SIEMENS

SINUMERIK 840D sl

Special Functions

Function Manual

Valid for

Controller
SINUMERIK 840D sl / 840DE sl

Software *Version*
NCU system software for 840D sl/840DE sl 1.5/2.5

01/2008
6FC5397-2BP10-3BA0

Safety Guidelines

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

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

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

⚠CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

⚠WARNING
This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Automation and Drives
Postfach 48 48
90327 NÜRNBERG
GERMANY

Order number: 6FC5397-2BP10-3BA0
℗ 2/2008

Preface

SINUMERIK® Documentation

The SINUMERIK documentation is organized in three parts:

- General documentation
- User documentation
- Manufacturer/service documentation

A monthly updated publications overview with respective available languages can be found in the Internet under:

http://www.siemens.com/motioncontrol

Select the menu items "Support" → "Technical Documentation" → "Overview of Publications".

The Internet version of DOConCD (DOConWEB) is available under:

http://www.automation.siemens.com/doconweb

Information on the range of training courses and FAQs (frequently asked questions) are available on the Internet under:

http://www.siemens.com/motioncontrol under menu item "Support".

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.

Installation

Structure of this Function Manual:

- 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
  - Fax template for feedback for documentation
  - Overview
- Index of terms

Note

For detailed descriptions of data and alarm see:

- machine and setting data:
  Detailed description of machine data (only electronically on DOConCD or DOConWEB)
- NC/PLC interface signals:
  Function Manual Basic Functions; NC/PLC Interface Signals (Z1)
  Function Manual Basic Functions; NC/PLC Interface Signals (Z2)
  Function Manual Special Functions; NC/PLC Interface Signals (Z3)
- 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 data (signal name)	When the new gear step 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)

Data types

The following elementary data types are used in the control system:

Type	Meaning	Value range
INT	Signed integers	-2147483646 ... +2147483647
REAL	Figures with decimal point acc. to IEEE	±(2,2*10^{-308} … 1,8*10^{+308})
BOOL	Boolean values: TRUE/FALSE	TRUE ≠ 0; FALSE = 0
CHAR	1 ASCII character corresponding to the code	0 … 255
STRING	Character string, number of characters in [...], maximum of 200 characters	Sequence of values with 0 ... 255
AXIS	Axis identifier	Axis identifier for all channel axes
FRAME	Geometrical parameters for translation, rotation, scaling, and mirroring	
Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.		

Quantity structure

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

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

Technical Support

If you have any technical questions, please contact our hotline:

	Europe / Africa
Phone	+49 180 5050 222
Fax	+49 180 5050 223
Internet	http://www.siemens.com/automation/support-request

	America
Phone	+1 423 262 2522
Fax	+1 423 262 2200
E-Mail	mailto:techsupport.sea@siemens.com

	Asia/Pacific
Phone	+86 1064 719 990
Fax	+86 1064 747 474
E-mail	mailto:adsupport.asia@siemens.com

Note

Country specific telephone numbers for technical support are provided under the following Internet address:

http://www.siemens.com/automation/service&support

Calls from the German fixed line network are charged (e.g. at 0.14 €/min). Charges of other phone services may be different and may vary.

Questions about the manual

If you have any queries (suggestions, corrections) in relation to this documentation, please send a fax or e-mail to the following address:

Fax: +49 9131 - 98 63315
Email: mailto:docu.motioncontrol@siemens.com

A fax form is available in the appendix of this document.

SINUMERIK Internet address

http://www.siemens.com/sinumerik

Contents

3-Axis to 5-Axis Transformation (F2)

1

1.1 Brief description

Note

The transformations described below require that individual names are assigned to machine axes, channels and geometry axes when the transformation is active. Compare macchine data:

MD10000 $MN_AXCONF_MACHAX_NAME_TAB (machine axis name)

MD20080 $MC_AXCONF_CHANAX_NAME_TAB (name of the channel axis in the channel)

MD20060 $MC_AXCONF_GEOAX_NAME_TAB (name of the geometry axis in the channel)

Besides this no unambiguous assignments are present.

1.1.1 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.

1.1.2 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.

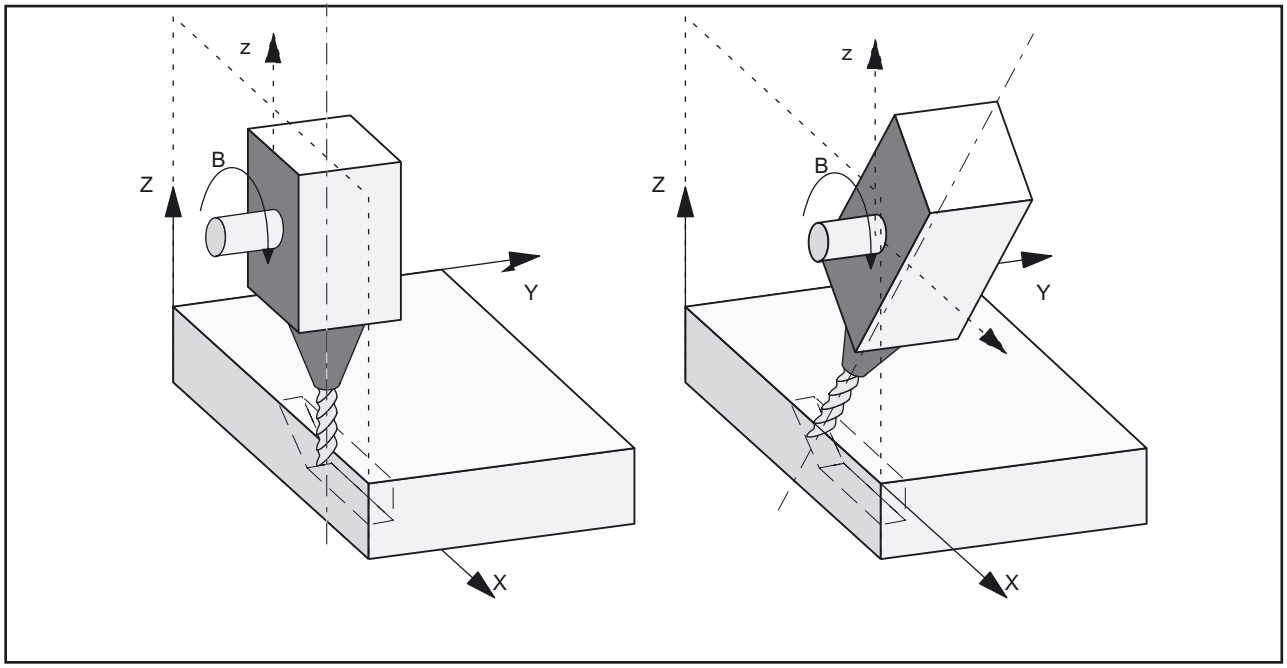


Figure 1-1 Schematic diagram of 3-axis transformation

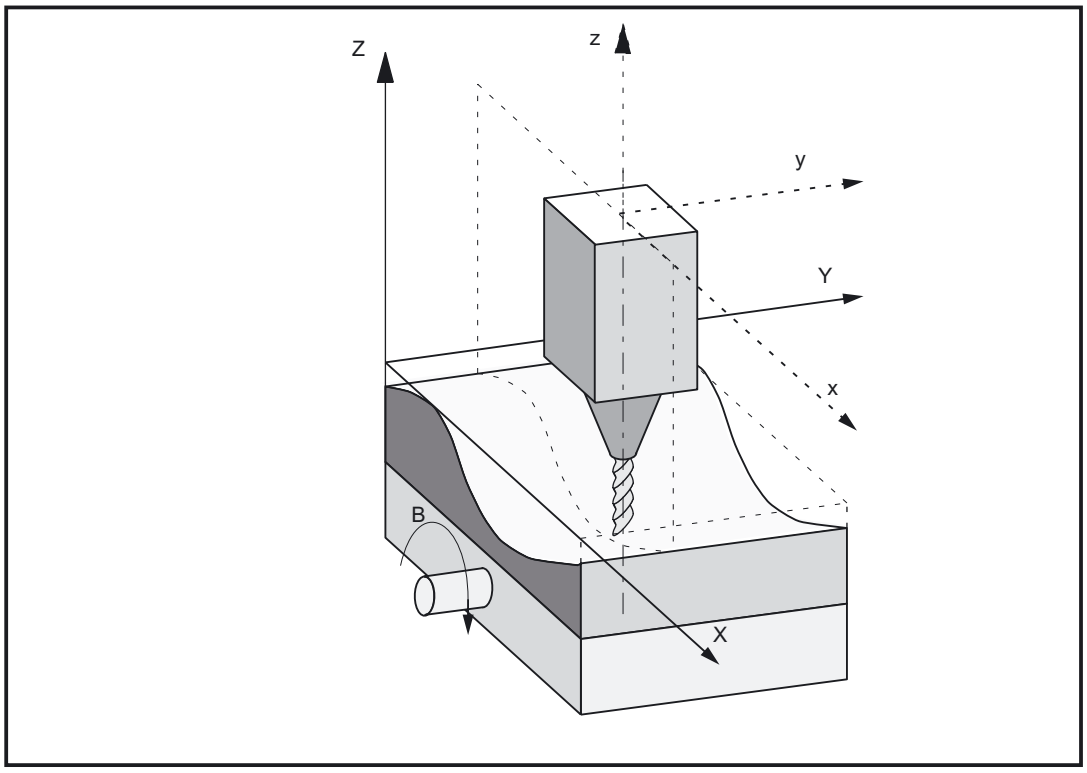


Figure 1-2 Schematic diagram of a 4-axis transformation with moveable workpiece

1.1.3 Orientation transformation with a swivelling linear axis.

Function

The orientation transformation with swivelling 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.

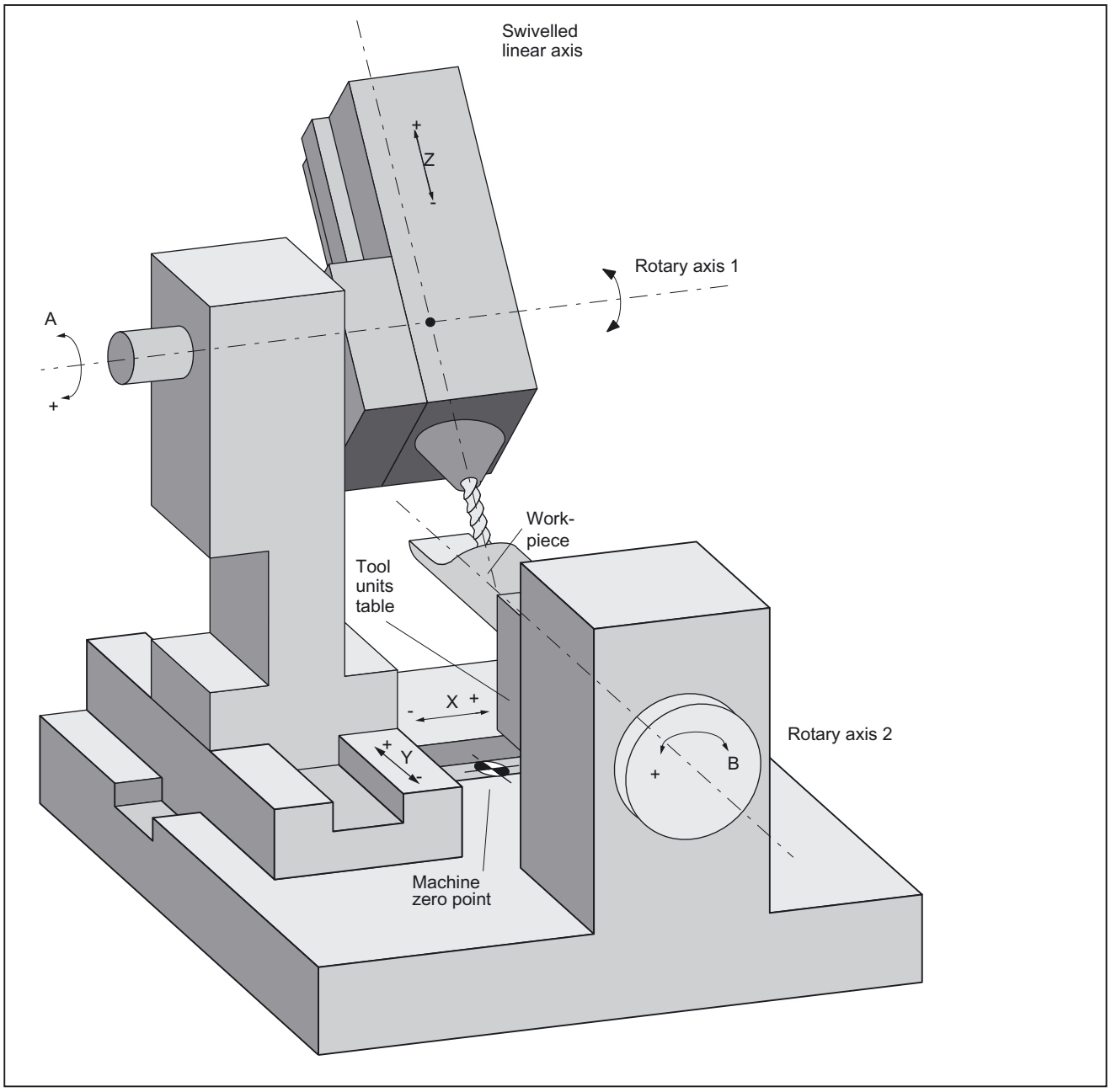


Figure 1-3 Schematic diagram of a machine with swivelling linear axis

1.1.4 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)

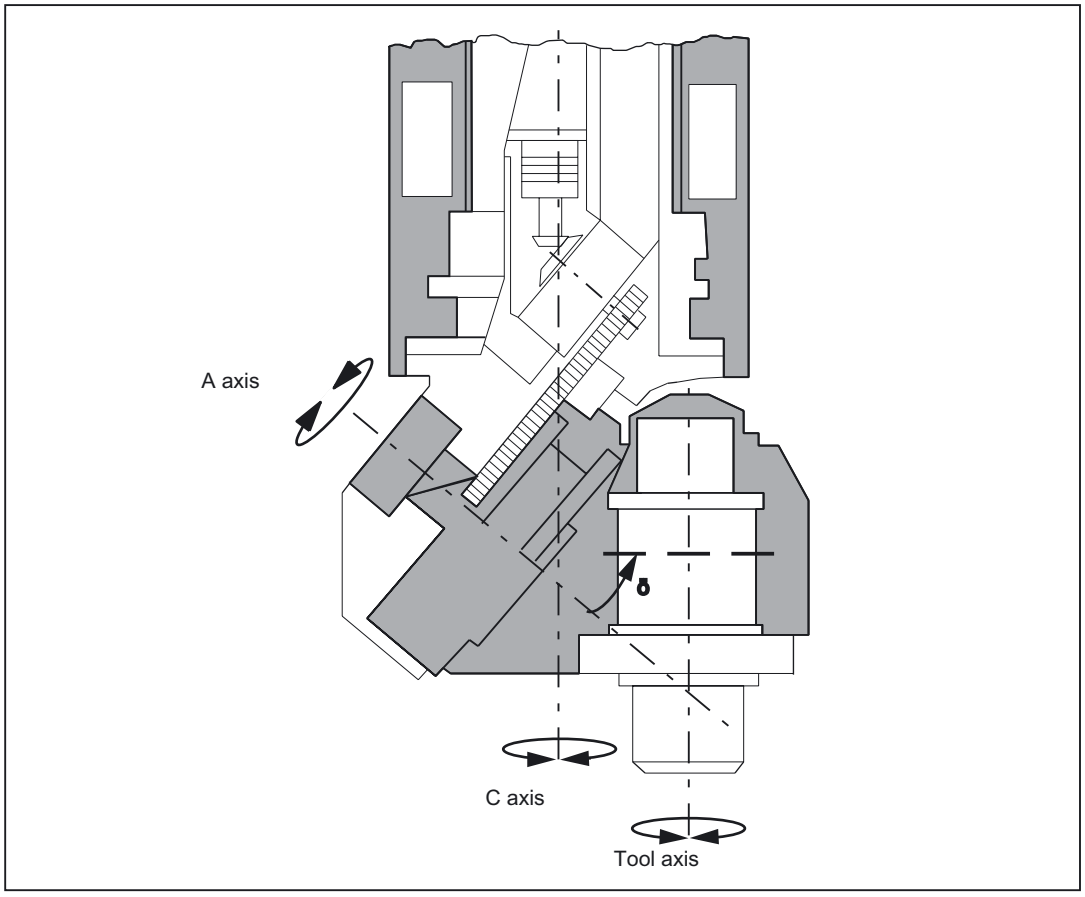


Figure 1-4 Schematic diagram of a machine tool with universal milling head

1.1.5 Orientation axes

Model for describing change in orientation

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

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

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

Real-time transformation

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

These Cartesian coordinates comprise:

- **Geometry axes**

  Geometry axes describe the machining point.

- **Orientation axes**

  Orientation axes describe the orientation of a tool in space.

Tool orientation

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

- Direct programming of rotary axis positions A, B, C

  5-axis transformation by programming:

  – The Euler- or RPY angle in degrees through `A2,B2,C2`

    or

  – The direction vector over `A3,B3,C3`

- Programming using lead angle `LEAD` and tilt angle `TILT`

1.1.6 Cartesian manual travel

Function

The "Cartesian Manual Operation" function can be used to set one of the following coordinate systems as reference system for JOG motion to be selected separately for translation and orientation as:

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

1.1.7 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.

1.1.8 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.

1.1.9 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.

1.1.10 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 parts program/softkey and it is not necessary to boot the control system.

References:
/FB2/ Function Manual, Extended Functions; Kinematic Transformations (M1),
Chapter: Cartesian PTP travel

1.1.11 Orientation compression

During the execution of NC programs containing blocks with relatively short traverse paths, the interpolation time 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.

1.2 5-axis transformation

1.2.1 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.

1.2.2 Machine types for 5-axis transformation

Kinematics of machines for 5-axis transformation

5-axis machines are generally equipped with three linear and two rotary axes: the latter may be implemented as a two-axis swivel head, a two-axis rotary table or as a combination of single-axis rotary table and swivel head. These types of machine are characterized by:

1. Three linear axes form a right-handed, Cartesian coordinate system.
2. Rotary axes are parallel to the traversing direction of one of the linear axes.

   Example:

   – A parallel to X
   – B parallel to Y
   – C parallel to Z
3. Rotary axes are positioned vertically one above the other.

4. Rotary axes turn
   - Tool with two-axis swivel head (machine type 1)
   - Workpiece with two-axis rotary table (machine type 2)
   - Tool and workpiece with single-axis rotary table and swivel head (machine type 3)
5. The following applies to machine types 1 and 2:
   - Rotary axis 1 is treated as the 4th machine axis of the transformation.
   - Motion of 1st rotary axis changes the orientation of the 2nd rotary axis.
   - Rotary axis 2 is treated as the 5th machine axis of the transformation.
   - Motion of 2nd rotary axis does not change the orientation of the 1st rotary axis.
6. The following applies to machine type 3:
   - 1. Rotary axis (4th machine axis of transformation) turns the tool.
   - 2. Rotary axis (5th machine axis of transformation) turns the tool.
7. Initial tool position:
   - in negative Z direction.

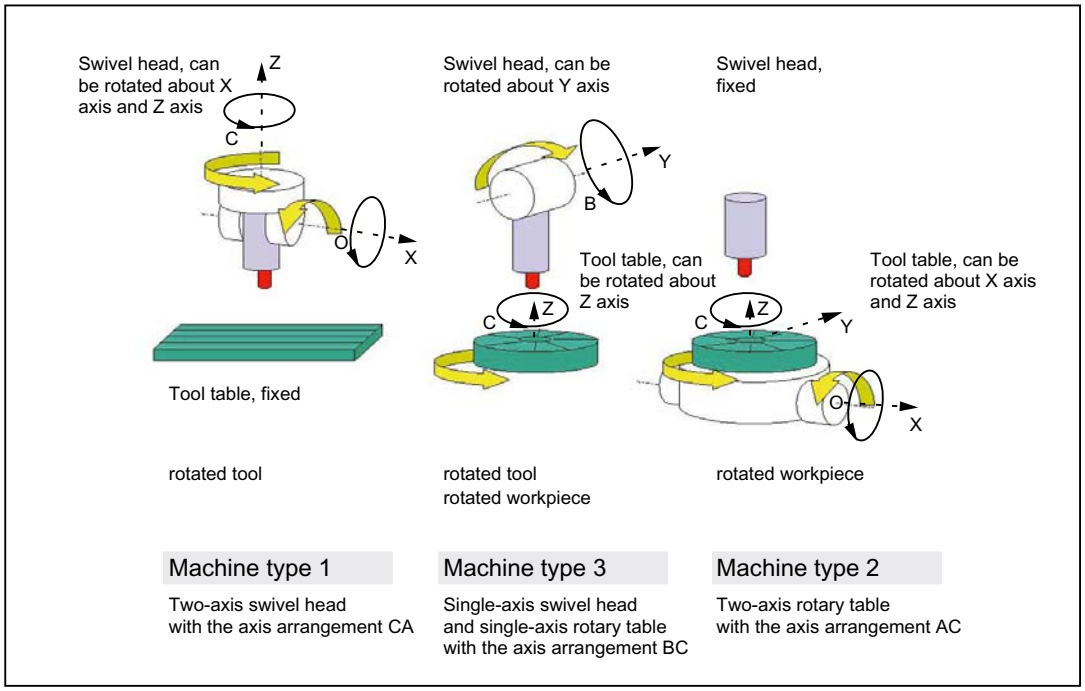


Figure 1-5 Machine types for 5-axis transformation

Note

Transformations that do not fulfill all the conditions mentioned here (3 and 4-axis transformations, orientation transformation with swivelling linear axes, universal milling head) are described in separate sub chapters.

1.2.3 Configuration of a machine for 5-axis transformation

To ensure that the 5-axis transformation can convert the programmed values to axis motions, certain information about the mechanical design of the machine is required; this information is stored in machine data:

- Machine type
- Axis assignment
- Geometry information
- Assignment of direction of rotation

Machine type

The machine types have been designated above as types 1 to 3 and are stored in the following machine data as a two-digit number:

MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

...

MD24480 $MC_TRAFO_TYPE_10 (definition of channel transformation 10)

The following table contains a list of machine types, which are suitable for 5-axis transformation.

Axis sequence	Machine type 1 with swivelling / rotary tool	Machine type 2 with swivelling / rotary workpiece	Machine type 3 with swivelling / rotary tool/workpiece
AB	16	32	48
AC	x	33	49
BA	18	34	50
BC	x	35	51
CA	20	x	x
CB	21	x	x

Combinations that are not meaningful, whose C-axis corresponds to a rotation of the tool about its longitudinal axis (symmetry axis), are marked by x.

Identification of axis sequence

The axis sequence is identified in the following way:

- AB means: A is 4th axis, B is 5th axis of 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

The axis assignment at the start of the 5-axis transformation defines the axis that will be mapped by the transformation internally onto a channel axis. Thus, the following is defined in the machine data below:

MD24110 $MC_TRAFO_AXES_IN_1 (Axis assignment for transformation 1)

...

MD24482 $MC_TRAFO_AXES_IN_10 (Axis assignment for transformation 10)

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):

MD24500 $MC_TRAFO5_PART_OFFSET_1 (workpiece-oriented offset)

- for machine type 1 (two-axis swivel head)

  Vector from machine reference point to table zero point (zero vector)

- for machine type 2 (two-axis rotary table)

  Vector from last table swivel joint to zero point of table

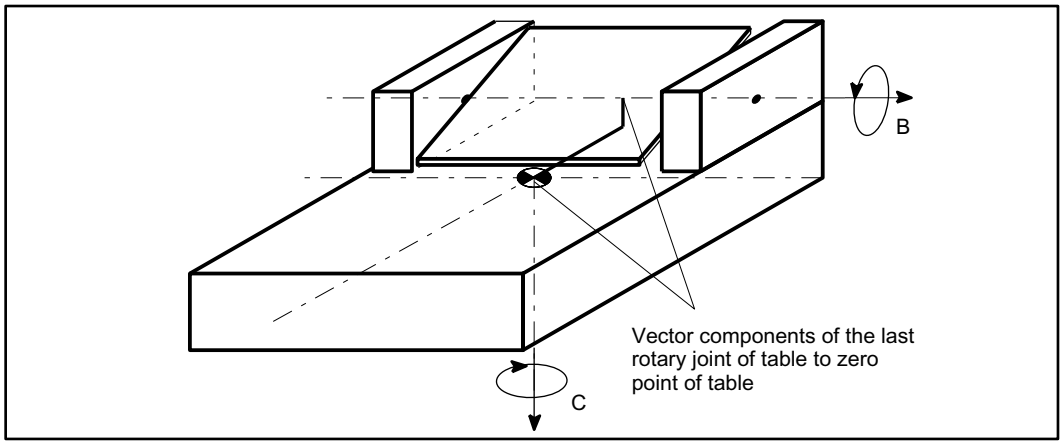


Figure 1-6 Machine data MD24500 $MC_TRAFO5_PART_OFFSET_1 for machine type 2

- for machine type 3 (single-axis swivel head and single-axis rotary table)

  Vector from joint of table to zero point of table

MD24560 $MC_TRAFO5_JOINT_OFFSET_1 (vector of the kinematic offset of 5-axis transformation 1)

Vector from the first to the second swivel joint (machine type 1 and 2)

Vector from machine zero point to the swivel joint of the table (machine type 3)

MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1 (position offset of rotary axes 1/2/3 5-axis transformation 1) angle offset of the first or second rotary axis

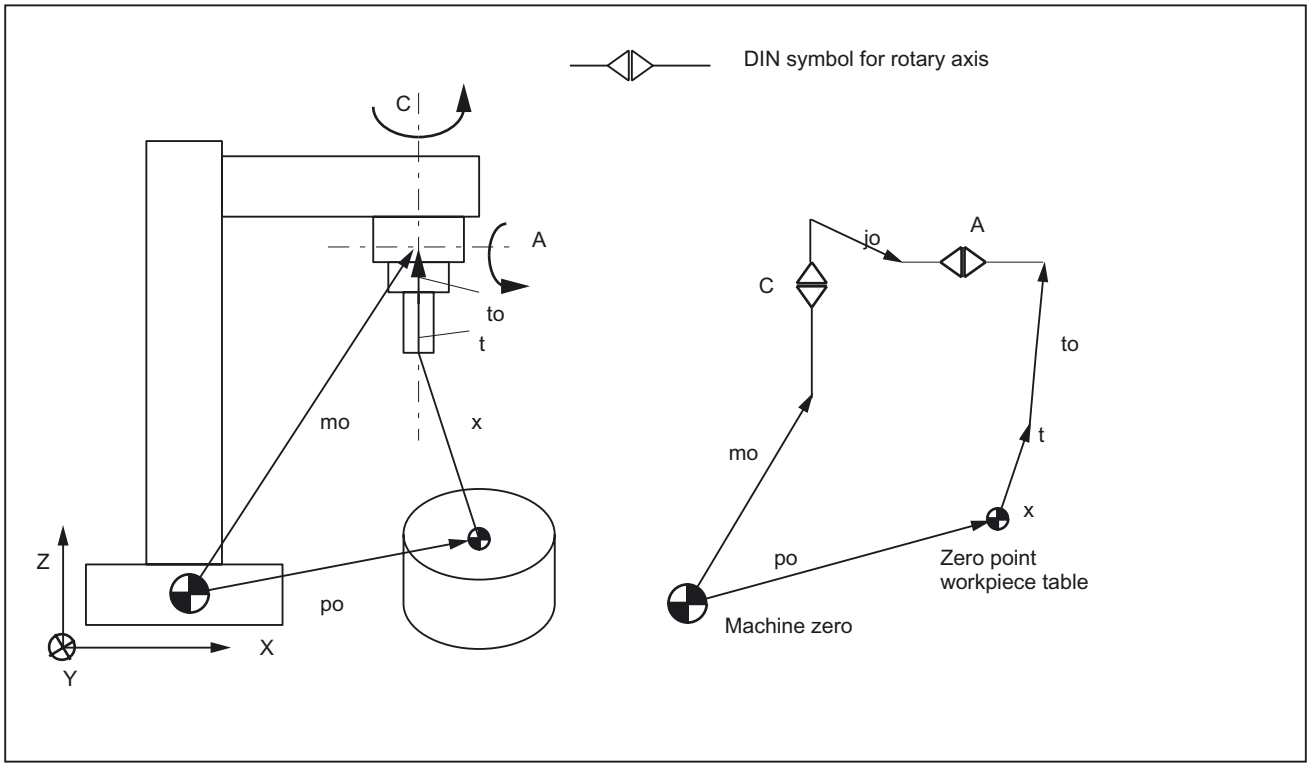


Figure 1-7 Schematic diagram of CA kinematics, moved tool

mo	Position vector in MCS
po	$MC_TRAFO5_PART_OFFSET_n[0 ..2]
x	Vector of programmed position in BCS
t	Tool correction vector
to	$MC_TRAFO5_BASE_TOOL_n[0 .. 2]
jo	$MC_TRAFO5_JOINT_OFFSET_n[0 .. 2]

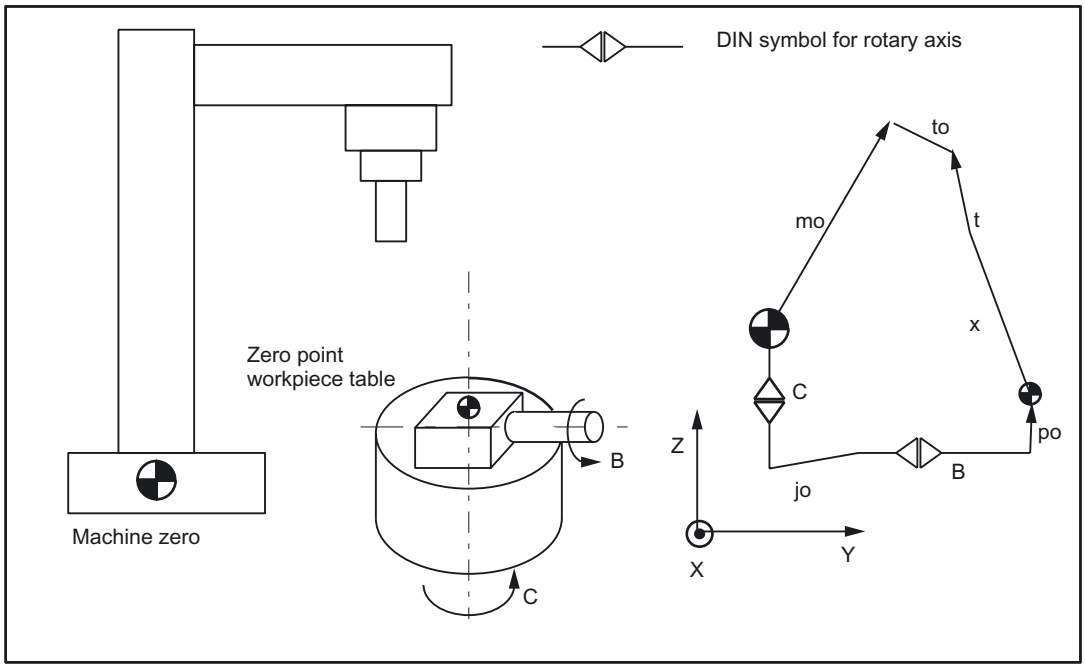


Figure 1-8 Schematic diagram of CB kinematics, moved workpiece

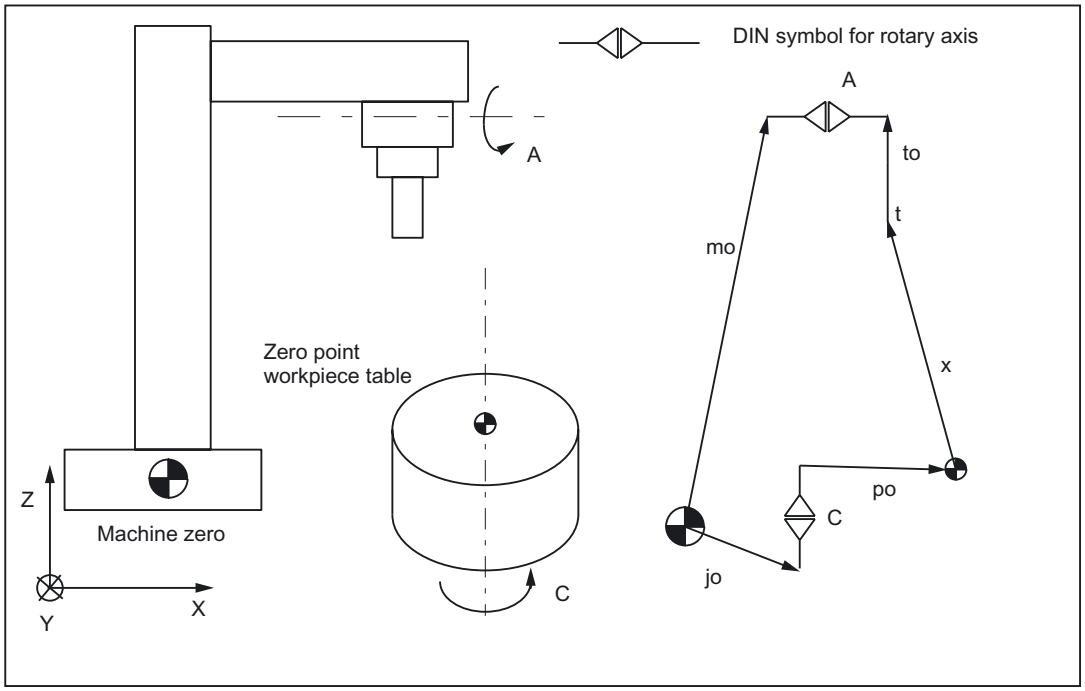


Figure 1-9 Schematic diagram of AC kinematics, moved tool, moved workpiece

Assignment of direction of rotation

The sign interpretation setting for a rotary axis is stored in the sign machine data for 5-axis transformation.

MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1[n] (sign of rotary axis 1/2/3 for 5-axis transformation 1)

MD24620 $MC_TRAFO5_ROT_SIGN_IS_PLUS_2[n] (sign of rotary axis 1/2/3 for 5-axis transformation 2)

Transformation types

Ten transformation types per channel can be configured in the following machine data:

MD24100 $MC_TRAFO_TYPE_1 ...MD24480 $MC_TRAFO_TYPE_10 (definition of transformation 1 in channel … definition of transformation 10 in channel)

Of these eight types, a maximum of two may be 5-axis transformations.

Activation

Activation of the 5-axis transformation is described in the section "Activation and Application of 3- to 5-axis Transformation".

1.2.4 Tool orientation

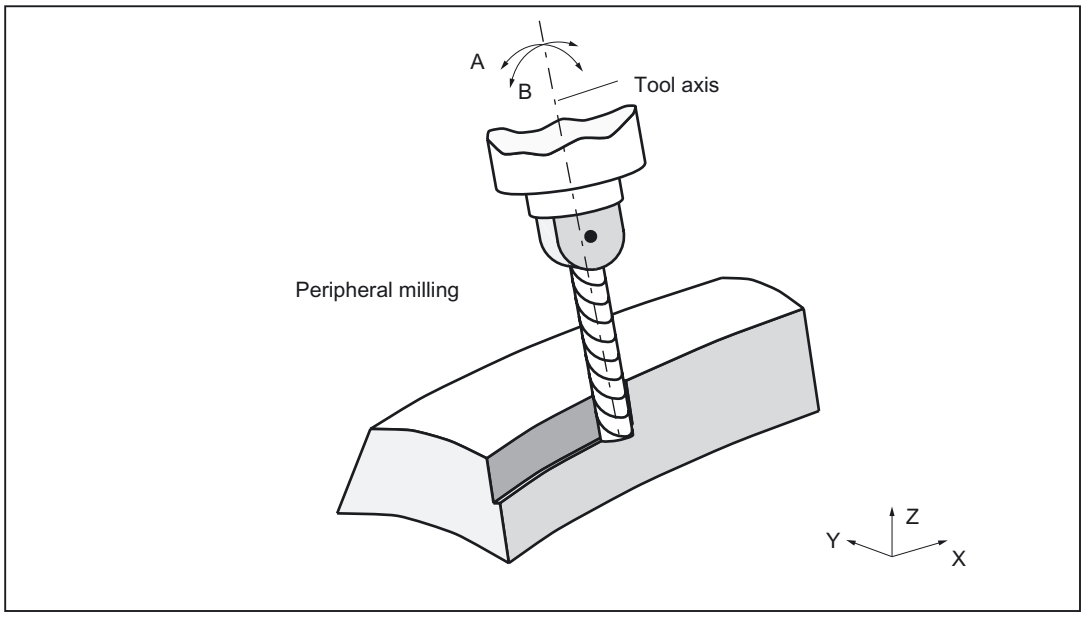


Figure 1-10 Machining of workpieces with 5-axis transformation

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)

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 ORIWCS
- in the machine coordinate system with command ORIMCS

ORIWCS 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.

ORIMCS 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 ORIWCS and ORIMCS.

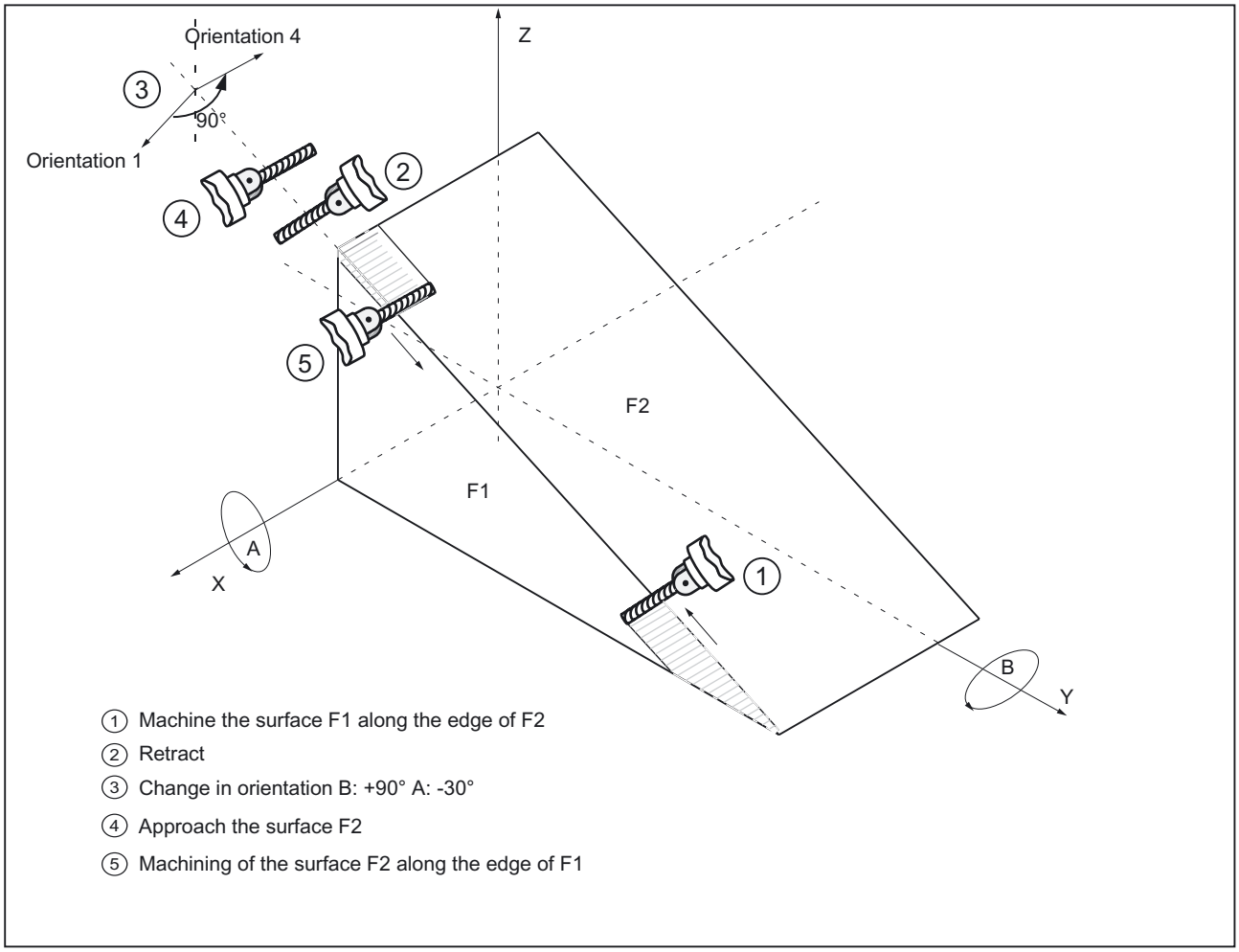


Figure 1-11 Change in cutter orientation while machining inclined edges

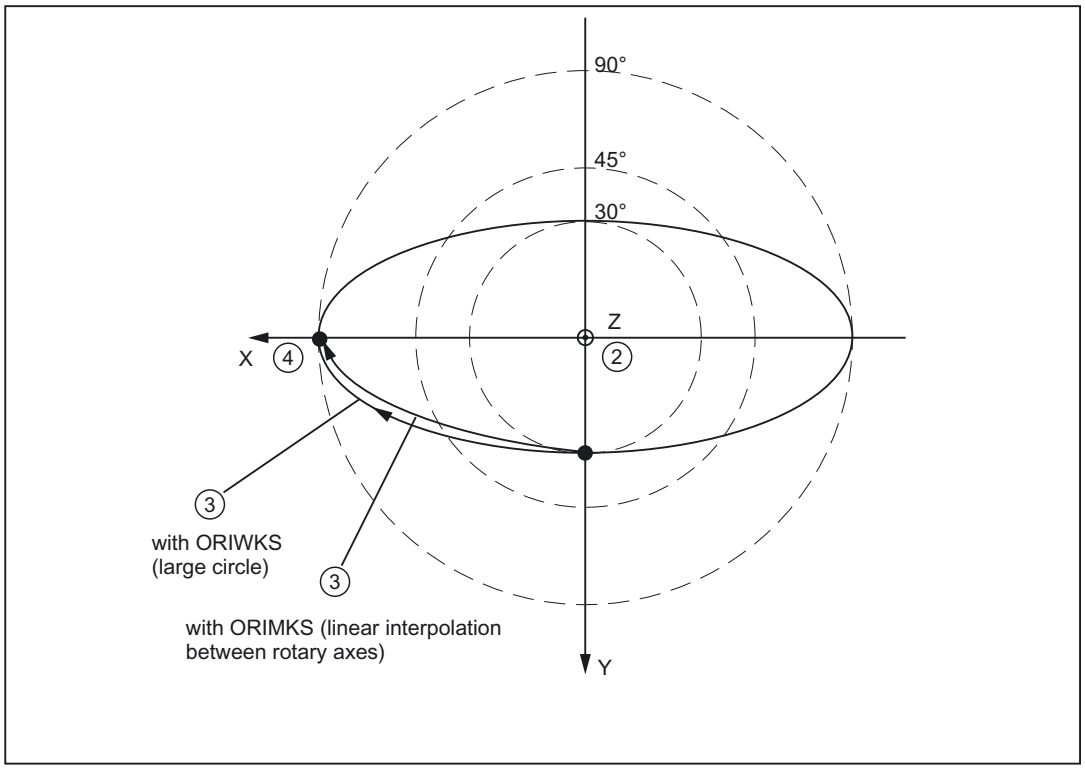


Figure 1-12 Change in orientation while machining inclined edges

ORIMCS 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 ⇒ ORIWCS is basic setting

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

Illegal tool orientation

If tool position is prgrammed 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 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 0n, 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 parts 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 Chapter "Polynomial Interpolation of Orientation Vectors".

Note

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

References: /FB1/ Function Manual, Basic Functions; Tool Offset; Orientable Tool Carriers (W1)

1.2.5 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-xis transformation)

MD24640 $MC_TRAFO5_POLE_LIMIT_2 (closing angle tolerance for interpolation by pole for 5-xis transformation)

Note

Singularities are dealt with differently in SW 5.2 and higher: There is now only one relevant machine data $MC_TRAFO5_POLE_LIMIT (see Chapter "Singularities of Orientation" or "Programming Manual Work Preparation").

$MC_TRAFO5_NON_POLE_LIMIT

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 MD, then a deviation from the specified interpolation is made so that the interpolation passes through the pole.

$MC_TRAFO5_POLE_LIMIT

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.

$MC_TRAFO5_NON_POLE_LIMIT

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.

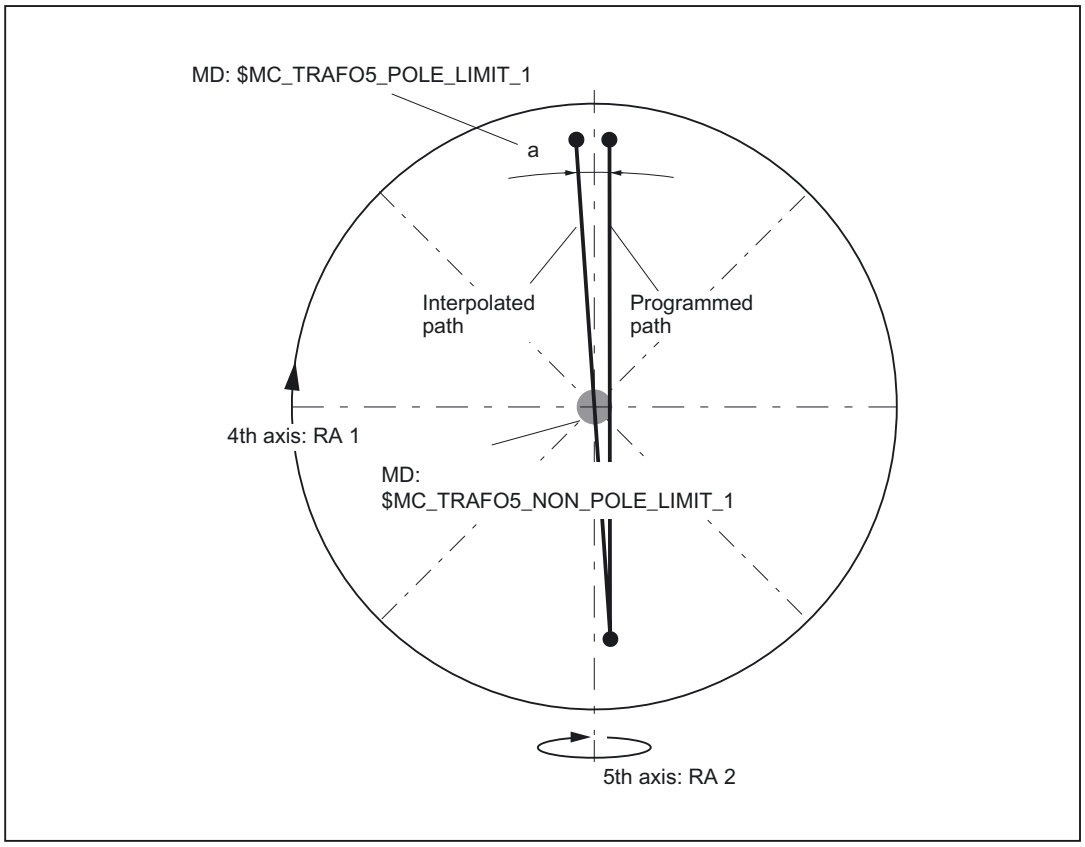


Figure 1-13 5-axis transformation; orientation path in pole vicinity. Example for machine type 1: 2-axis swivel head with rotary axis RA 1 (4th axis of transformation) and rotary axis RA 2 (5th axis of transformation)

MD21108 $MC_POLE_ORI_MODE

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 (behavior during large circle interpolation at pole position)

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.

TRAFO5_NON_POLE_LIMIT1/2

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:

$ MC_TRAFO5_NON_POLE_LIMIT1/2

All setting values are described in "Channel-specific Machine Data".

1.3 3-axis and 4-axis transformations

Introduction

3-axis and 4-axis transformations are special types of the 5-axis transformation initially described. Orientation of the tool is possible only in the plane perpendicular to the rotary axis. The transformation supports machine types with movable tool and movable workpiece.

Kinematics variants

The variants specified in the following table apply both for 3-axis and 4-axis transformations.

Variants of 3-axis and 4-axis transformations

Machine type	Swiveling/ rotary	Rotary axis is parallel	Orientation plane	MD: $MC_TRAFO_ TYPE_n	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	any
2	Workpiece	X	Y - Z	32, 33	Z
		Y	X - Z	34, 35	
		any	any *	40	any

*Note: on types 24 and 40 *

In the case of transformation types 24 and 40, the axis of rotation and tool orientation can be set so that the change in orientation takes place at the outside of a taper and not in a plane.

Zero position

Tool orientation at zero position is the position of the tool with G17 as the active working plane and position of the rotary axis at 0 degrees.

Axis assignments

The three translatory 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[geoax] = 0
  - $MC_TRAFO_AXES_IN_n[geoax] = 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[geoax] = ...
  - $MC_TRAFO_AXES_IN_n[geoax] = ...
    $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 Chapter "Example for 3- and 4-axis Transformation".

1.4 Transformation with swivelled linear axis

Applications

Transformation with a swiveling linear axis can be used if the application is characterized by the kinematics described in Chapter "Orientation Transformation with Linear Swivel Axis" and only a small swivel range (<<± 90 degrees) is crossed by the first rotary axis.

Kinematics variants

Orientation transformation with swivelling linear axis forms a transformation group of its own. It is specified by the following machine data $MC_TRAFO_TYPE_n (n = 1, 2, 3, 4) with the values:

Transformation type	1st rotary axis	2nd rotary axis	Swivelled linear axis
64	A	B	Z
65	A	C	Y
66	B	A	Z
67	B	C	X
68	C	A	Y
69	C	B	X

Pole

The corresponding transformation has a pole with tool orientation parallel to the second rotary axis. Singularity occurs in the pole position because the third linear axis is parallel to the plane of the first two linear axes, thus excluding the possibility of compensating movements perpendicular to this plane.

Parameterization

The following machine data with the following meanings are used to adjust the transformation equations to the machine (n=1,2):

$MC_TRAFO5_PART_OFFSET_n	Vector from the second rotary axis to workpiece table zero
$MC_TRAFO5_ROT_AX_OFFSET_n	Axis positions of the two rotary axes at the initial position of the machine
$MC_TRAFO5_ROT_SIGN_IS_PLUS_n	Sign with which the rotary axis positions are included in the transformation
$MC_TRAFO5_JOINT_OFFSET_n	Vector from machine zero to the second rotary axis
$MC_TRAFO5_BASE_TOOL_n	Vector from the toolholder (flange) to the first rotary axis (measured at machine initial position)
$MC_TRAFO5_TOOL_ROT_AX_OFFSET_n	Vector from machine zero to the first rotary axis (measured at machine initial position)

Definition of required values

As an aid for defining the values for the above-mentioned machine data, the following two sketches show the basic interrelations between the vectors.

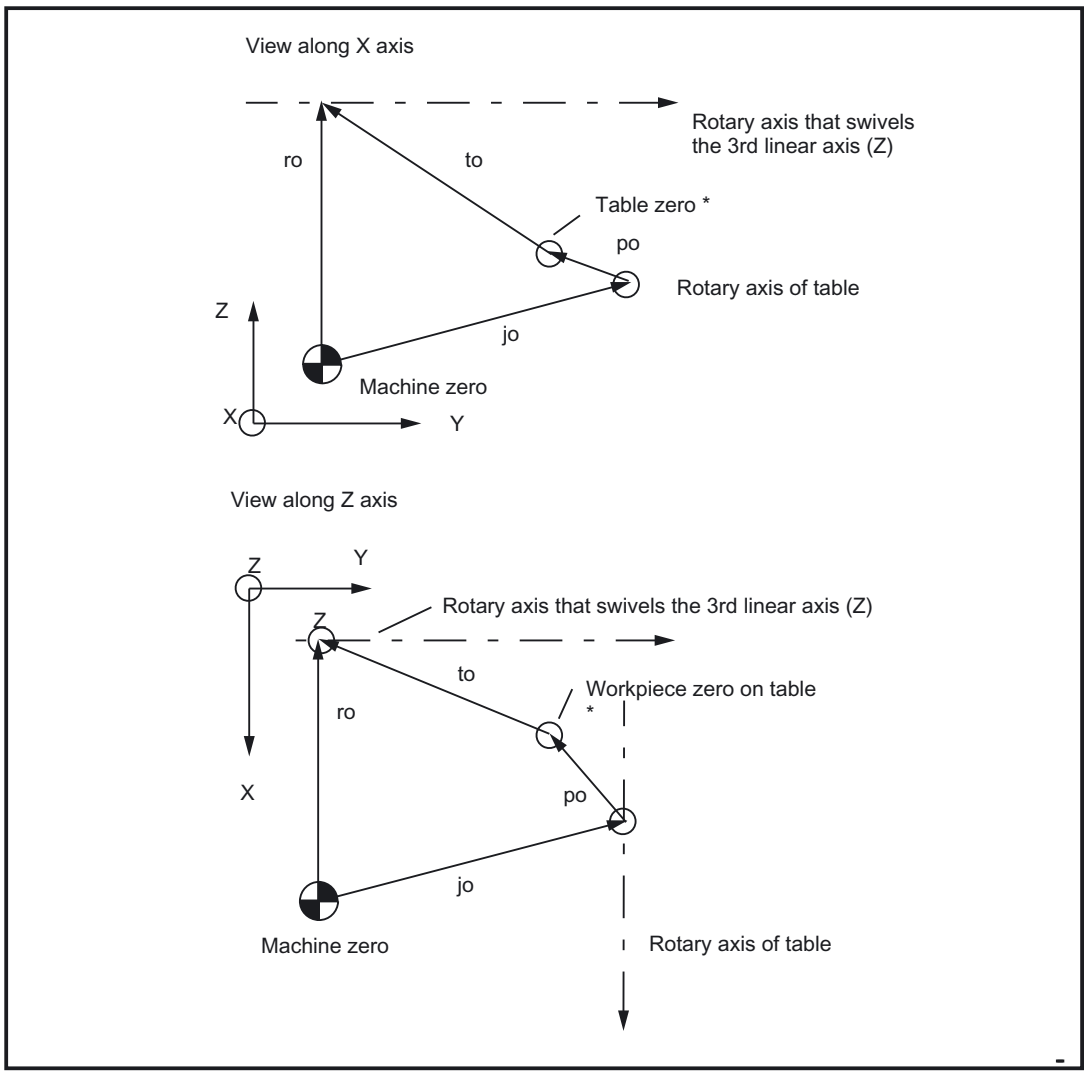


Figure 1-14 Projections of the vectors to be set in machine data

Meanings of vector designations:

- $MC_TRAFO5_PART_OFFSET_n: **po**
- $MC_TRAFO5_TOOL_ROT_AX_OFFSET_n: **ro**
- $MC_TRAFO5_JOINT_OFFSET_n: **jo**
- $MC_TRAFO5_BASE_TOOL_n: **to**

Note

For the previous diagram, it has been assumed that the machine has been traversed so that the tool holding flange is in line with table zero (marked by *). If this cannot be implemented for geometric reasons, the values for **to** must be corrected by the deviations.

The Figure "Example of Vector Designation for MD-settings" shows vector components for the machine shown in Figure "Machine with linear swivel axis in Zero Position" (Chapter "Orientation Transformation with Linear Swivel Axis") and they are named accordingly.

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.

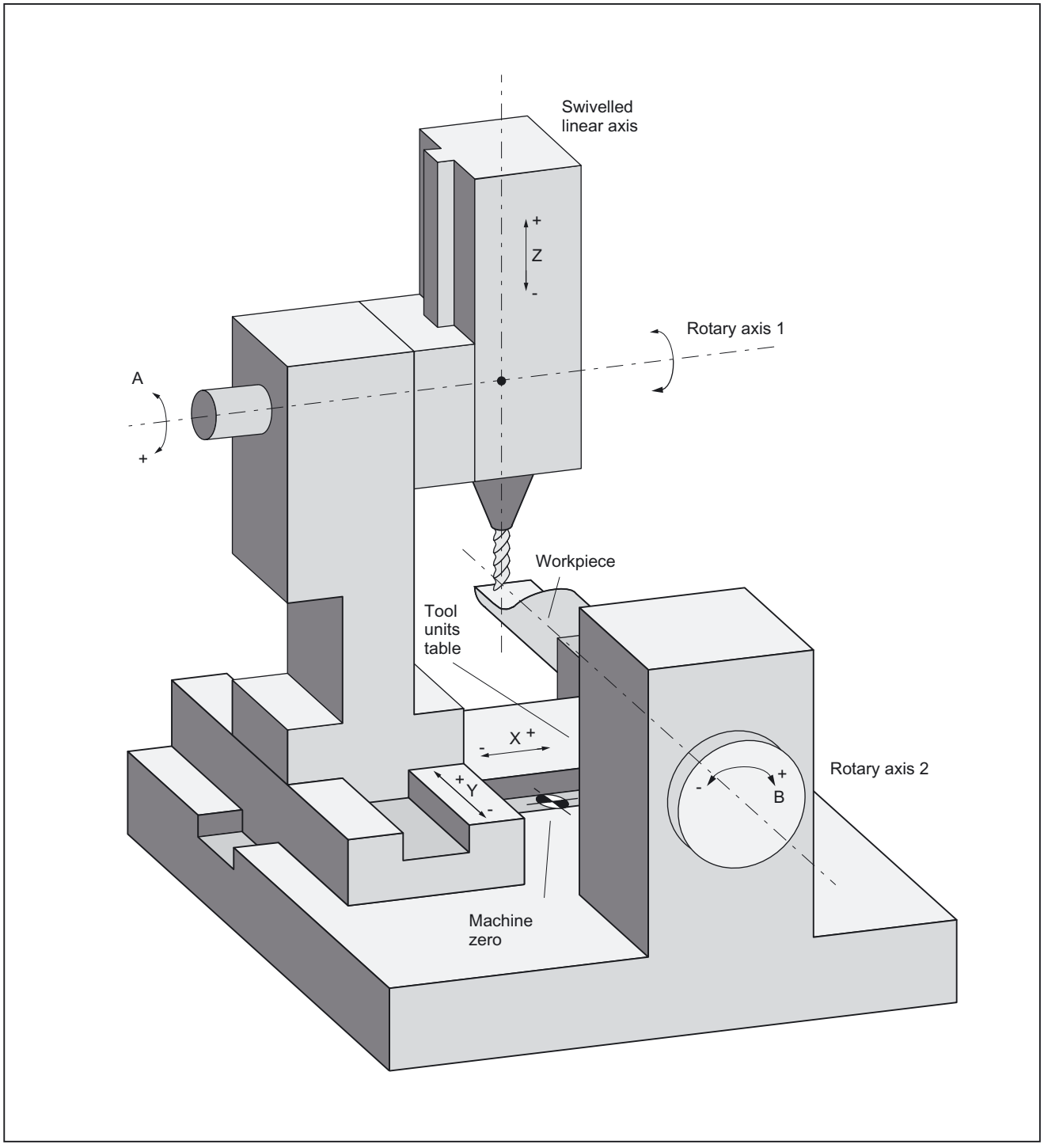


Figure 1-15 Machine with swivelling linear axis in position zero

The following conversion of geometry into machine data to be specified, is based on the example in Figure "Machine with swivelling linear axis in position zero".

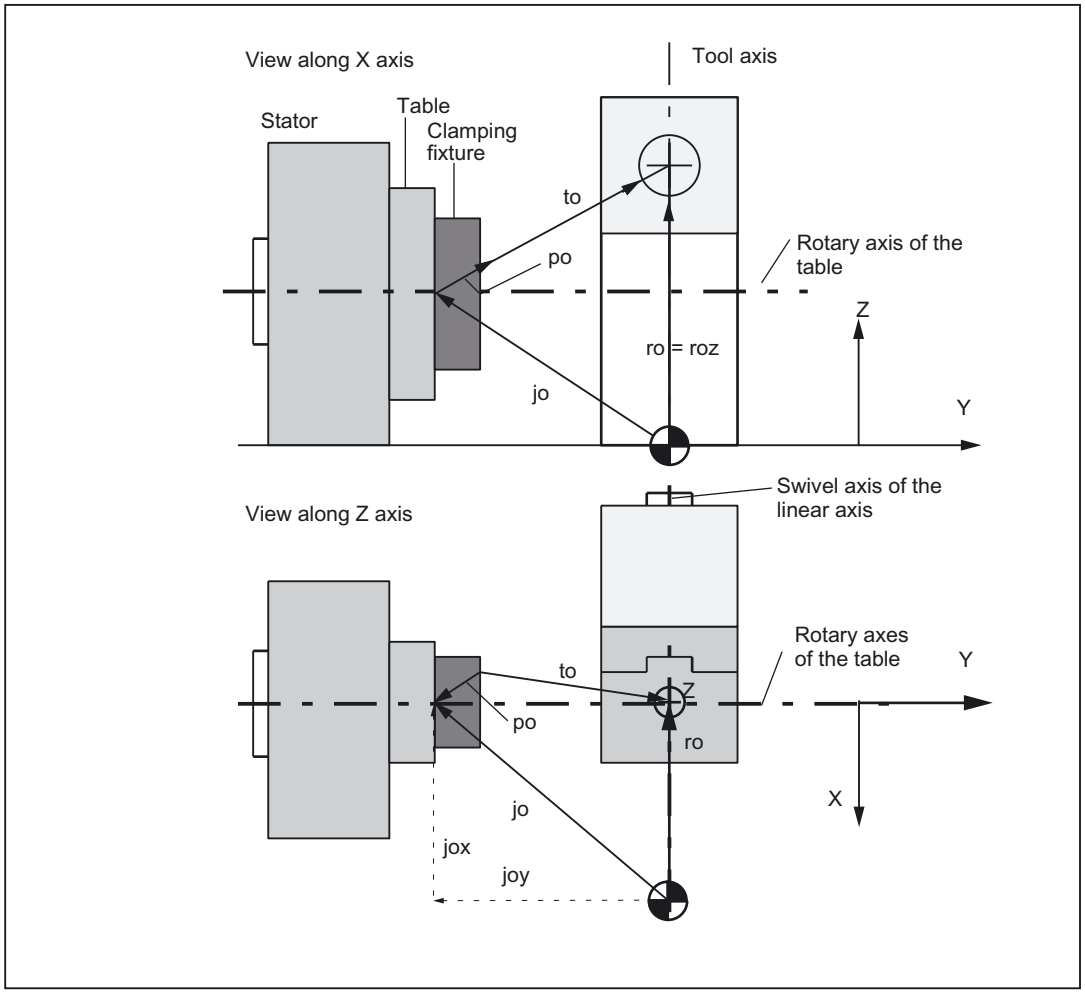


Figure 1-16 Example of vector designation for MD-settings in Figure "Machine with Swivelling Linear Axis in Zero Position"

Procedure for setting machine data

Perform the following operation:

- Calculate the x- and y-components for vector **jo**, listed in the figure below, "Example of Vector Designation for MD-settings" in Figure "Machine with Swivelling Linear Axis in Zero Position".
- Determine the z components of the corresponding vectors, as shown in the upper section for $\mathbf{ro}_z$.
- Apply the four machine data accordingly:

  $MC_TRAFO5_PART_OFFSET_n

  $MC_TRAFO5_TOOL_ROT_AX_OFFSET_n

  $MC_TRAFO5_JOINT_OFFSET_n

  $MC_TRAFO5_BASE_TOOL_n

This procedure can be used for all kinematics specified under "Kinematics Variants". Observe the tips given in Figure "Projections of Vectors to be set in Machine Data".

Zero components

With certain geometries or machine zero positions, individual components or complete vectors can become zero.

Machine type

The machine shown in Figure "Projections of Vectors to be set in Machine Data" corresponds to variation 1. Therefore, type of transformation 64 must be set in the following machine data (4 least-significant bits in MD).

$MC_TRAFO_TYPE_n

Activation

The transformation for a swiveled linear axis is activated in the same way as 5-axis transformations. Details are described in Chapter "Activation and Application of 3- to 5-axis Transformation".

Tool orientation

The same applies with regard to tool orientation, as described in Chapter "Tool Orientation".

1.5 Universal milling head

1.5.1 Fundamentals of universal milling head

Note

The following description of the universal milling head transformation has been formulated on the assumption that the reader has already read and understood the general 5-axis transformation described in Chapter "5-axis Transformation". Please note that where no specific statements relating to the universal milling head are made in the following section, the statements relating to general 5-axis transformation apply.

Applications

A universal 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.

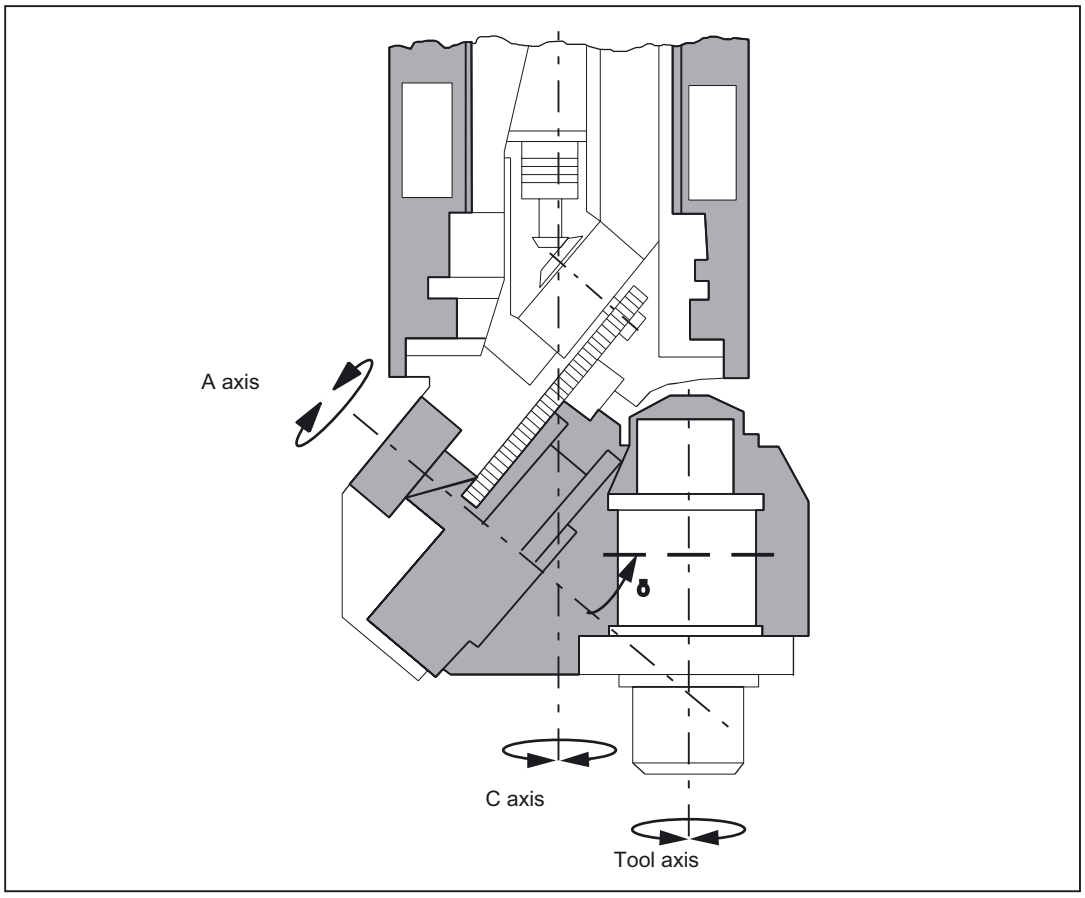


Figure 1-17 Universal milling head

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

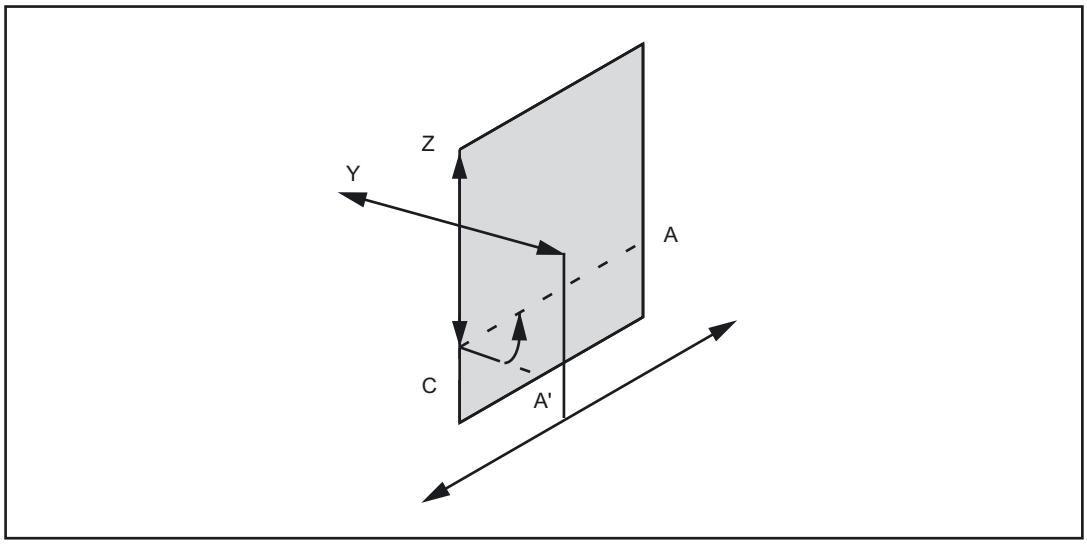


Figure 1-18 Position of axis A'

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.

1.5.2 Parameterization

Setting the type of transformation

Using the following table, the data required in order to set the following machine data appropriately for any given machine kinematics can be read (general concept).

$MC_TRAFO_TYP_n

Table 1-1 MD $MC_TRAFO_TYPE_n

Bit	Decimal	Description
8	128	Bit indicating the type of transformation: 1: Transformation for universal milling head
7 6	0 32 64	00: Movable tool 01: Movable workpiece 10: Movable tool and workpiece
5 4	 0 8 16	Direction of orientation of tool in position zero 00: X direction 01: Y direction 10: Z direction

Bit	Decimal	Description
3		Axis sequence
2	0	000: AB'
1	1	001: AC'
	2	010: BA'
	3	011: BC'
	4	100: CA'
	5	101: CB'

Among the full range of options specified in the general concept above, the settings highlighted in gray in the following table are implemented in SW 3.1, the others in SW 3.2 and higher.

Implementation

The table below gives the values for $MC_TRAFO_TYPE_n for the configurable axis sequences and for the orientation direction of the tool in position zero, showing separate data for movable tool, movable workpiece and movable tool and workpiece. The transformation does not support any table elements which do not contain a preset value.

	Direction of orientation of tool in position zero								
	Tool			Workpiece			Tool/workpiece		
Axis sequence	X	Y	Z	X	Y	Z	X	Y	Z
AB'	128	136							
AC'	129		145						
BA'	130	138							
BC'		139	147						
CA'	132		148						
CB'		141	149						

Example of transformation type

$MC_TRAFO_TYPE = 148 for example, means:

The first rotary axis is parallel to the Z axis,

the second rotary axis is an inclined X axis and tool orientation in zero position points in the Z direction.

Only the tool is moved by both rotary axes.

Bit 8 = 1: Universal milling head

Bit 6 and 7 = 00: Movable tool

Bits 5 and 4 = 10: Orientation in zero position in the Z direction

Bit 3 -1 = 100: Axis sequence CA'

Active machining plane

Since tool orientation in position zero can be set in directions other than just the Z direction, the user must ensure the active machining level is set so that tool length compensation takes effect 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 universal milling head transformation for calculation of the axis values is set in the same way as that of the other 5-axis transformations.

1.5.3 Traverse of universal milling head in JOG mode

JOG

The linear axes can be traversed normally in JOG mode. It is, however, difficult to set the orientation correctly by traversing these axes.

Activation of universal milling head

The transformation for a universal milling head in the program is activated as described in the following Section.

1.6 Activation and application of 3-axis to 5-axis transformation

Switch on

3-axis to 5-axis transformations (including the transformations for swiveled linear axis and universal milling head) are activated with the `TRAORI(n)` command, where n represents the number of the transformation (n = 1 or 2).

Once the `TRAORI(n)` command has been executed and the transformation thus activated, the following interface signal is set to "1":

DB21, ... DBX33.6 (transformation active)

If the machine data has not been defined for an activated transformation grouping, the NC program stops and the control system displays the alarm 14100 "Orientation transformation does not exist".

Deactivation

`TRAFOOF` or `TRAFOOF()` de-activates the currently active 3- to 5-axis transformation. This sets the interface signal to "0":

DB21, ... DBX33.6 (transformation active)

Switch-over

A switch-over can be made from one active transformation to another transformation configured in the same channel. To do this the `TRAORI(n)` command must be entered again with a new value for n.

RESET/end of program

The behavior of the control system with regard to 3-axis/5-axis transformations after run-up, end of program or RESET, is determined by machine data:

MD20110 $MC_RESET_MODE_MASK (specifying control basic setting na)

Bit 7: Reset behavior of "active kinematic transformation"

Bit 7=0:

Basic setting for active transformation after program end of the parts program or RESET according to MD20140 $MC_TRAFO_RESET_VALUE (transformation data set run-up (RESET/parts program end)) is designated with the following meaning:

0: After RESET no transformation is active

1 to 8: the set transformation according to machine data MD24100 $MC_TRAFO_TYPE_1 (transformation definition 1 in the channel) to machine data MD24460 $MC_TRAFO_TYPE_8 (transformation 8 in the channel) is active.

Bit 7=1:

Current setting for the active transformation is retained beyond RESET/end of parts program.

Option

The "5-axis transformation" function and its special types described in this Function Manual are only available as an option. f this option is not implemented in the control system and a transformation is called with the `TRAORI` command, error message 14780 "Block uses a function that has not been enabled" appears and the NC program stops.

If the 3 to 5-axis transformation in the machine data:
MD24100 $MC_TRAFO_TYPE_1 (definition of Transformation 1 in the channel) ...
MD24460 $MC_TRAFO_TYPE_8 (Transformation 8 in the channel)
is not specified, the the programming of `TRAORI` (1 or 2) triggers Alarm 14100 "Channel x block y orientation transformation not present".

If the following machine data is set without the 5-axis transformation option being enabled, there is no alarm.

$MC_TRAFO_TYPE_n

1.7 Generic 5-axis transformation and variants

1.7.1 Functionality

Scope of functions

The scope of functions of generic 5-axis transformation covers implemented 5-axis transformations (see Chapter "5-axis Transformation") for perpendicular rotary axes as well as transformations for the universal milling head (one rotary axis parallel to a linear axis, the second rotary axis at any angle to it, see Chapter "Universal Milling Head").

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 Chapter "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`.

1.7.2 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 tool. 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

Machine configuration is defined as usual (see Chapter "Configuration of a 5-axis Transfromation") in the following machine data:

$MC_TRAFO_TYPE_1, ..., _8

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

Table 1-2 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 ne 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

1.7.3 Generic orientation transformation variants

Expansion

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:

References:

/FB1/ Function Manual, Basic Functions; Tool Offset; Orientable Tool Carriers (W1)

/PG/ Programming Guide, Basics

Comparison

Besides the 3- and 4-axis transformations mentioned in Chapter "3- and 4-axis Transformations", the follwing 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.

1.7.4 Parameterization of orientable tool holder 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 tool holders. In both cases, machine kinematics is determined by the same data, which, due to different parameters, previously had to be entered twice - for tool holder via system varaibles 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 tool holder.

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 1-3 Machine types for generic 5-axis transformation

Machine type	1	2	3	4
Swivel/ rotatable:	Tool	Workpiece	Tool/workpiece	Type 3 or orientable tool holder
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 are 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 NEWCONFIG. 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 first 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 second 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)

Transformation type "P" (in accordance with MD24100 $MC_TRAFO_TYPE_1 = 40)	
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)

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)
MD 24550 $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
                                       ; tool holder
```

1.7.5 Extension of the generic transformation to 6 axes

Application

With the maximum 3 linear axes and 2 rotary axes, the motion and direction of the tool in space can be completely described with the generic 5-axis transformation. Rotations of the tool around itself, as is important for a tool that is not rotation-symmetric or robots, require an additional rotary axis. The previous generic 5-axis transformation will therefore be extended by a 3rd rotary axis and further functions added.

- Extension to 3 linear axes and 3 rotary axes, i.e. 6 axes.
- General use of the generic orientation transformation with unchanged parameterization of machine data.
- Cartesian manual travel also for the generic transformation.

Kinematics for the 6-axis transformation

The 6-axis transformation is based on the generic 5-axis transformation and is extended by transformation type 57. Therefore, four different machine kinematics exist that are differentiated through the specification of the transformation type in the following machine data:

MD24100 $MC_TRAFO_TYPE_1 = Trafotyp (definition of transformation 1 in the channel)

Table 1-4 Overview of machine types for the generic 6-axis transformation

Machine type	1	2	3	4
Swivel/rotatable	Tool	Workpiece	Tool/ workpiece	Tool/ workpiece
Transformation types	24	40	56	57
Orientation in space, rotation of the axes	Unchanged. All three axes rotate the tool	Unchanged. All three axes rotate the workpiece	Tool by two axes, workpiece by a rotary axis	Tool by one axis, workpiece by two rotary axes

In all four cases, the first rotary axis is the one which closest to the workpiece and the third rotary axis the one which closest to the tool in the kinematic chain.

Note

The four specified 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.

Dedicated machine data exist for each general transformation or for each orientation transformation that are differentiated by the suffixes _1, _2 etc. (e.g. MD24100 $MC_TRAFO_TYPE_1, MD24200 $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.

Configuration

For configuration **of a 6-axis transformation** the extensions of the following machine data are required:

- The channel axis index of the 3rd rotary axis must be entered in the following machine data:

  MD24110 $MC_TRAFO_AXES_IN_1[5] (axis assignment for transformation)

- The direction of the 3rd rotary axis must be specified in the following machine data:

  MD24573 $MC_TRAFO5_AXIS3_1[0..2] (direction third rotary axis)

- An orientation normal vector with a length not equal to zero and which is not parallel or anti-parallel to the orientation vector defined in machine data MD24574 $MC_TRAFO5_BASE_ORIENT_1[0..2] (basic tool orientation), must be specified in machine data MD24576 $MC_TRAFO6_BASE_ORIENT_NORMAL_1[0..2] (tool normal vector).

The **previous offsets (vector):**

- MD24550 $MC_TRAFO5_BASE_TOOL_1[0..2]

  (vector of the basic tool with activation of the 5-axis transformation 1)

- MD24560 $MC_TRAFO5_JOINT_OFFSET_1[0..2]

  (vector of the kinematic offset of 5-axis transformation 1)

- MD24558 $MC_TRAFO5_JOINT_OFFSET_PART_1[0..2]

  (vector of kinematic offset in table)

- MD24500 $MC_TRAFO5_PART_OFFSET_1[0..2]

  (translation vector 5-axis transformation 1)

The following machine data is added as **new offset (vector)**, describing the offset between the second and third rotary axis:

- MD24561 $MC_TRAFO6_JOINT_OFFSET_2_3_1[0..2]

  (vector of kinematic offset)

Note

Existing machine data blocks are compatible for transfer, without any changes having to be made in the machine data. The new machine data therefore do not have to be specified for a 3-/4-/5-axis transformation.

Programming of orientation

With the extension of the generic orientation transformation to 6 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. One axis direction, that of the third axis, (typically in the Z direction) defines the orientation.

Two degrees of freedom are required for the specification of this direction. The third degree of freedom is defined via **a rotation** around this direction, e.g. through the specification of an angle THETA or a direction vector for one of the two other axes of the coordinate system, see Chapter "Rotation of the Orientation Vector".

The new addresses AN3, BN3, CN3 define the direction of the second axis, of the coordinate system (typically the Y axis) of the orientation normal vector. The programmed orientation normal vector should be perpendicular to the orientation and is only possible when both programmed vectors are not parallel or anti-parallel. Otherwise, alarm 4342 is output.

The direction of the first axis, the X axis, is then uniquely defined.

Default setting of the orientation normal vector

The default setting of the orientation normal vector in the transformation can also be defined as for the default setting of the orientation in one of three ways:

Specification for the **activation of the transformation**

1. Vector components are transferred as parameters 8 to 10:

   Parameter 1: Transformation No.

   Parameter 2 - 4: Orientation vector,

   Parameter 5 - 7: Rotary axis offsets
2. If **no** orientation normal vector has been specified and **a** tool is active, the vector is taken from the tool data.
3. If **no** orientation normal vector has been specified and also **no** tool is active, the vector defined in the following machine data is used.

   MD24567 $MC_TRAFO6_BASE_ORIENT_NORMAL_1[0..2] (tool normal vector)

The position of the orientation coordinate system of a standard tool depends on the **active plane** G17, G18, G19 according to the following table:

Table 1-5 Position of the orientation coordinate system

	G17	G18	G19
Direction of the orientation vector	Z	Y	X
Direction of the orientation normal vector	Y	X	Z

Note

The orientation vector of a tool can also be defined via system the variables $TC_DPV bzw. $TC_DPV3 - $TC_DPV5 in tool data - see /FB1/ Function Manual, Basic Machine, Tool Corrections (W1), "Sum and Set-up Corrections".

This option is expanded in order to specify the orientation normal vector, using system varables $TC_DPVN3 - $TC_DPVN5. The meaning of the vector components is similar to the meaning of the components of the tool orientation:

$TC_DPVN3 is the component in the direction of tool length L1,

$TC_DPVN4 is the component in the direction of tool length L2,

$TC_DPVN5 is the component in the direction of tool length L3,

The following machine data must have the value 3 in order to allow the new tool parameters to be used:

MD18114 $MN_MM_ENABLE_TOOL_ORIENT (assign orientation to tool cutting)

The coordinate system is not rotated through the programming of a rotation of the tool with AN3, BN3, CN3 or THETA.

Programming example

A programming example for generic 5-axis transformation is shown in Chapter "Example for Generic 5-axis Transformation".

Restrictions

Generic 6-axis transformation requires 6 axes and is therefore only available on systems with at least 6 axes.

1.7.6 Extension of the generic transformation to 7 axes

Application

The generic 5-/6-axis transformation with transformation type 24 is extended by a 7th or 6th axis, which rotates the workpiece. The work space of the transformation can be expanded in this way.

Requirement

Generic 7-axis transformation requires 7 (6) axes and is therefore only available on systems with at least 6 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 Transformer 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 pages, 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 recorded here.

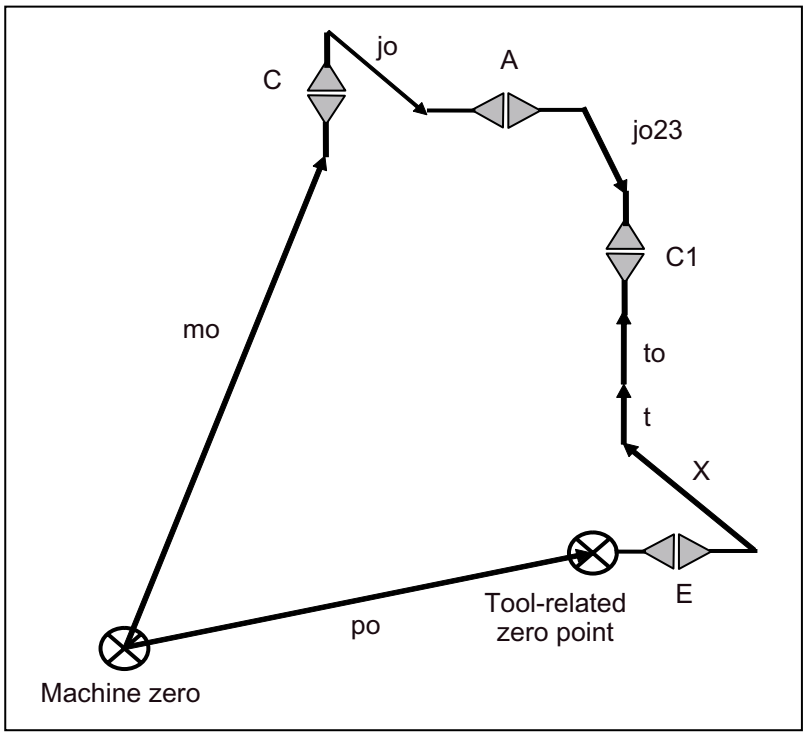


Figure 1-19 Schematic diagram of 7-axis kinematics

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]

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 code.

   – 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 codes of Groups 25 and 51 must be set accordingly:

   G code group 25: ORIMKS

   G code group 51: ORIAXES (if MD 21104 $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 approached directly. The orientation is programmed in relation to the machine.

   Example:

```
TRAORI(1)
ORIAXES
ORIMKS
G1 X500 Y300 Z800 C15 A5 C1=10 E1=120
```

2. **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 code group 25: ORIWKS

   G code group 51: ORIVECT (if MD 21104 $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:

```
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 the 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.

1.7.7 Cartesian manual travel with generic transformation

Note

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

Functionality

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

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

The following machine data not only activates the function, but also sets the permitted coordinate systems.

MD21106 $MC_CART_JOG_SYSTEM (coordinate systems for Cartesian JOG)

For JOG motion, one of the three reference systems can be set not only for the translation/movement of the geometry axes, but also for tool orientation/movement of the orientation axes via the setting data SD42650 $SC_CART_JOG_MODE (coordinate systems for Cartesian manual traverse) independently from one another.

Activation

The following machine data not only activates the function, but also sets the permitted coordinate systems.

MD21106 $MC_CART_JOG_SYSTEM (coordinate systems for Cartesian JOG)

The following setting data sets the virtual kinematics used for traversing motion of the orientation:

SD42660 $SC_ORI_JOG_MODE (definition of virtual kinematics for JOG)

As opposed to the generic 5-/6-axis transformation, only kinematics can be set in which the rotary axes are perpendicular to one another.

The traversing of the geometry and orientation axes is performed via the VDI interface signals of the geometry or orientation axes.

Translations

A translatory movement can be used to move the tool tip (TCP) 3-dimensionally in parallel to the axes of the set reference system. Traversing is performed via the VDI interface signals of the geometry axes.

Note

For further information about the representation of the translations for the Cartesian manual travel in the corresponding coordinate systems, see:
References:
Function Manual Extension Functions; Kinematic Transformation (M1)

Tool orientation

The tool can be aligned to the workpiece surface via an orientation movement. The motion of the orientation axes is triggered by the PLC via the VDI interface signals of the orientation axes. The virtual orientation axes execute rotations around the fixed directions of the relevant reference system. Virtual kinematics is defined by the following setting data via the active transformation:

SD42660 $SC_ORI_JOG_MODE = 0

Rotations of the orientations

Rotation of the orientation axes is defined by additional settings of the following setting data:

SD42660_$MC_ORI_JOG_MODE

The options are as follows:

Rotations with JOG

With JOG, the rotations around the specified directions of the respective reference system can be performed with Euler angle or RPY angle.

SD42660 $SC_ORI_JOG_MODE = 1: When jogging, **Euler angles** are traversed, i.e.:

the first axis rotates around the z direction,

the second axis rotates around the x direction,

the third axis (if present) rotates around the new z direction.

SD42660 $SC_ORI_JOG_MODE = 2: When jogging, **RPY angles** are traversed with rotation sequence XYZ, i.e.:

the first axis rotates around the x direction,

the second axis rotates around the y direction,

the third axis (if present) rotates around the new z direction.

SD42660 $SC_ORI_JOG_MODE = 3: When jogging, **RPY angles** are traversed with rotation sequence ZYX, i.e.:

the first axis rotates around the z direction,

the second axis rotates around the y direction,

the third axis (if present) rotates around the new x direction.

Rotation sequence of the rotary axes

Rotation sequence of the rotary axes is set via the following setting data:

- **SD42660 $SC_ORI_JOG_MODE = 4:**

  via machine data MD21120 $MC_ORIAX_TURN_TAB_1 (definition of reference axes for ORI axes)

- **SD42660 $SC_ORI_JOG_MODE = 5:**

  via machine data MD21130 $MC_ORIAX_TURN_TAB_2 (definition of reference axes for ORI axes)

For further explanations of orientation movements, see Chapters "Orientation" and "Orientation Axes".

Note

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

1.8 Restrictions for kinematics and interpolation

Fewer than 6 axes

Not all degrees of freedom are available for orientation. The following special rules therefore apply.

5-axis kinematics

This has only two degrees of freedom for orientation. The assignment of orientation axes and tool vector direction must be selected so that there is no rotation about the tool vector itself. As a result, only two orientation angles are required to describe the orientation. If the axis is traversed by `ORIVECT`, the tool vector performs a pure swivelling motion.

3-and 4-axis kinematics

For 3- and 4-axis kinematics only degree of freedom is available. The respective transformation determines the relevant orientation angle.

It only makes sense to traverse the orientation axis with ORIAXES. Linear interpolation for the orientation axis is direct.

Interpolation across several blocks

Machine tools with the kinematics of an orientable tool holder are capable of orienting the tool in space. The orientation of the tool is almost always programmed in each block. It is possible to specify tool orientation directly via rotary axis settings.

If orientations of a tool are interpolated over several successive blocks, undesirable abrupt changes in the orientation vector may be encountered at the block transitions. This causes irregular velocity and acceleration changes in the rotary axes at the block transitions.

Large circle interpolation can be used to generate a movement of the orientation axes with continuous velocity and acceleration across several blocks. Orientation axes behave like normal linear axes if only G1 blocks are interpolated.

In the case of linear axes, movement with continuous acceleration is achieved by using polynomials for axis interpolation.

Tool orientation using orientation vectors

A much better method is to use orientation vectors to program tool orientation in space.

Consider the features of polynomial interpolation of orientation vectors described in Chapter "Polynomial Interpolation of Orientation Vectors".

Further explanations about the tool orientation by means of orientation vectors for machine tools can be found in Chapter "Polynomial Interpolation of Orientation Vectors" and in subsequent chapters.

