the → PLC.

Fast retraction from the contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. An interrupt routine can also be executed following the fast retraction.

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1 to 200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the control. The control moves the relevant axes in → rapid traverse, whenever possible.

Frame

A frame is an arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system. A frame contains the following components: → work offset, → rotation, → scaling, → mirroring.

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

The geometry axes form the 2 or 3-dimensional → workpiece coordinate system in which, in → part programs, the geometry of the workpiece is programmed.

Ground

Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation

The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.

The helix comprises two motions:

- Circular motion in one plane
- A linear motion perpendicular to this plane

High-level CNC language

The high-level language is used to write NC programs, → synchronized actions, and → cycles. It provides: control structures → user-defined variables, → system variables, → macro programming.

High-speed digital inputs/outputs

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). High-speed, program-driven switching functions can be initiated via the digital CNC outputs

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

HW Config

SIMATIC S7 tool for the configuration and parameterization of hardware components within an S7 project

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subprograms, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

Inch measuring system

Measuring system which defines distances in inches and fractions of inches.

Inclined surface machining

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

Increment

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

Incremental dimension

Incremental dimension: A destination for axis traversal is defined by a distance to be covered and a direction referenced to a point already reached. See → Absolute dimension.

Intermediate blocks

Motions with selected → tool offset (`G41`/`G42`) may be interrupted by a limited number of intermediate blocks (blocks without axis motions in the offset plane), whereby the tool offset can still be correctly compensated for. The permissible number of intermediate blocks which the controller reads ahead can be set in system parameters.

Interpolator

Logic unit of the → NC that defines intermediate values for the motion to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Mechanical deviations of the machine are compensated for by means of interpolatory compensation functions, such as → leadscrew error, sag, angularity, and temperature compensation.

Interrupt routine

Interrupt routines are special → subprograms that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

The time required for the path of a block to be traversed can also be programmed for the axis motion instead of the feed velocity (G93).

JOG

Operating mode of the control (setup mode): The machine can be set up in JOG mode. Individual axes and spindles can be traversed in JOG mode by means of the direction keys. Additional functions in JOG mode include: → Reference point approach, → Repos, and → Preset (set actual value).

Key switch

The key switch on the → machine control panel has four positions that are assigned functions by the operating system of the controller. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KV

Servo gain factor, a control variable in a control loop.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The controller uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool travels along a straight line to the destination point while machining the workpiece.

Load memory

The load memory is the same as the → working memory for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Physically existent axes on the machine tool.

Machine control panel

An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system

A coordinate system, which is related to the axes of the machine tool.

Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block preceded with ":" that contains all information to start the operating sequence in a → part program.

Main program

The term "main program" has its origins during the time when part programs were split strictly into main and → subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program in the channel can be selected and started. It then runs through in → program level 0 (main program level). Further part programs or → cycles as subprograms can be called up in the main program.

MDI

Operating mode of the control: Manual Data Input. In the MDI mode, individual program blocks or block sequences with no reference to a main program or subprogram can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and → alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the deletion criterion. Alarms and messages are displayed separately.

Metric measuring system

Standardized system of units: For length, e.g. mm (millimeters), m (meters).

Mirroring

Mirroring reverses the signs of the coordinate values of a contour, with respect to an axis. It is possible to mirror with respect to more than one axis at a time.

Mode

An operating concept on a SINUMERIK control The following modes are defined: → Jog, → MDI, → Automatic.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a mode group can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

NC

Numerical Control component of the → CNC that executes the → part programs and coordinates the movements of the machine tool.

Network

A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK

Numeric robotic kernel (operating system of → NC)

NURBS

The motion control and path interpolation that occurs within the control is performed based on NURBS (**N**on **U**niform **R**ational **B**-**S**plines). This provides a uniform procedure for all internal interpolations.

OEM

The scope for implementing individual solutions (OEM applications) has been provided for machine manufacturers, who wish to create their own user interface or integrate technology-specific functions in the control.

Offset memory

Data range in the control, in which the tool offset data is stored.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Working memory
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable possibility of intervention that enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

Part program

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

Part program block

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

Part program management

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

Path axis

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

Path feedrate

Path feedrate affects → path axes. It represents the geometric sum of the feedrates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is a utility program for sending and receiving CNC user data (e.g. part programs, tool offsets) via the serial interface. The PCIN program can run under MS-DOS on standard industrial PCs.