1.8.1 Singularities of orientation

Description of problem

As described in Chapter "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-xis 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: /PGA/ Programming Manual, Work Preparation, Transformations, Cartesian PTP travel

Example for machine type 1

Rotatable tool

Both rotary axes change the orientation of the workpiece. The orientation of the workpiece is fixed.

2-axis swivel head with rotary axis RA 1 (4th transformation axis) and rotary axis RA 2 (5th transformation axis)

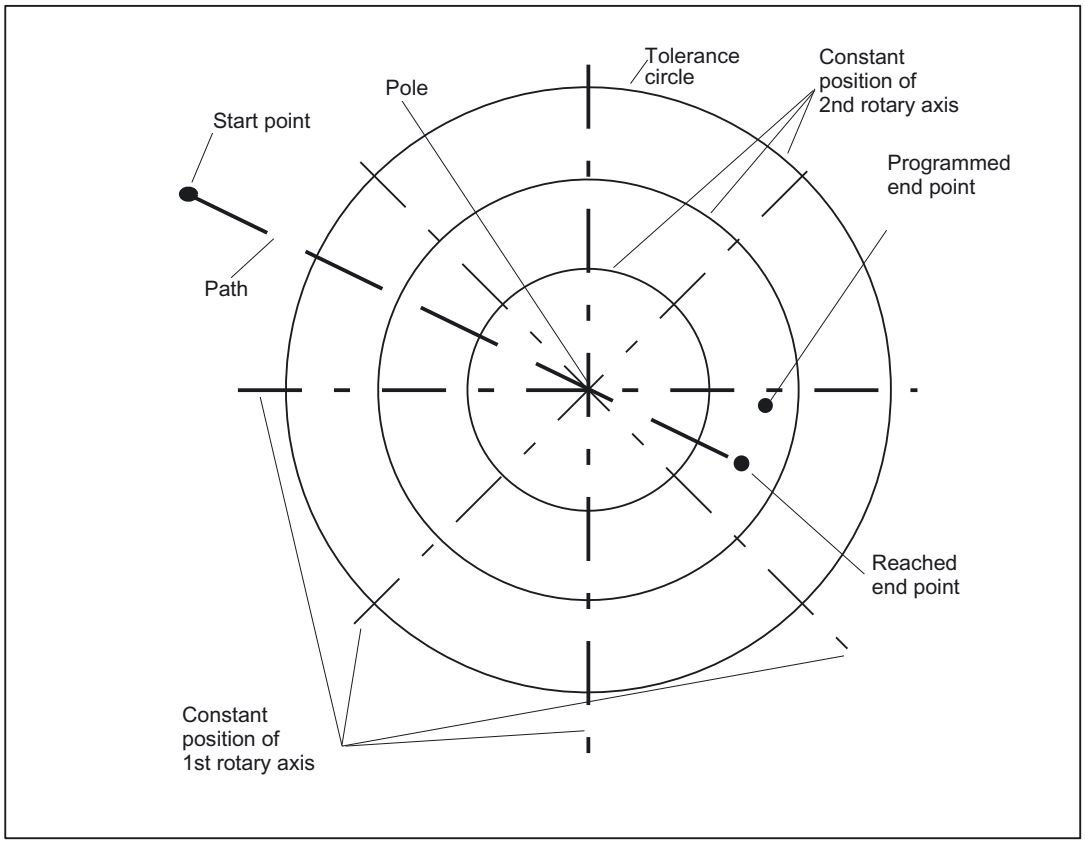


Figure 1-20 Generic 5-axis transformation; end point of orientation inside tolerance circle.

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.

1.9 Orientation

1.9.1 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

There are three different ways to define basic orientation:

1. via the transformation call
2. via the orientation of the active tool
3. via a machine data

via the transformation call

For 1.:

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; the vector in the example therefore 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 will not be modified by any active frame.

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 methods described below.

If a basic orientation is defined by the above method, it cannot be altered while a transformation is active. The orientation can be changed only by selecting the transformation again.

Via the orientation of the active tool

For 2.:

The basic orientation is determined by the tool

- if it has not been defined through specification of a direction vector in the transformation call
- and if 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:

References:
/FB1/ Function Manual, Basic Functions; Tool Offset; Orientable Tool Holders (W1)

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 cancelling the definition of tool orientation, the basic orientation programmed in machine data becomes operative.

Via machine data

For 3.:

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 (universal 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.

1.9.2 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)

The following machine data specifies the conditions under which the rotary axis positions may be modified:

MD21180 $MC_ROT_AX_SWL_CHECK_MODE

Value 0: No modification permitted (default, equivalent to previous behavior).

Value 1: Modification is only permitted if axis interpolation is active (ORIAXES or ORIMKS).

Value 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 is shown in Chapter "Example for Generic 5-axis Transformation" for modifying the rotary axis motion of a 5-axis machine with a rotatable tool.

1.9.3 Orientation compression

Function

The compressor functions COMPON, COMPCURV and COMPCAD can also be used to compress NC programs containing orientations programmed with the help of direction vectors, to a definable tolerance.

Requirement

A precondition for the compression of orientations is the availability of the "Orientation Transformation" option.

Conditions

The orientation movement is compressed in the following cases:

- Active orientation transformation (`TRAORI`)
- Active large radius circular interpolation (i.e. tool orientation is changed in the plane which is determined by start and end orientation).

  Large circle interpolation is performed under the following conditions:

  – MD21104 $MC_ORI_IPO_WITH_G_CODE = 0,
  ORIWKS is active and
  orientation is programmed with the help of vectors (with `A3`, `B3`, `C3` or `A2`, `B2`, `C2`).

  – MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 and
  `ORIVECT` or `ORIPLANE` is active.

The tool orientation can be programmed either as a direction vector or with rotary axis positions. If one of the G-codes ORICONxx or ORICURVE is active, or if polynomials for the orientation angles (PO[PHI] and PO[PSI]) are programmed, no large circle interpolation will be executed.

Parameter assignment

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 axis, the compressor creates a spline curve which deviates from the programmed end points of each axis by a maximum of the value set with the following machine data.

MD33100 $MA_COMPRESS_POS_TOL (maximum tolerance with compression)

Contour accuracy

The maximum contour deviations (geo axes) and tool orientation are specified via the following setting data:

SD42475 $SC_COMPRESS_CONTUR_TOL (maximum contour deviation for compressor)

SD42476 $SC_COMPRESS_ORI_TOL (maximum angular deviation for tool orientation compressor)

SD42477 $SC_COMPRESS_ORI_ROT_TOL (maximum angular deviation for the angle of rotation of the tool) (only available on 6-axis machines)

Note

It is only possible to specify a maximum angular deviation for tool orientation if an orientation transformation (TRAORI) is active.

Compression mode

The manner in which the tolerances are to be considered is set via the **unit position** in the machine data:

MD20482 $MC_COMPRESSOR_MODE (mode of compression)

Value	Meaning
0	The tolerances specified with MD33100 $MA_COMPRESS_POS_TOL are observed for all the axes (geo and orientation axes).
1	The contour tolerance specified with SD42475 $SC_COMPRESS_CONTUR_TOL is effective for the geometry axes. The axis-specific tolerances MD33100 $MA_COMPRESS_POS_TOL are effective for the orientation axes.
2	The axis-specific tolerances MD33100 $MA_COMPRESS_POS_TOL are effective for the geometry axes. The maximum angular deviations SD42476 $SC_COMPRESS_ORI_TOL or SD42477 $SC_COMPRESS_ORI_ROT_TOL are effective for the orientation axes.
3	The contour tolerance specified with SD42475 $SC_COMPRESS_CONTUR_TOL is effective for the geometry axes. The maximum angular deviations SD42476 $SC_COMPRESS_ORI_TOL or SD42477 $SC_COMPRESS_ORI_ROT_TOL are effective for the orientation axes.

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 **vectors** via:

  `A3=<...> B3=<...> C3=<...>`

- Programming of the Euler**angles** or RPY-**angles** via:

  `A2=<...> B2=<...> C2=<...>`

Rotation of the tool

For **six-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 additional rotation is programmed, can only be compressed if the angle of rotation changes **linearly**, meaning that a polynomial with `PO[THT]=(...)` for the angle of rotation should not be programmed.

General structure of an NC block that be compressed

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.

Activation/deactivation

Compressor functions are activated using the modal G codes `COMPON`, `COMPCURV` or `COMPCAD`.

`COMPOF` terminates the compressor function.

Programming example

Refer to " example: Compression of an orientation (Page 133) "

References

The COMPON, COMPCURV and COMPCAD compression functions are described in: Function Manual, Basic Functions; Continuous Path Mode, Exact Stop, LookAhead (B1), Chapter: "NC block compression"

1.9.4 Orientation relative to the path

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 parts 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 using 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, can **not** be used in conjunction with the function "Orientation smoothing". Fot this OSOF must be active in the parts program. Otherwise alarm 10980 "Orientation smoothing not possible" is generated.

Activate orientation relative to the path

The extended function "Orientation relative to the path" is activated with the following machine data:

MD21094 $MC_ORIPATH_MODE > 0 (setting for path relative orientation ORIPATH)

The tool orientation relative to the path is activated in the part program by programming ORIPATH. A kink in the orientation path, e.g. as can occur at a corner in the contour, can be smoothed with ORIPATH.

Orientation at block transition

In case of the following machine data, tool orientation is always continuous at a block transition:

MD21094 $MC_ORIPATH_MODE = 0

With the following machine data a jump in tool orientation can occur at a block transition:

MD21094 $MC_ORIPATH_MODE > 0

A jump in orientation always occurs when either the path tangent or the surface normal vector does not change smoothly at a block transition.

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 circule 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 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 normal vector as at the end of the previous block.
2. Not only the tangent, but also the 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 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.

Set orientation relative to the path

The following machine data is used to set in which way the orientation relative to the path is to be interpolated.

MD21094 $MC_ORIPATH_MODE (setting for path relative orientation `ORIPATH`)

With `ORIPATH` the behavior of tool orientation interpolation relative to the path can be activated for various functions:

Meaning of **units**	activate proper orientation relative to the path

0: The tool orientation only has the reference to the path tangent and to the normal vector programmed with `LEAD` and `TILT` at the end of the block, whereas, during the block, the orientation does not follow the path tangent (previous behavior).

1: Reference of the tool orientation to the path tangent and to the surface normal vector programmed with `LEAD/TILT` is maintained **throughout the entire block**.

Meaning of **decades**	Interpretation of the turning angle TILT

0: `LEAD` = rotation about the direction perpendicular to the tangent and normal vector (forward angle)

`TILT` = rotation of orientation around the **normal vector**

1: `LEAD` = rotation about the direction perpendicular to the tangent and normal vector (forward angle)

`TILT` = rotation of orientation around the **direction of path tangent** (sideways angle)

Meaning of **centuries** Retracting movement with re-orientation

0: There is no retracting movement

There is a retracting movement in the tool coordinate system, i.e. the direction programmed by the retracting vector is interpreted in a coordinate system, which is specified in the following way:

1: **Current tool direction** (z coordinate) and orientation change (x coordinate)

2: **Active plane** (z coordinate is normal vector to the active plane) and orientation change (x coordinate)

Smoothing of the orientation jump ORIPATHS

Smoothing of the oreintation jump is done within the setting data SD42670 $SC_ORIPATH_SMOOTH_DIST (path distance to smoothing orientation) of the specified path. The programmed reference of the orientation to the path tangent and normal vector is then no longer maintained within this distance. If this distance is set too small, the path velocity may have to be reduced significantly.

A velocity jump of the orientation axes can also be smoothed. In the case where the orientation path does not perform a jump, but whose first derivation is not smooth, the resulting velocity jump can be smoothed. The setting data SD42672 $SC_ORIPATH_SMOOTH_TOL > 0 (tolerance for smoothing the orientation) is used to specify how much the orientation may deviate from the "tangential" alignment. This orientation smoothing is only performed if G code `ORIPATHS` ORIPATHS is active and setting data SD42672 SC_ORIPATH_SMOOTH_TOL > 0.

Insertion of intermediate blocks for the smoothing of the orientation path

If the following setting data is set, a separate intermediate block is inserted for the smoothing of the orientation path:

SD42670 $SC_ORIPATH_SMOOTH_DIST = 0.0 (path distance for smoothing the orientation)

This means that the path motion then stops at the corner of the contour and only then the jump in the tool orientation is executed. The orientation change is then only performed with with continuous acceleration when `ORIPATHS` is active. Otherwise the orientation is transferred from the start orientation to the end orientation by means of linear large circle interpolation.

Execute tool retracting movement

A tool retracting movement can be performed during this re-orientation. The direction and path length of the retracting movement is programmed via the vector using the components A8=x, B8=y and C8=z. If the length of this vector is exactly zero, no retracting movement is executed.

In which coordinate system the tool retracting vector is interpreted, depends on the value of the following machine data:

MD21094 $MC_ORIPATH_MODE (setting for path relative orientation `ORIPATH`)

This specifies in which coordinate system the retracting vector is interpreted.

1. Tool coordinate system: z coordinate defined by current tool direction.
2. Workpiece coordinate system: z coordinate defined by active plane.

Normally the retracting movement is performed simultaneously to the orientation change. A factor can be programmed with the identifier ORIPLF = r , which defines a "safety clearance". In this way, tool orientation only changes when the tool has retracted by **r * retraction path**. The programmed retraction factor must be in the interval 0 = r 1, in order to avoid alarm14126.

Path relative interpolation of the rotation ORIROTC

With **6-axis transformations**, in addition to the complete interpolation of the tool orientation relative to the path and the rotation of the tool, there is also the option that **only the rotation** of the tool relative to the path tangent is interpolated. The tool orientation can be programmed and interpolated independently of this. This is activate by the G-code ORIROTC in the 54th G-code group. Tool orientation direction can be programmed as usual with direction vectors, Euler or RPY angle. Their interpolation method can be specified as usual with the G codes ORIVECT, ORIAXES, ORICONxx, ORICURVE , see Chapter "Rotation of the Orientation Vector".

1.9.5 Programming of orientation polynominals

Functionality

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-code G01
- Polynomial interpolation with G-code POLY
- Circular interpolation with G-code G02, G03 or CIP
- Involute interpolation with G-code INVCW or INVCCW

This enables a number of polynomials to be programmed for one contour **at the same time**.

Note

For further 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], please see:

References: /PGA/ Programming Manual, Work Preparation

Two different types of orientation polynomials are defined:

1. Polynomials for angles with reference to the plane defined by the start and end orientation (type 1 orientation polynomials).
2. Polynomials for curves in space on a reference point on the tool (type 2 orientation polynomials).

Type 1 polynomials

Orientation polynomials of type 1 are polynomials **for angles**

PO[PHI]:	in the plane between start and end orientation
PO[PSI]:	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

For **path relativeorientation interpolation** relative to the path `ORIPATH` or `ORIPATHS` , the additional rotation can be programmed with the angle THETA=<...>. Polynomials up to the 5th degree can also be programmed with PO[THT]=(...) for this angle of rotation.

The three 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. Is only possible if tool orientation has a total of 3 degrees of freedom, see "Extension of the generic transformation to 6 axes".

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. Polynomials are programmed with the PHI and PSI angles:

PO[PHI] = (a2, a3, a4, a5)	Polynomial for the LEAD angle
PO[PSI] = (b2, b3, b4, b5)	Polynomial for the TILTangle

Polynomials can be programmed up to the 5th degree for both angles. The angle values at the block end are programmed with the NC addresses LEAD= <...> bzw. 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.

A polynomial up to the 5th degree can also be programmed with PO[THT]=(c2, c3, c4, c5) for the offset angle.

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, please see:

References: /PGA/ Programming Manual, Work Preparation, 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 boundary 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.

Interrupts

If an illegal polynomial is programmed, the following alarms are generated:

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.

1.9.6 Tool orientation with 3-/4-/5-axis transformations

Tool direction can be read with the following system variables:

$P_TOOLO[n]	Tool orientation active in the **interpreter** cannot be applied in sychronous actions
$AC_TOOLO_ACT[n]	Active setpoint orientation in the **interpolator**
$AC_TOOLO_END[n]	End orientation of the current block
$AC_TOOLO_DIFF	Residual angle of tool orientationin the active block
$VC_TOOLO[n]	Actual value orientation direction
$VC_TOOLO_DIFF	Angle between actual and setpoint orientation
$VC_TOOLO_STAT	Status of calculations of actual value orientation

1.9.7 Orientation vectors with 6-axis transformation

With 6-axis transformations, the complete orientation is described by two vectors that are perpendicular to one another.

- The **first** vector points in the direction of the tool (see above), while
- the **second** is in the plane perpendicular to this and describes rotations of the tool around itself.

Both vectors can be read via system variables and also via the OPI interface.

The reading of the direction of rotation vector with the following system variables is only meaningful for a 6-axis transformation.

$P_TOOLROT[n]	Rotation direction vector active in the **interpreter** not applicable in synchronous actions
$AC_TOOLR_ACT[n]	Active rotation direction vector in the **interpolator**
$AC_TOOLR_END[n]	End orientation of the active block
$AC_TOOLR_DIFF	Residual angle of the direction of rotation vector in the active block in degrees
$VC_TOOLR[n]	Actual value of the rotation direction vector
$VC_TOOLR_DIF	Angle between actual value and setpoint of the direction of rotation vector in degrees
$VC_TOOLR_STAT	Calculation status of the actual value of the direction of rotation vector

References: /PGA/ LHB System variables

For further information about the programming of polynomials for axis movements with orientation vectors, see Chapter "Orientation vectors".

1.10 Orientation axes

Direction

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 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, please see Chapter "Example for Orientation Axes".

1.10.1 JOG mode

It is not possible to traverse **orientation axes** in JOG mode until the following conditions are fulfilled:

- 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`command)

Axis traversal using traverse keys

When using the traverse keys to move an axis continuously (momentary-trigger mode) or incrementally, it must be noted that only **one** orientation axis can be moved at a time.

If more than one orientation axis is moved, alarm 20062 "Channel 1 axis 2 already active" is generated.

Traversal using handwheels

More than one orientation axis can be moved simultaneously via the handwheels.

Feedrate in JOG

When orientation axes are traversed manually, the channel-specific feedrate override switch or the rapid traverse override switch in rapid traverse override is applied.

Until now, velocities for traversal in JOG mode have always been derived from the machine axis velocities. However, geometry and orientation axes are not always assigned directly to a machine axis.

For this reason, new machine data have been introduced for geometry and orientation axes, allowing separate velocities to be programmed for these axis types:

- MD21150 $MC_JOG_VELO_RAPID_ORI[n]

  (conventional fast traverse for ORI axes)
- MD21155 $MC_JOG_VELO_ORI[n]

  (conventional ORI axis speed)
- MD21160 $MC_JOG_VELO_RAPID_GEO[n]

  (conventional fast traverse for GEO axes)
- MD21165 $MC_JOG_VELO_GEO[n]

  (conventional GEO axis speed)

Appropriate speed values for the axes must be programmed in these data.

Acceleration

Acceleration for the orientation axes can be set by means of the following machine data:

MD21170 $MC_ACCEL_ORI[n] (acceleration for orientation axes)

1.10.2 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 identifiers **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-code) 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: /PG/ Programming Manual, Basics

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 an example of orientation axes for kinematics with 6 or 5 transformed axes, please see Chapter "Example of Orientation Axes".

Interpolation type

The following machine data is used to specify which interpolation type is used:

MD21104 $MC_ORI_IPO_WITH_G_CODE (G-code for orientation interpolation):

- ORIMKS or ORIWKS (for description, see Chapter "Workpioece Orientation")
- G-code 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.

Value range

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).

1.10.3 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:

References: /PGA/ Programming Manual, Work Preparation, Chapter "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 point 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 tool holder with additive offset

On an orientable tool holder, 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 tool holder 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 tool holders, please see:

References: /FB1/ Function Manual, Basic Machine, Tool Offset (W1)

1.10.4 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.

1.10.5 Modulo display of orientation axes

Function

The positions of orientation axes can be displayed for BCS and WCS display in a settable modulo area. Whether the concerned machine axes are linear or rotary is not relevant in this context, i.e. this display option can be enabled even for normal generic 5-/6-axis transformation.

Preconditions

- Orientation axes must be available. This is the case if an orientation transformation is active (e.g. generic 5-/6-axis transformation).
- The following MD must be set additionally for OEM transformations:
  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[0..2]

Parameter assignment

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 becomes effective with NewConfig.
- 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

1.11 Orientation vectors

1.11.1 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:

References: /PGA/ Programming Manual, Work Preparation

A block with POLY is used to program polynomial interpolation. Whether the programmed polynomials are then interpolated as polynomial, depends on whether the G-code POLY is active or not.

- The G-code ist **not active**: The programmed axis end points are traversed linearly.
- The G-code is **active**: The programmed polynomials are interpolated as polynomials.

MD10674

If machine data MD10674 $MN_PO_WITHOUT_POLY = **FALSE**
(polynomial programming without G-function POLY programmable) it can be specified, 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-code POLY.

By default MD 10674 is: 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 Chapter "Programming of Orientation Polynomials".

POLYPATH:

In addition to the modal G function POLY, the predefined subprogram POLYPATH(argument) can be used to activate polynomial interpolation selectively for different 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 identifiers.
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-code 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 the start and end vectors (large circle interpolation, see Figure Rotation of the Orientation Vector in the Plane between Start and End Vector). The orientation is interpolated exactly as in Example 1.

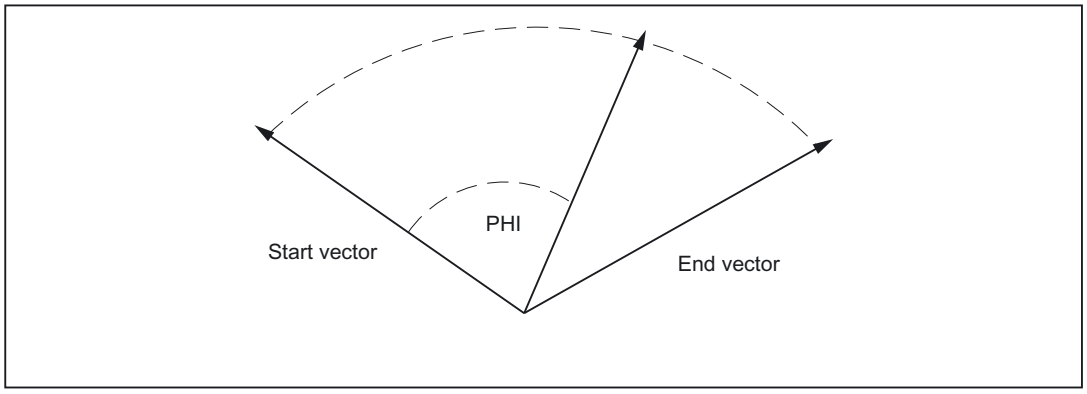


Figure 1-21 Rotation of the orientation vector in the plane between start and end vector

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.

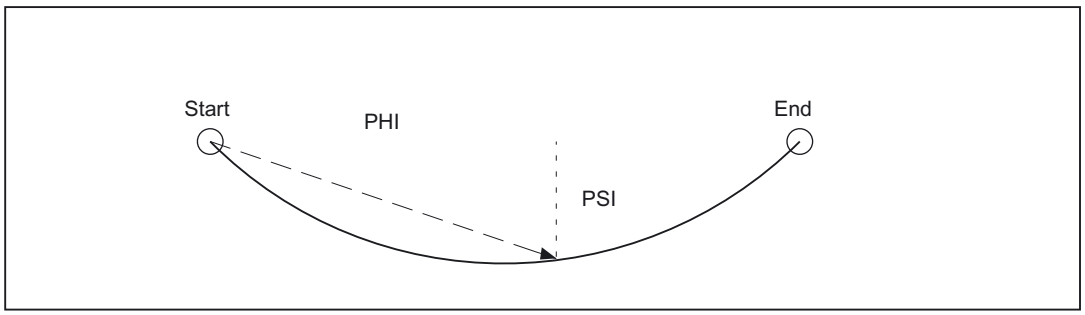


Figure 1-22 Movement of the orientation vector in plan view

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

5th degree polynomials are the maximum possible for programming 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.

Points to note

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:

References: /PGA/ Programming Manual, Work Preparation

Constraints

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

1.11.2 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 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-codes `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 5th degree maximum, in the format:

  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 angleand the constant and linear coefficient d_n of the polynomial cannot be programmed directly.

The linear coefficient d_n is defined by means of the end angle q_e in degrees.

The end angle q_e is derived from programming of the rotation vector.

The start angle q_s is derived 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 starting angle of the polynomial.

The rotation vector is always perpendicular to the current tool orientation and forms the angle THETAin conjunction with the basic rotation vector.

Note

During machine configuration, the direction in which the rotation vector points at a specific angle of rotation can be defined, when the tool is in the basic orientation.

Formula

In general, the angle of rotation is interpolated with a 5th degree polynomial:

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

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

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

Interpolation of the rotation vector

The programmed rotation vector can be interpolated in the following way, using modal G-codes:

- 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):

  The angle of rotation THETA is interpreted relative to the plane defined by the start and end orientation.

- ORIROTT (**ori**entation **rot**ation **t**angential):

  The angle of rotation THETA is interpreted relative to the change in orientation. That means the rotation vector interpolation is tangential to the change in orientation for THETA=0.

  This is different to ORIROTR, only if the change in orientation does not take place in one plane. This is the case if at least one polynomial was programmed for the "tilt angle" PSI for the orientation. An additional angle of rotation THETA can then be used to interpolate the rotation vector such that it always produces a specific angle referred to the change in orientation.

Activation of rotation

A rotation of the orientation vector is programmed with the identifier THETA. The following options are available for programming:

THETA=<value>	Programming of an angle of rotation at the end of the block.
THETA = q_e	Programmed angles q_e can be interpreted as absolute (G90 is active) or as relative (G91 is active incremental).
THETA = AC(...)	Switch over to absolute dimensions per block
THETA = IC(...)	Switch over to incremental dimensions per block
PO[THT] = (...)	Programming of a polynomial for rotation angle THETA.

The angle THETA is programmed in degrees.

Interpolation of the rotation vector is defined by modal G-codes:

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 tangential rotationvector of the orientation vector to the orientation change
ORIROTC	Angle of rotation relative to the change of the tangential rotationvector of the orientation vector to the path tangent
PL	Length of the parameter interval where polynomials are defined. The interval always starts at 0. If no PL has been programmed, PL = 1 will be applied.

These G-codes define the reference direction of the angle of rotation. The meaning of the programmed angle of rotation changes accordingly.

Boundary 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 angles via A2, B2, C2
3. RPY angles 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).

1.11.3 Extended interpolation of orientation axes

Functionality

To execute a change in orientation along the peripheral surface of a cone located in space, it is necessary to perform an extended interpolation of the orientation vector. The vector around which the tool orientation is to be rotated must be known. The start and end orientation must also be specified. The start orientation is given by the previous block and the en orientation must either be programmed or defined by other conditions.

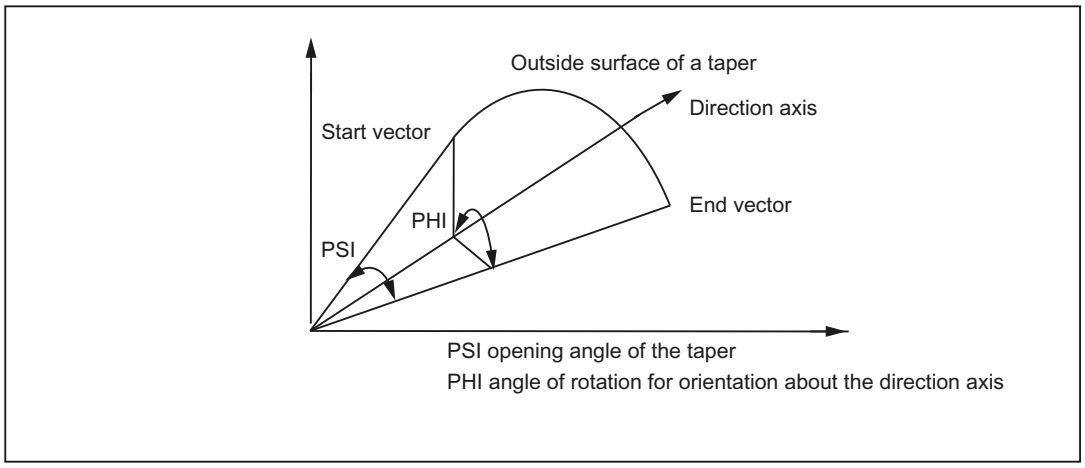


Figure 1-23 Change in orientation of the peripheral surface of a cone located in space

Required definitions

Generally, the following data are 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 counterclockwise 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-code 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-codes 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 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) 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.

Boundary 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-code of group 51 through extended interpolation of the orientation vector, using the following commands:

ORIPLANE	Interpolation in a plane with specification of the end orientation (same as ORIVECT)
ORICONCW	Interpolation on a peripheral surface of a taper in clockwise directionwith specification of the end orientation and taper direction or opening angle of the cone.
ORICONCCW	Interpolation on the peripheral surface of a cone in the counterclockwise direction. Specification of the end orientation andt cone direction or opening angle of the taper.
ORICONIO	Interpolation on a peripheral surface of a cone with specification of end orientation and an intermediate orientation.
ORICONTO	Interpolation on a peripheral surface of a cone with tangential transition and specification of end orientation.
ORICURVE	Interpolation of orientation with specification of motion of two contact points of the tool.
ORIPATH	Tool orientation in relation to the path
ORIPATHS	Tool orientation in relation to the path, when, for example, a kink in the orientation path, e.g. at a corner in the contour, is to be smoothed, see Chapter "Orientation Relative to the Path".

Examples

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

```
...
N10 G1 X0 Y0 F5000
N20 TRAORI              ; Orientation transformation active.
N30 ORIVECT             ; interpolate WZ orientation as a 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 is rotated 45 degrees
                        ; at the end of the block, the orientation (0, 1/√2),
                        ; 1/√2) is reached.
N70 ORICONCW            ; Orientation vector is interpolated in the clockwise
                        ; direction on the outside of the cone with the direction;
N80 A6=0 B6=0 C6=1 A3=1 B3=0 C3=1; (0, 0, 1) to the orientation
                        ; (1 / √2, 0, 1 / √2)
                        ; interpolated clockwise, the rotation angle is 270 degrees.
                        ; WZ orientation traverses a full
N90 A6=0 B6=0 C6=1      ; Rotation on the same outside surface of the cone
...
```

1.12 Online tool length offset

Functionality

Effective tool length can be changed in real time so that the length changes are also considered for changes in orientation of the tool. System variable $AA_TOFF[ ] applies 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 geometry identifiers are used as index. 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

Online tool length offset is an option and must be enabled beforehand. This function is only practical in conjunction with an active orientation transformation or an active orientable toolholder.

References: /FB/ Function Manual, Basic Machine; Tool Offset; Orientable Tool Holders (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 compensation
MD21190 $MC_TOFF_MODE	The contents of $AA_TOFF[ ] are traversed as an absolute value or integrated
MD21194 $MC_TOFF_VELO (speed online tool offset)	Speed of online tool length offset
MD21194 $MC_TOFF_ACCEL (acceleration online tool offset)	Acceleration of online tool length offset
SD42970 $SC_TOFF_LIMIT (upper limit of offset value $AA_TOFF)	Upper limit of tool length offset value

With the acceleration margin, 20% is reserved for the overlaid movement of online tool length offset, which can be changed via the following machine data:

MD20610 $MC_ADD_MOVE_ACCEL_RESERVE(acceleration margin for overlaid movements)

Activation

The TOFFON instruction can be used to activate online tool length offset from the parts 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:

References: /PGA/ Chapter "Transformations"

As long as online tool length offset is active, the VDI signal on the NCK → 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 JOG mode to PTP as long as a tool length compensation 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 variables

In the case of online tool length offset, the following system variables are available to the user:

System variables	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.

References: /PGA1/ LHB 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 tool holders".

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.

1.13 Examples

1.13.1 Example of a 5-axis transformation

```
CHANDATA(1)

$MA_IS_ROT_AX[AX5] = TRUE
$MA_SPIND_ASSIGN_TO_MACHAX[AX5] = 0
$MA_ROT_IS_MODULO[AX5]=0



;------------------------------------------------------------------------------------------------
; general 5-axis transformation
;
; kinematics:                          1st rotary axis is parallel to Z
;                                      2nd 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:

```
; 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
```

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

1.13.2.1 Example of a 3-axis transformation

Example: For the machine schematically presented in Figure "Schematic Presentation of a 3-axis Transformation" (see Chapter "3- and 4-axis Transformation", Short Description), the 3-axis transformation can be projected in the following way:

```
$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
```

1.13.2.2 Example of a 4-axis transformation

Example: For the machine schematically presented in Figure "Schematic 4-axis Transformation with movable Workpiece" (see Chapter "3- and 4-axis Transformation", Short Description), the 4-axis transformation can be projected in the following way:

```
$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
```

1.13.3 Example of a universal milling head

General information

The following two subsections show the main steps which need to be taken in order to activate a transformation for the universal milling head.

Machine data

; machine kinematics CA' with tool orientation in zero position in the z direction

$MC_TRAFO_TYPE_1 = 148

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2

$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3

; angle of second rotary axis

$MC_TRAFO5_NUTATOR_AX_ANGLE_1 = 45

Program

```
; Definition of tool T1
$TC_DP1[1,1] = 120;                          Type
$TC_DP2[1,1] = 0;
$TC_DP3[1,1] = 20;                           Z length offset vector G17
$TC_DP4[1,1] = 8.;                           Y
$TC_DP5[1,1] = 5.;                           X

TRAORI(1);                                   Activation of transformation
ORIMKS;                                      Orientation reference to MCS
G0 X1 Y0 Z0 A0 B0 F20000 G90 G64 T1 D1 G17
; Programming of direction vector
G1 G90
a3 = 0 b3 = 1 c3 = 0
; Programming in Euler angles
G1 G90
a2 = 0 b2 = 0 X0

; Programming of rotary axis motion
G1 X10 Y5 Z20 A90 C90


m30
```

References: /PA/ Programming Manual, Fundamentals

1.13.4 Example for orientation axes

Example 1:

3 orientation axes for the 1st orientation transformation for kinematics with 6 transformed axes. Rotation must be done in the following sequence:

- firstly about the Z axis.
- then about the Y axis and
- finally about the Z axis again.

The tool vector must point in the X direction.

```
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 axis
$MC_TRAFO5_ORIAX_ASSIGN_TAB_1[2]=6        ; Channel index of third orientation axis
$MC_ORIAX_TURN_TAB_1[0]=3                 ; Z direction
$MC_ORIAX_TURN_TAB_1[1]=2                 ; Y direction
$MC_ORIAX_TURN_TAB_1[2]=3                 ; Z direction

CHANDATA(1)
M17
```

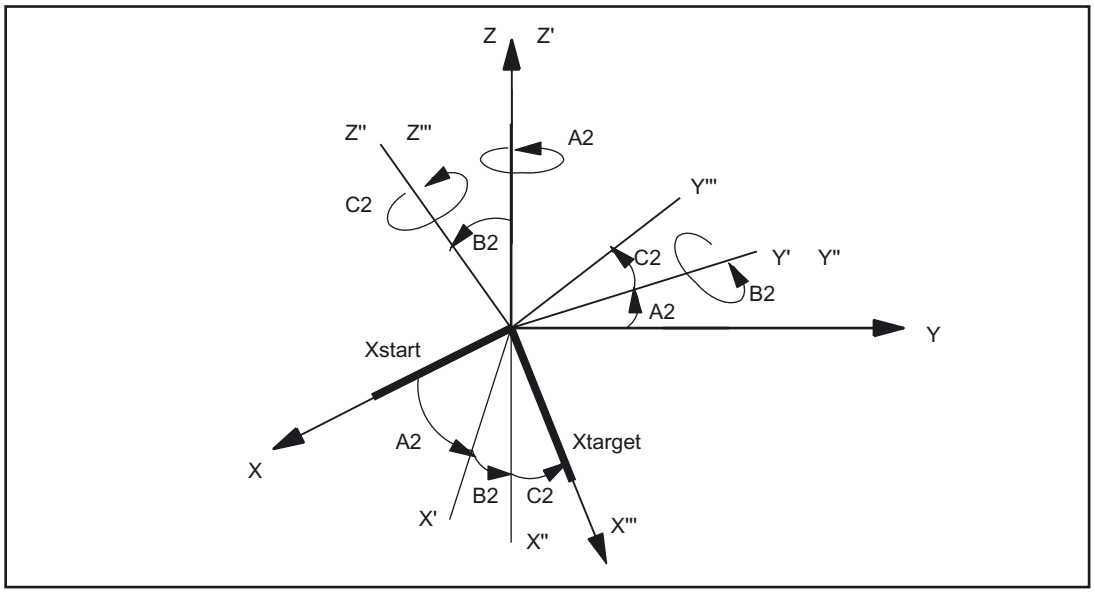


Figure 1-24 3 orientation axes for the 1st orientation transformation for kinematics with 6 transformed axes

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.

```
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

CHANDATA(1)
M17
```

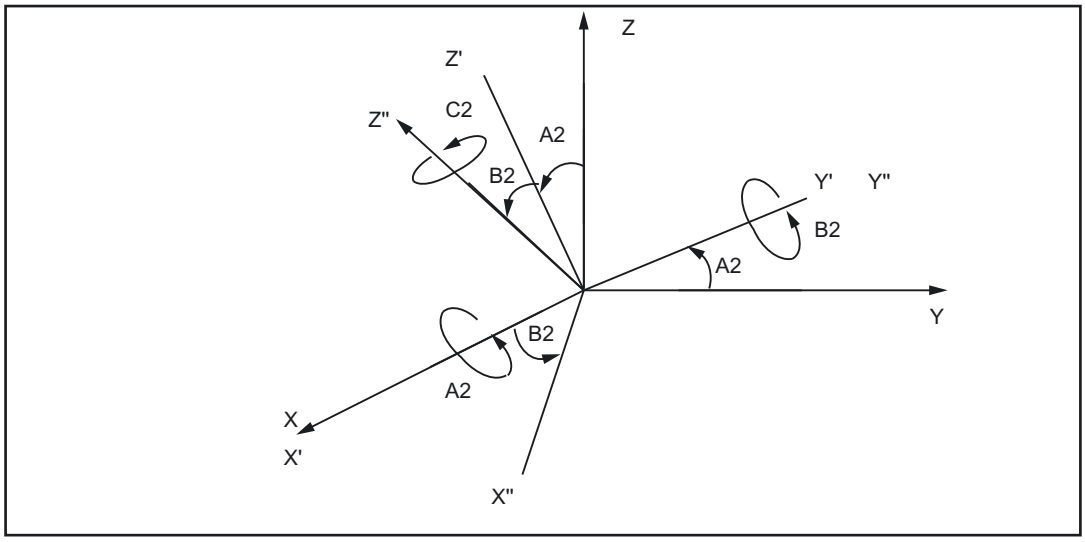


Figure 1-25 3 orientation axes for the 2nd orientation transformation for kinematics with 5 transformed axes

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: /PGA/ Programming Manual, Work Preparation

1.13.5 Examples for orientation vectors

1.13.5.1 Example for polynomial interpretation of orientation vectors

Orientation vector in Z-X plane

The orientation vector is programmed directly in the examples below. The resulting movements of the rotary axes depend on the particular kinematics of the machine.

```
N10 TRAORI
N20 POLY                          ; Polynomial interpolation is possible.
N30 A3=0 B3=0 C3=1                ; Tool vector 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 angle 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}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).

```
N10 TRAORI
N20 POLY                    ; Polynomial interpolation is possible.
N30 A3=0 B3=0 C3=1          ; Tool vector in +Z direction (start vector)
N40 A3=1 B3=0 C3=0          PO[PHI]=(10)        ; in +X direction (end vector)
                            PO[PSI]=(10)
```

1.13.5.2 Example of rotations of orientation vector

Rotations with angle of rotation THETA

In the following example, the angle of rotation is interpolated in linear fashion from starting value 0 degrees to end value 90 degrees. The angle of rotation changes according to a parabola or a rotation can be executed without a change in orientation. tool orientation is rotated from the Y direction to the X direction.

```
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
                                    ; of 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 of orientation.
N80 A3=1 B3=0 C3=0 THETA=30         ; Rotation vector in angle of
                                    ; 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.

1.13.6 Example of generic 5-axis transformation

The following example is based on a machine with rotatable tool on which the first rotary axis is a C axis and the second a B axis (CB kinematics, see Figure). The basic orientation defined in the machine data is the bisecting line between the X and Z axes.

Relevant machine data are 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              ; 1st 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              ; 2nd 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:

```
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 1st rotary axis
                                   ; in X direction
N60      $TC_CARR11[1] = 1         ; Component of 2nd rotary axis in 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 transformationorientation
                                   ; from machine data
N120     C3=1                      ; Orientation parallel to Z
                                   ; set → B-45 C0
N130     T1 D1                     ; Basic orientation now is parallel to Z
N140     C3=1                      ; Orientation parallel to Z
                                   ; set → B0 C0
N150     G19                       ; Basic orientation now is parallel to X
N160     C3=1                      ; Orientation parallel to Z
                                   ; set → B-90 C0
N170     G17 TCARR=1 TCOABS        ; Basic orientation now is angle-
                                   ; bisecting 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 on basic orientation
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                      ; traverse 7 mm in new Z direction
                                      ; → X2 Y3 Z6
N240      C3=1                        ; Orientation parallel to the
                                      ; new Z-axis → B0 C0
N250      M30
```

1.13.6.1 Example of a generic 6-axis transformation

Activation of a 6-axis transformation with subsequent orientation changes and traversing:

```
N10       A0 B0 X0 Y0 Z0              ;
N20       TRAORI(1, ,,, 0,0,0,        ; Passing on the orientation vector
          0,1,0)                      ; and the orientation normalized vector,
                                      ; Transformation selection
N30       T1 D1 X10 Y20 Z30 A3=0.5    ; Change of orientation, rotation
          C3=1 BN3=1 ORIPLANE ORIWKS  ; traverse motion
N40       B3=0.5 C3=1 AN3=-1          ; Rotation programmed,
                                      ; Orientation constant
N50       M30
```

A tool, of which the orientation differs from the default, is defined in the following example. With G17, the orientation vector is in the X-Z plane and its inclination to the Z axis is 26.565 degrees because of tan(26.565) = 0.5 = $TC_DPV3[2,2] / $TC_DPV5[2,2].

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.

An orthogonalization is therfore **not necessary**, and therefore the programmed orientation normal vector is not modified.

```
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
N130       $TC_DPV3[2,2] = 1         ; X component tool cutting edge orientation.
N140       $TC_DPV4[2,2] = 0         ; Y component tool cutting edge orientation
N150       $TC_DPV5[2,2]= 0.5        ; Z component tool cutting edge orientation
N160       $TC_DPVN3[2,2]= 0         : X component orientation normal vector
N170       $TC_DPVN4[2,2]= 1         ; Y component orientation normal vector
N180       $TC_DPVN5[2,2]= 0         ; Z component orientation normal vector
```

```
N200       TRAORI()                ; pass on basic orientation
N210       A3=5 C3=10 BN3=1        ; Get rotary axes in initial state
N220       C3=1                    ; Orientation in Z direction → tool
                                   ; rotates 26.565 degrees
N230       THETA=IC(90)            ; Rotate orientation normal vector
                                   ; incrementally by 90 degrees
                                   ; Vector points in negative
                                   ; X direction
N240       M30
```

1.13.6.2 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                    ;   1st rotary axis shows in Z direction
N40 $MC_TRAFO5_AXIS1_2[0] = 1                    ;   2nd rotary axis shows in X direction
N50 $MC_TRAFO5_AXIS1_3[2] = 1                    ;   3rd rotary axis shows in Z direction
N60 $MC_TRAFO7_EXT_AXIS1_1[0] = 1                ;   7th 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               ;   1st quadrant
N130 G2 y0 z–50 c3=–1 e=DC(180) CR=50            ;   2nd quadrant
N140 G2 y–50 z0 b3=–1 e=DC(270) CR=50            ;   3rd quadrant
N150 G2 y0 z50 c3=1 e=DC(0) CR=50                ;   4th 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.

1.13.6.3 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.

1.13.7 Example: Compression of an orientation

In the example program below, a circle approached 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 in an unsteady way, the compressor function generates a smooth motion of the orientation.

```
Programming                            Comment
DEF INT NUMBER=60
DEF REAL RADIUS=20
DEF INT COUNTER
DEF REAL ANGLE
N10 G1 X0 Y0 F5000 G64

$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
```

Programming	Comment
	; 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	

1.14 Data lists

1.14.1 Machine data

1.14.1.1 General machine data

Number	Identifier: $MN_	Description
10620	EULER_ANGLE_NAME_TAB	Name of Euler angles or names of orientation axes
10630	NORMAL_VECTOR_NAME_TAB	Name of normal vectors
10640	DIR_VECTOR_NAME_TAB	Name of direction vectors
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

1.14.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups
20152	GCODE_RESET_MODE[n]	Setting after RESET/end of part program
20482	COMPRESS_MODE	Mode of the compressor
20621	HANDWH_ORIAX_MAX_INCR_SIZE	Limitation of handwheel increment
20623	HANDWH_ORIAX_MAX_INCR_VSIZE	Orientation velocity overlay
21094	ORIPATH_MODE	Setting for path relative orientation
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21102	ORI_DEF_WITH_G_CODE	Definition of orientation angles A2, B2, C2
21104	ORI_IPO_WITH_G_CODE	Definition of interpolation type for orientation
21106	CART_JOG_SYSTEM	Coordinate system for Cartesian JOG
21108	POLE_ORI_MODE	Behavior during large circle interpolation at pole position
21120	ORIAX_TURN_TAB_1[n]	Assignment of rotation of orientation axes about the reference axes, definition 1 [n = 0..2]
21130	ORIAX_TURN_TAB_2[n]	Assignment of rotation of orientation axes about the reference axes, definition 2 [n = 0..2]

Number	Identifier: $MC_	Description
21132	ORI_DISP_IS_MODULO[n]	Modulo display of the orientation axes positions [n = 0..2]
21134	ORI_DISP_MODULO_RANGE	Size of the module range for the display of the orientation axes
21136	ORI_DISP_MODULO_RANGE_START	Starting position of the module range for the display of the orientation axes
21150	JOG_VELO_RAPID_ORI[n]	Rapid traverse in jog mode for orientation axes in the channel [n = 0..2]
21155	JOG_VELO_ORI[n]	Orientation axis velocity in jog mode [n = 0..2]
21160	JOG_VELO_RAPID_GEO[n]	Rapid traverse in jog mode for geometry axes in the channel [n = 0..2]
21165	JOG_VELO_GEO[n]	Geometry axis velocity in jog mode [n = 0..2]
21170	ACCEL_ORI[n]	Acceleration for orientation axes [n = 0..2]
21180	ROT_AX_SWL_CHECK_MODE	Check software limits for orientation axes
21186	TOCARR_ROT_OFFSET_FROM_FR	Offset of TOCARR rotary axes
21190	TOFF_MODE	Operation of online offset in tool direction
21194	TOFF_VELO	Speed of online offset in tool direction
21196	TOFF_ACCEL	Acceleration of online offset in tool direction
24100	TRAFO_TYPE_1	Definition of transformation 1 in channel
24110	TRAFO_AXES_IN_1[n]	Axis assignment for transformation 1 [axis index]
24120	TRAFO_GEOAX_ASSIGN_TAB_1[n]	Assignment geometry axis to channel axis for transformation 1 [geometry no.]
24200	TRAFO_TYPE_2	Definition of transformation 2 in channel
24210	TRAFO_AXES_IN_2[n]	Axis assignment for transformation 2 [axis index]
24220	TRAFO_GEOAX_ASSIGN_TAB_2[n]	Assignment geometry axis to channel axis for transformation 2 [geometry no.]
24300	TRAFO_TYPE_3	Definition of transformation 3 in channel
24310	TRAFO_AXES_IN_3[n]	Axis assignment for transformation 3 [axis index]
24320	TRAFO_GEOAX_ASSIGN_TAB_3[n]	Assignment geometry axis to channel axis for transformation 3 [geometry no.]
24400	TRAFO_TYPE_4	Definition of transformation 4 in channel
24410	TRAFO_AXES_IN_4[n]	Axis assignment for transformation 4 [axis index]
24420	TRAFO_GEOAX_ASSIGN_TAB_4[n]	Assignment geometry axis to channel axis for transformation 4 [geometry no.]
24430	TRAFO_TYPE_5	Definition of transformation 5 in channel
24432	TRAFO_AXES_IN_5[n]	Axis assignment for transformation 5 [axis index]
24434	TRAFO_GEOAX_ASSIGN_TAB_5[n]	Assignment geometry axis to channel axis for transformation 5 [geometry no.]
24440	TRAFO_TYPE_6	Definition of transformation 6 in channel
24442	TRAFO_AXES_IN_6[n]	Axis assignment for transformation 6 [axis index]
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]

Number	Identifier: $MC_	Description
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]
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

Number	Identifier: $MC_	Description
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
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

Number	Identifier: $MC_	Description
25395	TRAFO7_EXT_AXIS1_4	Direction of the 4th external rotary axis
28580	MM_ORIPATH_CONFIG	Configuration for path relative orientation ORIPATH

1.14.2 Setting data

1.14.2.1 General setting data

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

1.14.2.2 Channel-specific setting data

Number	Identifier: $SC_	Description
42475	COMPRESS_CONTOUR_TOL	Max. contour deviation for compressor
42476	COMPRESS_ORI_TOL	Max. angular displacement of tool orientation for the compressor
42477	COMPRESS_ORI_ROT_TOL	Max. angular displacement of tool rotation 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	Smoothing tolerance of the orientation
42970	TOFF_LIMIT	Upper limit for offset value $AA_TOFF

1.14.3 Signals

1.14.3.1 Signals from channel

DB number	Byte.Bit	Description
21, ...	29.4	Activate PTP traversal
21, ...	33.6	Transformation active
21, ...	232	Number of active G function of G function group 25
21, ...	317.6	PTP traversal active
21, ...	318.2	Activate online tool length offset
21, ...	318.3	Activate offset motion

Gantry Axes (G1) 2

2.1 Brief description

Gantry axes

Gantry axes are mechanically grouped machine axes. Because of this mechanical coupling, gantry axes are always traversed in unison. The control occurs through the "gantry axes" function.

The machine axis that is directly traversed is called the leading axis. The machine axis that is traversed in conjunction with the leading axis is called the synchronized axis. The leading axis and the synchronized axes together form a gantry axis grouping.

The difference between the actual positions of the leading axis and the synchronized axes is monitored continuously. When the actual position values of the synchronized axes deviate too much from that of the master axis, the control automatically shuts down all axes in the gantry grouping to prevent any damage to the machines.

Application

Two feed drives are required to traverse the gantry on large gantry-type milling machines, i.e. one drive with its own position measuring system on each side. Owing to the mechanical forced coupling, both drives must be operated in absolute synchronism to prevent canting of mechanical components.

Configurations

In total, eight gantry connections can be defined.

One gantry grouping consists of a master axis and up to two synchronized axes.

2.2 "Gantry axes" function

Application

On large gantry-type milling machines, various axis units (e.g. gantry or crossbeam) are moved by a number of drives, which are mutually independent. Each drive has its own measuring system and thus constitutes a complete axis system. When these mechanically rigidly-coupled axes are traversed, both drives must be operated **in absolute synchronism** in order to prevent canting of mechanical components (resulting in power/torque transmission).

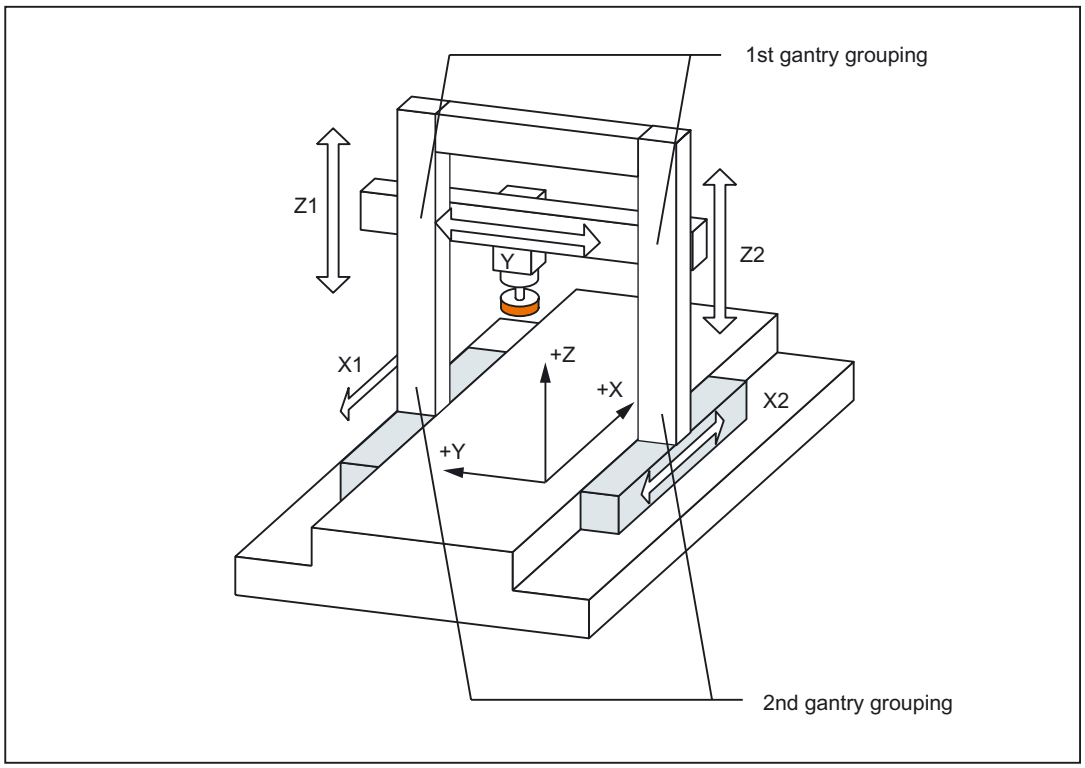


Figure 2-1 Example: Gantry-type milling machine with 2 gantry groupings

The purpose of the "gantry axes" function is to control and monitor machine axes which are rigidly coupled in this way.

Terms

The following terms are frequently used in this functional description:

Gantry axes

Gantry axes comprise at least one pair of axes, the leading axis and the synchronized axis. As these axes are mechanically coupled, they must always be traversed simultaneously by the NC. The difference between the actual positions of the axes is monitored continuously. The axes in a gantry grouping are either all linear axes or all rotary axes.

Gantry axis grouping

The gantry axis grouping defines which synchronized axes are controlled by which leading axis, based on machine data settings. The leading axis and synchronized axes cannot be traversed separately.

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. The axis name of the leading axis identifies all axes in the gantry axis grouping.

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".

Axis definition

MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition)

This axial machine data establishes the following:

- Which **gantry grouping** belongs to the axis.
- Whether the axis is defined as a **leading axis** or a **synchronized axis** within this grouping.

In total, up to eight gantry connections can be defined. Each gantry grouping consists of **one** leading axis and **one or two** synchronized axes.

Conditions for a gantry grouping

- A gantry grouping must not contain a spindle.
- A synchronized axis must not be a concurrent POS axis.
- A synchronized axis must not be addressed by a transformation.
- A synchronized axis must not be the slave axis in another type of axis coupling.
- A synchronized axis must not be defined as the leading axis in another axis grouping.

Note

Each axis in the gantry grouping must be set so that it can take over the function of the leading axis at any time, i.e. matching velocity, acceleration and dynamic response settings.

The control performs a plausibility check on the axis definition.

Components

The "gantry axes" function can be subdivided into the following functional units:

1. Setpoint generation of synchronized axis
2. Monitoring of actual value difference
3. Referencing and synchronization of leading axis and synchronized axes

Setpoint generation of synchronized axis

From the point of view of the operator, all coupled gantry axes are traversed as if only one axis, i.e. the leading axis, were programmed in the NC. Analogously, only the leading axis is programmed in the part program. The commands and traverse requests from the operator, the PLC interface or via the part program therefore apply in equal measure to all axes in the gantry grouping.

When the "gantry axes" function is active, the synchronized axis setpoint is generated directly from the setpoint of the leading axis in all operating modes.

Note

The dynamic control response of the leading and synchronized axes must be set identically.

Monitoring the actual value difference

The position actual values of the leading and synchronized axes are continuously compared with one another in the interpolation clock cycle and monitored to check that they are still within the permissible tolerance range.

Machine data can be set to specify the following limit values for alarm output and termination of the traversing motion for specific axes:

Gantry warning limit

The "warning limit exceeded" signal will be relayed to the operator when the position actual value difference of the gantry warning limit is exceeded:

MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

In addition, the following interface signal will be output to the PLC:

DB31, ... DBX101.3 (warning limit exceeded)

When below the warning limit, the message and interface signal will automatically be cancelled.

When MD37110 = 0 the message will be disabled.

Gantry trip limit

Alarm 10653 "error limit exceeded" will be signaled when the machine's maximum permissible actual position values are exceeded:

MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit)

In order to prevent any damage to the machines, the gantry axes will be immediately shut down via the break ramp.

The value of MD37120 is applied when the gantry grouping is synchronized.

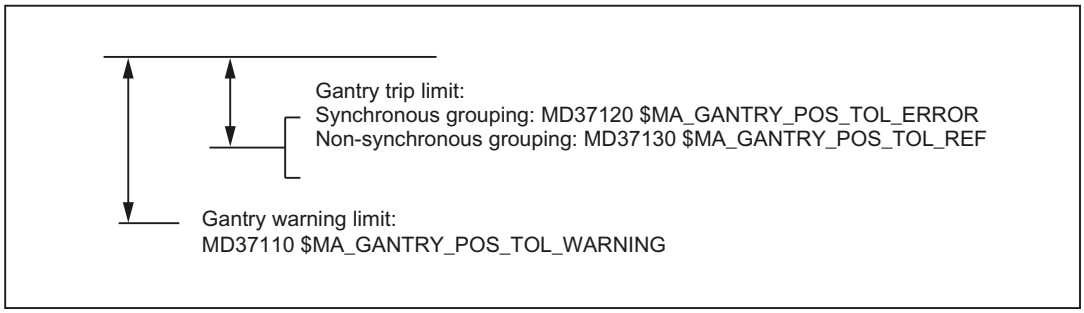
The alarm must be acknowledged with "RESET".

In addition, the following interface signal will be output to the PLC:

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

If the gantry axis grouping has not yet been synchronized, the limit value for the gantry trip limit is derived from the machine data:

MD37130 $MA_GANTRY_POS_TOL_REF (gantry trip limit)



Gantry shutdown limit exceeded is also activated if the gantry grouping is jammed (no servo enable, gantry grouping in "Hold" state).

The monitoring functions are deactivated while the grouping is operating in "Follow-up" mode.

Extended monitoring

An extended monitoring can be activated with the machine data:

MD37150 $MA_GANTRY_FUNCTION_MASK (gantry functions)

Effect:

An offset between the master and slave axis resulting from follow-up or Break_Up 1 is included in the synchronous operation difference and triggers an alarm when the corresponding warning or error limits are reached.

After power-up, extended monitoring is activated when the first reference point or synchronization (absolute encoder) is reached, irrespective of the axis status: e.g. RFG, NF, reference point approach, etc.

Referencing and synchronization of gantry axes

As the example "Gantry-type milling machine" shows, the forced coupling between gantry axes must remain intact in all operating modes as well as immediately after power ON. In cases where an incremental measuring system is being used for the leading or the synchronized axis, the reference point must be approached while maintaining the axis coupling immediately after machine power ON.

After every axis in the gantry grouping has approached its reference point, any misalignment that may exist between the axes must be eliminated (this is known as the gantry synchronization process). As soon as this occurs, the following interface signal will be signaled to the PLC:

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

The operational sequence for referencing and synchronizing gantry axes is described in detail under the topic "Referencing and synchronizing of gantry axes".

Closed-loop control

The dynamic control response settings of the coupled gantry axes must be identical (see " Starting up gantry axes (Page 157) "). This ensures that in normal operation, the leading and synchronized axes move in positional synchronism even during acceleration and braking.

Load effects are compensated by the appropriate drive of the leading or synchronized axis.

Disturbance characteristic

When a disturbance occurs which causes shutdown of one gantry axis owing, for example, to cancellation of the controller enabling signal (example: EMERGENCY STOP), all other coupled gantry axes are also shut down.

Separation of forced coupling

In certain situations (e.g. one gantry axis is no longer referenced owing to an encoder failure), it may be necessary to correct or reduce the misalignment between the gantry axes prior to referencing. To do this, it must be possible to traverse the leading or the synchronized axis **manually in the uncoupled state**.

The forced coupling between the gantry axes can be separated by means of the following MD setting and subsequent RESET:

MD37140 $MA_GANTRY_BREAK_UP = 1 (invalidate gantry grouping)

The gantry axes can then be traversed separately by hand; the monitoring of the warning and trip limits is not operative in this state.

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

2.3 Referencing and synchronization of gantry axes

2.3.1 Introduction

Misalignment after starting

Immediately after the machine is switched on, the leading and synchronized 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

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 distance-coded 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:

/FB1/ Function Manual, Basic Functions, Reference Point Approach (R1)

The appropriate synchronized 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 synchronized axes

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

References:

/FB1/ Function Manual, Basic Functions, Reference Point Approach (R1)

The dependency between the leading axis and synchronized axis is inverted in the control for this phase so that the leading axis now traverses in synchronism with the synchronized axis. IS "Referenced/synchronized" of the synchronized 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 synchronized axis is defined in the grouping, then this is also referenced in the way described above.

Section 3:

Gantry synchronization

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 synchronized 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.

  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 (Distance-coded 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 synchronized 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.

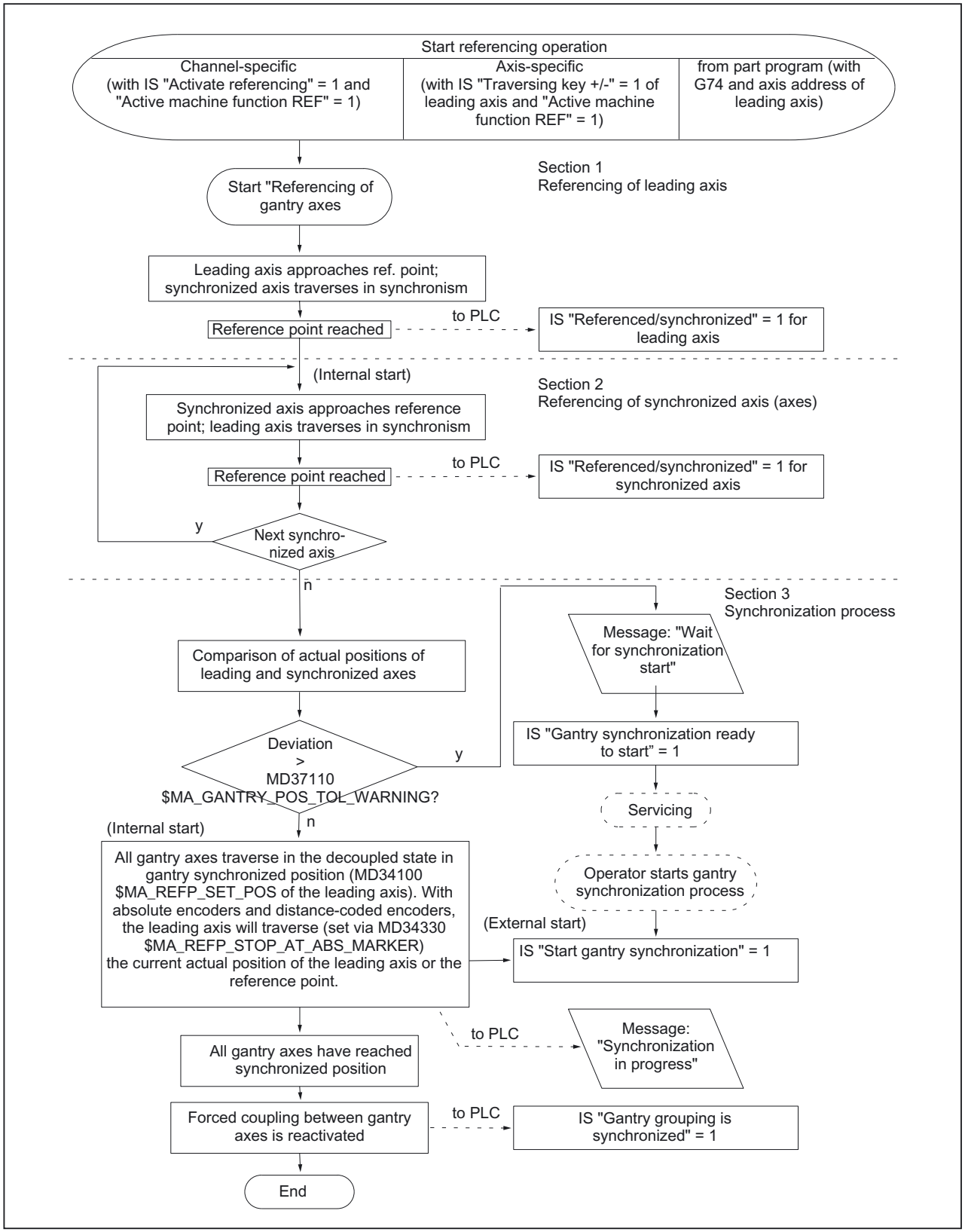


Figure 2-2 Flowchart for referencing and synchronization of gantry axes

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 (REF active machine function)

- DB31, ... DBX 101.5 = 0 (gantry grouping is synchronized)
- All grouping axes operate within the tolerance windows:

  DB31, ... DBX 101.4 ("Gantry synchronization ready to start")= 1

- 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", then instead of giving the following reference position:

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distance-coded system)

The current actual position of the leading axis is specified as the target position and approached in the uncoupled state.

Note

Automatic synchronization can be locked by the leading axis by means of the following interface signal:

DB31, ... DBX29.5 (automatic synchronization locking)

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 interface signal:

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

Loss of synchronization

The synchronization of the gantry grouping is lost (IS "Gantry grouping is synchronized" → 0) if:

- The gantry axes were in "Follow-up" mode
- The reference position of a gantry axis is lost, e.g. during "Parking" (no measuring system active)
- One gantry axis is re-referenced (IS "Referenced/Synchronized" changes to 0)
- The gantry grouping was invalidated:

  MD37140 $MA_GANTRY_BREAK_UP (invalidate gantry grouping)

In cases where the gantry grouping has lost synchronization during operation as the result of a disturbance, then the gantry synchronization process can be restarted directly by means of IS "Start gantry synchronization" (condition: IS "Referenced/Synchronized" = 1 for all axes in the gantry grouping). In this case, the synchronizing axes traverse the current actual position of the leading axis in the decoupled state.

If an emergency stop occurs while a gantry grouping is moving, and then rescinds, and both axes have drifted apart less than the zero speed tolerance of the slave axes, then the gantry grouping will automatically synchronize. It does not have to go in the BA REFP anymore. Automatic synchronization can be stopped with the interface signal DBxx.DBX29.5 from the master axis.

Reference point selection

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

MD34100 $MA_REFP_SET_POS (reference point value/destination point for distance-coded 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 leveling function of the slave axis can be defined via the machine data:

MD37150 $MA_GANTRY_FUNCTION_MASK Bit 1

Bit	Value	Meaning
1	0	The zero mark leveling function of the slave axis analogous to the machine data: MD34010 $MA_REFP_CAM_DIR_IS_MINUS
	1	The zero mark leveling function of the master axis is the same as the slave 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 (Distance-coded linear measuring system without destination point)

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

2.3.2 Automatic synchronization

Automatic synchronization can take place:

- in referencing mode (see Section "Detailed description", "Introduction")
- in other modes, as described in the following.

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 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)

2.3.3 Points to note

Two position measuring systems 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.

Channel-specific 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 are 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 distance-coded reference marks

In order that return traverses do not have to be made over large distances, it is possible to use a position measuring system with distance-coded reference marks as a sole or second measuring system 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:

/FB1/ Function Manual, Basic Functions, Reference Point Approach (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 distance-coded 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 (Distance-coded 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:
/FB1/ Function Manual, Basic Functions; Mode group, Channel, Program Mode (K1), Chapter "Program Test"

2.4 Start-up of gantry axes

General information

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

If the synchronized axis is being overloaded by the leading axis due to reduced dynamics, this is acknowledged with alarm 10656.

References:

/IAD/ Start up manual

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

Axis 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 by means of axial machine data:

MD32100 $MA_AX_MOTION_DIR (travel direction).

Activation of the axis grouping

MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition)

This machine data is determined for the following gantry axis:

- To which gantry grouping (1, 2 or 3) the axis must be assigned,
- Whether it is to act as the leading axis (single-decade MD value only) or as a synchronized axis.

Note

Please make sure that a gantry grouping specified as cross-channel or cross-NCU does not clash with gantry grouping numbers already assigned. In this case the gantry grouping numbers must be clearly assigned cross-channel and cross-NCU. If this is not the case, alarm 10651 is output with reason 40XX. XX is the gantry grouping causing the clash.

For start-up purposes, all axes in a gantry grouping must be declared either as linear axes or as rotary axes:

MD30300 $MA_IS_ROT_AX (rotary axis/spindle)

Table 2-1 Examples for defining the gantry axis grouping

MD37100 $MA_GANTRY_AXIS_TYPE	Gantry axis	Gantry grouping
0	None	-
1	Leading axis	1
1**1**	Synchronized axis	1
2	Leading axis	2
1**2**	Synchronized axis	2
3	Leading axis	3
1**3**	Synchronized axis	3
...		
8	Leading axis	8
1**8**	Synchronized axis	8

Entering gantry trip limits

For the monitoring of the actual position values of the synchronized axis in relation to the actual position of the leading axis, the limit values for termination, as well as for the leading and synchronized 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 trip limit for 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 the digital 611D 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 leading and synchronized axes are set to the **same dynamic response value**.

To ensure the best possible synchronism, the leading axis and synchronized 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.

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

- MD32200 $MA_POSCTRL_GAIN (KV 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:

/FB2/ Function Manual, Extended Functions, Compensations (K3)

The following control parameters must be set to the same value for the leading axis and synchronized 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:

/FB1/ Function Manual, Basic Functions, Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)

Dynamics matching

The leading axis and the coupled 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.

Leading axis:

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

Following axis:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME [n] = 3ms

- Time constant of dynamic response adaptation for synchronized axis:

  MD32910 $MA_DYN_MATCH_TIME [n] = 5ms - 3ms = 2ms (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 leading and synchronized 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 adjustservo gain factors or feedforward control parameters slightly to achieve an optimum result.

Referencing gantry axes

The positions of the reference points of the leading and synchronized axes 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 leading and synchronized axes, the gantry grouping must be aligned before the axes are traversed. After this, gantry axes referencing must be performed according to the Section "Referencing and synchronizing of gantry axes" and:

References:

/FB1/ Function Manual, Basic Functions, Reference Point Approach (R1)

After the leading and synchronized axes have been referenced, the difference between them must be measured (comparison of position actual value indication in "Service axes" display of "Diagnosis" operating area). This difference must be applied 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 position actual values of the leading and synchronized 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 process must be activated with IS "Start gantry synchronization" (see Section "Referencing and synchronizing of gantry axes"). Once the axes have been synchronized (IS "Gantry grouping is synchronized" = 1), the dimensional offset between the leading and synchronized axes must be checked to ensure that it equals 0. Corrections may need to be made in the machine data mentioned above.

Input of gantry warning limit

Once the reference point values for the leading and synchronized 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 the value 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 leading axis **and** the synchronized axis must be calculated and entered in the appropriate parameters or tables.

References:

/FB2/ Function Manual, Extended Functions, Compensations (K3)

Function generator/measuring function

The activation of the function generator and measuring function will be aborted on the synchronized axis with an error message.

When an activation of the synchronized axis is absolutely necessary (e.g. to calibrate the machine), the leading and synchronized axes must be temporarily interchanged.

Note

Generally, the start of the function generator, measuring functions and AM setup triggers the virtual axes to abort upon error recognition.

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:

/FBA/ Description of Functions, Drive Functions, Speed Control Loop (DD2)

Note

As a supplement to the more general description given here of features of start-up and dynamic control response of drives, a complete example of a concrete constellation defined on the basis of its machine data can be found in Chapter "Example".

Start-up support for gantry groupings

The start-up functions of the function generator and measuring are parameterized via the PI service. All parameterized axes commence traversing when the NC Start key on the MCP panel is pressed in JOG mode.

A window is displayed in the "Measuring function and function generator in gantry grouping" operator 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.

2.5 PLC interface signals for gantry axes

Special IS for gantry axes

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

Table 2-2 Assignment of gantry-PLC interface signals to leading and synchronized axes

PLC interface signal	PLC$NCK	DB31, ... DBX...	Leading axis	Synchronized axis
Start gantry synchronization	!	29.4	X	
No automatic synchronization	!	29.5	X	
Gantry axis	z	101.7	1	1
Gantry leading axis	z	101.6	1	0
Gantry grouping is synchronized	z	101.5	X	
Gantry synchronization ready to start	z	101.4	X	
Gantry warning limit exceeded	z	101.3		X
Gantry trip limit exceeded	z	101.2		X

Effect of axial interface signals on gantry axes

a) Axial interface signals from PLC to axis (PLC → NCK)

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:

Table 2-3 Effect of interface signals from PLC to axis on leading and synchronized axes

PLC interface signal	DB31, ... DBX ...	Effect on	
		Leading axis	Synchronized axis
Axis/spindle disable	1.3	On all axes in gantry grouping	No effect
Position measuring system 1/2	1.4 and 1.5	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	4.4	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 hardware 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
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. DB31, ... DBX2.1 (servo 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 axis to PLC (NCK → 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)

2.6 Miscellaneous points regarding gantry axes

Manual travel

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

Handwheel override

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

DRF offset

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

Programming in part program

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

PLC or command axes

Only the leading 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 leading axis. All axes in the gantry grouping are reevaluated internally in the control when PRESET is activated. The gantry axis 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 leading axis is known in several channels, its synchronized 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 replacement

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

A replacement of the leading 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 change 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 MD parameterization is ignored for the synchronized axes:

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

The state of the leading 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 leading axis and the synchronized axes. The same applies to the service display values in the "Diagnosis" operating area.

Software limit switch

The SW limit switch monitor is processed for the leading axis only. If the leading 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 leading and synchronized 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 synchronized axes must be set like the leading 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, Basic Functions, Reference Point Approach (M3)

2.7 Examples

2.7.1 Creating a gantry grouping

Introduction

The gantry grouping, the referencing of its axes, the orientation of possible offsets and, finally, the synchronization of 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 describe 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 trip limit for referencing)

MD37140 $MA_GANTRY_BREAK_UP = 0 (invalidate gantry 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. 100mm (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.

2.7.2 Setting of NCK PLC interface

Introduction

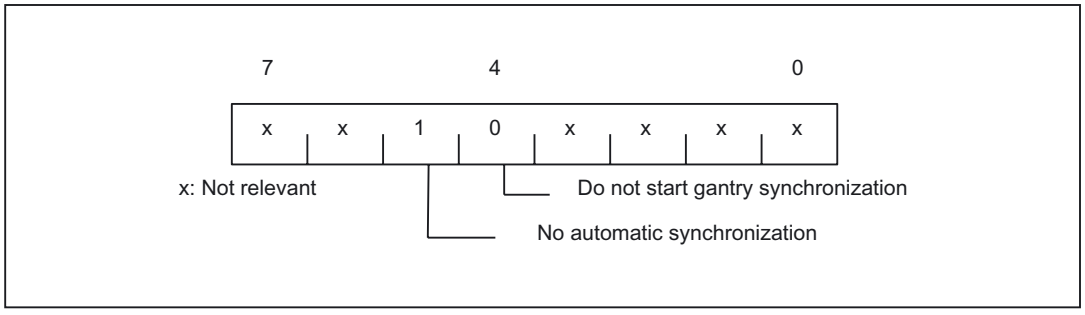
An automatic synchronization process during axis referencing must be disabled initially so as to prevent any damage to grouping axes that are misaligned.

Disabling of automatic synchronization

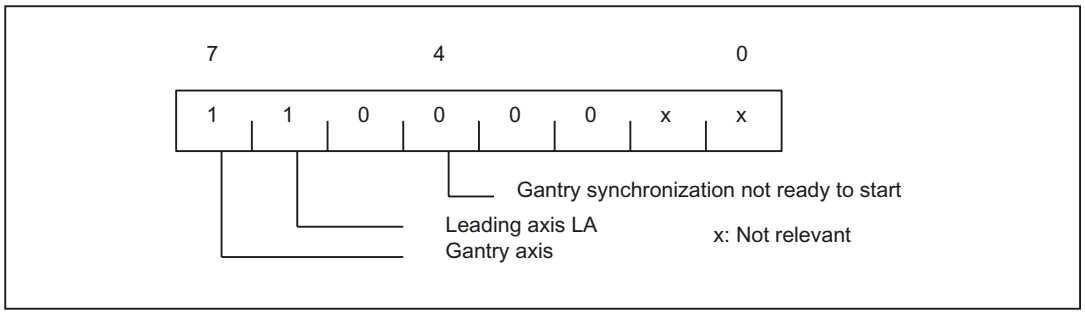
The PLC user program sets the following for the axis data block of axis 1:

DB31, ... DBX29.4 = 0

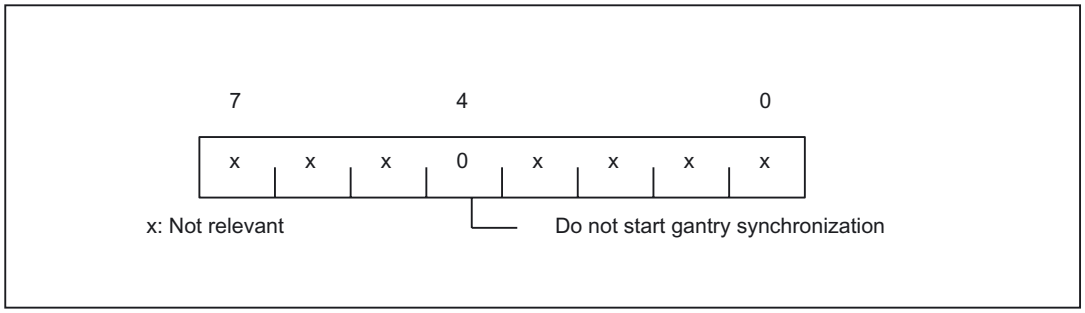
DB31, ... DBX29.5 = 1



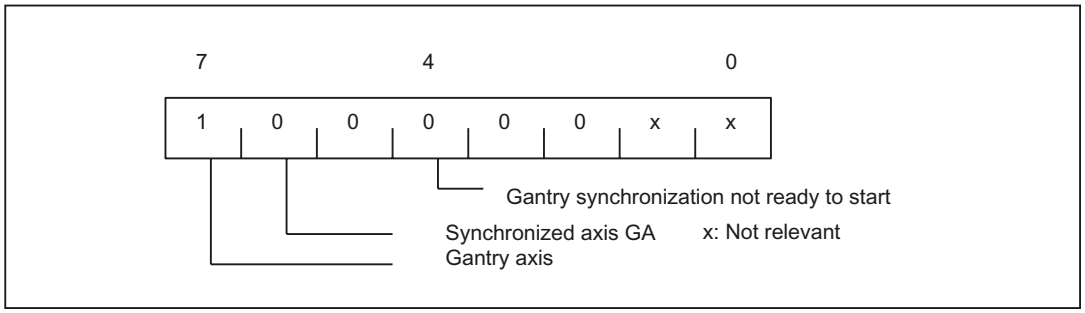
The NCK sets the following as a confirmation in the axis block of axis 1:
DB31, ... DBB101:



The PLC user program sets the following for the axis data block of axis 3:
DB31, ... DBX29.4 = 0



The NCK sets the following as a confirmation in the axis block of axis 3:
DB31, ... DBB101:



2.7.3 Commencing start-up

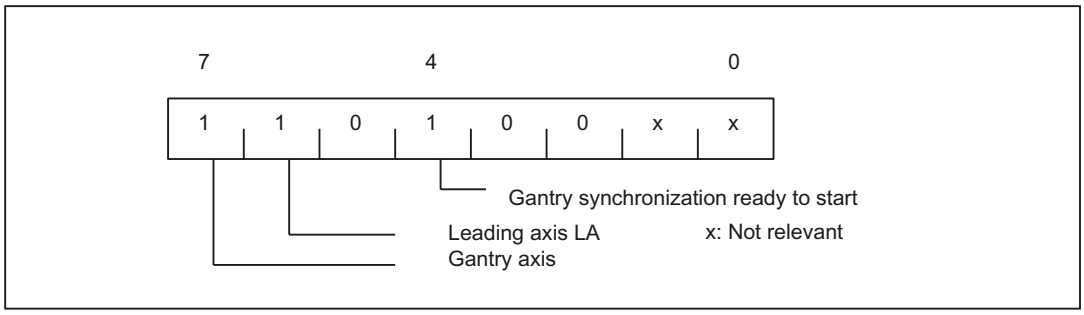
Referencing

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 NCK has prepared axis 1 for synchronization and registers this to the interface signal:

DB31, ... DBB101 with:



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

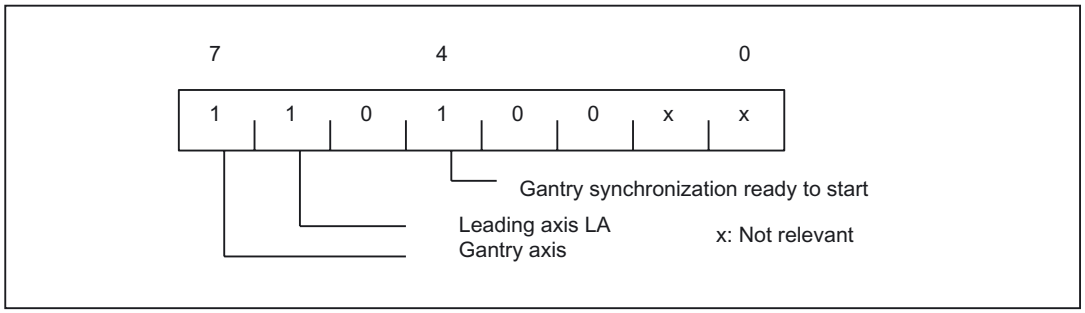
This MD is effective after power ON. To avoid having to perform a power ON now, the value can also be entered in the following machine data:

MD34080 $MA_REFP_MOVE_DIST (reference point distance)

The MD 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 NCK has prepared axis 1 for synchronization and registers this to the interface signal:

  DB31, ... DBB101(gantry)



- Examine actual positions of machine. Case A or B might apply:

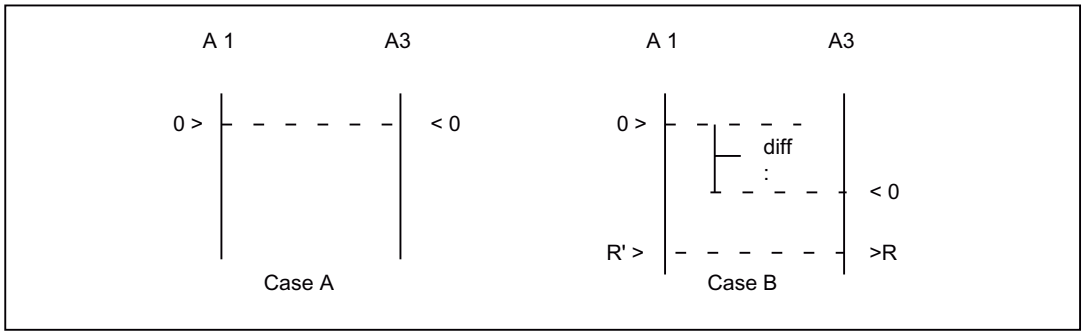


Figure 2-3 Possible results after referencing of axis 1 (master axis)

If Case A applies, the synchronization process can be started immediately. See step "Start synchronization". 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 Step 1 (see above).
- Start gantry synchronization. PLC sets:

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

2.7.4 Setting warning and trip limits

As soon as the gantry grouping is set and synchronized, the following machine data must still be set to correspond:

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)

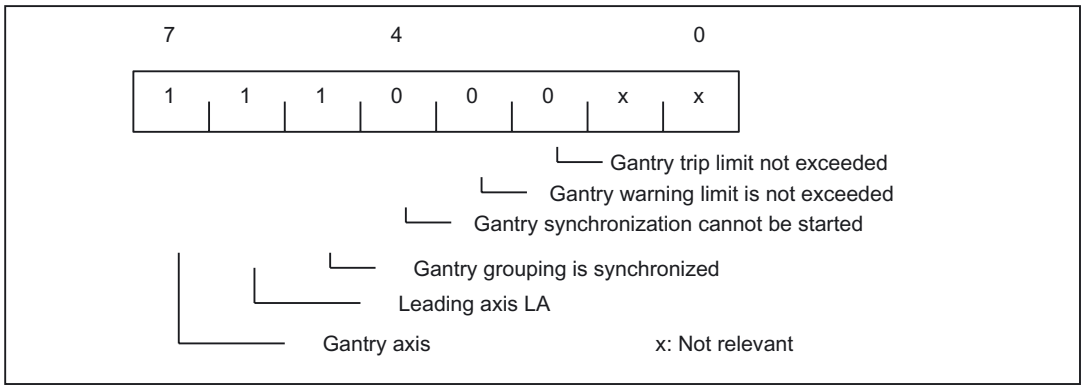
  When you put a heavy, dynamic strain on the axes, always be careful to re-enter the self-canceling alarm "10652 channel %1 axis %2 gantry warning limit exceeded".

- Now increase the following machine data:

  MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

  Do this until the alarm no longer appears. The interface indicates the status specified below. (That must occur in the appropriate window, according to production.)

In case the monitoring still only chimes very sporadically, one can program a centering maker into 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

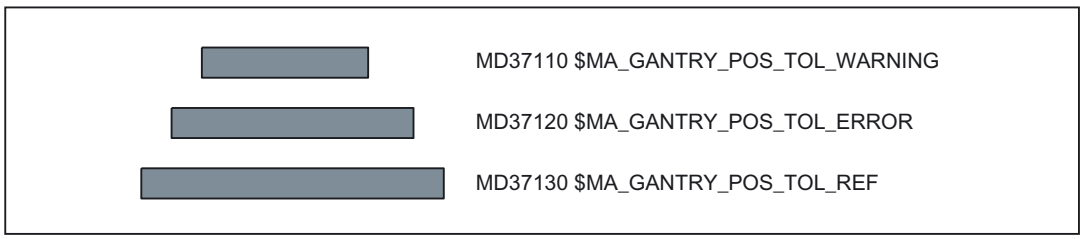
Values are entered in the following machine data:

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 trip limit for referencing)

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



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 IS "Gantry is synchronous" is not present (e.g. to be resynchronized after canceling alarms without gantry).

2.8 Data lists

2.8.1 Machine data

2.8.1.1 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
32200	POSCTRL_GAIN	K_V factor
32400	AX_JERK_ENABLE	Axial jerk limitation
32410	AX_JERK_TIME	Time constant for axis 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 mode
32650	AX_INERTIA	Moment of inertia for torque feedforward control
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant, current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant, speed control loop for feedforward control
32900	DYN_MATCH_ENABLE	Dynamic response adaptation
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
33000	FIPO_TYPE	Fine interpolator type
34040	REFP_VELO_SEARCH_MARKER	Creep velocity
34070	REFP_VELO_POS	Reference point start velocity
34080	REFP_MOVE_DIST	Reference point approach distance
34090	REFP_MOVE_DIST_CORR	Home position 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	Exact stop coarse/fine factor and zero speed
36030	STANDSTILL_POS_TOL	Zero speed 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 trip limit
37130	GANTRY_POS_TOL_REF	Gantry trip limit for referencing
37140	GANTRY_BREAK_UP	Invalidate gantry axis grouping

2.8.2 Signals

2.8.2.1 Signals from mode group

DB number	Byte.bit	Description
11, ...	5.2	Active machine function REF

2.8.2.2 Signals from channel

DB number	Byte.bit	Description
21, ...	33.0	Referencing active

2.8.2.3 Signals to axis/spindle

DB number	Byte.bit	Description
31, ...	29.4	Start gantry synchronization
31, ...	29.5	No automatic synchronization

2.8.2.4 Signals from axis/spindle

DB number	Byte.bit	Description
31, ...	60.4, 60.5	Referenced/synchronized 1, referenced/synchronized 2
31, ...	101.2	Gantry trip limit exceeded
31, ...	101.3	Gantry warning limit exceeded
31, ...	101.4	Gantry synchronization ready to start
31, ...	101.5	Gantry grouping is synchronized
31, ...	101.6	Gantry leading axis
31, ...	101.7	Gantry axis

Cycle Times (G3)

3

3.1 Brief description

This description explains the relationships and machine data of the various system cycles of the NC:

- Basic system clock cycle
- Interpolator cycle
- Position controller cycle

SINUMERIK 840D

For SINUMERIK 840D the position control cycle and the interpolator cycle (IPO cycle) are derived from the system clock cycle, which is set in the machine data of the NC.

SINUMERIK 840Di

In SINUMERIK 840Di the position control cycle and the interpolator cycle are derived from the system clock cycle, which is synchronized with the equidistant DP cycle T_{DP} set while creating the configuration in the S7 project (⇒ No setting of system clock cycle in SINUMERIK 840Di via machine data of NC).

3.2 Startup

Parameter assignment

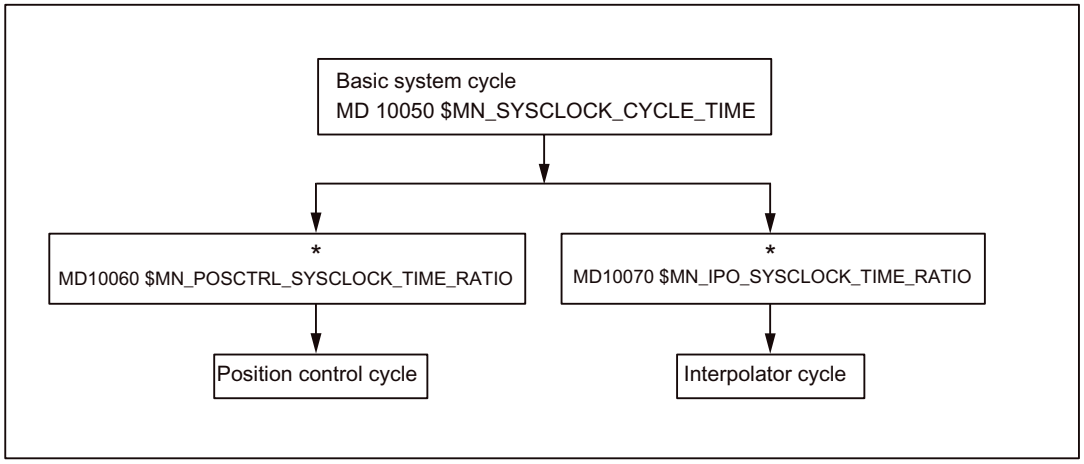
System clock cycle, position control cycle and interpolator cycle are defined with the following machine data:

MD10050 $MN_SYSCLOCK_CYCLE_TIME (system clock cycle)

MD10060 $MN_POSCTRL_SYSCLOCK_TIME_RATIO (factor for position-control cycle)

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (factor for the interpolation cycle)

MD10050, $MN_SYSCLOCK_CYCLE_TIME sets the system clock cycle of the system software in seconds. The other time cycles are derived by multiplying the system clock cycle with the factor specified in the concerned machine data:



Default values

Cycle	840D NCU 571	840D NCU 572	840D NCU 573
Basic system cycle	6 ms	4 ms	4* / 8# ms
Position control cycle	6 ms	4 ms	4* / 8# ms
Interpolator cycle	18 ms	12 ms	12* / 40# ms
* with 2 channels and 12 axes # with > 2 channels			

The default settings ensure that a maximum configuration of the system can power up reliably.

The cycle times, e.g. for the NCU 573, can generally be set to the lower values.

Example of cycle settings

The machine data is assigned as follows:	This results in the following cycle times:
MD10050 = 0.002	⇒ System clock cycle = 2 ms
MD10060 = 1	⇒ Position control cycle = 1 * 2 ms = 2 ms
MD10070 = 3	⇒ Interpolator cycle = 3 * 2 ms = 6 ms

3.3 SINUMERIK 840D

Interpolator cycle

The interpolator cycle defines the cycle time in which the setpoint interface to the position controllers is updated. The interpolator cycle is important for two reasons in normal processing:

- The product of velocities and interpolator cycles defines the geometry resolution of the interpolated contour. A long interpolator cycle causes a large path error along curved contours. This error is, however, reduced in the ratio interpolator / position-control cycle by cubic fine interpolation MD 33000 $MA_FIPO_TYPE.
- The interpolator cycle determines the possible resolution of the velocity profiles. It must be adapted to the dynamics of the drives so that the machine axes traverse and accelerate evenly (i.e. position-control cycle time ≤ interpolator cycle << acceleration time constant).

Position control cycle

The position-control cycle is the time which it takes for the control to calculate the actual value and transfer a new speed setpoint to the speed controller.

Block cycle time

The block cycle time is the sum of the block change time and block preparation time. It is at least as big as the cycle in which the position setpoints are assigned to the position controllers - normally equal to the interpolation cycle.

The block cycle time is a common form of measurement used to judge whether the control is suitable for traversing contours defined in points (frequent problem with 3 and 5-axis milling). It determines the maximum possible velocity at which a defined point pattern can be traversed (max. feedrate = average distance between points/block cycle time).

Setting the IPO cycle and position-control cycle

The interpolator and position-control cycles are set in **integer multiples** of the system clock cycle in the following machine data:

MD10060 $MN_POSCTRL_SYSCLOCK_TIME_RATIO (factor for position-control cycle)

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (factor for the interpolation cycle)

The smallest possible position-control and interpolator cycle should be aimed for.

Apart from special applications in which MD10060 is set to a value greater than 1, the position control cycle corresponds to the system clock cycle (MD10060 = 1).

The ratio of interpolator to position-control cycle must be an integer value and greater than or equal to 1.

$$\frac{\text{Interpolator cycle}}{\text{Position control cycle}} \geq 1$$

If this is not the case, there is an automatic offset and the following alarm is displayed:

Alarm 4102 "IPO cycle increased to [ ] ms"

3.4 SINUMERIK 840Di with PROFIBUS DP

Note

Further information on SINUMERIK 840Di can be found in:
References:
/HBI/ SINUMERIK 840Di Manual, PROFIBUS-DP Communication

3.4.1 Description of a DP cycle

Actual values

At time T_I, the actual position values are read from all the equidistant drives (DP slaves). In the next DP cycle, the actual values are transferred to the DP master in the time T_{DX}.

Position controller

In time T_M (where $T_M > T_{DX,}$) the NC position controller is started and will use the transferred actual position values to calculate the new speed setpoints.

Setpoints

At the start of the next DP cycle, the speed setpoints are transferred from the DP master to the DP slaves (drives) in the time T_{DX}.

At time T_O, the speed setpoints are taken as new specified values for all drive controllers.

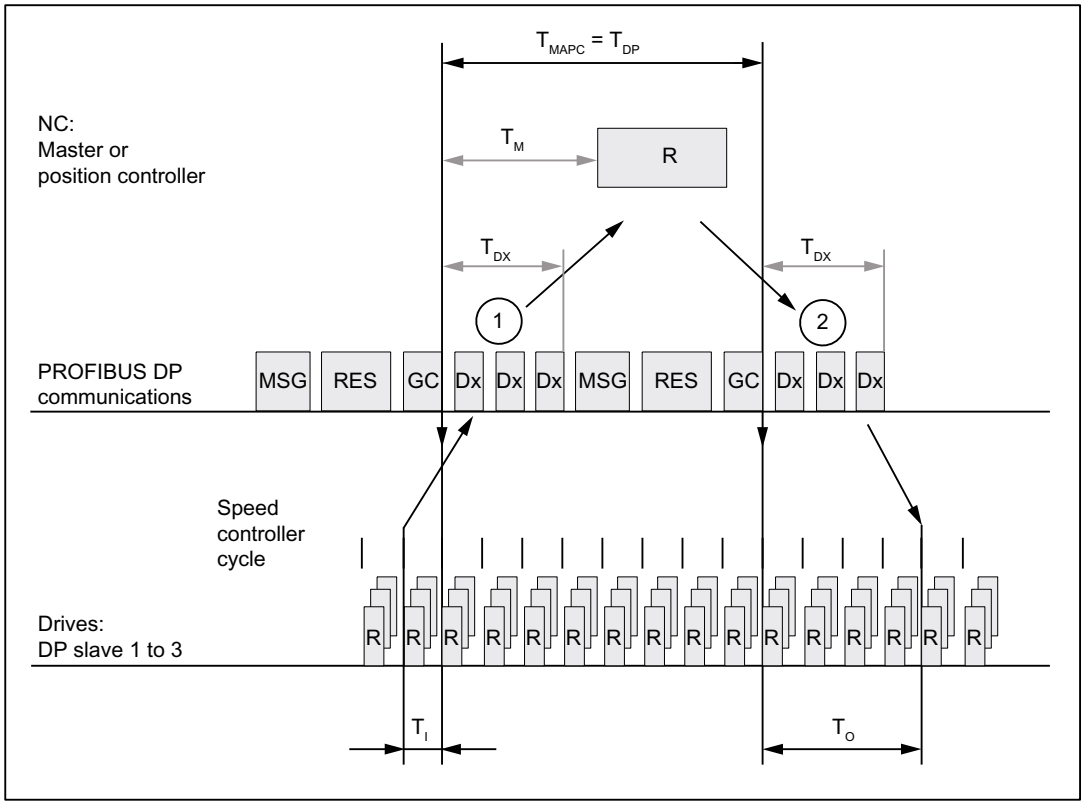


Figure 3-1 Optimized DP cycle with 3 DP slaves with a SIMODRIVE 611 universal

T_{MAPC}:	Master application cycle NC position control cycle for SINUMERIK 840Di always applies for: $T_{MAPC} = T_{DP}$
T_{DP}:	DP cycle time: DP cycle time
T_{DX}:	Data exchange time: Total transfer time for all DP slaves
T_M:	Master time: Offset of the start time for NC position control
T_I:	Input time: Time of actual value sensing
T_O:	Output time: time of setpoint activation
GC:	Global control message frame (broadcast message frame) for cyclic synchronization of the equidistance between the DP master and DP slaves
R:	Computation time for speed or position controller
Dx:	Useful data exchange between the DP master and DP slaves
MSG:	Acyclic services (e.g. DP/V1, pass token)
RES:	Reserve: "Active pause" until the isochronal (equidistant) cycle has expired
①	The actual values for the current DP cycle/position control cycle are transferred from the DP slave drives to the NC position controller.
②	The setpoints computed by the NC position controller are transferred to the DP slave drives.

3.4.2 Clock cycles and position-control cycle offset

Cycle times

The NC derives the cycle times (system clock cycle, position-control cycle and interpolator cycle) from the equidistant PROFIBUS-DP cycle set in the SIMATIC S7 project during configuration of the PROFIBUS.

Basic system clock cycle

MD10050 $MN_SYSCLOCK_CYCLE_TIME (system clock cycle)

The system clock cycle is set to the fixed ratio 1:1 with respect to the PROFIBUS-DP cycle. It cannot be changed.

Position controller cycle

MD10061 $MN_POSCTRL_CYCLE_TIME (position-control cycle)

The position controller cycle is set to the fixed ratio 1:1 with respect to the system clock cycle. It cannot be changed.

Interpolation cycle

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (factor for the interpolation cycle)

The interpolator cycle may be chosen freely as a integer multiples of the position control cycle.

Position control cycle offset

MD10062 $MN_POSCTRL_CYCLE_DELAY (position control cycle offset)

The offset for the position-control cycle (T_M) is set independently of the conditions described below within a PROFIBUS-DP/system cycle and independently of the cyclic communication with the DP slave:

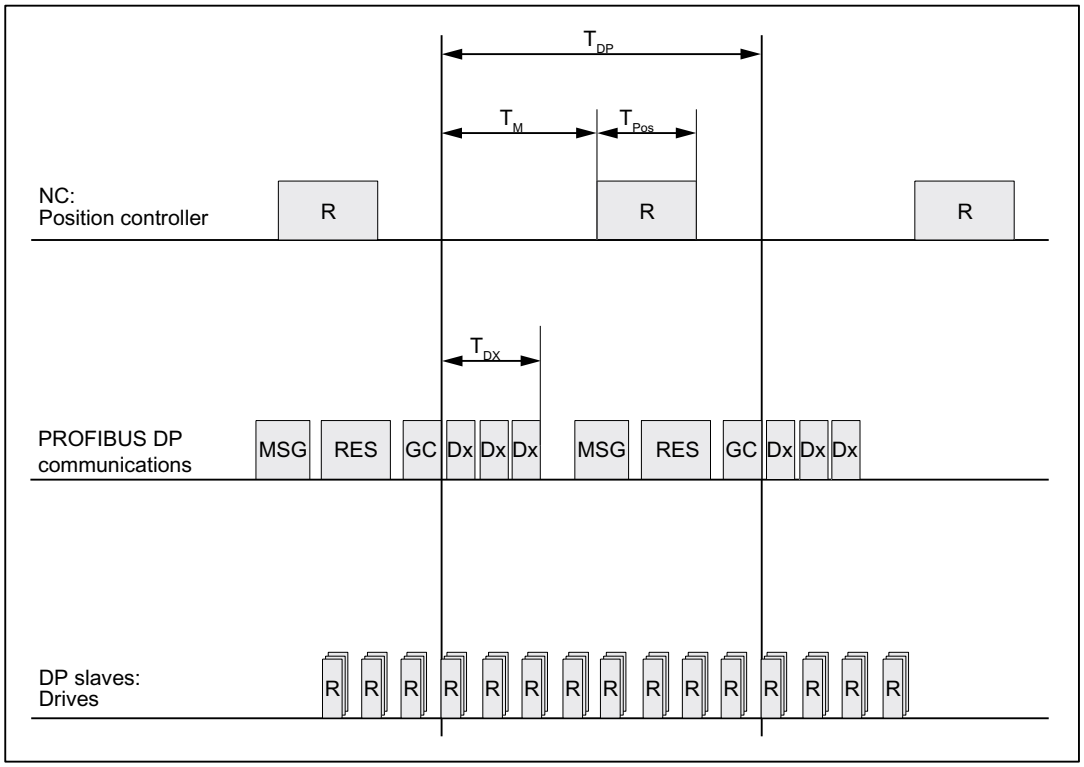


Figure 3-2 Position control cycle offset compared to PROFIBUS-DP cycle

Key to figure above:

T_{Pos}:	Computing time requirements for the position controller
T_{DP}:	DP cycle time: DP cycle time
T_{DX}:	Data exchange time: Total transfer time for all DP slaves
T_M:	Master time: Offset of the start time for NC position control
GC:	Global Control: Broadcast message for cyclic convergence of the equidistance between DP master and DP slaves
R:	CPU time
Dx:	Useful data exchange between the DP master and DP slaves
MSG:	Acyclic services (e.g. DP/V1, pass token)
RES:	Reserve: "Active pause" until the isochronal (equidistant) cycle has expired

Conditions and recommendations for MD10062

MD10062 $MN_POSCTRL_CYCLE_DELAY (position control cycle offset)

The position controller cycle offset (T_M) must be set such that the following conditions are fulfilled within a PROFIBUS-DP/system cycle:

- Cyclic communication with the DP slaves (drives) must be completed before the position controller is started.

  Condition: $T_M > T_{DX}$

- The position controller must have finished before the PROFIBUS-DP/system cycle comes to an end.

  Condition: $T_M + T_{Pos} < T_{DP}$

The following setting is recommended as approximate value for the position control cycle offset:

$T_M = T_{DP} - 3*Tmax_{position\ controller}$

T_{DP}: Position-control cycle or PROFIBUS-DP cycle

$Tmax_{position\ controller}$: Position controller maximum time

Note

HMI Advanced

The position controller maximum time is displayed in the "NC load" dialog under "Menu range switchover" > "Diagnosis" > "Service displays" > "System resources".

Error response

Alarm 380005 "PROFIBUS-DP: Bus access conflict, type t, counter z"

Causes of error/error handling:

- t = 1

  The position-control cycle offset selected is too small. Cyclic PROFIBUS-DP communication with the drives was not completed before the position controller started.

  Remedy: Increase the position-control cycle offset.

- t = 2

  The position-control cycle offset selected is too large. Cyclic PROFIBUS-DP communication with the drives started before the position controller had finished. The position controller requires more computation time than is available in the PROFIBUS-DP cycle.

  Remedy:

  – Decrease the position-control cycle offset

    or

  – Increase the PROFIBUS-DP cycle.

    The PROFIBUS-DP cycle must be set in the SIMATIC S7 project in advance.

Alarm requests in the event of a conflict during startup

MD10059 $MN_PROFIBUS_ALARM_MARKER (PROFIBUS alarm marker)

In this machine data, alarm requests on the PROFIBUS level are stored even after reboot.

If a conflict is found between the following machine data and the data in the PROFIBUS-SDB during the booting, the machine data is adjusted to this SDB and an alarm is set during the next booting:

MD10050 $MN_SYSCLOCK_CYCLE_TIME (system clock cycle)

MD10060 $MN_POSCTRL_SYSCLOCK_TIME_RATIO (factor for position-control cycle)

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO (factor for the interpolation cycle)

Points to note

The following special points must be observed for cycle-specific machine data:

- MD10050 $MN_SYSCLOCK_CYCLE_TIME (system clock cycle)

  The machine data is simply for display purposes. The system cycle is always equal to the equidistant PROFIBUS-DP cycle.

- MD10060 $MN_POSCTRL_SYSCLOCK_TIME_RATIO (factor for position-control cycle)

  The factor for the position-control cycle is set at a fixed value of 1 and cannot be modified.

⚠ CAUTION

If you change the cycle times, check the behavior of the controller in all operating modes before you finish commissioning.

Note

The smaller the cycle times (PROFIBUS DP cycle) chosen, the greater the control quality for the drive and better the surface quality on the workpiece.

3.5 Data lists

3.5.1 Machine data

3.5.1.1 General machine data

Number	Identifier: $MN_	Description
10050	SYSCLOCK_CYCLE_TIME	Basic system clock cycle
10059	PPOFIBUS_ALARM_MARKER	PROFIBUS alarm marker (internal only)
10060	POSCTRL_SYSCLOCK_TIME_RATIO	Factor for position control clock cycle
10061	POSCTRL_CYCLE_TIME	Position control cycle
10062	POSCTRL_CYCLE_DELAY	Position control cycle offset
10070	IPO_SYSCLOCK_TIME_RATIO	Factor for interpolator clock cycle
10071	IPO_CYCLE_TIME	Interpolator cycle
10080	SYSCLOCK_SAMPL_TIME_RATIO	Division factor of the position control cycle for actual value acquisition
11250	PROFIBUS_SHUTDOWN_TYPE	PROFIBUS DP shut down handling

3.5.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
33000	FIPO_TYPE	Fine interpolator type
37600	PPOFIBUS_ACTVAL_LEAD_TIME	Actual-value acquisition time (PROFIBUS Ti)
37602	PPOFIBUS_OUTVAL_DELAY_TIME	Setpoint delay time (Profibus To)

Contour Tunnel Monitoring (K6) 4

4.1 Brief description

4.1.1 Contour tunnel monitoring

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 recognised deviation, the system reacts as quick as possible. However, at least one interpolation 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.

Deceleration 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.

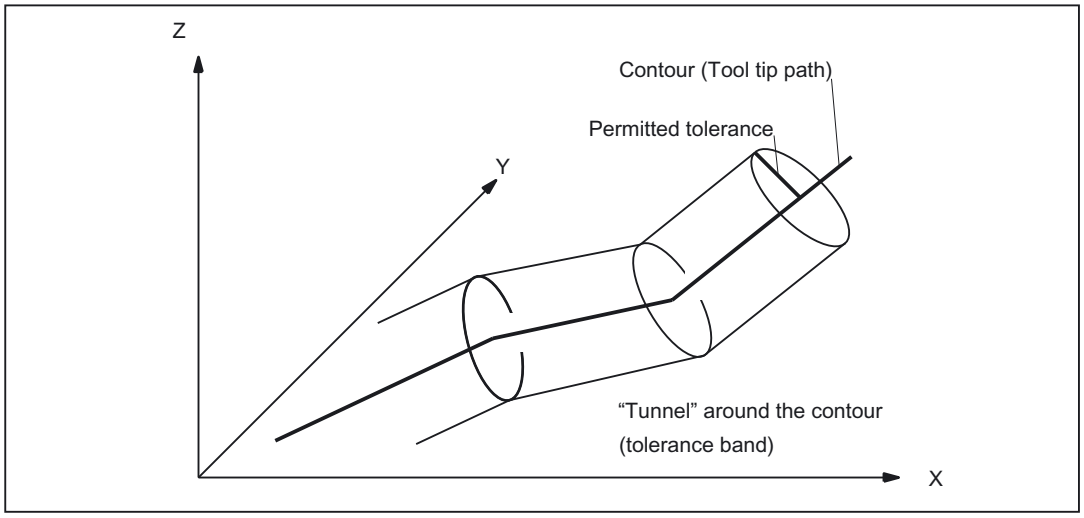


Figure 4-1 Position of the contour tunnel around the programmed path

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

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.

The radius is set in the machine data:

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 (Tolerance on position actual value switchover)

Activating

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

- The contour tunnel monitoring function is set.
- MD21050 is higher than 0.0.
- At least two geometry axes have been defined.

Stopping

Monitoring can be stopped by enabling the MD 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 analogue output (accuracy monitoring).

The parameter assignment is activated with machine data:

MD21070 $MC_CONTOUR_ASSIGN_FASTOUT
(Assignment of an analog output for output of the contour error)

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

Initial situation

There is always a velocity-dependent difference between setpoint and actual position when an axis is traversed without feedforward control. This lag results in inaccurate curved contours.

References:
/FB1/ Function Manual, Basic Functions
/PG/ Programming Manual, Fundamentals

Function

The "Programmable contour accuracy" function permits the user to specify a maximum error for the contour in the NC program, which may not be exceeded. The control calculates the K_V factor (servo gain factor) for the axes concerned and limits the maximum path velocity so that the contour error resulting from the lag does not exceed the value specified. The "LookAhead" function then ensures that the velocity necessary for maintaining the required contour accuracy is not exceeded at any point along the path.

Application

The function ensures a defined contour accuracy in situations where feedforward control cannot or must not be used.

Positioning axes

The function does not affect the velocities of positioning axes.

Active feedforward control

The function is also operative in conjunction with active feedforward control if the following machine data is set to TRUE:

MD20470 $MC_MC_CPREC_WITH_FFW (Programmable contour accuracy)

With active feedforward control, the reduction of the path velocity is calculated on the basis of the effective K_V factor with feedforward control.

Minimum feed

In order to avoid burn marks, the feed is limited to a minimum value:

SD42460 $SC_MINFEED (Minimum path feed with CPRECON)

Activating

The function can be switched on and off by means of the modal G codes `CPRECON` and `CPRECOF` (**C**ontour **Prec**ision **On**/**Off**).

The magnitude of the contour accuracy is specified with setting data:

SD42450 $SC_CONTPREC (contour accuracy)

Changes to the setting data become valid during preprocessing.

RESET/end of program

On RESET/program end the response set in the following machine data for the G code group 39 will become effective:

MD20110 $MC_RESET_MODE_MASK
(Definition of control default settings after reset/TP end)

MD20112 $MC_START_MODE_MASK
(Definition of the control default settings in case of NC start)

e.g. for programmable contour accuracies there are no particularities.

References:
/FB1/ Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2), subject: Workpiece-related actual-value system

4.4 Constraints

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 Examples

4.5.1 Programmable contour accuracy

Extract from part program

```
N10 X0 Y0 G0
N20 CPRECON                    ; Enabling of contour accuracy defined by MD
N30 F10000 G1 G64 X100         ; Machine contour at 10 m/min in continuous-path mode
N40 G3 Y20 J10                 ; Automatic limitation of feed in circle block
N50 G1 X0                      ; Feedrate again 10m/min
N100 CPRECOF                   ; Disabling of programmed contour accuracy
N110 G0 ...
```

4.6 Data lists

4.6.1 Machine data

4.6.1.1 Channel-specific machine data

Number	Identifier: $MC_	Description
20470	CPREC_WITH_FFW	Programmed 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

4.6.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover

4.6.2 Setting data

4.6.2.1 Channel-specific setting data

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

Axis couplings (M3)

5

5.1 Coupled motion

5.1.1 Product brief

5.1.1.1 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.

5.1.1.2 Preconditions

Coupled motion function

The coupled motion function forms part of the NCK 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 NCK software is supplied with the relevant options of generic coupling (refer to " Preconditions (Page 263) " in the "Brief Description" of Generic Coupling).

5.1.2 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.

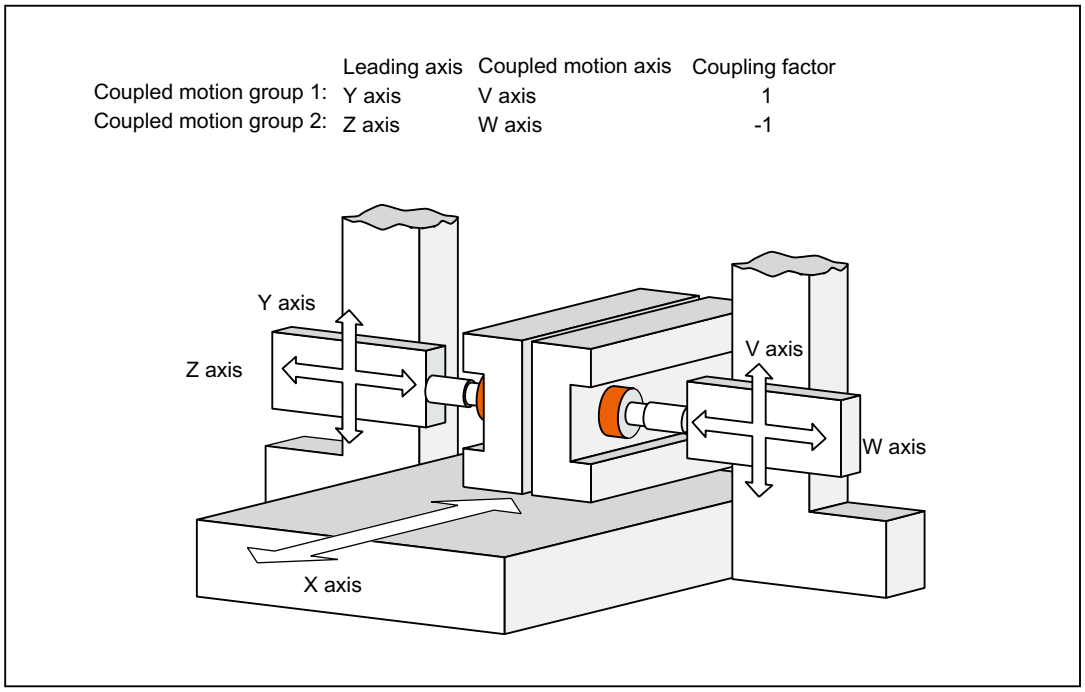


Figure 5-1 Application Example: Two-sided machining

Multiple couplings

Up to two 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).

Switch ON/OFF

Coupled motion can be activated/deactivated via the part programs and synchronous actions. In this context please ensure that the switch on and switch off is undertaken with the same programming:

- Switch on: Part program → Switch off: Part program
- Switch on: Synchronous action → Switch off: Synchronous 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

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 with NC-START)

Bit	Value	Description
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	Description
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.

5.1.3 Programming

5.1.3.1 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>])

Effective: modal

Parameters:

Coupled motion axis:	Type: AXIS	
	Range of values:	All defined axis and spindle identifiers in the channel
Leading axis:	Type: AXIS	
	Range of values:	All defined axis and spindle identifiers 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:

```
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.
```

5.1.3.2 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>)
or (in abbreviated form):
TRAILOF(<coupled-motion axis>)

Effective: modal

Parameters:

Coupled motion axis:	Type: AXIS	
	Range of values:	All defined axis and spindle identifiers in the channel
Leading axis:	Type: AXIS	
	Range of values:	All defined axis and spindle identifiers in the channel

Example

```
TRAILOF(V,Y)            ; Switch off of the coupling between the coupled-motion
                          axis V and leading axis Y.
```

5.1.4 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 (feed offset)
- 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 feed 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 the measuring system 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 identifier] (see: Status of coupling)

5.1.5 Status of coupling

The coupling status of an axis can be determined using the following system variables:

$AA_COUP_ACT[axis identifier]

Value	Description
0	No coupling active
1, 2, 3	Tangential tracking
4	Synchronous spindle coupling
8	Coupled motion active
16	Master value coupling
32	Following axis of electronic gearbox

Note

Only one coupling mode may be active at any given time.

5.1.6 Dynamics limit

The dynamics limit is dependent on the type of activation of the coupled axis grouping:

- Activation in part program

  If activation 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

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.

5.2 Curve tables

5.2.1 Product brief

5.2.1.1 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.

5.2.1.2 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)

5.2.2 General functionality

Curve tables

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.

References:
/PGA/ Programming Manual, Work Preparation

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.

5.2.3 Memory organization

Memory configuration

The storage place available for the curve table in the static and dynamic NC memory is defined during memory configuration (→ refer to Commissioning / Memory Configuration).

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 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 alram 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 accomodate 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.

5.2.4 Commissioning

5.2.4.1 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.

5.2.4.2 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.

5.2.4.3 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.

5.2.5 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 cahnge.

- 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:
  Identifier of axis via which the following axis is programmed in the definition.
- Leading axis:
  Identifier 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

References:
/PGA/ Programming Manual, Work Preparation; Path Behavior Chapter Curve Tables (CTAB)

Restrictions

The following restrictions apply when programming:

- The NC block must not generate 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 removed.

Example 1

Without tool radius compensation, without memory type

```
N100 CTABDEF(AX2, AX1, 3,0)     ; Beginning of the definition of non-
                                ; periodic curve table No. 3
N110 AX1=0 AX2=0                ; 1st motion instruction specifies the start value
                                ; master 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 definition, curve table
                                ; is generated in internal representation.
                                ; Preprocessing reorganizes to state at
                                ; the beginning 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.

```
N10 CTABDEF(Y, X, 1, 0)         ; Beginning of the definition of non-
                                ; periodic curve table No. 1
N20 X0 Y0
N30 G41 X10 Y0                  ; WZR compensation on
N40 X20 Y20
N50 X40 Y0
N60 X60 Y20
N70 X80 Y0
N80 G40 X90 Y0                  ; WZR 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 identifiers that have been programmed in CTABDEF as the leading axis and following axis identifiers. In the case of programming errors, alarms or incorrect contours may be generated.

5.2.6 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 parts 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 segements 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 interpolation operation.

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:

```
N10 DEF REAL STARTPOS                        ; Beginning of the definition of
                                             ; start and starting position of
                                             ; curve table
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
N100 CTABDEF(Y, X, 1, 0)                     ; Begin of table definition
N110 X0 Y0                                   ; Starting position, 1st table
                                             ; segment
N120 X20 Y10                                 ; End position 1st table segment =
                                             ; start position 2nd table segment
N130 X40 Y40
N140 X60 Y10                                 ; 3rd curve segment
N150 X80 Y0                                  ; 4th 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 is associated with
                                           ; LW X = 30.0
```

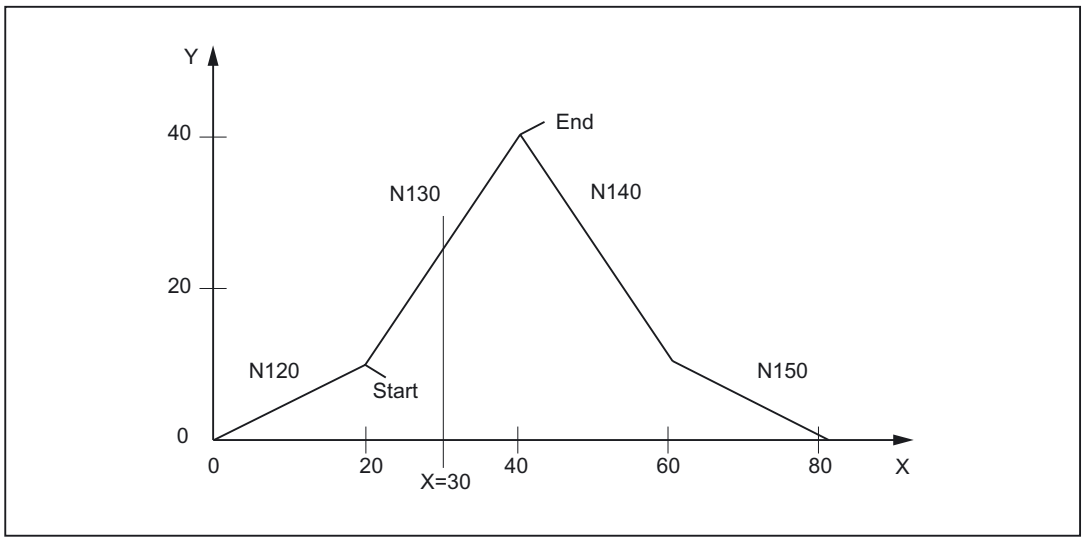


Figure 5-2 Determining the curve segment associated with master value X = 30

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 following value

The following example illustrates how the minimum and maximum values of the table are determined using CTABTMIN and CTABTMAX:

```
N10 DEF REAL STARTVAL                      ; Beginning of definition of start and
N20 DEF REAL ENDVAL                        ; Initial values of curve table
N30 DEF REAL STARTPARA
N40 DEF REAL ENDPARA
N50 DEF REAL MINVAL
N60 DEF REAL MAXVAL
N70 DEF REAL GRADIENT
…
N100 CTABDEF(Y, X, 1, 0)                   ; Begin 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 table
                                          ; segment ...
N150 X80 Y20                              ; End position of the 4th table
                                          ; 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
                                          ; Master value at end of curve table
...                                       ; read from value range of following
                                          ; axis
N240 MINVAL = CTABTMIN(1)                 ; Minimum value when Y = 5 and
N250 MAXVAL = CTABTMAX(1)                 ; Maximum value when Y = 40
```

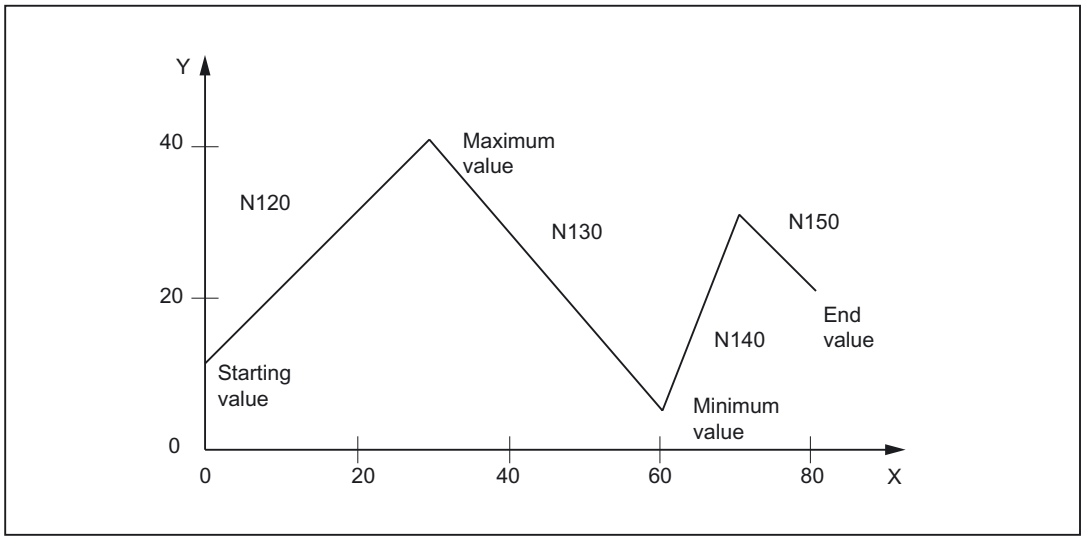


Figure 5-3 Determining the minimum and maximum values of the table

5.2.7 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.

5.2.8 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°

5.2.9 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.

5.2.10 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.

5.2.11 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

5.3 Master value coupling

5.3.1 Product brief

5.3.1.1 Function

The "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.

Either an **axis-** or **path-**master value coupling can be used and activated and deactivated in the NC part program or by using a synchronous action.

Coupling with the master value is defined using a **curve table**.

5.3.1.2 Preconditions

The option "Master Value Coupling and Curve Tables Interpolation" or the relevant option of generic coupling (refer to " Preconditions (Page 263) " in the "Brief description" for generic coupling) is a prerequisite for utilization of the function.

5.3.2 General functionality

Master value couplings are divided into axis and path master value couplings. In both cases, the axis and path positions are defined by the control system on the basis of master values (e.g. positions of an additional axis).

Axis master value coupling

Axis master value coupling is an axis coupling with rules of motion that are represented internally as a one-dimensional real function, a curve table.

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 MD 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 cycle, the interpolator for the axis is initialized by the NCK 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 these setting data.

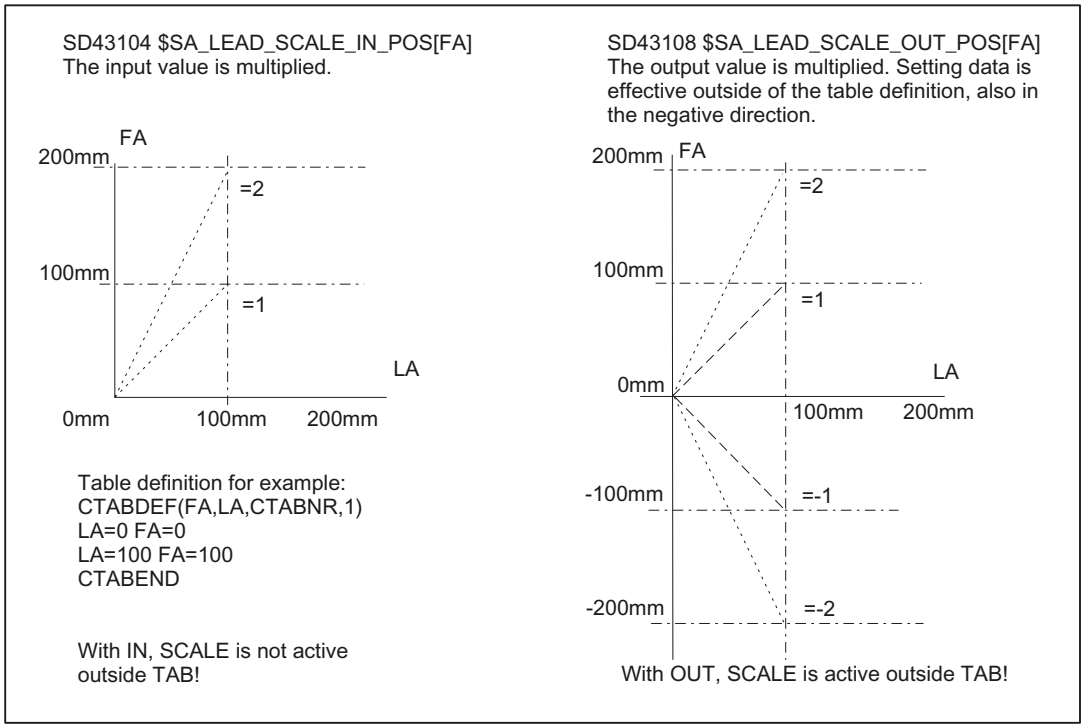


Figure 5-4 Master value coupling offset and scaling (multiplied)

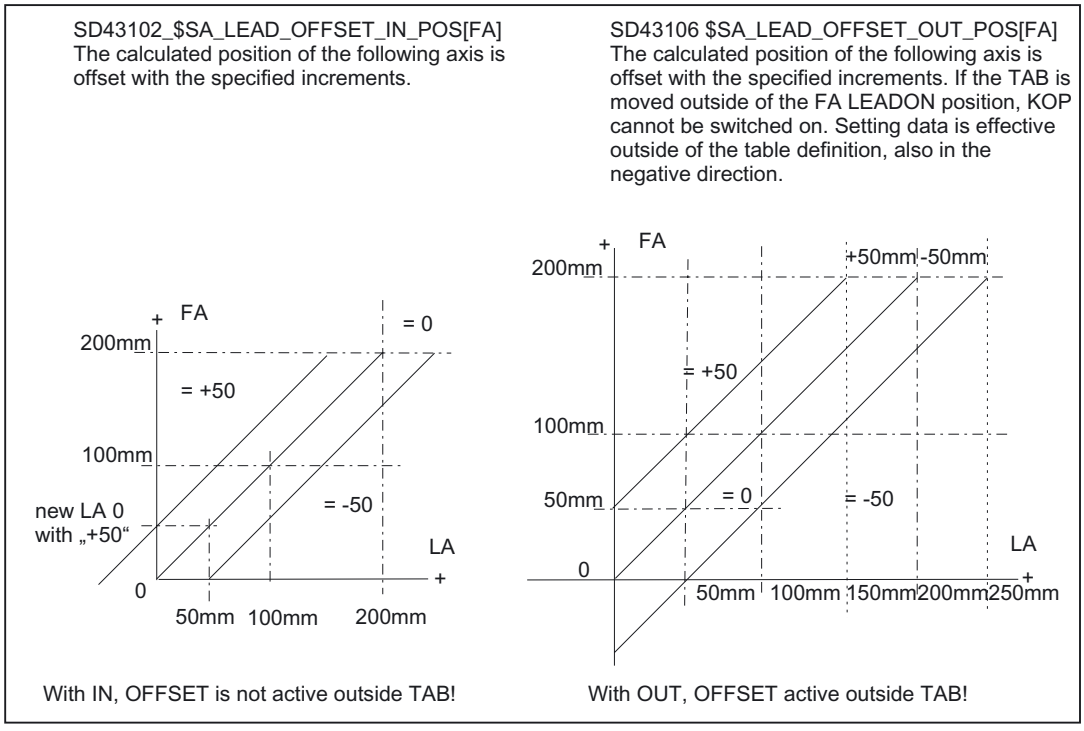


Figure 5-5 Master value coupling offset and scaling (with increment offset)

Reaction to Stop

All master 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 synchronous 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 the control default settings in case of 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)                                            ; Release for synchronous
                                                        action.
ID=1 WHEN ($AA_IM[X]<-50) DO LEADON(B,X,2)            ; Y is coupled to X via curve
                                                        table No.2.
```

5.3.3 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.
```

Constraints:

- 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)

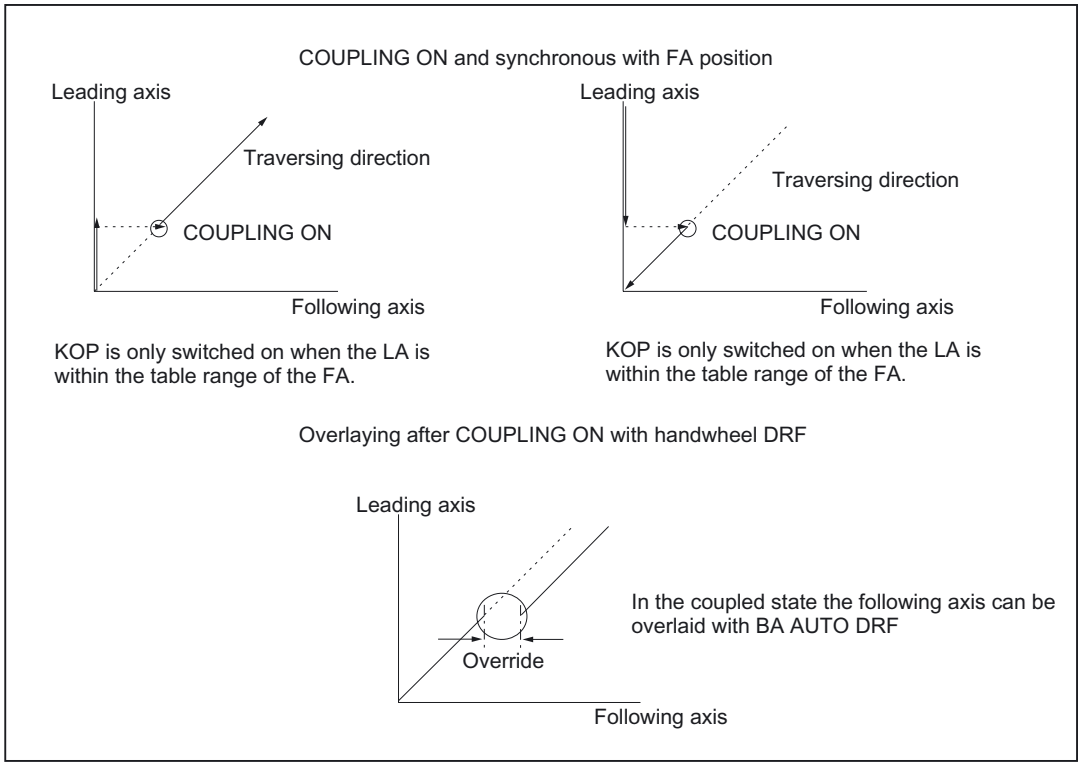


Figure 5-6 Activating master value coupling

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

The following system variables of the following axis can be read from the part program and from the synchronous actions:

System variable	Meaning	
$AA_SYNC[ax]	Condition of the coupling between following and leading axes	
	= 0	not synchronized
	= 1	"Coarse synchronism" achieved according to machine data: MD37200 $MA_COUPLE_POS_TOL_COARSE (threshold value for coarse synchronism)
	= 3	"Fine synchronism" achieved according to machine data: MD37210 $MA_COUPLE_POS_TOL_FINE (threshold value for fine synchronism)
	The information from system variable $AA_SYNC[ax] corresponds to the assigned NC/PLC interface signals: DB31, ... DBX98.0 (fine synchronism) DB31, ... DBX98.1 (coarse synchronism)	
$AA_IN_SYNC[ax]	Condition of the synchronization between following and leading axes	
	= 0	Synchronization not started or ended.
	= 1	Synchronization is running, i. e., the following axis is synchronized.
	The information of the system variable $AA_SYNC[ax] corresponds to the assigned NC/PLC interface signals: DB31, ... DBX99.4 (synchronization running)	

Note

If the following axis is not enabled for travel, it is stopped and is no longer synchronous.

5.3.4 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 system 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 is to 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 system 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.

References:
/FB1/ 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 system 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

5.3.5 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.

5.3.6 Special characteristics of the axis master value coupling function

Control system dynamics

Depending on the application in question, it may be advisable to match the position controller parameter settings (e.g. servo gain 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 "Coupled Motion", "Special Features of the Function"

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.

5.4 Electronic gearbox (EG)

5.4.1 Product brief

5.4.1.1 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

5.4.1.2 Preconditions

The "Electronic Gearbox" option or the relevant option of generic coupling (refer to " Preconditions (Page 263) " in the "Brief description" of Generic Coupling) is required for the usage of function.

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 parts 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
Knowledge of the control technology and measurements with servo trace are an absolute prerequisite for using this function.

References:
/IAD/ Commissioning Guide,
/FB1/ Function Manual, Basic Functions; Speeds, Setpoint/Actual Value Systems, Closed-Loop Control System (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")

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 "Definition of an EG Axis Group")

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".

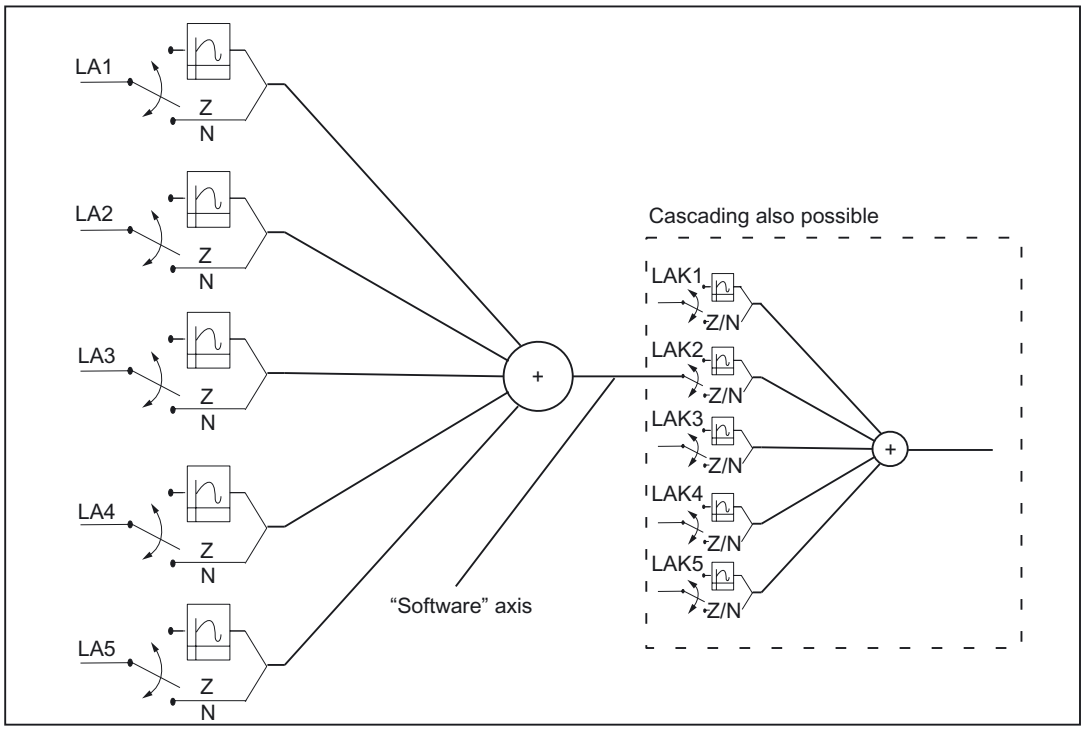


Figure 5-7 Block diagram of an electronic gearbox

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 "Definition of an EG Axis Group")

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.

   EGON (see Chapter "Switch-on of an EG Axis Group")

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 "Switch-on of an EG Axis Group")

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:** If 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 travers 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 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 **difference in synchronism** 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 difference in synchronism can be polled from within the parts program. See below.

Changes in the difference in synchronous traverse

The mass inertia of the axis systems during acceleration can cause dynamic fluctuations in the difference in synchronous traverse. The difference in synchronous traverse is checked continuously and the tolerance values in the machine data used to produce interface signals.

The difference in synchronous traverse is compared with the following machine data:

MD37200 $MA_COUPLE_POS_TOL_COARSE

and

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

and

NST "synchronous travers coarse" DB31, ... DBX98.1

Difference > .. TOL_COARSE

As long as the synchronous traverse 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 synchronous traverse difference is smaller than the following machine data, IS "Coarse synchronism" DB 31, ... DBX 98.1 is at the interface and IS "Fine synchronism" DB 31, ... DBX 98.0 is deleted.

MD37200 $MN_COUPLE_POS_TOL_COARSE

Difference > .. TOL_FINE

If synchronous traverse 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

Difference in synchronism for EG cascades

Deviation in synchronism 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:

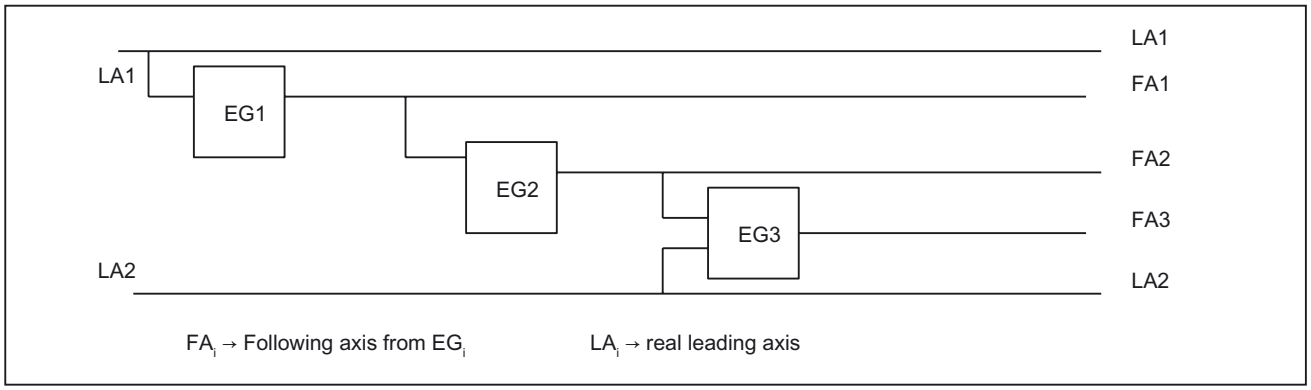


Figure 5-8 Three-level EG cascade

According to the definition given, the difference in synchronism 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, ... DBX98.5

The monitoring signals can be used as trigger criteria for emergency retraction. See Chapter "Possible trigger sources".

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 synchronous traverse difference

1. The result of the synchronism difference calculation can be read as an amount in the parts 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 synchronization 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.

5.4.2 Performance Overview of EG (Summary)

EG

An EG has:

- a maximum of 5 lead axes
- 1 following axis
- a maximum of 5 assigned curve tables or
- a maximum of 5 assigned coupling factors (Z/N) or
- a combination of curve tables and coupling factors for a maximum of 5 leading axes

Following axis

A following axis can:

- identify the EG uniquely
- be the leading axis of a different EG (cascading)
- not simultaneously be the leading axis of the same EG (no feedback)
- not be command axis

Leading axis

A leading axis can:

- be used once in the same EG
- be used as leading axis in several coupling modules
- be PLC axis
- be command axis

Leading and following axis

The following is allowed for leading and following axes:

real	simulated
linear axis	
rotary axis	
modulo-corrected rotary axis	

Type of coupling

For each leading axis, the EG may refer to:

- the actual value or
- the setpoint

Reference system

The calculations are made in the basic co-ordinate system BCS.

Synchronous actions

Synchronous actions (see **References:** /FBSY/) are not supported.

Block search

EG commands are ignored in the case of block search.

Mode change

In the case of a mode change:

- the EG status remains unchanged
- the EG configuration is retained

RESET

For RESET:

- the EG status remains unchanged
- the EG configuration is retained

End of parts program

On end of a part program:

- the EG status remains unchanged
- the EG configuration is retained

Warm start and cold start

In the case of a warm start per HMI operation and cold start (POWER OFF/ POWER ON):

- the EG status **does not** remain unchanged
- the EG configuration **is not** retained

Violated synchronism conditions

If the synchronism conditions are violated, all axes are stopped. In this case, their positions are checked by the control up to the stop. Extended stop and retract (ESR) may be active in this situation, see Chapter "Extended Stop and Retract (ESR)".

Power-up conditions of EG

The EG may be powered up:

- at the current axis positions (EGON) or
- at the synchronized positions to be specified (EGONSYN)
- at synchronized positions to be specified with details of an approach mode (EGONSYNE)

Block change behavior

In the EG activation commands (EGON, EGONSYN, EGONSYNE), it can be specified for which condition (with respect to synchronism) the next block of the parts program is to be processed. Options:

- NOC: no conditions
- FINE: Sum of the difference between the setpoint and actual positions of all axes less than machine data:

  MD37210 $MA_COUPLE_POS_TOL_FINE
- COARSE: Sum of the difference between the setpoint and actual positions of all axes less than machine data:

  MD37200 $MA_COUPLE_POS_TOL_COARSE
- IPOSTOP: When the specified end positions of the axes is reached

5.4.3 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".

5.4.4 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 identifier 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 identifier 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 synchronization"

"COARSE": Block change is performed in "Coarse synchronization"

"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": AbsoluteCo-ordinateNegative, Absolute measurement specification, rotary axis traverses in negative rotation direction

"ACP": AbsoluteCo-ordinatePositive, Absolute measurement specification, rotary axis traverses in positive rotation direction

"DCT": DirectCo-ordinateTime-optimized, Absolute measurement specification, rotary axis traverses time-optimized to programmed synchronized position

"DCP": DirectCo-ordinatePath-optimized, Absolute measurement specification, rotary axis traverses path-optimized to programmed synchronized position

LA_i: Axis identifier 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).

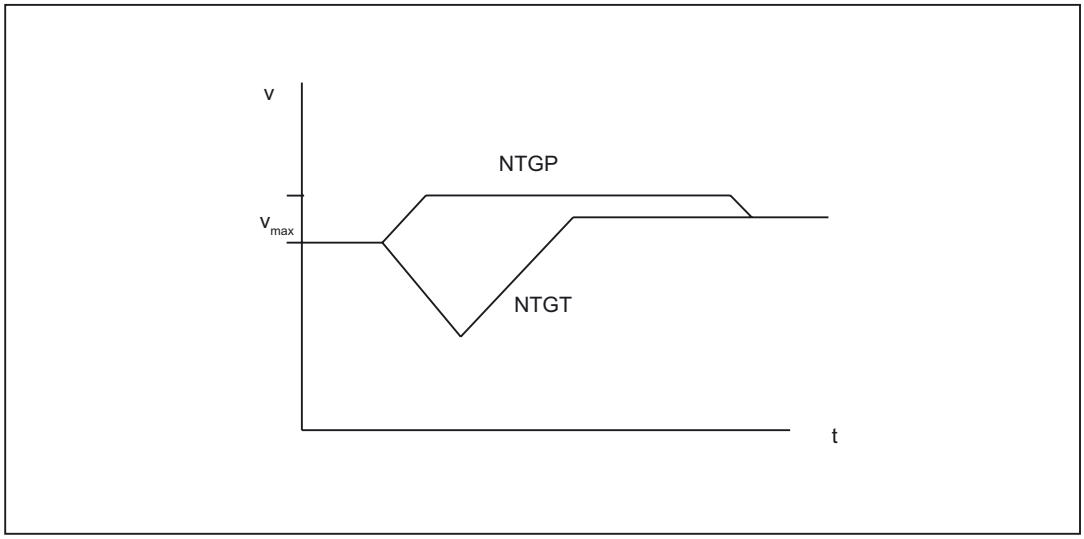


Figure 5-9 Reaching the next tooth gap, FA path-optimized (top) vs. time-optimized (bottom)

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).

5.4.5 Deactivating an EG axis group

Variant 1

There are different ways to deactivate an active EG axis grouping.

EGOFS(following axis)

The electronic gearbox 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 identifier must be programmed.

5.4.6 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.

5.4.7 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

5.4.8 Response to POWER ON, RESET, operating mode change, block search

No coupling is active after POWER ON.

The status of active couplings is not affected by RESET or operating mode switchover.

More detailed information on special states can be found under "Performance Overview of the Electronic Gearbox".

Block search for certain simulations

Certain active axis couplings can be simulated in the block search. As this does not apply to all possible coupling types, it is also possible to skip areas in which block searches cannot be made with an "automatic interrupt pointer". The electronic gear can be simulated for all block search types under the following conditions:

- Simulation always takes place with setpoint coupling.
- No cross-channel leading axes may be disabled.
- Axis movements for which all real positions are known to the NC.

References: /K1/ BAG, Channel, Program Operation, Reset Response

5.4.9 System variables for electronic gearbox

Application

The following system variables can be used in the parts program to scan the current states of an EG axis group and to initiate appropriate reactions if necessary:

Table 5-1 System variables, R means: Read access possible

Name	Type	Access		Preprocessing stop		Meaning, value	Cond. Index
		Parts program	Sync act.	Parts program	Sync act.		
$AA_EG_ TYPE[a,b] (ab SW 5.2)	INT	R		R		Type of coupling: 0: Actual value 1: Setpoint value coupling	Axis identifier a: Following axis b: Leading axis
$AA_EG_ NUMERA[a,b] (from SW 5.2) **------** **(SW 6 and higher)**	REAL	R		R		Numerator of coupl. factor KF KF = numerator/denominator preset: 0 ---- Number of curve table when $AA_EG_DENOM[a,b] is 0.	Axis identifier a: Following axis b: Leading axis

Name	Type	Access		Preprocessing stop		Meaning, value	Cond. Index
		Parts program	Sync act.	Parts program	Sync act.		
$AA_EG_ DENOM[a,b] (from SW 5.2) ----- **(SW 6 and higher)**	REAL	R		R		Denominator of coupl. fact. KF KF = numerator/denominator preset: 1 Denominator must be positive. ---- Denominator is 0, if instead of the numerator $AA_EG_NUMERA[a,b] the number of a curve table is specified.	Axis identifier a: Following axis b: Leading axis
$AA_EG_ SYN[a,b] (from SW 5.2)	REAL	R		R		Synchronized position for specified leading axis Default: 0	Axis identifier a: Following axis b: Leading axis
$AA_EG_ SYNFA[a] (from SW 5.2)	REAL	R		R		Synchronized position for specified following axis Default: 0	Axis identifier a: Following axis
$P_EG_BC[a]	STRING	R		R		Block change criterion for EG activation calls: EGON, EGONSYN: "NOC": immediately "FINE": Synchronoous traverse fine "COARSE": Synchronous traverse coarse "IPOSTOP": Setpoint-based synchronism	Axis identifier a: Following axis
$AA_EG_ NUM_LA[a]	INT	R		R		Number of leading axes defined with EGDEF. 0 if no axis has been defined as a following axis with EGDEF.	Axis identifier a: Following axis
$AA_EG_ AX[n,a]	AXIS	R		R		Axis identifier of leading axis whose index n has been specified.	Axis identifier n: Index of leading axis in the EG coupling 0 ... 4 a: Following axis
$AA_EG_ ACTIVE[a,b] (from SW 5.2)	BOOL	R		R		Determine power-on state of a leading axis: 0: Switched off 1: activated	Axis identifier a: Following axis b: Leading axis
$VA_EG_ SYNCDIFF[a]	REAL	R	R	R		Actual value of synchronism difference. A comparison of machine data MD37200 $MA_COUPLE_POS_TOL_COARSE and MD37210 $MA_COUPLE_POS_TOL_FINE produces interface signals.	Axis identifier a: Following axis

5.5 Generic coupling

5.5.1 Product brief

5.5.1.1 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-ordiante system) is programmable.
- Certain coupling properties can also be programmed with synchronous actions.

  References:
  /FBSY/ 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 "Adaptive cycles (Page 307)").

5.5.1.2 Preconditions

CP Versions

Generic coupling is available in a basic version as part of the NCK software and the four optional versions CP_STATIC, CP-BASIC, CP-COMFORT and 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 5-2 Scaling of the number of simultaneously permitted coupling modules

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	Synchronized spindle with 1:1 coupling	-	1	-	-	-
C	o./u. Synchronous spindle/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 "plain"	-	-	-	1	8
E	o./u. Electronic gearbox o./u. Free generic coupling	-	-	-	-	5

o./u. stands for over/under

Table 5-3 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
-		+	+	+	From part program and synchronous actions
-		-	-	+	Cascading permitted
BCS		BCS / MCS	BCS / MCS	BCS / MCS	Co-ordinate reference (default): CPFRS="BCS")
Synchronized 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
	-				Cascading permitted
	MCS				Co-ordinate 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
		-	-	-	Cascading permitted
		MCS	MCS	MCS	Co-ordinate 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
		-	-	+	Cascading permitted
		BCS / MCS	BCS / MCS	BCS / MCS	Co-ordinate reference (default): CPFRS="BCS")

Type A	Type B	Type C	Type D	Type E	
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
			-	+	Cascading permitted
			BCS / MCS	BCS / MCS	Co-ordinate reference (default): CPFRS="BCS")
			-	+ (max. 2x)	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
			-	+	Cascading permitted
			BCS / MCS	BCS / MCS	Co-ordinate reference (default): CPFRS="BCS")
			-	+	Non-linear coupling law (CPLCTID) permitted

[1)] Refer to "Coupling types (CPSETTYPE) (Page 308)".

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

Utilization of the "CP-EXPERT" option requires the application of:

- Systems with more than 6 axes
- NCUs ≧ NCU572 or NCU720

5.5.2 Basics

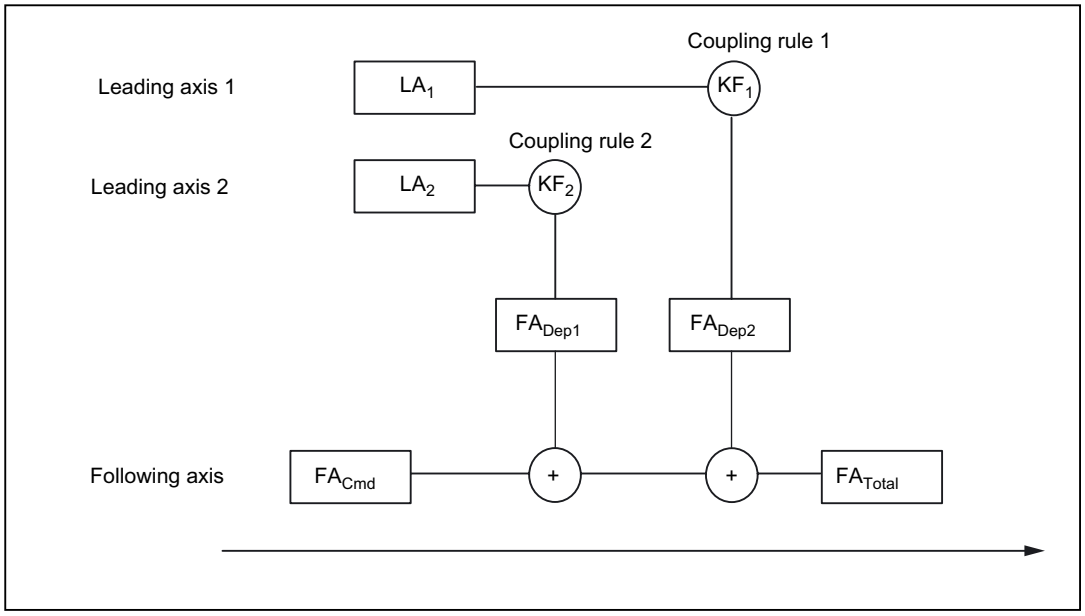
5.5.2.1 Coupling module

With the aid of a coupling module, the motion of one axis, (→ following axis), can be interpolated depending on other (→ 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.

5.5.2.2 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 (CPSETTYPE="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
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 at part program start under 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

Keyword	Coupling characteristics / meaning	Default setting (CPSETTYPE="CP")
CPSYNCOP	Threshold value of position synchronism "Coarse"	MD37200
CPSYNFIP	Threshold value of position synchronism "Fine"	MD37210
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
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 (refer to "Coupling types (Page 308)").

5.5.2.3 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 preprocessing stop.
$**A**A_CP	Reading of the current state of the coupling module (across channels).	Use in the part program and in synchronous actions is possible. Generates an implicit preprocessing stop when used in the part program.

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 (refer to "System variables (Page 341)").

For a detailed description of system variables, refer to:
References:
/PGA1/ List Manual, System Variables

5.5.3 Creating/deleting coupling modules

5.5.3.1 Creating a coupling module (CPDEF)

An axial coupling module is created through the definition of the following axis.

Programming

Syntax:	CPDEF= (<following axis/spindle>)	
Identifiers:	**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 identifiers in the channel

Example:

Programming	Comment
CPDEF=(X2)	; A coupling module is created with axis X2 as following axis.

Constraints

- The maximum number of coupling modules is limited (refer to "Preconditions (Page 263)").
- The application of CPDEF to an already created coupling module is possible and will not result in an alarm being generated.

5.5.3.2 Delete coupling module (CPDEL)

A coupling module created with CPDEF can be deleted with CPDEL.

Programming

Syntax:	CPDEL= (<following axis/spindle>)
Identifiers:	**C**oup**l**ing **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 identifiers in the channel

Example:

Programming	Comment
CPDEL=(X2)	; Deletion of the coupling module with following axis X2.

Constraints

- The switch command CPDEL results in a preprocessing stop with active coupling.

  Exception: No preprocessing stop occurs in CPSETTYPE="COUP".
- 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.

5.5.3.3 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>)`

Identifiers: **C**oup**l**ing **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 identifiers 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>)`

Identifiers: **C**oup**l**ing **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 identifiers in the channel

Example:

Programming	Comment
`CPDEF=(X2) CPLA[X2]=(X1)`	`; Definition of leading axis X1 for following axis X2.`

Constraints

- `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 (refer to "Preconditions (Page 263)").
- 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.

5.5.3.4 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>)

Identifiers: **C**oup**l**ing **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 identifiers 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>)
Identifiers:	**C**oup**l**ing **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 identifiers in the channel

Example:

Programming	**Comment**
CPDEL=(X2) CPLA[X2]=(X1)	; Deletion of leading axis X1 of the coupling to following axis X2.

Constraints

- 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.

5.5.4 Switching coupling on/off

5.5.4.1 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 topic "Programming Coupling Characteristics").

Without programming, a coupled motion group or a synchronous spindle pair becomes effective based on a setpoint coupling (default setting for CPSETVAL) with the coupling rule 1:1 (default setting for CPLNUM/CPLDEN).

Programming

Syntax:	CPON= (<following axis/spindle>)
Identifiers:	Coupling 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 identifiers in the channel

Example:

Programming	Comment
CPON=(X2)	; Activation of coupling of the following axis X2.

Constraints

Application of CPON to an already active coupling results in a neurosynchronization. If applicable, changed coupling properties become effective as a result. Any lost synchronization (for example, following axis was in tracking mode) is restored.

5.5.4.2 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>)
Identifiers:	Coupling Off
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 identifiers in the channel

Example:

Programming	Comment
CPOF=(X2)	; Deactivation of coupling of following axis X2.

Constraints

- 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.

5.5.4.3 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>)
Identifiers:	**C**oup**l**ing **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 identifiers in the channel

Example:

Programming	**Comment**
CPLON[X2]=(X1)	; The coupling of leading axis X1 to following axis X2 is activated.

Constraints

CPON can be programmed in synchronous actions.

5.5.4.4 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**oup**l**ing **L**ead Axis **Off**
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.

5.5.4.5 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) or CPLON[X2]=(X1) ... CPOF=(X2)	; Creates a coupling module for following axis X2 with leading axis X1 and activates the coupling module. ; 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.

5.5.5 Programming coupling characteristics

5.5.5.1 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>`

Identifiers: **C**oup**l**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>`

Identifiers: **C**oup**l**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 tables

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>
Identifiers:	**C**oup**l**ing **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.

Constraints

- 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 "Requirements (Page 263)").

5.5.5.2 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"	**C**o**m**man**d**ed **Pos**ition	Setpoint value coupling
"CMDVEL"	**C**o**m**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.

5.5.5.3 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**oup**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.

5.5.5.4 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**oup**l**ing **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`.

5.5.5.5 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>

Identifiers: **C**oup**l**ing **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 position 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.

Constraints

- 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 (refer to "Following Axis Position on Switch off (Page 294)").
- 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.

5.5.5.6 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**oup**l**ing **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 axis 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 following 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.`

5.5.5.7 Synchronization mode (CPFMSON)

Synchronization mode determines synchronization behavior during switch-on of the coupling.

Programming

Syntax:	CPFMSON[FAx]= "<synchronization mode>"		
Identifiers:	**C**oup**l**ing **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 If 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.
"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.`

5.5.5.8 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**oup**F**ollowing **M**ode **On**

Functionality: Defines the behavior of the following axis/spindle during switch-on of the coupling.

Power-on 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.
```

5.5.5.9 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**oup**l**ing **F**ollowing **M**ode **Off**

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.`

5.5.5.10 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 position 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 following axis.

Constraints

- 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 "Synchronized Position of the Following Axis on Switch-on (Page 288)").
- 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.

5.5.5.11 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**oup**l**ing **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 308)".

5.5.5.12 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**oup**l**ing **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 308)".

5.5.5.13 Status during part program start in search run via program test (CPMPRT)

At part program start during 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>`

Identifiers: **C**oup**l**ing **M**ode **P**rogram **T**est

Functionality: Defines the behavior of a coupling at part program start during search run via program test.

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
CPMPRT[X2]="ON"             ; At part program start during search run via program
                              test, coupling to following axis X2 is switched on.
```

Constraints

- The coupling characteristics set with `CPMPRT` is retained until the coupling module is deleted with (`CPDEL`).
- If `CPMPRT="NONE"` is set, then the response at part program start during 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 at part program start during 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 the control default settings in case of NC START)

  → See Section "Defaults" in "Coupling Types (CPSETTYPE) (Page 308)".

5.5.5.14 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>`

Identifiers: **C**oup**l**ing **L**ead **I**n **T**ranslation 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>`

Identifiers: **C**oup**l**ing **L**ead **I**n **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>`

Identifiers: **C**oup**l**ing **L**ead **O**ut **T**ranslation 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>`

Identifiers: **C**oup**l**ing **L**ead **O**ut **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.`

NOTICE
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.

5.5.5.15 Synchronism monitoring (CPSYNCOP, CPSYNFIP, CPSYNCOV, CPSYNFIV)

Synchronism monitoring

The setpoint and actual value synchronism of the coupling group is monitored in each interpolation cycle. The synchronism monitoring is activated as soon as **Synchronism 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 programmed threshold values:

Command	Meaning
CPSYNCOP[FAx]	Threshold value of position synchronism "Coarse"
CPSYNFIP[FAx]	Threshold value of position synchronism "Fine"
CPSYNCOV[FAx]	Threshold value of speed synchronism "Coarse"
CPSYNFIV[FAx]	Threshold value of speed synchronism "Fine"

The current synchronism difference can be read out with the following CP system variables.

System variable	Meaning
$AA_SYNCDIFF[FAx]	Synchronism difference of the setpoint
$VA_SYNCDIFF[FAx]	Synchronism difference of the actual value

Note

The synchronism differences are signed and enable the advance or delay of the following axis to be determined.

Status of the coupling during synchronism

State	Description
not synchronized	As long as the synchronism difference is greater than CPSYNCOP or CPSYNCOV, the coupling group is considered as not synchronous.
"Coarse" synchronism reached	The synchronism difference has reached the threshold value for the "Coarse" position synchronism (CPSYNCOP) or "Coarse" speed synchronism (CPSYNCOV). In case the actual value-based synchronism difference leads to the following NC/PLC interface signal: DB31, ... DBX98.1 (coarse synchronism)
"Fine" synchronism reached	The synchronism difference has reached the threshold value for the "Fine" position synchronism (CPSYNFIP) or "Fine" speed synchronism (CPSYNFIV). In case the actual value-based synchronism difference leads to the following NC/PLC interface signal: DB31, ... DBX98.0 (fine synchronism)

The status of the coupling can be read out with the following CP system variables:

System variable	Meaning	
$AA_SYNC[FAx]	Status of the coupling	
	Value	State
	0	not synchronized
	1	"Coarse" synchronism reached
	2	"Fine" synchronism reached

Signal reaction

Synchronism signals themselves do not stop the involved axes, but can release them via synchronized action or NC/PLC interface signals (refer to Chapter "Extended Shutdown/Retraction (ESR)").

Programming

Threshold value of position synchronism "Coarse"

Syntax: `CPSYNCOP[FAx]= <value>`

Identifiers: **Cou**pling **Syn**chronous Difference **Co**arse **P**osition

Functionality: Defines the threshold value for the "Coarse' position synchronism.

Value: Type: REAL

Threshold value of position synchronism "Fine"

Syntax:	`CPSYNFIP[FAx]= <value>`
Identifiers:	**C**oup**l**ing **Syn**chronous Difference **Fi**ne **P**osition
Functionality:	Defines the threshold value for the "Fine' position synchronism.
Value:	Type: REAL

Threshold value of speed synchronism "Coarse"

Syntax:	`CPSYNCOV[FAx]= <value>`
Identifiers:	**C**oup**l**ing **Syn**chronous Difference **Co**arse **V**elocity
Functionality:	Defines the threshold value for the "Coarse'' speed synchronism.
Value:	Type: REAL

Threshold value of speed synchronism "Fine"

Syntax:	`CPSYNFIV[FAx]= <value>`
Identifiers:	**C**oup**l**ing **Syn**chronous Difference **Fi**ne **V**elocity
Functionality:	Defines the threshold value for the "Coarse' velocity synchronism.
Value:	Type: REAL

Example

```
Programming                                          Comment
CPDEF=(S2) CPLA[S2]=(S1)                               Definition of a spindle
                                                       coupling:
                                                       Leading spindle S1 with
                                                       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 synchronism
                                                       are set to 0.5 ("coarse")
                                                       and 0.25 ("fine").
...
```

Constraints

- When considering the synchronism difference, an active coupling cascade is not taken into account. This means that: 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 synchronism difference. This synchronism difference therefore does not show the total synchronism 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 synchronism monitoring of the actual value. In this case, the modeled actual values (according to MD setting) are used.

5.5.5.16 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>`

Identifiers: **C**oup**l**ing **M**ode **Brake**

Functionality: `CPMBRAKE` is a bit-coded CP keyword, which defines the braking behavior of the following axis FAx with reference to the following events:

Bit	Event
0	Approaching the NC/PLC interface signal: DB31, ... DBX4.3 (feed stop / spindle stop) **Note:** Bit 0 is significant only to "Free programming" coupling type (`CPSETTYPE=CP`).

Value	Meaning
0	The NC/PLC interface signal: DB31, ... DBX4.3 (feed stop / spindle stop) has no influence on the coupling.
1	The NC/PLC interface signal: DB31, ... DBX4.3 (feed stop / spindle stop) brakes the coupling group.

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

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" 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" should brake the
                                   coupling group.
...
```

5.5.6 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)
```

Constraints

- The availability of cascading is option-based (refer to "Preconditions (Page 263)").
- 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).

5.5.7 Compatibility

5.5.7.1 Adaptive cycles

Adaptive cycles

The provision of adaptive cycles as fixed component of the NCK 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

Coupling commands	Adaptive cycle
EGOFC	Cycle715
EGOFS	Cycle716
EGDEL	Cycle717

Memory 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 NCK software, a log file is saved in the "CST" directory (Changelog), indicating necessary changes of the adaptive cycles.

5.5.7.2 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` oder `CPDEF`/`CPLDEF`), the keyword `CPSETTYPE` needs to be used also.

Programming

Syntax: `CPSETTYPE[FAx]`= <value>

Identifiers: **Coup**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	-	-	Not active	-
CPSETVAL	CMDPOS	CMDPOS	CMDPOS	CMDPOS	CMDPOS
CPFRS	BCS	BCS	BCS	BCS	MCS
CPBC	NOC	NOC	NOC	NOC	IPOSTOP
CPFPOS + CPON	Not set	-	0.0	Not active	Not active
CPFPOS + CPOF	Not set	-	-	-	Not active

	Coupling type				
CPFMSON	CFAST	CFAST	CCOARSE	CFAST	CFAST
CPFMON	STOP	VL[1]: STOP HL[2]: CONT	VL[1]: STOP HL[2]: CONT	STOP	CONT
CPFMOF	STOP	VL[1]: STOP HL[2]: CONT	CONT	STOP	CONT
CPLPOS + CPON	Not set	-	0.0	Not active	Not active
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
CPSYNFIP	MD37210	-	-	MD37210	MD37210
CPSYNCOV	MD37220	-	-	-	MD37220
CPSYNFIV	MD37230	-	-	-	MD37230
CPMBRAKE	1	-	-	-	-

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 specified characteristics depends on the available version (see "Preconditions (Page 263)").

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.

Constraints

- 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		
CPCTID	Alarm 16686			Alarm 16686
CPSETVAL	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	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 (∑ LA)	Alarm 16672 with ∑ LA > 2	Alarm 16672 with ∑ LA > 1	Alarm 16672 with ∑ LA > 5	Alarm 16672 with ∑ LA > 1
Following axis type				Alarm 14092 with axis

5.5.7.3 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, Extended Functions; Synchronous spindles (S3);
Chapter "Programming of synchronous spindle couplings"

Programming

Syntax:	CPRES= (<following spindle>)
Identifiers:	**C**oup**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 identifiers 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.

Constraints

- 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.

5.5.8 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. Constraints and properties still apply to axis change. The axis replacement via channel axes is released again after deactivating the coupling module.

References:
/FB2/ 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.

5.5.9 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.

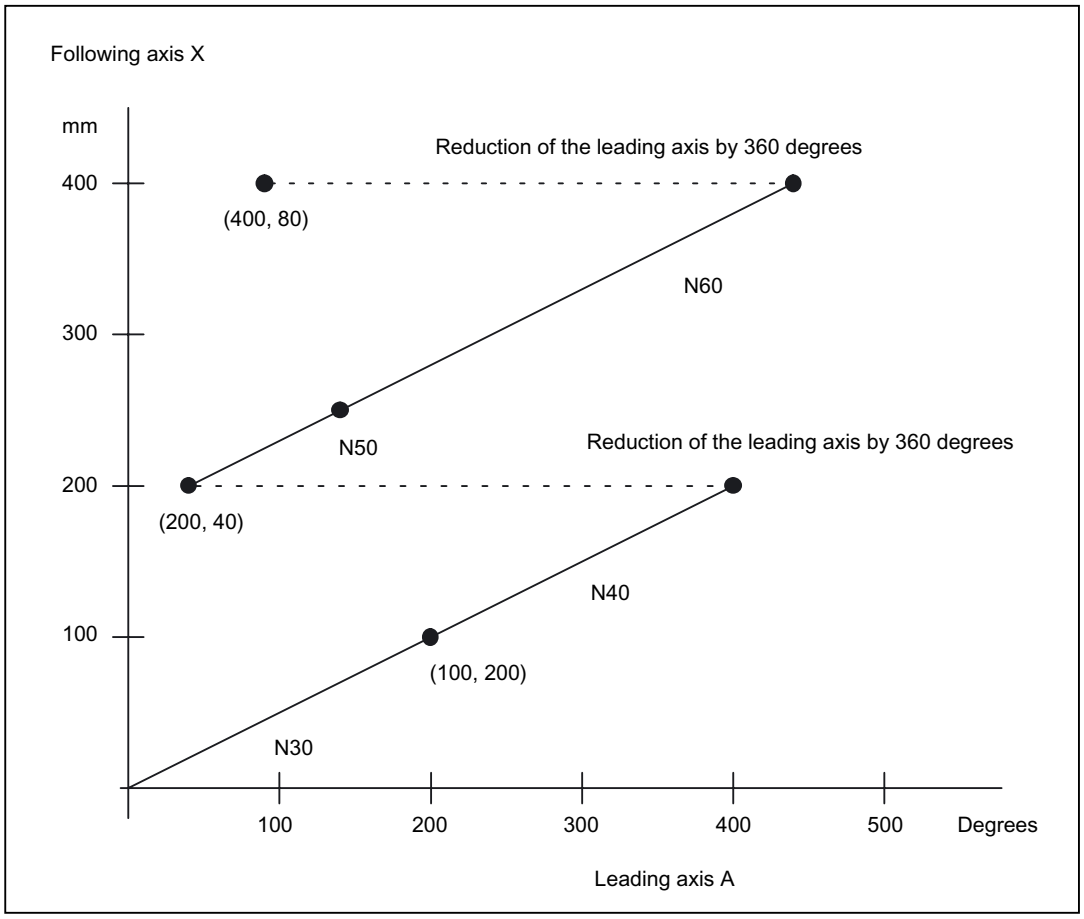


Figure 5-10 Example: Modulo reduced rotary axis to linear axis

(...) Position indication for X, A

5.5.10 Behavior during POWER ON, RESET ...

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.

5.5.11 Disturbance characteristic

5.5.11.1 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.

5.6 Dynamic response of following axis

5.6.1 Parameterized dynamic limits

The dynamics of the following axis is limited by the following MD values:

MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

MD32300 $MA_MAX_AX_ACCEL (Maximum axis acceleration)

5.6.2 Programmed dynamic limits

5.6.2.1 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	Description
`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 paramterized dynamic limits (MD32000 and MD32300) are to be considered:

Range of values	Description
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 synchronized difference. The monitoring from leading to following value may exceed the allowed range for extended periods.
Dynamic increase	Shortens the synchronized 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 BRISKAis available for the following axis, i.e., abrupt axis acceleration. Acceleration modes SOFTA and DRIVEAare not available for the following axes described.

Furthermore, it is also possible to configure the positions controller as a PI controller.

⚠ CAUTION

This option can only be used in conjunction with servo trace and with the appropriate technical knowledge of the control.

References:
/IAD/ Commissioning Manual 840D/611D
/FB1/ Function Manual, Basic Functions;
Velocities, Setpoint/Actual Value Systems, 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	Description
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:
/FBSY/ Function Manual, Synchronized Actions

5.6.2.2 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)
...
```

5.6.2.3 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.

5.7 Boundary conditions

5.7.1 Coupled motion

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.

5.7.2 Curve tables

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.

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.

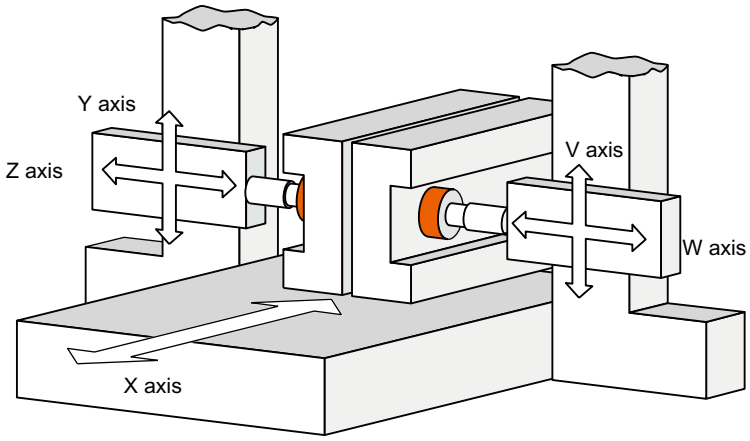
When attempting to reposition without release of the axis, the message "Wait: Feed stop active" is displayed and the processing of the part program is not continued.

5.8 Examples

5.8.1 Coupled motion

Application Example: Two-sided machining

	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



Example 1

Example of an NC part program for the axis constellation shown in the figure:

```
TRAILON(V,Y,1)          ; Activation of 1st coupled axis group
TRAILON(W,Z,-1)         ; Activation of 2nd coupled axis group
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.

```
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 of 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)
```

5.8.2 Curve tables

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)
N21 PO[X]=(147.754,-0.116,0.103) PO[Y]=(3.359,-0.188,0.277)
```

```
%_N_TAB_1_NOTPERI_MPF
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)
N230 M30
```

5.8.3 Electronic gear for gear hobbing

5.8.3.1 Example of 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.

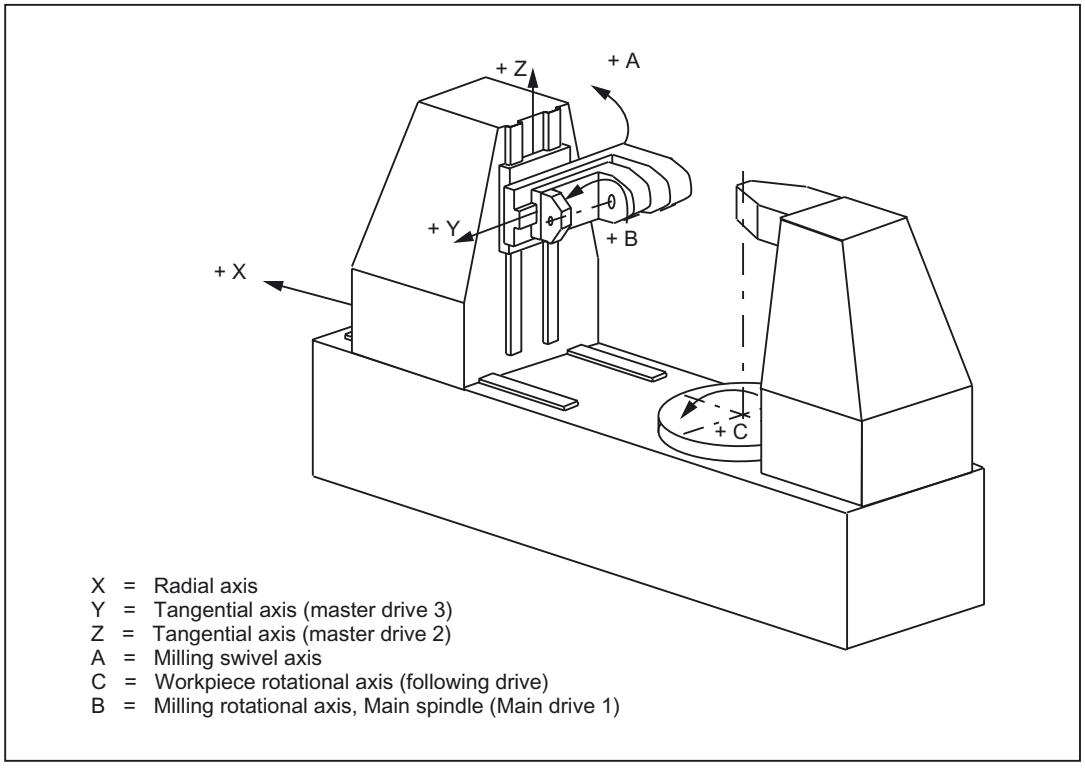
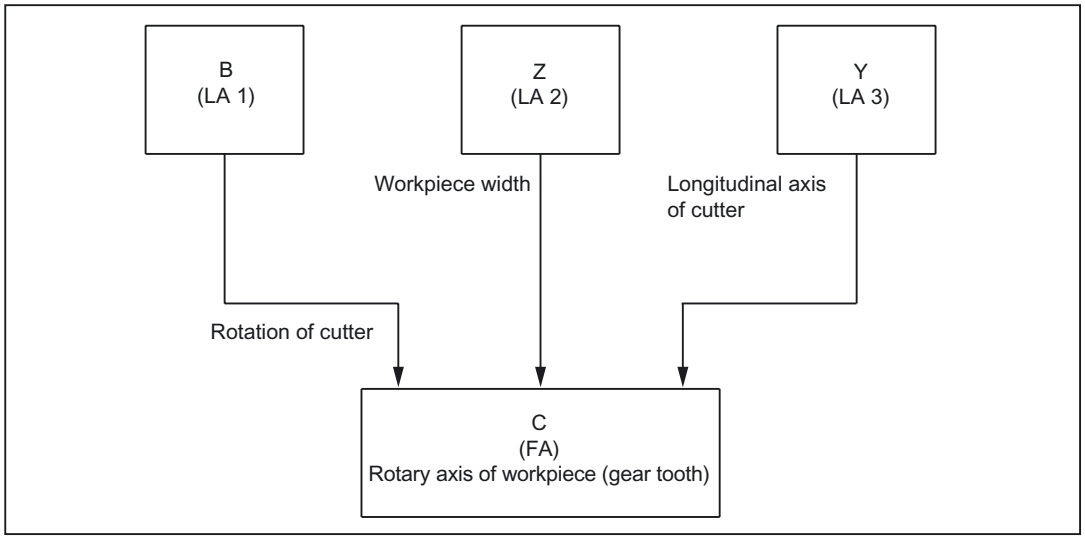


Figure 5-11 Definition of axes on a gear hobbing machine (example)

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:

$n_c = n_b * (z_0 / z_2) + v_z * (u_{dz} / z_2) + v_y * (u_{dy} / z_2)$

n_c: Rotational speed of workpiece axis (C)

n_b: Rotational speed of milling spindle (B)

z_0: Gear number of 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 the parts 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.

$u_{dz} = (\sin_{\beta^\circ} / m_n * \pi) * 360$

[degrees / mm]

$u_{dy} = (\cos_{\gamma^\circ} / m_n * \pi) * 360$

[degrees / mm]

with:

m_n = Normal module (in mm)

β° = Incline angle of gear

γ° = Pitch angle of hobbing machine

Extract from parts program:

```
; Definition of EG axis group with
; setpoint coupling (1) from B, Z, Y to C (following axis)
EGDEF(C, B, 1, Z, 1, Y, 1)
; Activate coupling
EGON(C, "FINE", B, z0, z2, Z, udz, z2, Y, udy, z2)
```

5.8.3.2 Extended example with non-linear components

Introduction

The following example extends the example in Figure "Axis Definition of a Hobbing Machine" in Chapter "Example of Linear Couplings" 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.