Peripheral module

I/O modules represent the link between the CPU and the process.

I/O modules are:

- → Digital input/output modules
- → Analog input/output modules
- → Simulator modules

PLC

Programmable **L**ogic **C**ontroller: → Programmable logic controller. Component of → NC: Programmable control for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D sl: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the **STEP 7** software. The STEP 7 programming software is based on the **WINDOWS** standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D sl).

Positioning axis

Axis that performs an auxiliary motion on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subprograms of → part programs.

Program level

A part program started in the channel runs as a → main program on program level 0 (main program level). Any part program called up in the main program runs as a → subprogram on a program level 1 ... n of its own.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable logic controller

Programmable logic controllers (PLCs) are electronic controllers, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the controller. The programmable logic control has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Characters and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse velocity of an axis. It is used, for example, when the tool approaches the → workpiece contour from a resting position or when the tool is retracted from the workpiece contour. The rapid traverse velocity is set on a machine-specific basis using a machine data item.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rotary axis

Rotary axes apply a workpiece or tool rotation to a defined angular position.

Rotation

Component of a → frame that defines a rotation of the coordinate system around a particular angle.

Rounding axis

Rounding axes rotate a workpiece or tool to an angular position corresponding to an indexing grid. When a grid index is reached, the rounding axis is "in position".

RS-232-C

Serial interface for data input/output. Machining programs as well as manufacturer and user data can be loaded and saved via this interface.

Safety functions

The controller is equipped with permanently active monitoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Setting data

Data which communicates the properties of the machine tool to the NC as defined by the system software.

Softkey

A key, whose name appears on an area of the screen. The choice of softkeys displayed is dynamically adapted to the operating situation. The freely assignable function keys (softkeys) are assigned defined functions in the software.

Software limit switch

Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation

With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

Standard cycles

Standard cycles are provided for machining operations which are frequently repeated:

- For the drilling/milling technology
- For turning technology

The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g. positional data.

Subprogram

The term "subprogram" has its origins during the time when part programs were split strictly into →main and subprograms. This strict division no longer exists with today's SINUMERIK NC language. In principle, any part program or any → cycle can be called up as a subprogram within another part program. It then runs through in the next → program level (x+1) (subprogram level (x+1)).

Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output
   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.
2. Fast auxiliary function output
   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands timers, counters, markers

System variable

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character **$**. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the access of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the controller which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (`G41`/ `G42`).

Transformation

Additive or absolute zero offset of an axis.

Travel range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).

User interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

User memory

All programs and data, such as part programs, subprograms, comments, tool offsets, and work offsets / frames, as well as channel and program user data, can be stored in the shared CNC user memory.

User program

User programs for the S7-300 automation systems are created using the programming language STEP 7. The user program has a modular layout and consists of individual blocks.

The basic block types are:

- Code blocks
  These blocks contain the STEP 7 commands.
- Data blocks
  These blocks contain constants and variables for the STEP 7 program.

User-defined variable

Users can declare their own variables for any purpose in the → part program or data block (global user data). A definition contains a data type specification and the variable name. See → System variable.

Variable definition

A variable definition includes the specification of a data type and a variable name. The variable names can be used to access the value of the variables.

Velocity control

In order to achieve an acceptable traverse rate in the case of very slight motions per block, an anticipatory evaluation over several blocks (→ Look Ahead) can be specified.

WinSCP

WinSCP is a freely available open source program for Windows for the transfer of files.

Work offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable
   A configurable number of settable work offsets are available for each CNC axis. The offsets - which are selected by means of G commands - take effect alternatively.
2. External
   In addition to all the offsets which define the position of the workpiece zero, an external work offset can be overridden by means of the handwheel (DRF offset) or from the PLC.
3. Programmable
   Work offsets can be programmed for all path and positioning axes using the `TRANS` statement.

Working area

Three-dimensional zone into which the tool tip can be moved on account of the physical design of the machine tool. See → Protection zone.

Working area limitation

With the aid of the working area limitation, the traversing range of the axes can be further restricted in addition to the limit switches. One value pair per axis may be used to describe the protected working area.

Working memory

The working memory is a RAM in the → CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the → workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the → workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the → workpiece coordinate system. It is defined in terms of distances to the → machine zero.

Index