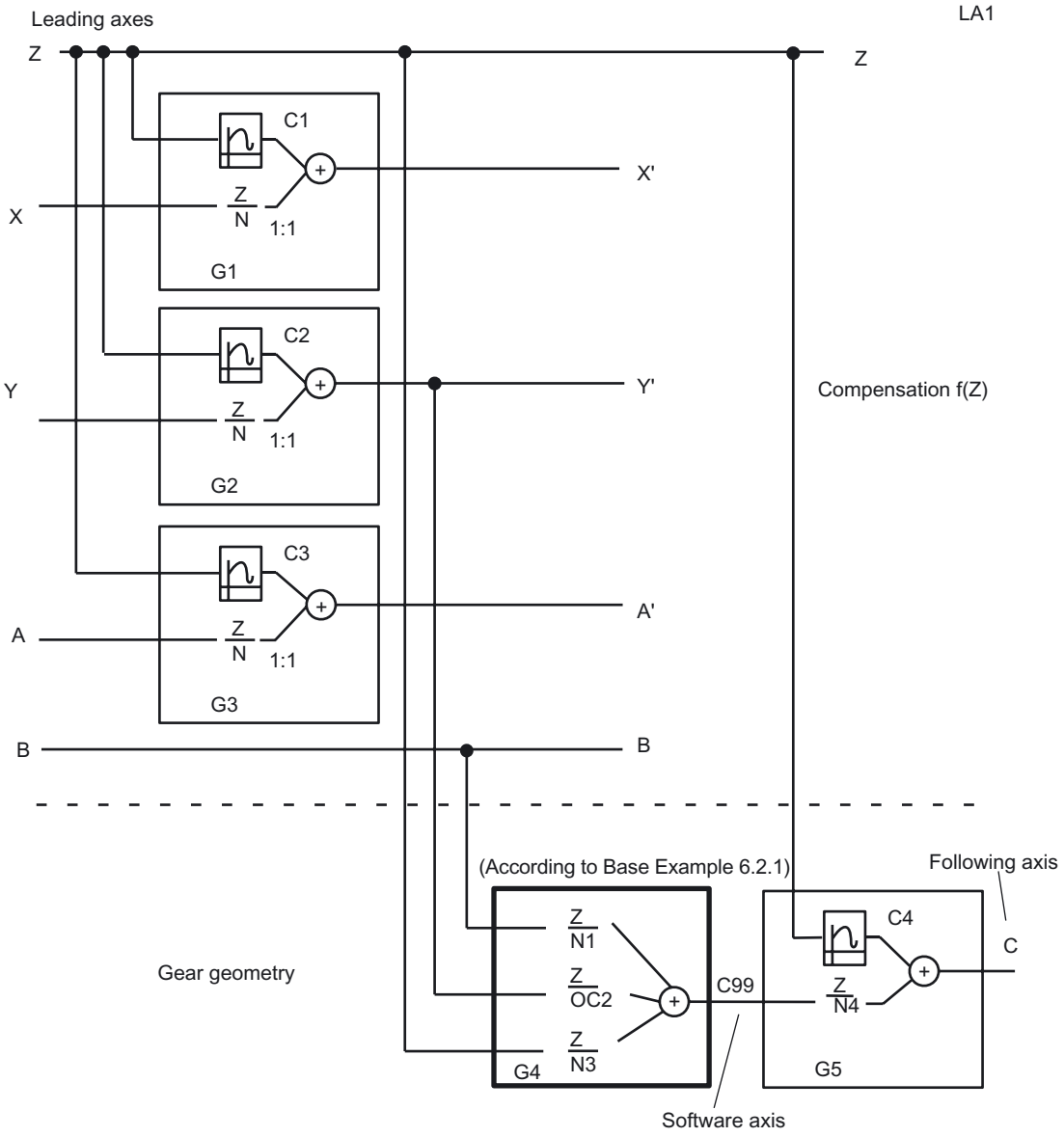


Figure 5-12 Extended example with non-linear machine fault compensation and non-linear components on the tooth geometry

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.

```
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",     ; Path switch-on via C1
                <SynPosX>, Z,
                <SynPosX_Z>, 1, 0)
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,            ; Path switch-on via C2
                "COARSE",
                <SynPosY>, Z,
                <SynPosY_Z>, 2, 0)
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",    ; Path switch-on via C3
                <SynPosA>, Z,
                <SynPosA_Z>, 3, 0)
```

```
; 1st 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",   ; Switch-on of leading axis B
                 <SynPosC99>, B,
                 <SynPosC99_B>, 18,
                 2, &
                 Y, <SynPosC99_Y>,     ; Switch-on of leading axis Y
                 R1 * π, 1, &
                 Z, <SynPosC99_Z>,     ; Switch-on of leading axis Z
                 10, 1)
                                       ; "&" character means: command continued in
                                       next line, no LF nor comment permissible in
                                       program
; 2nd gear stage
N900             EGDEF(C, C99, 1, Z,   ; Declaration of following C99 of step 1 as
                 1)                    leading axis of step 2,
                                       ; Setpoint value coupling
N910                                   ; Declaration of path via C4, setpoint
                                       coupling
N920             EGONSYN(C, "NOC",     ; Switch-on of software axis C99
                 <SynPosC>, C99,
                 <SynPosC_C99>, 1,
                 1, &
                 Z, <SynPosC_Z>, 4,    ; and of leading axis Z via C4
                 0)
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 Chapter "Electronic Gearbox (EG)" is used, the following applies, as in the case of an offset:

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. Options of access to these system variables are described in SW 7.1 and higher:

References:
/PGA1/ 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                      ; Leading axis identifier
$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                      ; Leading axis identifier
$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                      ; Leading axis identifier
$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 factor_y
$AA_EG_DENOM[C99, Y] = 2                 ; Denominator for coupling factor_y
$AA_EG_TYPE[C99, Z] = 1                  ; Setpoint value coupling
$AA_EG_NUMERA[C99, Z] = R1 * π           ; Numerator for coupling factor_z
$AA_EG_DENOM[C99, Z] = 1                 ; Denominator for coupling factor_z
$AA_EG_TYPE[C99, B] = 1                  ; Setpoint value coupling
```

```
$AA_EG_NUMERA[C99, B] = 10                ; Numerator for coupling factor_b
$AA_EG_DENOM[C99, B] = 1                  ; Denominator for coupling factor_b
$P_EG_BC[C99] = "NOC"                     ; Block change criterion
$AA_EG_NUM_LA[C99] = 3                    ; Number of leading axes
$AA_EG_AX[0, C99] = Y                     ; Leading axis Y identifier
$AA_EG_AX[1, C99] = Z                     ; Leading axis Z identifier
$AA_EG_AX[2, C99] = B                     ; Leading axis B identifier
$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 factor_C99
$AA_EG_DENOM[C, C99] = 1                  ; Denominator for coupling factor_C99
$P_EG_BC[C] = "NOC"                       ; Block change criterion
$AA_EG_NUM_LA[C] = 2                      ; Number of leading axes
$AA_EG_AX[0, C] = Z                       ; Leading axis Z identifier
$AA_EG_AX[1, C] = C99                     ; Leading axis C99 identifier
$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
```

5.8.4 Generic coupling

5.8.4.1 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
                                     deactivated 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
                                             according to the coupling rule (coupling
                                             component) 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
                                             contribute 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
                                             axis 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)
```

5.8.4.2 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 5-13 Cycle 700 after adaption. Changes are indicated by a colored bar.

5.9 Data lists

5.9.1 Machine data

5.9.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
11410	SUPPRESS_ALARM_MASK	Mask for supporting special 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

5.9.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20112	START_MODE_MASK	Definition of control basic setting after run-up and RESET
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

5.9.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30130	CTRLOUT_TYPE	Setpoint output type
30132	IS_VIRTUAL_AX	Axis is virtual axis
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 synchronism"
37210	COUPLE_POS_TOL_FINE	Threshold value for "Fine synchronism"
37220	COUPLE_VELO_TOL_COARSE	Velocity tolerance "coarse"
37230	COUPLE_VELO_TOL_FINE	Velocity tolerance "fine"
30455	MISC_FUNCTION_MASK	Axis functions
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"

5.9.2 Setting data

5.9.2.1 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

5.9.3 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	Identifier for nth leading axis
$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 synchronism.
$VA_EG_SYNCDIFF	Synchronization 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	Compensation value for synchronous spindle coupling
$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

Identifier	Meaning
$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
$AA_CPLSETVAL	Coupling reference of the leading axis
$AA_CPLSTATE	Status 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	Identifier 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

Identifier	Meaning
$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
$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	Synchronization 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

5.9.4 Signals

5.9.4.1 Signals to axis/spindle

DB number	Byte.bit	name
31, ...	0.0-0.7	Feed offset
31, ...	1.3	Axis disable
31, ...	2.1	Controller enable
31, ...	4.0-4.2	Activate handwheel
31, ...	4.3	Feed stop
31, ...	26.4	Enable following axis overlay
31, ...	31.4	Synchronize following spindle
31, ...	31.5	Disable synchronization
31, ...	31.6	Track synchronism

5.9.4.2 Signals from axis/spindle

DB number	Byte.Bit	Name
31, ...	83.1	Limiting of differential speed
31, ...	83.5	Spindle in setpoint range, differential speed
31, ...	83.6	Speed limit exceeded, total speed
31, ...	83.7	Actual direction of rotation clockwise, total speed
31, ...	84.4	Synchronous mode
31, ...	98.0	Synchronism fine
31, ...	98.1	Synchronism coarse
31, ...	98.2	Actual value coupling
31, ...	98.4	Superimposed motion
31, ...	98.5	Velocity warning threshold
31, ...	98.6	Acceleration warning threshold
31, ...	99.0	Leading spindle active
31, ...	99.1	Following spindle active
31, ...	99.3	Following axis accelerated
31, ...	99.4	Synchronization in progress
31, ...	99.5	Maximum velocity reached
31, ...	99.6	Maximum acceleration reached

Setpoint Exchange (S9)

6

6.1 Brief description

Function

The "setpoint exchange" function is used in applications in which the same motor is used to traverse different machine axes.

Operating conditions

The function described below replaces the "Setpoint exchange" technology card function (TE5) for systems with NCK SW ≧ 7.1.

An option is required for the function.

Compatibility

Migration to NCK SW 7.1 requires adaptations to be made to machine data and the PLC user program.

6.2 Startup

The "setpoint exchange" function is required in applications in which a single motor needs to drive a number of axes/spindles such as, for example, on milling machines with special millheads. The spindle motor is operated as both a tool drive and a millhead orienting mechanism.

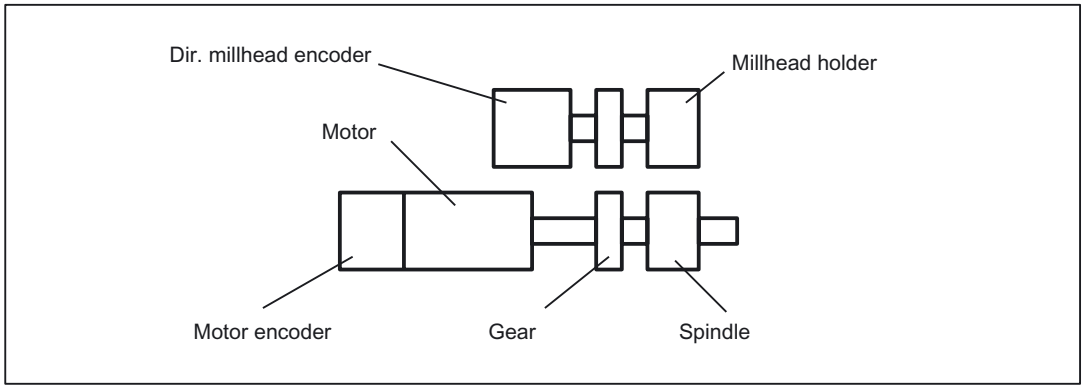


Figure 6-1 Example 1: 1 motor encoder, extra encoder for millhead

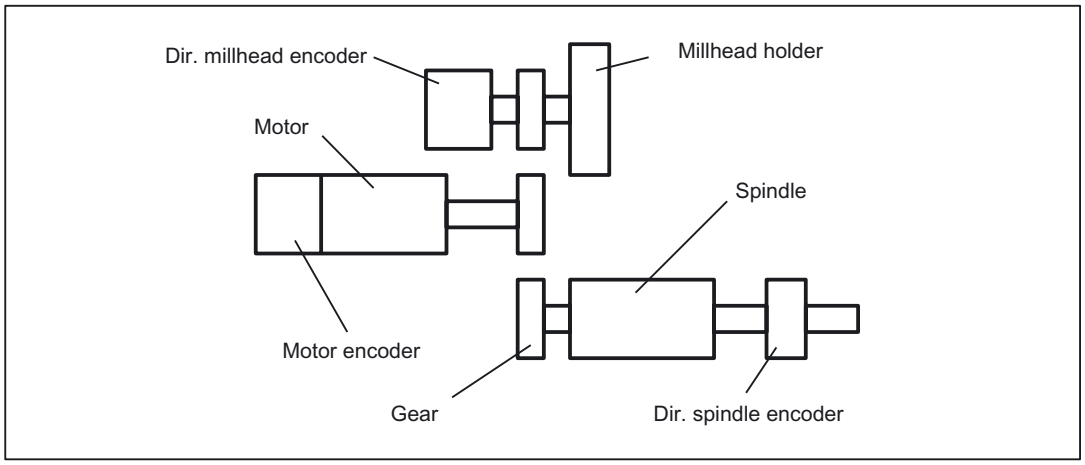


Figure 6-2 Example 2: 1 motor encoder, separate millhead encoder and spindle encoder

Configuration

Setpoint exchange enables a number of axes to use the same drive.

The same setpoint channel on this drive is assigned a number of times to define the axes participating in setpoint exchange. For this, the following machine data must be pre-assigned with the same logical drive number for every axis:

MD30110 $MA_CTRLOUT_MODULE_NR (setpoint assignment: module number)

Note

Alarm 26018 is output if the option is missing.

The encoder assignment is programmed in machine data:

MD30230 $MA_ENC_INPUT_NR
(actual-value assignment: Input on drive module/measuring circuit module)

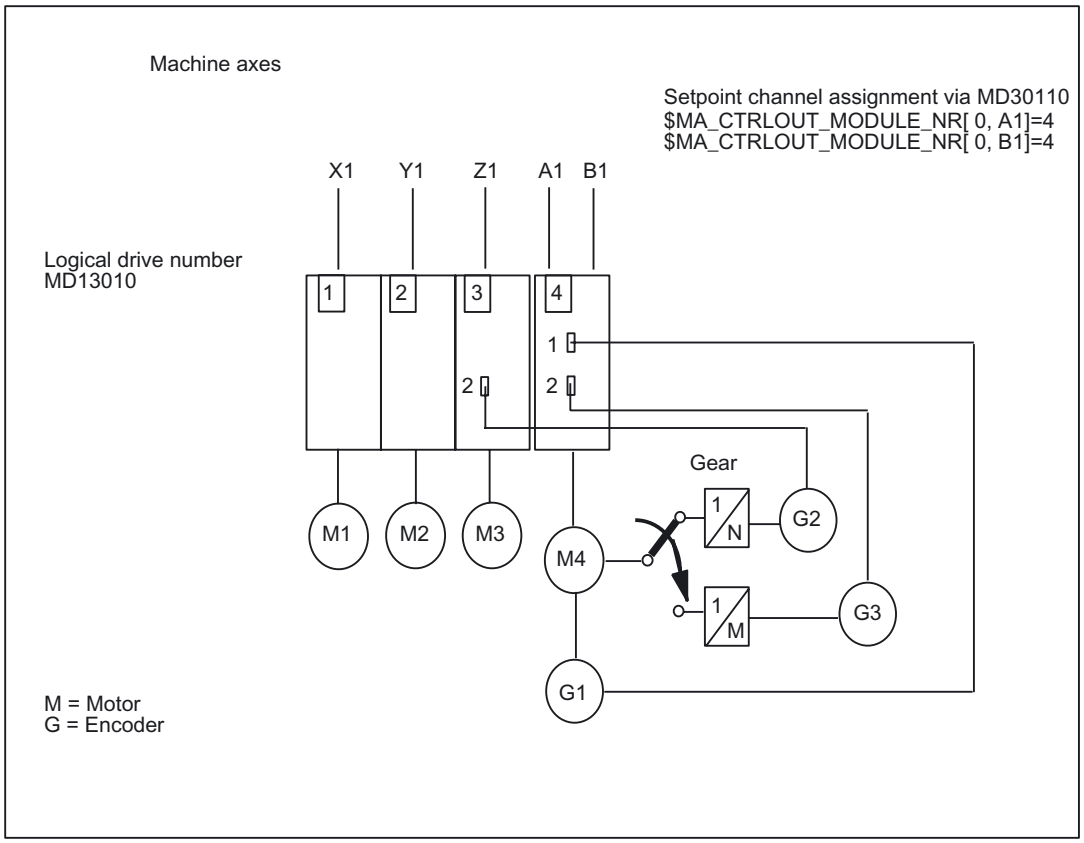


Figure 6-3 Setpoint exchange with 2 axes

Activation

The setpoint is exchanged and the corresponding interface signals are evaluated in the PLC user program.

Note

An existing PLC user program may need to be modified due to changes in the meaning of interface signals in comparison with the technology card solution.

Only one of the machine axes with the appropriate logical drive number may have control via the setpoint channel of the drive at any one time.

Requests to transfer drive control are sent using NC/PLC interface signal:

DB31, ... DBX24.5 (change setpoint output assignment)

The current status of drive control is displayed in the NC/PLC interface signal:

DB31, ... DBX96.5 (setpoint output assignment)

Access rights to the shared drive must be managed in the PLC user program.

Transfer conditions

- Axis standstill of all axes involved.
- Special functions such as reference point approach, measuring, travel to fixed stop, function generator, star/delta changeover, drive parameter set changeover are not active in the axis with drive control.
- No sign-of-life error and no faults pending on PROFIBUS drive.

The PLC interface provides constant information about the current state of the exchange. At any one time, only one axis has drive control DB31, ... DBX96.5=1.

During exchange, servo enables on all axes involved are automatically withdrawn by the control.

Axes without drive control are not in closed-loop control. Therefore, a brake control must be set up for vertical axes.

Special cases

If a number of transfer requests are made simultaneously, exchange will not take place. The last axis used remains in control of the drive. This is also the case if there are no transfer requests pending.

If there are no transfer requests during machine power-up, drive control is assigned to the first machine axis located with the same logical drive number. The interface signal DB31, ... DBX24.5 (change setpoint output assignment) must be set explicitly in advance to be able to traverse the axis. Logical drive numbers are scanned in ascending order.

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 ; Drive control during power-up

MD30110 $MA_CTRLOUT_MODULE_NR[0,AX5] = 4

6.3 Interface signals

Axis-specific signals

Despite assignment of an individual drive to several axes, the use NC/PLC interface signals remains unchanged.

This requires explicit access coordination in the PLC user program.

As the same drive is being used, the same status signals from DB31, ... DBB92-95 are displayed in all axes involved in the exchange.

However, control signals DB31, ... DBB20-21 must only be set in the axis with exclusive control of the drive. Servo enable DB31, ... DBX2.1 is effective only if the NC/PLC interface signal DB31, ... DBX24.5 (change setpoint output assignment) is set.

Axes without the corresponding drive control (DB31, ... DBX24.5 = 0 or
DB31, ... DBX96.5 = 0) are subject to functional restrictions and are therefore operated in follow-up mode. For this purpose, the servo enables are deleted automatically by the control.

Diagram of the PLC program

The following example illustrates a possible setpoint exchange sequence. The setpoint output assignment should take place once in advance via DB31, ... DBX24.5.

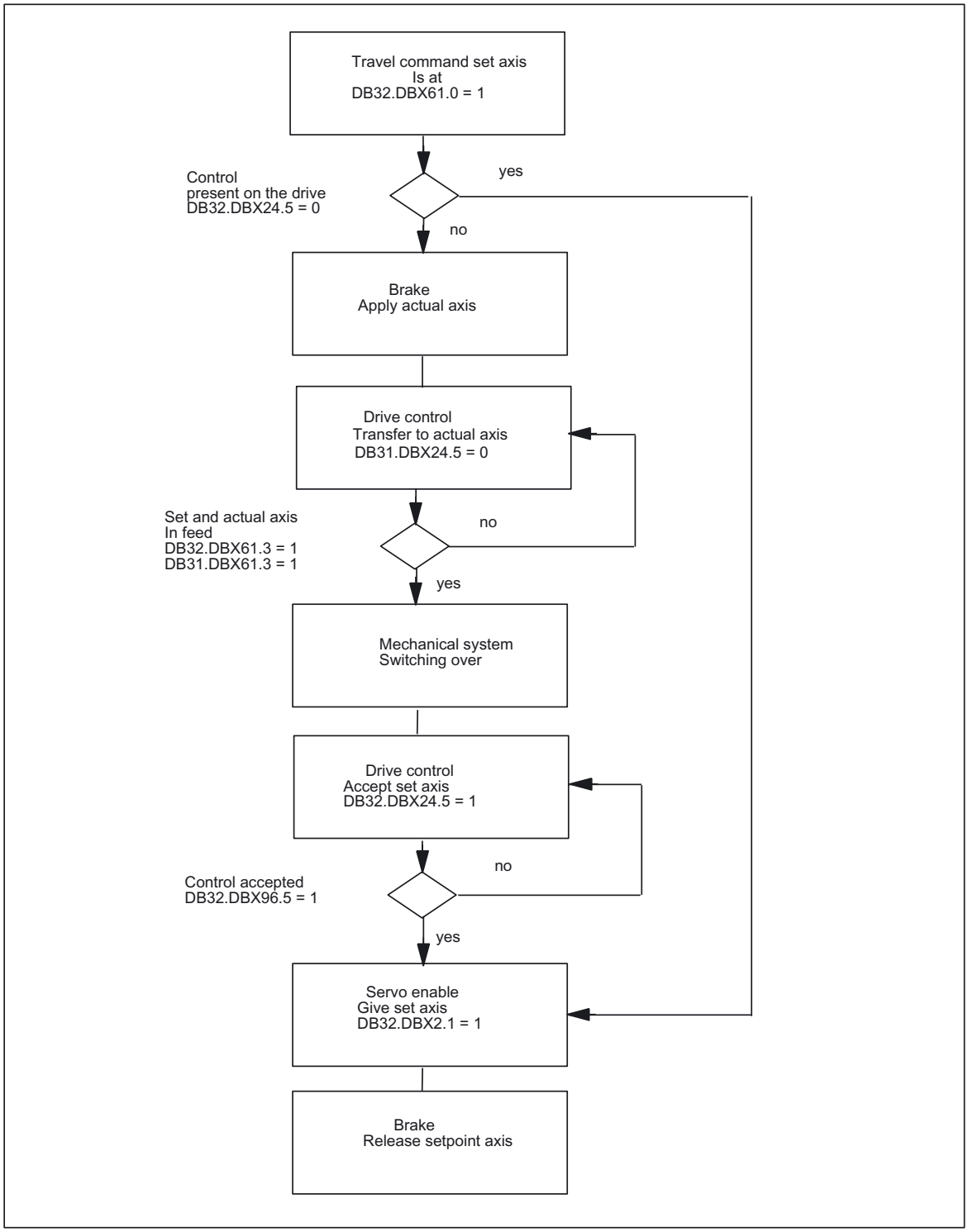


Figure 6-4 PLC-controlled sequence of a setpoint exchange between AX1 → AX2

6.4 Interrupts

Drive alarms are only displayed for axes with drive control

6.5 Position control loop

During setpoint exchange, the drive train and therefore the position control loop are isolated. In order to avoid instabilities, exchange only takes place at standstill and once all servo enables have been deleted.

The use of a single drive means that only one of the control loops can be closed at any one time. Axes without drive control are operated with open position controller and following positions.

6.6 Reference points

The use of load-side encoders does not affect the axial reference points of a setpoint exchange.

However, the mechanical reference to the load can be lost following setpoint exchange for a load-side position derived solely from the motor encoder. These types of axis must be referenced again after every setpoint exchange.

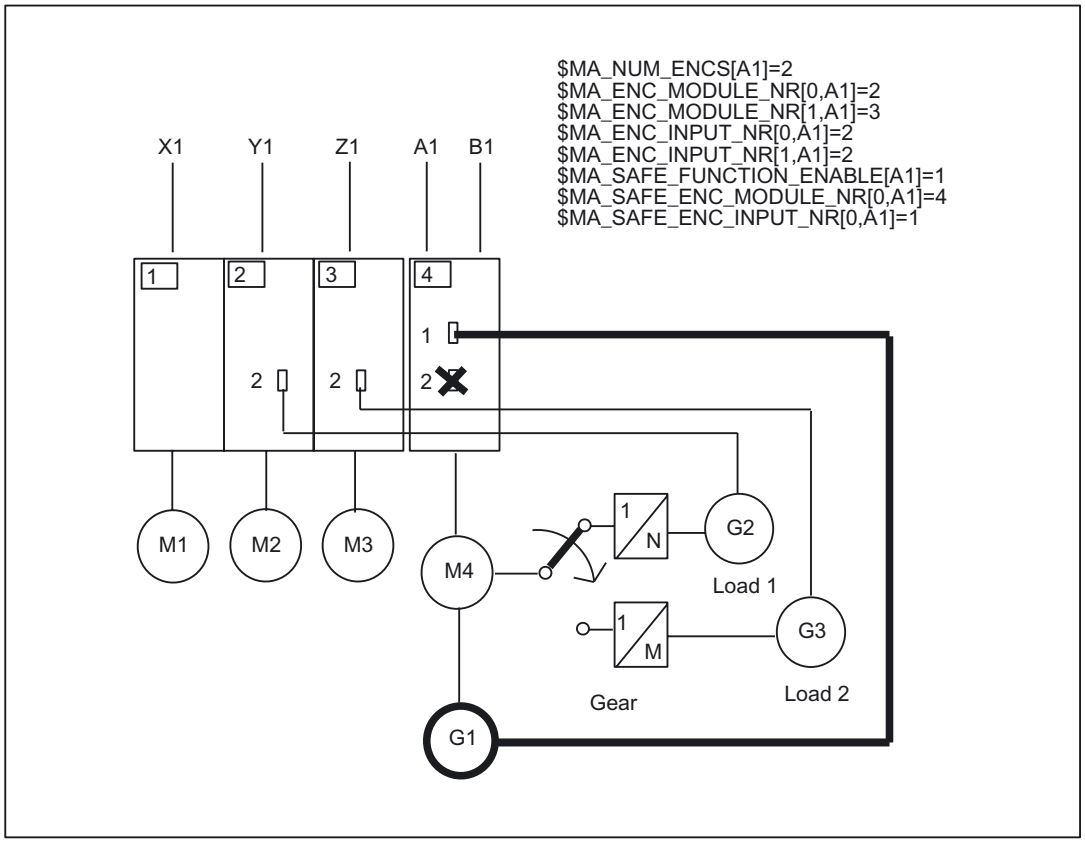


Figure 6-5 Setpoint exchange in conjunction with single-encoder safety integrated system

6.7 Differences in comparison with the technology card

The setpoint exchange implemented in NCK SW 7.1 and higher differs from the compile cycles solution described in TE5 as follows. These differences must be taken into account during installation and start-up and when creating the PLC user program:

- The machine data MD63750 is no longer required.
- The meanings of associated interface signals have changed. Therefore, PLC user programs must be updated accordingly.
- Alarms 70451 and 70452 are no longer used.
- Setpoint exchange with simulated axes (MD30130=0) is no longer supported.
- Known restrictions of the technology card function no longer apply.

6.8 Constraints

Availability

Setpoint exchange is available in SW 7.1 and higher.

MD30100

Setpoint exchange is only possible in conjunction with 611D and PROFIBUS drives with:

MD30100 $MA_CTRLOUT_SEGMENT_NR=**1. 5** or **6**.

All other settings generate alarm 26018.

"Parking" operating status

The "parking" operating state can only be exited using the axis with the drive checking function.

Service display drive

The "Drive Service Display" HMI diagnostics screen does not take into account changes in assignments between axis and drive.

SinuComNc

Setpoint exchange can only be started up via SinuComNc via the Expert List. A dialog is not supported.

Safety integrated

See
References:
/FBSI/ Function Description Safety Integrated (11.02 and later) for supplementary conditions for "Safety Integrated" in conjunction with setpoint exchange.

6.9 Data lists

6.9.1 Machine data

6.9.1.1 Axis/spindle-specific 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

Tangential Control (T3)

7

7.1 Brief description

Tangential control

The tangential control function belongs to the category of NC functions with coupled axes. It is characterized by the following features:

- There are two leading axes which are moved independently by means of normal traversing instructions (leading axes). In addition there is a following axis whose position is determined as a function of the status of these leading axes (position, tangent).
- The leading axes and following axis are only coupled at certain times, i.e. the coupling can be switched on and off by program instructions.
- Tangential control is defined for the basic coordinate system/workpiece coordinate system.
- The leading axes are defined as geometry axes and the following axis as a rotary axis.
- The coupled axes are assigned to the same channel.
- The position of the following axis can be the input value for a transformation.
- Tangential control is only active in AUTOMATIC and MDA modes.

Corners in the path contour

If the contour defined by the leading axes contains a corner the following points must noted with respect to the following rotary axis. You can select one of two different types of response:

- The path velocity is reduced to such an extent that the following axis reaches its target position synchronously with the other axes.
- If `TLIFT` has been programmed, an intermediate block is inserted at any corner whose angle is greater than the following machine data:

  MD37400 $MA_EPS_TLIFT_TANG_STEP (Tangent angle for corner recognition)

  In this inserted intermediate block, the rotary axis is moved as fast as possible to the position corresponding to the tangent after the corner. The limit values set for this axis are not violated.

Canceling of follow-up grouping

The definition of a follow-up grouping can be canceled in order to track new leading axes with the following axis.

Applications

The tangential control function can be used for example for the following applications:

- Tangential positioning of a rotatable tool for nibbling operations.
- Follow-up control of tool alignment for a bandsaw.
- Positioning a dressing tool on a grinding wheel.
- Positioning of a gear shaping cutter in glass or paper processing applications.
- Tangential feed of a wire for 5-axis welding.

7.2 Characteristics of tangential follow-up control

Task specification

Follow-up control for the rotary axis must be implemented so that the axis is always positioned at a specified angle on the programmed path of the two leading axes.

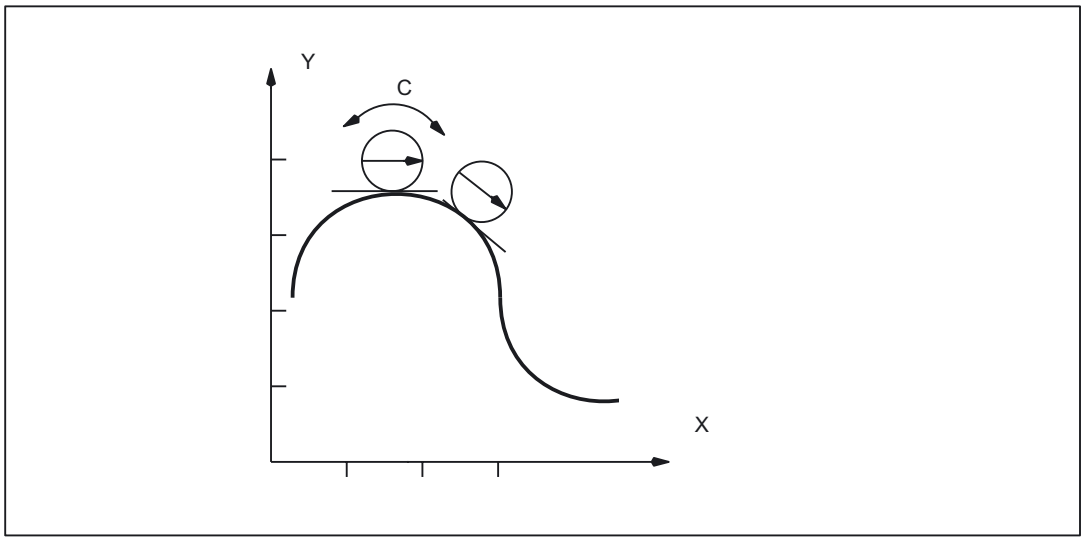


Figure 7-1 Tangential control, offset angle of zero degrees to path tangent

In the diagram, X and Y are the leading axes in which the path is programmed; C is the following axis whose position is determined by the control as a function of the leading axis values and of the desired offset angle between tangent and alignment in C.

The tangential control will function only if the leading axes are used as path axes. A leading axis which is programmed as a positioning axis (POS or POSA) does not specify values required for the follow-up control function.

Response on follow-up

A difference must made between the following cases:

- Without intermediate block (TLIFT)

  The path velocity of the leading axes is reduced to such an extent that the following axis reaches its target position synchronously with the other axes.

- With intermediate block (TLIFT), rounding off without G641

  The intermediate block generates the required turn of the tangentially following axis. It is interpolated in such a way that the following axis travels at its limit velocity. The intermediate block is not rounded. At the beginning of the intermediate block, the path velocity of the leading axes is zero.

Special cases

- G641 rounding is possible between two blocks, both of which move at least one of the two leading axes of the tangentially following axis.
- G641 rounding is possible between two blocks, both of which do not move either of the leading axes of the tangentially following axis.

In both cases, an intermediate block for the tangentially following axis is not created. An intermediate block is not required because in the preprocessing run the rounded contour is detected and the limit values for the following axis are calculated.

- Hidden corner in area

  A corner relevant for the tangential follow-up control can be hidden in space. (The projection of the contour on the plane defined by the two leading axes is relevant).

  If there is a hidden corner in space, an intermediate block is inserted before the block (here N6) causing the tangential jump. This intermediate block moves the following axis to the new position. The block transition is not rounded.

```
Programming                          Comment
N1 TANG (C, X, Y, 1)
N2 TLIFT (C)
N3 G1 G641 X0 Y0 F1000
N4 TANGON (C)
N5 X10
N6 Y10                               ; The rotary axis is repositioned before working
                                       off this block.
N7 M3001.97
```

7.3 Using tangential follow-up control

Activating

The following axis can only be aligned if:

- The assignment between the leading and following axes is declared to the system (TANG)
- Follow-up control is activated explicitly (TANGON)
- The response at corners is specified, if required (TLIFT).

Additional functions

Further functions are provided in order to:

- Terminate follow-up control of the following axis (TANGOF)
- Deactivate the special response at corners (TANG() without a subsequent TLIFT)
- Cancel the definition of a follow-up grouping (TANGDEL).

Effect on transformation

The position of the rotary axis to which follow-up control is applied can act as the input value for a transformation.

References:
/FB2/ Function Manual, Extended Functions; Kinematic Transformation (M1)

Note

The user is recommended to program TLIFT if tangential control is used together with a transformation. TLIFT prevents the follow-up axis from overtraveling and protects against excessive compensating movements.

Explicit programming of the follow-up axis

If a following axis, which is being made to follow its leading axes, is positioned explicitly, then the position specification is added to the offset angle programmed in the activation instruction TANGON. see Section "Activation of follow-up control". Motional commands (AC, IC, DC, POS,) are permitted.

Reference point approach

Follow-up control is deactivated while the following axis executes a reference point approach.

Cross-channel block search

The cross-channel block search in Program Test mode (SERUPRO "**Se**rch-**R**un by **Pro**gram test") can be used to stimulate tangential follow-up axes.

More information on cross-channel block search SERUPRO:

References:
/FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Operation, (K1), Section: Program test

7.3.1 Assignment between leading axes and following axis

Programming

The programming is carried out using the pre-defined sub-program TANG. The following parameters are transferred to the control:

Following axis (additional rotary axis)	here C
Leading axis 1 (geometrical axis)	here X
Leading axis 2 (geometrical axis)	here Y
Coupling factor	default 1
Coordinate system identifier "B" = Base coordinate system, "W" = Workpiece coordinate system is not available.	default "B"

The appropriate axis identifiers are used to specify the axes. The coupling factor is generally 1.

The coupling factor can be omitted.

```
TANG(C, X, Y)
```

7.3.2 Activation of follow-up control

Programming

The programming is carried out using the pre-defined sub-program TANGON. When the tangential control is activated, the name of the following axis which must be made to follow is transferred to the control. This specification refers to the assignment between master and following axes made beforehand with TANG. Refer to Section "Assignment between leading axes and following axis". An angle between the tangent and the position of the following axis can be specified optionally when follow-up is activated. This angle is maintained by the control for as long as the following axis is made to follow. The angle is added to the angle stored in the following machine data:

MD37402 $MA_TANG_OFFSET (preselection angle for tangential follow-up)

If the angle is zero both in TANGON and in the MD, the following axis takes the direction of the tangent.

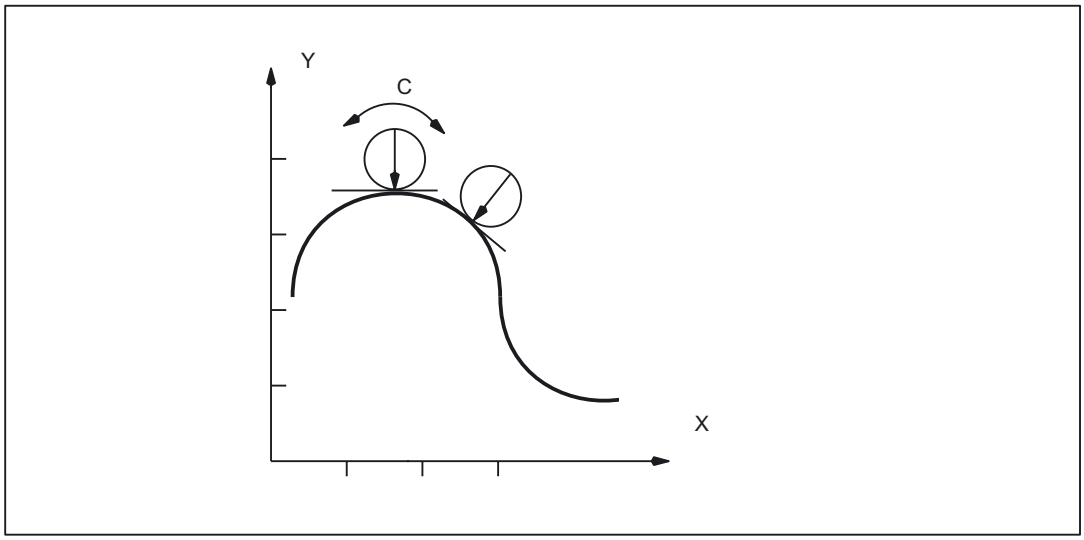


Figure 7-2 Tangential control, offset angle of 90 degrees to path tangent

Activation is programmed as follows for the above example and an offset angle of 90 degrees:

```
TANGON(C, 90)
```

In response to every motion in path axes X and Y, following axis C is rotated to an angle of 90 degrees in the relation to the path tangent.

7.3.3 Switching on corner response

After axis assignment with `TANG()`, the `TLIFT()` instruction must be written if the corner response is to be contained in an intermediate block.

```
TLIFT (C)
```

The control reads the following machine data for the tangential following axis C:

MD37400 $MA_EPS_TLIFT_TANG_STEP (Tangent angle for corner recognition)

If the tangential angle jump exceeds the angle (absolute value) of the angle set in the MD, the control recognizes a "corner" and approaches the new position of the follow-up axis via an intermediate block.

System variable $AC_TLIFT_BLOCK

The system variable $AC_TLIFT_BLOCK indicates whether the current block is an intermediate block generated by `TLIFT`. If the value of the system variable is 1, `TLIFT` inserted the current block as an intermediate block.

7.3.4 Termination of follow-up control

Programming

The programming is carried out using the pre-defined sub-program `TANGOF`. The name of the following axis to be decoupled from its leading axes for the remainder of the machining operation must be transferred to the control in conjunction with the subroutine name `TANGOF`.

With reference to the example in section "Assignment between leading axes and following axis", the termination is as follows:

```
TANGOF(C)
```

The follow-up control process initiated with `TANGON` is terminated.

Termination of follow-up control initiates a preprocessing stop.

RESET/end of part program

An activated tangential control can remain active for further machining operations.
For further details, see:

References:

/FB1/ Function Manual; Basic Functions; Coordinate Systems, Axis Types, Axis Configuration (K2)

Section: Actual-value system for workpiece
Section: External zero offset

Note

The assignment between 2 master axes and a slave axis programmed with `TANG( ... )` is not canceled by `TANGOF`. Refer to section "Canceling the definition of a follow-up axis assignment".

7.3.5 Switching off intermediate block generation

In order to stop generating the intermediate block at corners during program execution with active tangential follow-up control, the `TANG()` block must be repeated without following `TLIFT()`.

7.3.6 Canceling the definition of a follow-up axis assignment.

A follow-up axis assignment specified by TANG() remains active after TANGOF. This inhibits a plane change or geometry axis switchover.

The predefined subprogram TANGDEL is used to cancel the definition of a follow-up axis assignment so that the follow-up axis can be operated dependent on new leading axes when a new follow-up axis assignment is defined.

TANGDEL(C)

The existing definition in the example of TANG(A, X, Y) is canceled.

Example for plane change

```
N10 TANG(A, X, Y, 1)
N20 TANGON(A)
N30 X10 Y20
...
N80 TANGOF(A)
N90 TANGDEL(A)                ; delete defined coupling of A to X and Y as leading
                              axes
...
N120 TANG(A, X, Z)            ; A can be coupled to new leading axes
N130 TANGON(A)
...
N200 M30
```

Example for geometry axis switchover

If the definition of the follow-up axis assignment is not canceled, an attempt to execute a geometry axis switchover is suppressed and an alarm is output.

```
N10 GEOAX(2,Y1)
N20 TANG(A, X, Y)
N30 TANGON(A, 90)
N40 G2 F8000 X0 Y0 I0 J50
N50 GEOAX(2, Y2)                 ; Alarm 14415, geo axis to be deleted is still
                                 ; Leading axis of follow-up axis assignment
```

Geometry axis switchover with TANGDEL

The following example shows how TANGDEL is used correctly in association with an axis switchover.

```
N10 GEOAX(2,Y1)                 ; Geo axis group is determined
N20 TANG(A,X, Y)                ; Channel axis Y1 is being assigned
N30 TANGON(A, 90)               ; Follow-up group with Y1 is being activated
N40 G2 F8000 X0 Y0 I0 J50       ; Movement block for the leading axes
N50 TANGOF(A)                   ; Activation of follow-up control
N60 TANGDEL(A)                  ; Canceling the definition of a follow-up axis
                                assignment
N70 GEOAX(2, Y2)                ; Geometry axis switchover permitted
N80 TANG(A, X, Y)               ; Redef. Follow-up axis group
N90 TANGON(A, 90)               ; Follow-up group with Y2 is being activated
...
```

7.4 Limit angle

Description of problem

When the axis moves backwards and forwards along the path, the tangent turns abruptly through 180 degrees at the path reversal point. This response is not generally desirable for this type of machining operation (e.g. grinding of a contour). It is far better for the reverse motion to be executed at the same offset angle (negative) as the forward motion.

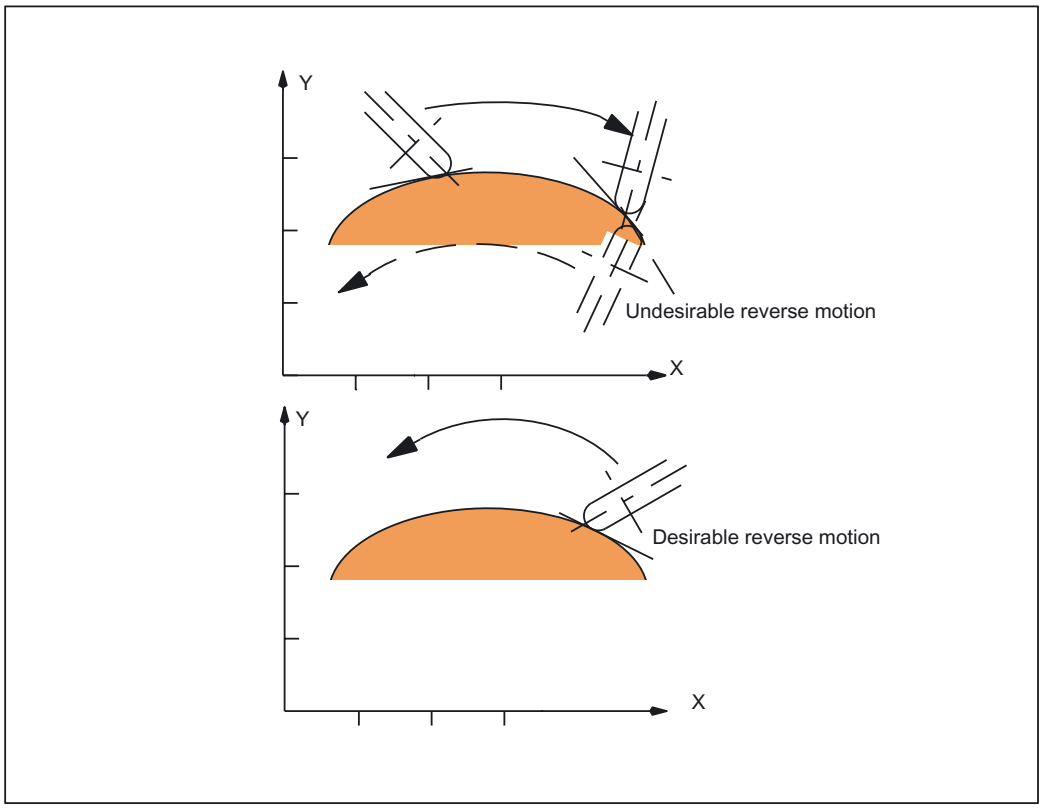


Figure 7-3 Backward and forward motion on the path

Programming

A minimum and a maximum value for the position of the axis made to follow ("C" in example) referred to the base coordinate system are transferred to the control with `G25` and `G26`. These working area limitations are activated with `WALIMON` and deactivated again with `WALIMOF`. The working area limitation must be active at the instant of path reversal.

References:
/PG/ Programming Manual; Fundamentals

Activation

If the current offset angle is outside the active working area limit for the following axis, an attempt is made to return to within the permissible working area by means of the negative offset angle. This response corresponds to that shown in the lower diagram of the above Fig.

7.5 Constraints

Availability

The "Tangential control" function is an option and available for

- SINUMERIK 840D mit NCU 572/573

The special response at path corners, controlled by `TLIFT` () is available.

7.6 Examples

Positioning of workpiece

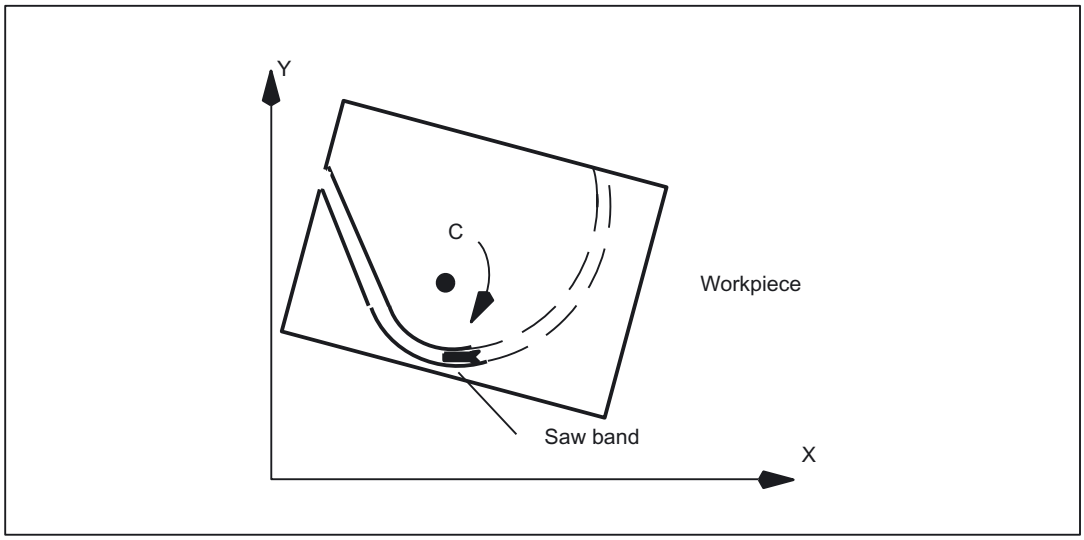


Figure 7-4 Tangential positioning of a workpiece on a bandsaw

Positioning of tool

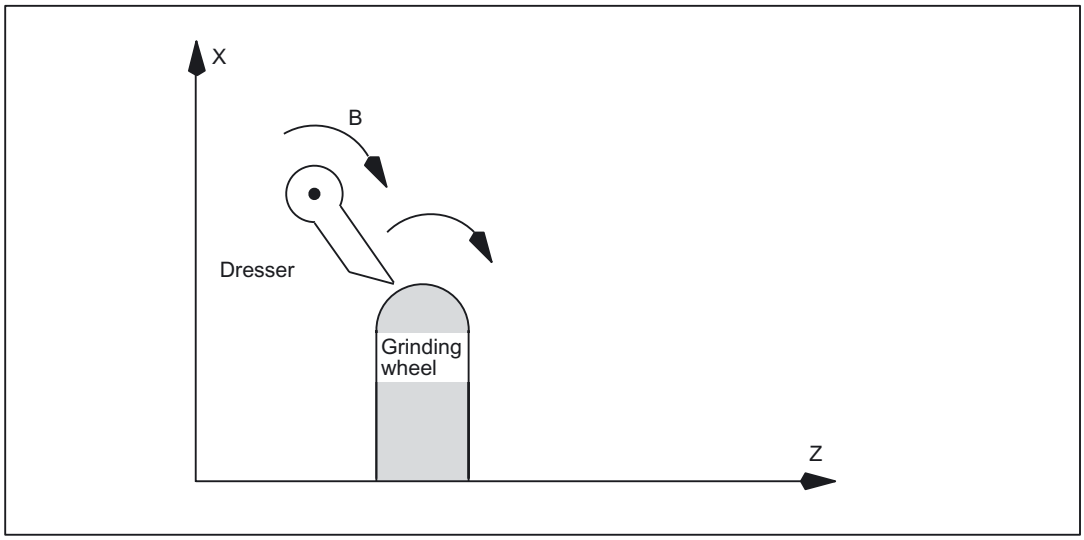


Figure 7-5 Positioning of a dressing tool on a grinding wheel

Example Corner in area

```
Programming
TANG(A,X,Y,1.0,"B")
TLIFT(A)
G1 G641 X0 Y0 Z0 A0
TANGON(A,0)
N4 X10
N5 Z10
N6 Y10
M30
```

Here, a corner is hidden in the area between N4 and N6. N6 causes a tangent jump. That is why there is no rounding between N5 and N6 and an intermediate block is inserted.

In the case of a hidden corner in area, an intermediate block is inserted before the block that has caused the tangent jump. The intermediate block moves the following axis to the new tangent position.

7.7 Data lists

7.7.1 Machine data

7.7.1.1 Axis/spindle-specific 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

7.7.2 System variables

Identifier	Description
$AC_TLIFT_BLOCK	Current block is an intermediate block generated by `TLIFT`

Installation and Activation of Loadable Compile Cycles (TE01)

8

8.1 Brief description

Contents

This present document describes how technology functions are installed and activated in the form of loadable compile cycles.

Technology functions

The description applies to all of the following technology functions available from Siemens:

- 1D/3D clearance control in position control cycle
  Order no.: 6FC5 251-0AC05-0AA0
  Compile cycle: CCCLC.ELF
  References: /FB3/ Clearance Control (TE1)
- Handling transformation package
  Order no.: 6FC5 251-0AD07-0AA0
  Compile cycle: CCRCTRA.ELF
  References: /FB3/ Handling Transformation Package (TE4)
- Setpoint exchange
  Order number: 6FC5 251-0AC05-0AA0
  Compile cycle: CCSETP.ELF
  References: /FB3/ Setpoint Exchange (S9)
- Axial coupling in the machine coordinate system (MCS coupling)
  Order no.: 6FC5 251-0AD11-0AA0
  Compile cycle: CCMCSC.ELF
  References: /FB3/ MCS Coupling (TE6)
- Continue machining at the contour (retrace support)
  Order no.: 6FC5 251-0AE72-0AA0
  Compile cycle: CCRESU
  References: /FB3/ Retrace Support (TE7)

- Fast laser switching signal

  Order no.: 6FC5 251-0AE74-0AA0

  Compile cycle: CCHSLC.ELF

  References: /FB3/ Cycle-Independent Path-Synchronous Signal Output (TE8)
- Axis pair collusion protection

  Order number: 6FC5 397-7CB10-0AA0

  Compile cycle: CCPROT.ELF

  References: /FB3/ Axis pair collusion protection (TE9)

The description applies uniformly to user-specific technological functions.

The following technology functions are no longer available in the form of compile cycles:

- Analog axis

  The compile cycle is now available as a hardware solution.
- Speed/torque coupling

  The compile cycle is a generally-available function from SW 6.4 and higher.

SW 6.3 and lower

Up to and including SW 6.3, the technology functions were supplied in the form of compile cycles on the **Technology PC Card**. Several technological functions were stored on one PC card, of which normally only individual functions were used.

SW 6.4 and higher

From SW 6.4, the technology functions are supplied individually as **loadable compile cycles**.

Tips for use

The following chapters describe how to load and activate the compile cycles and set the necessary NCK machine data.

Please follow the instructions in the "Detailed description", if you have not already used compile cycles before.

Follow the instructions in the "Constraints" section if you have made an archive from an operational control using compile cycles from a technology PC card and want to replace these compile cycles with more recent versions in the form of loadable compile cycles.

Preconditions

One of the following programs is required for the installation:

- SinuCom NC
- SinuCopy FFS
- HMI Advanced

Furthermore, a programming device/PC with MPI connection to the NCU must also be available.

Note

The following manuals must be followed for system startup:

- /IAD/ Commissioning Manual
- /HMI/ Commissioning Manual HMI Advanced
- Current standard upgrade instructions

System

The description is valid for the SINUMERIK 840D / 840D sl systems.

The description for loading and activating compile cycles in connection with SINUMERIK 840Di / 840Di sl can be found in:
References:
/HBI/ SINUMERIK 840Di Manual
or
/HBI/ SINUMERIK 840Di sl Manual
NC startup with HMI Advanced: Loadable compile cycles

Compile cycles

Compile cycles are functional expansions of the NCK 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 NCK system architecture, compile cycles have comprehensive access to data and functions of the NCK system level via defined software interfaces. In this way, compile cycles significantly extend the functionality of the NCK.

Including a compile cycle in the NCK system software is performed by loading the compile cycle into the file system of the NCK. 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

8.2 Loading compile cycles

8.2.1 Loading a compile cycle with HMI Advanced

Requirement

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 in the NCK:

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.

8.2.2 Loading a compile cycle with HMI Embedded

Requirement

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.

Implementation

Perform the following operation to load a compile cycle from a USB FlashDrive in the NCK:

1. Insert the USB FlashDrive into the PCU 20.
2. Open the USB FlashDrive as the local drive:

   Operating area switchover > Services > USB front

   If the "Local USB" is displayed as disabled, it implies the USB FlashDrive was not detected. Wait until HMI Embedded has detected the USB FlashDrive.
3. Select the compile cycle and copy it:

   vertical "Copy" softkey
4. Open the "Compile Cycles" directory and add the compile cycle:

   Vertical "Add" softkey
5. Initiate an NCK reset.

8.2.3 Loading a compile cycle from an external computer with WinSCP3

Requirement

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.

Implementation

Perform the following operation to load a compile cycle from an external computer into the NCK:

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:

   /card/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 NCK reset.

8.3 Interface version compatibility

The compile cycle and the NCK system software communicate via a SINUMERIK-specific interface. The interface version used by the loaded compile cycle must be compatible with the interface version of the NCK system software.

Interface versions

The relevant interface versions are displayed under:

- Interface version of the NCK system software

  HMI Advanced:

  Diagnosis > Service Display > Version > NCU Version

  Display (excerpt)

  CC Interface Version:

  @NCKOPI . . . .@Interfaces=<*1st digit*>.<*2nd digit*> . . . .

  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=<*1st digit*>.<*2nd digit*> . . . .

- 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 digit*><*2nd digit*>_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 NCK system software:

- 1st position of the interface version number

  The 1st digit of the interface version number of a compile cycle and the NCK system software must be **the same**.

- 2nd position of the interface version number

  The 2nd digit of the interface version number of a compile cycle must be **less than or equal** to the 2nd digit of the NCK system software.

CAUTION
If alarm 7200 is displayed after start-up, this means **no** compile cycle has been loaded!

8.4 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.

8.5 Activating the technological functions in the NCK

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 NCK boots 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 NCK 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 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 NCK is booted up next, the activated technology functions are integrated into the system software.

⚠ CAUTION

SINUMERIK 840D

The following alarm is displayed when a bit is set for the first time in the function-specific NCK 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** triggering NCK RESET.

8.6 Function-specific startup

Further function-specific installation routines are described in the corresponding function description (TE1 - TEn).

8.7 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.

8.7.1 Creating alarm texts with HMI sl

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. Please copy the "oem_alarms_deu.ts" file from the "/siemens/sinumerik/hmi/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 Sentence %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. Please copy the "oem_slaesvcadapconf.xml" file from the "/siemens/sinumerik/hmi/base" directory to the "/oem/sinumerik/hmi/cfg" directory.
7. Rename the file to "slaesvcadapconf.xml".

8. Open the "slaesvcadapconf.xml" file in the editor and record 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 HMI sl.

Further information about creating alarm text files with HMI sl can be taken from:
References:
/IAM/ SINUMERIK 840D sl Commissioning Manual; Chapter: Configuring user alarm texts

8.7.2 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:
/IAM/ SINUMERIK 840D sl/840Di sl/840D/840Di Startup CNC Part 2 (HMI); Startup HMI Advanced (IM4), Chapter: Creating user alarm texts

---

Note

HMI reinstallation

Retain the added alarm texts in the text files of the F:\oem even after a reinstallation of HMI.

8.7.3 Creating alarm texts with HMI Embedded

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. Create the required language-dependent sub-directories in the "/oem/sinumerik/hmi/lng" directory, e.g.:

   "deu" for German

   "eng" for English
2. Create the "alc.txt" text file in the language directories.
3. 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.
4. Restart HMI Embedded.

For more information about creating alarm texts with HMI Embedded, please refer to:
References:
/IAM/ SINUMERIK 840D sl/840Di sl/840D/840Di Commissioning CNC Part 2 (HMI); Commissioning HMI Embedded (IM2), Chapter: Creating In-House Texts

8.8 Boundary conditions

8.8.1 Transition to newer NCK versions (840D)

In order to be able to use technology functions from an existing archive in conjunction with newer NCK versions (NCK 06.03.23 and later), the archive must first be updated before being loaded in the NC.

Requirements

The following requirements must be met in order to update an archive:

- A PC card with a SINUMERIK 840D standard system, version NCK 06.03.23 or later
- The ELF files for the technology functions to be activated
- The conversion program arc4elf.exe (archive conversion)

Updating

Proceed as follows to update an archive:

- Create backup archive
  - Standard procedure

    or
  - Procedure including optimization of static NC memory usage
- Insert new PC card
- Incorporate the ELF files
- NCU RESET
- Activate technology function
- Reactivate NCU RESET
- Convert archive using arc4elf.exe
- Load converted archive

8.8.1.1 Create backup archive

Standard

Creating an archive to backup user data as a default action is described in:
References:
/IAD/ Startup Manual 840D/SIMODRIVE 611D: "Data Backup" section

Optimized

Data backup with optimization of the static NC memory usage is only necessary if an archive for an NCK Version 6.3.xx is being used and the static NC memory usage needs to be optimized.

The following memory configuration machine data is to be reset to enable the static NC memory previously explicitly requested for technology functions:

MD18238 $MN_MM_CC_MD_MEM_SIZE = 1

An explicit request no longer has to be sent to the static NC memory as the technology functions loaded via ELF files will request the required static NC memory on a function-specific basis from the NC memory management.

Resetting the above machine data will reorganize the static NC memory the next time the NCK starts up and all user data will be lost.

In this state, you must create a new archive BEFORE resetting the NCK.

Work through the following points in the sections from "Insert new PC card" to "Load converted archive" in order.

8.8.1.2 Insert new PC card

Proceed as follows

1. Replace the previous PC card with the one with the new system.
2. Activate erasing the static NC memory using:

   NCU RESET with NCU switch S3 to position 1

   After this start-up, alarm 4060 "standard machine data loaded" is present.

8.8.1.3 Loading compile cycles

See sections:
"Loading a compile cycle with HMI Advanced"
"Loading a compile cycle with HMI Embedded"
"Loading a compile cycle from an external computer with WinSCP3"

8.8.1.4 NCU RESET

When the NCK is rebooted after an NCU reset, the compile cycles are loaded onto the NCK system software.

You can check the versions of the loaded compile cycles (see "Loading a compile cycle ..." section).

8.8.1.5 Activate technology function

Option

The option bits for the loaded ELF files can be set (see "Activation of the technology functions in the NCK" section).

Channel activation

The "Activation of the technology functions in the NCK" section describes how the channel information approved for the individual technologies have to be set in the following machine data:

MD... $MN_CC_ACTIVE_IN_CHAN_XXX[0]

and

MD... $MN_CC_ACTIVE_IN_CHAN_XXX[1]

The associated MD number is derived from the loading sequence (see: "NCU RESET" section).

8.8.1.6 Reactivate NCU RESET

The NCK is rebooted; no alarms should appear.

8.8.1.7 Convert archive

The archive created in the standard procedure or together with optimization of the static NC memory usage (see "Create backup archive" section) has to be converted. The **arc4elf.exe** program is required for this purpose (available from E-Support).

You can call up help for using this program by means of **arc4elf -h**.

The general callup form is:

arc2elf ORIGINAL.ARC CONVERTED.ARC

Proceed as follows

1. Replace ORIGINAL:ARC and CONVERTED.ARC with the actual archive names.

   The converted archive is created in the same directory as the original archive.

8.8.1.8 Load converted archive

Proceed as follows

1. Load the converted archive.

   References:
   /BAD/ Operator's Manual HMI Advanced

2. Activate the imported data by NC-RESET.

8.9 Data lists

8.9.1 Machine data

8.9.1.1 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

Simulation of Compile Cycles (TE02) 9

9.1 Brief description

9.1.1 Function

If part programs, which use compile cycles, are simulated on the SINUMERIK user interface (e.g. HMI Advanced) simulation is aborted and corresponding error messages are issued. The reason is that compile cycle support has not yet been implemented on the HMI.

The measures described below show how to set up the simulation runtime environment to enable the simulation of part programs, which use compile cycles, without error messages.

9.1.2 Requirements

OEM transformations

At least the following software version is required in order to use the functions of OEM transformations to simulate part programs: SW 6.2.12

9.2 OEM transformations

When using OEM transformations, the simulation runtime environment has to be set.

Proceed as follows

1. Create a new directory: "<*installation path*>/OEM" in addition to the standard directory: "<*installation path*>/MMC2" in the directory structure of the HMI application on the computer on which the HMI application (e.g. HMI Advanced) is installed.
2. In the "OEM" directory, create the file "DPSIM.TEA" with the following contents:

   $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 contents:

   [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 contents:

   PROC TRAORI(INT II)

   RET

7. In the directory for the manufacturer cycles, create the file "TRACON.SPF" with the following contents:

   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.

Clearance Control (TE1)

10

10.1 Brief description

10.1.1 General information

Function Description

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

Availability

The "clearance control" technological function is available for the following systems:

- SINUMERIK 840D
- SINUMERIK 840Di

Compile cycle

The "clearance control" technological function is a compile cycle.

A description of the system-specific availability of compile cycles and information on how to deal with them can be found in:
References:
/FB3/ Function Manual, Special Functions;
Installation and Activation of Loadable Compile Cycles (TE01)
/HBi/ SINUMERIK 840Di Manual; NC Installation and Startup with HMI Advanced,
"Loadable compile cycles"

10.1.2 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 (840D)

An overview of the system components required for clearance control in conjunction with SINUMERIK 840D is provided in the following diagram.

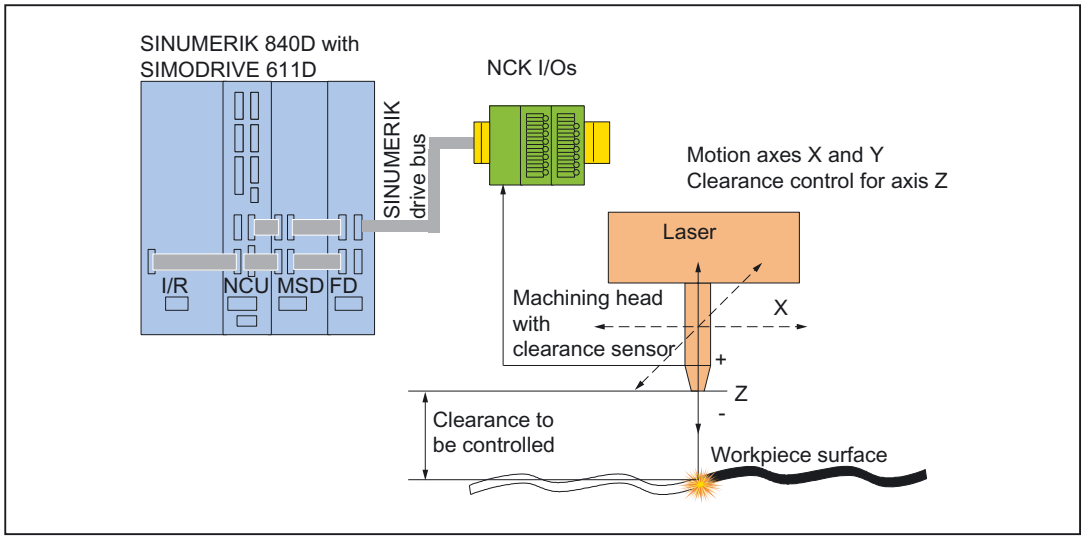


Figure 10-1 System components for clearance control with SINUMERIK 840D

System overview (840Di)

An overview of the system components required for clearance control in conjunction with SINUMERIK 840Di is provided in the following diagram.

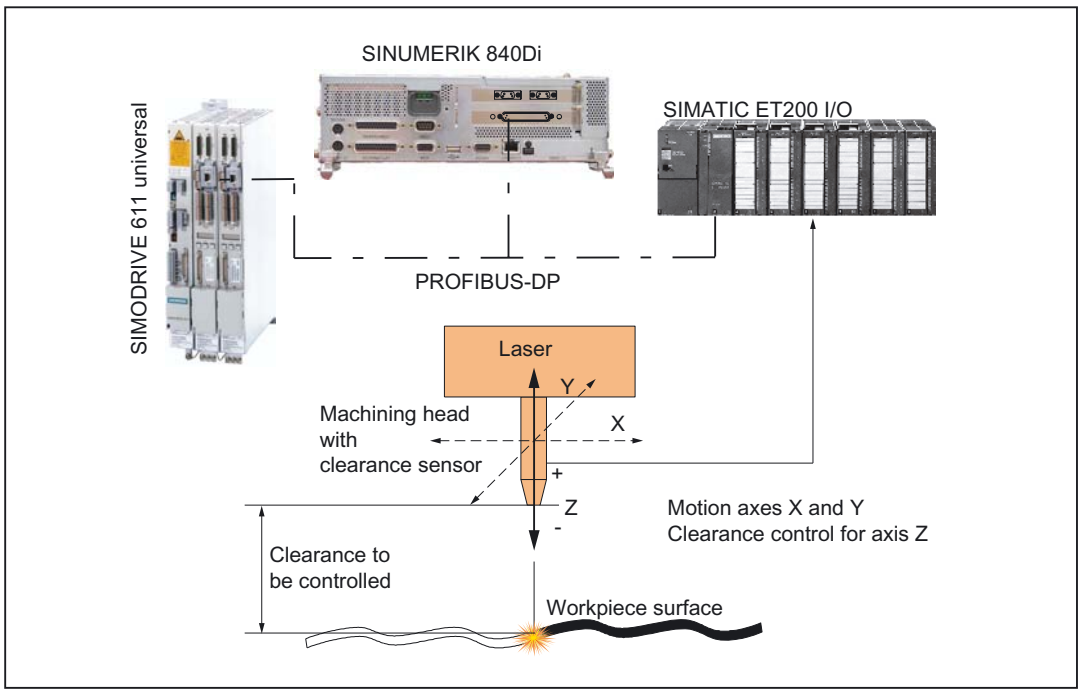


Figure 10-2 System components for clearance control with SINUMERIK 840Di

1D/ 3D machining

Clearance control can be used for 1D and 3D machining with up to five interpolatory axes.

- 1D machining

  In the case of 1D machining, clearance control is only applied to one axis, e.g. axis Z, as shown in the example machine configuration in the system overview for each SINUMERIK system (see "System components for clearance control with SINUMERIK 840D" and "System components for clearance control with SINUMERIK 840Di" above). Clearance control acts only in the direction of the Z axis.

- 3D machining

  3 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 3 linear axes are controlled by the clearance control. The direction of the compensation movement can be defined either in the direction:

  – of the tool orientation vector (normal case)

  – of the programmable compensation vector

10.2 Clearance control

10.2.1 Control dynamics

Closed-loop control gain Kv

The dynamic response of the closed control loop (sensor - open-loop control - axis) is determined by the maximum closed-loop control gain Kv.

The closed-loop control gain Kv is defined as:

$$Kv = \frac{\text{Velocity [m/min]}}{\text{Following error [mm]}}\ ; \quad \text{in } [\ \frac{\text{[m/min]}}{\text{[mm]}}\ ]$$

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)

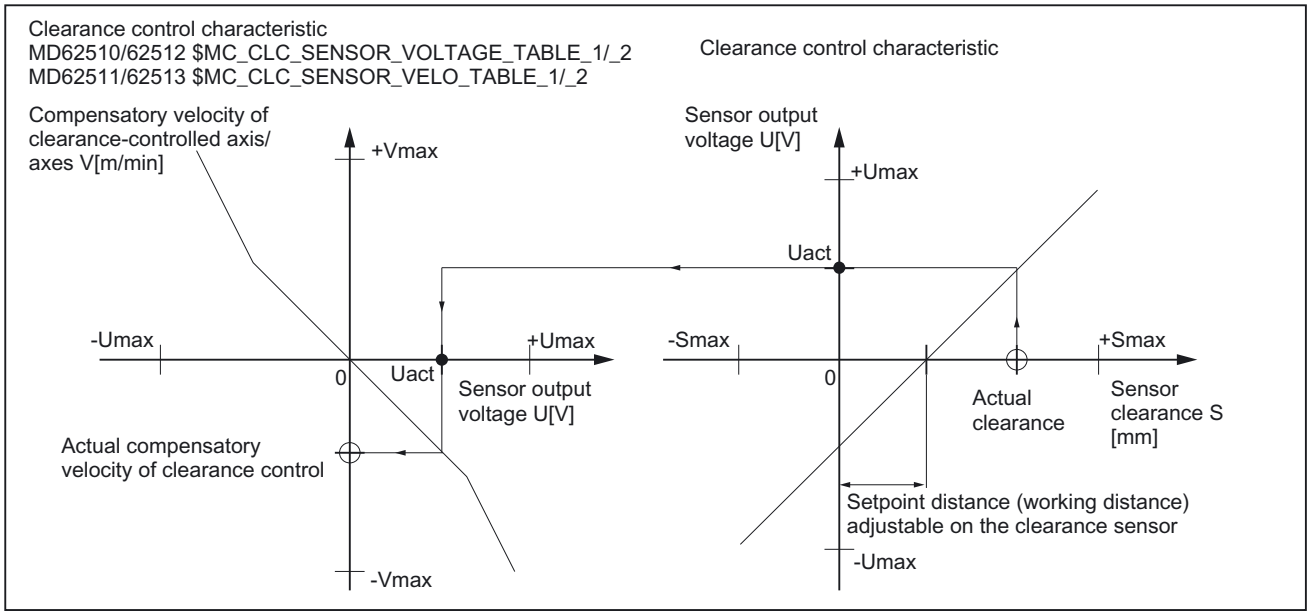


Figure 10-3 Correlation between characteristics: Clearance sensor and clearance control

- 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.

Deadtimes

In order to maximize the dynamics of the control response, clearance control takes place on the highest priority position controller level of the NCK. System-specific differences in the interface between the I/O modules and drives produce the following non-identical deadtimes.

(840D)

SINUMERIK 840D with I/O modules connected to the SINUMERIK drive bus and SIMODRIVE 611D drives produces a deadtime T_{dead} of:

T_{dead} = 2 * position controller cycle + 2 * speed controller cycle

(840Di)

SINUMERIK 840Di 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 + conversion time + channel cycle time + 2 * "PROFIBUS-DP cycle" + To

- Conversion time, channel cycle time:

  ET 200 S with "2 AI U high-speed" analog electronics module produces the following times:

  – Conversion time: 0.1 ms
  – Channel cycle time: 1 ms (both channels)

    → Average control deadtime of 0.5 ms

- To

  To is the setpoint transfer time of the drive parameterized in SIMATIC S7: HW-Config.

10.2.2 Velocity feedforward control

Eliminating the delay time

The closed-loop control gain, Kv, 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.

$$\text{With } \Delta t = \frac{1}{Kv}$$

and a closed-loop control gain Kv in seconds:

$$Kv \text{ in } \left[\frac{\text{m/min}}{\text{mm}}\right] = \left[\frac{1000 \text{ mm/60 s}}{\text{mm}}\right] = 16.667 \left[\frac{1}{\text{s}}\right]$$

for an assumed closed-loop control gain Kv = 4, the corresponding delay time Δt is:

$$\Delta t = \frac{1}{4 * 16.667 \left[\frac{1}{\text{s}}\right]} = 14.999 \text{ ms}$$

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)

(840D)

The velocity filters of the SIMODRIVE 611D drive provide an additional means of damping:

- MD1502 $MD_SPEED_FILTER_1_TIME (time constant for speed setpoint filter 1)
- MD1503 $MD_SPEED_FILTER_2_TIME (time constant for speed setpoint filter 2)

(840Di)

The velocity filters of the SIMODRIVE 611 universal/ E and POSMO SI, CD, CA drives provide an additional means of damping:

- Parameter 1502: (time constant for speed setpoint filter 1)
- Parameter 1503: (time constant for speed setpoint filter 2)

⚠ CAUTION
Every damping measure implemented contributes to increasing the overall time constant of the control loop!

You will find a complete description of the velocity feedforward control in:

References:

/FB2/ Function Manual, Extended Functions, Compensations (K3), Following Error Compensation (Feedforward Control)

10.2.3 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.

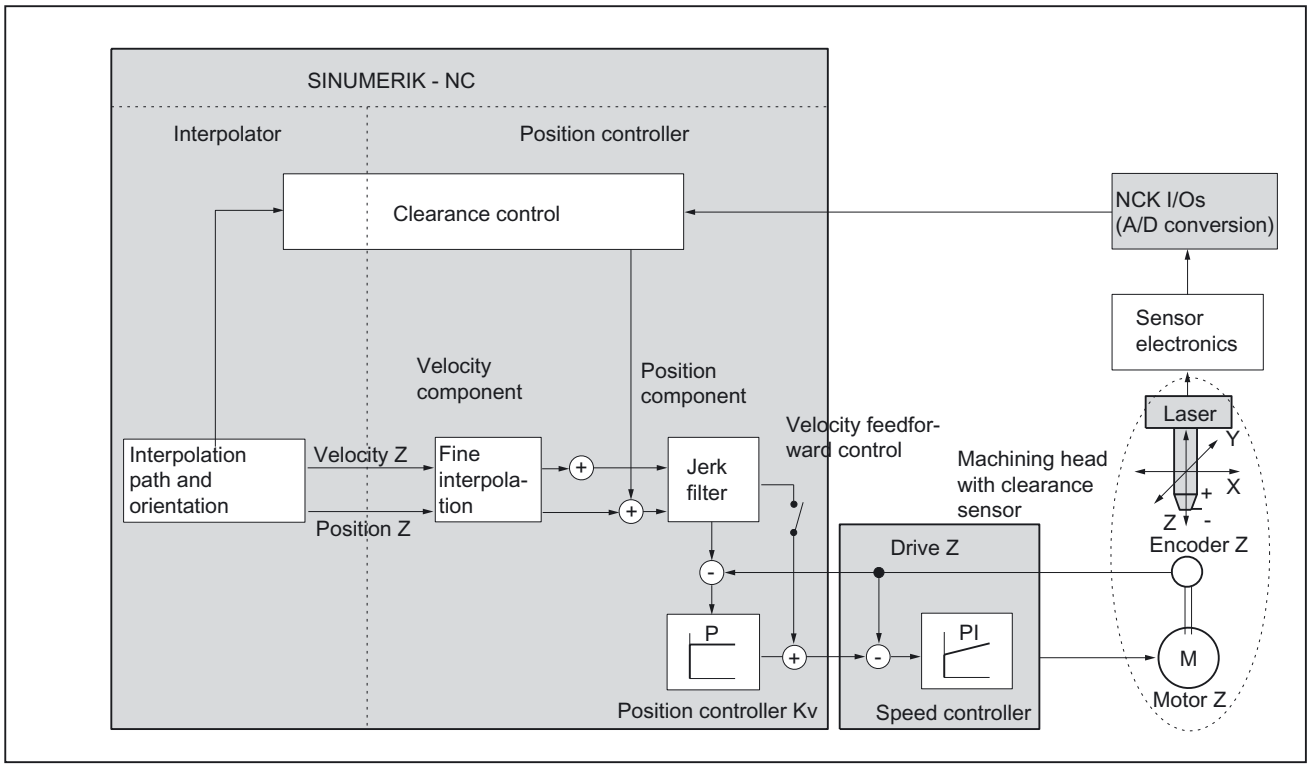


Figure 10-4 Control structure, position controller with clearance control (principle)

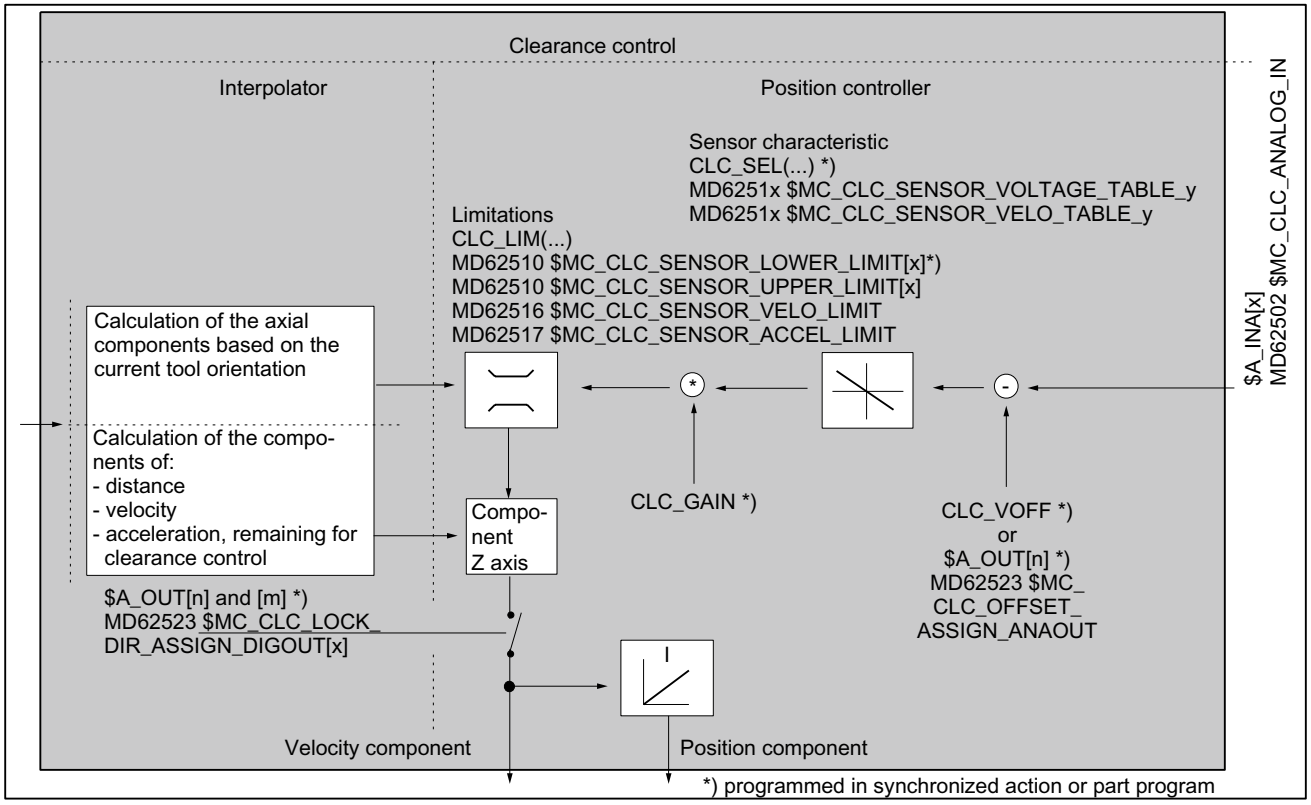


Figure 10-5 Control structure, clearance control (principle)

10.2.4 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.

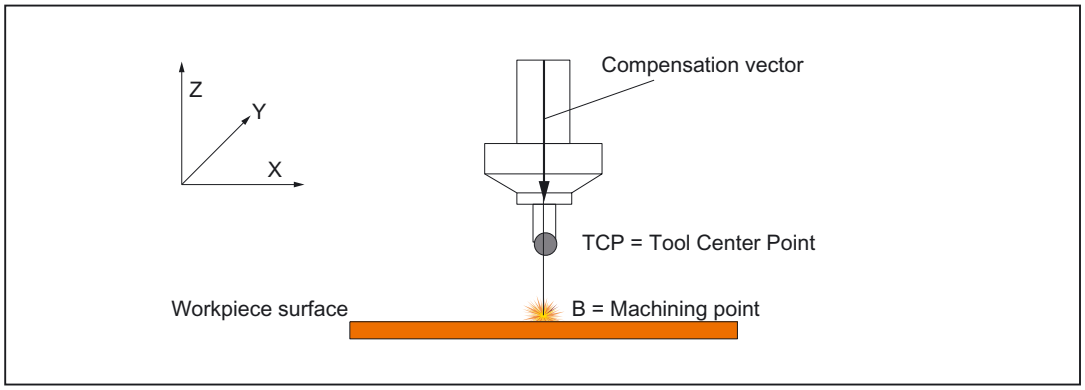


Figure 10-6 Clearance control with standard compensation vector

Note

In all the figures in this chapter, 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.

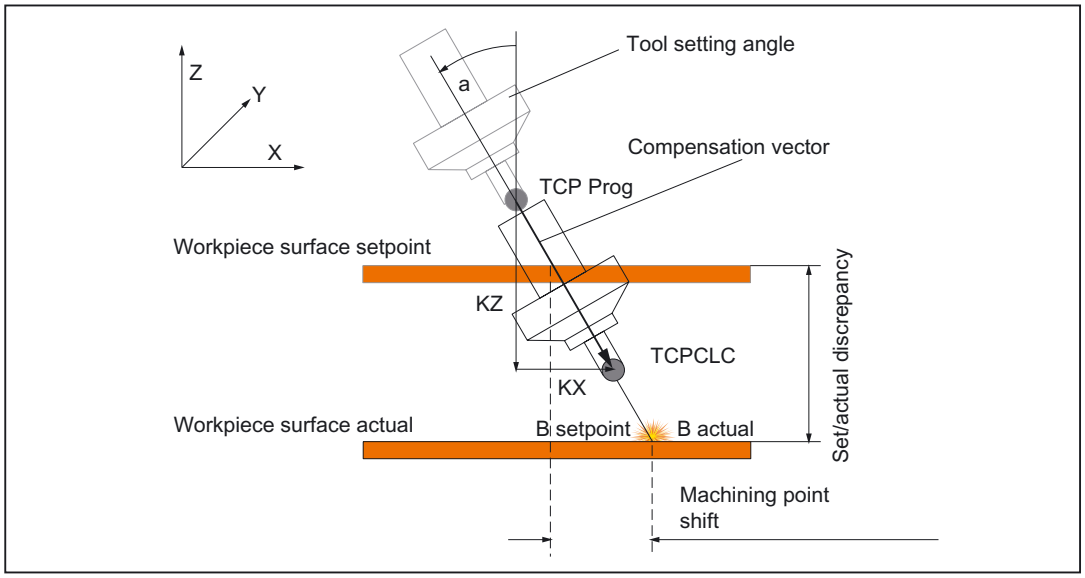


Figure 10-7 Standard compensation vector

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.

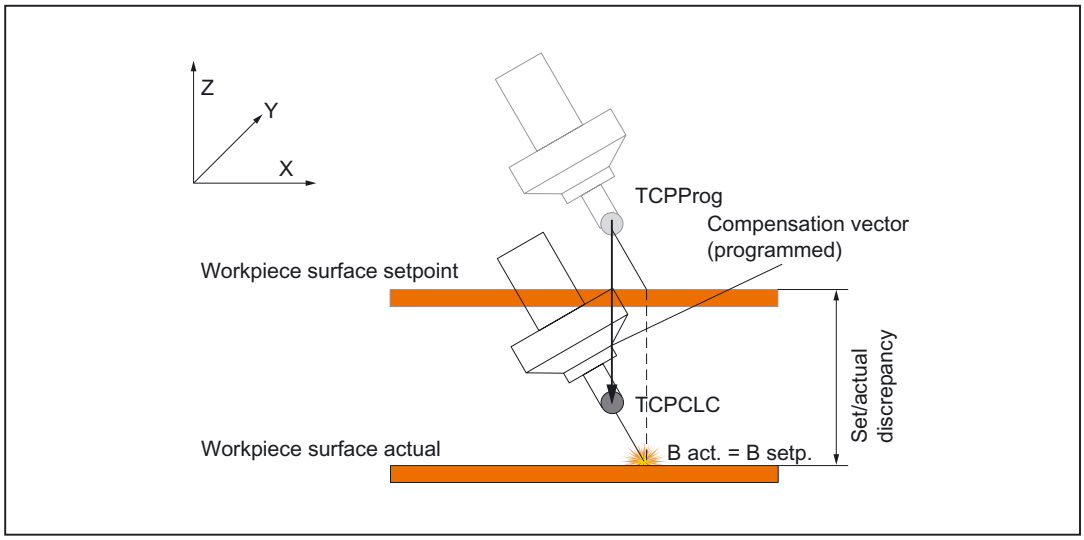


Figure 10-8 Programmable compensation vector

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.

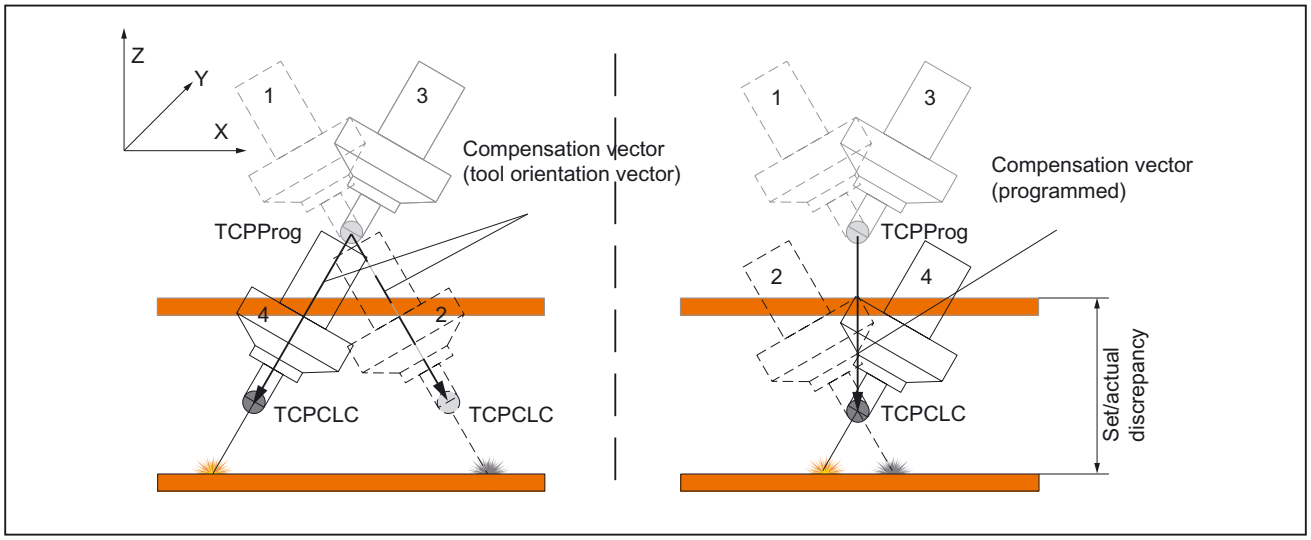


Figure 10-9 Change in orientation of the machining head

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. Actual programmed position of the machining head after the change in orientation
4. 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.

10.3 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 configured. 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.

10.4 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 NCK peripheral input.

In response to this signal, the clearance control function applies a retraction motion in all clearance-controlled axes. 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 stopped simultaneously.

Once all traversing movements have come to a standstill, part program processing can be resumed with NC START.

Parameterization

The digital peripheral input, which the "sensor collision" signal is wired to, is assigned to the clearance control function via 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).

10.5 Startup

Compile cycle

Before starting up the technological function, make sure that the corresponding compile cycle has been loaded and activated.

References:

/FB3/ Function Manual, Special Functions, Installation of Compile Cycles (TE01)

/HBI/ SINUMERIK 840Di Manual, NC Installation and Start-Up with HMI Advanced, Loadable Compile Cycles chapter

10.5.1 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.

10.5.2 Configuring the memory

Memory configuration

The technological function requires additional data in the NCK-internal block memory. The following memory-configuring channel-specific machine data must be parameterized:

- MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS = x + **4** 1) (number of block elements for compile cycles)
- MD28100 $MN_MM_NUM_CC_BLOCK_USER_MEM = x + **20** 1) (size of block memory for compile cycles (DRAM) in kBytes)
- 1) See note

Note

The values indicated must be entered in addition to the existing machine data value x.

10.5.3 Parameter settings for input signals (840D)

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 NCK inputs)
- MD10362 $MN_HW_ASSIGN_ANA_FASTIN (per analog module) (hardware assignment for the fast analog NCK inputs)

  Specifying the physical address activates the analog input module
- MD10384 $MN_HW_CLOCKED_MODULE_MASK (per terminal block) (clock-synchronous external NCK I/O processing)

  The slot of the analog input module on the terminal block must be set to clock-synchronous operation. To do this, set the machine data bit with the bit number of the module slot from the analog input module in the terminal block. (Example: Slot 5 → MD10384 = 10Hex)
- MD10380 $MN_HW_UPDATE_RATE_FASTIO (per terminal block) (update rate for the clock-synchronous external NCK I/O)

  Synchronization of the A/D converter to the position controller cycle

  MD10380 = 2

Digital input

The following machine data must be parameterized for the digital input:

- MD10350 $MN_FASTIO_DIG_NUM_INPUTS (number of active digital NCK input bytes)
- MD10366 $MN_HW_ASSIGN_DIG_FASTIN (per digital module) (hardware assignment for the external digital NCK inputs)

  Specifying the physical address activates the digital input module.

A complete description of the analog and digital inputs appears in:

References:

/FB2/ Function Manual, Extended Functions, Digital and Analog NCK I/Os (A4)

10.5.4 Parameter settings for input signals (840Di)

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 NCK inputs)
- MD10362 $MN_HW_ASSIGN_ANA_FASTIN (hardware inputs for the fast analog NCK inputs)

  I/O address of the I/O module

Digital input

Subsequent parameter settings do not have to be made if a digital input on the MCI board extension module (option) is used.

If an I/O module on PROFIBUS-DP is used for the digital input, the following machine data must be parameterized:

- MD10350 $MN_FASTIO_DIG_NUM_INPUTS (number of active digital NCK input bytes)
- MD10366 $MN_HW_ASSIGN_DIG_FASTIN (hardware assignment for the external digital NCK inputs)

  I/O address of the I/O module

A complete description of the parameter settings for analog and digital I/Os on a SINUMERIK 840Di appears in:

References:

/HBi/ SINUMERIK 840Di Manual, NC Installation and Start-Up with HMI Advanced, Digital and Analog I/Os chapter

A complete description of the analog and digital inputs appears in:

References:

/FB2/ Function Manual, Extended Functions, Digital and Analog NCK I/Os (A4)

10.5.5 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:

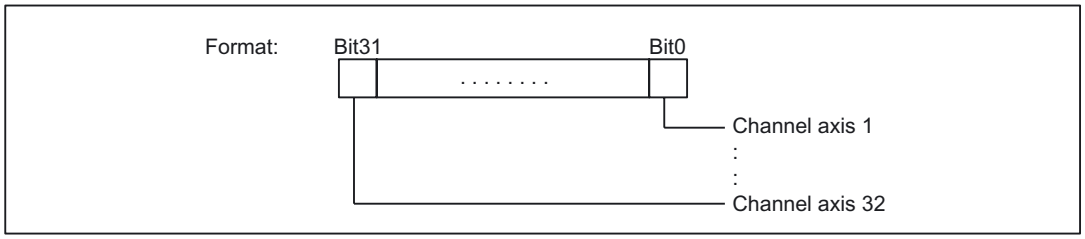
   Since the direction axes are only used to interpolate the direction components, they do not need mechanical axes and can therefore be configured as simulation axes.
3. [mm] or [inch] must be selected as the unit of measurement for the direction axes.
4. The direction axes may not participate in an axis 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 ()

10.5.6 Parameter settings for clearance control

Part program identifiers

The following machine data must be parameterized for the declaration of the function-specific part program identifiers 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 = x (axis assignment for clearance control)
  - x > 0: 1D clearance control where x = axis number of clearance-controlled channel axis
  - x = -1: 1st 5-axis transformation configured in channel
  - x = -2: 2nd 5-axis transformation configured in channel

Input signals

The clearance sensor input signals parameterized above:

- 840D: "Parameter settings for input signals (840D)" chapter
- 840Di: "Parameter settings for input signals (840Di)" chapter

are declared to the clearance control function via the following machine data:

- MD62502 $MN_CLC_ANALOG_IN = x (analog input for clearance control)

  x =input number, as when addressing system variables $A_INA[x]
- MD62504 $MN_CLC_SENSOR_TOUCHED_INPUT = x (assignment of the input bit for the "sensor collision" signal)

  x =input number, as when addressing system variables $A_IN[x]

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] (high velocity/ speed "axis/ spindle stopped")

  x = Axis number of clearance-controlled machine axis

Complete parameterization

Set the remaining clearance control parameters in accordance with the data and signal descriptions in the sections of the same name.

10.5.7 Starting up clearance control

Clearance sensor

Connect the clearance sensor outputs to the I/O modules activated via machine data:

- MD10362 $MN_HW_ASSIGN_ANA_FASTIN (I/O address of the I/O module) (hardware assignment for the fast analog NCK inputs)
- MD10366 $MN_HW_ASSIGN_DIG_FASTIN (I/O address of the I/O module) (hardware assignment for the external digital NCK inputs)

For more information about I/O modules, see Boundary Conditions, "I/O modules" section.

Test control direction

Proceed as follows to test the control direction of the clearance control function:

- Activate clearance control via a part program with CLC(1) (see "Activating and Deactivating Clearance Control (CLC)" section)
- Generate an input voltage, e.g. via the following synchronized action:

```
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
```

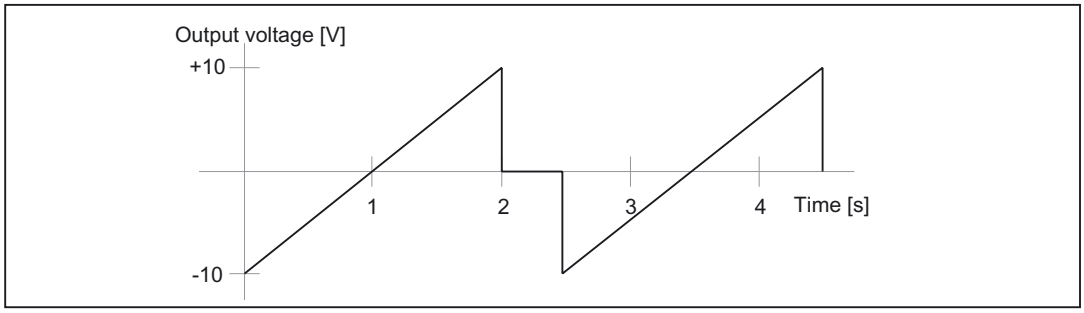


Figure 10-10 Output voltage for synchronized action

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:

MD10366 $MN_CLC_OFFSET_ASSIGN_ANAOUT = 6 (hardware assignment for the external digital NCK 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

The quality of the analog input signal can be checked using the function-specific display data described in the "Function-Specific Display Data" section.

Function-specific alarm texts

Function-specific alarm texts must first be integrated into the appropriate HMI data management before they can be displayed. A description of how to do this appears in the "Creating Alarm Texts" section.

Completion

A data backup is recommended once the start-up procedure has been completed.

References:

/IAD/ Commissioning Manual SINUMERIK 840D/611D, data backup

/HBi/ SINUMERIK 840Di Manual, user data backup/ series startup

Note

A data backup is recommended once the start-up procedure has been completed.

10.6 Programming

10.6.1 Activating and deactivating clearance control (CLC)

Syntax

CLC(*Mode*)

Mode

- Format: Integer
- Range of values: -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 response

CLC(0) is executed implicitly on a reset (NC RESET or end of program).

Parameterizable RESET response

The reset response of a 1D clearance control function can be determined via the channel-specific NCK OEM machine data:

- MD62524 $MC_CLC_ACTIVE_AFTER_RESET (reset response with active CLC)

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 above: activation/deactivation modes).

- MD62524 $MC_CLC_ACTIVE_AFTER_RESET = 1

  The current state of the clearance control function is retained.

⚠ CAUTION

The following channel-specific NCK OEM machine data is only effective in conjunction with a 1D clearance control function:

- MD62524 $MC_CLC_ACTIVE_RESET (reset response with active CLC)

For 3D clearance control, CLC(0) is always effective in the event of a RESET.

Boundary conditions

Please note the following boundary 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.

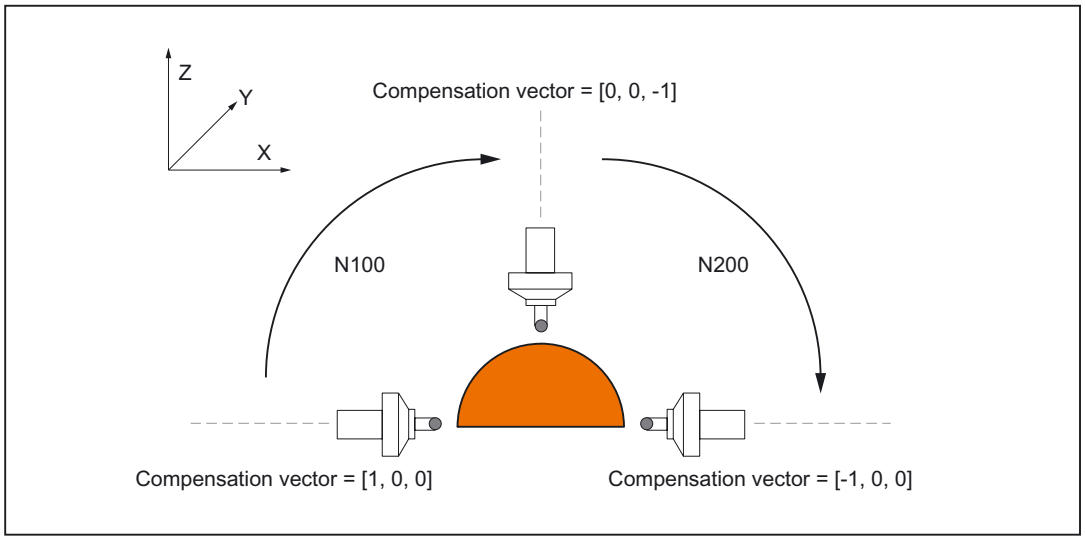


Figure 10-11 Interpolation of the compensation vector

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

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.

10.6.2 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.

⚠ CAUTION
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).

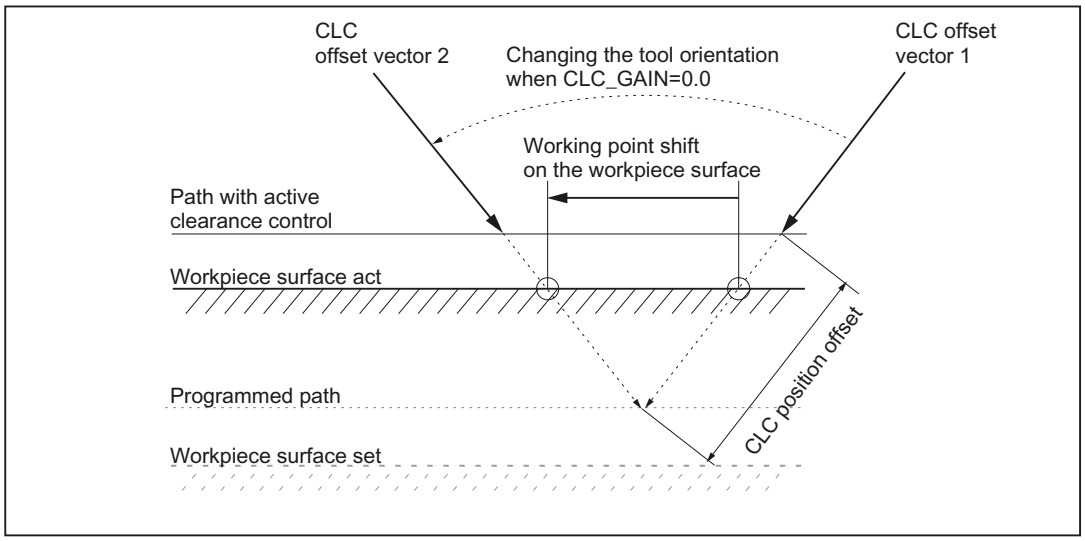


Figure 10-12 Response of the CLC offset vector when CLC_GAIN=0.0

Reset

Within a part program, a modified gain factor must be reset by means of explicitly programming CLC_GAIN=1.0.

RESET response

CLC_GAIN=1.0 becomes effective after a power on reset, NC RESET or end of program.

10.6.3 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)

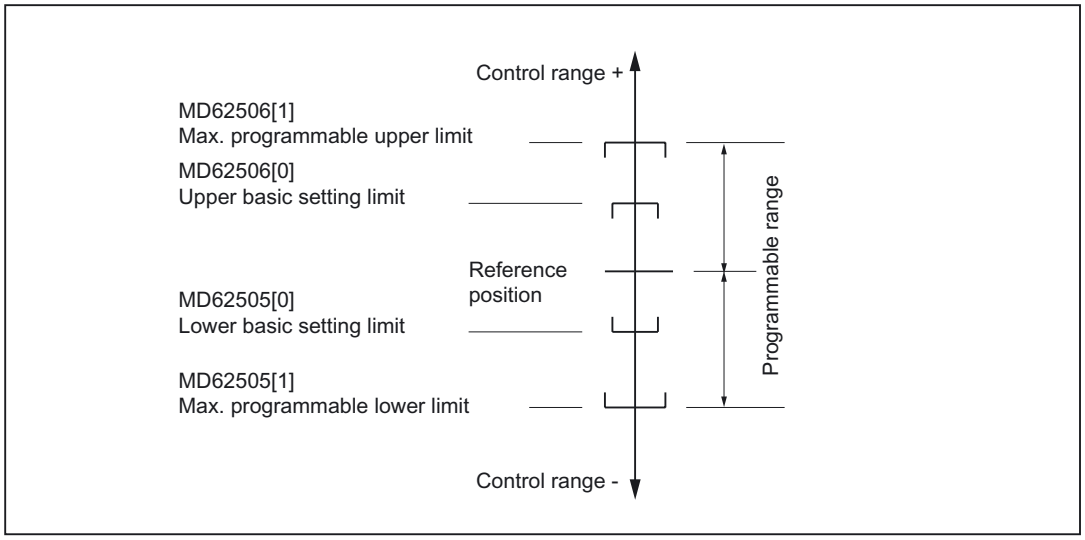


Figure 10-13 Value range limits for lower and upper limit

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"

10.6.4 Direction-dependent traversing motion disable

Syntax

$A_OUT[*number*] = *enabling signal*

Number

Number of the parameterized digital output (see below: Parameterization)

- Format: Integer
- Range of values: 1, 2, . . . max. number of digital outputs

Enabling signal

Enabling signal, can be inverted (see below: Parameterization)

- 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

10.6.5 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.

10.6.6 Voltage offset definable by synchronized action

Syntax

$A_OUTA[*number*] = V*oltage offset*

Number

Number of the parameterized analog output (see below: Parameterization)

- Format: Integer
- Range of values: 1, 2, . . .max. number of analog outputs

V*oltage offset*

As with voltage offset with CLC_VOFF (see "Voltage offset, can be set on a block-specific basis (CLC_VOFF)" section).

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 interpolation 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]

10.6.7 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.

10.7 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.

10.7.1 Channel-specific GUD variables

The "clearance control" technological function provides the following channel-specific GUD variables for HMI applications:

- SINUMERIK HMI Advanced
- SINUMERIK HMI Embedded

Table 10-1 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.

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

HMI Embedded

Proceed as follows to create and display the GUD variables in HMI Embedded.

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 > Services > Data admin > Programs/Data: 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 data > Channel spec. user data

SINUMERIK NCK

The new 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 NCK POWER ON RESET.

Note

Once the GUD variables have been created, an NCK POWER ON RESET must be carried out in order for the clearance control function to update the GUD variables.

10.7.2 OPI variable

The "clearance control" technological function provides the following channel-specific OPI variables as display data for the HMI application:

- SINUMERIK HMI Advanced

Table 10-2 Channel-specific OPI variable

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 standardized tool orientation vector	-	read only
CLC[7]	2nd component of the standardized tool orientation vector	-	read only
CLC[8]	3rd component of the standardized tool orientation vector	-	read only

Once the technological function has been started up successfully, the OPI variable is not available automatically.

OPI variable

Proceed as follows to define the 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 1.: Note

4. In file USER.NSK, supplement the call of the CLC-specific definition file: CLC.NSK. 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
```

10.8 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

10.9 Boundary conditions

10.9.1 I/O modules

For A/D conversion, the analog output current of the clearance sensor must be connected to NC via an I/O module with analog input to the NC.

10.9.1.1 I/O modules (840D)

The analog I/O module (DMP compact module) is connected to the drive bus via an NCU terminal block.

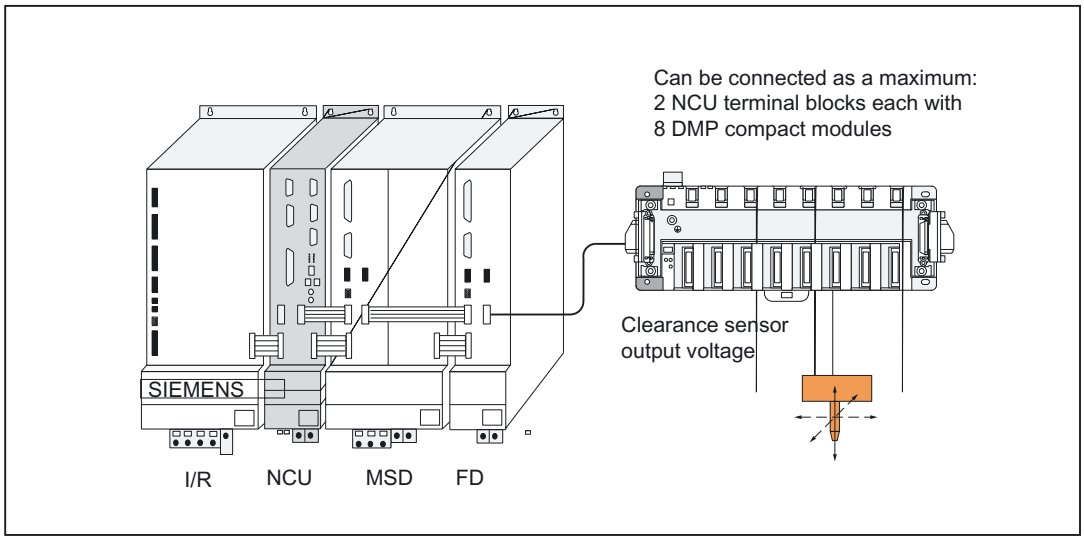


Figure 10-14 Clearance sensor connection via analog DMP module

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.

An I/O module suitable for clearance control is:

- DMP compact module 1I, NC analog

  Conversion time: 75 µs

  Order number (MLFB): 6FC5 211-0AA10-0AA0

I/O module connection

A description of how the I/O modules are connected appears in:

References:

/PHD/ SINUMERIK 840D NCU Configuration Manual, Terminal Block, NCU Terminal Block section (6FC5 211-0AA00-0AA0)

/PHD/ SINUMERIK 840D NCU Configuration Manual, DMP Compact Module, 1E NC Analog section (6FC5 211-0AA10-0AA0)

10.9.1.2 I/O modules (840Di)

On the SINUMERIK 840Di, the analog I/O module is connected via PROFIBUS-DP.

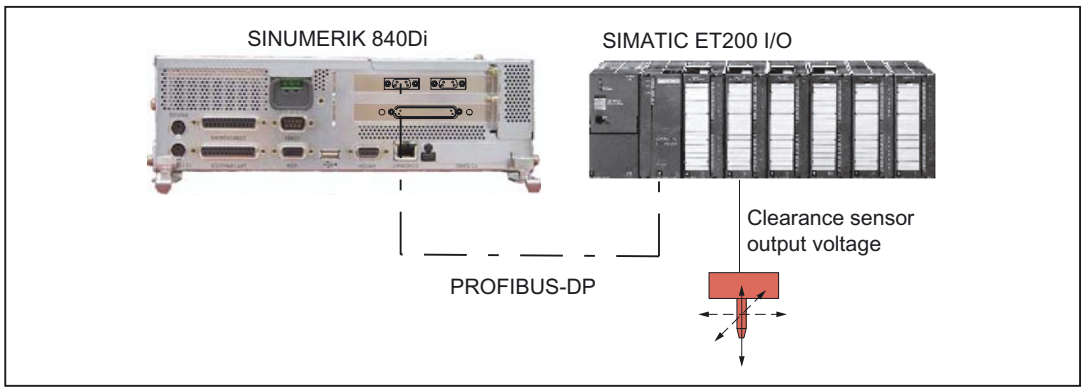


Figure 10-15 Clearance sensor connection via analog S7 I/O module

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.

A SIMATIC S7 module suitable for clearance control is:

- Analog I/O module 2 AI, U, high-speed for ET 200S

  Conversion time per channel (max. 2): 100 µs

  Cycle time for both channels: 1 ms

  Order number (MLFB): 6ES7 134-4FB50-0AB0

I/O module connection

A description of how the I/O modules are connected appears in:

References:

/HBi/ SINUMERIK 840Di Manual, PROFIBIUS-DP Communication, SIMATIC S7 I/O units section

10.9.1.3 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.

10.9.2 Function-specific boundary conditions

NC stop from PLC

If, in addition to the programmed path motion, the traversing movement of the clearance-controlled axes is also to be stopped in connection with an NC stop, the "NC stop axes and spindles" interface signal must also be set in addition to the "NC stop" interface signal:

- DB21, ... DBX7.3 (NC stop)
- DB21, ... DBX7.4 (NC stop axes and spindles)

Follow-up

If a clearance-controlled axis is to be switched as an alarm response or via the corresponding interface signal from the 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 traversing range used must be entered in the corresponding machine data 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 is not being 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 (Disable digital NCK inputs)

DB10, DBB146 (Disable analog NCK 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 one of the clearance-controlled axes may be configured as the master axis of a gantry grouping, defined via machine data:

MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition)

Following axes in a gantry grouping may not be used in the context of clearance control.

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 on a SINUMERIK standard HMI:

- SINUMERIK HMI Advanced
- SINUMERIK HMI Embedded

With the HMIs mentioned, the actual current axle position is displayed on the service screen **Operating area switchover > Diagnosis > Service Displays > Axle/ Spindle** as the "actual position value".

NC channels

The "clearance control" technological function is only available in the first NC channel, even on controls with more than one NC channel.

- Only channel axes in the first NC channel may be used as clearance-controlled axes.
- CLC part program commands may only be used in part programs processed in the first NC channel.

Conflict with other machine data

Using the following machine data is not permitted:

MD30132 $MA_IS_VIRTUAL_AX = 1 (axis is virtual axis)

Note

The "clearance control" technological function is only available in the **first** NC channel!

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 computing time required comes into effect when clearance control is activated in the part program (CLC(x)). 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.

10.10 Data lists

10.10.1 Machine data

10.10.1.1 Drive-specific machine data (840D)

Drive machine data (SIMODRIVE 611D)

Number	Identifier: $MD_	Description
1502	SPEED_FILTER_1_TIME [n]	Time constant for setpoint speed filter 1
1503	SPEED_FILTER_2_TIME [n]	Time constant for setpoint speed filter 2

10.10.1.2 Drive-specific machine data (840Di)

Drive parameter (SIMODRIVE 611D; POSMO SI/CD/CA)

Number	Identifier: $MD_	Description
1502	SPEED_FILTER_1_TIME [n]	Time constant for setpoint speed filter 1
1503	SPEED_FILTER_2_TIME [n]	Time constant for setpoint speed filter 2

10.10.1.3 NC-specific machine data

Number	Identifier: $MN_	Description
10300	FASTIO_ANA_NUM_INPUTS	Number of active analog NCK inputs
10350	FASTIO_DIG_NUM_INPUTS	Number of active digital NCK input bytes
10362	HW_ASSIGN_ANA_FASTIN	Hardware assignment of external analog NCK inputs: 0...7
10380	HW_UPDATE_RATE_FASTIO	Update cycle of synchronously clocked external NCK input/output modules
10382	HW_LEAD_TIME_FASTIO	Lead time of synchronously clocked external NCK input/outputs. Terminal block: 0...3
10384	HW_CLOCKED_MODULE_MASK	Synchronous processing of the individual external input/output modules. Terminal block: 0...3
10712	NC_USER_CODE_CONF_NAME_TAB	List of renamed NC identifiers

10.10.1.4 Channel-specific 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 clearance control
62510	CLC_SENSOR_VOLTABE_TABLE_1	Coordinate voltage of interpolation points sensor characteristic 1
62511	CLC_SENSOR_VELO_TABLE_1	Coordinate velocity of interpolation points sensor characteristic 1
62512	CLC_SENSOR_VOLTAGE_TABLE_2	Coordinate voltage of interpolation points sensor characteristic 2
62513	CLC_SENSOR_VELO_TABLE_2	Coordinate velocity of interpolation points sensor characteristic 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"
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

10.10.1.5 Axis/spindle-specific 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

10.10.2 Signals

10.10.2.1 Signals to channel

DB number	Byte.bit	Description
21, ...	1.4	Stop CLC motion
21, ...	1.5	Feedrate override acts on CLC

10.10.2.2 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

Speed/Torque Coupling, Master-Slave (TE3) 11

11.1 Brief description

SW 6 and higher

The speed/torque coupling function (master-slave) is used for mechanically-coupled axes that are driven by two separate motors. A further application is the compensation of gears and backlash in the gear tooth flank due to mutual tension in the drives.

Speed/torque coupling (master-slave) is a speed setpoint coupling between a master and a slave axis, involving a torque compensatory controller for even torque distribution. Each slave axis has exactly one master axis. Conversely, a master axis can also belong to several slaves; this is done by configuring several master-slave relationships using the same master axis. A configured slave axis must not be the master axis in one of the other master-slave relationships.

Differences compared to previous solution (up to SW 5.x)

- If traversing is programmed for a slave axis that has already been linked, an alarm is issued.
- The setpoint position of the coupled slave axis corresponds to the current actual position.
- On request, the coupling is made or released independent of the channel status the next time the axis stops. This allows the coupling status to be changed even during part program processing.
- For brake control, the interface signal "Master-slave coupling status active" should be used.
- If a master axis is simultaneously configured as the slave, an alarm is issued. So cascading is not possible.
- If a coupling is requested and closed, the control activation signals are derived directly from the master axis.
- If the coupling is closed, the slave axis is speed-controlled; status signal DB3x.DBX61.5 "Position control active" is not set.

Please see Chapter "Constraints" for more information about the differences.

SW 6.4 and higher

The function of the speed/torque coupling has been expanded to include the following options:

- Coupling/decoupling of **rotating, speed/controlled spindles**
- Dynamic configuration of couplings

A separate machine data has been provided for reversing the direction of the slave axis in coupled state.

Up to SW 5.x

The speed/torque coupling function (master-slave) is used for mechanically-coupled axes that are driven by two separate motors. This function was available up to SW 5 only via a technology card. It was not included in the standard scope of functions.

11.2 Speed/torque coupling, master-slave (SW 6 and higher)

11.2.1 General information

Speed/torque coupling (master-slave) is a speed setpoint coupling between a master and a slave axis, involving a torque compensatory controller for even torque distribution.

This function is mainly used for boosting the power of mechanically-coupled drives. Other application: Compensation of gears and backlash in the gear tooth flank due to mutual tension in the drives.

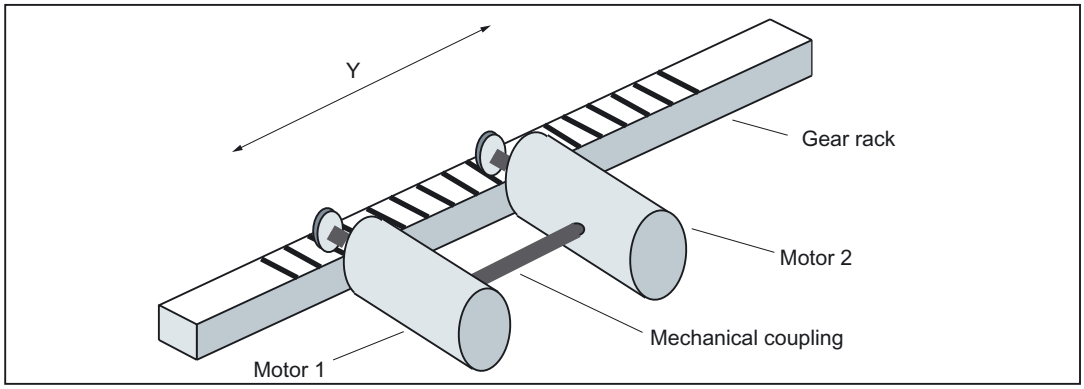


Figure 11-1 Permanent mechanical coupling

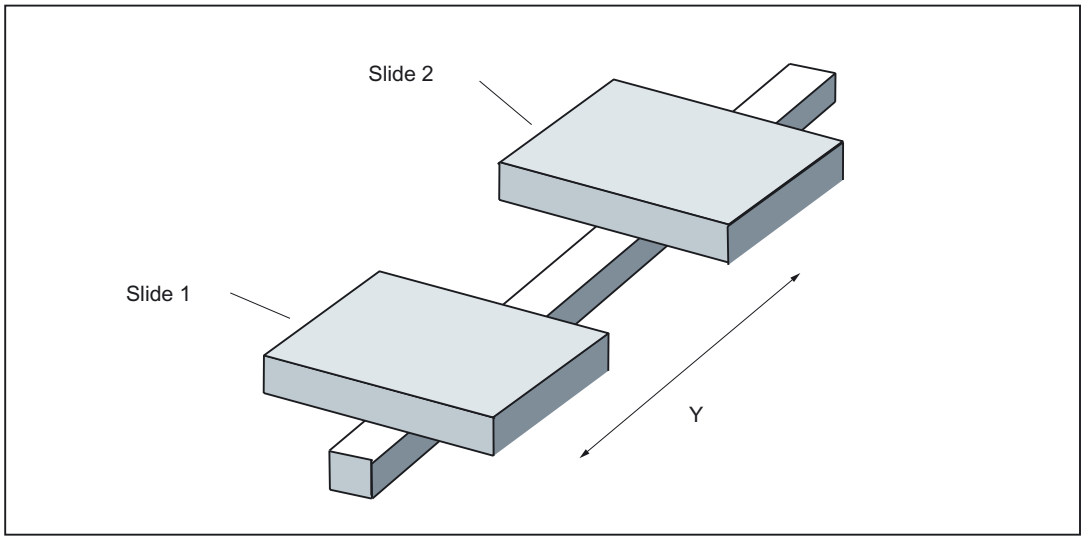


Figure 11-2 Slides (linear motor) for temporary coupling

Each slave axis has exactly one master axis.

Conversely, a master axis can also belong to several slaves; this is done by configuring several master-slave relationships using the same master axis. A configured slave axis must not be the master axis in one of the other master-slave relationships.

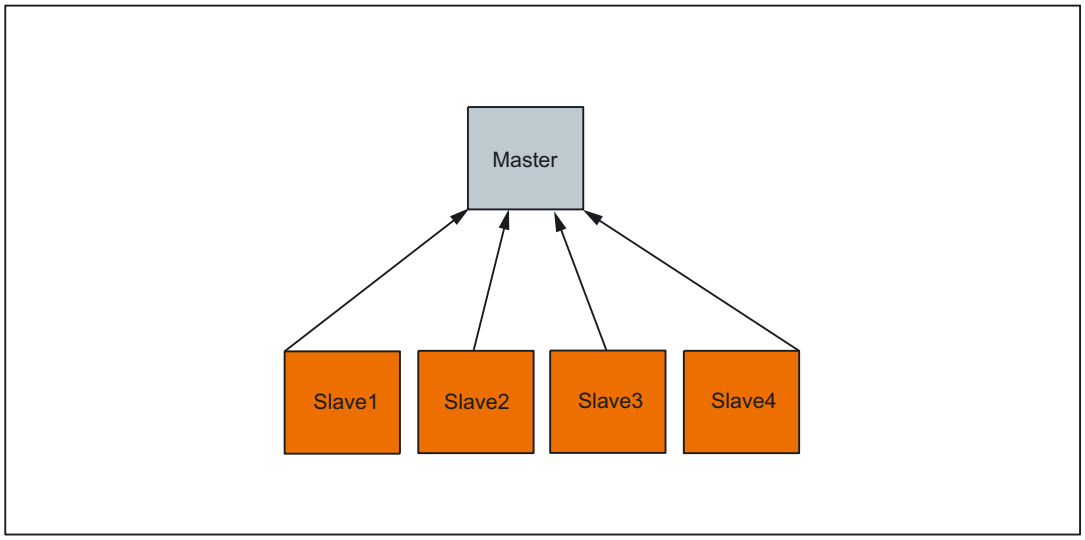


Figure 11-3 Four coupling relationships with the same master axis

11.2.2 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.

No positional deviation control is implemented between master and slave axes. A torque compensatory controller divides the torque evenly over the master and slave axes.

An additional torque can be used to achieve a tension between the master and slave axis.

If different motors are used, individual weighting factors can be used to adapt the torque distribution.

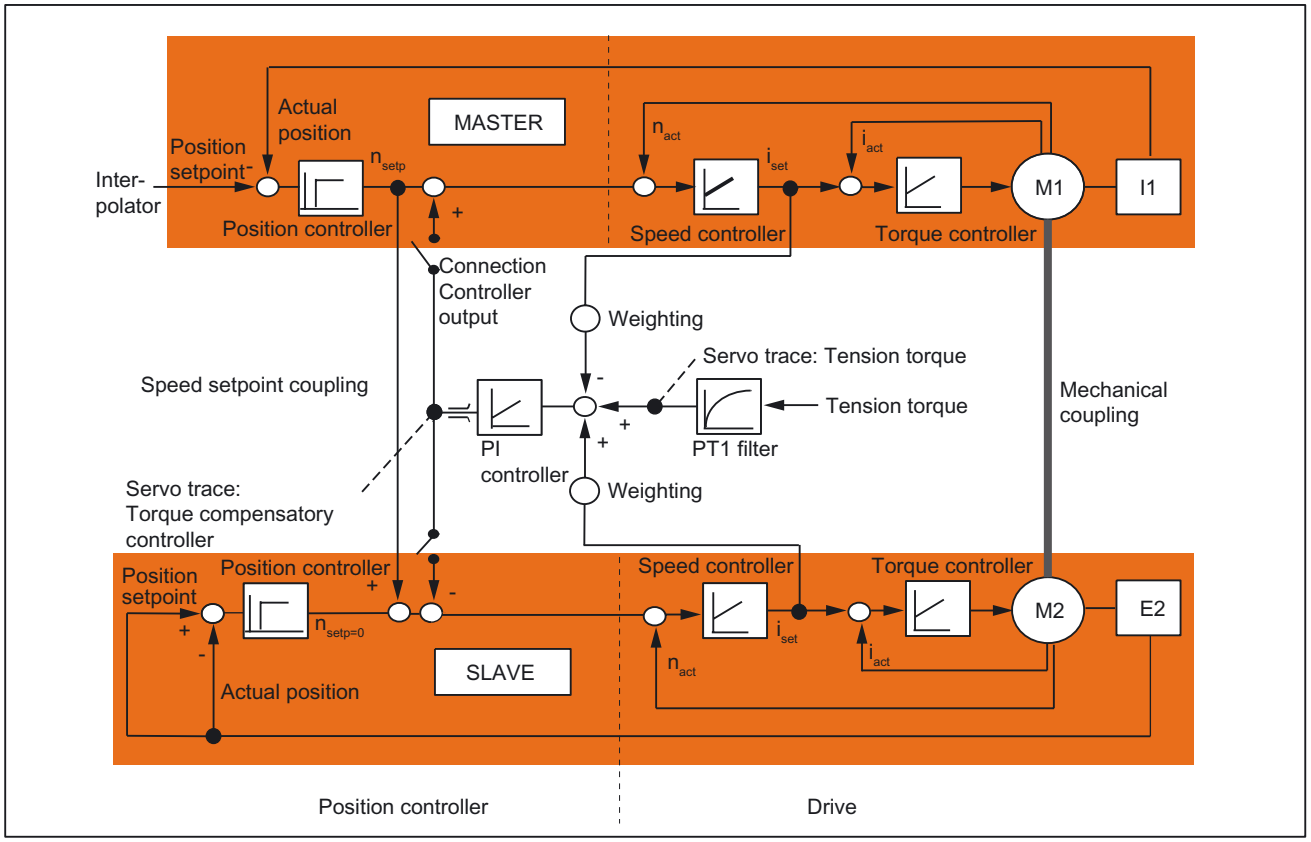


Figure 11-4 Control structure

11.2.3 Configuring a coupling

Static

A master-slave coupling is configured only in the slave axis.

This must be assigned permanently to one of the channels. Each slave axis is assigned one master axis for speed setpoint coupling and one for torque compensatory control.

In the default setting, the same master axis is used for torque compensatory control as for speed setpoint coupling.

The assignments done in the following machine data are automatically active in each control start-up.

MD37252 $MA_MS_ASSIGN_MASTER_TORQUE_CTR

MD 37250 $MA_MS_ASSIGN_MASTER_SPEED_CMD

Dynamics

The program commands `MASLDEF` and `MASLDEL` can be used to change the assignment from the part program dynamically.. This type of configuration can change the static configuration but does not have any reverse effect on the associated machine data.

The following instruction is used to assign one or more slave axes to a master axis.

MASLDEF(Slv1, Slv2, ..., master axis)

The following instruction cancels assignment of the slave axes to the master axis and simultaneously uncouples the current coupling, like MASLOF.

MASLDEL(Slv1, Slv2, ...,)

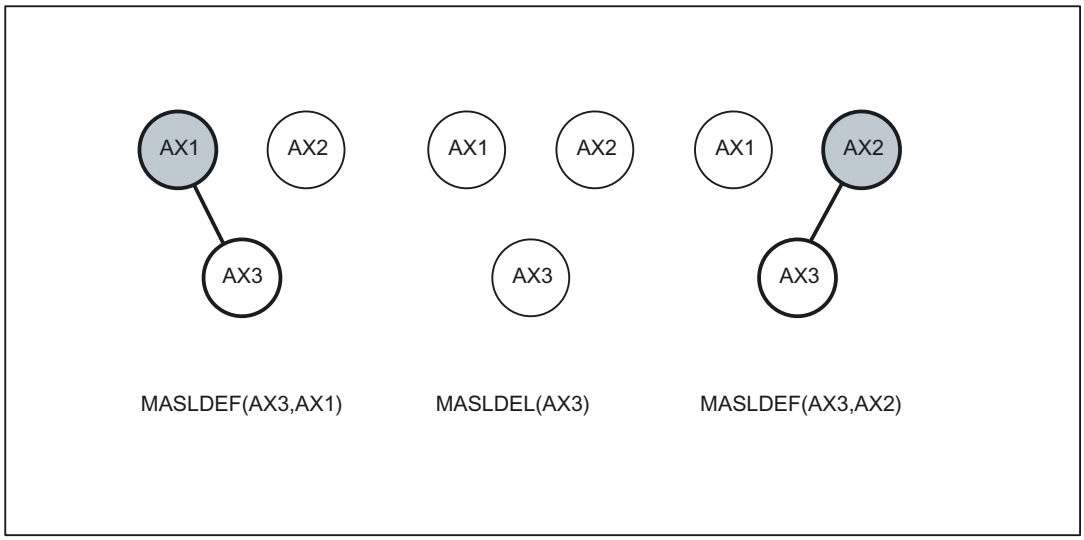


Figure 11-5 Varying configuration of master axis

Changing the configuration has no effect in the coupled state. The change is not accepted until the axes are next uncoupled.

Unlike static assignment, the master axis for torque compensatory control always corresponds to the speed setpoint coupling.

A plausibility check is not carried out until the coupling is closed. In the event of multiple assignment, Alarm 26031 is issued.

An assignment made with MASLDEF is retained after a mode change, reset or end of part program.

Note

To implement a standard assignment in each reset, you can add the corresponding MASLDEF and MASLDEL instructions to the PROG_EVENT.SPF user application. The event controlled is configured using MD20108 $MC_PROG_EVENT_MASK = 4.

11.2.4 Torque compensatory controller

A PI controller calculates a load-side additional speed setpoint from the torque difference between the master and slave axes. This is applied as standard to the command speed setpoint in the master and slave axes with different signs in each case.

If one master and several slaves axes are used, this distribution can cause to instabilities. The output of the torque compensatory controller should only be applied in the slave axes:

MD37254 $MA_MS_TORQUE_CTRL_MODE = 1

The torque setpoints used for torque compensation control are smoothed in the drive. The corner frequency of the PT1 filter is entered in the following machine data:

MD1252 $MD_TORQUE_FILTER_FREQUENCY

The same value should be set in the master and slave axes.

The gain factor MD37256 $MA_MS_TORQUE_CTRL_P_GAIN corresponds to the percentage ratio of the maximum axis speed.

MD32000 $MA_MAX_AX_VELO to the drive nominal torque = MD1725 / 8 of the slave axis.

The I component is disabled in the default setting.

The integral time MD37258 $MA_MS_TORQUE_CTRL_I_TIME is entered in seconds.

The torque compensatory controller output is actively limited to the MD37260 $MA_MS_MAX_CTRL_VELO value.

In the settings MD37256 $MA_MS_TORQUE_CTRL_MODE = 3 or MS_TORQUE_CTRL_P_GAIN = 0, the torque compensatory controller is inactive.

The torque distribution can be parameterized via the input variables of the torque compensatory controller. The slave axis drive torque is weighted using MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE, and the master axis drive torque is automatically weighted using (100 - MS_TORQUE_WEIGHT_SLAVE).

If motors with different rated torque values are used, the 50% to 50% standard distribution must be adapted to suit.

A mechanical coupling is absolutely necessary when the torque compensatory controller is used. Otherwise, the drives involved could accelerate from standstill.

Activation/deactivation via the PLC SW 6.4 and higher

The torque compensatory controller can be switched on and off directly via the PLC interface signal DB31, ... DBX24.4.

For this, the following machine data must be set:

MD37255 $MA_MS_TORQUE_CTRL_ACTIVATION=1

The activated status can be read back in DB31, ... DBX96.4. MS_TORQUE_CTRL_MODE is then only used for configuring the torque distribution.

11.2.5 Tension torque

By specifying a additional torque MD37264 $MA_MS_TENSION_TORQUE, you can achieve a tension between the master and slave axis when the torque compensatory controller is active. The tension torque is entered as a percentage of the rated torque and is active straight away.

The tension torque tension torque is applied via a PT1 filter. Specifying a filter time constant MD37266 $MA_MS_TENSION_TORQ_FILTER_TIME > 0 activates the filter.

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. To prevent unnecessary heating in the motor, you can reduce the tension torque when the motor is at standstill.

Specifying a tension torque without a mechanical coupling produces axis movement.

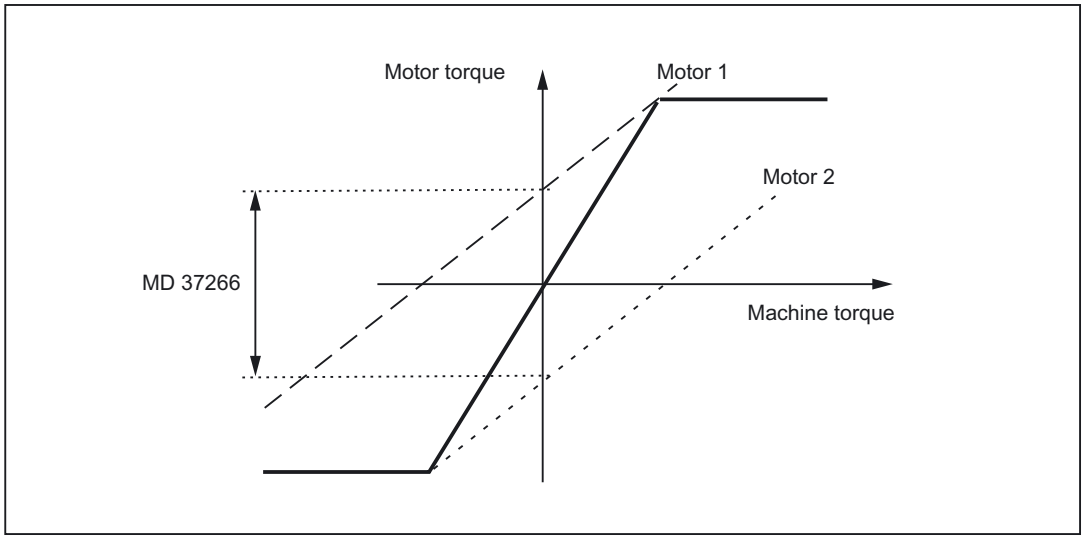


Figure 11-6 Resulting tension torque

11.2.6 Activating a coupling

The type of activation for a master-slave coupling is defined in the following machine data:

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE

Depending on the machine configuration, a distinction is made between a permanent and a temporary master-slave coupling.

Only a temporary master-slave coupling that is configured via machine data (MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 0), can be closed and disconnected using the axial PLC interface signal "Master/Slave ON" (DB31, ... DBX24.7) or within a part program using the following commands:

MASLON(slave axis1, slave axis2, ...)

MASLOF(slave axis1, slave axis2, ...)

The setpoint status of the coupling always corresponds to the last specification made.

The current coupling status can be read back in the slave axis via PLC interface signal "Master/slave coupling active" (DB31, ... DBX96.7).

In the part program and from the synchronized actions, the current coupling status can be output via the **system variable** of the slave axis $AA_MASL_STAT.

SW 6.4 and higher

The instruction MASLOFS(Slv1, Slv2, ...) can be used too disconnect the coupling in the same way as MASLOF and decelerate the slave spindle automatically.

Note

A permanent coupling (MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 1) does not require explicit activation.

For more information about how on a permanent master-slave coupling for the slave axis, PresetOn can be used to synchronize the actual value to the same value as the master axis, see:

References:

/PGA/ Programming Manual, Scheduling, Chapter "Master/Slave Combination"

Example

For an example of how to configure the master-slave coupling between AX1=master and AX2=slave, please see Chapter "Examples", "Speed/torque coupling"

Control system response

The control system response on POWER ON, mode changes, RESET, block searches and Repos is as follows:

- A master-slave coupling activated via PLC or `MASLON` instruction is retained after a mode change, RESET or end of part program.
- `MASLON / MASLOF / MASLOFS` becomes effective on block search. Changes in the positions of coupled slave axes and spindle speeds must be computed separately by the user (see "Block search", Chapter "Response in conjunction with other functions").

11.2.7 Response on activation/deactivation

Activating/deactivating during axis standstill

Activation/deactivation is not active until the axis next comes to a standstill. If the specification is changed, the sequence is the same as for axis replacement. The coupling is closed when the axis comes to a standstill. The coupled axes must be in feedback control mode.

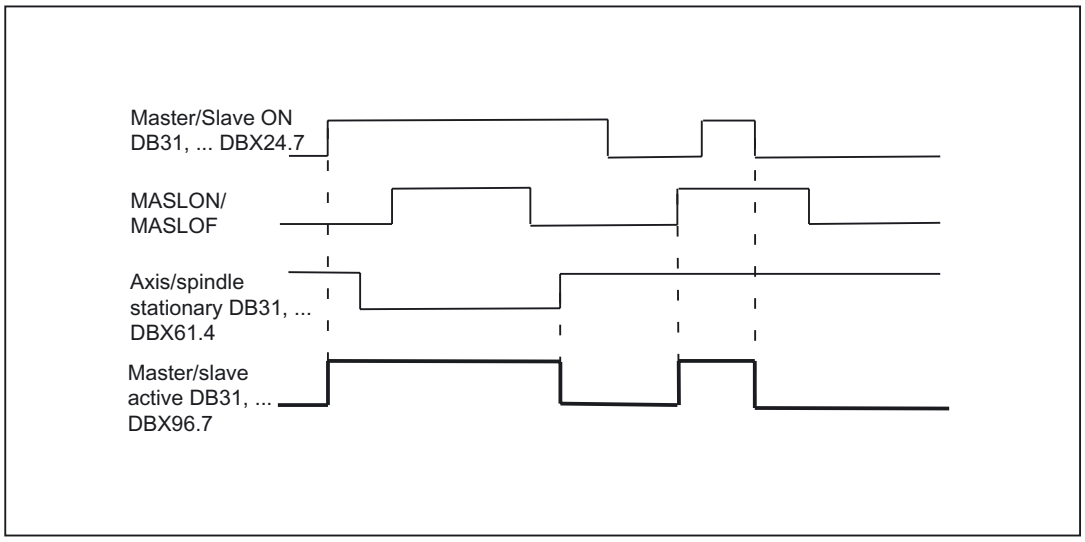


Figure 11-7 Activation procedure

Block stepping is halted for `MASLON` until the coupling has actually been closed. During this time, the associated channel operating message "Master-slave switchover inactive" is displayed on the MMC/HMI.

Activation/deactivation during motion, SW 6.4 and higher

Activation/deactivation of the coupling in motion has been implemented only for spindles in speed control mode. For axes and spindles in positioning mode, switchover is still carried out when the axis is at a standstill.

Activation during motion

The coupling procedure at different speeds is divided into two phases.

Phase 1

Closure of the coupling is requested with interface signal IS "Master/slave on" (DB31, ... DBX24.7). The slave spindle accelerates or decelerates along the ramp with the dynamic response available to it until it reaches the setpoint speed of the master spindle.

When the setpoint speed is reached, the coupling is closed and the interface signal "Coupling active" (DB31, ... DBX96.7) is set.

If the master spindle is accelerated during the coupling process, the first phase is extended according to the existing difference in dynamics between the master and slave spindles.

Phase 2

In the second phase, the actual difference speed between the master and slave spindle(s) is used to generate the following synchronism signals.

NST "Speed tolerance coarse" (DB31, ... DBX96.3)

NST "Speed tolerance fine" (DB31, ... DBX96.2)

The associated limits are defined via the following machine data:

MD37270 $MA_MS_VELO_TOL_COARSE ("Tolerance coarse")

MD37272 $MA_MS_VELO_TOL_FINE ("Tolerance fine").

Note

The "Tolerance coarse" signal can be used to implement a PLC monitoring function that checks a coupled group for loss of speed synchronism. The "Tolerance fine" signal can be used to derive the time for mechanical closure of the coupling and to activate the torque compensatory controller directly.

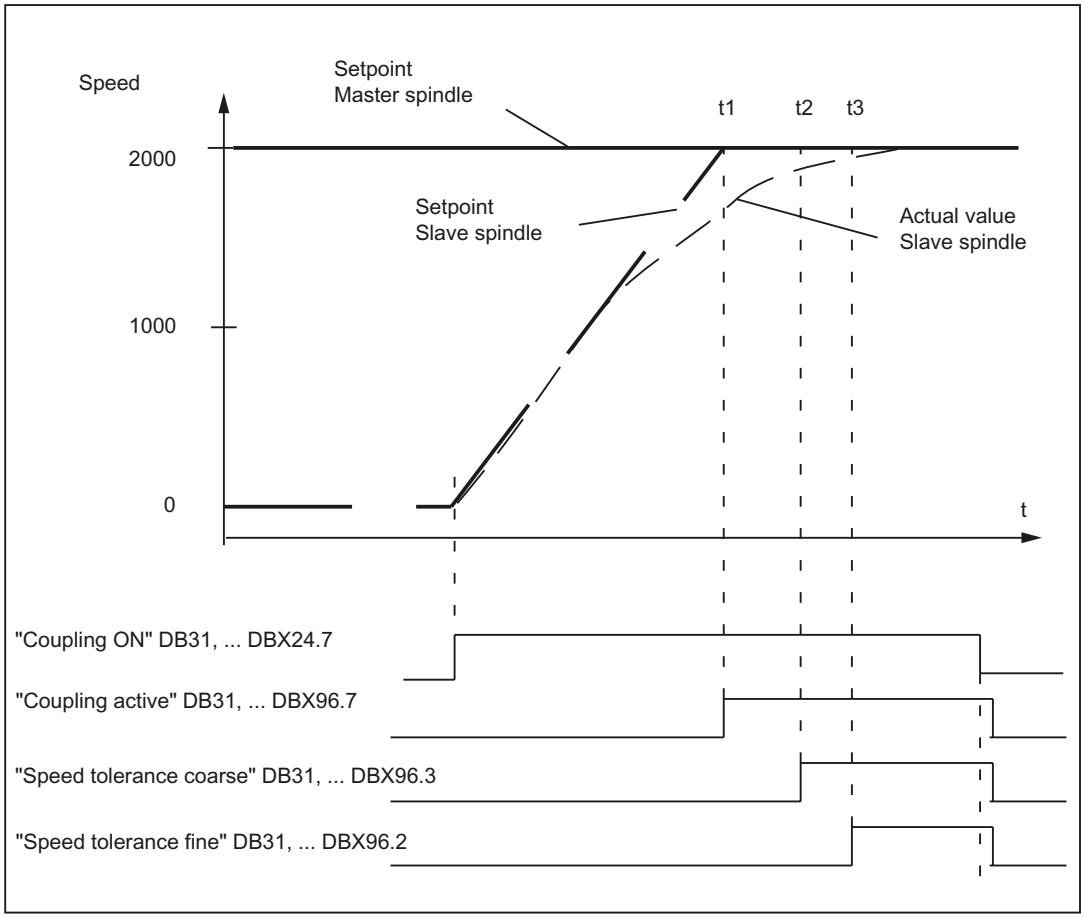


Figure 11-8 Coupling procedure between two spindles with different speeds

Deactivation during motion

An active coupling is disconnected using the MASLOF instruction.

This instruction is executed directly for spindles in speed control mode. The slave spindles that are rotating at this point in time retain their last speed until a new speed is programmed.

You can use the MASLOFS instruction to decelerate slave spindles automatically when disconnecting the coupling. For axes and spindles in positioning mode, the coupling is still only disconnected at standstill.

Note

The implicit preprocessor stop is omitted for `MASLON` and `MASLOF`. The missing preprocessor stop means that the $P system variables of the slave spindle do not supply updated values until reprogrammed.

Coupling characteristics (SW 6.5 and higher)

For spindles in speed control mode, the coupling characteristics of the `MASLON`, `MASLOF`, `MASLOFS`, `MASLDEL` instructions and the PLC with NST "Master/Slave ON" (DB31, ... DBX24.7) is defined explicitly via the following machine data:

MD37263 $MA_MS_SPIND_COUPLING_MODE

MD37263 = 0

Coupling and disconnection take place only at standstill.

The current coupling state is retained until all axes involved have actually come to a standstill. The MASLOFS and MASLOF instructions are identical; the slave spindle is not decelerated automatically.

MD37263 = 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 automatically.

11.2.8 Axial interface signals

When a master/slave coupling is requested, the PLC axis enables "Servo enable" (DB31, ... DBX2.1) and "Pulse enable" (DB31, ... DBX21.7) of the slave axis are derived directly from the specifications of the master axis. The separate PLC axis enable signals have no effect.

Removing the servo enable in the master axis also results in an interpolated deceleration of the slave axis that is carried out within the configured time of MD36610 $MA_AX_EMERGENCY_STOP_TIME. The associated speed and current controller enable signals für individual axes will be removed only after the time of MD36620 $MA_SERVO_DISABLE_DELAY_TIME has elapsed.

To further ensure the same braking response, the time set in the following machine data should be the same for all coupled axes:

MD36620 $MA_SERVO_DISABLE_DELAY_TIME

The same applies to the following drive machine data:

MD1403 $MD_PULSE_SUPPRESSION_SPEED

MD1404 $MD_PULSE_SUPPRESSION_DELAY

If the "Current controller active" (DB31, ... DBX61.7) or "Speed controller active" (DB31, ... DBX61.6) drive status signals are missing in the master or slave axis, the PLC interface signal "Master/slave active" (DB31, ... DBX96.7) is reset in the slave axis at standstill. When the master and slave axes return to closed-loop control mode, IS Master/Slave active (DB31, ... DBX96.7) is set on the slave axis.

With IS (DB31, ... DBX24.4), the torque compensatory controller is activated by the PLC. The status of the torque compensatory controller can be read from IS "Master/slave comp. contr. active" (DB31, ... DBX96.4).

Note

If the coupling is closed, the slave axis operates in speed control mode; status signal "Position controller active" (DB31, ... DBX61.5) is not enabled.

11.2.9 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 initiating the monitoring functions.

To achieve the same braking response for all coupled axes in the event of a fault, the same alarm reaction is applied to the entire coupling grouping when the coupling is active.

When correcting fault states, repositioning of slave axes on the interrupt point is suppressed.

11.2.10 Response in conjunction with other functions

Function Generator

To calibrate the speed control circuit for a closed master-slave coupling, slave axis MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE

should be set to a smaller value. Traversing of a coupled-motion slave axis is not prevented by the torque compensatory controller in this case.

Reference point approach

If the coupling is closed, only the master axis can be referenced. Referencing of slaves 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 not closed, each axis can be referenced separately as usual.

Compensation

Position offsets of the slave axis, such as spindle pitch errors, backlash, temperature and sag offsets are computed but not active because there is no position controller.

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

The Kv factor of the master axis is copied to the slave axis for an existing coupling and is thus also active in the slave drive. This is an attempt to achieve the same control response in the drive of the master and slave axis as far as possible. MD32640 $MA_STIFFNESS_CONTROL_ENABLE must be configured identically in all coupled axes.

Speed/torque feedforward control

The feedforward control in the slave axis does not have to be activated explicitly. The current settings of the master axis apply. The speed feedforward value of the master axis is already incorporated in the speed setpoint of the slave axis. For an active torque precontrol, the load-side torque precontrol value of the master axis is also active in the slave drive.

The mechanical situation changes in coupled mode. Axial settings must be adjusted 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.

Moving to fixed end stop

The travel to fixed stop 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 constraints:

- Torque distribution between the master and slave axes is not possible during clamping as the torque compensatory controller is deactivated 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

As the slave axis is traversed via the master axis speed setpoint, the axial setpoint limit MD36933 $MA_SAFE_DES_VELO_LIMIT is inactive in the coupled slave axes. All safety monitoring functions remain active in the slave axes however.

Weight counterbalance

The additional torque for the electronic weight counterbalance
MD32460 $MA_TORQUE_OFFSET

is computed in the slave axis irrespective of the coupling status.

Gear stage change with active 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 block to be specified, the following machine data must be set to the value 2:

MD35590 $MA_PARAMSET_CHANGE_ENABLE (parameter set change possible)

In the event of a gear stage change for the master/slave spindle, the associated parameter set index can be activated by the PLC via the VDI interface.

Note

For more information about gear stage change and parameter sets for changes in spindle mode, see:

References:

/FB1/ Function Manual, Basic Functions; Spindles (S1)

/FB1/ Function Manual, Basic Functions; Various Interface Signals (A2)

Interface signals from/to axis/spindle

Axis container

If a coupled slave axis is configured in an axis container, alarm "4025 Switch axis container %3 not permitted: Master-slave active channel %1 Axis %2" is output. The axis container may not be advanced because the coupling is active.

SW 6.4 and higher

In the event that masters change, dynamic configuration can be used to match 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 container rotation using a different spindle in each case, the old coupling must be disconnected before the rotation, the configuration deleted and the new coupling closed after the rotation.

Example for cyclic coupling sequence (position=3/container=CT1)

```
MASLDEF(AUX,SPI(3))                    ; S3 master for AUX
MASLON(AUX)                            ; Coupling in for AUX
M3=3 S3=4000                           ; Machining ...
MASLDEL(AUX)                           ; Clear configuration and uncoupling
AXCTSWE(CT1)                           ; Container rotation
```

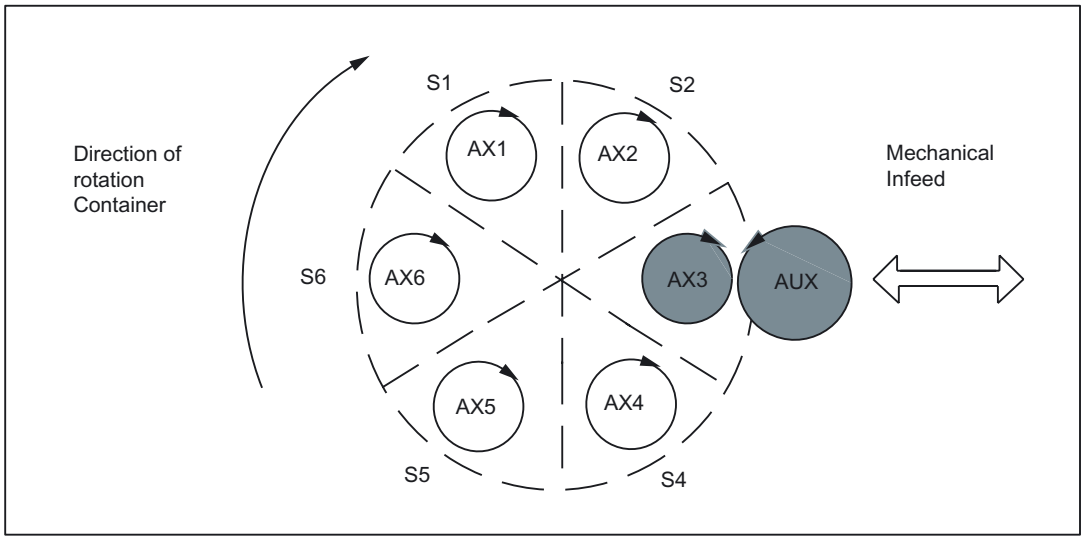


Figure 11-9 Coupling between container spindle S3 and auxiliary motor AUX (prior to rotation)

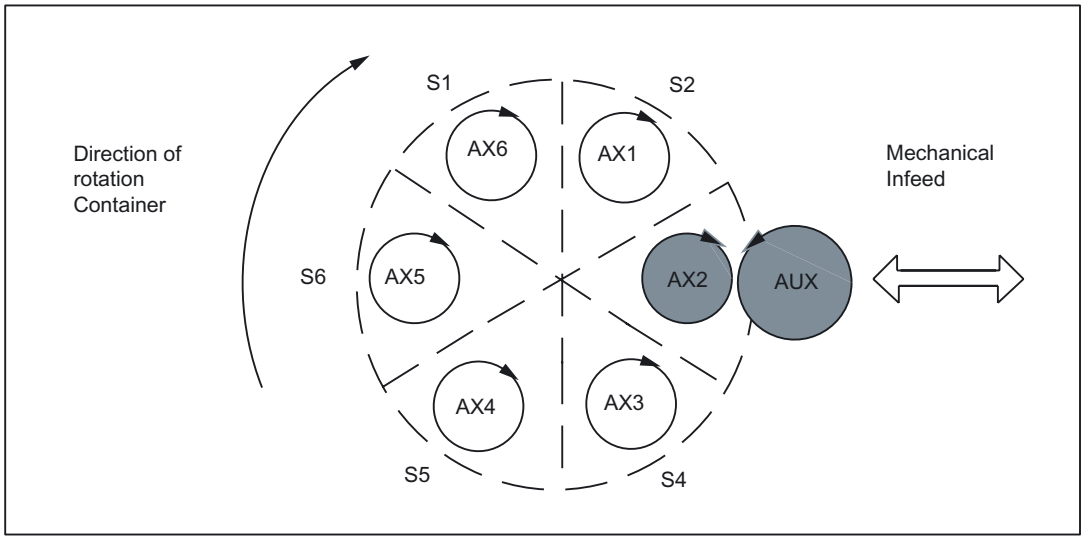


Figure 11-10 Coupling between container spindle S3 and auxiliary motor AUX (after to rotation)

Hardware and software limit switches

Crossing of hardware and software limit switches is detected in coupled axes; in the coupled state, the software limit switch is generally crossed on slave axes. The alarm is output on the slave axis, while braking is initiated via the master axis.

The path traveled after detection of the slave software limit switch equals the distance required by the master axis to brake the coupling.

The master axis controls the movement away from the limit switch, since the coupling cannot be disconnected until the cause of the alarm has been eliminated.

Block search

The SERUPRO "block search with calculation" function can be used without restriction in combination with a permanent master-slave coupling in the machine data:

MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE=1 (permanent master-slave coupling)

The following restrictions apply when the coupling is programmed using MASLON and MASLOF commands:

- The coupled axes must be in the same channel when the block search is executed. If they are not in the same channel, the block search is aborted with alarm 15395.
- The coupled axes are operated on the same NCU.
- Once the block search has been completed, the associated axis positions and speeds must be modified subsequently by the user via a system ASUB (asynchronous subroutine) "PROGEVENT.SPF" of the coupling status. System variables are available for this purpose.

$P_PROG_EVENT

This variable provides information about the event, which activated the subroutine. A value of 5 stands for block search.

$P_SEARCH_MASLC[slave axis identifier]

The variable stands for alteration of the coupling status during a block search.

$P_SEARCH_MASLD[slave axis identifier]

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 identifier]

This variable indicates the current coupling status.

- The system ASUB "PROGEVENT.SPF" must be stored under /_N_CMA_DIR/_N_PROG_EVENT_SPF so that it can be accessed by the control system.

Example 1 for PROGEVENT.SPF:

```
N10 IF $P_PROG_EVENT==5                    ; Block search active
N20 IF (($P_SEARCH_MASLC[Y]<>0)            ; The coupling state has
 AND ($AA_MASL_STAT[Y]<>0))                ; changed during block search
                                           ; current state is "closed".
N30 MASLOF(Y)                              ; Disconnect coupling
N40 SUPA Y=$AA_IM[X]-$P_SEARCH_MASLD[Y]
                                           ; Compensate position offset
                                           , via slave axis
N50 MASLON(Y)                              ; Close coupling
N60 ENDIF
N70 ENDIF
N80 REPOSA
```

Example 2 for PROGEVENT.SPF:

```
N10 IF $P_PROG_EVENT==5                    ; Block search active
N20 IF (($P_SEARCH_MASLC[SPI(2)]<>0)       ; The state of the second spindle has
 AND ($AA_MASL_STAT[SPI(2)]==0))           ; changed during block search
                                           ; current state is "disconnected".
N30 M2=$P_SEARCH_SDIR[2]                   ; Update direction of rotation
N40 S2= $P_SEARCH_S[2]                     ; Update speed
N50 ENDIF
N60 ENDIF
N70 REPOSA
```

- In order that the PROGEVENT.SPF subroutine can start automatically, the following machine data must be parameterized accordingly:
  - MD11450 $MN_SEARCH_RUN_MODE = H02
  - MD11602 $MN_ASUP_START_MASK = H03
  - MD11604 $MN_ASUP_START_PRIO_LEVEL = 100

For more application examples, see Chapter "Examples.

Note

For more information about event-driven program calls and block searches in program test mode (SERUPRO), please see:

References:

/FB1/ Function Manual, Basic Functions; K1(BAG, Channel, Program Operation Mode)

11.2.11 Compatibility of SW 6.4 with earlier versions

Implicit preprocessor stop

The implicit preprocessor stop is omitted for `MASLON, MASLOF`.

For spindles in speed control mode, the time at which the coupling is closed or disconnected changes. The coupling is closed or disconnected immediately, without waiting for standstill.

If activation/deactivation is to remain the same despite the new function, a `WAITS` must be programmed explicitly before `MASLOF` as in the example on the right. The coupling is not disconnected until all coupled spindles have come to a standstill.

Software version 6.4 and lower	SW 6.4 and higher
MASLON(S3)	MASLON(S3)
M2=3 S2=1000	M2=3 S2=1000
G4 F4	G4 F4
M2=5	M2=5
MASLOF(S3)	WAITS(2); for compatibility reasons
	MASLOF(S3)

Multiple assignment

The time at which configuration alarm 26031 is output changes from the time at which the control starts up to the time at which an attempt is made to close the coupling. The alarm is acknowledged with a reset.

11.2.12 Constraints in SW 6.4 and higher

See Chapter "Constraints" In addition:

The coupling for axes and spindles in positioning mode is still closed and disconnected only at standstill.

In the coupled state, the acceleration of spindles at the current limit may not provide an adequate adjustment reserve for the torque compensatory controller in order to maintain the desired distribution of torque between master and slave.

Prior to gear change or a star/delta switchover, the master/slave coupling must be deactivated.

The maximum master spindle chuck speed in the following machine data is to be configured less or equal to that of the slave spindles:

MD35100 $MA_SPIND_VELO_LIMIT

The axial speed monitoring function in the following machine data should be adapted to the chuck speed:

MD36200 $MA_AX_VELO_LIMIT

For dynamic configuration, no distinction is made between the speed and torque master. The response corresponds to that of the default setting:

MD37252 $MA_MS_ASSIGN_MASTER_TORQUE_CTR = 0.

11.3 Speed/torque coupling (up to SW 5.x)

11.3.1 General information

The speed/torque coupling (master-slave) function is required for configurations in which two drives are mechanically coupled to one axis. With this type of axis, a torque controller must ensure that each motor produces exactly the same torque, otherwise the two motors would work in opposition.

Master-slave operation possible only with digital 611D drives.

One of the two drives, the "master", is programmed, while the other drive, the "slave", is linked via the speed setpoint coupling.

This function essentially consists of:

- A speed setpoint coupling and
- A torque controller between the master and slave axes

A master-slave operation without permanent mechanical coupling does not make sense because no torque distribution to the common mechanical connection can take place in this case.

When you activate a master-slave coupling, the NC loses the position reference of the slave axis. It is maintained on the real axis by way of a fixed mechanical coupling.

The function is not implemented as a difference position control but only as a coupling on the speed/torque plane. A difference position control would not make sense as it would cause the controllers between the master and slave to work in opposition.

This function allows each axis to be assigned to a master as a slave, which means that several master-slave couplings can co-exist.

To achieve a tensioning between the master and slave, a configurable tension torque can be applied to the torque controller via machine data.

Master and slave axis need not be programmed in the same channel. The speed setpoint coupling is done in position control cycles.

11.3.2 Control structure

The control structure of a master-slave coupling is shown in the following figure For a better overview, only one master/one slave coupling is illustrated.

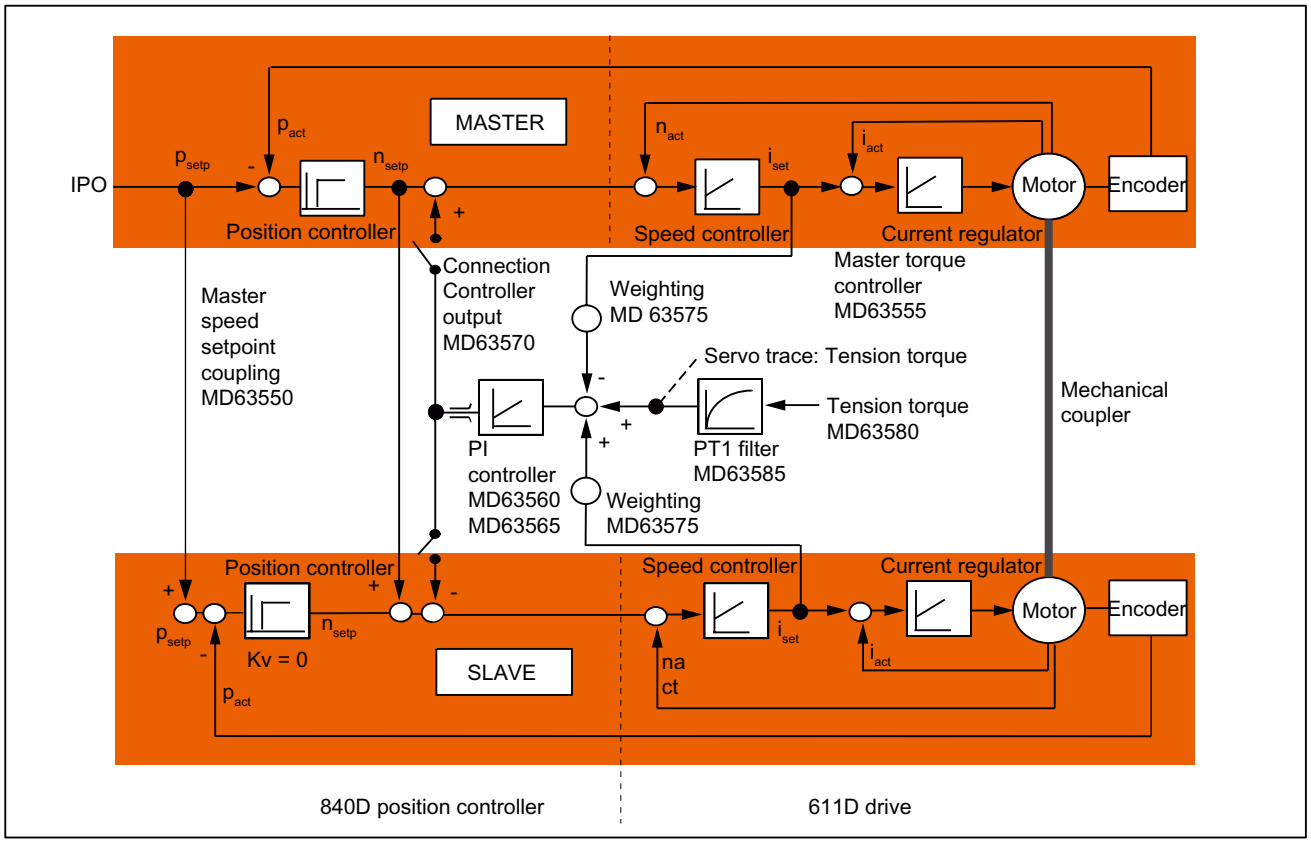


Figure 11-11 Control structure

11.3.3 Configuring a coupling

Defining a coupling

Each axis involved in a master-slave coupling must be assigned to a channel as an NC axis. By means of the following axis specific machine data, a master axis for speed setpoint coupling and a master axis for torque control are assigned to each potential slave axis.

MD63550 $MA_MS_ASSIGN_MASTER_SPEED_CMD

MD63555 $MA_ASSIGN_MASTER_TORQUE_CTRL

In most cases, the same master is used for speed setpoint coupling and torque control. If MD ASSIGN_MASTER_TORQUE_CTRL is set to 0, the master axis for torque control is identical to that for speed setpoint coupling.

Several couplings

A master can be assigned to each slave axis to produce several couplings. In a simple case, the couplings are mutually independent, i.e. each axis is involved in only one coupling. An example of this is a gantry axis with a master-slave coupling on each side.

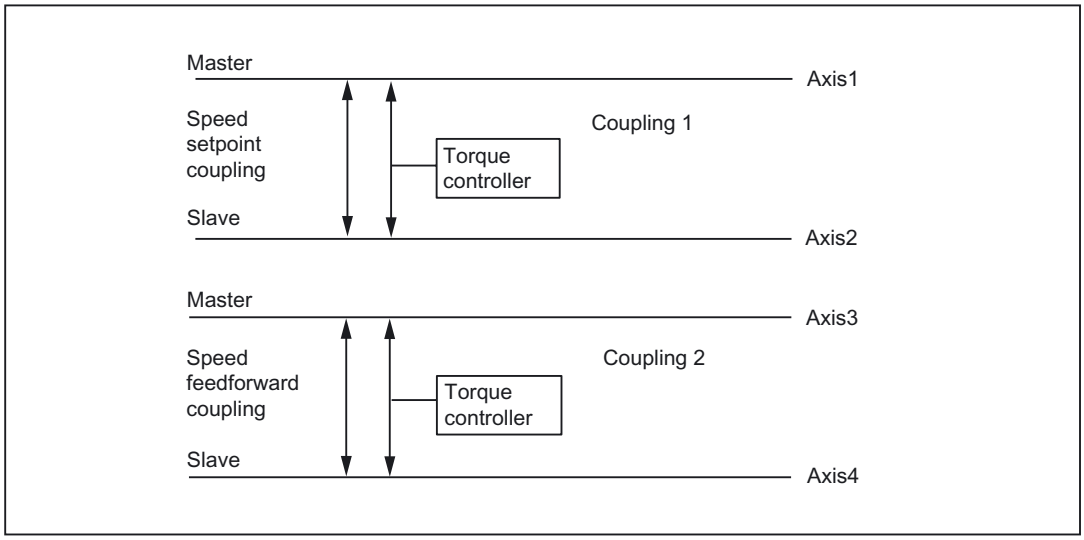


Figure 11-12 Independent master-slave couplings

One master, several slaves

It is also possible to configure master-slave couplings where one axis is the master axis for several couplings. In this example, axis 1 is the master axis for coupling 1 and coupling 2. Please note:

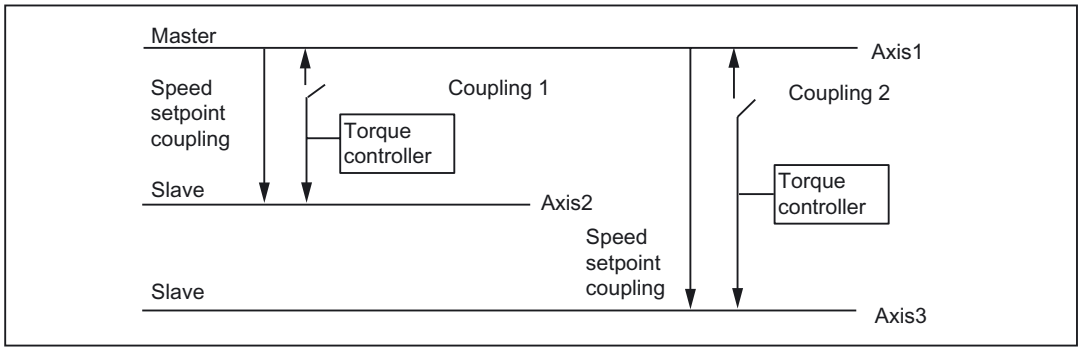


Figure 11-13 One master, two slaves

The torque controller for coupling 1 attempts to maintain the same torque between axis 1 and axis 2 by writing a speed setpoint to axis 1 and axis 2. The torque controller for coupling 2 also tries to maintain the same torque between axis 1 and axis 3. Both controllers would write speed setpoints to axis 1.

In order to ensure a stable system, both controllers must be parameterized so that the controller output is added only to the slave axes (axis 2 and axis 3) but not to the master axis (axis 1). This is achieved by setting the following machine date to 1 (controller output to slave axis only) for both couplings:

MD63570 $MA_MS_TORQUE_CTRL_MODE = 1 (connecting the torque controller output)

Both torque controllers now attempt to match the torque of the slave axis to the torque of the master axis, without adding speed setpoints to the master axis.

Axis in the channel

When the coupling is active, the motion of the slave axis is not displayed in the automatic basic display and the actual value is frozen. If a coupling is always active, i.e. the slave axis is never traversed individually, we recommend that this axis is displayed as the last axis in the automatic basic display. This is achieved by entering this axis in the following machine data as the last axis in the channel:

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

Multiple channels

The master axis and slave axis do not have to be programmed in the same channel. Multi-channel couplings are possible for several active channels.

Axis replacement

An axis replacement between channels may be prepared (MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN), but can be performed only with restrictions. These restrictions are described in Chapter "Constraints", "Speed/torque Coupling (SW 6 and higher)".

Spindles

A master-slave coupling can also include spindles. The slave axis must then always operate in speed control mode and the position controller is deactivated. (DB3x.DBB61.5 = 0).

The master axis can be operated in all spindle modes, open-loop control mode with/without position controller, oscillation mode or positioning mode; even changeover between spindle modes is possible. Restrictions that apply to actual value display are described in Chapter "Constraints", "Speed/torque Coupling (SW 6 and higher)".

Rotary axes

Master and slave axes can also be rotary axes. The restrictions described in Chapter "Constraints", "Speed/torque Coupling (up to SW 5)", apply.

Motors rotating in opposite directions

If the motors have been mounted to run in opposite directions, the traversing direction is inverted for one of the drives with MD32100 $MA_MOTION_DIR. In this case, the speed setpoint and the output of the torque controller are calculated correctly; there is no need to set further machine data.

Different motor speeds

The master and slave axis can have different gear reduction ratios between the motors and the mechanical coupling. With these types of axes, the master and slave rotate at different speeds. When the coupling is active, the same load speed is standardized internally so that different motor speeds are possible for the master and slave without having to set further machine data.

Speed feedforward control

If speed feedforward control is active in the master axis, speed feedforward control must also be activated in the slave axis. Non-active speed feedforward control in the slave axis causes a "Contour monitoring" alarm in the slave axis.

Computing time load

Each master-slave coupling places a load on the position control level and the interpolation level. The table shows the computing time required depending on the NCU hardware.

NCU	Position control	Interpolator level
572	1st coupling 0.120 ms Each addtional coupling +0.050 ms	1st coupling 0.100 ms Each addtional coupling +0.020 ms
573	1st coupling 0.040 ms Each addtional coupling +0.020 ms	1st coupling 0.030 ms Each addtional coupling +0.010 ms

Configuration alarms

During power ON of the control, the configuration machine data are checked and alarms set as necessary:

If the master and slave axes are identical for speed coupling, the alarm "75150 Slave axis AX1 and master axis are identical for speed setpoint coupling" is present after POWER ON.

If the master axis and slave axis are identical for torque control, alarm "75151 Slave axis AX1 and master axis identical for torque controller" is present.

All axes of the mode group follow on; the alarms can only be reset with POWER ON.

11.3.4 Torque controller

The torque controller between master and slave ensures even torque distribution between the master and slave axis. The input variable of the controller is the torque difference Mdiff between the master and slave axis; the output is a setpoint speed nset, which is applied to the master and slave axes.

The controller consists of a P component and an I component. Both parts must be parameterized separately.

The machine data of the slave axis is always relevant for the configuration of the particular master-slave torque control.

P controller

The P controller calculates a speed setpoint nset by multiplying the torque difference Mdiff by a gain factor Kp. The resulting speed setpoint is added to the master and slave axes.

nset = Mdiff * Kp

The P gain Kp of the torque compensatory controller has the dimension [(mm/min)/Nm].

The P gain factor is entered in axial MD63560 $MA_MS_TORQUE_CTRL_P_GAIN as a percentage value of the ratio:

Maximum drive speed [mm/min] to rated torque [Nm].

The maximum drive speed is the contents of MD32000 $MA_MAX_AX_VELO. The rated torque is obtained from the product of the following machine data:

MD1113 $MD_TORQUE_CURRENT_RATIO

MD1118 $MD_MOTOR_STANDSTILL_CURRENT

Only the data of the slave axis are relevant for the torque controller.

Example:	
Maximum drive velocity of the slave axis	30000 mm/min
Motor rated torque of the slave axis	10 Nm
MS_TORQUE_CTRL_P_GAIN	15%
Kp: (30000/10) * 15%	450 (mm/min)/Nm

I controller

The I controller calculates a speed setpoint nset by multiplying the torque difference Mdiff by a gain factor Ki.

nset = Mdiff * Ki

The gain factor Ki of the I controller is parameterized via the reset time of the torque compensatory controller I_TIME. Ki can only be calculated if the gain factor of the P controller Kp ≠ 0. The I controller is not active unless the P component is also activated.

Ki = 1/ position controller cycle * I_TIME * Kp

The integral time is entered in the axial MD63565 $MA_MS_TORQUE_CTRL_I_TIME in seconds.

The default setting 0 deactivates the I component if the P component already ensures appropriate torque distribution.

Limiting the controller output

MD63600 $MA_MS_MAX_CTRL_VELO can be used to limit the output of the controller to a maximum value. The value is entered as a percentage value relative to the maximum speed of the slave-axis. The default is 100%. The limit works in both a positive and a negative direction.

Connection of the torque control output

An addtional MD63570 $MA_MS_TORQUE_CTRL_MODE can be used to connect the output of the torque controller freely to the master and slave axes. In most cases, the output value is applied to the master and slave. The user is responsible for setting parameters meaningfully. The MD of the slave axis is the important setting.

Meaning:

0: Switch controller output to master and slave.

1: Switch controller output only to slave.

2: Switch controller output only to master.

3: The controller is deactivated; if the coupling is active, only speed setpoint coupling applies.

Even if the controller output is not connected to an axis, the controller is calculated.

Weighting

MD63575 $MA_MS_TORQUE_WEIGHT_SLAVE is used to apply a weighting to the input variables of the torque compensatory controller in order to enable parameterizable torque distribution over the two drives. If the motors are identical and the same drive parameters are to be set for the motors to produce the same torque, the standard parameterization 50% is recommended. The MD refers to the torque of the slave axis and the torque of the master axis is weighted with the difference between the MD and 100%.

Example:

Slave is intended to be 30% involved in the total torque. 70% are provided the by master axis.

MD37268 $MA_MS_TORQUE_WEIGHT_SLAVE = 30

Tension

An axiales MD63580 $MA_MS_TENSION_TORQUE can be used to apply a constant tension torque as input to the torque control. This tension torque is applied continuously and produces a mutual tensioning of the coupled drives.

The MD of the slave axis is relevant for the tension of a coupling.

The tension torque can be positive or negative. The value to be input is a percentage value relative to the rated torque of the slave axis. The rated torque is obtained from the product of the following drive machine data:

MD1113 $MD_TORQUE_CURRENT_RATIO

MD1118 $MD_MOTOR_STANDSTILL_CURRENT

The tension torque is active immediately after a change. In this way, it is possible to implement various tension torques to suit individual machining situations.

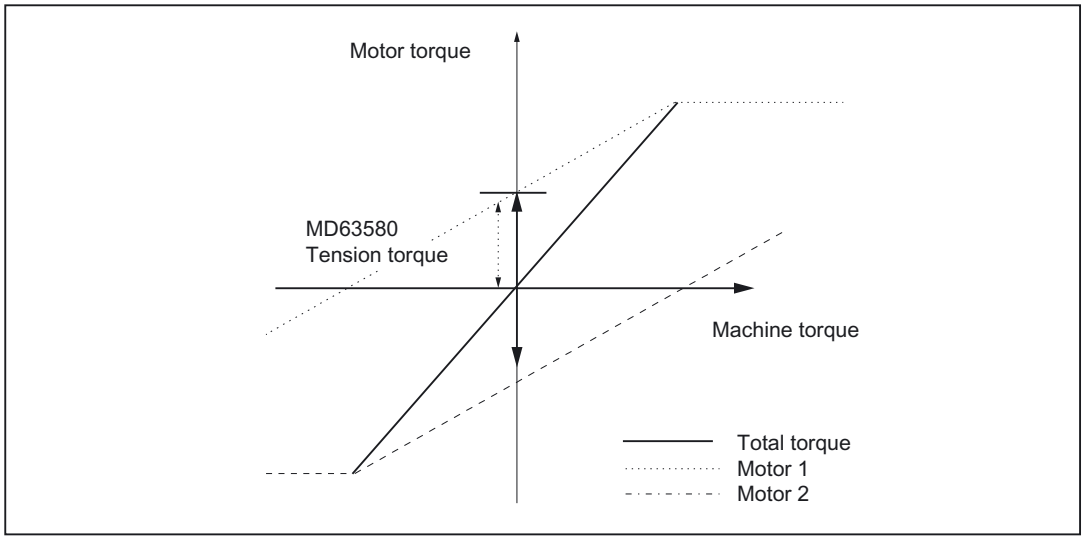


Figure 11-14 Tension torque

PT1 filter

The tension torque is applied to the torque control via a PT1 filter. The PT1 filter ensures a continuous increase or decrease of the tension torque when the tension torque value is changed. Without the PT1 filter, changing the tension torque causes a step change in the speed setpoint at the torque controller output when the controller is operated without an I component. The PT1 filter is configured using the following machine data:

MD63585 $MA_MS_TENSION_TORQ_FILTER_TIME

The time is entered in seconds. Enter 0 to deactivate the PT1 filter.

Note

The function ensures distribution of the torque-producing currents (Iq) and not distribution of the torques.

This means that torque distribution is also assured on FSD synchronous motors (no field weakening). In contrast, however, only current distribution is assured on MSD asynchronous motors in the field-weakening range. Torque distribution is assured only on motors of the same type operating simultaneously at the same speed. If MSD motors are not operated in the field-weakening range, torque distribution can also be assured for different motor types operating at different speeds.

11.3.5 Presetting the drive machine data

P component in the speed controller

If axes are put into operation individually in a master-slave coupling, whereby an individual axis takes the full load, the P component in the speed controller must then be halved in the two axes. This is the only way to ensure that overshoot is avoided when traversing the axis with active coupling.

11.3.6 Activating and deactivating a coupling

Conditions for activation and deactivation

A coupling is activated or deactivated only under the following conditions:

- Master and slave axes are operating in position control mode (DB3x.DBB 61.5) or, in the case of spindles, in speed control mode.
- Master and slave axis are at standstill (DB3x.DBB 61.4).
- The channels of the master and slave axes are in the "RESET" state.
- (DB2x.DBB35.7). This condition can be switched on or off by means of a bit in MD63595 $MA_TRACE_MODE.

If the master axis and slave axis are in different channels, both channels must be in the "RESET" state. In the event of axis replacement, the state of the master channel is decisive:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

A channel is in the "RESET" state after the end of a program (M30) or after a "RESET" from the operator panel.

Master-slave coupling after POWER ON always active

MD63590 $MA_MS_COUPLING_ALWAYS_ACTIVE defines a coupling as always active. The coupling is activated as soon as the conditions for activation of a coupling are satisfied after a POWER ON. It can no longer be deactivated, i.e. it is no longer possible to operate the drives separately.

The machine data of the slave axis are always relevant for a coupling.

If it is not possible to activate a coupling after POWER ON because, for example, the axes are not in the position control state, alarm "75160 slave axis AX1, master-slave coupling not active" is output. Further attempts are made to close the coupling. Once all the conditions have been satisfied, the coupling is closed and the alarm deleted.

Activating and deactivating a master-slave coupling via PLC signal

A coupling is activated or deactivated via an axis-specific PLC signal "to axis". Only the signal to the slave axis is relevant here. The signal resides in the technologies area.

DB3x.DBB24.7	"Activate master-slave coupling"
	1 = Activate master-slave coupling
	0 = Deactivate master-slave coupling

If one of the conditions for activation or deactivation is not satisfied, the slave axis does not react to the PLC signal, i.e. the status of the coupling remains unchanged. No NC alarm is output.

Example:

- A part program is processed in channel 1, channel state: "active".
- A master-slave coupling is active, master axis and slave axis are in channel 1, PLC signal to slave axis DB3x.DBB24.7 = 1.
- The coupling is to be deactivated, PLC sets DB3x.DBB24.7 = 0.
- Since the channel is not in the "RESET" state, the coupling is not deactivated.
- The coupling is not deactivated until the part program is terminated with M30 or RESET.

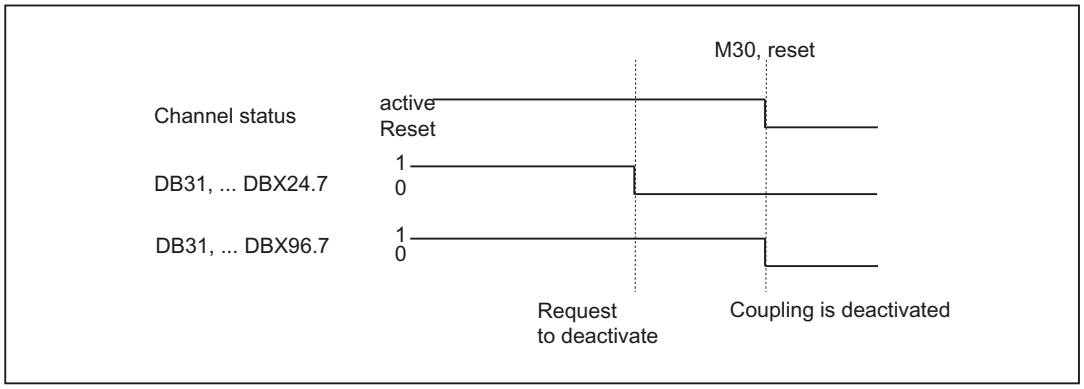


Figure 11-15 Deactivating a coupling

PLC signal: State of a master/slave coupling

The status of a master-slave coupling is displayed in an axis-specific VDI signal "from axis". The machine data of the slave axis are always relevant for a coupling. This signal is set irrespective of whether the coupling is always active (MD63590) or has been activated via the PLC (DB3x.DBB24.7).

DB3x.DBB96.7	"State of master-slave coupling"
	1 = Master-slave coupling active
	0 = Master-slave coupling not active

11.3.7 System response when a coupling is active

PLC signal: Traversing the slave axis

If a slave axis is traversed via the master axis when the coupling is active, the following PLC signals are output depending on the travel state:

DB3x.DBB60.6	"Exact stop fine"
DB3x.DBB60.7	"Exact stop coarse"
DB3x.DBB61.4	"Axis/spindle stationary"

Since the coupling is processed in the position controller cycle, the travel command signals are not output:

DB3x.DBB64.7 "Travel command +/-"

All other signals show the current state of the axis.

Actual value display

In the automatic basic display, the motion of the slave axis is not displayed for an active coupling and the actual value is frozen. If the coupling is deactivated, the actual-value display jumps to the current actual position. At the next NC start, the slave axis is synchronized with the NC so that the slave axis can be traversed from this position.

The slave axis motion is always displayed, also at an active coupling, in the "Dialog" menu, "Service display" softkey.

Traversing the slave axis

A slave axis in an active coupling must not be traversed by the part program, by the PLC or manually in JOG mode. If a coupled slave axis is traversed, this produces the reset alarm "75170 Axis AX1 overlaid motion not permissible".

Reference point approach

The status of the coupling determines the method of reference point approach. This applies to referencing in JOG Ref mode, and referencing from within the part program (G74).

If a master-slave coupling can be activated via a PLC signal (DB3x,DBB24.7), the master and slave axes are referenced individually in the "not active" coupling state.

If a master-slave coupling is always active after POWER ON (MD63590 = 1), only the master axis is referenced. In this case, the slave axis is never referenced. Since the coupling is active, the slave axis follows when the master axis is referenced.

MD34110 $MA_REFP_CYCLE_NR of the slave axis must be set to -1 so the NC can start up without having to reference the slave axis.

Reaction to errors

In the event of error conditions for alarms with alarm reaction "Follow-up in master and/or slave", each axis is decelerated to 0 speed. The master-slave coupling is deactivated.

For avoiding mechanical tensions, the following machine data and drive machine data for master and slave axes must be set to the same values.

Machine data:

MD36620 $MA_SERVO_DISABLE_DELAY_TIME

MD36610 $MA_AX_ENERGY_STOP_TIME

Drive machine data:

MD1403 $MD_PULSE_SUPPRESSION_SPEED

MD1404 $MD_PULSE_SUPPRESSION_DELAY

This remains the responsibility of the user. The master-slave coupling does not become active again until both axes have returned to the "control active" state following a channel reset.

11.4 Constraints

NCU 572.2

The Master/Slave for Drives function can be utilized on NCU 572.2 hardware only on condition that is has been specifically enabled for the customer.

SINUMERIK 840Di

The compile cycles function of the SINUMERIK 840D are currently available only on request for the SINUMERIK 840Di.

Virtual axis

In connection with "master-slave", the use of the function "Virtual axis" is not permissible in the following machine data:

MD30132 $MA_IS_VIRTUAL_AX (axis is virtual axis)

11.4.1 Speed/torque coupling (SW 6 and higher)

Option

The speed/torque coupling function is an option and not available in every control variant.

The master-slave function requires the master and slave axes to be operated on the same NCU.

Note

The master-slave function requires the master and slave axes to be operated on the same NCU.

- A coupled slave axis cannot be rotated around the **axis container**.
- **Closing and separating** the master-slave coupling is carried out when the axis has stopped.
- **Traversing a slave axis** with the coupling closed is possible only via the master axis.
- **Axis replacement** is not performed for coupled slave axes.
- When the coupling is closed via the slave axis, the master axis is **braked automatically** (if defined in the same channel). This produces an asymmetric response on closure and separation of the coupling. In contrast to closing, there is no automatic braking on separation.
- **Block search with calculation (SERUPRO)** takes into account the positional changes of coupled slave axes after a block search only if a system ASUB (asynchronous subroutine) "PROGEVENT.SPF" has been generated. This can be used to subsequently adjust the coupling state and associated axis positions so as to update changes to the coupling state.

Differences compared to previous solution (up to SW 5.x)

- If a traversing movement is programmed for a slave axis that has already been coupled, the alarm "14092 Channel %1 Block %2 Axis %3 has the wrong type" appears.
- The setpoint position of the coupled slave axis corresponds to the current actual position.
- On request, the coupling is made or released independent of the channel status the next time the axis stops. This allows the coupling status to be changed even during part program processing.
- PLC interface signal DB3x.DBX61.5 "Position controller active" must no longer be interpreted in the braking control logic of the slave axes. This is no longer set for an active coupling. Instead, the interface signal "Master-slave coupling status active" should be used.
- If a master axis is simultaneously configured as a slave axis, the alarm "26031 Axis %1 Configuration error master-slave" appears. So cascading is not possible.
- If a coupling is requested and closed, the control activation signals are derived directly from the master axis.

11.4.2 Speed/torque coupling (up to SW 5.x)

11.4.2.1 Axis replacement

An axis replacement can be performend considering the following restrictions:

For activating or deactivating a coupling, the channels of the master and slave axes must be in "RESET" state. The states of the default channels of the axes are scanned prior to activation/deactivation. At the time of activation and deactivation, the axes must be located in the default channel assigned by MD30550. A change of axis is possible in between times, even if the coupling is active.

11.4.2.2 Rotary axis modulo, spindles

Modulo rotary axes

Master and slave axes can also be rotary axes. Please note the following:

For the slave axis, the actual value in the "Diagnosis" menu, "Service" softkey, exceeds 360 degrees, even if modulo operation is set for the axis in the following machine data:

MD30310 $MA_ROT_IS_MODULO

The automatic basic display and the service display do not show the actual value modulo 360 until the coupling is deactivated.

Spindles

If a master-slave coupling is activated with spindles, the slave axis is in speed control mode. In this case too, the actual value of the slave axis exceeds 360 degrees in the service display. No modulo calculation is active. However, the value shown in the automatic basic display is modulo 360 degrees.

11.4.2.3 Simultaneous operation of master/slave coupling and clearance control function

The "speed/torque coupling (master-slave)" and "clearance control" functions can be operated simultaneously with the following restriction:

An axis that is traversed by the clearance control must be neither a master nor a slave axis in the master-slave function.

11.4.2.4 Displaying torque values and controller output in NCK GUD

To support installation, the current axial torque values in [Nm] and the speed setpoints in [mm/min] or [rpm] of the P controller and the I controller of a torque controller can be displayed on the operator panel front in the "Parameter - user data" area.

For this purpose, the appropriate GUDs must be set up.

Procedure

Create SGUD:

- "Services" menu
- If the "Definitions" directory does not appear, select definitions using the "Data selection" softkey
- Open the Definitions directory
- "Manage data" softkey
- "New" softkey
- Create file

  Name: SGUD

  File type: Select global data/system
- OK
- The file opens. Enter the following lines:

```
DEF NCK REAL MASTER_SLAVE_TORQUE[number of active axes]
DEF NCK REAL TORQUE_CTRL_P[number of active axes]
DEF NCK REAL TORQUE_CTRL_I[number of active axes]
M30
```

- Close file and load

File: Create "Initial.ini":

- Menu: Services > "Manage data" softkey > "New" softkey
- Create new directory type "NC data backup" and in this create the file: "Initial.ini"

  Name: Initial

  Type: Initialization program
- OK
- The file is opened. Enter the following line:

```
M17
```

- Close file and load

The following axis data are then displayed:

MASTER_SLAVE_TORQUE[0]	Current torque in [Nm]
TORQUE_CTRL_P[0]	P component of an active torque control in [mm/min] or [rpm]
TORQUE_CTRL_I[0]	I component of an active torque control in [mm/min] or [rpm]

References

/PGA/ Programming Manual, Job Planning; Chapter "File and Program Administration"

11.4.2.5 Servo Trace

To support installation, the current torque values and the torque controller output can be displayed on the MMC in the Servo Trace function.

⚠ CAUTION

The existing Servo Trace function has been expanded for master and slave. The operation of the "Servo Trace" is described in the Installation Guide.

In order to be able to select the data of a master-slave coupling in the menu in the servo trace, the following files must be created on the MMC. You can use the DOS shell and the editor "edit" to do this.

File: \ oem \ ibsvt.ini

Contents:

[OemSignalList]

Item0 = Type := Title, Signalindex := – 1, Unit := No

Item5 = Type := Signal, Signalindex := 200, Unit := Torque|Force

Item10 = Type := Signal, Signalindex := 201, Unit := Torque|Force

Item15 = Type := Signal, Signalindex := 202, Unit := NcSpeed

File: \ oem \ language \ Ibsvt_gr.ini

Contents:

[OemComboBoxItemNames]

Item0 = "MASTER-SLAVE"

Item5 = "Master torque"

Item10 = "Master torque"

Item15 = "Controller output"

This file is language-specific and must be created with the corresponding language code (uk for English) for all available languages.

Following the next MMC POWER ON, you can use the selection menu to select the following signals in the Servo Trace menu.

- Master torque
- Slave torque
- Controller output

⚠ CAUTION
To achieve a higher signal resolution, the data is displayed in the following units: Torques in [Nm] Controller output in [internal increment/s]

No further machine data need be set to activate a measurement.

Up to 4 signals can be recorded in one measurement. The associated machine axis is selected in the axis selection for the torque values; for the controller output, the machine axis of the slave axis of this control is selected.

Example:

Master axis: X1

Slave axis: Y1

The following is to be recorded:

Master torque, axis selection, X1.

Slave torque, axis selection, Y1.

Controller output, axis selection, Y1.

With 4 active couplings, it is possible to record all 4 torque values of the master axes or 4 controller outputs.

With automatic scaling, the measured curves of a display are always overlaid. In order to compare the values of the curves properly, the scaling must be set the same for both curves (see graphic 2 in the following figure). The scaling can still be modified in the Scale menu after the measurement.

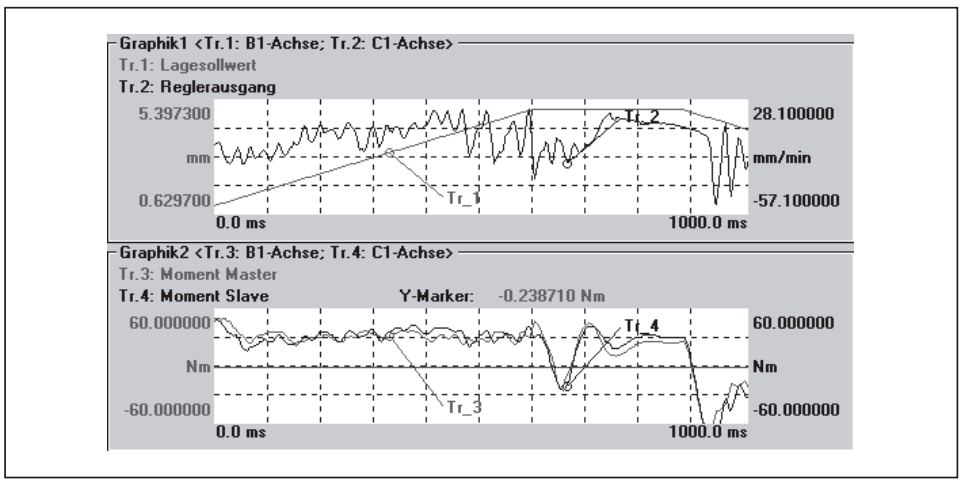


Figure 11-16 Example of a measurement with 4 measured values

11.4.2.6 Controller data to analog output

Controller data output to an analog output can be activated using the following machine data:

MD63595 $MA_TRACE_MODE Bit0 (Trace setting)

Then the following data is output to analog output at the terminal block:

- Torque of the master axis at analog converter 1
- Torque of the slave axis at analog converter 2
- Torque control output at analog converter 3

The torques, relative to the rated torque, are normalized to 8 volts, the torque controller output, relative to the maximum slave axis speed, is normalized to 8 volts and shown in mm/min.

The following machine data is used to determine the terminal block slots that are occupied by analog converters:

MD10364 $MN_HW_ASSIGN_ANA_FASTOUT (Hardware assignment of external analog NCK outputs).

11.4.2.7 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

11.5 Examples

11.5.1 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

11.5.2 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.
• Move master-slave grouping	The master axis is moved.

11.5.3 Close/separate coupling via part program

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.
- Activation of a master-slave coupling via the machine data:

  MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE= 0 (permanent master-slave coupling)
- The coupling is open.

Parts program

```
N10 G0 AX1=0 AX2=0      ; Approach coupling position Each axis moves to
                        the coupling position.
N20 MASLON (AX2)        ; Close coupling mechanically. Both axes are mechanically
                        ; coupled to one another.
N30 AX1=100;            ; Move master-slave grouping.
                        ; The master axis is moved.
                        ; Coupled via the speed setpoint, the slave
                        ; axis follows the master axis.
N40 MASLOF (AX2)        ; Disconnect coupling The axes are mechanically
                        ; disconnected from one another.
N50 AX1=200 AX2=200;    ; Move master and slave axes.
                        ; The maxster axis is moved,
                        ; separately from the slave axis.
N60 M30
```

11.5.4 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.

11.6 Data lists

11.6.1 Machine data

11.6.1.1 Axis/spindle-specific machine data

SW 6 and higher

Number	Identifier: $MA_	Description
37250	MS_ASSIGN_MASTER_SPEED_CMD	Machine axis number of master axis for speed setpoint coupling
37252	MS_ASSIGN_MASTER_TORQUE_CTR	Master axis number for torque control
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

Up to SW 5.x

Number	Identifier: $MA_	Description
34110	REFP_CYC_NR	NC Start is possible without referencing of axis R1
36620	SERVO_DISABLE_DELAY_TIME	Cutout delay servo enable A2
36610	AX_ENERGY_STOP_TIME	Duration of braking slope A3
63550	MS_ASSIGN_MASTER_SPEED_CMD	Master axis for speed setpoint coupling
63555	MS_ASSIGN_MASTER_TORQUE_CTRL	Master axis for torque control
63560	MS_TORQUE_CTRL_P_GAIN	P gain of the torque control
63565	MS_TORQUE_CTRL_I_TIME	I component of the torque control
63570	MS_TORQUE_CTRL_MODE	Connection of the torque control output
63575	MS_TORQUE_WEIGHT_SLAVE	Weighting of the torque values
63580	MS_TENSION_TORQUE	Tension torque

Number	Identifier: $MA_	Description
63585	MS_TENSION_TORQ_FILTER_TIME	Time constant for PT1 filter tension torque
63590	MS_COUPLING_ALWAYS_ACTIVE	Master-slave coupling active after power ON
63595	MS_TRACE_MODE	Trace setting
63600	MS_MS_MAX_CTRL_VELO	Control output limit

11.6.2 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[slave axis identifier]	This variable registers a change in the coupling state during the SERUPRO block search.
$P_SEARCH_MASLD[slave axis identifier]	This variable indicates the positional offset between the slave and master axes at the instant the coupling was closed.
$AA_MASL_STAT[slave axis identifier]	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.

11.6.3 Signals

11.6.3.1 Signals to axis/spindle

DB number	Byte.bit	Description
31, ...	24.4	Activate torque compensatory controller
31, ...	24.7	Activate master-slave coupling

11.6.3.2 Signals from axis/spindle

DB number	Byte.bit	Description
31, ...	96.2	Differential speed "Fine"
31, ...	96.3	Differential speed "Coarse"
31, ...	96.4	State of torque compensatory controller
31, ...	96.7	State of master-slave coupling

Handling Transformation Package (TE4) 12

12.1 Brief description

Functionality

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).

Structure of Chapter 2

Chapter 2 (Detailed Description) deals with the following topics:

- Section 2.1 describes the environment for kinematic transformation.
- Section 2.2. provides an explanation of basic terms.
- Section 2.3 explains the machine data required to configure transformations.
- Section 2.4 uses configuring examples to illustrate the most commonly used 2-axis to 5-axis kinematics that can be configured with the handling transformation package.
- Sections 2.5 to 2.9 deal with the subject of 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
WS	Workpiece coordinate system
WZ	Tool coordinate system
x_3, y_3, z_3	Coordinates of the first wrist axis

12.2 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. The current package offers kinematics, which include between 2 and 5 axes in the transformation, corresponding to up to five spatial degrees of freedom. In this case, a maximum of 3 degrees are available for translation and 2 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.

12.3 Definition of terms

12.3.1 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.

12.3.2 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 only effects an offset between the coordinate system and the reference system, while rotation actually rotates the coordinate system in relation to the reference. Coordinates X, Y and Z are used to describe the translation. They are defined so as to result in a legal coordinate system.

Translation

The 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

The rotation is described by the RPY angles A, B and C (RPY stands for Roll Pitch 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. In this respect, it must be noted that A and C are defined at intervals [-180; +180] and B at intervals [-90; +90].

The definitions of the RPY angles are as follows:

- Angle A: 1st rotation about the Z axis of the initial system
- B angle: 2nd Rotation through the rotated Y axis
- C angle: 3rd 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]

An example of the rotation of the RPY angles is specified in Fig. "Example of rotation through RPY angles".

In this example, the initial coordinate system X1, Y1, Z1 is first rotated through angle A about axis Z1, then through angle B about axis Y2 and finally through angle C about axis X3.

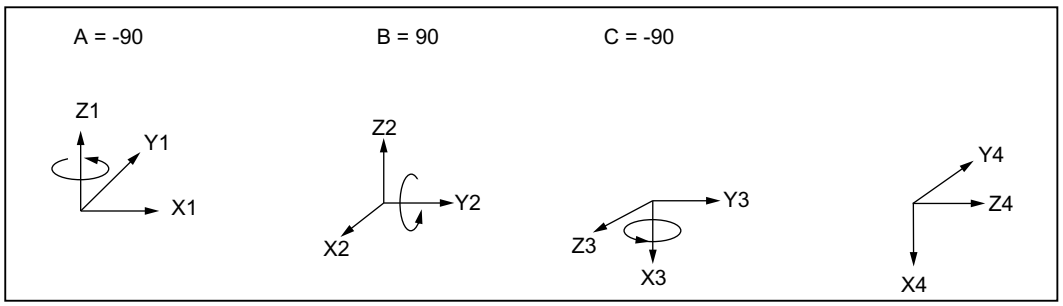


Figure 12-1 Example of rotation through RPY angles

12.3.3 Definition of a joint

Meaning

The term joint describes either a translational or a rotary axis.

The basic axis identifiers result from the arrangement and sequence of the individual joints. These are described by identifying letters (S, C, R, N), which are explained below.

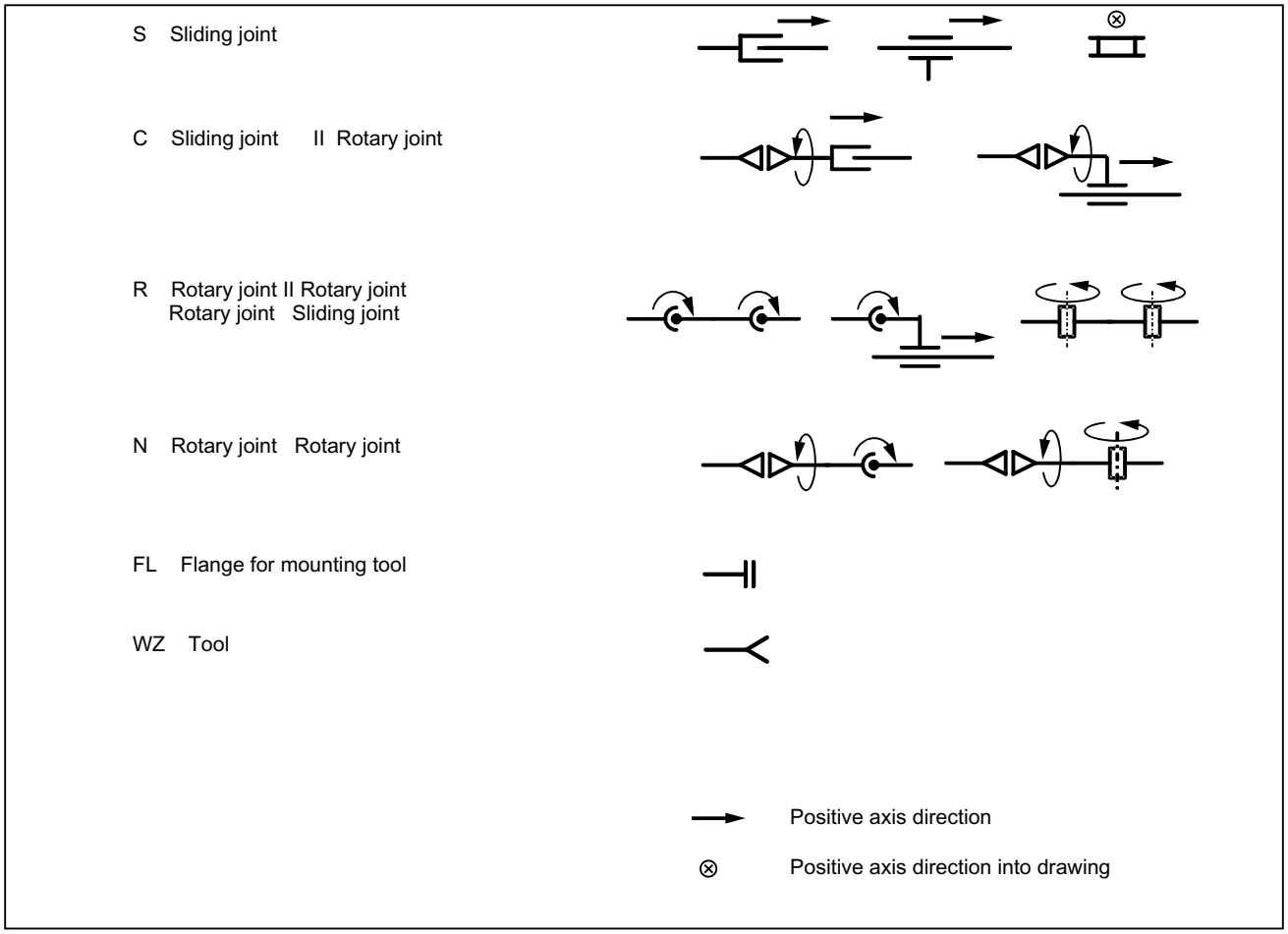


Figure 12-2 Joint identifying letters

12.4 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

12.4.1 General machine data

MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

The value 4099 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, i.e. the first n channel axes must be assigned to the n transformation axes in ascending sequence:

- MD24110 $MC_TRAFO_AXES_IN_1[0] = 1
- MD24110 $MC_TRAFO_AXES_IN_1[1] = 2
- MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
- MD24110 $MC_TRAFO_AXES_IN_1[3] = 4
- MD24110 $MC_TRAFO_AXES_IN_1[4] = 5
- MD24110 $MC_TRAFO_AXES_IN_1[5] = 6

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.

- 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

12.4.2 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. For this purpose, frames (see Subsection "Definition of positions and orientations using frames") are used to describe the geometry. While the control is powering up, the configuration machine data are checked and alarms generated when necessary.

All axes in the mode group are made to follow, the alarms can only be reset by a `POWER ON` operation.

As shown in Figure "Closed kinematic loop illustrated by the example of a robot", the kinematic transformation effects a conversion of the tool operating point (tool coordinate system: XWZ, YWZ, ZWZ), that is specified in relation to the basic coordinate system (BCS = robot coordinate system: XRO, YRO, ZRO), 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: XWS, YWS, ZWS). The programmable frames make it possible to create an offset between the workpiece coordinate system WCS and the basic coordinate system BCS.

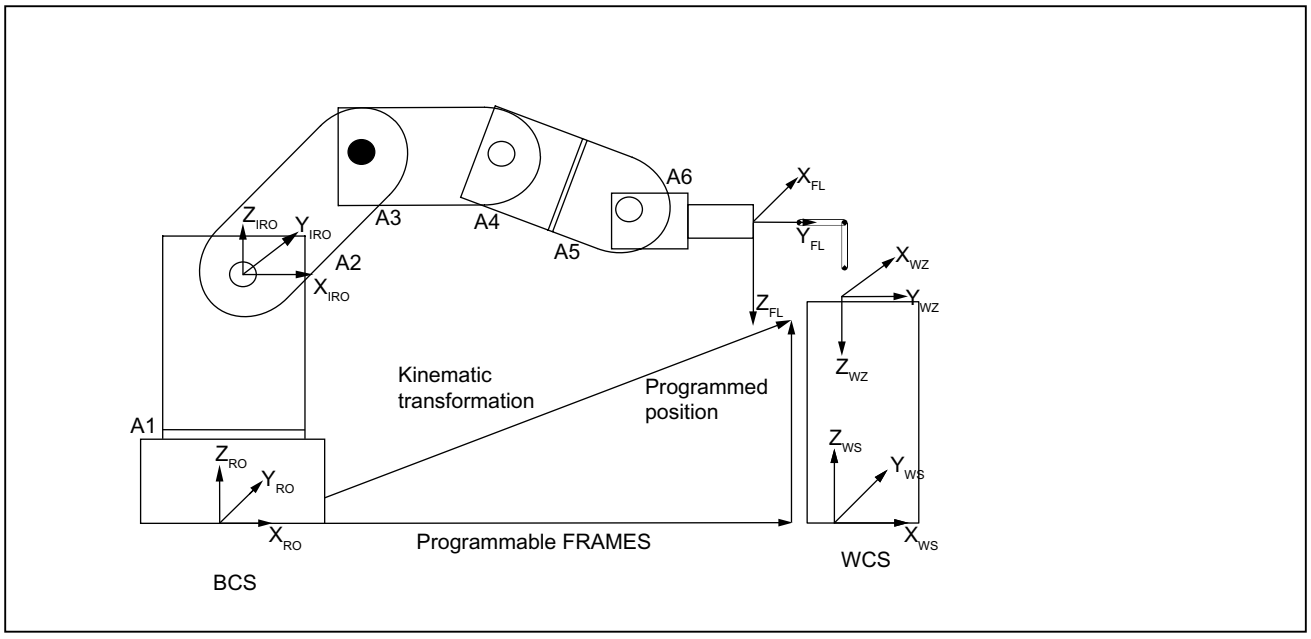


Figure 12-3 Closed kinematic loop illustrated by the example of a robot

Note

For more detailed information about coordinate systems, please see:

References:
/PG/ Programming Guide, Fundamentals

The following machine data are available for configuring kinematic transformations:

MD62612, MD62613

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)

MD62603

The type of basic axis arrangement is specified in machine data:

MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)

The basic axes are generally the first 3 axes to be included in the transformation.

MD62607

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.

MD62606

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)

MD62608, MD62609

Frame T_X3_P3 links the last coordinate system of the basic axes with the first hand coordinate system.

MD62608 $MC_TRAFO6_TX3P3_POS (attachment of hand (position component], n = 0...2)

MD62609 $MC_TRAFO6_TX3P3_RPY (attachment of hand (rotation component), n = 0...2)

MD62604, MD62616

These parameters describe the hand geometry.

MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

MD62616 $MC_TRAFO6_DHPAR4_5.. (parameter ALPHA for configuring the hand, n = 0...1)

MD62604

The hand type is specified in machine data:

MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

The term wrist axes generally refers to axes four to six.

MD62610, MD62611

Frame T_FL_WP links the last hand 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.

Basic axes included in every transformation

MD62603

The first 3 axes included in the transformation are generally referred to as the "basic axes". They must always be mutually parallel or perpendicular. Each of the following basic axis arrangements has its own special identifier (see Subsection "Definition of a joint"). The basic axis identifier is specified via machine data:

MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)

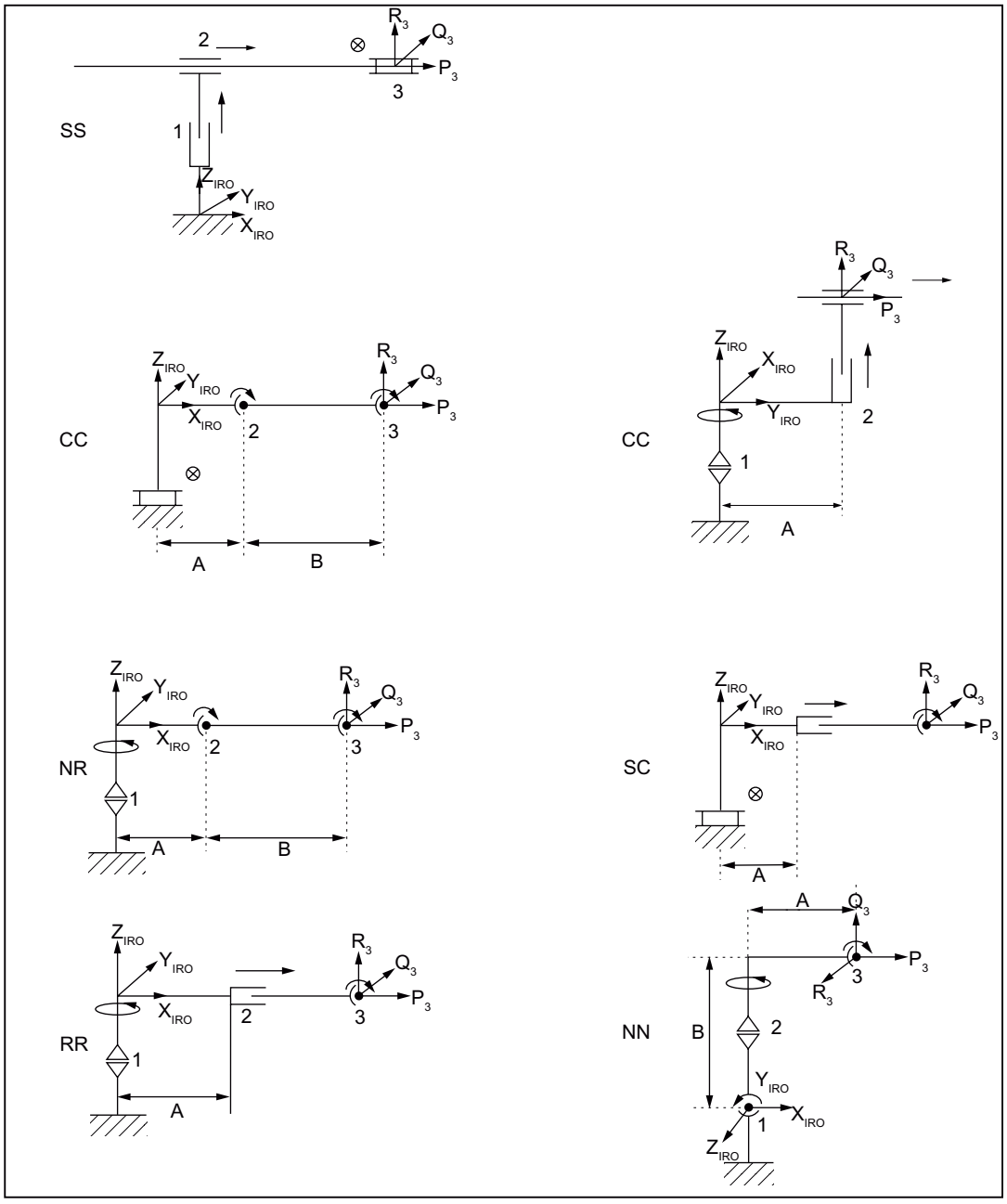


Figure 12-4 Overview of basic axis configuration

The handling transformation package contains the following basic axis kinematics:

- SS: Gantry (3 linear axes, rectangular)
- CC: SCARA (1 linear axis, 2 rotary axes (in parallel))
- SC: SCARA (2 linear axes, 1 rotary axis (swivel axis))
- CS: SCARA (2 linear axes, 1 rotary axis (axis of rotation))
- NR: Articulated arm (3 rotary axes (2 axes in parallel))
- NN: Articulated arm (3 rotary axes)
- RR: Articulated arm (1 linear axis, 2 rotary axes (perpendicular))

Wrist axes included in every transformation

MD62604

The fourth axis and all further axes are generally referred to as "wrist axes". The handling transformation package can only identify hands with rotary axes. The wrist axis identifier for three-axis hands is entered in machine data:

MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

In the case of hands with fewer than three axes, the identifier for a beveled hand with elbow or a central hand is entered in machine data:

MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)

The current software supports only wrist axis types beveled hand with elbow or central hand.

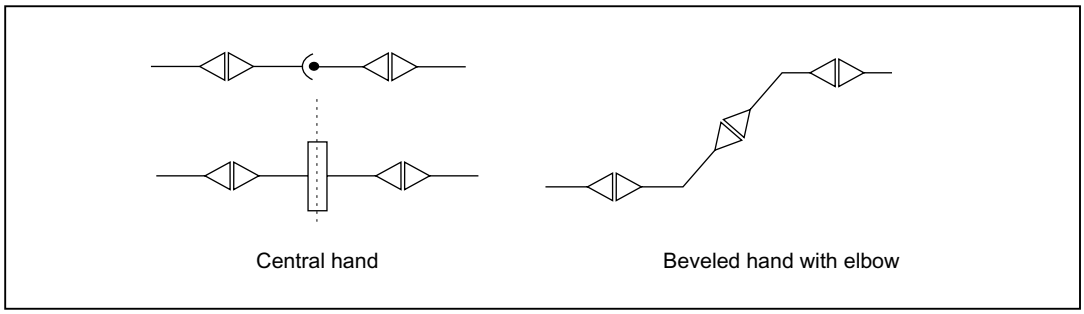


Figure 12-5 Overview of wrist axis configuration

Parameterization of wrist axes

MD62614 - MD62616

Hands are parameterized via parameter:

MD62614 $MC_TRAFO6_DHPAR4_5A (parameter A for configuring the hand, n = 0...1)

MD62615 $MC_TRAFO6_DHPAR4_5D (parameter D for configuring the hand, n = 0...1)

MD62616 $MC_TRAFO6_DHPAR4_5ALPHA (parameter ALPHA for configuring the hand, n = 0...1)

These data are special types of frame which describe the relative positions of the coordinate systems in the hand. For this purpose, the machine data corresponds to certain frame components (see Subsection "Definition of positions and orientations using frames"):

- MD62614 $MC_TRAFO6_DHPAR4_5A dem …_POS[0] (x component)
- MD62615 $MC_TRAFO6_DHPAR4_5D dem …_POS[2] (z component)
- MD62616 $MC_TRAFO6_DHPAR4_5ALPHA dem ..._RPY[2] (C angle)

  The other components of the frame are then zero.

Central hand (CH)

On a central hand, all wrist axes intersect at one point. All parameters must be set as shown in Table "Configuring data for a central hand".

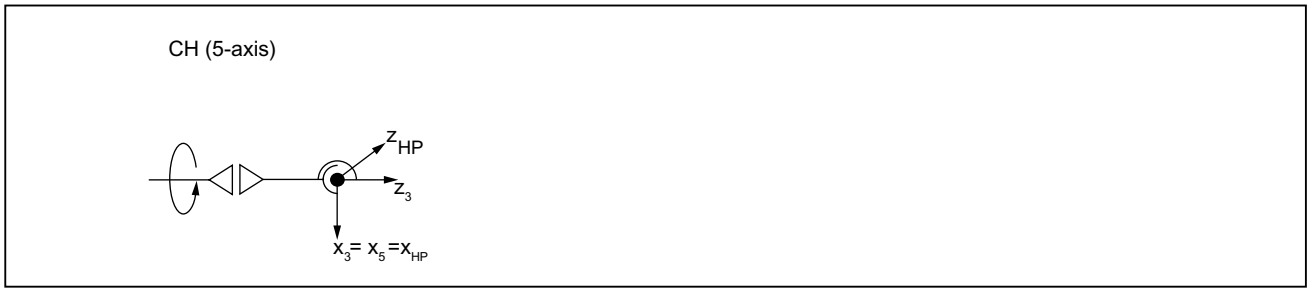


Figure 12-6 Central hand

Table 12-1 Configuring data for a central hand

Machine data	Value
MD62604 $MC_TRAFO6_WRIST_AXES	2
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, 90.0]

Beveled hand with elbow (BHE)

The beveled hand with elbow differs from the central hand 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 hand".

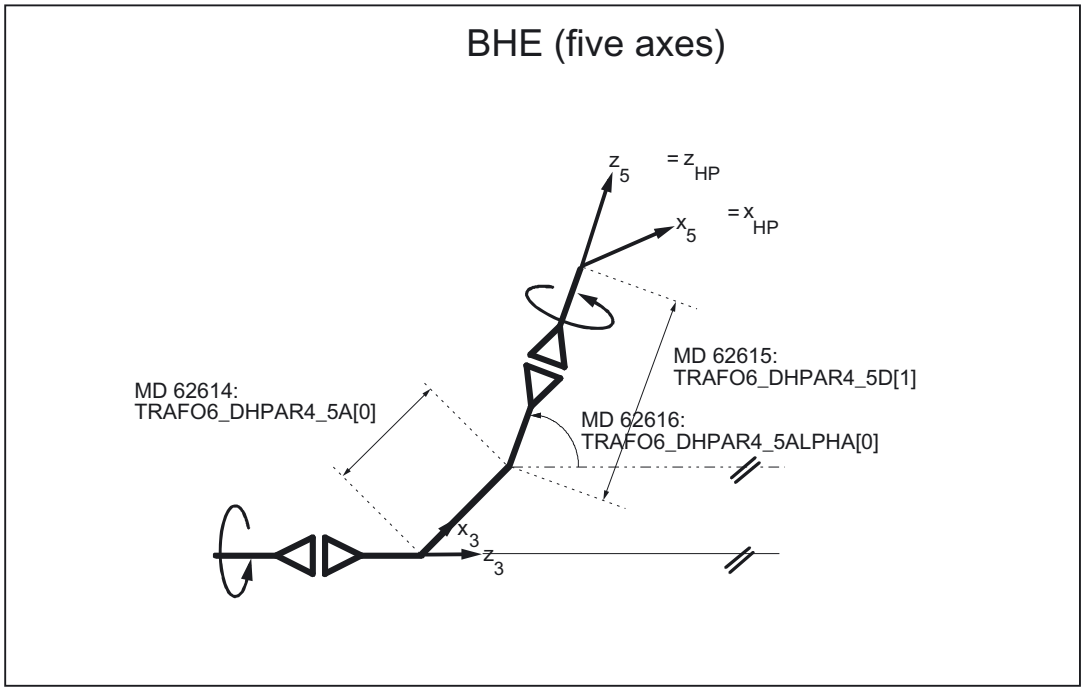


Figure 12-7 Beveled hand with elbow

Table 12-2 Configuring data for a beveled hand with elbow (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]

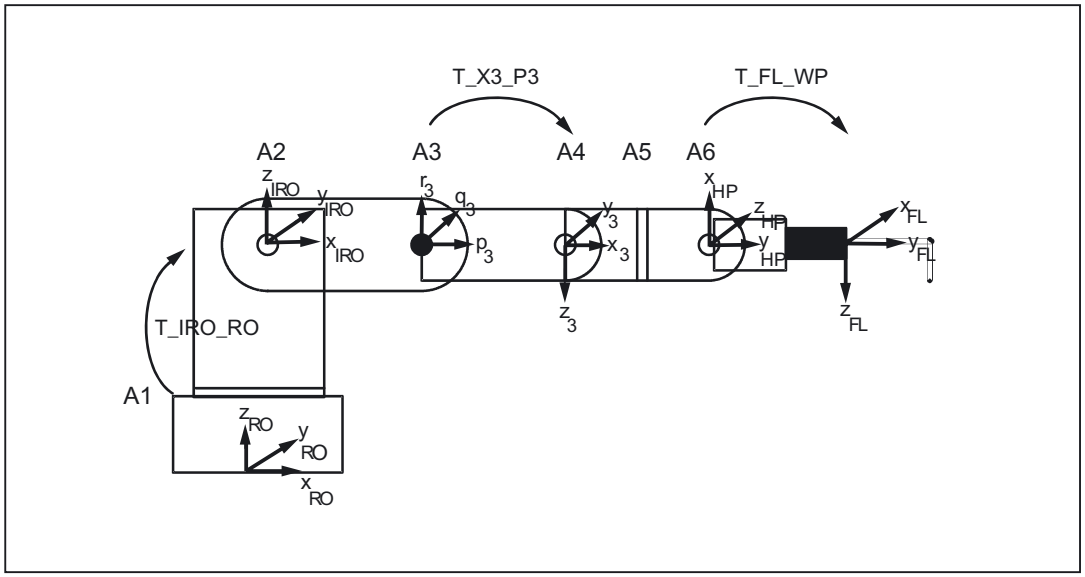


Figure 12-8 Link frames

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:
/PG/ Programming Guide, 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.

T_X3_P3

Frame T_X3_P3 describes the method used to attach the hand 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 hand 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 of the CC, CS or SC type, the transformation must either include a central hand (ZEH) or the 4th axis must be positioned in parallel to the last rotary basic axis.
  - If the basic axes are of the NR or RR type, the transformation must either include a central hand (ZEH) 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 hand.
- With 4-axis kinematics, the z3 axis must always be parallel/anti-parallel

or perpendicular to the last basic axis.

T_FL_WP

Frame T_FL_WP links the flange with the last internal coordinate system (hand-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)

With the current software, the machine data can be set to between 2 and 5 transformed axes.

Changing the axis sequence

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 5 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 12-3 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

This example involves two kinematics such as those illustrated in Fig. "Rearrangement of axes (example 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

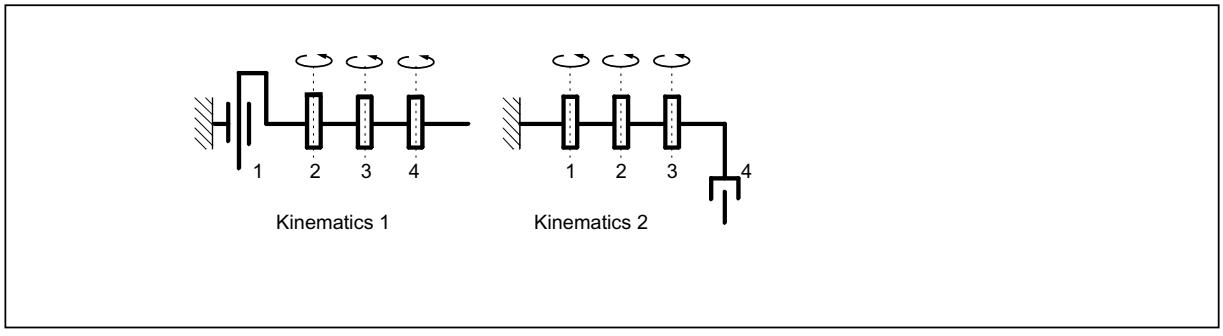


Figure 12-9 Rearrangement of axes

Example 2

This example involves a SCARA kinematic transformation as illustrated in Fig. "Rearrangement of axes (example 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.

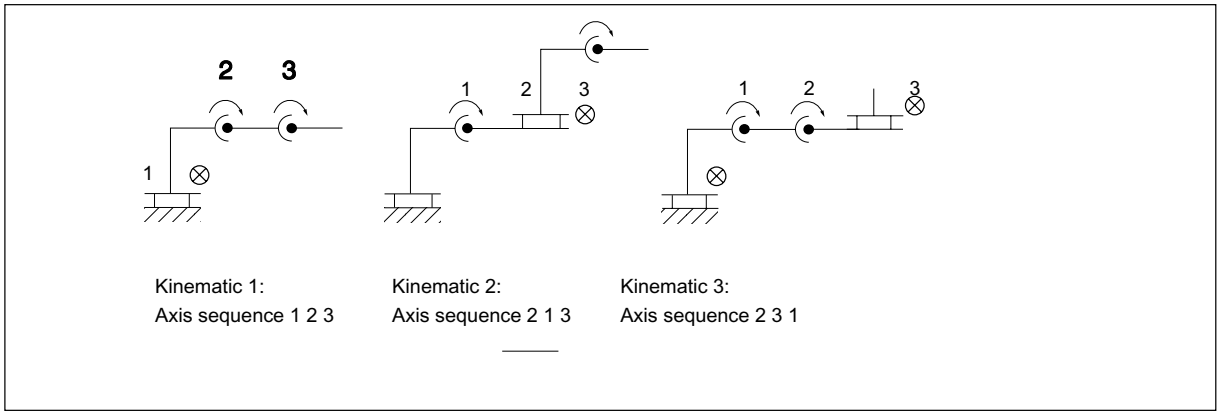


Figure 12-10 Rearrangement of axes 2

Changing the directions of axes

MD62618

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

MD62617

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

The example (Fig. "Matching mathematical and mechanical zero points") shows an articulated arm kinematic. The mathematical zero point for axis 2 is 90º. This value must be set for axis 2 in machine data:

MD62617 $MC_TRAFO6_MAMES[1] (offset between mathematical and mechanical zero points)

Axis 3 is counted relative to axis 2 and therefore has a value of -90º as a mathematical zero point.

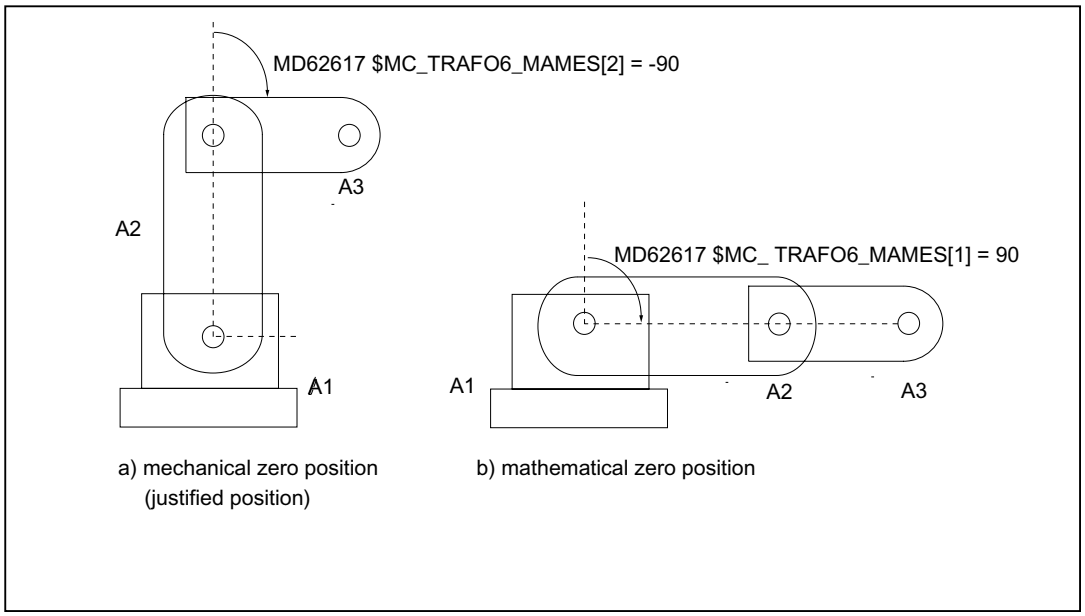


Figure 12-11 Matching mathematical and mechanical zero points

Axis types

MD62601

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
- Rotary axis

Velocities and acceleration rates

Separate velocities for the Cartesian movement components are introduced for axes that are traversed with `G00` and active transformation.

The velocity is preset via path feed F for axis traversal with `G01` or `G02`.

MD62629

The velocities for individual translational motion directions for axis traversal with G00 can be preset in 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

MD62630

The acceleration rates for individual translational motion directions for axis traversal with G00 can be preset in machine data:

MD62630 $MC_TRAFO6_ACCCP[i] (Cartesian acceleration rates [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

MD62631

The velocities for individual directions of orientation for axis traversal with G00 can be preset in machine data:

MD62631 $MC_TRAFO6_VELORI[i] (orientation angle velocities [no.]: 0...2)

Index i = 0 : A bracket

Index i = 1 : B bracket

Index i = 2 : C angle

MD62632

The acceleration rates for individual directions of orientation for axis traversal with G00 can be preset in machine data:

MD62632 $MC_TRAFO6_ACCORI[i] (orientation angle acceleration rates [no.]: 0...2)

Index i = 0 : A bracket

Index i = 1 : B bracket

Index i = 2 : C angle

12.5 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.

12.5.1 3-axis kinematics

3-axis kinematics normally possess 3 translational degrees of freedom, but do not have a degree of freedom for orientation. In other words, they only include 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 hand):

   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 basic axes are divided into 3 categories depending on how they are mutually positioned.

- CC types
- CS types
- SC types (cf. Fig. "Overview of basic axis configurations")

3-axis CC kinematics

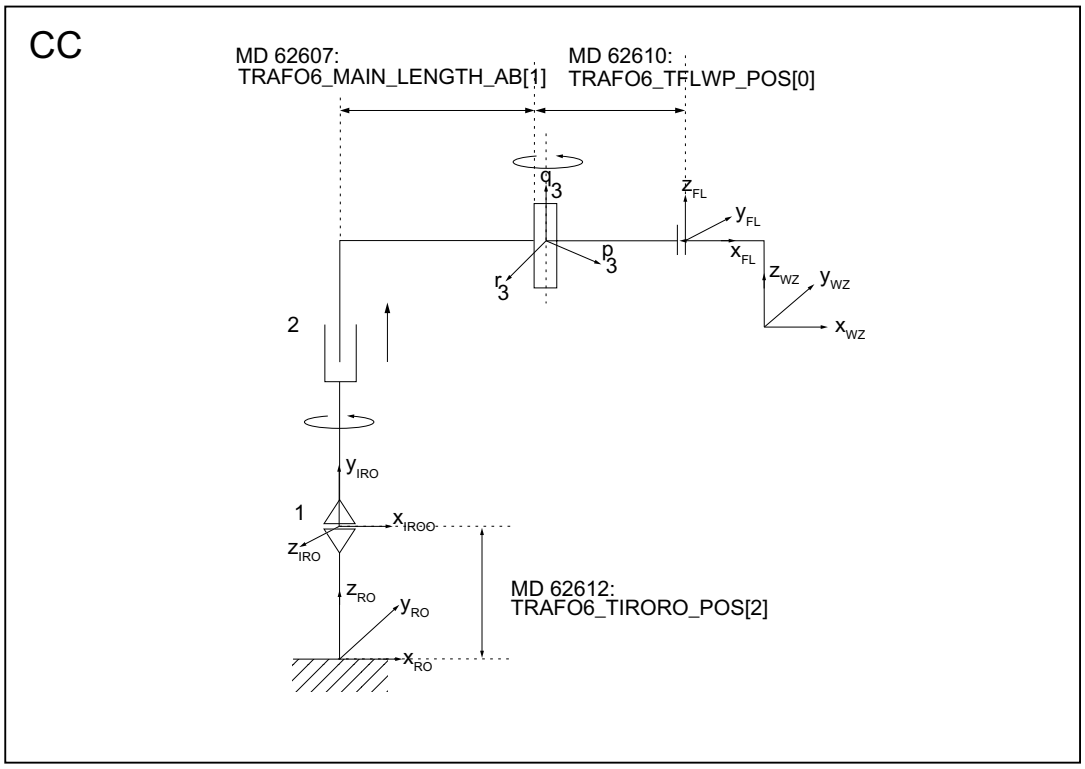


Figure 12-12 3-axis CC kinematics

Table 12-4 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

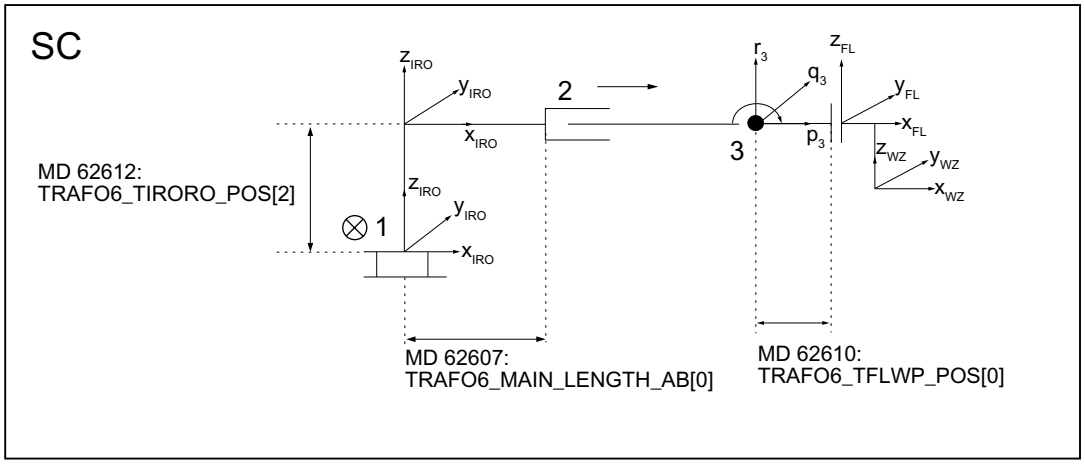


Figure 12-13 3-axis SC kinematics

Table 12-5 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]
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

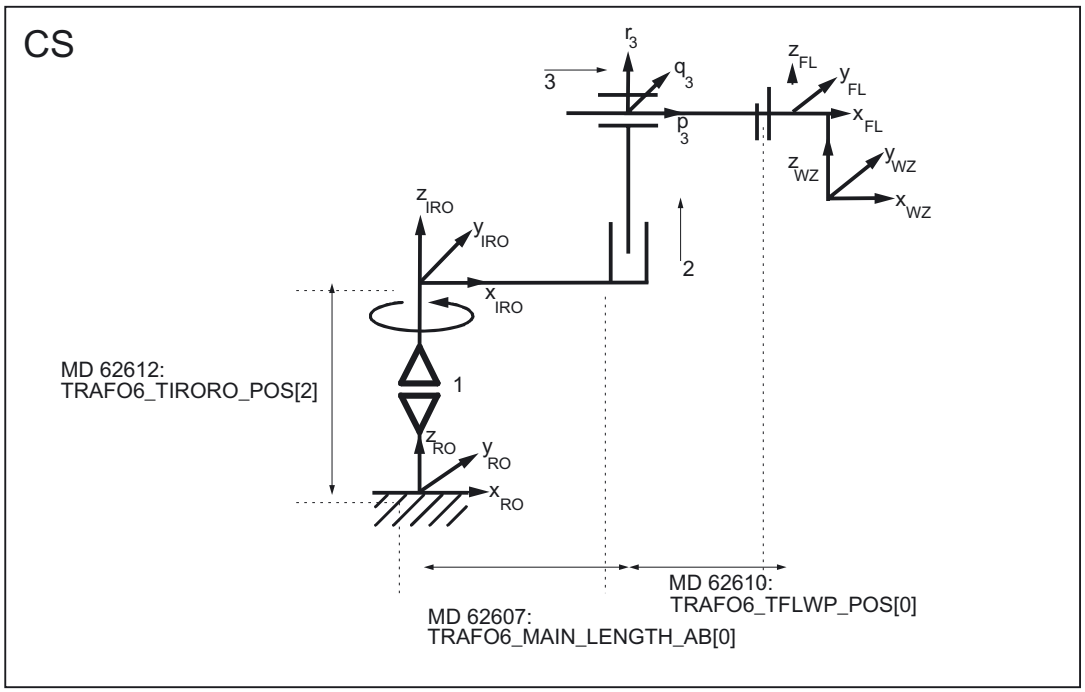


Figure 12-14 3-axis CS kinematic

Table 12-6 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]
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

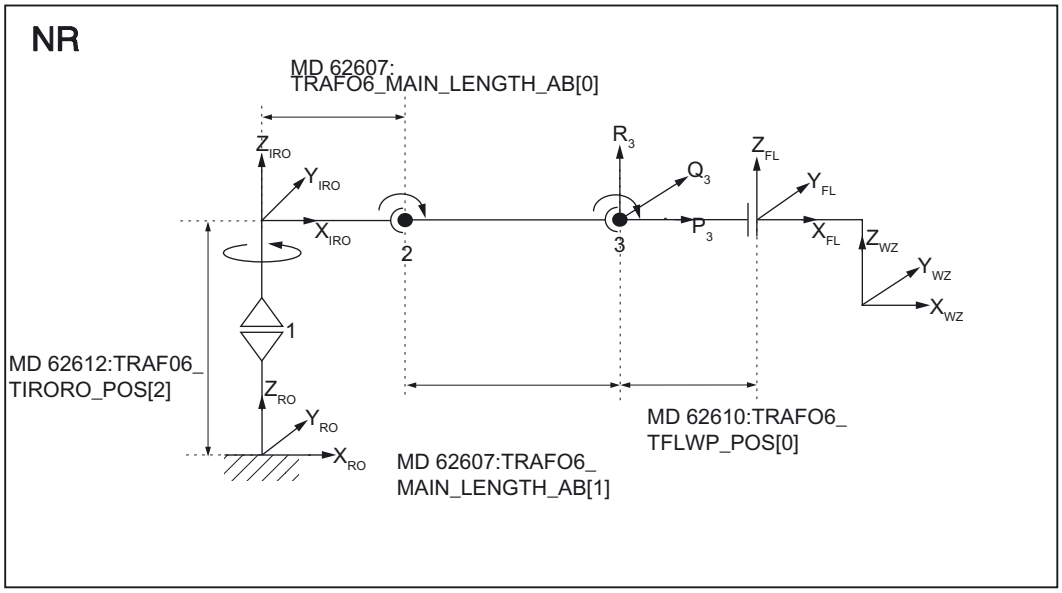


Figure 12-15 3-axis NR kinematics

Table 12-7 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]
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

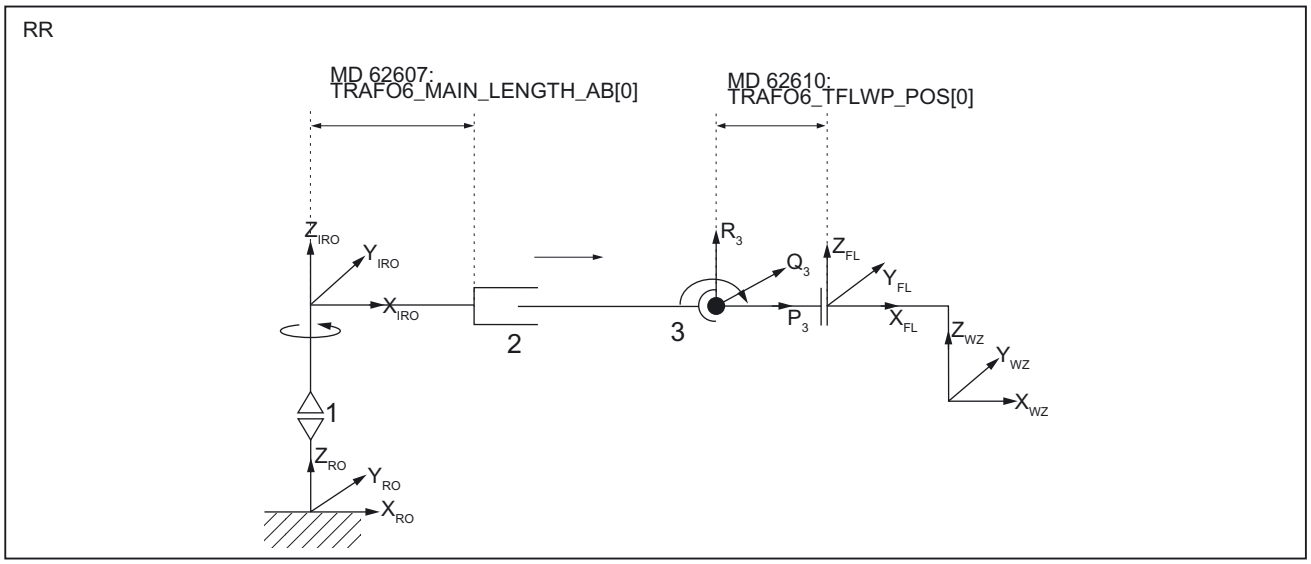


Figure 12-16 3-axis RR kinematics

Table 12-8 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

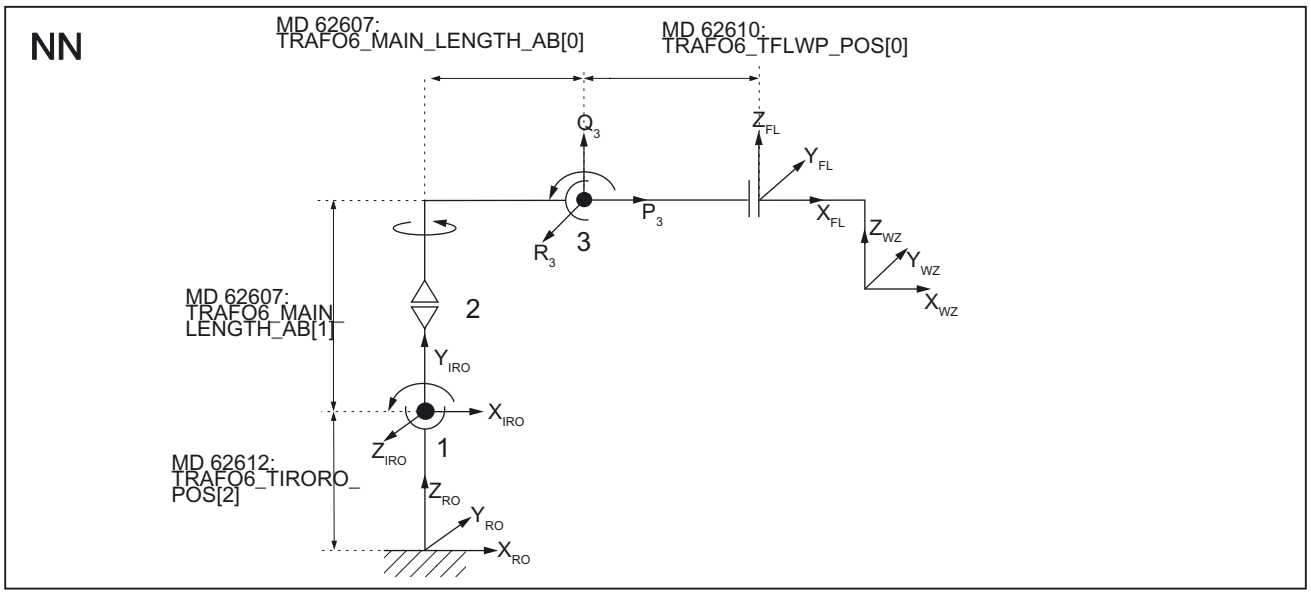


Figure 12-17 3-axis NN kinematics

Table 12-9 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]

12.5.2 4-axis kinematics

4-axis kinematics usually imply 3 translational degrees of freedom and one 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 hand):

   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 hand. 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 hand (position component))

    The rotation is entered in the machine data:

    MD62609 $MC_TRAFO6_TX3P3_RPY (attachment of hand (rotation component)) is entered.

13. Determine the flange coordinate system. For this purpose, the hand-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

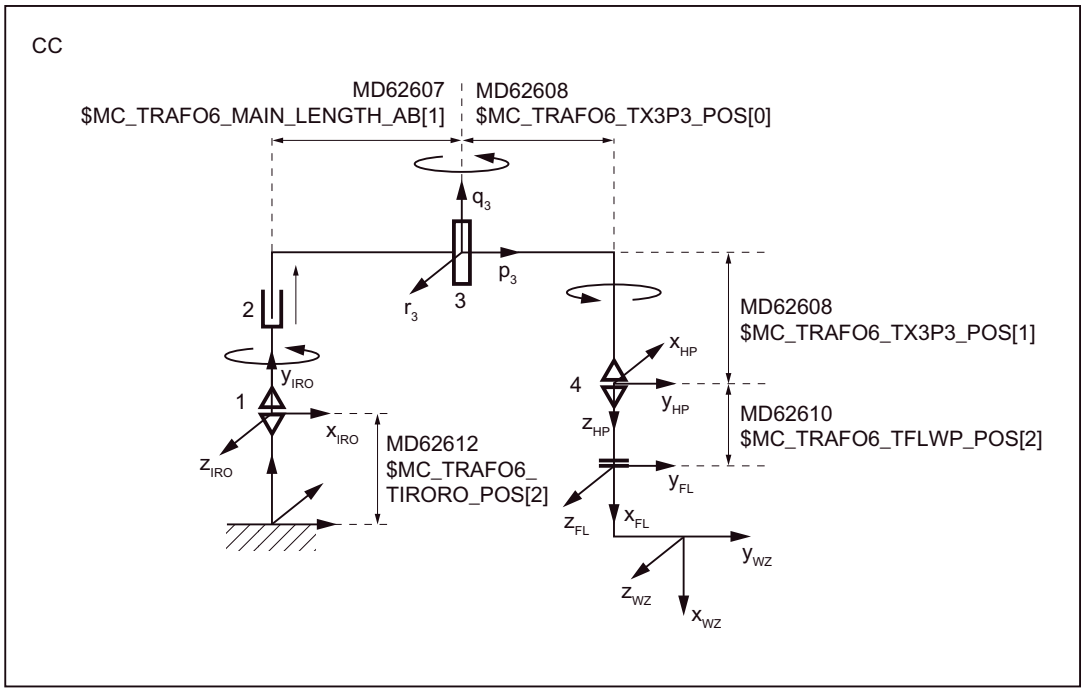


Figure 12-18 4-axis CC kinematics

Table 12-10 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

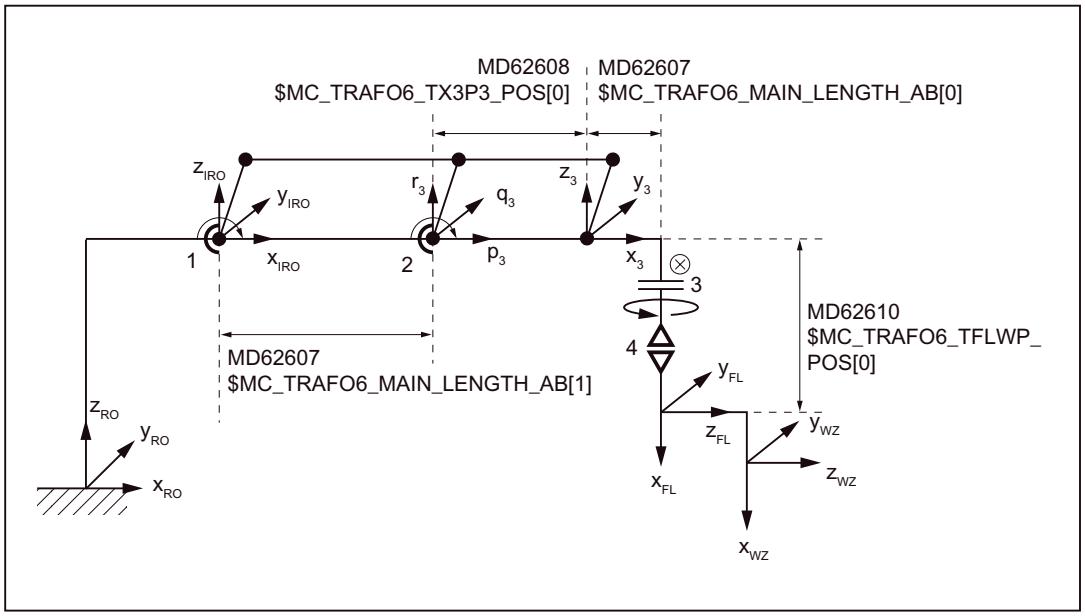


Figure 12-19 4-axis SC kinematics

Table 12-11 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

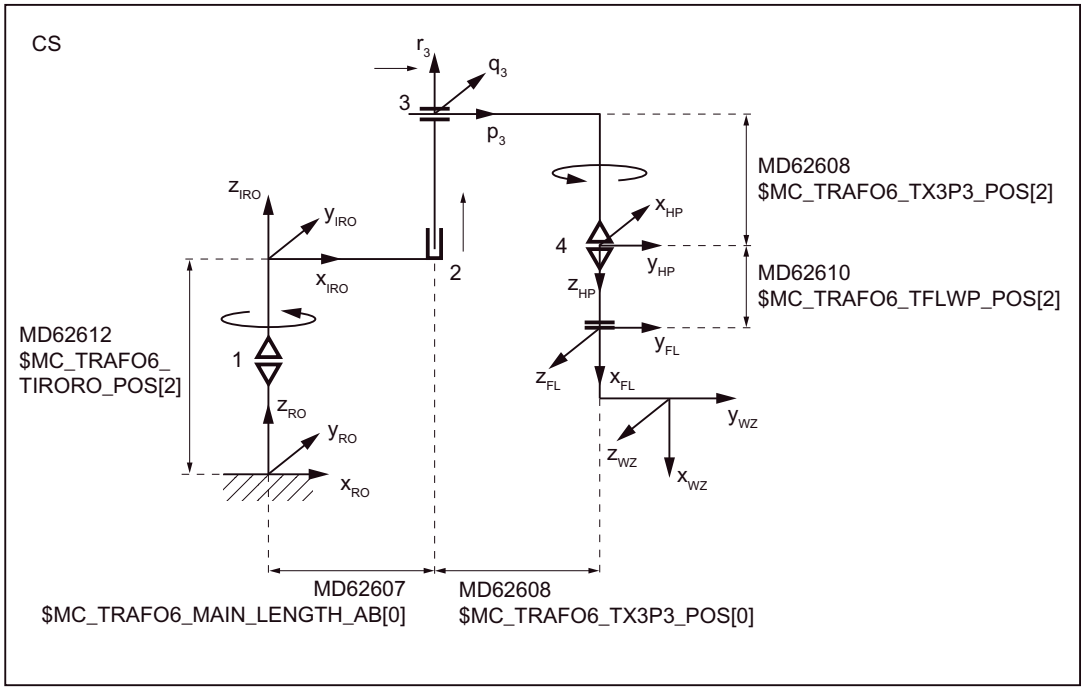


Figure 12-20 4-axis CS kinematic

Table 12-12 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	[200.0, 0.0, 0.0]
MD62611 $MC_TRAFO6_TFLWP_RPY	[0.0, -90.0, 0.0]

Articulated-arm kinematics

4-axis NR kinematics

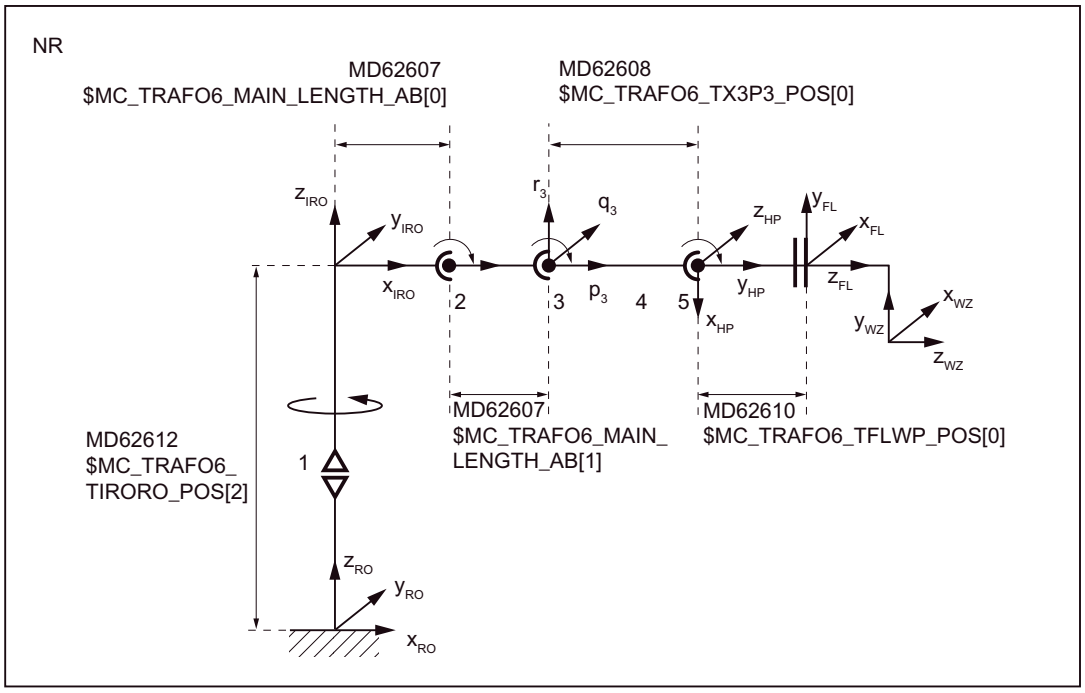


Figure 12-21 4-axis NR kinematics

Table 12-13 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]

12.5.3 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:
   - 4th axis parallel to the 3rd axis: 2-dimensional tool is possible [X, 0.0, Z]
   - 4th 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:
   - 4th 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

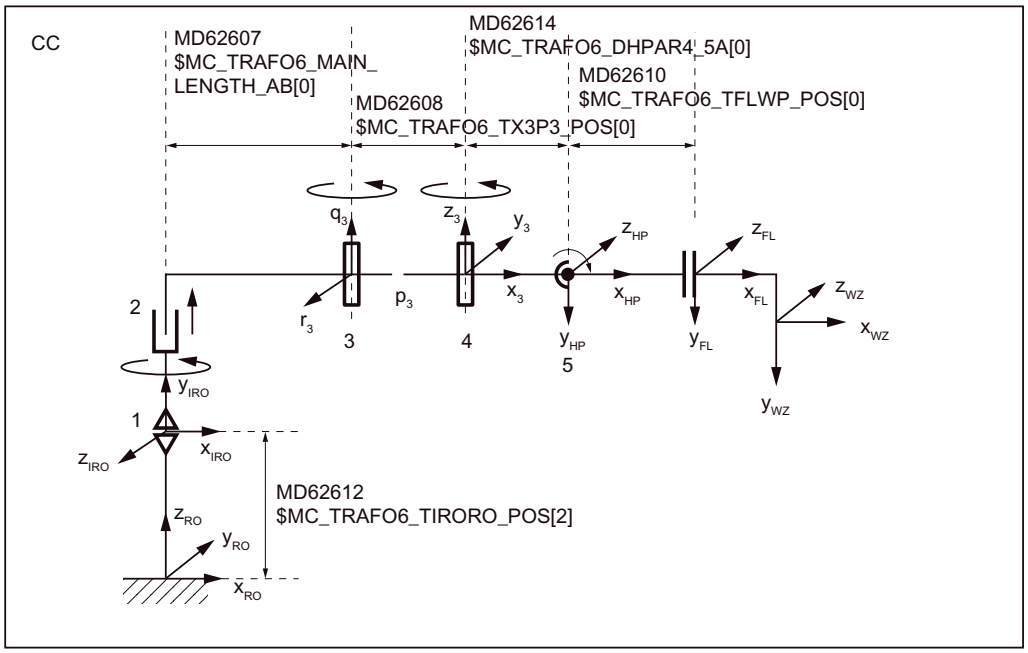


Figure 12-22 5-axis CC kinematics

Table 12-14 Configuration data for 5-axis CC kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	1
MD62605 $MC_TRAFO6_NUM_AXES	5
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

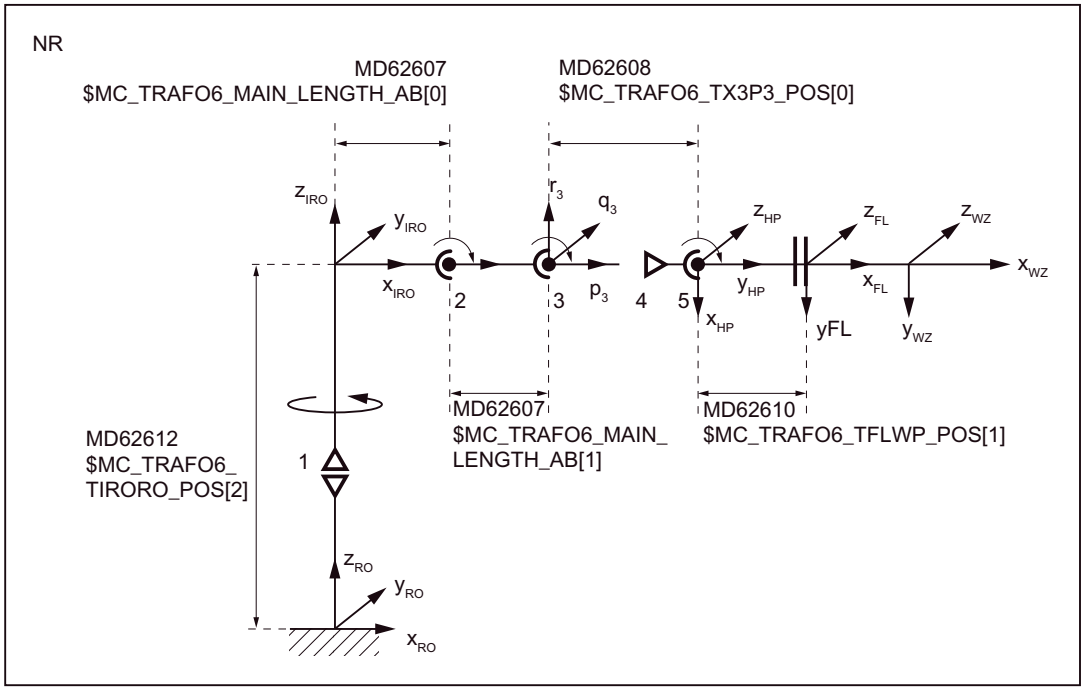


Figure 12-23 5-axis NR kinematics

Table 12-15 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]

12.5.4 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.

12.5.5 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)

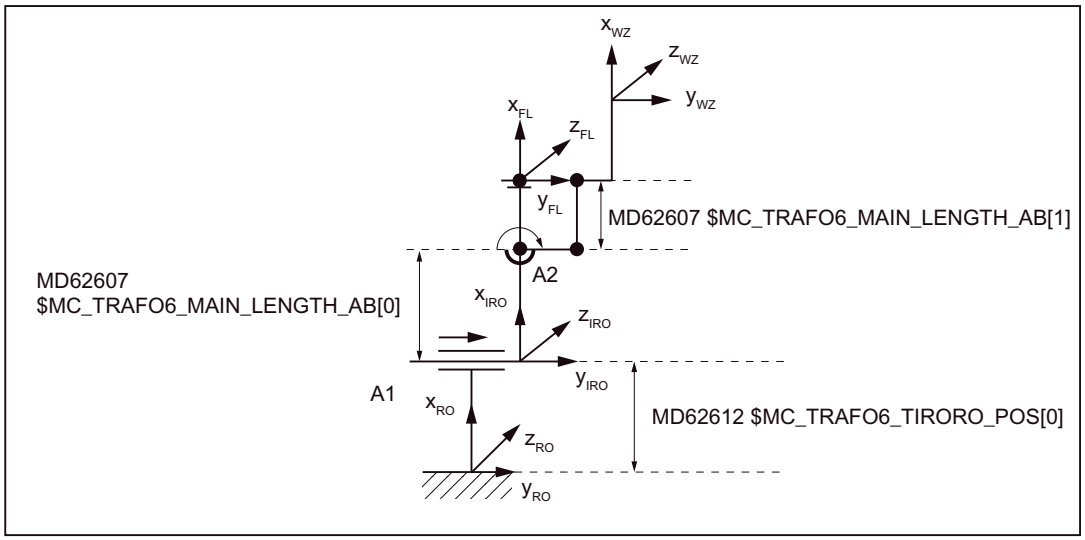


Figure 12-24 Special 2-axis SC kinematic

Table 12-16 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, ...]
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)

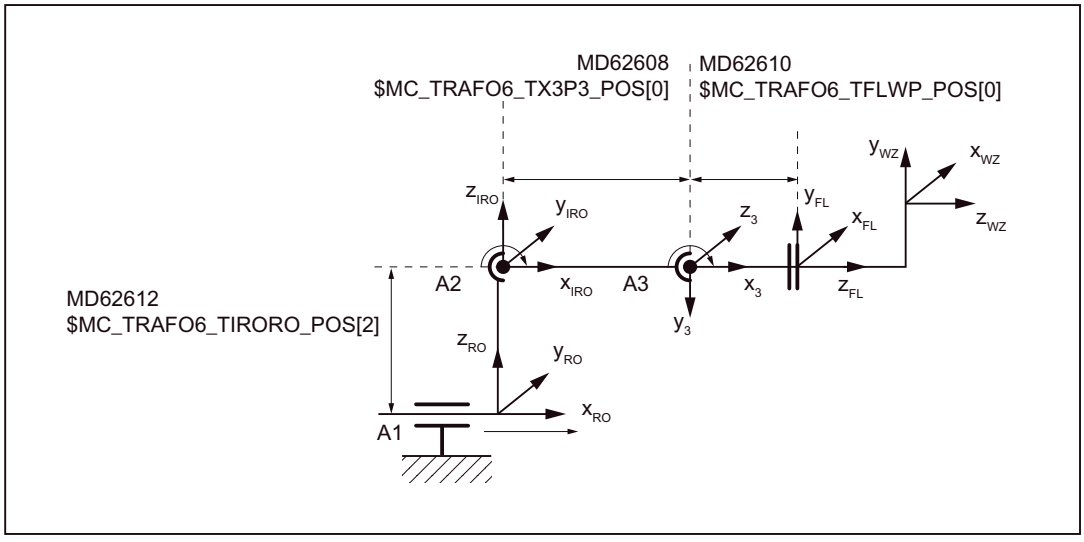


Figure 12-25 Special 3-axis SC kinematic

Table 12-17 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]
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)

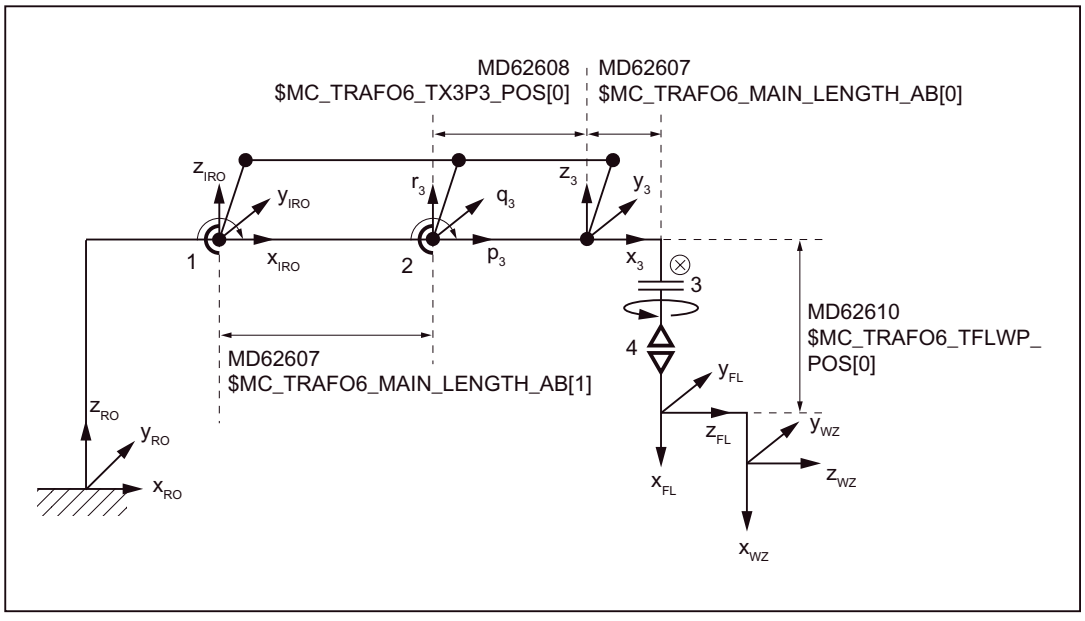


Figure 12-26 Special 4-axis SC kinematic

Table 12-18 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
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)

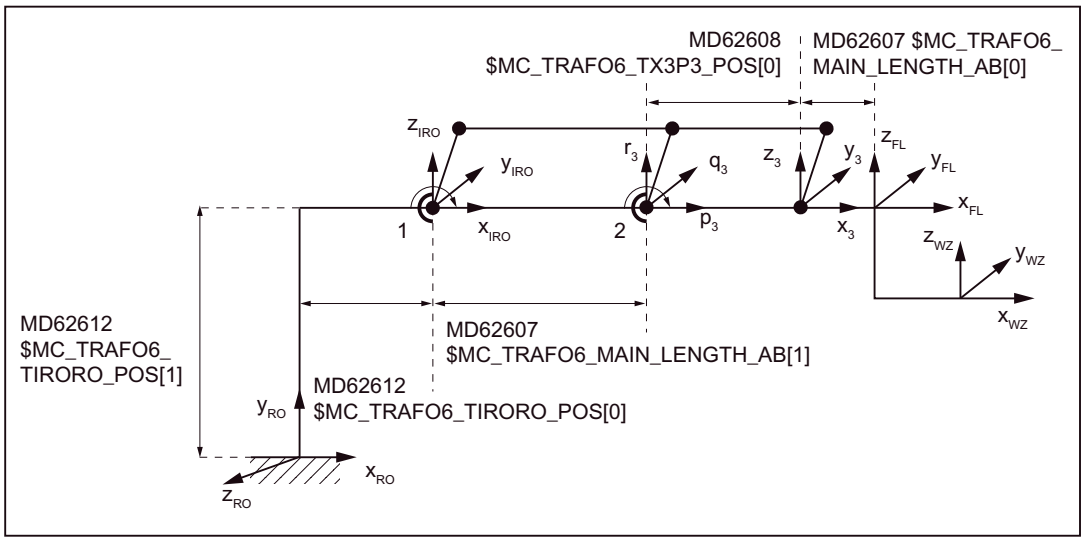


Figure 12-27 Special 2-axis NR kinematics

Table 12-19 Configuration data for special 2-axis NR kinematics

Machine data	Value
MD62600 $MC_TRAFO6_KINCLASS	2
MD62602 $MC_TRAFO6_SPECIAL_KIN	5
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]

12.6 Tool orientation

12.6.1 Tool orientation

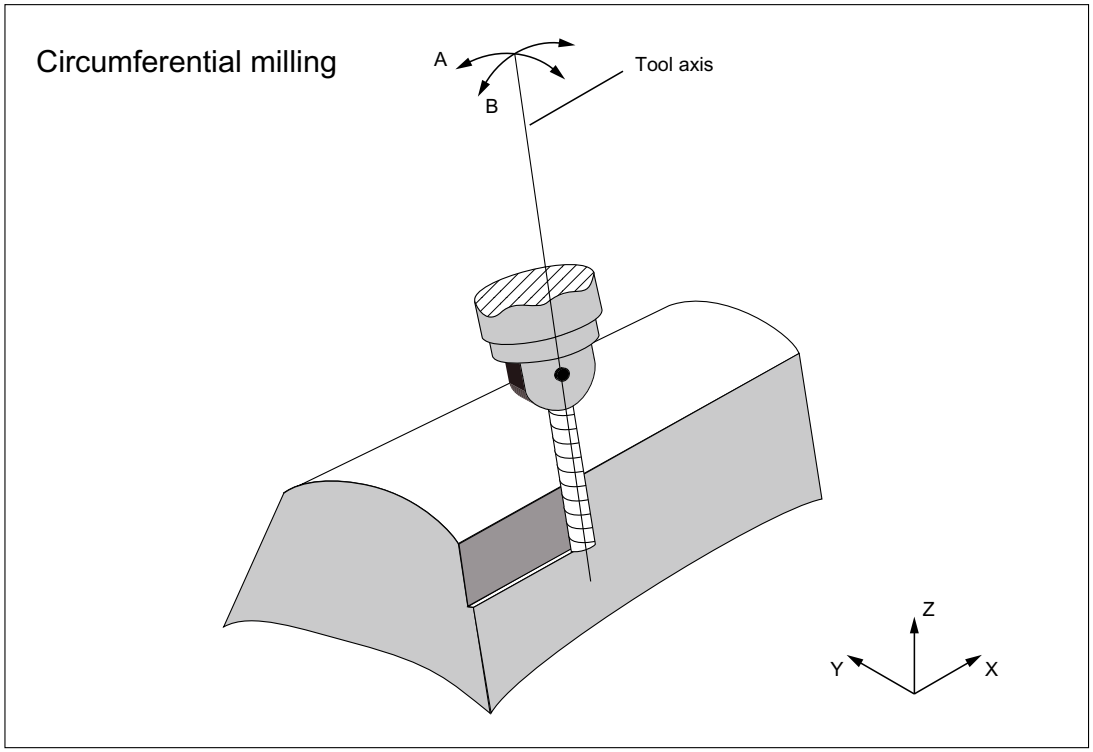


Figure 12-28 Workpieces with 5-axis transformation

Programming

Three possible methods can be used to program the orientation of the tool:

- directly as "orientation axes" A, B and C in degrees
- via Euler or RPY angles in degrees with A2, B2, C2
- using direction vectors A3, B3, C3.

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

Euler angle via machine data:

MD10620 $MN_EULER_ANGLE_NAME_TAB (name of Euler angle)

Direction vectors via machine data:

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.

Euler or RPY

Can be switched between Euler and RPY input via the machine data:

MD21100 $MC_ORIENTATION_IS_EULER (angle definition for orientation programming)

Note

It is not possible to program using Euler angles, RPY angles or direction vectors for kinematics involving fewer than 5 axes. In such cases, only one degree of freedom is available for orientation. This orientation angle can only be programmed with "Orientation axis angle" "A".

Orientation reference

A tool orientation at the start of a block can be transferred to the block end in the workpiece coordinate system only using the ORIWKS command.

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.

Note

Transferring an orientation using ORIMKS is not allowed in the handling transformation package. With an active transformation, it is not the machine axis angles that are programmed and traversed, but "orientation angles" (RPY angles according to robotics definition, see Subsection "Definition of positions and orientations using frames").

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

ORIMKS is the initial setting (SW version 2 and higher). The initial setting can be modified via machine data:

MD20150 $MC_GCODE_RESET_VALUES (RESET position of G groups)

GCODE_RESET_VALUES [24] = 1 → ORIWKS is initial setting

GCODE_RESET_VALUES [24] = 2 → ORIMKS is initial setting

GCODE_RESET_VALUES [24] = 3 → ORIPATH

When ORIPATH is active, the orientation is calculated from the lead and side angles relative to the path tangent and surface normal vector.

Improper tool orientation

If the tool orientation is programmed in conjunction with the following functions

- G04 Dwell time
- G33 Thread cutting with constant lead
- G74 Approach reference point
- G75 Approach fixed point
- REPOSL Repositioning
- REPOSQ Repositioning
- REPOSH Repositioning

Alarm 12130 "Illegal tool orientation" is output when Euler angles and direction vectors are selected. The NC program stops (this alarm can also occur in connection with G331, G332 and G63). 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.

Alarm 14400 is output if the transformation is switched on or off when a tool offset is active.

In the reverse situation, i.e. a tool 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=0 C3=0

If the tool orientation is input several times, e.g. with direction vectors and Euler angles:

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

error message 12240 "Channel X block Y tool orientation xx defined more than once" is displayed and the NC part program stops.

12.6.2 Orientation programming for 4-axis kinematics

Tool orientation for 4-axis kinematic

4-axis kinematics possess only one degree of freedom for orientation. When the orientation is programmed using RPY angles, Euler angles or direction vectors, it is not generally possible to guarantee that the specified orientation can be approached. If used at all, this type of orientation programming is only suitable for certain types of kinematic, i.e. those which feature an invariance in orientation angles relative to the basic axes. This is the case for the SCARA kinematic, 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 according to the robotics definition, i.e. one rotation about the Z-RO axis, as illustrated in Fig. "Orientation angle for 4-axis kinematic".

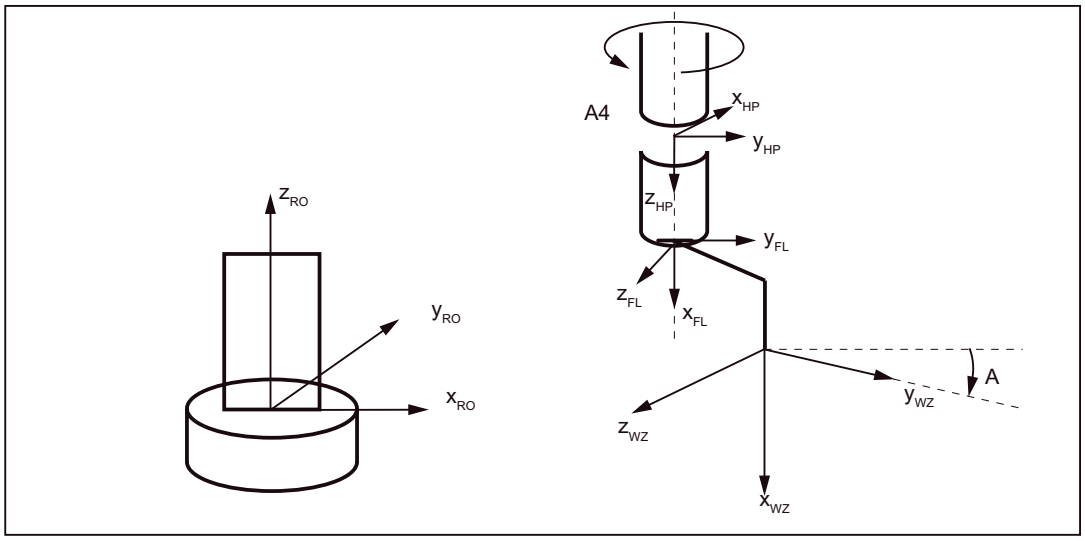


Figure 12-29 Orientation angle for 4-axis kinematic

12.6.3 Orientation programming for 5-axis kinematics

Tool orientation 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 as 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.

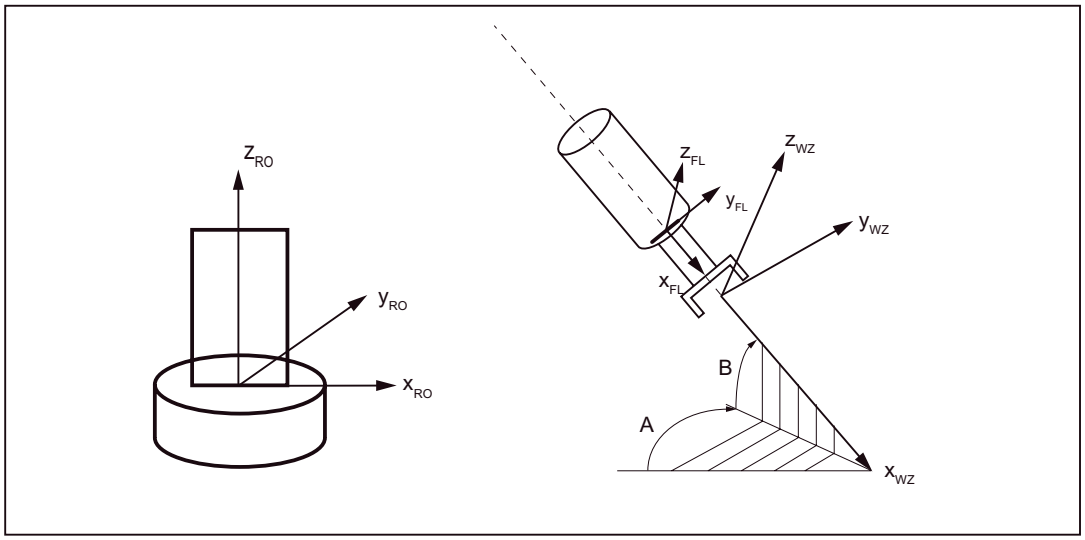


Figure 12-30 Orientation angle for 5-axis kinematic

It is possible to define orientation axes for the handling transformation package.

Note

Additional information can be found in:

/FB3/ Function Manual, Special Functions, "Orientation Axes"
and
/PGA/ Programming Guide, Advanced, "Orientation Axes".

12.7 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.

12.8 Call and application of the transformation

Powering-up

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:
/PGA/ Programming Guide, Advanced, "5-axis machining"

Deactivation

The currently active transformation is deactivated by means of `TRAFOOF` or `TRAFOOF()`.

Note

When deactivating the "handling transformation package" transformation, a preprocessing stop and a preprocessing synchronization are implicitly executed with the main run if the following machine data is set to 4099:

MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

If the following machine data is set to 4100, there is no implicit preprocessing stop:

MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

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 (reset/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 channel transformation 1) 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.

12.9 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 coordinates. Otherwise the geometry axis coordinates will be displayed.

12.10 Tool programming

Meaning

The tool lengths are specified in relation to the flange coordinate system. Only 3-dimensional tool compensations are possible. Depending on the kinematic type, there are additional tool restrictions for 5-axis and 4-axis kinematics. For a kinematic as illustrated in Fig. "5-axis NR kinematic", only a 1-dimensional tool with lengths in the x direction is permitted.

The tool direction is dependent on the machine's initial setting, as specified with G codes `G17`, `G18` and `G19`. The tool lengths refer to the zero position specified by `G17`. This zero position should not be modified 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) is specified as the tool identifier. 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 offset vector
$TC_DP4[1,1] = Y-TOOL ;	(y) Length offset vector
$TC_DP5[1,1] = X-TOOL ;	(x) Length offset vector

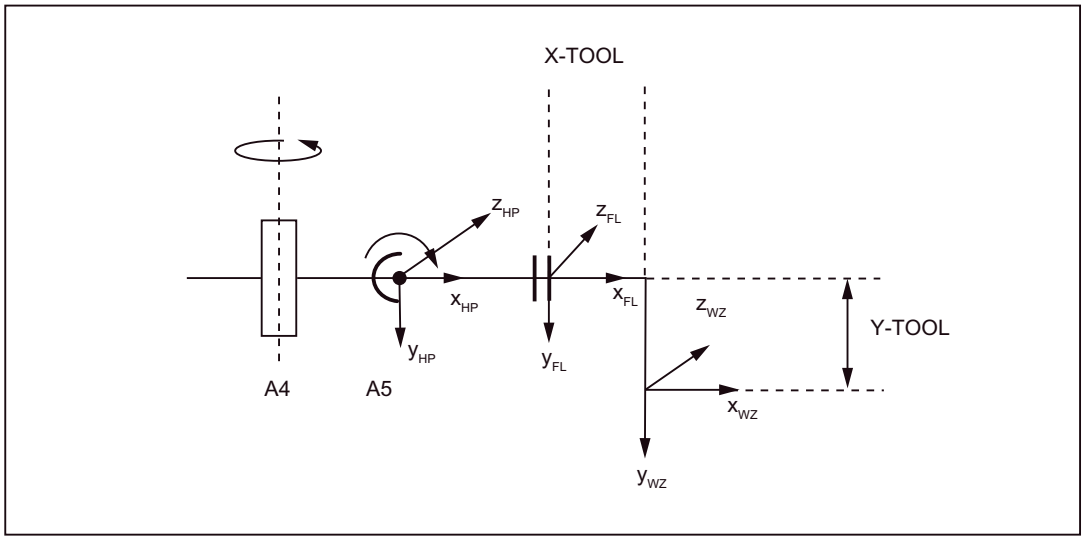


Figure 12-31 Tool length programming

12.11 Cartesian PTP travel with handling transformation package

It is possible to use the Cartesian PTP travel function with the handling transformation package. For this purpose the following machine data must be set to 4100:

MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

Note

For more information, see the Description of Functions, Special Functions F2 (Part 3) Section "Cartesian PTP Travel" and the "Programming Guide Advanced", "Cartesian PTP Travel".

12.12 Boundary conditions

12.12.1 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

12.12.2 Functional restrictions

NCU 572.2

The handling transformation package can be utilized on NCU 572.2 hardware only on condition that is has been specifically enabled for the customer.

Clearance control

The handling transformation package cannot be operated together with the technology function: "clearance control", as generally the three basic axes are not arranged perpendicular to one another.

Moving to fixed end stop

The handling transformation package cannot be operated in conjunction with the "travel to fixed stop" function.

Several transformations

The handling transformation package can only be activated once per channel.

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 crossed when a transformation is active. Singular positions can cause axis overloads. The feedrate is not automatically adjusted. The user must reduce the feedrate appropriately at the relevant points.

12.13 Examples

12.13.1 General information about start-up

Note

The compile cycles are supplied as loadable modules. The general start up of such compile cycles is described in TE01. The specific installation measures for this compile cycle can be found from Section "Starting up a kinematic transformation" onwards.

Requirement

HMI software version must be 3.5 or higher. An NCK-OEM Jeida card (technology card 2 or higher) must be available for 840D.

Procedure

Saving SRAM contents

As the first step in the installation of a compile cycle function for 840D, the technology card is exchanged for the original card provided in the NCU.

This step is equivalent to any procedure which involves the normal upgrade of your software platform and requires the deletion of any (battery-supported) static control memory. To avoid the consequential loss of all data in the SRAM, back up the SRAM before performing the operation. For a detailed description, please see the Manufacturer/Service Documentation "SINUMERIK 840D/SIMODRIVE 611D Installation and Start-Up Guide":

1. Enter the machine manufacturer password.
2. Switch to the "Services" operating area.
3. Press the "Series start-up" softkey.
4. Select "NC" and "PLC" as the areas to be saved and enter a name of your choice for the archive file to be created on the hard disk. Finish by pressing the RETURN key.
5. If the control system contains machine-specific compensation data, then these must be saved in a separate archive file:
   - Press the softkey "Data from 2" and select the data from the "NC active data" menu according to your needs
   - "Measuring system compensations"
   - "Sag/angularity comp."
   - "Quadrant error compensation".
   - Save this data by selecting softkey "Archive..." and enter another file name for a second archive file.

Keep the archive files you have created in a safe place. They will allow you to restore original settings in your system.

Inserting the PC card

1. Deactivate the control.
2. Insert the PC card with the new firmware (technology card) in the PCMCIA slot of the NCU.
3. Turn switch S3 on the front panel of the NCU to 1.
4. Switch the control system back on again.
5. During run-up, the firmware is copied from the PC card to the NCU memory.
6. Wait until number "6" is displayed on the NCU digital display (after approximately one minute).
7. Turn switch S3 back to zero.

⚠ CAUTION

If number "6" does not appear, an error has occurred.

- Incorrect PC card (e.g. card for NCU2 in NCU3 hardware)
- Card hardware is defective

Copy back SRAM contents

Note

The exact procedure of resaving the backed up data in the control is described in:
References:
/IAD/ SINUMERIK 840D Start-up Instructions; Chapter "Data Backup"

Please read all information provided by the manufacturer about new software versions.

1. Enter the machine manufacturer password.
2. Select "Data in" and "Archive...". Then load the archive with backup compensation data (if applicable).

12.13.2 Starting up a kinematic transformation

The next step necessary to start up the kinematic transformation is to activate the handling transformation package (option).

Set the option data for handling transformation package.

Interrupts

Record the alarm texts in the corresponding language-specific text files.

Set option data for transformation.

Configure the transformation

1. Enter transformation type 4099 or 4100 (if PTP travel is active) in the machine data:

   MD24100 $MC_TRAFO_TYPE_1 (definition of channel transformation 1)

2. Enter the assignment of the channel axes involved in the transformation in the machine data:

   MD24110 $MC_TRAFO_AXES_IN_1[0 to 5] (axis assignment for transformation)

   Axis numbers beginning with 1.

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 to 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:
   - Basic axis identifier in machine data:

     MD62603 $MC_TRAFO6_MAIN_AXES (basic axis identifier)
   - Basic axis lengths in machine data:

     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 hand:
    - Wrist axis identifier in the machine data:

      MD62604 $MC_TRAFO6_WRIST_AXES (wrist axis identifier)
    - Parameters for hand in the machine data:

      MD62614 $MC_TRAFO6_DHPAR4_5A (parameter A for configuring the hand)

      MD62615 $MC_TRAFO6_DHPAR4_5D (parameter D for configuring the hand)

      MD62616 $MC_TRAFO6_DHPAR4_5ALPHA (parameter ALPHA for configuring the hand)
    - 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.

12.14 Data lists

12.14.1 Machine data

12.14.1.1 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

12.14.1.2 Channel-specific 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

12.14.1.3 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)

Number	Identifier: $MC_	Description
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
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
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
62630	TRAFO6_ACCCP	Cartesian acceleration rates
62631	TRAFO6_VELORI	Orientation angle velocities
62632	TRAFO6_ACCORI	Orientation angle acceleration rates
62633	TRAFO6_REDVELJOG	Reduction factor for Cartesian velocities in JOG

12.14.2 Signals

12.14.2.1 Signals from channel

DB number	Byte.bit	Description
21, …	29.4	Activate PTP traversal
21, …	33.6	Transformation active
21, …	232	Number of active G function of G function group 25 (ORIWKS, ORIMKS, ORIPATH)
21, …	317.6	PTP traversal active

MCS Coupling (TE6)

13

13.1 Brief description

If a machine tool has 2 or more mutually independent traversing machining heads (in this case K1 (Y/ Z/ C/ A/ W or K2 (Y2/ Z2/ C2/ A2/ W2)), and if a transformation needs to be activated for the machining operation, the orientation axes cannot be coupled by means of the standard coupling functions (COPON, TRAILON). The only coupling function currently available in the machine coordinate system (MCS) is the GANTRY function. However, this cannot be activated in a part program and only permits 1:1 couplings.

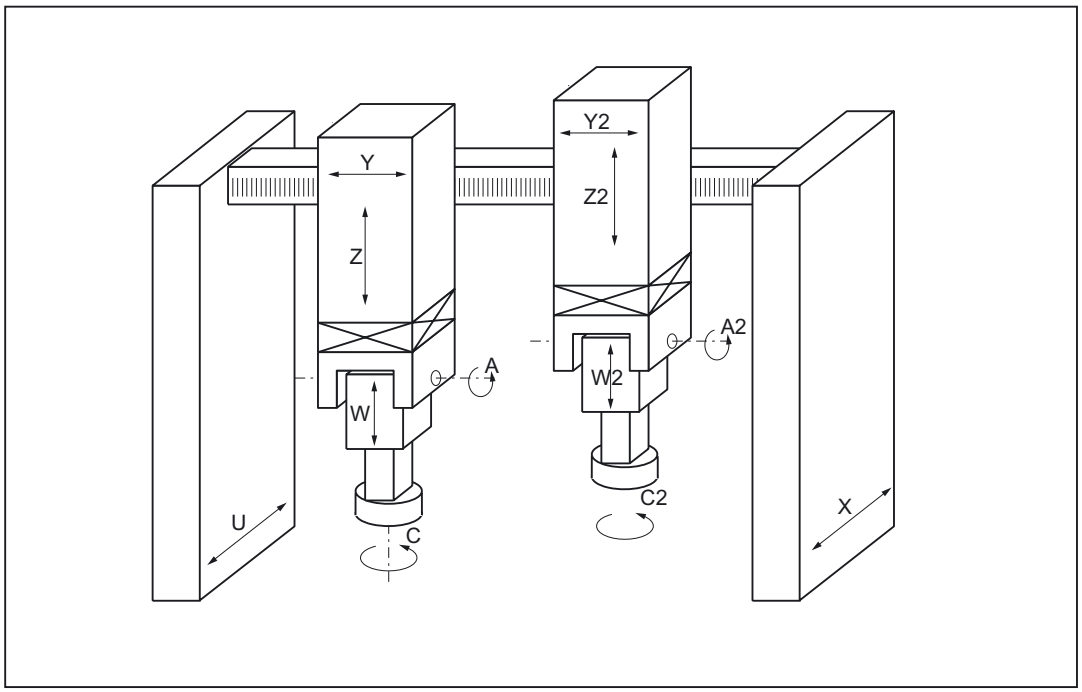


Figure 13-1 Compile cycle function "MCS coupling"

The compile cycle function "MCS coupling" allows a 1:1 or 1:-1 coupling in the machine coordinate system to be switched ON and OFF by part program commands.

MCS coupling

A 1:1 coupling in the machine coordinate system (MCS coupling) has been introduced in the compile cycle application.

The axes involved in the coupling are defined in an axial machine data. The machine data is updated by RESET to allow new axis pairs to be defined in operation.

CC_Master, CC_Slave

There are CC_Master and CC_Slave axes. A CC_Master axis can have several CC_Slave axes, but a CC_Slave axis cannot be a CC_Master axis (error message).

The coupling between these pairs is activated and deactivated by means of an OEM-specific language command and can thus be active in all operating modes. If a CC_Slave axis is programmed in a part program, either an alarm is output or a "GET" operation initiated, depending on machine data:

MD30552 $MA_AUTO_GET_TYPE (automatic GET from axis)

The following restrictions apply to CC-Slave axes:

- it cannot be made into a PLC axis,
- it cannot be made into a command axis,
- it cannot be operated separately from its CC_Master axis in JOG mode.

A tolerance window between the CC_Master and CC_Slave axes is specified via an axial machine data. When an MCS coupling is active, the actual values of the two axes must not leave this window.

Collision protection

To protect machining heads against collision in decoupled operation or in mirrored coupling mode, a collision protection can be set in a machine data. This is then activated either via a machine data or via the VDI-IN interface. The assignment of the protected pairs is not related to the CC_Master and CC_Slave pairs.

13.2 Description of MCS coupling functions

13.2.1 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 coupling's axes can only be changed when the coupling is not active.

A CC_Slave axis is displayed in axial VDI-Out byte:

DB31, … DBX97.0 (axis is slave axis)

Requirements

- 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 the CC_Master nor CC_Slave axis may be an exchange axis ($MA_MASTER_CHAN[AXn]=0)

13.2.2 Switching the coupling ON/OFF

CC_COPON()

CC_COPON([A1][A2][A3][A4][A5])

Switching the 1:1 coupling. Monitoring of tolerance window is active.

CC_COPONM()

CC_COPONM([A1][A2][A3][A4][A5])

Switching the 1:-1 coupling (mirror). Monitoring of tolerance window is not active.

A1 to A5 are axis names. These can be used to program either the machine axis names, channel axis names or geometry axis names of the axis assigned to a coupling. In other words, either the CC_Master axes or the CC_Slave axes or both can be programmed at the same time. An alarm is output if an axis not involved in a coupling is programmed in A1-A5. All defined couplings are switched on with CC_COPON() or CC_COPONM(). An active coupling is displayed via the CC_Slave axis in axial VDI-Out byte:

DB31, … DBX97.1 (activate coupling)

If mirroring is active, it is displayed additionally in:

DB31, … DBX97.2 (activate mirroring)

A coupling can be disabled via the CC_Slave axis in axial VDI-Out byte:

DB31, … DBX24.2 (disable CC_Slave axis coupling)

This does not generate an alarm.

CC_COPOFF()

CC_COPOFF([A1][A2][A3][A4][A5])

As CC_COPON or CC_COPONM() except for the fact that no alarm is generated if A1 to A5 is used to program an axis that is not involved in a coupling.

An existing coupling can also be switched off via the axial VDI-In bit on the CC-Slave axis.

The coupling can be switched ON or OFF only if all axes involved are stationary.

13.2.3 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 the tolerance window is violated.

The monitoring function is not active:

- if the machine data is set to 0,
- if the coupling is switched off,
- if axis/spindle inhibit is set for one of the axes,
- if an axis is in 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 by NC => PLC VDI-SS:

DB31, … DBX97.3 (offset after start-up)

Note

The offset might 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.

13.3 Description of collision protection

13.3.1 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.

13.3.2 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

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 (activate monitoring)

- 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

13.3.3 Configuring example

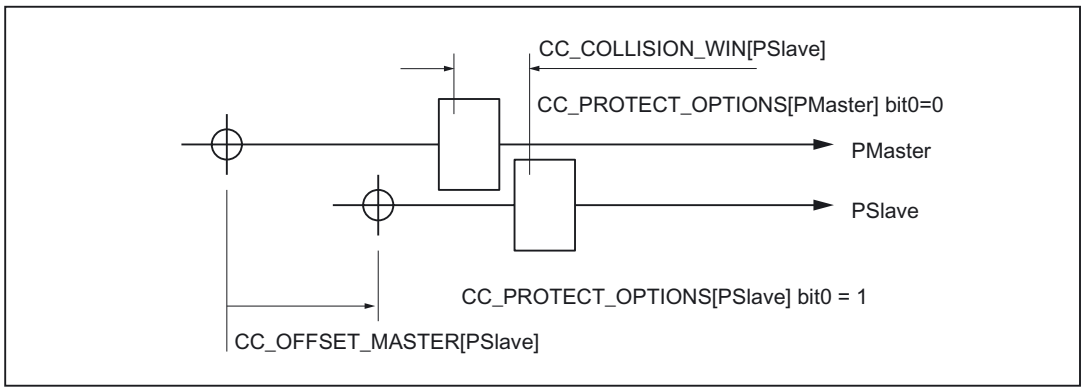


Figure 13-2 Configuring example

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.

13.4 User-specific configurations

Parking the machining head

In this context, "parking" means that the relevant machining head is not involved in workpiece machining. All axes are operating under position control and positioned at exact stop.

Even if a machining head is being used in production, coupling should be active! This is essential primarily if only the second head (Y2....) is being used. "Axis/spindle inhibit" must then be set axially (PLC -> NCK) 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 default

  Speed and direction of rotation inputs can be detected via synchronized actions or PLC and passed on to all other active spindles.

- Synchronous spindle function

13.5 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.

13.6 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.

13.7 Examples

13.7.1 General start-up of a compile cycle function

Note

The compile cycles are supplied as loadable modules. The general procedure for installing such compile cycles can be found in TE0. You can find the specific supplements to this compile cycle in the section entitled "Update of NCKOEM_CC_0013_01.02.00".

Saving SRAM contents

As the first step in the installation of a compile cycle function, the technology card is exchanged for the original card provided in the NCU.

This step is equivalent to any procedure which involves the normal upgrade of your software platform and requires the deletion of any (battery-supported) static control memory. To avoid the consequential loss of all data in the SRAM, back up the SRAM before performing the operation.

Please proceed as follows:

1. Enter the machine manufacturer password.
2. Switch to the "Services" operating area.
3. Press the "Series start-up" softkey.
4. Select "NC" and "PLC" as the areas to be saved and enter a name of your choice for the archive file to be created on the hard disk. Finish by pressing the RETURN key.
5. If the control system contains machine-specific compensation data, then these must be saved in a separate archive file:
   - Press the "Data out" softkey.
   - Select from the "NC active data" menu:
   - "Measuring system compensations"
   - "Sag/angularity comp."
   - "Quadrant error compensation".
   - Save these data by selecting softkey "Archive...".
   - Enter another file name for a second archive file.

These archive files will enable you to restore the original status if required.

A detailed description is contained in:

References:

/IAD/ Commissioning Manual SINUMERIK 840D/SIMODRIVE 611D

Insert the PC card

- Switch off control system
- Insert the PC card with the new firmware (technology card) in the PCMCIA slot of the NCU.
- Then proceed as follows:
  - Turn switch S3 on the front panel of the NCU to 1.
  - Switch the control system back on again.
  - During run-up, the firmware is copied from the PC card to the NCU memory.
  - Wait until number "6" appears on the NCU digital display(after approximately 60 s).
  - Turn switch S3 back to zero.

Note

If number "6" does not appear, then one of the following errors might have occurred:

- Incorrect PC card (e.g. card for NCU2 in NCU3 hardware)
- Card hardware defective.

Copy back SRAM contents

To copy the saved data back into the control system, proceed as described in the section entitled "Series start-up". Please read all information provided by the manufacturer about new software versions.

- Enter the machine manufacturer password.
- Select "Data in" and "Archive...".
- Load the archive with the backed up compensation data (if applicable).

13.8 Data lists

13.8.1 Machine data

13.8.1.1 Channel-specific 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

13.8.1.2 Axis/spindle-specific 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

Retrace Support (TE7)

14

14.1 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 retrace option is selected and deselected in advance using part program commands within the machining program. The program continuation point can be selected at any position within the contour ranges specified in this way.

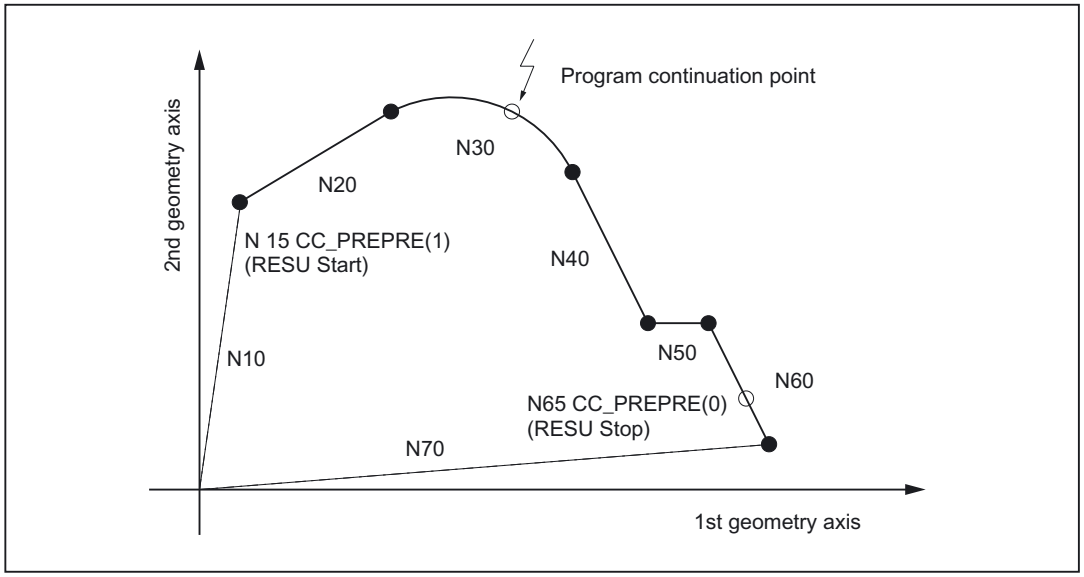


Figure 14-1 Programmed contour with program continuation and interruption points

Precise retracing of contours is possible on all programmed contours comprising straight and circular elements. During retracing, other programmed 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 through 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 first channel of 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 axis of the channel; refer to).

References

The "Continue Machining - Retrace support" technological function is a compile cycle. The description of the system-specific availability and use of compile cycles is located in:

- for SINUMERIK 840D in:

  /FB3/ Function Manual, Special Functions, Installation and Activation of Loadable Compile Cycles (TE01)

- for SINUMERIK 840Di in:

  /HBi/ SINUMERIK 840Di Manual; NC Start-up with HMI Advanced, Loadable Compile Cycles

14.2 Functional description

14.2.1 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 the machining continuation via 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.

Retraceable contour areas

RESU is activated via programming of 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 "continue machining".

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 starting and end points during reverse/forward travel.

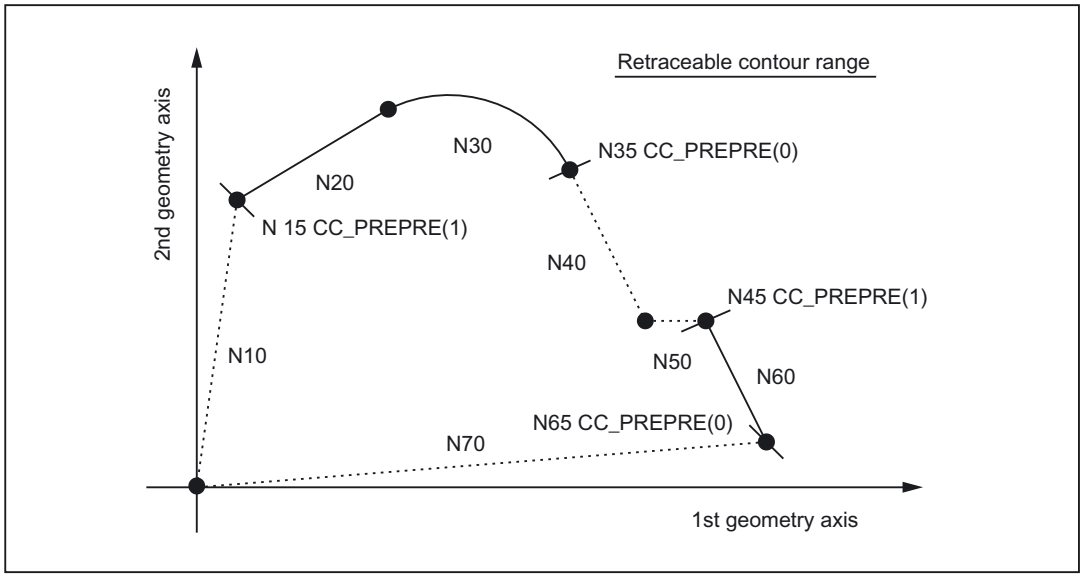


Figure 14-2 Retraceable contour range

14.2.2 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 area

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)` (refer to the "Retraceable contour ranges" screen).

14.2.3 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.

Preconditions

A part program with traversing blocks in the configured RESU working plane (e.g. 1st and 2nd geometry axis of the channel) as well as the part program 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 the reverse travel is by PLC interface signal:

   DB21, … DBX0.1 = 1 (forward / reverse)

3. Reverse travel:

   The contour is traversed in the RESU working plane in the reverse direction with NC start. Instead of the current machining program, RESU selects the automatically generated RESU main program.

   For more about RESU programs see Chapter "RESU-specific part programs".

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 PLC interface signal:

   DB21, … DBX0.1 = 0

6. Forward travel (optional):

   The contour is traversed in the RESU working plane 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 ASUB "CC_RESU_BS_ASUP.SPF" is initiated when the last action block is reached:

   DB21, … DBX32.6 = 1 (last action block active)

   For more information about ASUB, see Chapter "RESU-specific part programs".

   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 chart for interface signals

The principle sequence of the RESU function is illustrated in the following figure as a signal chart of the interface signals involved:

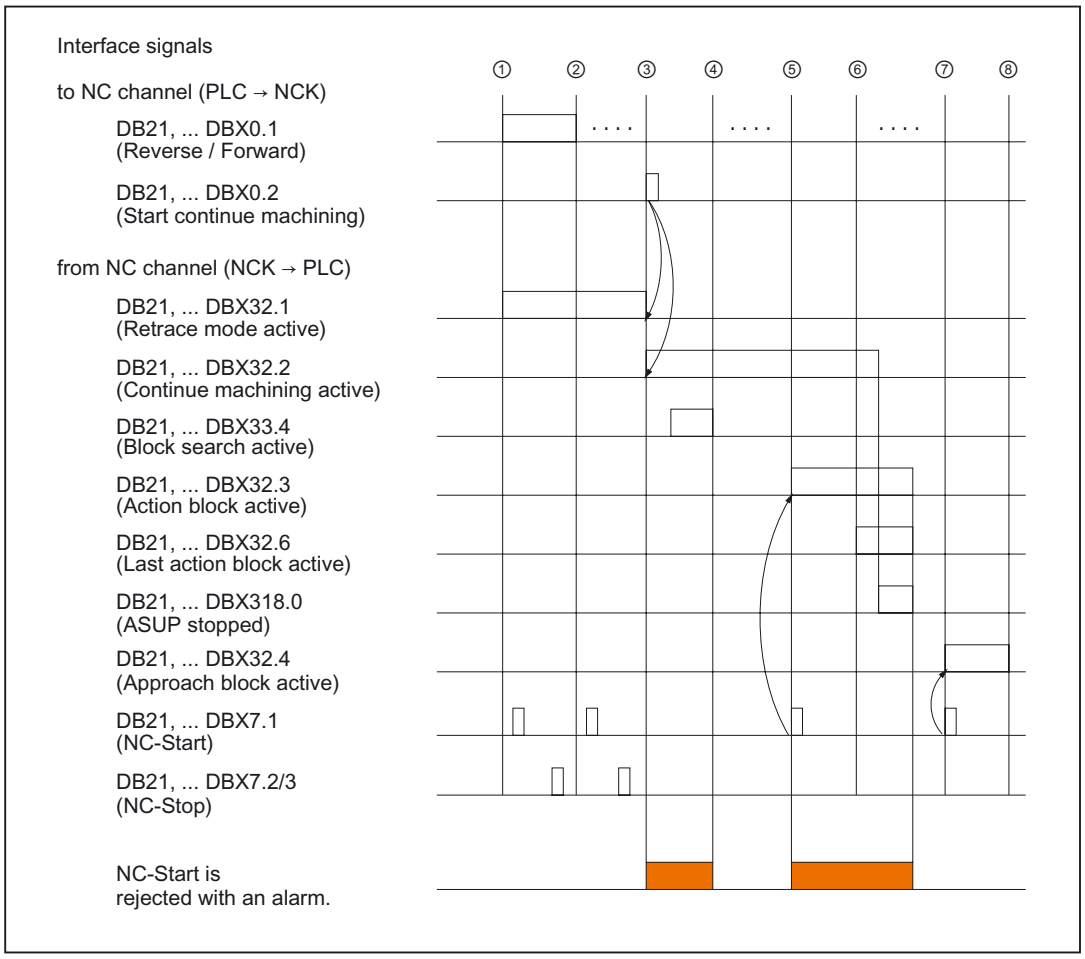


Figure 14-3 Signal chart

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 ASUB "CC_RESU_BS_ASUP.SPF" is triggered when the last action block is activated.

⑦ 2nd NC start →Return travel to approach block for program continuation point.

⑧ Part program processing (target block) resumed

14.2.4 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:

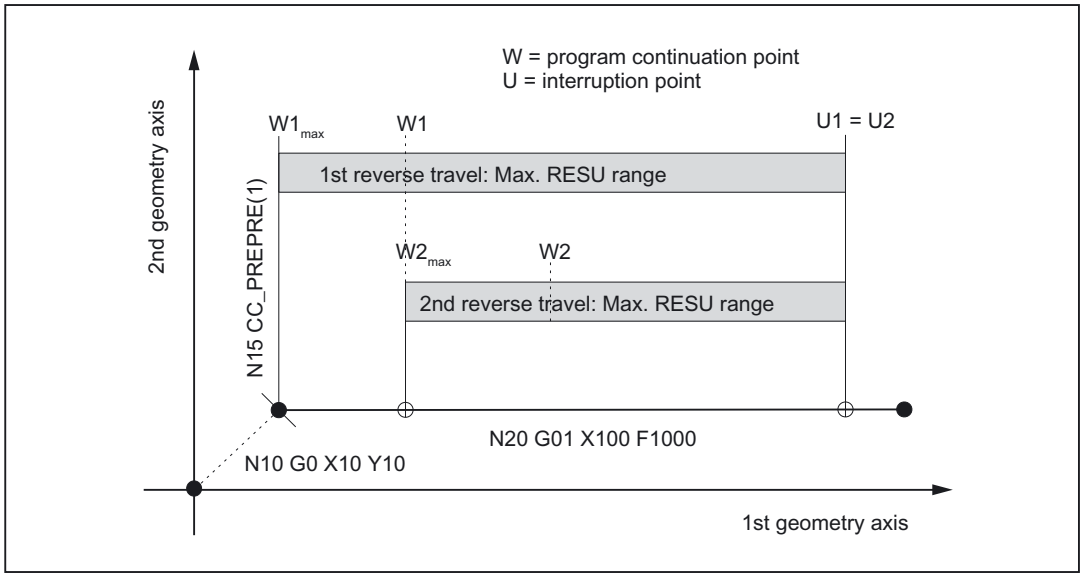


Figure 14-4 Maximum Retraceable contour range

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.

14.3 Startup

14.3.1 Activation

Before starting up the technological function, make sure that the corresponding compile cycle has been loaded and activated.

References:
/FB3/ Function Manual, Special Functions, Installation and Activation of Loadable Compile Cycles (TE01)

/HBi/ SINUMERIK 840Di Manual; NC Start-up with HMI Advanced, Loadable Compile Cycles

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.

14.3.2 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).

14.3.3 Memory configuration: Block memory

RESU requires additional memory area in the NCK-internal block memory.

No. of block elements

The number of the block elements that can be used for compile cycles is set via the memory configuring channel-specific machine data:

MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS

For RESU, the already existing machine data value (x) is adjusted as follows:

MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS = x + **4**

Size of the block memory

The size of the block memory in KB that can be used by the user for compile cycles is defined via the memory configuring channel-specific machine data:

MD28100 $MC_MM_NUM_CC_BLOCK_USER_MEM

For RESU, the already existing machine data value (x) is adjusted as follows:

MD28100 $MC_MM_NUM_CC_BLOCK_USER_MEM = x + **20**

14.3.4 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 Fig. "RESU program structure") the more part program blocks can be traversed in reverse.

  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 subroutine searches in the context of RESU.

  180 bytes are required for each subroutine. The block search buffer requires at least 2880 bytes (16 subroutine calls with 180 bytes each).

  The block search buffer cannot be parameterized directly.

  The size of the block search buffer is displayed via a function-specific GUD variable (for creation of GUD variables, please refer to the Chapter 'Function-specific Display Data / Channel-specific GUD Variables").

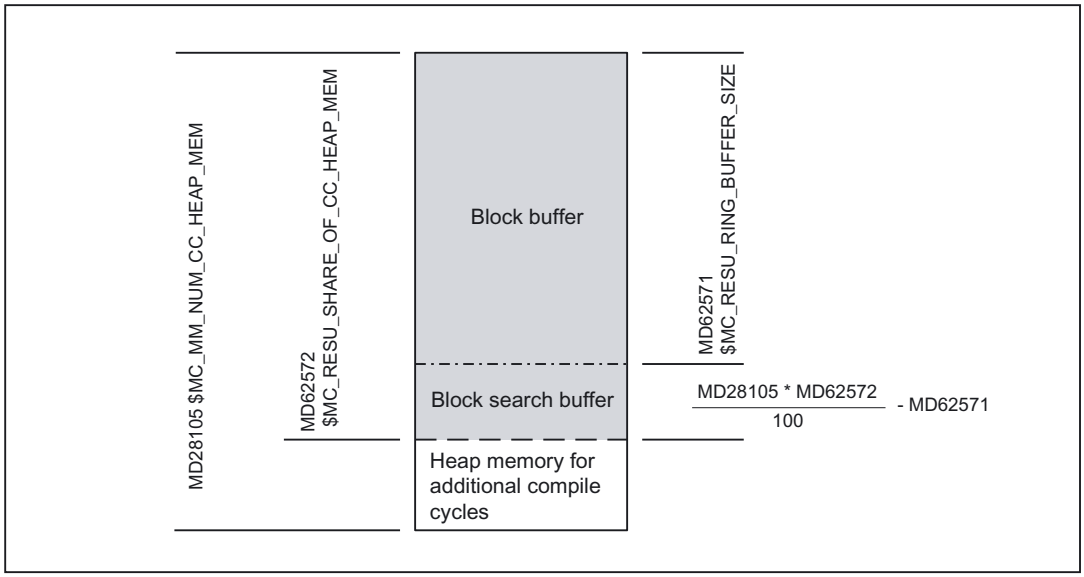


Figure 14-5 Division of the heap memory for compile cycles

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 subroutine 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"

14.3.5 RESU main program memory area

Memory configuration

The storage location of the RESU main program CC_RESU.MPF can be set via the following machine data (refer to Main program (CC_RESU.MPF) (Page 580)):

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.

14.3.6 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

14.3.7 ASUB enable

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)

14.3.8 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
```

14.4 Programming

14.4.1 RESU Start/Stop/Reset (CC_PREPRE)

Start / stop / reset / of RESU is done with the program instruction:

CC_PREPRE (**Prep**are **Re**trace)

Programming

Syntax: CC_PREPRE(<mode>)

Parameters:

Mode: Type: INTEGER
Range of values: -1, 0, 1

CC_PREPRE(...) is a procedure call and must therefore be programmed in a dedicated part program block.

Functionality

The following modes are available:

Statement	Meaning
CC_PREPRE(1)	Starts logging the traversing blocks. The information required for reverse travel is logged on a block-specific basis in a RESU-internal block buffer. The travel 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
CC_PREPRE(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 during reverse travel.
CC_PREPRE(-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

In reset events:

- NCK POWER ON RESET (warm start)
- NCK Reset
- End of program (M30)

`CC_PREPRE(-1)` is executed implicitly.

Error messages

The following programming errors are detected and displayed with alarms:

- Invalid mode programmed:

  RESU alarm 75601 "Channel *number* Block *number* Invalid parameter for CC_PREPRE( )"

- More than one parameter programmed:

  Alarm 12340 "Channel *number* Block *number* Too many parameters"

- RESU technology function is not available:

  The technological function is not available. The compile cycle is possibly not loaded or it was not activated.

  Alarm 12340 "Channel *number* Block *number* Name CC_PREPRE not defined or option not available"

14.5 RESU-specific part programs

14.5.1 Overview

RESU uses the following, automatically generated and partially adjustable part programs:

Program	Name
Main program	CC_RESU.MPF
INI program	CC_RESU_INI.SPF
END program	CC_RESU_END.SPF
Continue machining ASUB	CC_RESU_BS_ASUP.SPF
RESU ASUB	CC_RESU_ASUP.SPF

The following figure provides an overview of the internal structure of the technological function and the relationship between the various part programs.

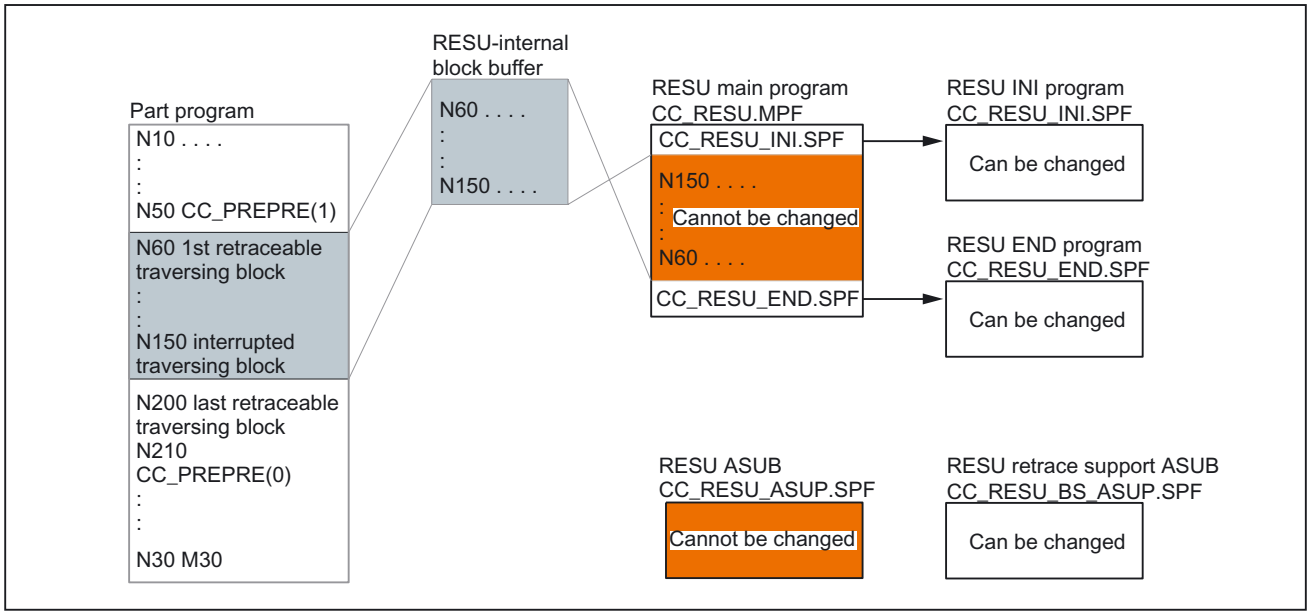


Figure 14-6 RESU-specific part programs

14.5.2 Main program (CC_RESU.MPF)

Function

In addition to the calls for the RESU-specific subroutines, 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 subroutines.

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 (refer to RESU main program memory area (Page 574)), 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.

14.5.3 INI program (CC_RESU_INI.SPF)

Function

The RESU-specific subroutine "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 (refer to Frames (Page 594)): `G500`
- To switch off the active tool offsets (refer to Tool offsets (Page 595)): `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
```

Note

CC_RESU_INI.SPF may be changed.

CC_RESU_INI.SPF must not contain any RESU part program commands. `CC_PREPRE(x)`.

⚠ CAUTION

By changing the content of the RESU-specific subroutine "CC_RESU_INI.SPF", the user (machine manufacturer) takes over the responsibility for the correct sequence of the technological function.

14.5.4 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
```

Note

CC_RESU_END.SPF may be changed.

CC_RESU_END.SPF must not contain any RESU part program commands. `CC_PREPRE(x)`.

> **⚠ CAUTION**
>
> 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.

14.5.5 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
```

Note

CC_RESU_BS_ASUP.SPF may be changed.

User-specific modifications must be inserted before the part program block `RMN`.

> **⚠ CAUTION**
>
> 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.

14.5.6 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.

14.6 Retrace support

14.6.1 General

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 Functional sequence (principle) (Page 567).

Subfunctions

The two essential subfunctions of retrace support are the standard NC functions:

- Block search with calculation at contour
- Repositioning on the contour via shortest route (`REPOS RMN`)

14.6.2 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:
/FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Mode (K1), Program test

14.6.3 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.

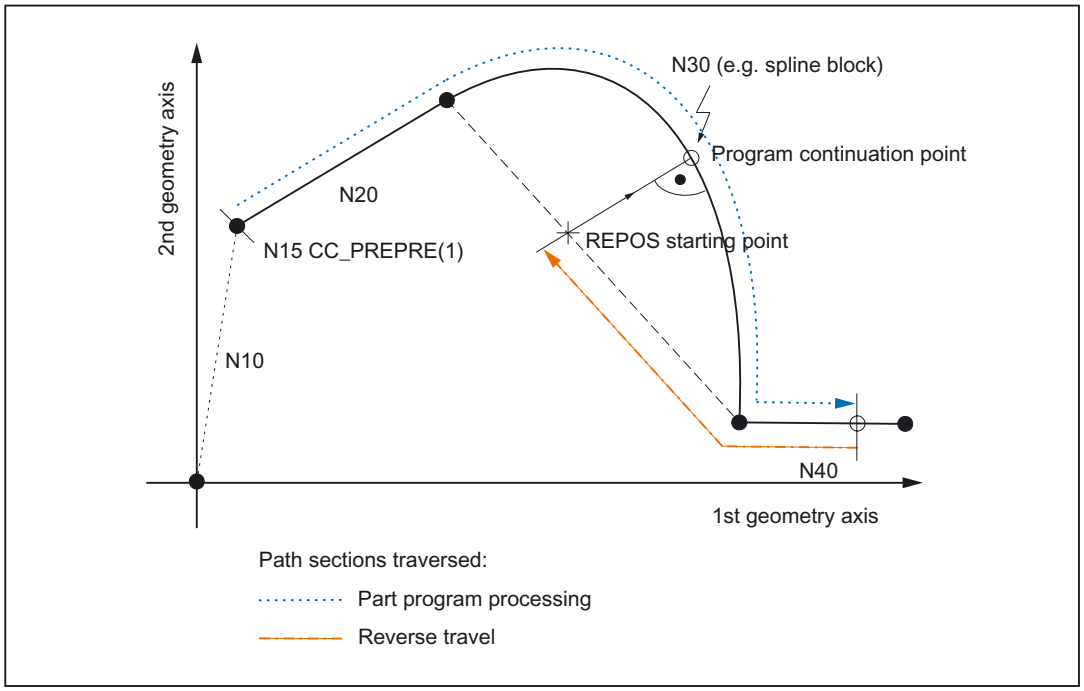


Figure 14-7 Retraceable contour ranges and REPOS

Channel axes

All other channel axes programmed in the part program travel to the relevant position calculated in the block search.

14.6.4 Temporal conditions concerning NC start

NC start should be initiated twice by the machine operator within the framework of continue machining (refer to Functional sequence (principle) (Page 567)).

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)

14.6.5 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. Subroutines 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 subroutines.

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:
/PG/ 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.

Constraints

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
```

14.7 Function-specific display data

14.7.1 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 NCK POWER ON Reset. Hence, a NCK 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. Hence, the corresponding operating manuals are referenced here:
/BHU sl/ HMI sl Universal Operating Manual
/BAD/ HMI Advanced Operating Manual
/BEM/ HMI Embedded Operating Manual

14.8 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

14.9 Boundary conditions

14.9.1 Function-specific boundary conditions

14.9.1.1 Continue machining within subroutines

Subroutine call outside or inside a program loop

Clear retrace support within subroutines depends on whether the subroutine call is made outside or inside a program loop:

- Outside

  Clear retrace support is possible if a subroutine is called outside a program loop.

- Inside

  Clear retrace support may not be possible if a subroutine is called inside a program loop (refer to Continue machining within program loops (Page 591)).

Number of passes P

Subroutine 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.

14.9.1.2 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`

If retrace support is performed within program loops, the retrace support is always effective in the first loop run.

⚠ WARNING

If the machining continuation point on the programmed contour is the result of a loop run not equal to the first loop run, there can be considerable contour deviations in the further course of the machining under certain circumstances, which pose a threat to man and machine.

14.9.1.3 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.

14.9.1.4 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.

14.9.2 Boundary conditions for standard functions

14.9.2.1 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)`

14.9.2.2 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
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.

14.9.2.3 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*"

14.9.2.4 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.

14.9.2.5 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.

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:
/FB2/ Function Manual, Extended Functions, Kinematic Transformation (M1)
/FB3/ Function Manual, Special Functions, Transformation Package Handling (TE4)

14.9.2.6 Compensation

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:
/FB2/ Function Manual, Extended Functions; Compensations (K3)

14.9.2.7 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 subroutine "CC_RESU_INI.SPF" (refer to INI program (CC_RESU_INI.SPF) (Page 581)).

A full description of the frames can be found in:
References:
/FB1/ Function Manual, Basic Functions; Axes, Coordinate Systems, Frames (K2)

14.9.2.8 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 subroutine "CC_RESU_INI.SPF" (refer to INI program (CC_RESU_INI.SPF) (Page 581)).

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:
References:
/FB1/ Function Manual, Basic Functions; Tool Compensation (W1)

14.10 Data lists

14.10.1 Machine data

14.10.1.1 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)

14.10.1.2 Channel-specific 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
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

Cycle-Independent Path-Synchronous Signal Output (TE8)

15

15.1 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 relevant NC module can be used as the digital I/O via which the switching signal is output:

SINUMERIK	Output of the switching signal
840D	The switching signal can only be output via one of the 4 on-board digital outputs on the NCU module.
840Di	The switching signal can only be output via one of the 4 on-board digital outputs on the MCI board extension module (option).

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. The description of the system-specific availability and use of compile cycles is located in:

- for SINUMERIK 840D in:

  /FB3/ Function Manual, Special Functions, Installation and Activation of Loadable Compile Cycles (TE01)

- for SINUMERIK 840Di in:

  /HBi/ SINUMERIK 840Di Manual; NC Start-up with HMI Advanced, Loadable Compile Cycles

15.2 Functional description

15.2.1 General

Note

The functionality is described with examples, with the help of the "High speed laser cutting technology (HSLC, High Speed Laser Cutting).

15.2.2 Calculating the switching positions

15.2.2.1 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.

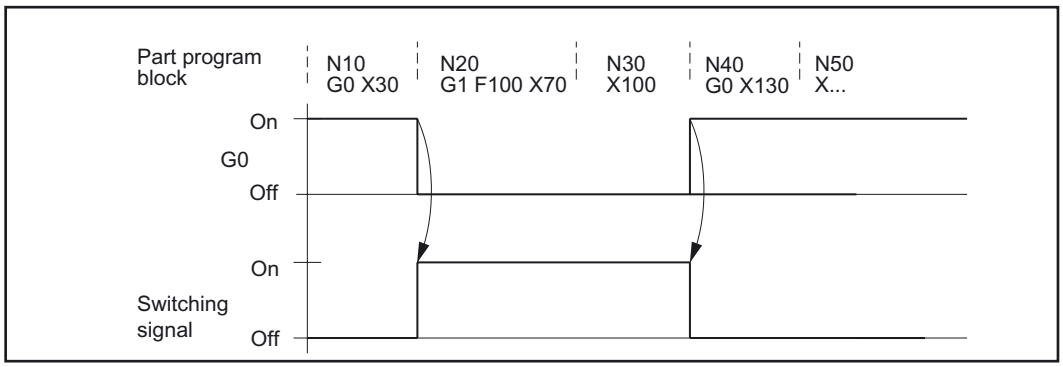
The following criteria define the setpoint position programmed in the part program block (end of block position) as a switching position:

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`

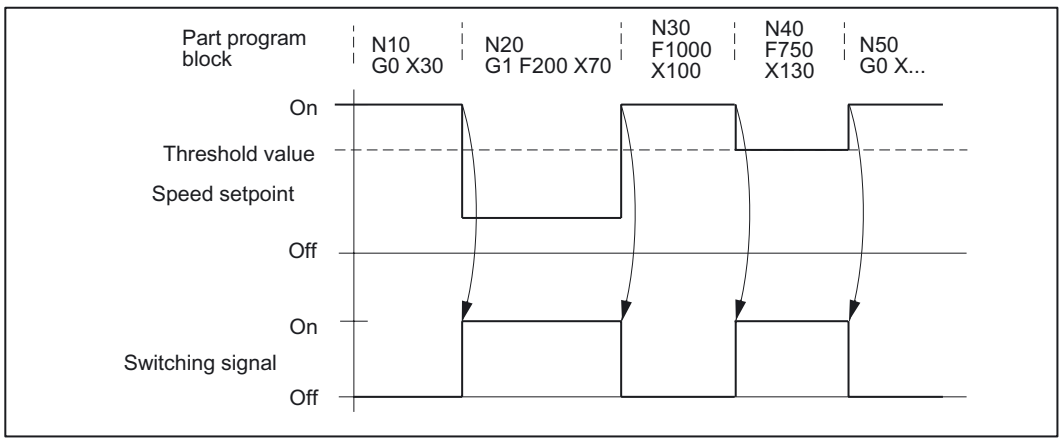
Freely programmable velocity threshold value as switching criterion

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.

15.2.2.2 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).

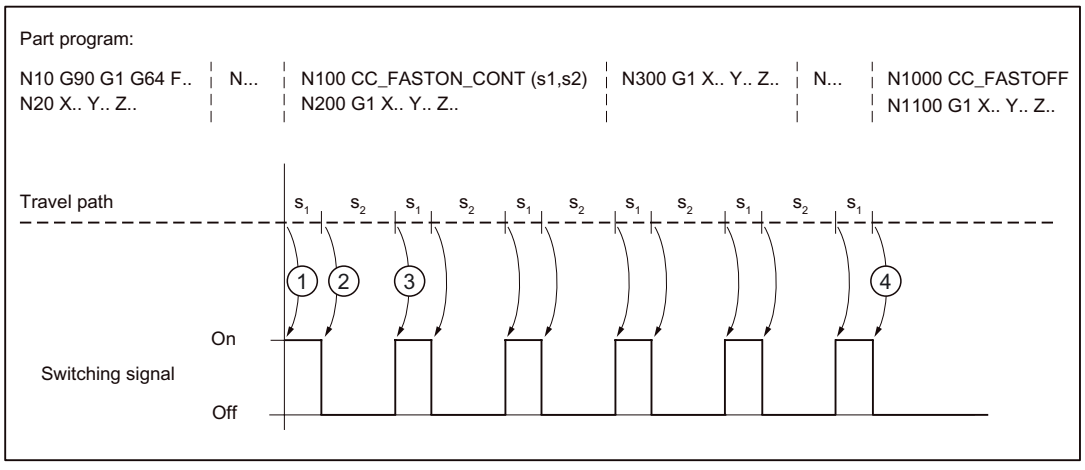


Figure 15-1 Path length-related switching signal output

15.2.3 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.

15.2.4 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.

15.2.5 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.

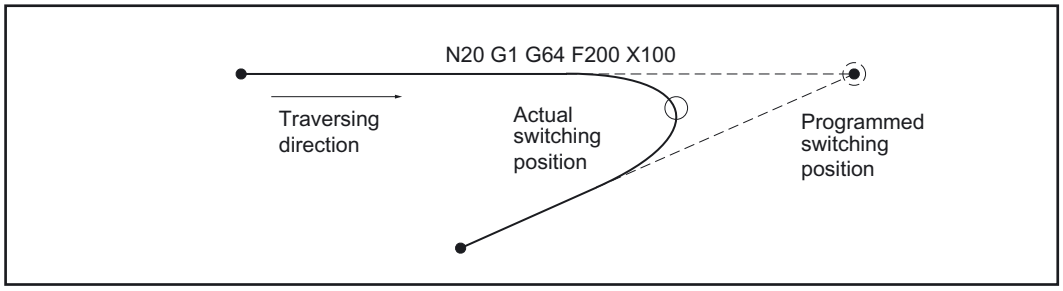


Figure 15-2 Switching position offset in path control mode

15.2.6 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.

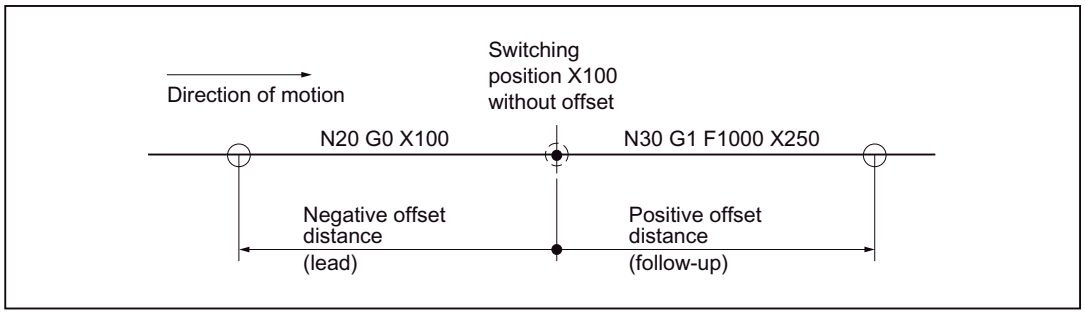


Figure 15-3 Programmed switching position offset

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)

15.2.7 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).

15.3 Start-up

15.3.1 Activation

Before starting up the technological function, make sure that the corresponding compile cycle has been loaded and activated.

References:
/FB3/ Function Manual, Special Functions, Installation and Activation of Loadable Compile Cycles (TE01)

/HBi/ SINUMERIK 840Di Manual; NC Start-up with HMI Advanced, Loadable Compile Cycles

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.

15.3.2 Memory configuration

The "Cycle-independent path synchronized switching signal output" technology function requires additional memory area in the NCK internal block memory.

No. of block elements

The number of the block elements that can be used for compile cycles is set via the memory configuring channel-specific machine data:

MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS

The already existing machine data value (x) is adjusted as follows for the "Cycle-independent path synchronized switching signal output" technology function:

MD28090 $MC_MM_NUM_CC_BLOCK_ELEMENTS = x + **1**

Size of the block memory

The size of the block memory in KB that can be used by the user for compile cycles is defined via the memory configuring channel-specific machine data:

MD28100 $MC_MM_NUM_CC_BLOCK_USER_MEM

The already existing machine data value (x) is adjusted as follows for the "Cycle-independent path synchronized switching signal output" technology function:

MD28100 $MC_MM_NUM_CC_BLOCK_USER_MEM = x + **10**

15.3.3 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:

- for SINUMERIK 840D in:

  /FB2/ Function Manual, Extended Functions, Digital and Analogue NCK I/Os (A4)

- for SINUMERIK 840Di in:

  /HBi/ SINUMERIK 840Di Manual, NC Start-up with HMI Advanced, Digital and Analog I/Os

15.3.4 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.

15.3.5 Parameterization of the geometry axes

Standard 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 next NCK booting.

15.4 Programming

15.4.1 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.

15.4.2 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.

15.4.3 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.

15.5 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

15.6 Boundary conditions

15.6.1 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`.

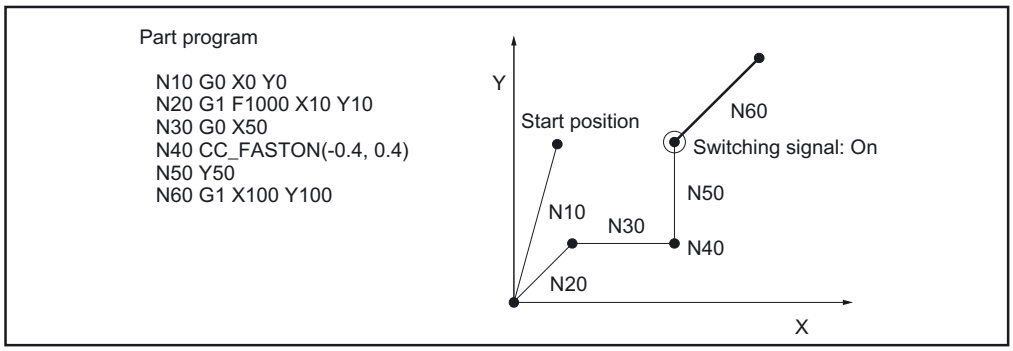


Figure 15-4 Switching signal for part program machining operation

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.

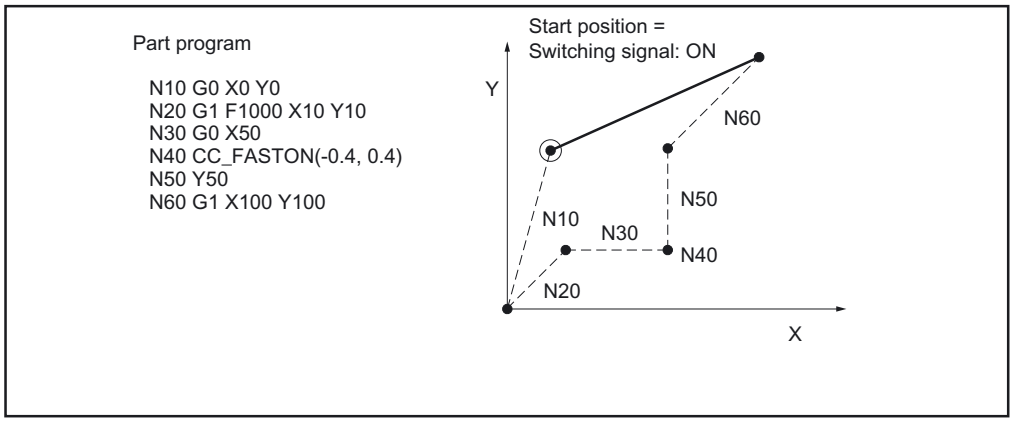


Figure 15-5 Switching signal after block search

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:

/FB1/ Function Manual, Basic Functions, Mode Group, Channel, Program Operation, Reset Response (K1)

15.6.2 Transformations

The function will only run correctly with deactivated transformation. There is no monitoring function.

A description of the transformations can be found in:
References:
/FB2/ Function Manual, Extended Functions; Kinematic Transformations (M1)

/FB3/ Function Manual, Special Functions, Transformation Package Handling (TE4)

15.6.3 Compensations

The following compensations are considered while calculating the switching positions:

- Temperature compensation
- Sag compensation

A description of the compensations can be found in:
References:
/FB2/ Function Manual, Extended Functions; Compensations (K3)

15.6.4 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:
References:
/FB1/ Function Manual, Basic Functions; Tool Compensations (W1), Tool Radius Compensation

15.6.5 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:

/FB1/ Function Manual, Basic Functions, Continuous-Path Mode, Exact Stop and LookAhead (B1)

15.6.6 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:
References:
/FB2/ Function Manual, Extended Functions; Software cams, Position switching signals (N3)

15.7 Data lists

15.7.1 Machine data

15.7.1.1 General machine data

Number	Identifier: $MN_	Description
10360	FASTO_NUM_DIG_OUTPUTS	Number of digital output bytes

15.7.1.2 Channel-specific 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

Axis pair collusion protection (TE9) 16

16.1 Brief description

The "Axis pair collusion protection" (compile cycle file CCPROT.ELF) enables a collision protection for axis pairs (max. 5) moving on the same guide rails that can thus collide. The axis pairs should not be located in the same channel.

16.2 Parameter assignment

Switch on CC channel

The CC channel (compile cycle channel) is specifically activated in the machine data:

MD60972 $MN_CC_ACTIVE_IN_CHAN_PROT[2]

Defining axis pairs

All axis pairs are defined in the machine data:

MD61516 $MN_PROTECT_PAIRS[n]

The first axis is defined in the 1st and 10th decade and the second axis is defined in the 100th and 1000th decade.

Free travel direction of axes

The concerned free travel direction of the axes must be recorded in the machine data:

MD61517 $MN_CC_PROTECT_SAVE_DIR[n]

As in axis pair definition, the following is valid here too:

The first two decades are valid for the first axis of the pair, the next two decades for the second axis.

A value greater than 0 means "free travel in the plus direction", the value 0 means "free travel in the minus direction".

Zero point offset

A zero point offset between the first and second axes can be specified in the machine data:

MD61518 $MN_CC_PROTECT_OFFSET[n]

Protection window

The protection window is recorded in the machine data:

MD61519 $MN_CC_PROTECT_WINDOW[n]

16.3 Activation/deactivation

Enabling/disabling

The protection is active as soon as:

- a valid axis pair is defined in the machine data:

  MD61516 $MN_PROTECT_PAIRS[n]

- both axes are referred.

If the two axes are already inside the protection window at the turn-on time, then the operator must move the axes freely. The software does not prevent any colliding movement in this condition!

The function is deactivated by deleting the axis pair definition in MD61516

Status display

The status display of the axis pair is optional.

"Global resources" are locked without status display,

_PROTECT_STATUS

If the control detects the NCK-GUD "_PROTECT_STATUS[n]" at the time of the ramp-up, then the current status of each pair is recorded in it:

Value	Meaning
0	Not active
1	activated, but not yet active
2	brake three active axes
4	Deactivating

16.4 Function-specific alarm texts

The procedure to be following while creating function-specific alarm texts is described in:
References:
/FB3/ Function Manual, Special Functions; Installation and Activation of Readable Compile Cycles (TE01), Section: Creating alarm texts

16.5 Examples

Example

Machine axes A1, A3 and A12 move on a common guide rail (refer to following figure).

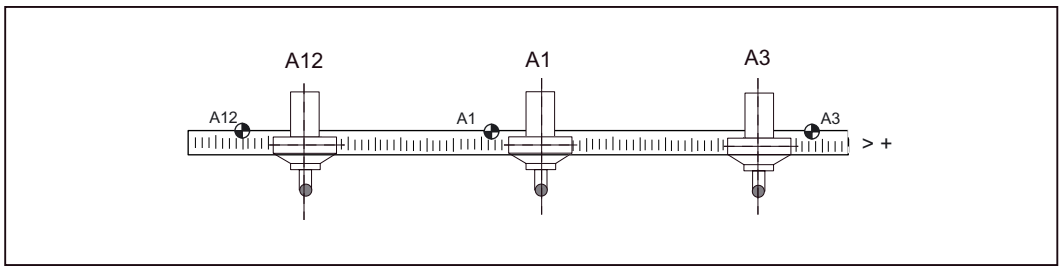


Figure 16-1 Machine axes A1, A3, A12

Collision protection

A full protection can be achieved through 2 pairs independently of the associated channel of the axes:

Pair 1

$MN_CC_PROTECT_PAIRS[0] = 103	1st axis = A3 2nd axis = A1
$MN_CC_PROTECT_SAVE_DIR[0] = 1	Free travel of A3 in + A1 in – A3 → A1
$MN_CC_PROTECT_OFFSET[0] =	10.0
$MN_CC_PROTECT_WINDOW[0] =	

Pair 2

$MN_CC_PROTECT_PAIRS[1] = 1201	1st axis = A1 2nd axis = A12
$MN_CC_PROTECT_SAVE_DIR[1] = 1	Free travel of A1 in + A12 in – A1 → A12
$MN_CC_PROTECT_OFFSET[1] =	10.0
$MN_CC_PROTECT_WINDOW[1] =	

Result

In each interpolation cycle, the control calculates the positions of the two axes, if they are braked at maximum acceleration (MD63514 $MA_CC_PROTECT_ACCEL).

If the destination points are located inside the protection window, then the axis moving in the direction of collision is braked.

The control closes even if the maximum braking acceleration is less than the current acceleration of the interpolators (MD32200 $MA_MAX_AX_ACCEL * (MD32433 $MA_SOFT_ACCEL_FACTOR or
MD32434 $MA_G00_ACCEL_FACTOR))!

NOTICE
A possible path group is cancelled. The axes do not brake on the programmed path! An alarm 75653 is output as soon as the axes reach zero speed.

16.6 Data lists

16.6.1 Machine data

16.6.1.1 General machine data

Number	Identifier: $ON_	Description
19610	TECHNO_EXTENSION_MASK[3 (6)]	Release of function with the help of MD19610 in Bit 4 $ON_TECHNO_EXTENSION_MASK[2] = 'H10'

16.6.1.2 Machine data of the compile cycle function

General machine data

Number	Identifier: $MN_	Description
60972	CC_ACTIVE_IN_CHAN_PROT[2]	The CC_channel is specifically activated with this MD (0 <= xx >= 99). Bit 0 = 1 => Channel 1 Bit 1 = 1 => Channel 2 etc.
61516	CC_PROTECT_PAIRS[5]	The axis pairs are determined with this MD. The 1st and 10th decades define the machine axis number of the first axis, and the 100th and 1000th decade define the machine axis number of the second axis. Example: $MN_CC_PROTECT_PAIRS[0] = 1201 ; Axis 1 = 1, Axis 2 = 12 A wrong configuration is rejected with an alarm.
61517	CC_PROTECT_SAVE_DIR[5]	The concerned free direction of travel of the two axes are recorded here. The 1st and 10th decades are valid for the first axis, and the 100th and 1000th decade are valid for the second axis. A value > 0 means free travel in the "+" direction. 0 means free travel in the "–" direction.
61518	CC_PROTECT_OFFSET[5]	Offset of the two axis with reference to each other.
61519	CC_PROTECT_WINDOW[5]	Minimum distance that must be maintained between the two axes.

NOTICE
The MD61517 - MD61519 machine data is not updated for an active pair!

Axis-specific machine data

Number	Identifier: $MA_	Description
61514	CC_PROTECT_ACCEL	Brake acceleration of the axis. In case of protection, the axis is braked at this acceleration.

GUD data

DEF NCK INT_PROTECT_STATUS[5]

If the control detects the NCK-GUD "_PROTECT_STATUS[n]" at the time of the ramp-up, then the current status of each pair is recorded in it.

The values have the following meanings:

- 0 → not active
- 1 → activated but not yet active
- 2 → brake 3 active axes
- 4 → Disable

Remark: Must not be configured.

Preprocessing (V2)

17

17.1 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.

17.2 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 subroutines, 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 preprocessed in a compilation optimized for processing.
3	1	During control power-up, the user cycles in the _N_CUS_DIR directory are preprocessed 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 specifications 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

Subroutines (_SPF file extension) located in the directories for standard cycles (_N_CST_DIR, _N_CMA_DIR) and user cycles (_N_CUS_DIR) and any subroutines 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:
/FB2/ 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 subroutine 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:

```
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
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".

17.3 Program call

Overview

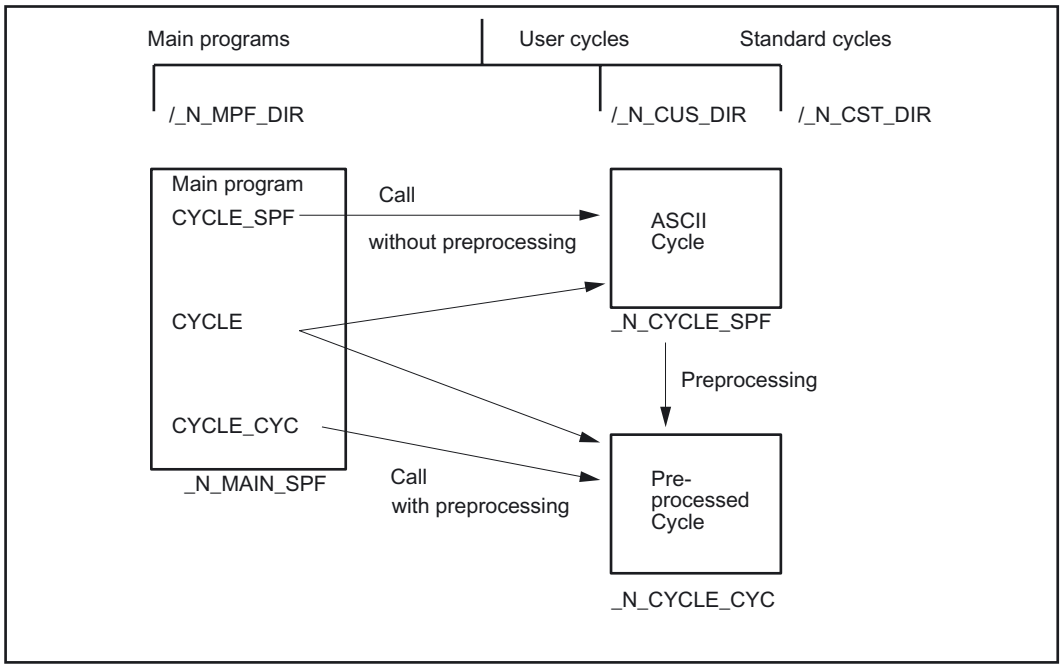


Figure 17-1 Generation and call of preprocessed cycles without parameters

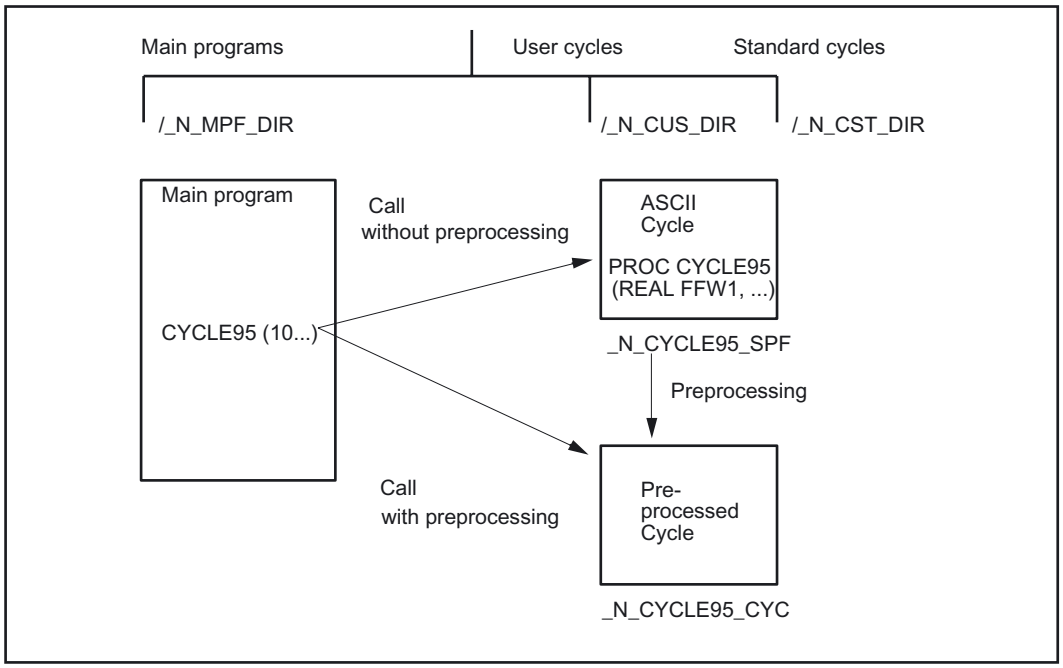


Figure 17-2 Generation and call of preprocessed cycles with parameters

Start

- 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 if available. An error message is output if no compilation is available.

    Example: CYCLE_CYC; Preprocessed cycle call
  - 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 the pre-compiled cycle is called, an explicit change is made to 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.

Do not use PUDs in cycles that are 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 current 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 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.

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 NCK POWER ON.

17.4 Constraints

Availability of the "preprocessing" function

The function is an option and is available with SINUMERIK 840D.

Vocabulary

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 identifier

Part programs are compiled independently of channels. For this reason, the geometry and channel identifiers 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 identifiers are not used directly in machining cycles since cycles are written as follows:

- independently of channels and
- independently of the axis identifiers 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)

17.5 Examples

17.5.1 Preprocessing individual files

```
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

Subroutine UP1 is pretranslated, and the call description is generated.

Subroutine UP2 is not pretranslated, but the call description is generated.

b) Bit 5 = 0

MD10700 $MN_PREPROCESSING_LEVEL=13 ; bit 0, 2, 3,

Both subroutines are pretranslated, and the call description is generated.

c) Example of an **invalid** subroutine with activated compiling:

```
PROC SUB1 PREPRO
G291                              ; ← Alarm when compiling, G291 not possible
G0 X0 Y0 Z0
M17
```

17.5.2 Preprocessing in the dynamic NC memory

Machine data for preprocessing only in the dynamic NC memory with selective selection:

```
                                  ; Bit 5 = 1 Selective program
                                    selection
                                  ; Bit 6 =0 No switching 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:

```
                                  ; Bit 5 = 1 Selective program
                                    selection
                                  ; Bit 6 = 1 Switching to static NC
                                    memory if dynamic NC memory is full
N30 $MN_MM_DRAM_FILE_MEM_SIZE = 800   ; Reserve space
N40 $MN_PREPROCESSING_LEVEL = 127     ; Bit 0-6 = 1
M17
```

17.6 Data lists

17.6.1 Machine data

17.6.1.1 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

17.6.1.2 Channel-specific 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)

3D Tool Radius Compensation (W5) 18

18.1 Brief description

18.1.1 General

Why 3D TRC?

3D tool radius compensation is used to machine contours with tools that can be controlled in their orientation independently of the tool path and shape.

Note

This description is based on the specifications for 2D tool radius compensation.

References:

/FB1/ Function Manual, Basic Functions; Tool Offset (W1)

How $2^1/_2$ D and 3D TRC differ

- With **$2^1/_2$D TRC**, it is assumed that the tool is always space-bound. Tools with constant orientation (cylindrical tools) are used for circumferential milling operations.

  While the orientation of the machining surface is not constant when other tools are used, it is determined by the contour and cannot thus be controlled independently of it.

- With **3D TRC**, surfaces with variable orientation are generated.

  The prerequisite for circumferential milling is that the tool orientation can be changed, i.e. in addition to the 3 degrees of freedom needed to position the tool (normally 3 linear axes), a further two degrees of freedom (2 rotary axes) are required to set the tool orientation (5-axis machining).

  End faces can be milled with 3 or 5 degrees of freedom.

Circumferential milling, face milling

The following diagram shows the differences between $2^1/_2$D and 3D TRC with respect to circumferential milling operations.

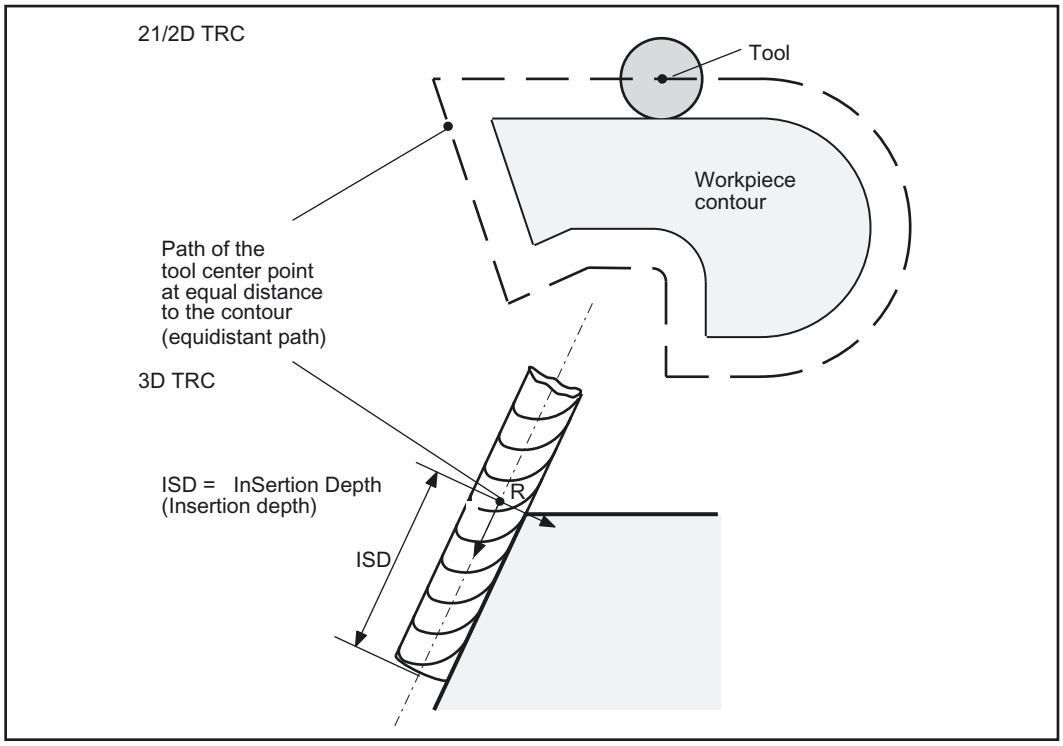


Figure 18-1 21/2D and 3D tool radius compensation

The parameters for the operation shown in Fig. "Face milling" are described in detail in the corresponding Subsection.

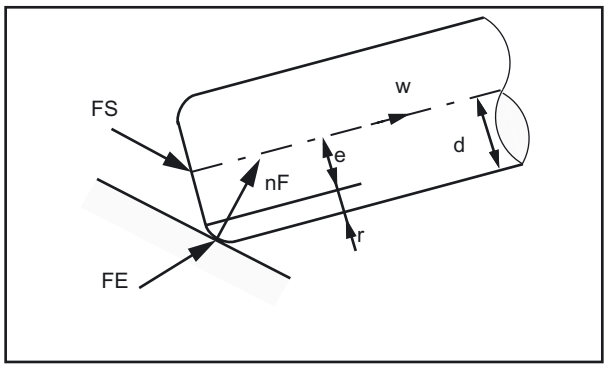


Figure 18-2 Face milling

Orientation

With 3D TRC, a distinction must be drawn between:

- Tools with space-bound orientation
- Tools with variable orientation

18.1.2 Machining modes

There are two modes for milling spatial contours:

- Circumferential milling
- Face milling

Circumferential milling mode is provided for machining so-called ruled surfaces (e.g. taper, cylinder, etc.) while face milling is used to machine curved (sculptured) surfaces.

Circumferential milling

Tools will be applied as follows for circumferential milling:

- with space-bound orientation ($2^1/_2$D TRC) and
- with variable orientation (3D TRC)

3D TRC can therefore be applied in circumferential milling only if the tool orientation is variable.

Intermediate blocks that are required from non-tangential transitions for mathematical reasons can be avoided using the intersection procedure. In these cases, the two curves in question are extended; the intersection of both extended curves is approached.

Face milling

Tools of both types, i.e. with constant or variable orientation, can be used for face milling operations.

Tools with variable orientation offer the following advantages:

- Better approximation of end contour
- Greater cutting capability
- Wider selection of tool shapes
- Wider range of surfaces can be machined (relief cuts).

18.2 Circumferential milling

Circumferential milling

The variant of circumferential milling used here is implemented through the definition of a path (directrix) and the associated orientation. In this machining mode, the tool shape is irrelevant on the path and at the outside corners. The only decisive factor is the radius at the tool contact point.

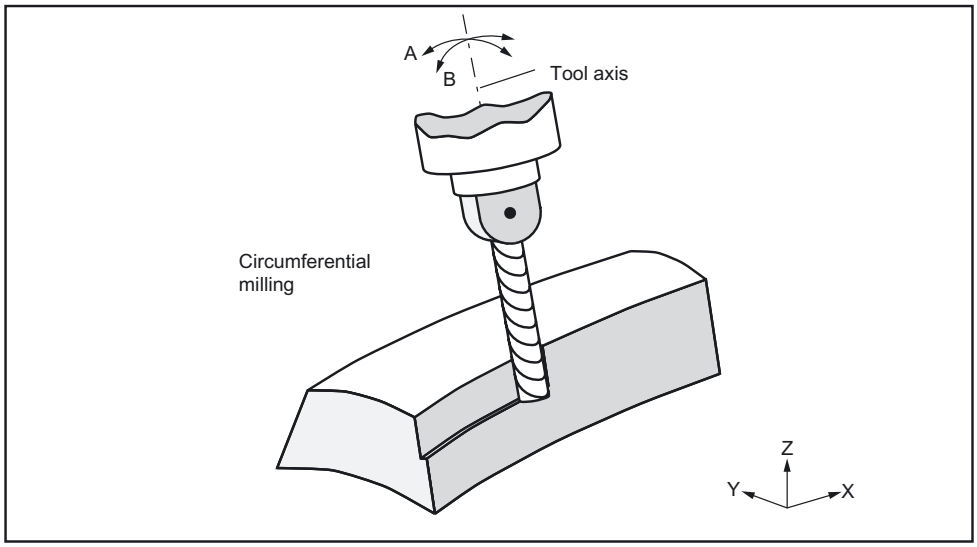


Figure 18-3 Circumferential milling

Insertion depth (ISD)

Program command `ISD` (insertion depth) is used to program the tool insertion depth for circumferential milling operations. This makes it possible to change the position of the machining point on the outer surface of the tool.

`ISD` defines the distance between cutter tip FS and cutter construction point FH. Point FH is obtained by projecting the programmed machining point onto the tool axis. `ISD` is evaluated only when 3D TRC is active.

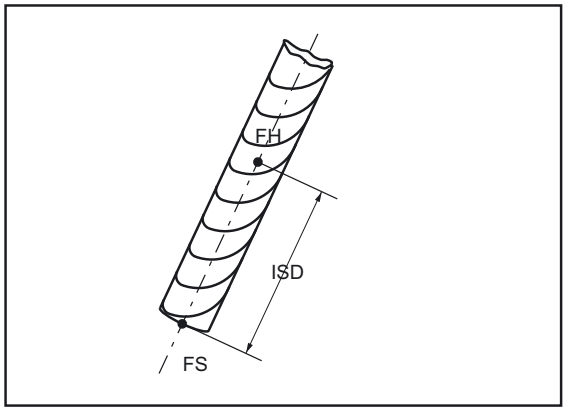


Figure 18-4 Insertion depth

18.2.1 Corners for circumferential milling

Outside corners/inside corners

Outside corners and inside corners must be treated separately. The terms inside corner and outside corner are dependent on the tool orientation. 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.

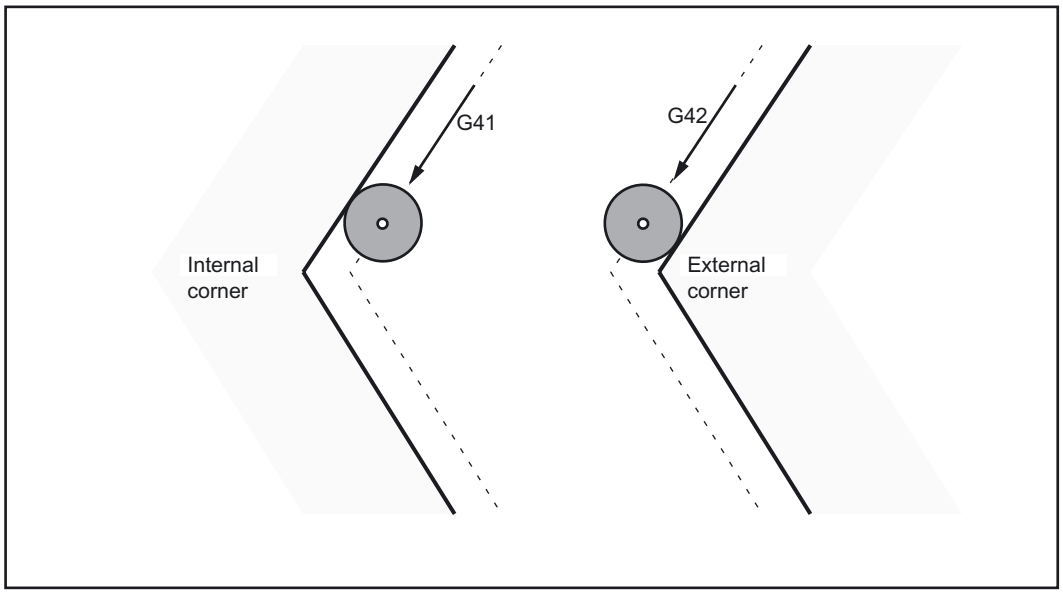


Figure 18-5 Corner type

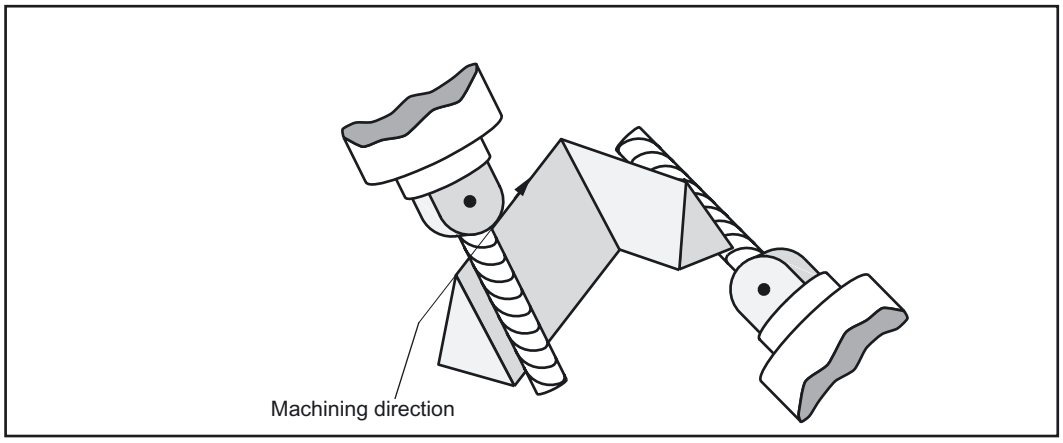


Figure 18-6 Change of corner type during machining

18.2.2 Behavior at outer corners

In the same manner as 21/2D tool radius compensation procedures, a circle is inserted at outer corners for G450 and the intersection of the offset curves is approached for G451.

With nearly tangential transitions, the procedure for active G450 is as with G451 (limit angle is set via machine data). 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 (MD).

If there is a change in orientation between the two traversing blocks, a circle is always inserted.

G450

Outside corners are treated as if they were circles with a 0 radius. The tool radius compensation acts on these circles in the same way as on any other programmed path.

The circle plane extends from the final tangent of the first block to the start tangent of the second block.

The orientation can be changed during block transition.

A change in orientation between two programmed blocks is executed either before the circle block or in parallel to it. Circles are always inserted. The command DISC is not evaluated.

Programming

- ORIC: Change in orientation and path movement in parallel

  (**ORI**entation Change **C**ontinuously)
- ORID: Change in orientation and path movement consecutively

  (**ORI**entation Change **D**iscontinuously)

The ORIC and ORID program commands are used to determine whether changes in orientation programmed between two blocks are executed before the inserted circle block is processed or at the same time.

When the orientation needs to be changed at outside corners, the change can be implemented in parallel to interpolation or separately from the path motion. When ORID is programmed, the inserted blocks are executed first without a path motion (blocks with changes in orientation, auxiliary function outputs, etc.). The circle block is inserted immediately in front of the second of the two traversing blocks which form the corner.

ORIC

If ORIC is active and there are two or more blocks with changes in orientation (e.g. A2= B2= C2=) programmed between the traversing blocks, then the inserted circle block is distributed among these intermediate blocks according to the absolute changes in angle.

Change in orientation

The method by which the orientation is changed at an outer corner is determined by the program command that is active in the first traversing block of an outer corner.

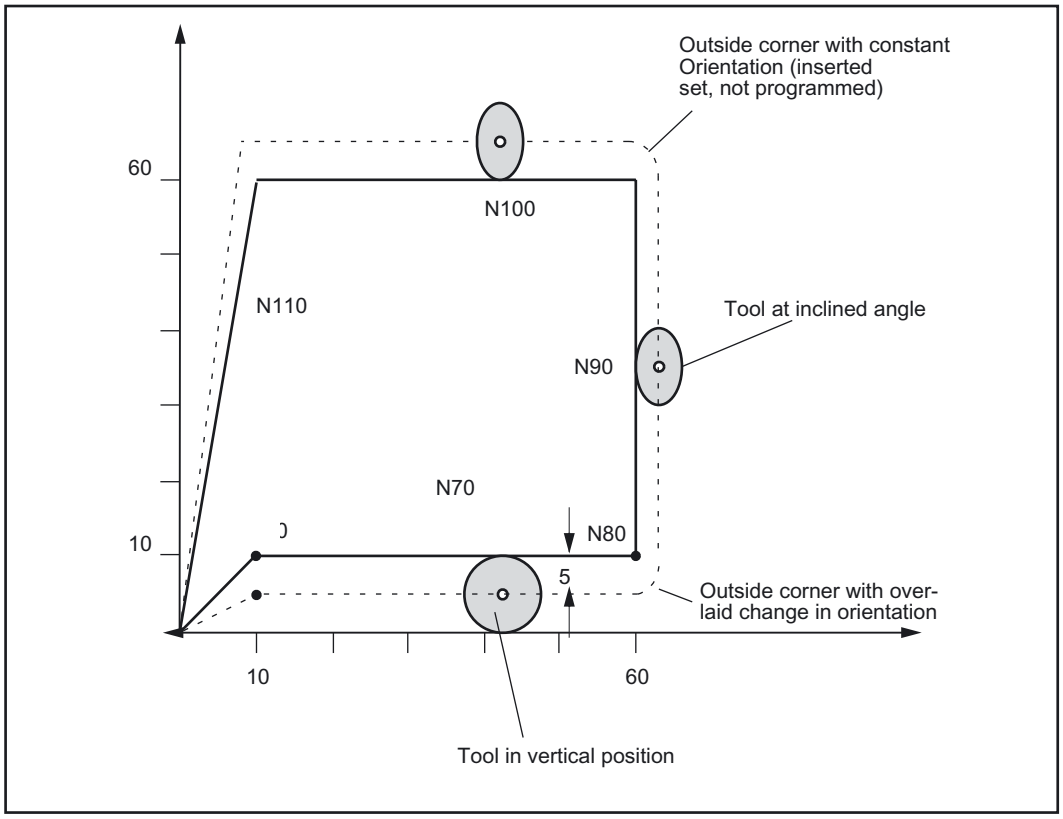


Figure 18-7 ORIC: Change in orientation and path movement in parallel

Example:

```
N10 A0 B0 X0 Y0 Z0 F5000          ; Radius=5
N20 T1 D1                         ; Transformation selection
N30 TRAORI(1)                     ; 3D TRC selection
N40 CUT3DC
N50 ORIC                          ; TRC selection
N60 G42 X10 Y10
N70 X60                           ; Change in orientation of external corners
                                  ; generated from N70 and N90
N80 A3=1 B3=0 C3=1
N90Y60
N100 X10
N110 G40 X0 Y0
N120 M30
```

The circular motion and change in orientation are executed in parallel in block `N80` (`ORIC` active).

Special case

Intermediate blocks without traversing and orientation motions are executed at the programmed positions, e.g. auxiliary functions.

Example:

```
N70 X60
N75 M20                          ; Auxiliary function call
N80 A3=1 B3=0 C3=1               ; Change in orientation of external corners
N90 Y60                          ; generated from N70 and N90
```

Blocks N75 and N80 are executed after N70. The circle block is then executed with the current orientation.

ORID

If ORID is active, then all blocks between the two traversing blocks are executed at the end of the first traversing block. The circle block with constant orientation is executed immediately before the second traversing block.

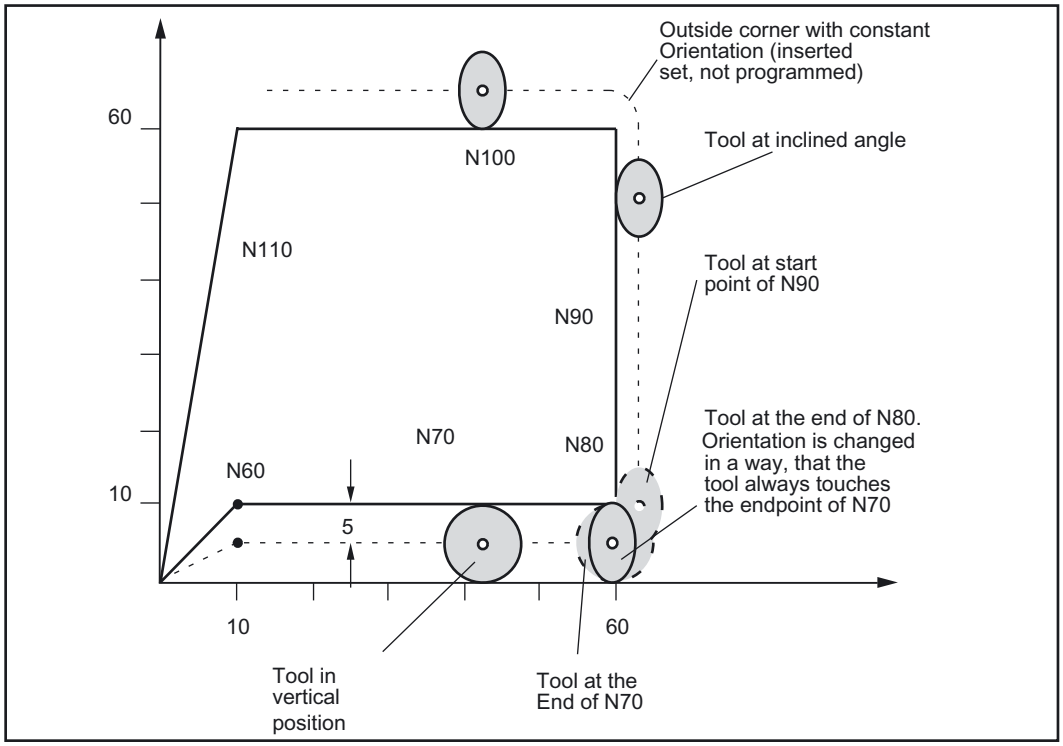


Figure 18-8 ORID: Change in orientation and path movement consecutively

Example:

```
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                          ; Radius=5
N30 TRAORI(1)                      ; Transformation selection
N40 CUT3DC                         ; 3D TRC selection
N50 ORID
N60 G42 X10 Y10                    ; TRC selection
N70 X60
N80 A3=1 B3=0 C3=1                 ; Change in orientation of external corners
N90 Y60                            ; generated from N70 and N90
N100 X10
N110 G40 X0 Y0
N120 M30
```

Note

The command `DISC` is not evaluated.

G451

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 no such intersection is available, a circle is inserted.

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.

No intersection procedure

The intersection procedure 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.

Blocks without traversing information

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.

18.2.3 Behavior at inside corners

Collision monitoring

With the 3D compensation function, only adjacent traversing blocks are taken into account in the calculation of intersections.

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.

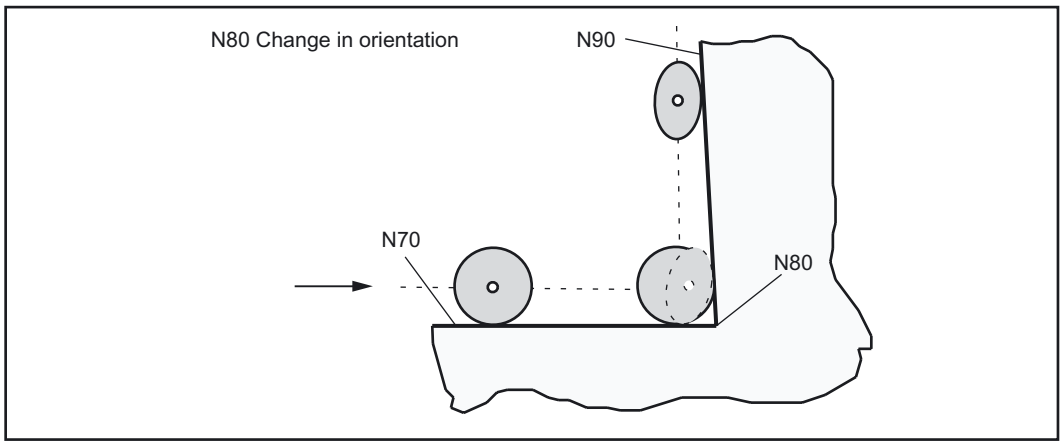


Figure 18-9 The contact points of the tool must not cross the limits of block N70 or N90 as a result of the change in orientation in block N80

Example:

```
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                          ; Radius=5
N30 TRAORI(1)                      ; Transformation selection
N40 CUT3DC                         ; 3D TRC selection
N50 ORID
N60 G42 X10 Y10                    ; TRC selection
N70 X60
N80 A3=1 B3=0 C3=1                 ; Change in orientation of inside corners
N90 X10                            ; generated from N70 and N90
N100 G40 X0 Y0
N120 M30
```

Without change in orientation

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 compensation.

With change in orientation

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.

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.

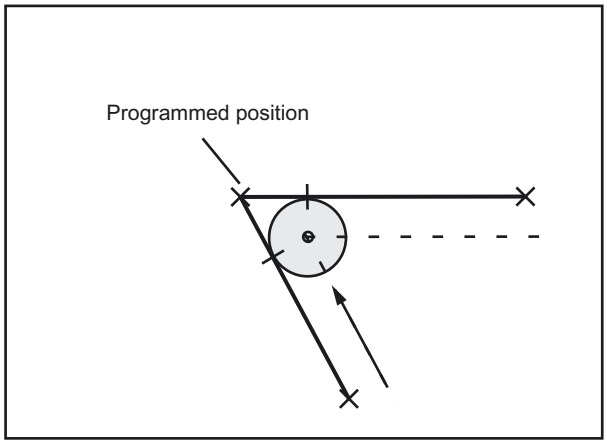


Figure 18-10 Path end position and change in orientation at inside corners

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.

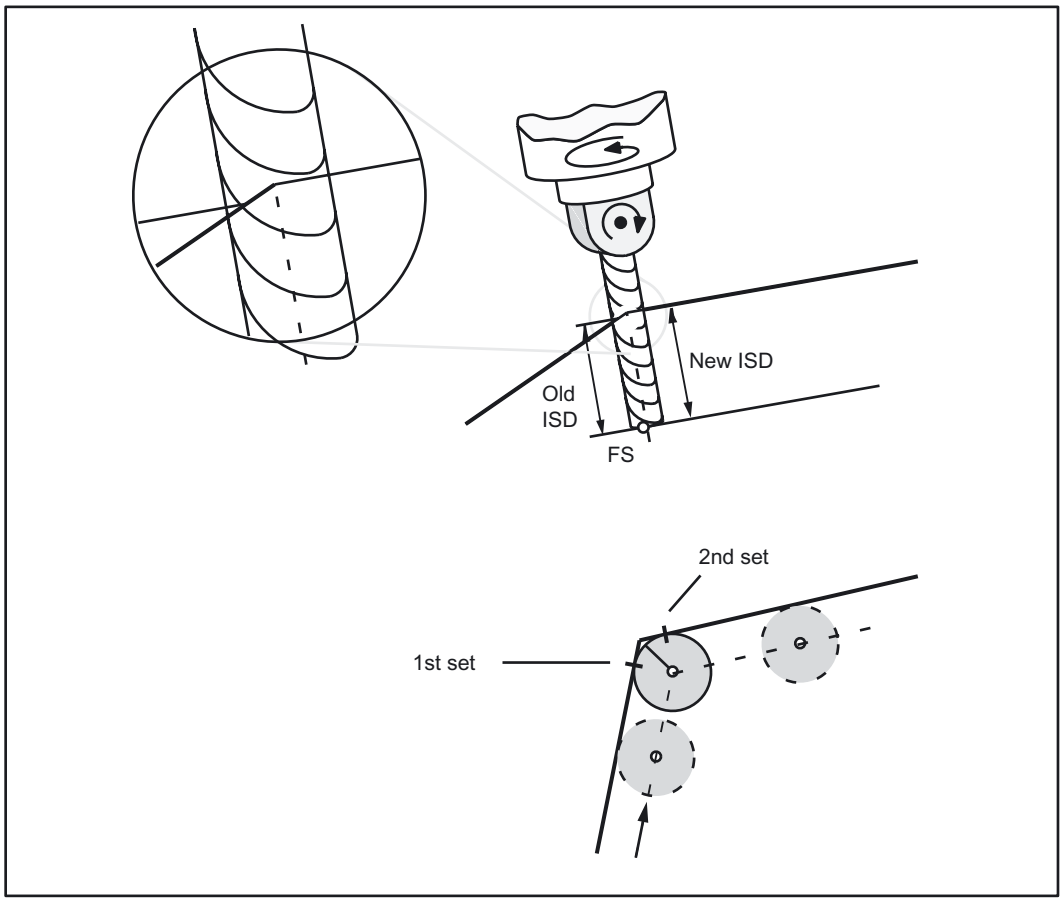


Figure 18-11 Change in insertion depth

Example of inside corners

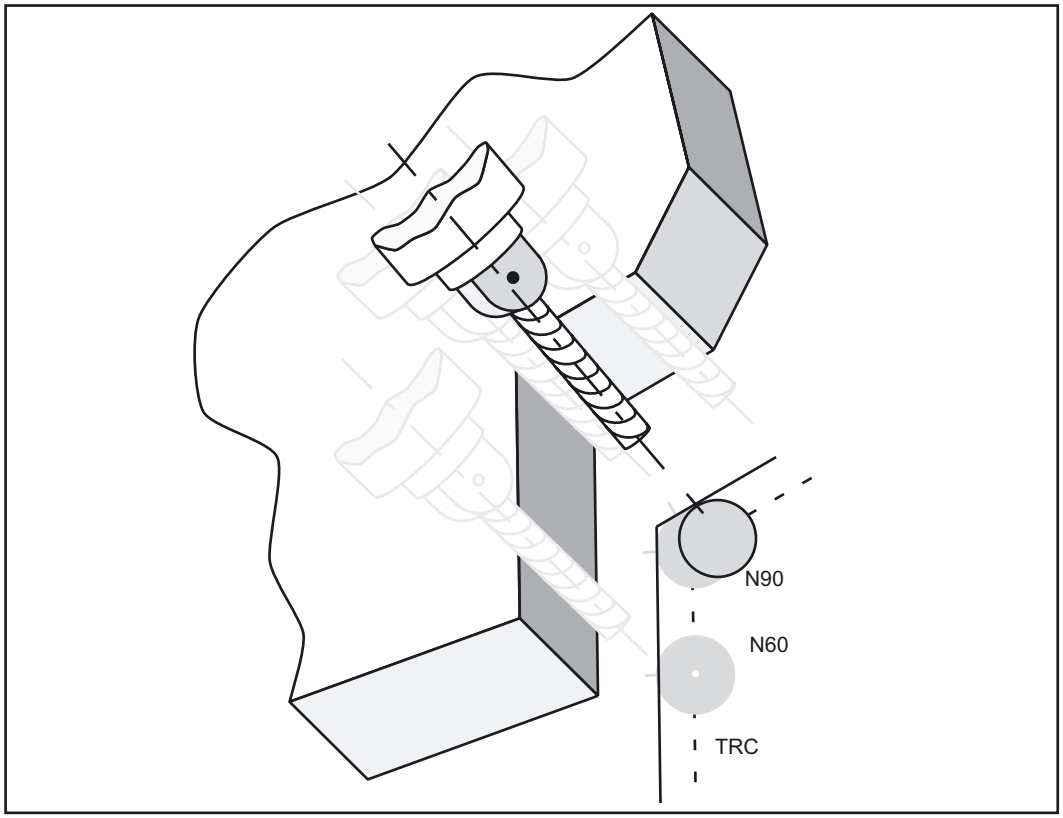


Figure 18-12 Change in orientation at an inside corner

Example:

```
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                          ; Radius=5
N30 TRAORI(1)                      ; Transformation selection
N40 CUT3DC                         ; 3D TRC selection
N50 ORID
N60 G42 X10 Y10 G451               ; TRC selection
N70 Y60
N80 A3=1 B3=0 C3=1                 ; Change in orientation of inside corners
N90 X60 Y90                        ; generated from N70 and N90
N100 G40 X... Y...
...
N190 CDOF
N200 M30
```

18.3 Face milling

The face milling function allows surfaces with any degree or form of curvature to be machined. In this case, the longitudinal axis of the tool and the surface normal vector are more or less parallel. In contrast, the longitudinal axis and the surface normal vector of the surface to be machined in a circumferential milling operation are at right angles to one another.

Information about the surfaces 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 relations in face milling are shown in the Figure below.

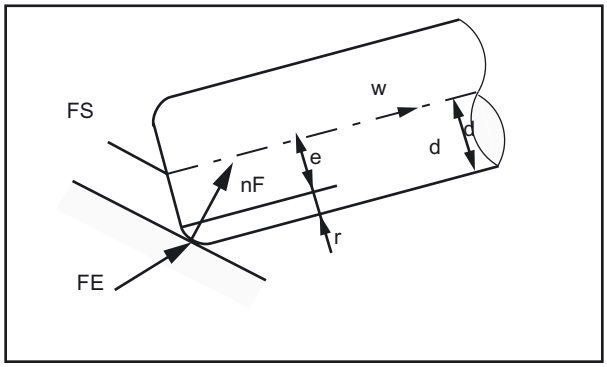


Figure 18-13 Face milling with a torus

18.3.1 Cutter shapes

The following table lists the possible tool shapes that may be used for face milling with their dimensions.

Table 18-1 Tool shapes for face milling

Cutter type	Tool No.	d	r	a
Ball end mill (cylindrical die sinker)	110	>0	X	X
Ball end mill (tapered die sinker)	111	>0	>d	X
End milling cutter without corner rounding	120, 130	>0	X	X
End mill with corner rounding (torus)	121, 131	>r	>0	X
Bevel cutter without corner rounding	155	>0	X	>0
Bevel cutter with corner rounding	156	>r	>0	>0

If a tool number other than any of those specified in the table above is used in the NC program, then the tool type is assumed to be a ball end mill (tool type 110). Tool parameters marked with an X in the tool table are not evaluated. A value other than zero is meaningless for the tool offset for face milling.

An alarm is output if tool data are programmed that violate the limits specified in the table above.

The shaft characteristics are not taken into account on any of the tool types. For this reason, the two tool types 120 (end mill) and 155 (bevel cutter), for example, have an identical machining action since only the section at the tool tip is taken into account. The only difference between these tools is that the tool shape is represented differently (dimensions).

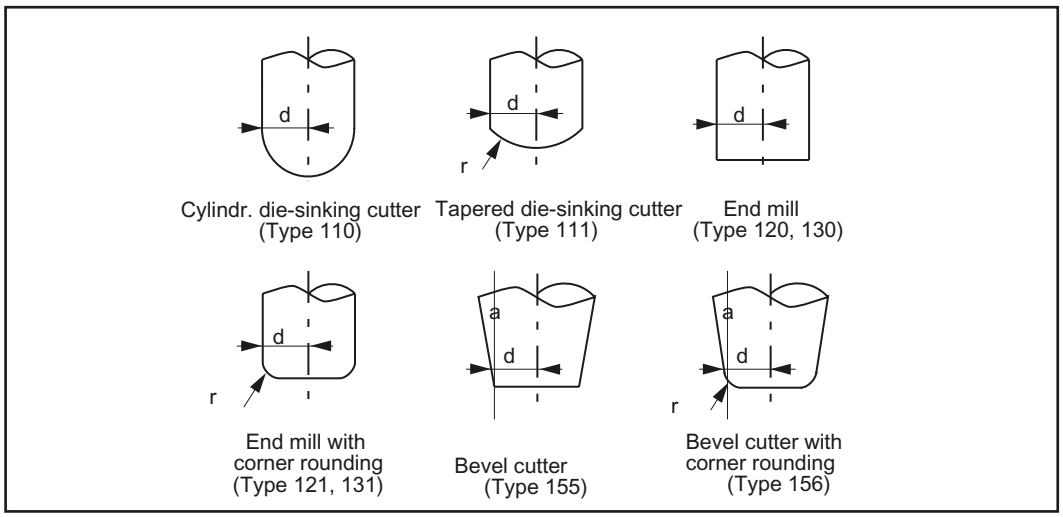


Figure 18-14 Tool types for face milling

The tool data are stored under the following tool parameter numbers:

Table 18-2 Tool parameter numbers for tool data

Tool data	Geometry	Wear
d	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

Note

The geometry and wear values of a tool data are added.

The reference point for tool length compensation (also referred to as tool tip or tool center point (TCP)) on all tool types is the point at which the longitudinal axis of the tool penetrates the surface.

A new tool with different dimensions may be programmed only when the tool compensation is activated for the first time (i.e. on transition from `G40` to `G41` or `G42`) or, if the compensation is already active, only when `G41` or `G42` are reprogrammed.

In contrast to circumferential milling, therefore, there are no variable tool dimensions in one block.

This restriction applies only to the tool shape (tool type, dimensions d, r and a).

A change in tool involving only a change in other tool data (e.g. tool length) is permitted provided that no other restrictions apply. An alarm is output if a tool is changed illegally.

18.3.2 Orientation

The options for programming the orientation have been extended for 3D face milling.

The tool offset for face milling cannot be calculated simply by specifying the path (e.g. a line in space). The surface to be machined must also be known. The control is supplied with the information it requires about this surface by 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 vector that 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 the start and end vectors are programmed, then interpolation takes place between both directions using the large-circle interpolation method. 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 has been programmed, it remains valid until another vector is programmed. In the basic setting, the surface normal vector is set to the same values as 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 as well as to those derived from the active plane. If ORIWKS is active, the surface normal vectors are adjusted when a new frame becomes active. 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 always applies when the programmed surface normal vector is not perpendicular to the path tangent. A new surface normal vector is then generated which is positioned in the plane extending from the path tangent to the programmed surface normal vector, but which is at right angles to the path tangent vector. This orthogonalization is necessary because the path tangent vector and surface normal vector for a real surface must always be perpendicular to one another. However, since the two values can be programmed independently, they may contain mutually contradictory information. 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 is smaller than the limit value programmed in machine data:

MD21084 $MC_CUTCOM_PLANE_PATH_LIMIT (minimum 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 lead angle is the angle between

the tool orientation and the surface normal vector. The side angle is the angle between the path tangent and the projection of the tool vector into the surface to be machined. Specification of the angle relative to the surface normal is merely an additional option for programming tool orientation at the block end. It does not imply that the lead and side angles reach their programmed values before the path end point is reached.

The final tool orientation is calculated from the path tangent, surface normal vector, lead angle and side angle at the block end. 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.

18.3.3 Compensation on path

Tool longitudinal axis parallel to surface normal

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 $\mathbf{n}_F$ and tool vector $\mathbf{w}$ 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 not, therefore, defined with this type of 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.

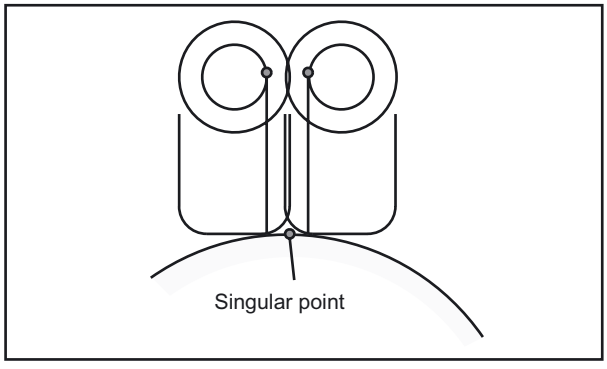


Figure 18-15 Change in the machining point on the tool surface close to a point in which surface normal vector and tool orientation are parallel

The problem is basically solved as follows: If the angle d between the surface normal vector $\mathbf{n}_F$ and tool orientation **w** is smaller than a limit value (machine data) δ_{min}, then the side angle y on tools with a flat end face (e.g. torus cutter or cylindrical mill) must be 0. This restriction does not apply to tool types with a spherical end face (e.g. ball end mill, die sinker) since angular changes close to the singular point do not lead to abrupt changes in the machining point on the surface of such tools. If δ now becomes 0, i.e. the sign of lead angle b changes, the machining point moves from its current position to the opposite side of the tool. This movement is executed in an inserted linear block.

The machining operation is aborted with an alarm if an attempt is made to machine within the illegal angular range for the side angle y (i.e. $\delta < \delta_{min}$ and $\gamma \neq 0$).

The insertion of linear blocks makes it necessary to split the original blocks at the singular points. The partial blocks created in this way are treated as if they were original, which means, for example, that a concave path containing a singularity is treated like an inside corner, i.e. there is no contour violation. Each new partial block must contain at least one tool contact point since this is always calculated on the basis of adjacent traversing blocks.

Singularities do not just occur at isolated points, but along whole 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, i.e. perpendicular to the osculating plane. A simple example is a circular arc in the x-y plane that is machined by a tool aligned in parallel to the z axis. On paths of this type, the tool offset is reduced to a tool length compensation, i.e. the tool is moved so that its tip FS is positioned on the programmed path.

On transition between singular and non-singular curves, linear blocks must be inserted in the same way as for isolated singular points so that the machining point on the tool can move from the tool tip FS to the periphery (on outside corners and convex surfaces) or the paths must be shortened to avoid contour violations (on inside corners and concave surfaces).

18.3.4 Corners for face milling

Two surfaces which do not merge tangentially form an edge. The paths defined on the surfaces make a corner. This corner is a point on the edge.

The corner type (inside or outside corner) is determined by the surface normal of the surfaces involved and by 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. Such transitions are not permissible 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. Failure to observe this rule results in an alarm. 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.

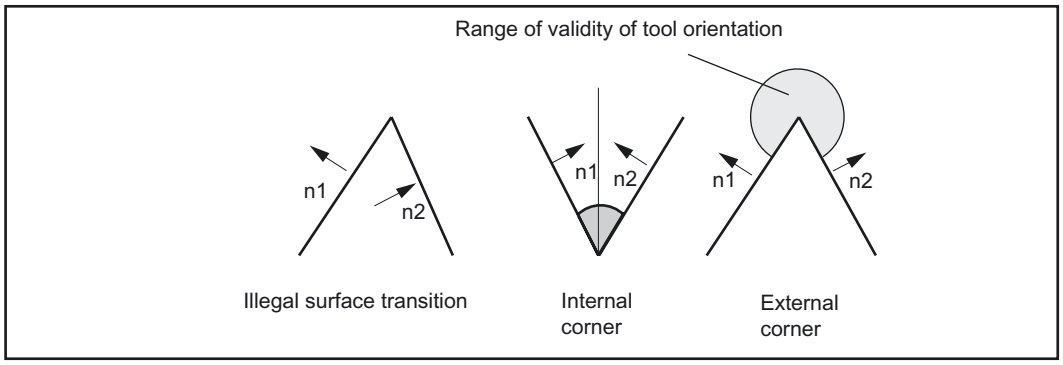


Figure 18-16 Corners for face milling

It is possible to insert blocks that contain no motion commands (e.g. auxiliary function outputs) and/or that include motions of axes not involved in the path between two blocks which contain a path definition. Changes in orientation can also be programmed in such intermediate blocks. The only exception applies to the activation and deactivation of the 3D tool radius compensation function, i.e. intermediate blocks with changes in orientation may not be inserted between the activation block and the first corrected block or between the last corrected block and the deactivation block. Other intermediate blocks are, however, permitted.

18.3.5 Behavior at outer corners

Outside corners are treated as if they were circles with a 0 radius. The tool radius compensation acts on these circles in the same way as on any other programmed path.

The circle plane extends from the final tangent of the first block to the start tangent of the second block.

The orientation can be changed during block transition.

A circle block is always inserted at an outside corner.

A change in orientation between two programmed blocks is executed either before the circle block or in parallel to it.

Programming

- `ORIC`: Change in orientation and path movement in parallel

  (**ORI**entation Change **C**ontinuously)

- `ORID`: Change in orientation and path movement consecutively

  (**ORI**entation Change **D**iscontinuously)

The `ORIC` and `ORID` program commands are used to determine whether changes in orientation programmed between two blocks are executed before the inserted circle block is processed or at the same time.

When the orientation needs to be changed at outside corners, the change can be implemented in parallel to interpolation or separately from the path motion. When `ORID` is programmed, the inserted blocks are executed first without a path motion (blocks with changes in orientation, auxiliary function outputs, etc.). The circle block is inserted immediately in front of the second of the two traversing blocks which form the corner.

ORIC

If ORIC is active and there are two or more blocks with changes in orientation (e.g. A2= B2= C2=) programmed between the traversing blocks, then the inserted circle block is distributed among these intermediate blocks according to the absolute changes in angle.

Change in orientation

The method by which the orientation is changed at an outer corner is determined by the program command that is active in the first traversing block of an outer corner.

If the tool orientation at an outside corner is not constant, then the change in orientation is implemented in exactly the same way as described in Subsection "Behavior at outer corners" for circumferential milling operations.

18.3.6 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.

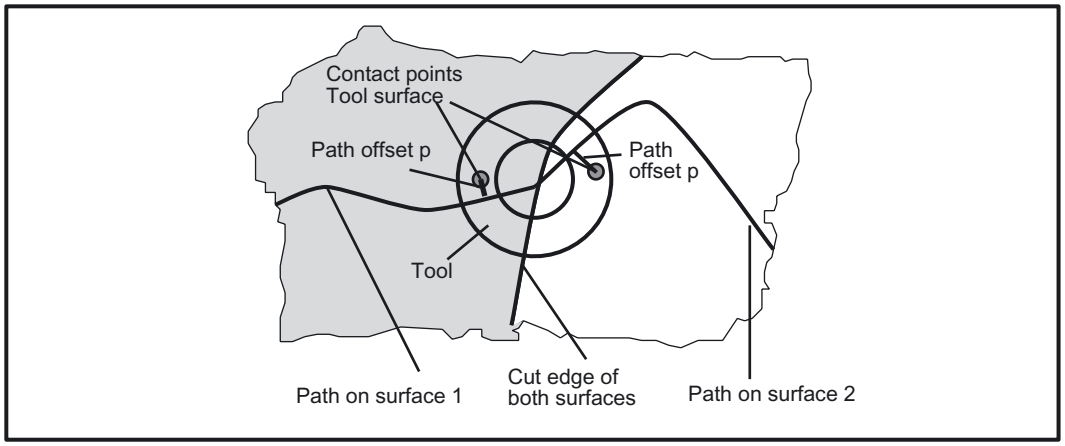


Figure 18-17 Inside corner with face milling (view in direction of longitudinal axis of tool)

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).

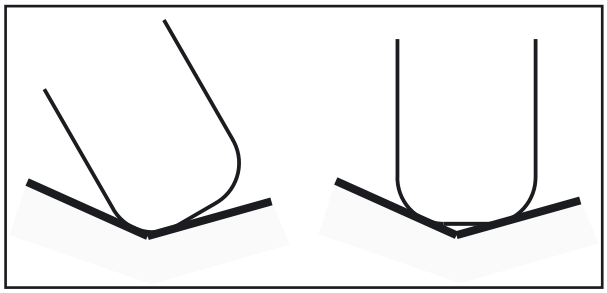


Figure 18-18 Machining at inside corners

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, 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.

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 Subsection "Behavior at inside corners"), 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).

18.3.7 Monitoring of path curvature

The path curvature is not monitored, i.e. the system does not usually detect any attempt to machine a concave surface that is curved to such a degree that the tool currently in use is not capable of performing the machining operation. 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. Except for such special cases, the user is responsible for ensuring that only tools that can machine along the entire contour without violating it are used.

18.4 Selection/deselection of 3D TRC

The following commands are used to select/deselect 3D tool radius compensation for circumferential milling or face milling:

- CUT3DC (circumferential milling)
- CUT3DFS (face milling)
- CUT3DFF (face milling)
- CUT3DF (face milling)

18.4.1 Selection of 3D TRC

CUT3DC

3D radius compensation for circumferential milling (only when 5-axis transformation is active).

CUT3DFS

3D tool offset for face milling with constant orientation. The tool orientation is determined by G17 - G19 and is not influenced by frames.

CUT3DFF

3D tool offset 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

This programming command selects the 3D tool offset for face milling with change in orientation (only when 5-axis transformation is active).

TRC selection

The program commands used to select 3D TRC are the same as those for 2D TRC. G41/G42 specify the compensation on the left or right in the direction of motion (the response on selection of G41 and G42 for 3D face milling is identical). Tool radius compensation is deactivated with G40. The approach response is always controlled with NORM. Activation must be in a linear block.

Example:

```
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                          ; Radius=5
N30 TRAORI(1)                      ; Transformation selection
N40 CUT3DC                         ; 3D TRC selection (insertion depth)
N50 G42 X10 Y10                    ; TRC selection
N60 X60
N70 ...
```

Intermediate blocks

Intermediate blocks are permitted when 3D TRC is active. The specifications for 2D TRC apply equally to 3D TRC.

18.4.2 Deselection of 3D TRC

Deselection

The 3D tool radius compensation function is deselected in a linear block `G0/G1` with geometry axes by means of `G40`.

Example:

```
N10 A0 B0 X0 Y0 Z0 F500
N20 T1 D1                          ; Radius=5
N30 TRAORI(1)                      ; Transformation selection
N40 CUT3DC                         ;3D TRC selection
N50 G42 X10 Y10                    ; TRC selection
N60 X60
N70 G40 X100 Y0 Z20                ; 3D TRC selection
N80 ...
```

Note

`D0` is programmed via active tool radius compensation, so there is no selection. If no geometry axis is programmed for the current plane in the block with the selection, no selection takes place.

18.5 Constraints

Availability of the "3D tool radius compensation" function

The function is an option and is available for

- SINUMERIK 840D with NCU 572/573

18.6 Examples

Example program for 3D circumferential milling:

```
                                        ; Definition of tool D1
$TC_DP1[1,1]=120                        ; Type (end mill)
$TC_DP3[1,1] =                          ; Length offset vector
$TC_DP6[1.1] =8                         ; Radius
N10 X0 Y0 Z0 T1 D1 F12000               ; Selection of the tool
N20 TRAORI(1)                           ; Activate transformation
N30 G42 ORIC ISD=10 CUT3DC G64 X30      ; Activate 3D circumferential milling, changes of
                                        ; orientation constant at outside corners, 10mm insertion
                                        ; depth.
N40 ORIWKS A30 B15                      ; Change of orientation at a corner
                                        ; through specification of axis positions.
N50 Y20 A3=1 C3=1                       ; Traversing block with change in orientation, specification
                                        ; orientation with direction vectors
N60 X50 Y30                             ; Traversing block with constant orientation
N70 Y50 A3=0.5 B3=1 C3=5                ; Traversing block with change in orientation
N80 M63                                 ; Block without traversing information
N90 X0 ISD=20                           ; Traversing block with change in insertion depth
N100 G40 Y0                             ; Deactivating the tool radius compensation
N110 M30
```

Example program for 3D face milling:

```
N10                                     ; Definition of tool D1
N20 $TC_DP1[1,1] = 121                  ; Tool type (torus cutter)
N30 $TC_DP3[1,1]=20                     ; Length compensation
N40 $TC_DP6[1,1]=5                      ; Radius
N60
N70
N50 $TC_DP7[1,1]=3                      ; Smoothing radius
N80 X0 Y0 Z0 A0 B0 C0 G17 T1 D1 F12000  ; Selection of the tool
N90 TRAORI(1)
N100 B4=-1 C4=1                         ; Plane definition
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 onset, relative programming
N270 X30 Y25 Z18                        ; the orientation remains during spline active
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 program
N320 G1 X0 A3=-0.123 B3=0.456 C3 =2.789 ; Path movement, orientation, non-constant plane
B4=-1 C4=5 B5=-1 C5=2
N330 X-20 G40                           ; Deactivation of the tool offset
N340 M30
```

18.7 Data lists

18.7.1 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

18.7.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after power-up and RESET / part program end
20120	TOOL_RESET_VALUE	Definition of tool for which tool length compensation is selected during power-up or on reset or parts program end as a function of 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	Maximum value for tool radius compensation
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	Max. no. of dummy blocks with no traversing movements
20270	CUTTING_EDGE_DEFAULT	Selected cutting edge after tool change
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
21080	CUTCOM_PARALLEL_ORI_LIMIT	Limit angle between path tangent and tool orientation with 3D tool radius compensation
21082	CUTCOM_PLANE_ORI_LIMIT	Minimum angle between surface normal and tool orientation with side angle not equal to 0

Number	Identifier: $MC_	Description
21084	CUTCOM_PLANE_PATH_LIMIT	Minimum angle between surface normal vector and path tangent vector, for 3D face milling
22550	TOOL_CHANGE_MODE	New tool offsets with M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

Path length evaluation (W6) 19

19.1 Brief description

With the "Path length evaluation" function, the NCK 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

Upon activation of the function, 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 current data values represent the sum of all measured values since the function's activation.

19.2 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 traverse path at high axis speeds [3]:
$AA_TRAVEL_TIME_HS	aaTravelTimeHS	Total travel time at high axis speeds [3]:
$AA_TRAVEL_COUNT_HS	aaTravelCountHS	Total travel count at high axis speeds [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 travel count with jerk:

1) System variables can be read from part programs and synchronized actions
2) MCS: machine coordinates system
3) Actual machine axis velocity >= 80% of the maximum parameterized axis velocity:
MD32000 MAX_AX_VELO (maximum axis velocity)

19.3 Parameterization

19.3.1 General activation

General activation of the function occurs via the NCK specified machine data:
MD18860 $MN_MM_MAINTENANCE_MON (activate recording of maintenance data)

19.3.2 Data groups

The data has been collected into data groups. Activation of data groups occurs via the axis-specific machine data:
MD33060 $MA_MAINTENANCE_DATA (configuration, recording 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 traverse path at high axis speeds: $AA_TRAVEL_DIST_HS / aaTravelDistHS • Total travel time at high axis speeds: $AA_TRAVEL_TIME_HS / aaTravelTimeHS • Total travel count at high axis speeds: $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

19.4 Examples

19.4.1 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.

Parameterization

Activation of the overall function:
MD18860 $MN_MM_MAINTENANCE_MON = TRUE

Group activation: "total traverse path, total travel time and travel count" for geometry axis AX1:
MD33060 $MA_MAINTENANCE_DATA[AX1] = 1

Part program

For values calculation based on the part program, the current value of the system variables at the beginning of the part program must be saved as calculational variables. The difference is acquired at the end of the part program.

Part program (extract)

```
Programming                              Comment
. . .
                                         ; Current values
R10 = $AA_TRAVEL_DIST[ AX1 ]             ; Total traverse path AX1
R11 = $AA_TRAVEL_TIME[ AX1 ]             ; Total travel time AX1
R12 = $AA_TRAVEL_NR[ AX1 ]               ; Travel count AX1
. . .                                    ; Motion of axes
                                         ; Difference calculation
R10 = R10 - $AA_TRAVEL_DIST[ AX1 ]       ; Traversal AX1 in part program
R11 = R11 - $AA_TRAVEL_TIME[ AX1 ]       ; Travel time AX1 in part program
R12 = R12 - $AA_TRAVEL_NR[ AX1 ]         ; Travel count AX1 in part program
```

19.5 Data lists

19.5.1 Machine data

19.5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
18860	MM_MAINTENANCE_MON	Activate recording of maintenance data

19.5.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
33060	MAINTENANCE_DATA	Configuration, recording maintenance data

NC/PLC interface signals (Z3) 20

20.1 Brief description

The section entitled "NC/PLC interface signals" includes a detailed description of NC/PLC interface signals relevant to function:

- PLC messages (DB2)
- NC (DB10)
- Mode group (DB11)
- OP (DB19)
- NCK channel (DB21-DB30)
- Axis/spindle (DB31-DB61)
- Loading/unloading a magazine (DB71)
- Spindle as change point (DB72)
- Tool turret (DB73)

20.2 3-Axis to 5-Axis Transformation (F2)

20.2.1 Signals from channel (DB21, ...)

DB21, ... DBX29.4	Activate PTP traversal
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Activate PTP traversal.
Signal state 0 or edge change 1 → 0	Activate CP traversal.
Signal irrelevant for ...	No handling transformations active.
Additional references	/FB3/ Function Manual, Special Functions; 3-Axis to 5-Axis Transformation (F2)

DB21, ... DBX33.6	Transformation active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Active transformation.
Signal state 0 or edge change 1 → 0	Transformation not (no longer) active.
Signal irrelevant for ...	No transformation used.
Additional references	/PG/ Programming Guide, Fundamentals

DB21, ... DBB232	Number of active G function of G function group 25 (tool orientation reference)
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	`ORIWKS`: The tool orientation is implemented in a workpiece coordinate system and is thus not dependent on the machine kinematics. This is the default setting.
Signal state 0 or edge change 1 → 0	`ORIMKS`: The tool orientation is defined in the machine coordinate system and is thus dependent on the machine kinematics. This is the default setting.

DB21, ... DBX317.6	**PTP traversal active**
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	PTP traversal active.
Signal state 0 or edge change 1 → 0	CP traversal active.
Signal irrelevant for ...	No handling transformations active.
Additional references	/FB3/ Function Manual, Special Functions, 3-Axis to 5-Axis Transformation (F2)

DB21, ... DBX318.2	**TOFF active**
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Activate online tool length offset.
Signal state 0 or edge change 1 → 0	Reset online tool length offset.
Signal irrelevant for ...	The "generic 5-axis transformation" option is not available and no handling transformations are active.
Additional references	/FB3/ Function Manual, Special Functions, 3-Axis to 5-Axis Transformation (F2)

DB21, ... DBX318.3	**TOFF motion active**
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Activate offset motion.
Signal state 0 or edge change 1 → 0	Deactivate offset motion.
Signal irrelevant for ...	The "generic 5-axis transformation" option is not available and no handling transformations are active.
Additional references	/FB3/ Function Manual, Special Functions, 3-Axis to 5-Axis Transformation (F2)

20.3 Gantry Axes (G1)

20.3.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX29.4	Start gantry synchronization
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Request from PLC user program to synchronize the leading axis with the assigned synchronized axes: MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition) (i.e. all gantry axes approach the reference position of the gantry grouping in the decoupled state). Synchronization of the gantry axes can be started only under the following conditions: • REF machine function must be active • DB11, ... DBX5.2 (REF active machine function) = 1 • DB31, ... DBX101.5 (gantry grouping is synchronized) = 0 • DB31, ... DBX101.4 (gantry synchronization ready to start) = 1 • No axis is being referenced in the appropriate NC channel: DB21, … DBX33.0 (referencing active) = 0
Signal state 0 or edge change 1 → 0	The PLC user program can then, for example, reset the interface signal to signal state "0" on completion of gantry synchronization (DB31, ... DBX101.5 = 1). If the IS is set continuously to "1", the gantry synchronization run would be started automatically as soon as the above conditions are fulfilled.
Signal irrelevant for ...	Gantry synchronized axis
Application example(s)	If the deviation between the position actual values and the reference position is greater than the gantry warning threshold after referencing of the gantry axes, automatic gantry synchronization is not started and IS "Gantry synchronization ready to start" is set to "1". Synchronization of the gantry axes can be started by the user or the PLC user program with IS "Start gantry synchronization".
Corresponding to	DB31, ... DBX101.5 (gantry grouping is synchronized) DB31, ... DBX101.4 (gantry synchronization ready to start) DB11, ... DBX5.2 (REF active machine function) DB21, ... DBX33.0 (referencing active)

DB31, ... DBX29.5	Automatic synchronization
Edge evaluation: No	Signal(s) updated:
Signal state 1 or edge change 0 → 1	No automatic synchronization.
Signal state 0 or edge change 1 → 0	The automatic synchronization process is active.
Signal irrelevant for ...	Gantry synchronized axis
Application example(s)	The automatic synchronization process can be disabled by sending a VDI signal to the axial PLC → NC interface of the master axis. This always makes sense when the axes are not activated by default. In this case, the synchronization process should also be started explicitly.

20.3.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX101.2	**Gantry trip limit exceeded**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The difference between the position actual values of the leading and synchronized axes has exceeded the maximum permissible limit value. The axes in the gantry grouping are shut down internally in the control. Alarm 10653 "Error limit exceeded" is also output. The monitored limit value is derived from the following machine data: • MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit) ... if gantry grouping is synchronized. • MD37130 $MA_GANTRY_POS_TOL_REF (gantry trip limit for referencing) ... if gantry grouping is not yet synchronized. **Note:** IS "Gantry trip limit exceeded" is output to the PLC via the PLC interface of the synchronized axis.
Signal state 0 or edge change 1 → 0	The difference between the position actual values of the leading and synchronized axes is still within the permissible tolerance range.
Signal irrelevant for ...	Gantry leading axis
Corresponding to	MD37120 $MA_GANTRY_POS_TOL_ERROR (gantry trip limit) MD37130 $MA_GANTRY_POS_TOL_REF (gantry trip limit for referencing) DB31, ... DBX101.5 (gantry grouping is synchronized)

DB31, ... DBX101.3	**Gantry warning limit exceeded**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The difference between the position actual values of the leading and synchronized axes has exceeded the limit value defined by the following machine data: MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit) The message "Warning limit exceeded" is also output. **Note:** IS "Gantry warning limit exceeded" is output to the PLC via the PLC interface of the synchronized axis.
Signal state 0 or edge change 1 → 0	The difference between the position actual values of the leading and synchronized axes is less than the limit value defined with machine data: MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)
Signal irrelevant for ...	Gantry leading axis
Application example(s)	When the gantry warning limit is exceeded, the necessary measures (e.g. program interruption at block end) can be initiated by the PLC user program.
Special cases, errors, ...	Warning limit monitoring is rendered inactive when the value 0 is input in the following machine data: MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)
Corresponding to	MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit)

DB31, ... DBX101.4	**Gantry synchronization ready to start**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	After gantry axis referencing, the monitoring function has detected that the position actual value deviation between the leading and synchronized axes is greater than the gantry warning limit: MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit) It is therefore not possible to start the **automatic** synchronization compensatory motion of the gantry axes internally in the control. The compensatory motion must be started by the user or the PLC user program (IS "Start gantry synchronization"). The signal is processed for the gantry leading axis only.
Signal state 0 or edge change 1 → 0	After the synchronization compensatory motion has been started by the PLC user program (IS "Start gantry synchronization" = "1").
Signal irrelevant for ...	Gantry synchronized axis
Corresponding to	MD37110 $MA_GANTRY_POS_TOL_WARNING (gantry warning limit) DB31, ...DBX29.4 (gantry synchronization ready to start) DB31, ...DBX60.4 and 60.5 (referenced / synchronized 1/2)

DB31, ... DBX101.5	**Gantry grouping is synchronized**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The gantry axis grouping defined with the following machine data is synchronized: MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition) Any existing misalignment between the leading and synchronized axes (e.g. after start-up of the machine) is eliminated by gantry axis synchronization (see Section "Start-up of gantry axes"). The synchronization process is initiated either automatically once the gantry axes have been referenced or via the PLC user program (IS "Start gantry synchronization"). The compensation values for temperature and sag do not become effective internally in the control until the gantry grouping is synchronized. **Note:** IS "Gantry grouping is synchronized" is output to the PLC via the PLC interface of the leading axis.
Signal state 0 or edge change 1 → 0	The gantry axis grouping defined with the following machine data is not synchronized: MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition) which means that the positions of the leading and synchronized axes may not be ideally aligned (e.g. gantry misalignment). Workpiece machining with a non-synchronized gantry axis grouping will result in impaired machining accuracy or mechanical damage to the machine. The gantry grouping becomes desynchronized if: • The gantry axes were in "Follow-up" mode • The reference position of a gantry axis is no longer valid or the axis is referenced again (IS "Referenced/Synchronized") • The gantry grouping has been invalidated via the following machine data: MD37140 $MA_GANTRY_BREAK_UP (invalidate gantry axis grouping)
Signal irrelevant for ...	Gantry synchronized axis

DB31, ... DBX101.5	Gantry grouping is synchronized
Application example(s)	Machining should be enabled only if the gantry axes are already synchronized. This can be implemented in the PLC user program by combining NC Start with IS "Gantry grouping is synchronized".
Corresponding to	DB31, ... DBX29.4 (gantry synchronization ready to start) DB31, ... DBX60.4 and 60.5 (referenced/synchronized 1 / 2) MD37140 $MA_GANTRY_BREAK_UP (invalidate gantry axis grouping)

DB31, ... DBX101.6	Gantry leading axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is defined as the **leading axis** within a gantry axis grouping (see MD37100). **Note:** The following interface signals are evaluated or output to the PLC via the PLC interface of the gantry leading axis: DB31, ... DBX29.4 (gantry synchronization ready to start) DB31, ... DBX101.5 (gantry grouping is synchronized)
Signal state 0 or edge change 1 → 0	The axis is defined as the **synchronized axis** within a gantry axis grouping (see MD37100). It is not possible to traverse a synchronized axis directly by hand (in JOG mode) or to program it in a part program. **Note:** The following interface signals are output to the PLC via the PLC interface of the gantry synchronized axis: DB31, ...DBX101.3 (gantry warning limit exceeded) DB31, ...DBX101.2 (gantry trip limit exceeded) The NCK does not evaluate individual axial PLC interface signals for the synchronized axis.
Corresponding to	MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition) DB31, ...DBX101.7 (gantry axis)

DB31, ... DBX101.7	Gantry axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is defined as a gantry axis within a gantry axis grouping (see MD37100). The PLC user program can read IS "Gantry leading axis" to detect whether the axis has been declared as a leading or synchronized axis.
Signal state 0 or edge change 1 → 0	The axis is not defined as a gantry axis (see MD37100).
Corresponding to	MD37100 $MA_GANTRY_AXIS_TYPE (gantry axis definition) DB31, ... DBX101.6 (gantry leading axis)

20.4 Axis Couplings and ESR (M3)

20.4.1 Signals to axis (DB31, ...)

DB31, ... DBX26.4	Active following axis overlay
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	An additional traversing motion can be overlaid on the following axis. This signal is required for the flying synchronization of master and slave axes. Until the "enable following axis overlay" signal stays set at 1, the following axis selected with EGONSYN will travel to synchronization in the EG coupling. Modulo axes included in the EG coupling reduce their position values in the modulo, thereby ensuring that they approach the next possible synchronization.
Signal state 0 or edge change 1 → 0	The following axis cannot be overlaid and traversed. If the "enable following axis overlay" signal has not been set for the following axis, the axis will not travel to synchronization. Instead, the program is stopped at the EGONSYN block and the self-clearing alarm 16771 is issued until the "enable following axis overlay" signal is set to 1.

20.4.2 Signals from axis (DB31, ...)

DB31, ... DBX98.5	Velocity warning threshold
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	When the following axis velocity in the axis grouping of the electronic gear (set in machine data MD32000) reaches or exceeds the velocity % block entered in machine data MD37550, the signal is set to 1.
Signal state 0 or edge change 1 → 0	The following axis velocity in the axis grouping of the electronic gear is less than the threshold value described above.
Signal irrelevant ...	Without electronic gear.
Corresponding to	the following machine data: MD37550 $MA_EG_VEL_WARNING (velocity warning threshold value) MD32000 $MA_MAX_AX_VELO (maximum axis velocity)

DB31, … DBX98.6	Acceleration warning threshold
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	When the following axis acceleration in the axis grouping of the electronic gear (set in machine data MD32300) reaches or exceeds the acceleration % block entered in machine data MD37550, the signal is set to 1.
Signal state 0 or edge change 1 → 0	The following axis acceleration in the axis grouping of the electronic gear is less than the threshold value described above.
Signal irrelevant ...	Without electronic gear.
Corresponding to	the following machine data: MD37550 $MA_EG_VEL_WARNING (velocity warning threshold value) MD32300 $MA_MAX_AX_ACCEL (axis acceleration)

DB31, … DBX98.7	ESR reaction is triggered
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Status signal The VDI signal "ESR reaction triggered" is available to the PLC as a check-back signal. The signal is set if $AA_ESR_STAT > 0, i.e. if: • Generator mode, cancellation or retraction functions • Intermediate circuit low voltage detected or • Generator speed lower than minimum setting
Signal state 0 or edge change 1 → 0	ESR is not active.
Application example(s)	For safety reasons, EMERGENCY STOP interrupts the interpolation and all traversing movements, and also dissolves the electronic coupling by canceling the servo enables. In applications where the coupling and traversing movements must remain valid after EMERGENCY STOP, this EMERGENCY STOP must be delayed long enough by the PLC for the required NC or drive-end reactions to terminate. Writing in $A_DBB allows the PLC to extensively influence the execution of the ESR reactions, if appropriate access is also integrated into the synchronized actions. On the 840D, the PLC has a "locking influence" on the ESR response. On the 840D, it is possible to link the relevant synchronized actions to produce the desired logic.

DB31, ... DBX99.3	**Axis accelerated**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	When the following axis acceleration in the axis grouping of the electronic gear (set in machine data MD32300) reaches or exceeds the acceleration % block entered in machine data MD37560, the signal is set to 1.
Signal state 0 or edge change 1 → 0	The following axis acceleration in the axis grouping of the electronic gear is less than the operating value described above.
Signal irrelevant ...	Without electronic gear.
Corresponding to	the following machine data: MD37560 $MA_EG_ACC_TOL (threshold value for "Accelerate axis") MD32300 $MA_MAX_AX_ACCEL (axis acceleration)

20.5 Setpoint Exchange (S9)

20.5.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX24.5	Activate setpoint exchange
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state = 1	Request to axis to take over drive control.
Signal state = 0	Request to axis to relinquish drive control.

20.5.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX96.5	Status of setpoint exchange
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1	The axis has taken over control of the drive.
Signal state 0	The axis has relinquished control of the drive.
Signal relevant for ...	All axes involved in setpoint exchange.

20.6 Tangential Control (T3)

20.6.1 Special response to signals

The movement of the axis under tangential follow-up control to compensate for a tangent jump at a corner of the path (defined by the movements of the leading axis) can be stopped by the following signals (e.g. for test purposes):

- NC Stop and override = 0
- Removal of the axis-specific feed enable

The signals are described in:
References:
/LIS2/ Lists (Book 2)

20.7 Clearance Control (TE1)

20.7.1 Signals to channel (DB21, ...)

DB21, ... DBX1.4	CLC stop
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Clearance control is deactivated in the same way as the part program command CLC_GAIN=0.0.
Signal state 0 or edge change 1 → 0	Clearance control is enabled.

DB21, ... DBX1.5	CLC_Override
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The channel-specific override DB21, … DBB4 (feed rate override) also applies to the clearance control Override settings < 100% reduce corresponding to the velocity limitation for the overriding motion, as defined in the following machine data: MD62516 $MC_CLC_SENSOR_VELO_LIMIT (velocity of clearance control motion) Override settings > 100% apply the limitation from the machine data.
Signal state 0 or edge change 1 → 0	The maximum velocity of the control motion is not dependent on the override setting
Application example	The difference for the operator is particularly dependent on whether the sensor motion is stopped or not with a 0 override.
Corresponding to	channel-specific override settings: DB21, ... DBB4 (feed rate override) DB21, ... DBX6.7 (feed rate override active)

20.7.2 Signals from channel (DB21, ...)

DB21, ... DBX37.3	CLC is active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Clearance control is activated.
Signal state 0 or edge change 1 → 0	Clearance control is deactivated.

DB21, ... DBX37.4	CLC motion at lower motion limit
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The traversing movement of the clearance-controlled axes based on clearance control has been stopped at the upper movement limit set in MD62505 $MC_CLC_SENSOR_LOWER_LIMIT (Lower motion limit of clearance control) or programmed with CLC_LIM(..). **Note:** The following signal is set at the same time: DB21, … DBX37.5 (CLC-stopped upper limit) → see above signal "CLC motion has stopped".
Signal state 0 or edge change 1 → 0	Clearance control has left the lower limitation.

DB21, ... DBX37.5	CLC motion at upper motion limit
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The traversing movement of the clearance-controlled axes based on clearance control has been stopped at the upper movement limit set in MD62506 $MC_CLC_SENSOR_UPPER_LIMIT (Upper motion limit of clearance control) or programmed with CLC_LIM(..). **Note:** The following signal is set at the same time: DB21, … DBX37.4 (CLC-stopped lower limit) → see above signal "CLC motion has stopped".
Signal state 0 or edge change 1 → 0	Clearance control has left the upper limitation.

DB21, ... DBX37.4-5	**CLC motion has stopped**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The traversing movement of the clearance-controlled axes based on clearance control is at a standstill. The following conditions will set the signal: The standstill conditions set in the following machine data are met: MD62520 $MC_CLC_SENSOR_STOP_POS_TOL (positional tolerance for status message "Clearance control zero speed") MD62521 $MC_CLC_SENSOR_STOP_DWELL_TIME (wait time for status message "Clearance control zero speed") Programming of CLC_GAIN=0.0 PLC interface signal: DB21, … DBX1.4 (stop CLC motion) **Note:** The "CLC motion at standstill" signal is only set if bits 4 and 5 are set at the same time. If only one of the two bits is set → see below.
Signal state 0 or edge change 1 → 0	Clearance control generates traversing movements directly in the clearance-controlled axes. As long as the axes are traversing due to clearance control, the axial interface signals cannot be set: DB31, … DBX60.6/7 (position reached with exact stop coarse/fine)
Corresponding to	DB3x DBX30.6/7 "Position reached, exact stop coarse/fine"

20.8 Speed/Torque Coupling, Master-Slave (TE3)

20.8.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX24.5	**Activate torque compensatory controller**
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Torque compensatory controller is to be activated. The following conditions must be fulfilled for activation: DB31, ... DBX96.2 ("fine" difference reached)
Signal state 0 or edge change 1 → 0	Torque compensatory controller is to be deactivated.

DB31, ... DBX24.7	**Activate master-slave coupling**
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Torque compensatory controller is to be activated.
Signal state 0 or edge change 1 → 0	Master-slave coupling should be deactivated. The following conditions must be fulfilled for activation and deactivation: • Master and slave axis under position control (DB31, ... DBB61.7) • Master and slave axis are at standstill (DB31, ... DBB61.4) • the channels of the master and slave axes are in the RESET state (DB21, ... DBB35.7) If a condition is not fulfilled, the coupling will not be activated or deactivated. No alarm appears and the status of the coupling remains the same. If, at a later point, all conditions are fulfilled, the coupling will be activated or deactivated depending on the status of the signal. The signal at the slave axis of a coupling is relevant.

20.8.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX96.2	Differential speed "Fine"
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The differential speed is in the range defined by the following machine data: MD37272 $MA_MS_VELO_TOL_FINE
Signal state 0 or edge change 1 → 0	The differential speed has not reached the range defined in MD37272.

DB31, ... DBX96.3	Differential speed "Coarse"
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The differential speed is in the range defined by the following machine data: MD37270 $MA_MS_VELO_TOL_COARSE
Signal state 0 or edge change 1 → 0	The differential speed has not reached the range defined in MD37270.

DB31, ... DBX96.4	Status of the torque compensatory control
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Torque compensatory control is active.
Signal state 0 or edge change 1 → 0	Torque compensatory control is not active. The signal at the slave axis of a coupling is relevant.

DB31, ... DBX96.7	Status of the master-slave coupling
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Master-slave coupling is active.
Signal state 0 or edge change 1 → 0	Master-slave coupling is not active. The signal at the slave axis of a coupling is relevant.

20.9 Handling Transformation Package (TE4)

20.9.1 Signals from channel (DB21, ...)

DB21, ... DBX29.4	Activate PTP traversal
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Activate PTP traversal.
Signal state 0 or edge change 1 → 0	Activate CP traversal.
Signal irrelevant for ...	No handling transformations active.
Additional references	/FB3/ Function Manual, Special Functions; 3-Axis to 5-Axis Transformation (F2)

DB21, ... DBX33.6	Transformation active
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Active transformation.
Signal state 0 or edge change 1 → 0	Transformation not (no longer) active.
Signal irrelevant for ...	No transformation used.
Additional references	/FB3/ Function Manual, Special Functions; 3-Axis to 5-Axis Transformation (F2)

DB21, ... DBB232	Number of active G function of G function group 25 (tool orientation reference)
Edge evaluation:	Signal(s) updated:
Meaning 1	ORIWKS: The tool orientation is implemented in a workpiece coordinate system and is thus not dependent on the machine kinematics.
Meaning 2	ORIMKS: The tool orientation is defined in the machine coordinate system and is thus dependent on the machine kinematics. This is the default setting.
Meaning 3	ORIPATH: The tool orientation is implemented with the programmed lead and side angles relative to the path tangent and surface normal vector.

DB21, ... DBX317.6	**PTP traversal active**
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	PTP traversal active.
Signal state 0 or edge change 1 → 0	CP traversal active.
Signal irrelevant for ...	No handling transformations active.
Additional references	/FB3/ Function Manual, Special Functions; 3-Axis to 5-Axis Transformation (F2)

20.10 MCS Coupling (TE6)

20.10.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX24.2	Deactivate or disable coupling
Edge evaluation: No	Signal(s) updated:
Signal state 1	An active coupling is not deactivated until the relevant axes are stationary. If CC_COPON is programmed for this axis, **no** error message is generated.
Signal state 0	Coupling may be activated
Signal irrelevant for ...	
Application example(s)	Evaluated only on the CC_Slave axis.

DB31, ... DBX24.3	Switch on collision protection
Edge evaluation: Yes	Signal(s) updated:
Signal state 1	Collision protection ON
Signal state 0	Collision protection OFF
Signal irrelevant for ...	This signal is processed only if collision protection is not activated via machine data: MD65543 $MA_CC_PROTECT_OPTIONS
Application example(s)	Evaluated only on the PSlave axis.

20.10.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX66.0	Activate monitor
Edge evaluation: No	Signal(s) updated:
Signal state 1	Monitoring is active. This readout must be activated by the PSlave axis in the the following machine data: MD63543 $MD_CC_PROTECT_OPTIONS **Note**: Conflicts may occur in connection with customer-specific compile cycles.
Signal state 0	Monitor is not active.

DB31, … DBX97.0	**Axis is a slave axis**
Edge evaluation: No	Signal(s) updated:
Signal state 1	Axis is a CC_Slave axis. The associated CC_Master axis can be found in the machine data.
Signal state 0	Axis is not a CC_Slave axis

DB31, … DBX97.1	**Activate coupling**
Edge evaluation: No	Signal(s) updated:
Signal state 1	Coupling active
Signal state 0	Coupling not active
Signal irrelevant for ...	
Application example(s)	Displayed only for the CC_Slave axis.

DB31, … DBX97.2	**Activate mirroring**
Edge evaluation: No	Signal(s) updated:
Signal state 1	Mirroring active (1:-1)
Signal state 0	1:1 coupling active
Signal irrelevant for ...	Relevant only if coupling is active (DBX97.1 = 1)
Application example(s)	Displayed only for the CC_Slave axis.

DB31, … DBX97.3	**Offset after point of activation**
Edge evaluation: Yes	Signal(s) updated:
Signal state 1	New offset after point of activation. This bit is set to 1 if a particular event (SW/HW limit switch on CC_Slave axis) causes a change in the offset between CC_Master and CC_Slave stored when the coupling was activated.
Signal state 0	No new offset since activation.
Signal irrelevant for ...	The bit is not set in the RESET phase.
Application example(s)	Displayed only for the CC_Slave axis.

20.11 Retrace Support (TE7)

20.11.1 Signals to channel

DB21, ... DBX0.1	Reverse/Forward
Edge evaluation: Yes	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Activate reverse travel. The RESU main program CC_RESU.MPF is generated from the traversing blocks recorded in the RESU-internal block buffer in order to initiate travel back along the contour on the next NC START.
Edge change 1 → 0	Activate forward travel. The RESU main program CC_RESU.MPF is generated from the traversing blocks recorded in the RESU-internal block buffer in order to initiate travel forwards along the contour on the next NC START.
Signal state 0	Not relevant.
Signal irrelevant for ...	RESU technological function not loaded or not activated.

DB21, ... DBX0.2	Start retrace support
Edge evaluation: No	Signal(s) updated:
Signal state 1	Start retrace support: The original machining program is reselected and a block search is carried out to locate the program continuation point.
Signal state 0	Not relevant.
Signal irrelevant for ...	The NC is not in Retrace mode or RESU is not active.

20.11.2 Signals from channel

DB21, ... DBX32.1	Retrace mode active
Edge evaluation: No	Signal(s) updated:
Signal state 1	The "Retrace mode active" signal is active as long as the control is in Retrace mode. This is the case from initial activation of the "reverse/forward" signal until activation of the "start retrace support" signal.
Signal state 0	The machining program is executed.
Signal irrelevant for ...	RESU technological function not active.

DB21, ... DBX32.2	Retrace support active
Edge evaluation: No	Signal(s) updated:
Signal state 1	The "retrace support active" signal is set as soon as signal state 1 is detected for the "start retrace support" signal. The "retrace support active" signal is reset at the end of retrace support once the last action block has been completed.
Signal state 0	Retrace support not active
Signal irrelevant for ...	RESU technological function not active.

Appendix

A

A.1 List of abbreviations

A	
AC	Adaptive Control
ADI4	Analog Drive Interface for 4 Axes
ALM	Active Line Module
ARM	Rotating induction motor
AS	Automation System
ASCII	American Standard Code for Information Interchange American Standard Code for Information Interchange
ASUB	Asynchronous subprogram
AuxF	Auxiliary function

B	
BA	Mode
BAG	Mode group
BCD	Binary Coded Decimals: Decimal numbers encoded in binary code
BCS	Basic Coordinate System
BERO	Proximity limit switch
BI	Binector Input
BICO	Binector Connector
BIN	BINary files
BO	Binector Output
BOT	Boot file (SIMODRIVE 611D)
BP	Basic Program (PLC)

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle
CF card	Compact Flash Card
CI	Connector Input

CNC	Computerized Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication
CP	Communication Processor
CPA	Compiler Projecting dAta
CPU	Central Processing Unit
CR	Carriage Return
CRC	Cutter Radius Compensation
CTS	Clear To Send: Signal from serial data interfaces
CU	Control Unit
CUTCOM	CUTter radius COMpensation: Tool radius compensation

D	
DAU	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)
DIN	Deutsche Industrie Norm (German Industry Standard)
DIO	Data Input/Output: Data transfer display
DIR	DIRectory
DO	Drive Object
DPM	Dual-Port Memory
DPR	Dual-Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function (handwheel)
DRIVE-CliQ	Drive Component Link with IQ
DRY	DRY run: Dry run feedrate
DSB	Decoding Single Block
DSC	Dynamic Servo Control / Dynamic Stiffness Control
DW	Data word
DWORD	Double word (currently 32 bits)

E	
EMC	Electro-Magnetic Compatibility
EN	European Standard
ENC	Encoder: Actual value encoder
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory

EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System
ESD	Electrostatic Sensitive Devices
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)
FD	Feed Drive
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: memory which works without address specification and whose data are read in the order in which they were stored.
FIPO	Fine InterPOlator
FRAME	Coordinate transformation
FST	Feed STop
FW	Firmware

G	
GC	Global Control (Profibus: Broadcast telegram)
GEO	Geometrie, e.g. geometry axis
GIA	Gear Interpolation dAta
GND	Signal GrouND
GS	Gearbox stage
GSD	Device master file for desribing a Profibus slave
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data
GWPS	Grinding Wheel Peripheral Speed

H	
HEX	Abbreviation for hexadecimal number
HHU	Handheld unit
HMI	Human Machine Interface, SINUMERIK operator interface
HW	Hardware
HW Config	SIMATIC S7 tool for configuration and parameterization of hardware components within an S7 project
HW limit switch	Hardware limit switch

I	
I/R	Infeed/regenerative feedback unit of SIMODRIVE 611(D)
IBN	Commissioning
ICA	Interpolatory compensation
INC	Increment
IPO	Interpolator
IS	Interface signal

J	
JOG	JOGging: Setup mode

K	
K_p	Proportional gain
$K_Ü$	Transmission ratio
K_V	Gain factor of control loop

L	
LAN	Local Area Network
LEC	Leadscrew error compensation
LED	Light-Emitting Diode
LF	Line Feed
LR	Position controller
LSB	Least Significant Bit
LUD	Local User Data

M	
MAC	Media Access Control
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine control panel
MCS	Machine coordinate system
MD	Machine Data
MDA	Manual Data Automatic: Manual input
MLFB	Machine-readable product designation
MM	Motor module
MMC	Man-Machine Communication
MPF	Main Program File: NC part program (main program)

MPI	Multi-Point Interface
MSD	Main Spindle Drive
MSGW	Message word
MSTT	Machine Control Panel

N	
NC	Numerical Control
NCK	Numerical Control Kernel
NCU	Numerical Control Unit
NRK	Name of operating system of the NCK
NX	Numerical eXtension (axis extension module)

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OLP	Optical Link Plug: Fiber-optic bus connector
OP	Operator Panel: Operating equipment
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	Options

P	
PC	Personal Computer
PCMCIA	Personal Computer Memory Card International Association
PCU	PC Unit
PCW	Program control word
PG	Programming device
PID	Parameter identification: Part of a PIV
PII	Process image of inputs
PIO	Process image of outputs
PIV	Parameter identification: Value: Parameterizing part of a PPO
PLC	Programmable Logic Control: Adaptive controller
PMS	Position Measuring System
PNO	PROFIBUS User Organization
PO	Power On
POS	Positioning: e.g. POS axis = positioning axis = channel axis which traverses non-interpolatory channel axes to their programmed positions, i.e. independently of other channel axes
POSMO A	Positioning Motor Actuator
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 DC	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 message frame for Profibus DP transmission and "Variable speed drives" profile
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program test
PTP	Point-To-Point Point-to-point
PUD	Program global User Data: Global program variable
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
RP	R parameter, arithmetic parameter, predefined user variable
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion
RTCP	Real Time Control Protocol

S	
SBC	Safe Brake Control
SBL	Single Block
SD	Setting Data
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test Search run by program test
SGE	Safety-related input
SGA	Safety-related output
SH	Safe stop
SIM	Single Inline Module
SK	Softkey
SKP	SKiP: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper Motor
SMC	Cabinet-mounted sensor module

SME	Sensor Module Externally-mounted
SPF	SubProgram File: Subprogram
SRAM	Static RAM (non-volatile)
SRM	Synchronous Rotary Motor
SSI	Synchronous Serial Interface (interface type)
SSL	Block search
STW	Control word
SW	Software
SW limit switch	Software limit switch
SYF	SYstem Files System files
SYNACT	SYNchronized ACTion

T	
T	Tool
TB	Terminal Board (SINAMICS)
TC	Tool change
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
TLC	Tool Length Compensation
TM	Tool management
TM	Terminal Module (SINAMICS)
TNRC	Tool Nose Radius Compensation
TO	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
TRC	Tool Radius Compensation
TTL	Transistor-Transistor Logic (interface type)

U	
UPS	Uninterruptible power supply (UPS)
USB	Universal Serial Bus

V	
VDI	Internal communication interface between NCK 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

W	
WCS	Workpiece Coordinate System
WOP	Workshop-Oriented Programming
WPD	Workpiece Directory
WZ	Tool

X	
XML	Extensible Markup Language

Z	
ZO	Zero offset
ZOA	Zero Offset Active: Identifier for zero offsets
ZSW	Status word (of drive)

A.2 Feedback on the documentation

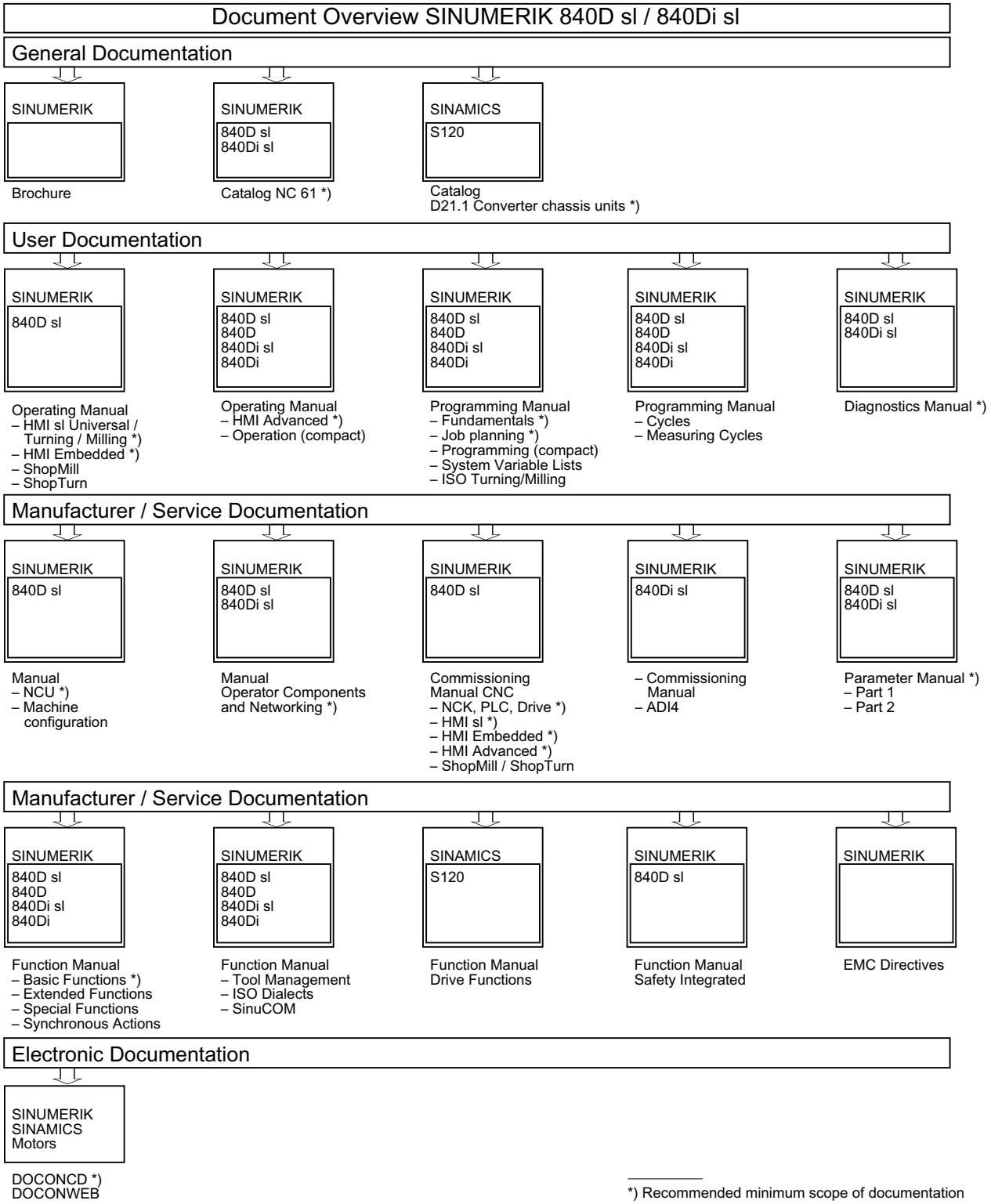
This document will be continuously improved with regard to its quality and ease of use. Please help us with this task by sending your comments and suggestions for improvement via e-mail or fax to:

E-mail:	mailto:docu.motioncontrol@siemens.com
Fax:	+49 9131 - 98 63315 Please use the fax form on the back of this page.

To	Sender
SIEMENS AG A&D MC MS1 P.O. Box 3180	Name:
	Address of your Company/Dept.
91050 ERLANGEN, GERMANY	Address:
	Zip code: City:
	Phone: /
Fax: +49 9131 - 98 63315 (Documentation)	Telefax: /

Suggestions and/or corrections

A.3 Overview


Document Overview SINUMERIK 840D sl / 840Di sl
General Documentation
SINUMERIK
Brochure
SINUMERIK
840D sl
840Di sl
Catalog NC 61 *)
SINAMICS
S120
Catalog
D21.1 Converter chassis units *)
User Documentation
SINUMERIK
840D sl
Operating Manual
– HMI sl Universal / Turning / Milling *)
– HMI Embedded *)
– ShopMill
– ShopTurn
SINUMERIK
840D sl
840D
840Di sl
840Di
Operating Manual
– HMI Advanced *)
– Operation (compact)
SINUMERIK
840D sl
840D
840Di sl
840Di
Programming Manual
– Fundamentals *)
– Job planning *)
– Programming (compact)
– System Variable Lists
– ISO Turning/Milling
SINUMERIK
840D sl
840D
840Di sl
840Di
Programming Manual
– Cycles
– Measuring Cycles
SINUMERIK
840D sl
840Di sl
Diagnostics Manual *)
Manufacturer / Service Documentation
SINUMERIK
840D sl
Manual
– NCU *)
– Machine configuration
SINUMERIK
840D sl
840Di sl
Manual
Operator Components and Networking *)
SINUMERIK
840D sl
Commissioning Manual CNC
– NCK, PLC, Drive *)
– HMI sl *)
– HMI Embedded *)
– HMI Advanced *)
– ShopMill / ShopTurn
SINUMERIK
840Di sl
– Commissioning Manual
– ADI4
SINUMERIK
840D sl
840Di sl
Parameter Manual *)
– Part 1
– Part 2
Manufacturer / Service Documentation
SINUMERIK
840D sl
840D
840Di sl
840Di
Function Manual
– Basic Functions *)
– Extended Functions
– Special Functions
– Synchronous Actions
SINUMERIK
840D sl
840D
840Di sl
840Di
Function Manual
– Tool Management
– ISO Dialects
– SinuCOM
SINAMICS
S120
Function Manual
Drive Functions
SINUMERIK
840D sl
Function Manual
Safety Integrated
SINUMERIK
EMC Directives
Electronic Documentation
SINUMERIK
SINAMICS
Motors
DOCONCD *)
DOCONWEB
*) Recommended minimum scope of documentation

Glossary

Absolute dimensions

A destination for an axis movement is defined by a dimension that refers to the origin of the currently active coordinate system. See → Chain measure

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 cancel 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 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.

Archive

Reading out of files and/or directories on an **external** memory device.

Asynchronous subroutine

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 control (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 feed rate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis identifier

Axis identifier

Axes are identifed using X, Y, and Z as defined in DIN 66217 for a right-handed, right-angled → coordinate system.

Rotary axes rotating around X, Y, and Z are identified using A, B, and C. Additional axes situated parallel to the specified axes can be designated using other letters.

Axis name

See → Axis identifier

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.

Base 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 between the two coordinate systems lies in the → axis identifiers.

Baud rate

Rate of data transfer (Bit/s).

Blank

Workpiece as it is before it is machined.

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 movement.

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.

Chip

"Block" is the term given to any files required for creating and processing programs.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feed rate, and the workpiece is thereby machined.

CNC

See → NC

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 memory

Data range in the control, in which the tool offset data are stored.

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.

Connecting cables

Connecting cables are pre-assembled or user-assembled 2-wire cables with a connector at each end. This connecting cable connects the → CPU to a → programming device or to other CPUs by means of a → multi-point interface (MPI).

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 → Memory-programmable control

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.

Cycles

Protected subroutines 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 modules contain data definitions for global user data. These data can be initialized directly when they are defined.

Data word

Two-byte data unit within a → data block.

Diagnose

1. Operating area of the control.
2. The control has both a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and lead values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (G70/G71), the controller is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental zero 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 zero offset

Zero-point offset specified by the → PLC.

Fast retraction from 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. After fast retraction, an interrupt routine can also be executed (SINUMERIK 840D).

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 (1-200%) 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: → zero offset, → rotation, → scaling, → mirroring.

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

Geometry axes are used to describe a 2- or 3-dimensional area in the workpiece coordinate system.

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 movements:

- Circular movement in one plane
- A linear movement perpendicular to this plane

High-level CNC language

The high-level language offers: → user-defined variables, → system variables, → macro techniques.

High-speed digital inputs/outputs

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). The digital CNC outputs can be used to trigger fast, program-controlled switching functions (SINUMERIK 840D).

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subroutines, 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

Also 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 control reads ahead can be set in system parameters.

Interpolator

Logic unit of the → NCK that defines intermediate values for the motions to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Interpolatory compensation is a tool that enables manufacturing-related leadscrew error and measuring system error compensations.

Interrupt routine

Interrupt routines are special → subroutines 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

With SINUMERIK 840D, the time required for the path of a block to be traversed can be programmed for the axis motion instead of the feed velocity (G93).

JOG

Control operating mode (setup mode): In JOG mode, the machine can be set up. 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 control. 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 control 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 → RAM 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 prefixed by ":" introductory block, containing all the parameters required to start execution of a -> part program.

Main program

The → part program designated by a number or an identifer in which additional main programs, subroutines, or → cycles can be called.

MDA

Control operating mode: Manual Data Automatic. In the MDA mode, individual program blocks or block sequences with no reference to a main program or subroutine 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 cancel 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 group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a BAG can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

Mode of operation

An operating concept on a SINUMERIK control. The following modes are defined: → Jog, → MDA, → Automatic.

NC

Numerical Control: Numerical control (NC) includes all components of machine tool control: → NCK, → PLC, HMI, → COM.

Note

A more correct term for SINUMERIK 840D controls would be: Computerized Numerical Control

NCK

Numerical Control Kernel: Component of NC that executes the → part programs and basically coordinates the motion operations for the machine tool.

NRK

Numeric robotic kernel (operating system of → NCK)

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). As a result, a uniform process is available within the control for all interpolations for SINUMERIK 840D.

OEM

The scope for implementing individual solutions (OEM applications) for the SINUMERIK 840D has been provided for machine manufacturers, who wish to create their own operator interface or integrate process-oriented functions in the control.

Operator Interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

Oriented tool retraction

`RETTOOL`: If machining is interrupted (e.g. when a tool breaks), a program command can be used to retract the tool in a user-specified orientation by a defined distance.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Work memory
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable control feature, which enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

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 feed affects → path axes. It represents the geometric sum of the feed rates 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 an auxiliary program for sending and receiving CNC user data (e.g. part programs, tool offsets, etc.) via a serial interface. The PCIN program can run in 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**ontrol: → Programmable logic controller. Component of → NC: Programmable controller for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D: 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).

Positioning axis

Axis that performs an auxiliary movement 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 subroutines of → part programs.

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 Control

Programmable logic controllers (PLC) are electronic controls, 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 control. The programmable logic controller 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

Character 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 rate of an axis. For example, rapid traverse is used 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 element.

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".

Safety Functions

The control is equipped with permanently active montoring 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.

Selecting

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.

Serial RS-232-C interface

For data input/output, the PCU 20 has one serial V.24 interface (RS232) while the PCU 50/70 has two V.24 interfaces. Machining programs and manufacturer and user data can be loaded and saved via these interfaces.

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 soft keys displayed is dynamically adapted to the operating situation. The freely assignable function keys (soft keys) 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.

SRT

Transformation ratio

Standard cycles

Standard cycles are provided for machining operations, which are frequently repeated:

- Cycles for drilling/milling applications
- 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.

Subroutine

Sequence of statements of a → part program that can be called repeatedly with different defining parameters. The subroutine is called from a main program. Every subroutine can be protected against unauthorized read-out and display. → Cycles are a form of subroutines.

Supply System

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.

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 times, counters, markers

System variables

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 reach 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 control 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.

Traversing 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 memory

All programs and data, such as part programs, subroutines, comments, tool offsets, and zero 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.

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

RAM is a work memory 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.

Zero offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable

   SINUMERIK 840D: A configurable number of settable zero offsets are available for each CNC axis. The offsets - which are selected by means of G functions - take effect alternately.

2. External

   In addition to all the offsets which define the position of the workpiece zero, an external zero offset can be overridden by means of the handwheel (DRF offset) or from the PLC.

3. Programmable

   Zero offsets can be programmed for all path and positioning axes using the `TRANS` statement.

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z